SIEMENS

SINUMERIK 840D sl/840D/840Di/810D

Basic Machine

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	Version
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Various NC/PLC Interface Signals and Functions A2
Axis Monitoring, Protection Zones A3
Continuous-Path Mode, Exact Stop, LookAhead B1
Acceleration B2
Diagnostic Tools D1
Travel to Fixed Stop F1
Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control G2
Auxiliary Function Output to PLC H2
Mode Group, Channel, Program Operation, Reset Response K1
Axis Types, Coordinate Systems, Frames K2
Emergency Stop N2
Transverse Axes P1
Power Line Basic PLC Program P3 Pl
PLC Basic Program Solution line P3 sl
Reference Point Approach R1
Spindles S1
Feeds V1
Tool Compensation W1
NC/PLC Interface Signals Z1

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Preface

SINUMERIK documentation

The SINUMERIK documentation is organized in 3 parts:

- General documentation
- User documentation
- Manufacturer/service documentation

Please contact your local Siemens office for more information about other SINUMERIK 840D sl/840D/840Di/810D publications and publications that apply to all SINUMERIK controls.

A list of documents, updated on a monthly basis and indicating the available languages, is available on the Internet at:
http://www.siemens.com/motioncontrol
Select "Support" → "Technical Documentation" → "Overview of Documents".

The Internet version of the DOConCD (DOConWEB) is available at:
http://www.automation.siemens.com/doconweb

Target audience

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

Standard version

This Description of Functions describes only 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.

Hotline

If you have any questions, please contact the following hotline:
A&D Technical Support
Phone: +49 (0) 180 / 5050 - 222
Fax: +49 (0) 180 / 5050 - 223
E-mail: mailto:adsupport@siemens.com
Internet: http://www.siemens.com/automation/support-request

If you have any comments, suggestions, or corrections regarding this documentation, please fax or e-mail them to:
Fax: +49 (0) 9131 / 98 - 63315
E-mail: mailto:motioncontrol.docu@siemens.com

Fax form: See the reply form at the end of the document.

Website address

http://www.siemens.com/motioncontrol

Objective

The Descriptions of Functions provide the information required for configuration and installation.

Target groups

The Descriptions of Functions are designed for:

- Design engineers
- PLC user-program programmers
- Startup engineers once the system has been configured and set up
- Maintenance personnel responsible for inspecting and interpreting status signals and alarms

Structure of the Function Manual

Structure of this Function Manual:

- Inside title page (page 3) with the title of this function manual, the SINUMERIK controls and the software and version for which this edition of the function manual applies, as well as the overview of the descriptions of functions.
- The description of each function is a separate book.
- Appendix with list of abbreviations and terms.

An overview of the functions described in this function manual is found on page three. The functions are listed in alphanumeric order of their abbreviations (e.g. A2, A3, etc.). This sequence also includes the Description of Functions books in the Function Manual.

A Description of Functions contains the following chapters:

- Brief description
- Detailed description
- Constraints
- Examples
- Data lists

Note

Detailed descriptions regarding data and alarms are provided for:

- Machine and setting data: Electronic only on DOConCD or DOConWEB
- NC/PLC interface signals: /FB1/ NC/PLC interface signals (Z1)
- Alarms: /DA/Diagnostics Guide

Technical information

The following notation is used in this documentation:

Signal/data	Notation	Example
NC/PLC interface signals	... NC/PLC interface signals: Signal data (signal name) ...	When the new gear step is engaged, the NC/PLC interface signals DB31, ... DBX16.0-16.2 (Actual gear step A to C) and DB31, ... DBX16.3 (Gear changed) are set by the PLC program.
Machine data	... Machine data: <Type><Number> <Complete Identifier> (<Meaning>) ...	The master spindle is the spindle stored in machine data: MD20090 $MC_SPIND_DEF_MASTER_SPIND (Initial setting for master spindle on channel).
Setting data	... Setting data: <Type><Number> <Complete Identifier> (<Meaning>) ...	Setting data: SD42800 $SC_SPIND_ASSIGN_TAB[0] (Spindle number converter) contains the logical master spindle.

Data types

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

Type	Meaning	Value range
INT	Signed integers	$\pm(2^{31} - 1)$
REAL	Figures with decimal point acc. to IEEE	$\pm(10^{-300} \ldots 10^{+300})$
BOOL	Boolean values: TRUE/FALSE	TRUE ≠ 0; FALSE = 0
CHAR	1 ASCII character corresponding to the code	0 … 255
STRING	Character string, number of characters in [...], maximum of 200 characters	Sequence of values with 0 ... 255
AXIS	Axis identifier	Axis identifier for all channel axes
FRAME	Geometrical parameters for translation, rotation, scaling, and mirroring	
Arrays can only be formed from similar elementary data types. Up to two-dimensional arrays are possible.		

Quantity framework

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.)

The actual possible number of components listed above for a specific product can be found in the following ordering documentation:
References:
/BU/ Ordering Documentation Catalog NC

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810

Various NC/PLC Interface Signals and Functions (A2)

Function Manual

Valid for

Control
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Function 1-1

2 Detailed Description 2-1

2.1 NC/PLC interface signals 2-1
2.1.1 General 2-1
2.1.2 Ready signals to PLC 2-3
2.1.3 Alarm signals to PLC 2-4
2.1.4 SINUMERIK 840Di-specific interface signals 2-4
2.1.5 Signals to/from panel front 2-5
2.1.6 Signals to channel 2-7
2.1.7 Signals to axis/spindle 2-7
2.1.8 Signals from axis/spindle 2-17
2.1.9 Signals to axis/spindle (digital drives) 2-18
2.1.10 Signals from axis/spindle (digital drives) 2-20

2.2 Functions 2-23
2.2.1 Screen settings 2-23
2.2.2 Settings for involute interpolation 2-24
2.2.3 Activate DEFAULT memory 2-27
2.2.4 Read/write PLC variable 2-27
2.2.5 Access protection via password and keyswitch 2-31
2.2.5.1 Access protection via password and keyswitch 2-31
2.2.5.2 Password 2-32
2.2.5.3 Keyswitch settings (DB10, DBX56.4 to 7) 2-34
2.2.5.4 Parameterizable protection levels 2-35

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Example 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 Memory specific machine data 5-1
5.1.2 NC-specific machine data 5-3
5.1.3 Channelspecific machine data 5-4
5.1.4 Axis/spindlespecific machine data 5-4
5.1.5 Drive-specific machine data 5-4

5.2 System variables 5-5

5.3 Signals 5-5
5.3.1 Signals to NC 5-5
5.3.2 Signals from NC 5-5
5.3.3 Signals to channel 5-6
5.3.4 Signals from channel 5-6

5.3.5 Signals to axis/spindle 5-6
5.3.6 Signals from axis/spindle 5-7
5.3.7 Signals to operator panel front 5-8
5.3.8 Signals from operator panel front 5-9

Index Index-1

Brief Description

1

1.1 Function

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)

Detailed Description 2

2.1 NC/PLC interface signals

2.1.1 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
- MMC (operator component)
- MCP (Machine Control Panel)

Data exchange is organized by the basic PLC program.

Cyclical 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 groupspecific signals
- Channelspecific signals
- Axis/spindlespecific 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 MMC
- NC status signals (alarm signals)

Channelspecific signals (DB21, etc.)

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

For detailed information about the following subject areas, please refer to:

- Description of the basic PLC program:
  /FB1/ Description of Functions, Basic Machine; Basic PLC Program (P3)
- Description of event-controlled signal exchange (auxiliary and G functions):
  /FB1/ Description of Functions, Basic Machine; Auxiliary Function Output to PLC (H2)
- Overview of all interface signals, function blocks and data blocks:
  /LIS/Lists

2.1.2 Ready signals to PLC

DB10, DBX104.7 (NC CPU Ready)

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

DB10,DBX108.1 (MMC CPU2 Ready)

MMC CPU2 is ready and registers itself cyclically to NC.

References:
/FB2/ Description of Functions, Expansion Functions; Several Control Panels on Multiple NCUs, Decentralized Systems (B3)

DB10, DBX108.2 (MMC CPU1 Ready, MMC to MPI)

The MMCCPU is ready and registers itself cyclically with the NC. MMC connection via MPI interface.

DB10, DBX108.3 (MMC CPU1 Ready, MMC to OPI)

The MMCCPU is ready and registers itself cyclically with the NC. MMC connection via OPI interface.

DB10, DBX108.6 (611D Ready)

Group signal: All available SIMODRIVE 611D drives ready.

Condition: All machine axes ready
DB31, ... DBX93.5 (Drive Ready) = 1

611D Ready is only output in conjunction with SIMODRIVE 611D drives.

DB10, DBX108.6 (NC Ready)

The NC is ready.

2.1.3 Alarm signals to PLC

DB10, DBX103.0 (MMC alarm pending)

The HMI component signals that at least one HMI alarm is pending.

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 channelspecific interface can be scanned to see which channels are involved and whether this will cause a processing stop.

DB21, ... DBX36.6 (channelspecific NCK alarm pending)

The NC sends this signal to the PLC to indicate that at least one NC alarm is pending for the affected channel. See also: DB21, ... DBX36.6 (NCK alarm with processing stop pending)

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.

2.1.4 SINUMERIK 840Di-specific interface signals

For a detailed description of the SINUMERIK 840Di-specific interface signals, please refer to:

References

/HBI/ SINUMERIK 840Di Manual

2.1.5 Signals to/from panel front

DB19, DBX0.0 (operator panel inhibit)

All inputs via operator components on the operator panel front are inhibited.

DB19, DBX0.1 (darken screen)

The operator panel screen is darkened or lightened.

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 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)

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 value 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, DBB12 (control of V24 interface) (HMI Embedded only)

Job interface to control RS-232C. The jobs refer to the user control files stored in DB19, DBB14 and DB19, DBB15 (see below).

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

Job byte to control file transfer via hard disk. The jobs refer to the user control file stored in DB19, DBB16 and DB19, DBB17.

DB19, DBB14 (control of V24 interface) (HMI Embedded only)

Description byte to specify the PLC index of the axis, channel or TO number for the standard control file. The standard control file is processed in accordance with the job stored in DB19, DBB12.

DB19, DBB15 (control of V24 interface) (HMI Embedded only)

Description byte to specify the line in the standard/user control file in which the file to be transferred is stored.

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 processed in accordance with the job stored in DB19, DBB13.

DB19, DBB17 (control of file transfer via hard disk) (HMI Advanced only)

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, DBB24 (control of V24 interface) (HMI Embedded only)

Status byte for current status of data transfer for "RS-232 ON", "RS-232 OFF", "RS-232 EXTERNAL", "RS-232 STOP", etc., or if an error occurred during data transfer.

DB19, DBB25 (control of V24 interface) (HMI Embedded only)

Output byte for RS-232 data transmission error values.

DB19, DBB26 (control of file transfer via hard disk) (HMI Advanced only)

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

DB19, DBB27 (control of file transfer via hard disk) (HMI Advanced only)

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

2.1.6 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.

2.1.7 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 required for the movement:
DB31, ... DBX61.0 = 1 (drive-test travel request).

The movement is executed as soon as the travel enable is sent:
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 standstill
If the machine axis is at standstill and NC/PLC interface signal:
DB31, ... DBX1.3 = 1
no traversing request (manual or automatic) is executed. The traversing request is maintained. If the axis disable is cancelled when a traversing request is pending:
DB31, ... DBX1.3 = 0
the movement is executed.

Axis disable when the machine axis is in motion
If the machine axis is in motion and NC/PLC interface signal:
DB31, to DBX1.3 = 1
the movement of the machine axis is decelerated to standstill following the currently active axis-specific brake characteristic or, if the axis is part of an interpolated path movement or coupling, it is decelerated on a path- or coupling-specific basis.

If the axis disable is cancelled while a traversing request continues to be pending:
DB31, ... DBX1.3 = 0
the movement is continued.

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 (servo enable)

DB31, ... DBX2.1 (servo enable)	DB31, ... DBX1.4 (follow-up mode)	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 are required for follow-up mode:
DB31, ... DBX2.1 = 0 (servo enable)
DB31, ... DBX1.4 = 1 (follow-up mode)

Checkback signal:
DB31, ... DBX61.3 = 1 (follow-up mode active)

Note

When the servo enable is set, if the part program is active, the last programmed position is approached internally in the NC (REPOSA: Approach along line on all axes). In all other cases, all subsequent movements 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 servo 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 (servo enable)
DB31, ... DBX1.4 = 0 (follow-up mode)

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

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

Notice

With the "hold" function, once the servo 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 "servo enable" set.
Clamping pushed the machine axis from the actual position Y_1 to the clamping position Y_k.

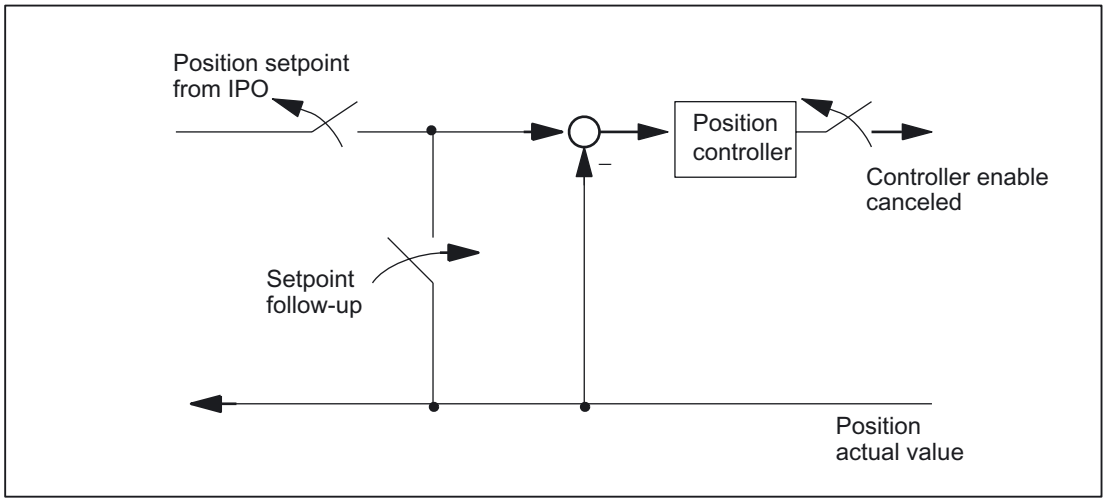


Fig. 2-1 Effect of servo enable and follow-up mode

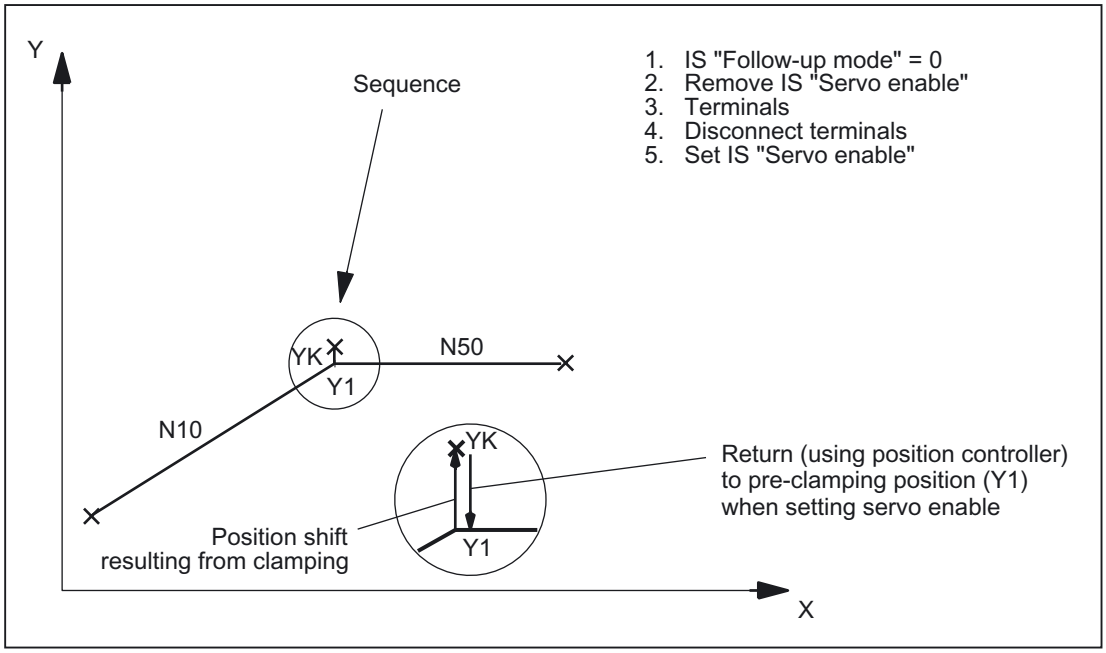


Fig. 2-2 Trajectory for clamping and "hold"

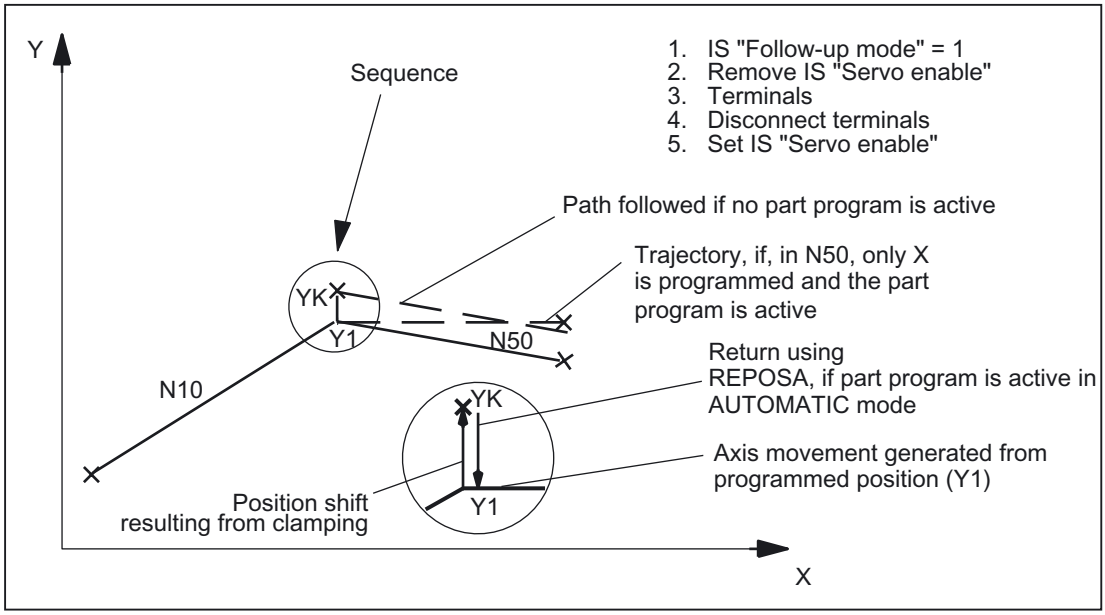


Fig. 2-3 Trajectory for clamping and "follow-up"

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 cancelled, homing is not required.

The following procedure is recommended:

1. Activate follow-up mode:
   DB31, ... DBX2.1 = 0 (servo enable)
   DB31, ... DBX1.4 = 1 (follow-up mode) (in the same or previous OB1 cycle)
   => Axis/spindle is in follow-up mode
2. Deactivate external servo 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 servo enable and cancel external speed setpoint

   => Axis/spindle stops
4. Cancel follow-up mode
   DB31, ... DBX2.1 = 1 (servo enable)
   DB31, ... DBX1.4 = 0 (follow-up mode)

   => NC synchronizes to current actual position. The next traversing movement begins at this position.

Note

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

Canceling follow-up mode

Once follow-up mode has been canceled, the machine axis does not have to be homed 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 (homed/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 movements due to the repositioning (REPOS) of 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)

2 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.

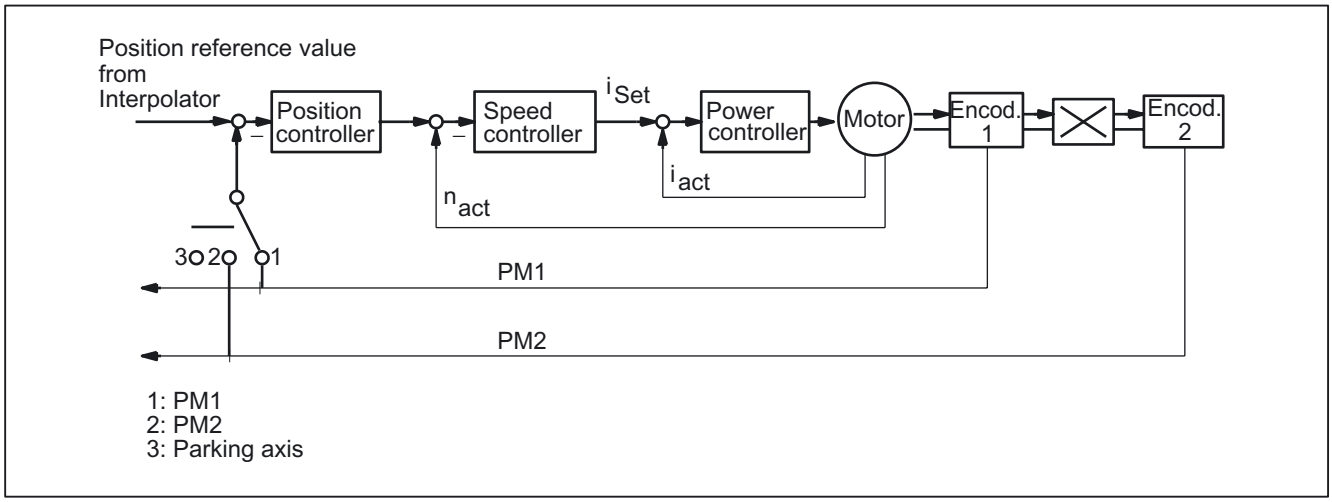


Fig. 2-4 Position measuring system 1 and 2

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

(DB31, ... DBX1.5) Position measuring system 1	(DB31, ... DBX1.6) Position measuring system 2	(DB31, ... DBX2.1) Servo enable	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 (servo enable)

Setting the servo enable closes the machine axis position control loop. The machine axis is in closed-loop position control.
DB31, ... DBX2.1 = 1

Canceling the servo enable opens the machine axis position control loop and, subject to a delay, speed control loop:
DB31, ... DBX2.1 = 0.

Activation modes

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

- PLC user program by means of the following NC/PLC interface signals:
  - DB31, ... DBX2.1 = 0 (servo enable)
  - DB31, ... DBX21.7 = 0 (pulse enable)
  - DB31, ... DBX93.5 = 0 (Drive Ready)
  - DB10, DBX56.1 = 1 (EMERGENCY STOP)

- NC-internal:
- Alarms that trigger cancellation of the servo enable on the machine axes. Alarms, which cancel the servo enable, are described in:
  References:
  /DA/ Diagnostics Guide

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

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

Canceling the servo enable when the machine axis is in motion:
If a machine axis is part of an interpolated path movement or coupling for which the servo enable is canceled, all axes involved are stopped with a fast stop (speed setpoint = 0) and an alarm is displayed:
Alarm: "21612 Servo enable reset during movement"

- Taking into account the parameterized temporal duration of the brake ramp for error states:
  MD36610 $MA_AX_EMERGENCY_STOP_TIME
  the machine axis is decelerated with a fast stop (speed setpoint = 0) and an alarm is displayed:
  Alarm: "21612 Servo enable reset during movement"

Note

The servo enable is canceled at the latest after the delay time has elapsed:
MD36610 $MA_AX_EMERGENCY_STOP_TIME

- The machine axis position control loop opens:
  DB31, ... DBX61.5 = 0 (position controller active)
  and the time for the assigned time delay for the servo enable is started:
  MD36620 $MA_SERVO_DISABLE_DELAY_TIME
- As soon as the actual speed reaches the zero-speed range, the drive servo enable is canceled:
  DB31, ... DBX61.6 = 0 (speed controller active)
- The position actual value of the machine axis continues to be acquired by the control.
- 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 description above for:
  DB31, ... DBX1.4 (follow-up mode).

Synchronize actual value (homing)
Once the servo enable has been set, the actual position of the machine axis does not need to be synchronized again (homing) if the maximum permissible limit frequency of the measuring system was not exceeded while the machine axis was not in position-control mode.

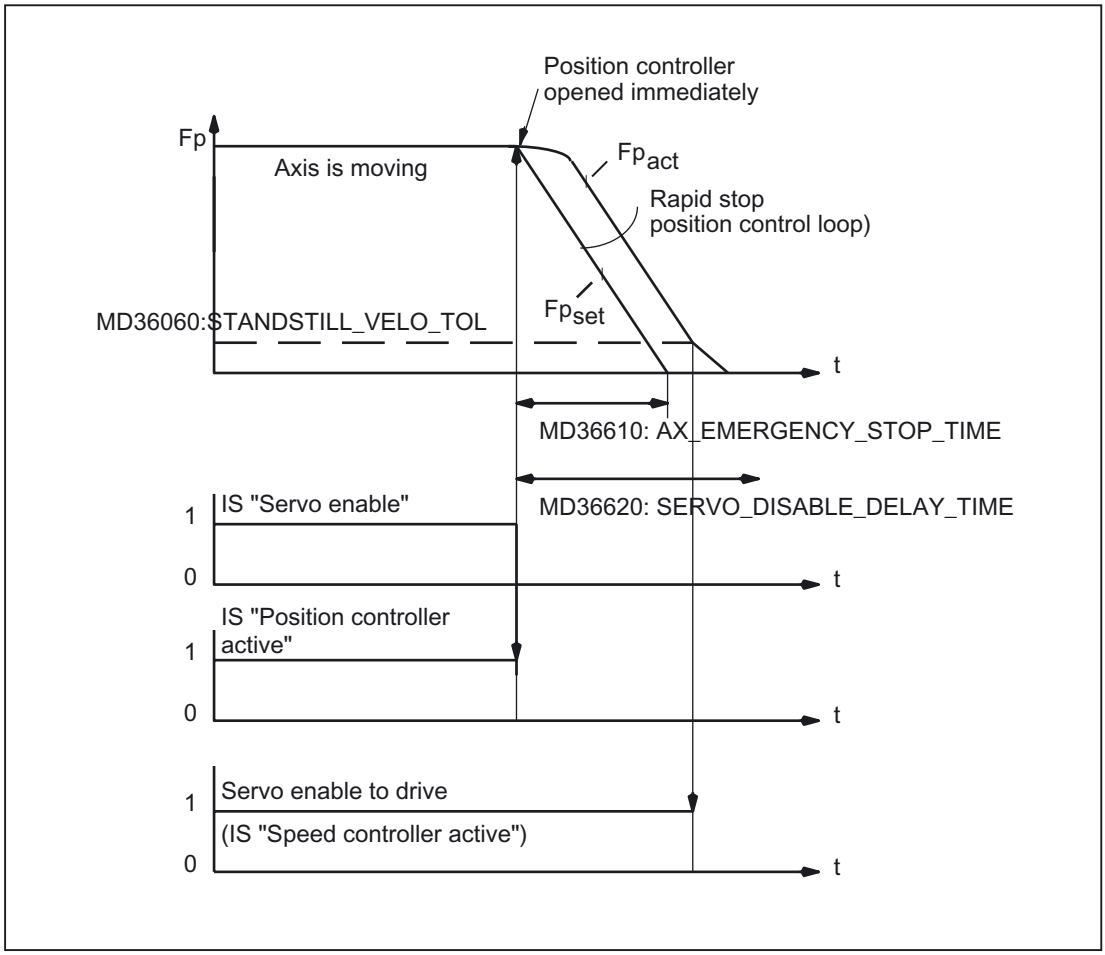


Fig. 2-5 Canceling the servo enable when the machine axis is in motion

DB31, ... DBX2.2 (Delete 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, please refer to
References:
/FB/ Description of Functions, Basic Machine; Spindles (S1)

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

The PLC user program sends a **binary** code request via the "controller parameter set selection" to activate the corresponding parameter set with that of the NC.

DBX9.2	DBX9.1	DBX9	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	6
1	1	1	6

The parameter set change must be enabled (not required for spindles):
MD35590 $MA_PARAMSET_CHANGE_ENABLE = 1 or 2

For detailed information about parameter-set changeover, please refer to:

References:
/FB/ Description of Functions, Basic Machine; Spindles (S1)
Chapter: Spindle Operating modes > Axis Mode;
Chapter: Programmable Gears > Gear Stages for Spindles and Gear Change

Parameter-set changeover when the machine axis is in motion
The response to a parameter-set changeover is determined by the consequential change in the closed-loop control circuit gain factor Kv:
MD32200 $MA_POSCTRL_GAIN (KV 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 movement.

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

  To ensure a relatively jerk-free changeover, the parameter set changeover does not take place until the axis is "stopped", i.e., the assigned standstill velocity has been reached or exceeded:
  DB31, ... DBX61.4 = 1 (axis/spindle has stopped)
  MD36060 $MA_STANDSTILL_VELO_TOL

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

For detailed information about auxiliary function output, please refer to:

References:
/FB/ Description of Functions, Basic Machine, Auxiliary Function Output (H2)

DB31, ... DBX9.3 (disable parameter-set default setting by NC)

Parameter-set changeover request will be ignored.

2.1.8 Signals from axis/spindle

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

The machine axis is to be traversed, e.g., by means of a part-program instruction or manually. The NC sends the travel request to the PLC. On the basis of the travel request, the PLC user program can then cancel the clamping of the axis, for example, and set the "servo enable".

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

The machine axis is in follow-up mode.

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

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

- No new setpoints are to be output **AND**
- The actual velocity of the machine axis is less than the assigned standstill velocity: MD36060 $MA_STANDSTILL_VELO_TOL (threshold velocity axis stopped)

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 (controller parameter set)

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 completes a distance longer than the parameterized traversing distance for lubrication.

MD33050 $MA_LUBRICATION_DIST (distance for lubrication by PLC)

2.1.9 Signals to axis/spindle (digital drives)

DB31, ... DBX20.1 (ramp-function-generator fast stop) (not on 810D)

A "fast stop" is requested for the drive. The drive is then stopped without a ramp function (speed setpoint 0). Servo enable is maintained.

DB31, ... DBX20.2 (torque limit 2) (not on 810D)

Request for torque limit 2. The limit value is specified in the drive parameters.

DB31, ... DBX20.3 (speed setpoint smoothing) (not on 810D)

Request to smooth the speed setpoint.

DB31, ... DBX21.0 / 21.1 / 21.2 (drive 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 checkback signal is sent via:

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

DB31, ... DBX21.3 / 21.4 (motor selection A, B) (not on 810D)

Selection of motor/operating mode.

DBX 21.4	DBX 21.3	Motor number	Operating mode
0	0	1	1
0	1	2	2
1	0	3	3 1)
1	1	4	4 1)

1) Can only be used on SIMODRIVE 611D Performance2 control module and SIMODRIVE 611U

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 made) (not on 810D)

The PLC user program sends this signal to the drive to indicate successful motor selection. For example, in the case of star/delta switchover on the SIMODRIVE 611D or 611U, a message or signal must be provided when the motor contactor has switched. The pulses are then enabled by the drive.

DB31, ... DBX21.6 (integrator disable n-controller) (not on 810D)

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).

Checkback signal via:
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
- Servo and pulse enable (terminal 63)
- Pulse enable (safe operational stop) (terminal 663)
- Stored hardware input

- Setpoint enable (terminal 64)
- "Status ready for traverse" (terminal 72/73)
  - No 611D drive alarm (DClink1 error)
  - DC link connected
  - Ramp-up completed

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

2.1.10 Signals from axis/spindle (digital drives)

DB31, ... DBX92.0 (setup mode active)

On the drive, "setup mode" is active.

DB31, ... DBX92.1 (ramp-function-generator fast stop active) (not on 810D)

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, ... DBX92.2 (torque limit 2 active) (not on 810D)

The drive signals back to the PLC that torque limit 2 is active for the axis/spindle. The torque limit value is defined with the drive parameters.

DB31, ... DBX92.3 (speed-setpoint smoothing active) (not on 810D)

The PLC user program requests speed-setpoint smoothing filter for the axis/spindle. The smoothing is activated in the drive module only under certain conditions.

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

The drive module sends this checkback 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 for the SIMODRIVE 611D.

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

The drive module (MSD) sends this checkback to the PLC to indicate which of the 4 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)

and

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

On the 611D with Performance2 controller module and 611U, combinations A = 0, B = 1 and A = 1, B = 1 may also be selected.

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) (not on 810D)

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 motor temperature has exceeded the warning threshold. If the motor temperature remains at this level, the drive will be stopped after a defined time (drive MD) and the pulse enable removed.

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

The heat-sink temperature has exceeded the warning threshold. The pulse enable will be removed for the drive module in question after 20 seconds.

DB31, ... DBX94.2 (ramp-up function completed)

This signal indicates that the actual speed value has reached the new setpoint taking into account the tolerance band set in the drive machine datum:
MD1426 $MD_SPEED_DES_EQ_ACT_TOL.

The acceleration procedure is thus completed. Any subsequent speed fluctuations due to load changes will not affect the interface signal.

DB31, ... DBX94.3 (|Md| < Mdx)

This signal indicates that the current torque $|M_d|$ is less than the assigned threshold torque M_{dx}:
MD1428 $MD_TORQUE_THRESHOLD_X

The threshold torque is entered as a percentage of the current speeddependent torque limitation.

DB31, ... DBX94.4 (|nact| < nmin)

This signal indicates that the actual speed value $|n_{actual}|$ is less than the set minimum speed n_{min}:
MD1418 $MD_SPEED_THRESHOLD_MIN

DB31, ... DBX94.5 (|nact| < nix)

This signal indicates that the actual speed value $|n_{actual}|$ is less than the set threshold speed $_x$:
MD1417 $MD_SPEED_THRESHOLD_X

DB31, ... DBX94.6 (nact = nsetp)

It is signaled to the PLC that the actual speed value n_{actual} has reached the new setpoint, taking into account the tolerance band set in the drive machine datum:
MD1426 $MD_SPEED_DES_EQ_ACT_TOL,
and is still within the tolerance band.

DB31, ... DBX94.7 (variable signaling function) (not 810D)

Using the variable signaling function, it is possible to monitor digitally for any axis any variable from SIMODRIVE 611 to check if it violates a certain threshold and to signal as an interface signal to the PLC.

The variable to be monitored is specified with 611D machine data.

DB31, ... DBX95.0 (DC link voltage < warning level)

A message is sent from the drive to the PLC that the DC link voltage U_{DClink} is smaller than the DC link undervoltage threshold:
MD1604 $MD_LINK_VOLTAGE_WARN_LIMIT

DB31, ... DBX95.7 (i^2t monitoring) (not 810D)

The variable signaling function can be used to protect the power section of the SIMODRIVE 611 universal and SIMODRIVE 611 digital drives against continuous overload. The PLC is signaled for each axis indicating whether the i^2t monitoring has been initiated.

The variable to be monitored is specified with 611D machine data.

2.2 Functions

2.2.1 Screen settings

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

Contrast

MD9000 $MM_LCD_CONTRAST (contrast)

For slimline operator panels 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)

On SINUMERIK 840D/840Di/810D, 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.

2.2.2 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.

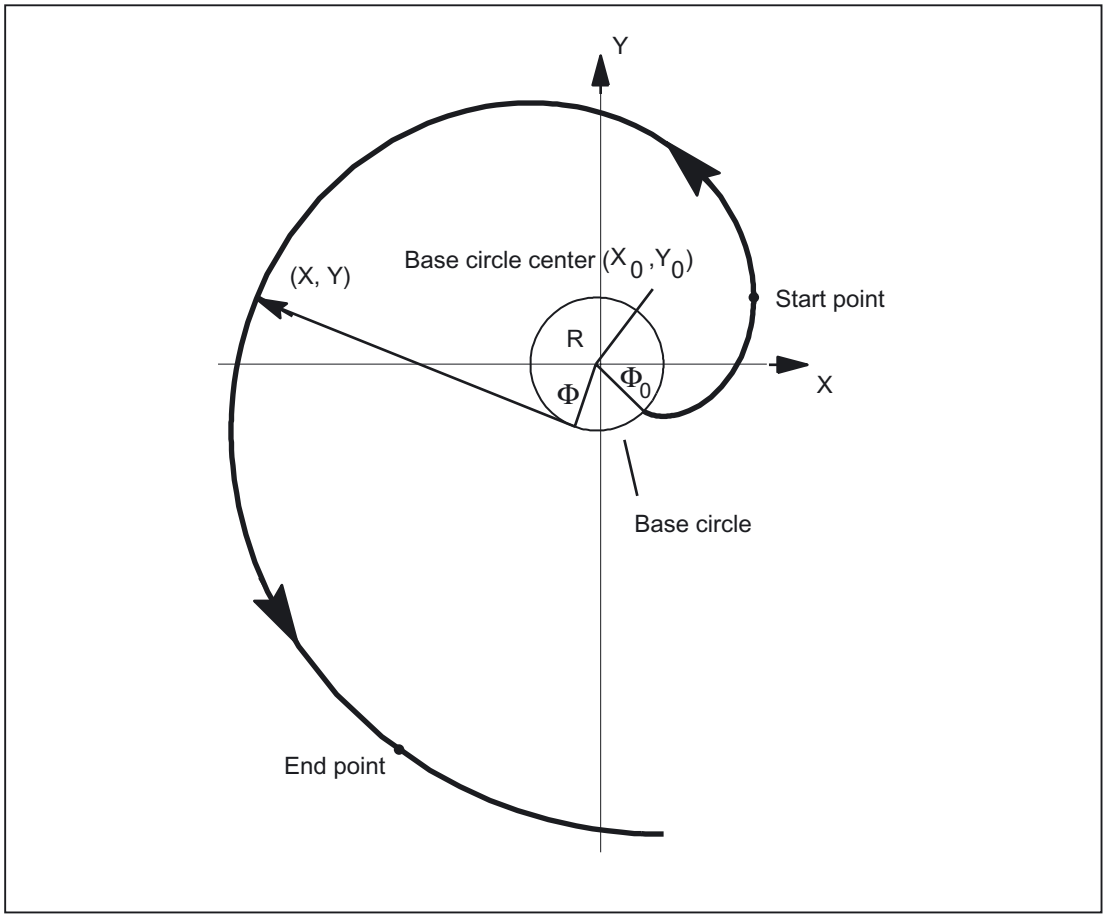


Fig. 2-6 Involute (unwound from base circle)

Programming

A general description of involute programming appears in:
References:
/PG/ Programming Guide, Basics

In addition to the programmed parameters, machine data are relevant in two instances of involute interpolation; these 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 endpoint is specified by:
MD21015 $MC_INVOLUTE_RADIUS_DELTA

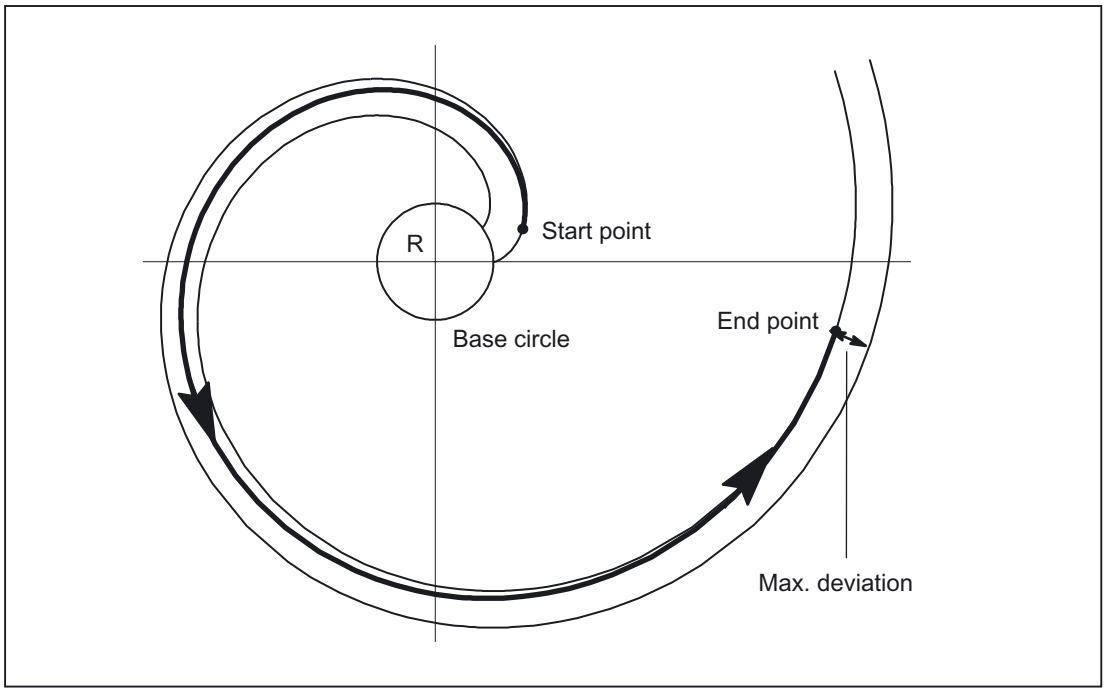


Fig. 2-7 MD21015 specifies the max. permissible deviation

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.

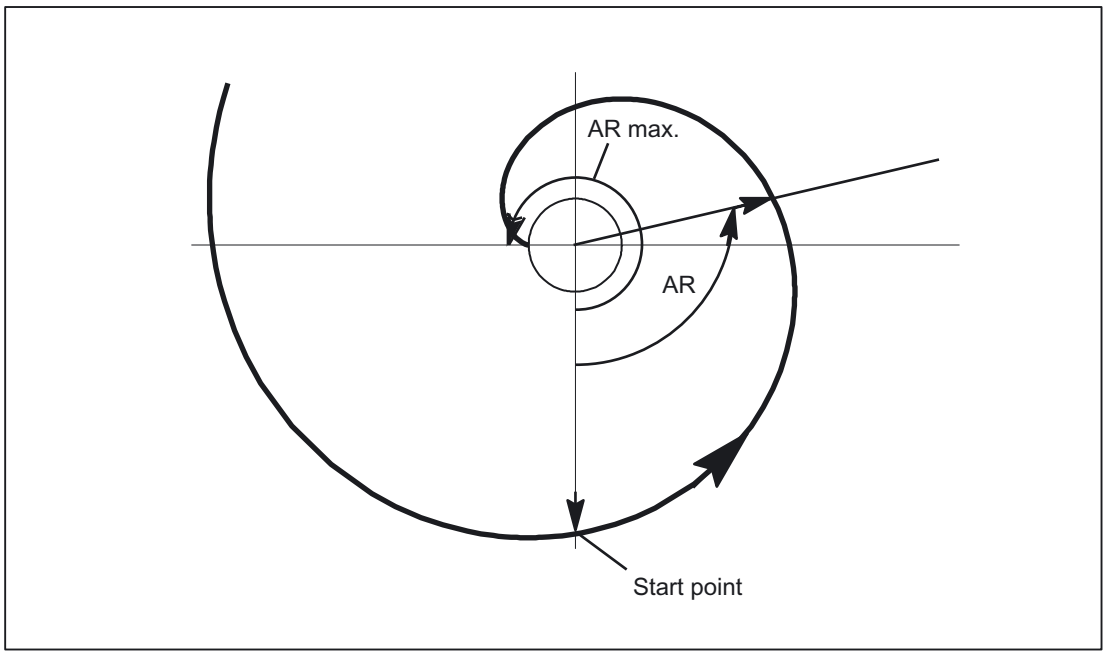


Fig. 2-8 Limited angle of rotation towards base circle

The alarm display can be suppressed with the following parameter assignment:
MD21016 $MC_INVOLUTE_AUTO_ANGLE_LIMIT = TRUE

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. The "Velocity limiting profiles" function must be activated in order for the velocity at this point to be sufficiently limited during active tool radius correction:
MD28530 $MC_MM_PATH_VELO_SEGMENTS > 1

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.

2.2.3 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 area for this is taken from the memory area assigned with machine datum:
MD18150 $MM_GUD_VALUES_MEM.

The setting for activation of the saved default values is made with machine datum:
MD11270 $MN_DEFAULT_VALUES_MEM_MASK.

References:
/FB/Description of Functions, Basic Machine/S7: "Memory Configuration"
/PGA/Programming Guide Advanced

2.2.4 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 (dualport 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 NCK uses '$ variables'.

Organization of memory area

The user's programming engineer (NCK 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

To allow the NC to access PLC variables (from a part program) quickly, $ variables are provided in the NCK. The PLC uses a function call (FC) to read and write $ variables. Data are transferred to and from the NCK immediately.

$ variables can be accessed (by the NCK) during preprocessing and in synchronized actions.

Data type information is determined by the $ variable data type, the position index is specified as an array index (in bytes).

The following $ variables are available:

$A_DBB	Data byte (8 bits)
$A_DBW	Data word (16 bits)
$A_DBD	Data double word (32 bits)
$A_DBR	Real data (32 bits)

Ranges of values

$A_DBB(n)	<= x <= 255
$A_DBW(n)	-32768 <= x <= 3276
$A_DBD(n)	-2147483648 <= x <= 2147483647

Access from PLC

The PLC uses function calls (FC) to access the memory. These FCs ensure that data are read and written in the DPR immediately, i.e., not just at the beginning of the PLC cycle. FCs receive data type information and the position offset as parameters.

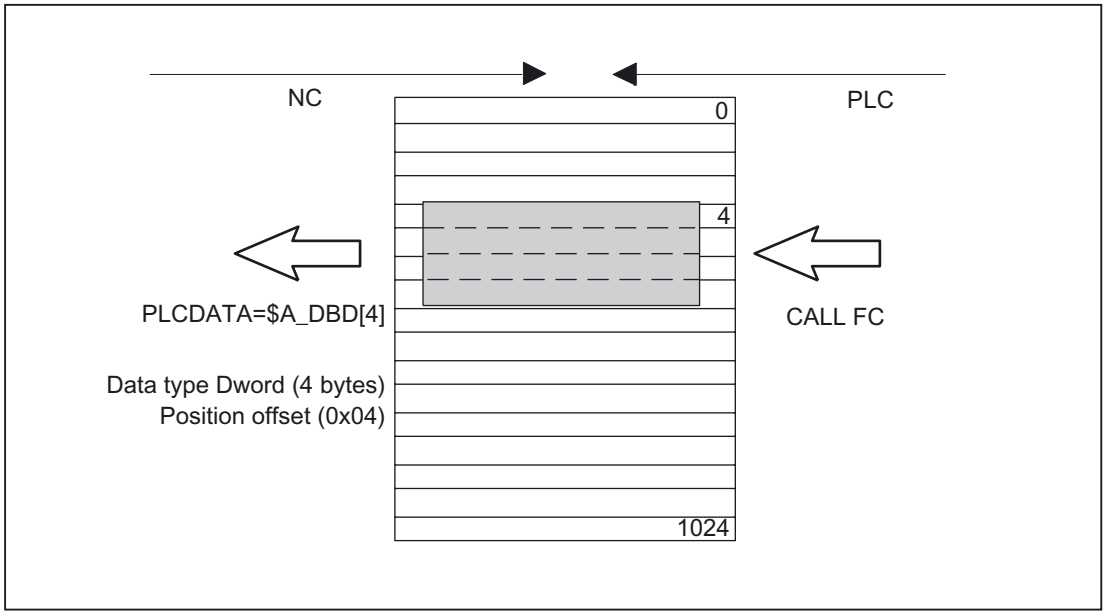


Fig. 2-9 Communications buffer (DPR) for NC/PLC communication

Supplementary conditions

- The user's programming engineer (NCK and PLC) is responsible for organizing the DPR memory area. No checks are made for inconsistencies in the configuration.
- A total of 1024 bytes are available in the input and output directions.
- Singlebit operations are not supported and must be linked back to byte operations by the user (programming engineer).
- 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's programming engineer is responsible for ensuring consistent transmission of 32-bit variables (double and real). A simple semaphore mechanism is provided 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) on the NCK, but only FLOAT (32 bits) on the PLC. The format used for storage in the dualport RAM is FLOAT. Conversion takes place respectively before/after storage in the dualport RAM.

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

Example
Addressing the problem by means of comparison on f "EPSILON" (minor deviation)

Block number	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 output variables that can be written to simultaneously can be set with:
MD28150 $MC_MM_NUM_VDIVAR_ELEMENTS

Example

A WORD is to be transferred from the PLC to the NC.

The position offset within the NCK 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 (extract)	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 (extract)                    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.

2.2.5 Access protection via password and keyswitch

2.2.5.1 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	Users	Access to (examples)
0	Password	Siemens	All functions, programs and data
1	Password	Machine manufacturer: Development	Defined functions, programs and data; e.g.,: entering options
2	Password	Machine manufacturer: Start-up engineer	Defined functions, programs and data; e.g.,: Bulk of machine data
3	Password	End user: Servicing	Assigned functions, programs and data
4	Keyswitch position 3	End user: Programmer, machine setter	Lower than protection level 0 to 3; defined by machine manufacturer or end user

Protection level	Type	Users	Access to (examples)
5	Keyswitch position 2	End user: Skilled operator without programming knowledge	Lower than protection level 0 to 3; defined by end user
6	Keyswitch position 1	End user: Trained operator without programming knowledge	Example: Program selection only, tool-wear entries, and work-offset entries
7	Keyswitch position 0	End user: Semi-skilled operator	Example: No entries and program selection possible, only the machine control panel can be operated

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.

2.2.5.2 Password

Set password

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

Example: HMI Advanced
DIAGNOSTICS operation area, softkey: SET PASSWORD

References:
/BAD/ Operator's Guide HMI Advanced

Delete password

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

Example: HMI Advanced
DIAGNOSTICS operation area, softkey: DELETE PASSWORD

References:
/BAD/ Operator's Guide HMI Advanced

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 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.

2.2.5.3 Keyswitch settings (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

Fig. 2-10 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
/IAM/ Installation & Startup Guide HMI/MMC; Access Protection

Default settings via the PLC user program

The keyswitch switch 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.

2.2.5.4 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.

Supplementary Conditions

3

3.1 Supplementary conditions

There are no supplementary conditions to note.

Examples

4

4.1 Example

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.

Prerequisites

Parameter-set changeover must be enabled:
MD35590 $MA_PARAMSET_CHANGE_ENABLE [AX1] = 1 or 2

The 1st parameter set for machine axis X1 is set, in accordance with machine data with index "0" NC/PLC interface:
DB31, DBB9.0 – DBB9.2 = 0

Parameter-set-dependent machine data

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

Machine data	Remarks
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
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

Machine data	Remarks
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.

- Requirement by PLC user program:
  DB31, DBB9.0 – DBB9.2 = 3
  - 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, DBB69.0 – DBB69.2 = 3
  - The NC confirms/acknowledges the parameter-set changeover.

Data Lists

5

5.1 Machine data

5.1.1 Memory specific machine data

Number		Identifier: $MM_	Description
HMI Advanced	HMI Embedded		
9000	9000	LCD_CONTRAST	Contrast
9001	9001	DISPLAY_TYPE	Monitor type
	9002	DISPLAY_MODE	External monitor (1: black and white, 2: color)
	9003	FIRST_LANGUAGE	Foreground language
9004	9004	DISPLAY_RESOLUTION	Display resolution
	9005	PRG_DEFAULT_DIR	Basic setting Program directory
	9006	DISPLAY_BLACK_TIME	Time setting for screen saver
	9007	TABULATOR_SIZE	Tabulator length
9008	9008	KEYBOARD_TYPE	Keyboard type (0: OP, 1: MFII/QWERTY)
9009	9009	KEYBOARD_STATE	Shift behavior of keyboard during booting
9010		SPIND_DISPLAY_RESOLUTION	Display resolution for spindle values
9011	9011	DISPLAY_RESOLUTION_INCH	Display resolution for INCH_system of units
9012	9012	ACTION_LOG_MODE	Set action mode for action log
	9013	SYS_CLOCK_SYNC_TIME	System clock synchronizing time
9020	9020	TECHNOLOGY	Basic configuration for simulation and free contour programming
	9030	EXPONENT_LIMIT	Number of places to be displayed without exponent
	9031	EXPONENT_SCIENCE	Technical exponent representation in three steps
	9180	USER_CLASS_READ_TCARR	Protection level read tool carrier offsets general
	9181	USER_CLASS_WRITE_TCARR	Protection level write tool carrier offsets general
9200	9200	USER_CLASS_READ_TOA	Protection level read tool offsets general
9201	9201	USER_CLASS_WRITE_TOA_GEO	Protection level write tool geometry

Number		Identifier: $MM_	Description
9202	9202	USER_CLASS_WRITE_TOA_WEAR	Protection level write tool wear data
9203	9203	USER_CLASS_WRITE_FINE	Protection level write fine
9204		USER_CLASS_WRITE_TOA_SC	Protection level change total tool offsets
9205		USER_CLASS_WRITE_TOA_EC	Protection level change tool setup offsets
9206		USER_CLASS_WRITE_TOA_SUPVIS	Protection level change tool-monitoring limit values
9207		USER_CLASS_WRITE_TOA_ASSDNO	Change D No. assigned to a tool edge
9208		USER_CLASS_WRITE_MAG_WGROUP	Change wear group magazine location/mag.
9209	9209	USER_CLASS_WRITE_TOA_ADAPT	Protection level write tool adapter data
9210	9210	USER_CLASS_WRITE_ZOA	Protection level write settable zero offset
9211	9211	USER_CLASS_READ_GUD_LUD	Protection level read user variables
9212	9212	USER_CLASS_WRITE_GUD_LUD	Protection level write user variables
9213	9213	USER_CLASS_OVERSTORE_HIGH	Protection level extended overstore
9214	9214	USER_CLASS_WRITE_PRG_CONDIT	Protection level program control
9215	9215	USER_CLASS_WRITE_SEA	Protection level write setting data
	9216	USER_CLASS_READ_PROGRAM	Protection level read part program
	9217	USER_CLASS_WRITE_PROGRAM	Protection level enter part program
9218	9218	USER_CLASS_SELECT_PROGRAM	Protection level part program selection
9219	9219	USER_CLASS_TEACH_IN	Protection level TEACH IN
9220	9220	USER_CLASS_PRESET	Protection level PRESET
9221	9221	USER_CLASS_CLEAR_RPA	Protection level delete R parameters
9222	9222	USER_CLASS_WRITE_RPA	Protection level write R parameters
	9223	USER_CLASS_SET_V24	Protection level for RS232C interface parameterization
	9224	USER_CLASS_READ_IN	Protection level for import data
	9225	USER_CLASS_READ_CST	Protection level standard cycles
	9226	USER_CLASS_READ_CUS	Protection level user cycles
	9227	USER_CLASS_SHOW_SBL2	Skip single block2 (SBL2)
	9228	USER_CLASS_READ_SYF	Access level select directory SYF
	9229	USER_CLASS_READ_DEF	Access level select directory DEF
	9230	USER_CLASS_READ_BD	Access level select directory BD
9231		USER_CLASS_WRITE_RPA_1	Write protection for first RPA area
9232		USER_BEGIN_WRITE_RPA_1	Start of the first RPA area
9233		USER_END_WRITE_RPA_1	End of the first RPA area
9234		USER_CLASS_WRITE_RPA_2	Write protection for second RPA area
9235		USER_BEGIN_WRITE_RPA_2	Start of the second RPA area
9236		USER_END_WRITE_RPA_2	End of the second RPA area
9237		USER_CLASS_WRITE_RPA_3	Write protection for third RPA area
9238		USER_BEGIN_WRITE_RPA_3	Start of the third RPA area

Number		Identifier: $MM_	Description
9239		USER_END_WRITE_RPA_3	End of the third RPA area
9240		USER_CLASS_WRITE_TOA_NAME	Change tool designation and duplo
9241		USER_CLASS_WRITE_TOA_Type	Change tool type
	9460	PROGRAMM_SETTINGS	Resetproof data storage for settings in the PROGRAM operating area
	9461	CONTOUR_END_TEXT	String to be added to end of contour on completion of input
	9478	TO_OPTION_MASK	Variants of tool offsets
	9500	NC_PROPERTIES	NC_properties: Bit 0: Digital drives, Bit 1: Software startup switch
9510		USER_CLASS_DIRECTORY1_P	Protection level for network drive 1 for PROGRAM operating area
9511		USER_CLASS_DIRECTORY2_P	Protection level for network drive 2 for PROGRAM operating area
9512		USER_CLASS_DIRECTORY3_P	Protection level for network drive 3 for PROGRAM operating area
9513		USER_CLASS_DIRECTORY4_P	Protection level for network drive 4 for PROGRAM operating area
9516		USER_CLASS_DIRECTORY1_M	Protection level for network drive 1 for MACHINE operating area
9517		USER_CLASS_DIRECTORY2_M	Protection level for network drive 2 for MACHINE operating area
9518		USER_CLASS_DIRECTORY3_M	Protection level for network drive 3 for MACHINE operating area
9519		USER_CLASS_DIRECTORY4_M	Protection level for network drive 4 for MACHINE operating area

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

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

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

5.1.5 Drive-specific machine data

Number	Identifier: $MD_	Description
1403	PULSE_SUPPRESSION_SPEED	Shutoff speed for pulse suppression
1404	PULSE_SUPPRESSION_DELAY	Time for pulse suppression
1417	SPEED_THRESHOLD_X	n_x for $n_{act} < n_{setp}$ signal
1418	SPEED_THRESHOLD_MIN	n_{min} for $n_{act} < n_{setp}$ signal
1426	SPEED_DES_EQ_ACT_TOL	Tolerance band for $n_{set} = n_{act}$ signal
1427	SPEED_DES_EQ_ACT_DELAY	Delay time $n_{set} = n_{act}$ signal
1428	TORQUE_THRESHOLD_X	Threshold torque
1429	TORQUE_THRESHOLD_X_DELAY	Delay time $n_d < n_{dx}$ signal
1602	MOTOR_TEMP_WARN_LIMIT	Maximum motor temperature
1603	MOTOR_TEMP_ALARM_TIME	Time for motor temperature alarm
1604	LINK_VOLTAGE_WARN_LIMIT	DC link under voltage warning threshold

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)

5.3 Signals

5.3.1 Signals to NC

DB number	Byte.Bit	Description
10	56.7	Keyswitch setting 0 to 3

5.3.2 Signals from NC

DB number	Byte.Bit	Description
10	103.0	Remote diagnostics (MMC alarm is active)
10	103.5	AT box ready
10	103.6	MMC temperature limit
10	103.7	MMC battery alarm
10	104.7	NCK CPU Ready
10	108.1	MMCCPU2Ready (MMC to OPI or MPI)
10	108.2	MMC CPU1 Ready (MMC to MPI)
10	108.3	MMC CPU1 Ready (MMC to OPI, standard link)
10	108.6	Drive Ready
10	108.7	NC Ready
10	109.0	NCK alarm is active
10	109.5	NCU573 heat-sink temperature alarm
10	109.6	Air temperature alarm
10	109.7	NCK battery alarm

5.3.3 Signals to channel

DB number	Byte.Bit	Description
21, ...	6.2	Delete distance-to-go (channelspecific)

5.3.4 Signals from channel

DB number	Byte.Bit	Description
21, ...	36.6	Channelspecific NCK alarm is active
21, ...	36.7	NCK alarm with processing stop present
21, ...	318.7	Overstore active

5.3.5 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	1.3	Axis/spindle disable
31, ...	1.4	Follow-up mode
31, ...	1.5	Position measuring system 1
31, ...	1.6	Position measuring system 2
31, ...	2.1	Controller enable
31, ...	2.2	Delete distance-to-go (axisspecific)/Spindle reset
31, ...	1, 2	Parameter set switchover (request)
31, ...	20.0	Rampup times
31, ...	20.1	Rampfunction-generator fast stop
31, ...	20.2	Torque limit 2
31, ...	20.3	Speed setpoint smoothing
31, ...	21.2	Drive parameter set selection A, B, C
31, ...	21.4	Motor selection A, B
31, ...	21.5	Motor selection to follow
31, ...	21.6	Speed controller integrator disable
31, ...	21.7	Pulse enable

5.3.6 Signals from axis/spindle

DB number	Byte.Bit	Description
31, ...	60.4 / 60.5	Referenced, synchronized 1 / Referenced, synchronized 2
31, ...	61.3	Follow-up mode active
31, ...	64.6 / 64.7	Traverse command minus / plus
31, ...	61.3	Follow-up mode active
31, ...	61.4	Axis/spindle stops
31, ...	61.5	Position controller active
31, ...	61.6	Speed control loop active
31, ...	61.7	Current controller active
31, ...	1, 2	Parameter set switchover (feedback)
31, ...	76.0	Lubrication pulse
31, ...	92.0	Setup mode active
31, ...	92.1	Rampfunction-generator fast stop active
31, ...	92.2	Torque limit 2 active
31, ...	92.3	Speed setpoint smoothing active
31, ...	93.2	Active drive parameter set A, B, C
31, ...	93.4	Active motor A, B
31, ...	93.5	Drive Ready
31, ...	93.6	Speed controller integrator disabled
31, ...	93.7	Pulses enabled
31, ...	94.0	Motor temperature prewarning
31, ...	94.1	Heat-sink temperature prewarning
31, ...	94.2	Ramp-up function completed
31, ...	94.3	$\|M_d\| < M_{dx}$
31, ...	94.4	$\|n_{act}\| < n_{min}$
31, ...	94.5	$\|n_{act}\| < n_x$
31, ...	94.6	$n_{act} = n_{set}$
31, ...	94.7	Variable signaling function
31, ...	95.0	$U_{DC\ link}$<warning threshold

5.3.7 Signals to operator panel front

DB number	Byte.Bit	Description
19	0.0	Screen bright
19	0.1	Darken screen
19	0.2	Key disable
19	0.3	Delete Cancel alarms (HMI Advanced only)
19	0.4	Delete Recall alarms (HMI Advanced only)
19	0.7	Actual value in WCS
19	10.0	Programming area selection
19	10.1	Alarm area selection
19	10.2	Tool offset selection
19	10.7	ShopMill control signal
19	12.2	COM2 active (job byte of PLC)
19	12.3	COM1 active (job byte of PLC)
19	12.4	RS232 stop (job byte of PLC)
19	12.5	RS232 external (job byte of PLC)
19	12.6	RS232 OFF (job byte of PLC)
19	12.7	RS232 ON (job byte of PLC)
19	13.5	Unload part program
19	13.6	Load part program
19	13.7	Part program selection
19	14.7	File system active/passive (for HMI Embedded, always active)
19	16.7	File system active/passive (for HMI Advanced, always passive)
19	44.0	Mode change disable

5.3.8 Signals from operator panel front

DB number	Byte.Bit	Description
19	20.1	Screen is dark
19	20.7	Switch over MCS/WCS
19	24.0	Error (Acknowledgment byte for current RS232 status)
19	24.1	O.K. (Acknowledgment byte for current RS232 status)
19	24.2	COM2 active (Acknowledgment byte for current RS232 status)
19	24.3	COM1 active (Acknowledgment byte for current RS232 status)
19	24.4	RS232 stop (Acknowledgment byte for current RS232 status)
19	24.5	RS232 External (Acknowledgment byte for current RS232C status)
19	24.6	RS232 OFF (Acknowledgment byte for current RS232 status)
19	24.7	RS232 ON (Acknowledgment byte for current RS232 status)
19	26.0	Error (Part program handling status)
19	26.1	O.K. (Part program handling status)
19	26.3	Active (Part program handling status)
19	26.5	Unload (Part program handling status)
19	26.6	Load (Part program handling status)
19	26.7	Select (Part program handling status)
19	42.0	FC9: Measure in JOG mode
19	45.0	FC9 Out: Active
19	45.1	FC9 Out: Done
19	45.2	FC9 Out: Error
19	45.3	FC9 Out: StartErr

Index

6

611D Ready, 2-3

A

Access authorization, 2-31
Access features, 2-32
Access security, 2-31
Actual value synchronization, 2-15
Actual value in workpiece coordinate system, 2-6
Air temperature alarm, 2-4
Axis disable, 2-8
Axis/spindle stops, 2-17

C

Cancel alarms, 2-6
Channelspecific NCK alarm is active, 2-4
Contrast, 2-23
Controller parameter set switchover, 4-1
Current controller active, 2-17
Cyclical signal exchange, 2-1, 2-2

D

Darken screen, 2-5
Data channel
  High-speed, 2-27
DB10
  DBX103.0, 2-4
  DBX104.7, 2-3
  DBX108.1, 2-3
  DBX108.2, 2-3
  DBX108.3, 2-3
  DBX108.6, 2-3
  DBX108.7, 2-3
  DBX109.0, 2-4
  DBX109.6, 2-4
  DBX109.7, 2-4
  DBX56.1, 2-13
  DBX56.4, 2-34
  DBX56.5, 2-34
  DBX56.6, 2-34
  DBX56.7, 2-34
DB19
  DBB12, 2-6
  DBB13, 2-6
  DBB14, 2-6
  DBB15, 2-6
  DBB16, 2-6
  DBB17, 2-6
  DBB24, 2-7
  DBB25, 2-7
  DBB26, 2-7
  DBB27, 2-7
  DBX0.3, 2-6
  DBX0.4, 2-6
  DBX0.0, 2-5
  DBX0.1, 2-5
  DBX0.2, 2-5
  DBX0.7, 2-6
  DBX20.3, 2-6
  DBX20.4, 2-6
DB21, ...
  DBX36.6, 2-4
  DBX36.7, 2-4
  DBX6.2, 2-7
DB31
  DBB69.0, 4-2
  DBB69.1, 4-2
  DBB69.2, 4-2
  DBB9.0, 4-1, 4-2
  DBB9.1, 4-1, 4-2
  DBB9.2, 4-1, 4-2
DB31, ...
  DBX1.0, 2-7
  DBX1.3, 2-8
  DBX1.4, 2-8, 2-9, 2-11, 2-14
  DBX1.5, 2-12, 2-13
  DBX1.6, 2-12, 2-13
  DBX2.1, 2-8, 2-9, 2-11, 2-13
  DBX2.2, 2-15
  DBX20.1, 2-18
  DBX20.2, 2-18
  DBX20.3, 2-18
  DBX21.0, 2-18

DBX21.1, 2-18
DBX21.2, 2-18
DBX21.3, 2-19
DBX21.4, 2-19
DBX21.5, 2-19
DBX21.6, 2-19
DBX21.7, 2-13, 2-19
DBX60.4, 2-12
DBX60.6, 2-12
DBX60.7, 2-12
DBX61.0, 2-7, 2-17
DBX61.3, 2-8, 2-9, 2-17
DBX61.4, 2-16, 2-17
DBX61.5, 2-14, 2-17
DBX61.6, 2-14, 2-17
DBX61.7, 2-17
DBX69.0, 2-17
DBX76.0, 2-18
DBX9.0, 2-16, 2-17
DBX9.1, 2-16, 2-17
DBX9.2, 2-16, 2-17
DBX9.3, 2-17
DBX92.0, 2-20
DBX92.1, 2-20
DBX92.2, 2-20
DBX92.3, 2-20
DBX93.0, 2-18, 2-20
DBX93.1, 2-18, 2-20
DBX93.2, 2-18, 2-20
DBX93.3, 2-21
DBX93.4, 2-21
DBX93.5, 2-13, 2-21
DBX93.6, 2-19, 2-21
DBX93.7, 2-20, 2-21
DBX94.0, 2-21
DBX94.1, 2-21
DBX94.2, 2-21
DBX94.3, 2-22
DBX94.4, 2-22
DBX94.5, 2-22
DBX94.6, 2-22
DBX94.7, 2-22
DBX95.0, 2-22
DBX95.7, 2-22
DB31, ...
DBX69.1, 2-17
DB31, ... DBX60.5, 2-12
DB31, ... DBX69.2, 2-17
DC link undervoltage warning threshold, 2-22
DC link voltage, 2-22
DClink, 2-22
Default passwords, 2-33
Delete distance-to-go, 2-7
Display resolution, 2-23
Drive-test travel enable, 2-7
Drive-test travel request, 2-17

F

Followup mode active, 2-17
Foreground language, 2-23

H

High-speed data channel, 2-27

I

Interface signals
For operator panel, 2-5
Interface signals (A2), 1-1
Interface signals from and to channel, 2-7
Interface signals spindle reset, 2-15
Interpolatory axis grouping, 2-14

K

Key disable, 2-5
Key switch, 2-34

L

Lockable data areas, 2-35
Lubrication pulse, 2-18

M

Machine coordinate system, 2-6
MCS, 2-6
Md
|Md| < Mdx, 2-22
MD10131 $MN_SUPPRESS_SCREEN_REFRESH, 2-24
MD11270 $MN_DEFAULT_VALUES_MEM_MASK, 2-27
MD1417 $MD_SPEED_THRESHOLD_X, 2-22
MD1418 $MD_SPEED_THRESHOLD_MIN, 2-22
MD1426 $MD_SPEED_DES_EQ_ACT_TOL, 2-21, 2-22
MD1428 $MD_TORQUE_THRESHOLD_X, 2-22
MD1604 $MD_LINK_VOLTAGE_WARN_LIMIT, 2-22
MD18150 $MM_GUD_VALUES_MEM, 2-27
MD21015 $MC_INVOLUTE_RADIUS_DELTA, 2-25
MD21016 $MC_INVOLUTE_AUTO_ANGLE_LIMIT, 2-26

MD28150 $MC_MM_NUM_VDIVAR_ELEMENTS, 2-30
MD28530 $MC_MM_PATH_VELO_SEGMENTS, 2-27
MD31050 $MA_DRIVE_AX_RATIO_DENOM, 4-1
MD31060 $MA_DRIVE_AX_RATIO_NUMERA, 4-1
MD32200 $MA_POSCTRL_GAIN, 2-16, 4-1
MD32800 $MA_EQUIV_CURRCTRL_TIME, 4-2
MD32810 $MA_EQUIV_SPEEDCTRL_TIME, 4-2
MD32910 $MA_DYN_MATCH_TIME, 4-2
MD33050 $MA_LUBRICATION_DIST, 2-18
MD35130 $MA_AX_VELO_LIMIT, 4-2
MD35590 $MA_PARAMSET_CHANGE_ENABLE, 2-16, 4-1
MD36060 $MA_STANDSTILL_VELO_TOL, 2-16, 2-17
MD36610 $MA_AX_EMERGENCY_STOP_TIME, 2-14
MD36620 $MA_SERVO_DISABLE_DELAY_TIME, 2-14
MD9000 $MM_LCD_CONTRAST, 2-23
MD9001 $MM_DISPLAY_TYPE, 2-23
MD9003 $MM_FIRST_LANGUAGE, 2-23
MD9004 $MM_DISPLAY_RESOLUTION, 2-23
MD9006 $MM_DISPLAY_BLACK_TIME, 2-5
MMC alarm, 2-4
MMC CPU1 Ready, 2-3
MMC2 CPU Ready, 2-3
Monitor type, 2-23

N

nact
  |nact| < nmin, 2-22
  |nact| < nx, 2-22
nact = nsetp, 2-22
NCK alarm is active, 2-4
NCK alarm with processing stop, 2-4
NCK battery alarm, 2-4
NCK CPU Ready, 2-3

P

Password, 2-32
  Resetting, 2-33
  Setting, 2-32
PLC/NCK interface, 2-1
Position controller active, 2-17
Position measuring system, 2-12
Protection levels, 2-35

R

Ramp-up function completed, 2-21
Read/write PLC variable, 2-27
Recall alarms, 2-6

S

Servo enable, 2-13
Signals
  Alarm signals, 2-4
  Axis/spindlespecific (DB31, etc.), 2-2, 2-3
  Channelspecific (DB21, etc.), 2-2
  General (DB10), 2-2
  Ready signals, 2-3
Speed control loop active, 2-17
Spindle disable, 2-8
Spindle reset, 2-15

U

U(DClink), 2-22
Undervoltage threshold, 2-22

V

Variable signaling function, 2-22

W

WCS, 2-6
Workpiece coordinate system, 2-6

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Axis Monitoring, Protection Zones (A3)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU System Software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Axis monitoring functions 1-1

1.2 Protection zones 1-1

2 Detailed Description 2-1

2.1 Motion monitoring functions 2-1
2.1.1 Contour monitoring 2-1
2.1.2 Following Error Monitoring 2-2
2.1.3 Positioning monitoring 2-3
2.1.4 Zero speed monitoring 2-5
2.1.5 Exact stop and standstill tolerance dependent on the parameter set 2-6
2.1.6 Clamping monitoring 2-7
2.1.7 Speed setpoint monitoring 2-14
2.1.8 Actual velocity monitoring 2-15

2.2 Encoder Monitoring 2-16
2.2.1 Encoder limit frequency monitoring 2-16
2.2.2 Zero mark monitoring 2-18
2.2.3 Hardware faults and contamination signal 2-21

2.3 Monitoring of static limits 2-22
2.3.1 Overview of the limit switch monitoring 2-22
2.3.2 Hardware limit switches 2-22
2.3.3 Software limit switch 2-23
2.3.4 Working area limitation 2-25

2.4 Protection zones 2-29
2.4.1 General 2-29
2.4.2 Types of protection zone 2-30
2.4.3 Definition via part program instruction 2-33
2.4.4 Definition via system variable 2-36
2.4.5 Activation and deactivation of protection zones 2-39
2.4.6 Protection zone violation and temporary enabling of individual protection zones 2-42
2.4.7 Restrictions in protection zones 2-48

3 Supplementary Conditions 3-1

3.1 Axis monitoring functions 3-1

4 Examples 4-1

4.1 Definition and activation of protection zones 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Channelspecific machine data 5-1
5.1.3 Axis/spindlespecific machine data 5-2

5.2 Setting data 5-3

5.2.1 Axis/spindlespecific setting data 5-3
5.3 Signals 5-4
5.3.1 Signals to channel 5-4
5.3.2 Signals from channel 5-4
5.3.3 Signals to axis 5-5
Index Index-1

Tables

Table 4-1 Part program excerpt for protection zone definition: 4-2
Table 4-2 Protection zone: Spindle chuck 4-3
Table 4-3 Protection zone: Workpiece and tool holder 4-5
Table 4-4 Part program excerpt for activating the three protection zones for spindle chuck, workpiece, and tool holder: 4-10

Brief Description

1

1.1 Axis monitoring functions

Axis monitoring functions

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

- Motion monitoring functions
  - Contour monitoring
  - Positioning monitoring
  - Zero speed monitoring
  - Clamping monitoring
  - Speed setpoint monitoring
  - Actual velocity monitoring
  - Encoder Monitoring
- Monitoring of static limits
  - Limit switch monitoring
  - Working area limitation

1.2 Protection zones

Function

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

During automatic execution of part programs in the AUTOMATIC or MDI 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, the zone can be entered. 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.

Detailed Description 2

2.1 Motion monitoring functions

2.1.1 Contour monitoring

Contour error

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

A distinction is made between:

- Linear signal distortions

  This is 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 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 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.

- Nonlinear signal distortions

  This is caused by:

  – Activation of the current limitation within the machining area

  – Activation of the limitation of the set speed

  – 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 movement along straight lines, a contour error arises due to a reversal error outside the position feedback 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.

2.1.2 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)

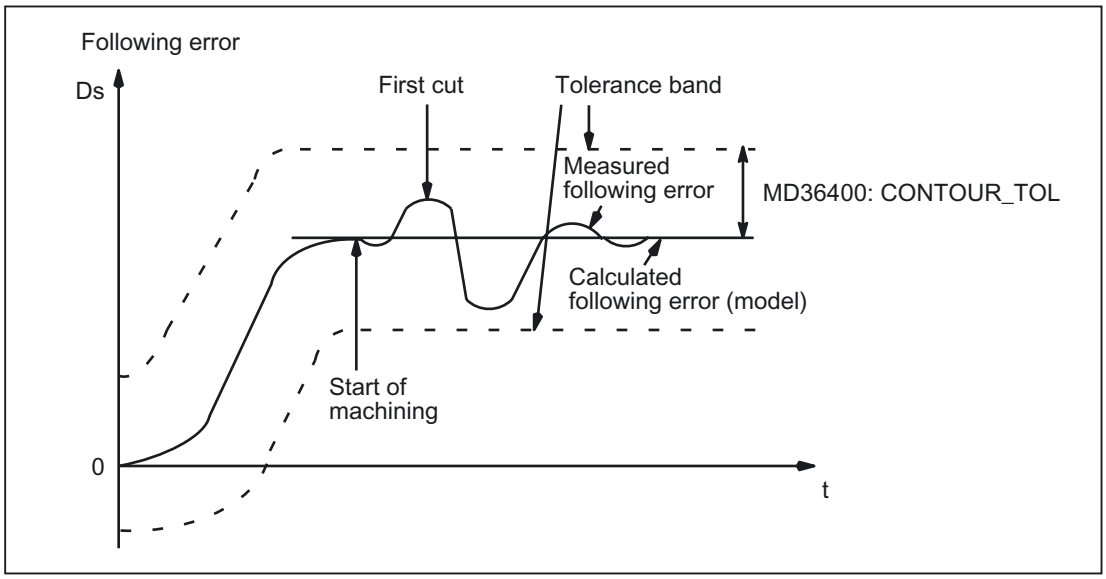


Fig. 2-1 Following Error Monitoring

Effectivity

The following error monitoring only operates with activated position control for:

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

Fault case

In case of a fault after exceeding of the configured tolerance limit, the following occurs:

- Alarm: "25050 Axis <*Axis identifier*> Contour monitoring"
- The affected axis is stopped via the configured braking ramp in follow-up mode:

  MD36610 $MA_AX_EMERGENCY_STOP_TIME
  (Time for braking ramp when an error occurs)
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

2.1.3 Positioning monitoring

Function

At the end of a positioning operation:

- Set velocity == 0 **AND**
- DB31, ... DBX64.6 / 64.7 == 0 (traverse command minus / plus)

checks the positioning monitoring 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)

Note

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

Relationship between the various position monitoring functions

After reaching "Exact stop fine", the position monitoring is deactivated and replaced by the zero speed monitoring.

The following image shows the relationship between the various position monitoring functions:

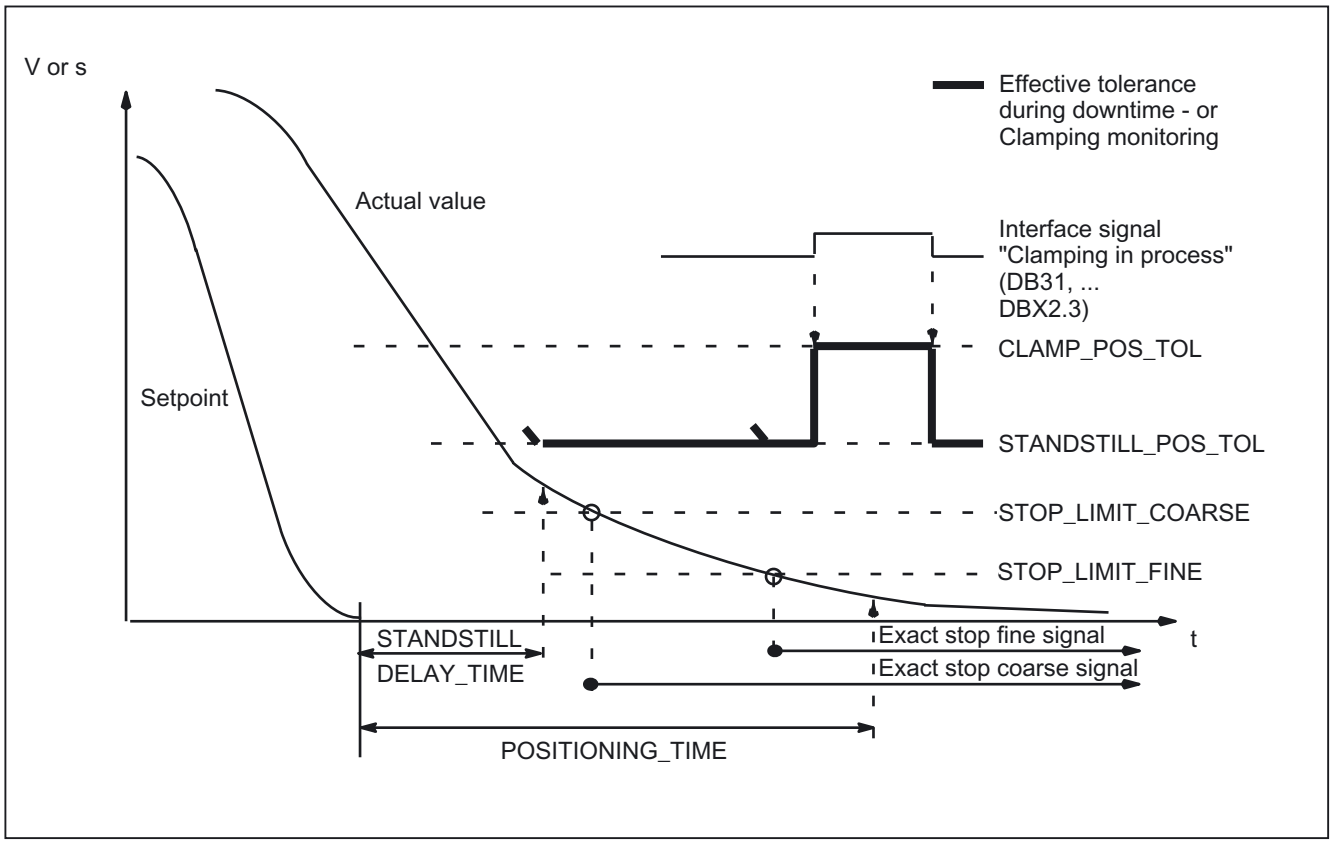


Fig. 2-2 Positioning, zero speed and clamping monitoring

Setting notes

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

Effectivity

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

- Linear axes
- Rotary axes
- Position-controlled spindles

Effects

Upon exceeding of the configured position monitoring time, the following occurs:

- Alarm: "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
  (Time for braking ramp when an error occurs)
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

2.1.4 Zero speed monitoring

Function

At the end of a positioning operation:
Set velocity == 0 AND
DB31, ... DBX64.6 / 64.7 == 0 (Travel command minus / plus)
checks the standstill monitoring 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)

MD36060 $MA_STANDSTILL_POS_TOL (Standstill tolerance)

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

After completion of the positioning operation:
DB31, ...DBX60.6 / 60.7 == 1 (Position reached with exact stop coarse / fine)
the machine axis may not leave the standstill tolerance as long as no new travel command is pending.

Effectivity

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

- Linear axes
- Rotary axes
- Position-controlled spindles

Effect

Upon exceeding of the delay time and/or the standstill tolerance, the following occurs:

- Alarm: "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
  (Duration of the brake ramp under fault conditions)
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

2.1.5 Exact stop and standstill tolerance dependent on the parameter set

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.

2.1.6 Clamping monitoring

Clamping monitoring

Afterward, the zero speed monitoring monitors the compliance with the configured standstill tolerance.

For machine axes that are mechanically clamped upon completion of a positioning operation, larger motions can result from 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 (follow-up mode)

Effect

Upon exceeding of the clamping tolerance, the following occurs:

- Alarm: "26000 clamping monitoring" is displayed.
- The affected axis is stopped via the configured braking ramp in follow-up mode:

  MD36610 $MA_AX_EMERGENCY_STOP_TIME
  (Time for braking ramp when an error occurs)
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

Automatic stopping for removal of the clamp

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'

Note

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

DB31, ... DBX2.2 = 0 (servo enable): no servo enable ⇒ axis is clamped

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

Prerequisites 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 = 0 (servo enable): Axis is clamped

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

The following image shows an example of the interface signals and states upon releasing of the axis clamp. The part program blocks `N310` and `N410` refer to the schematic example under certain boundary conditions.

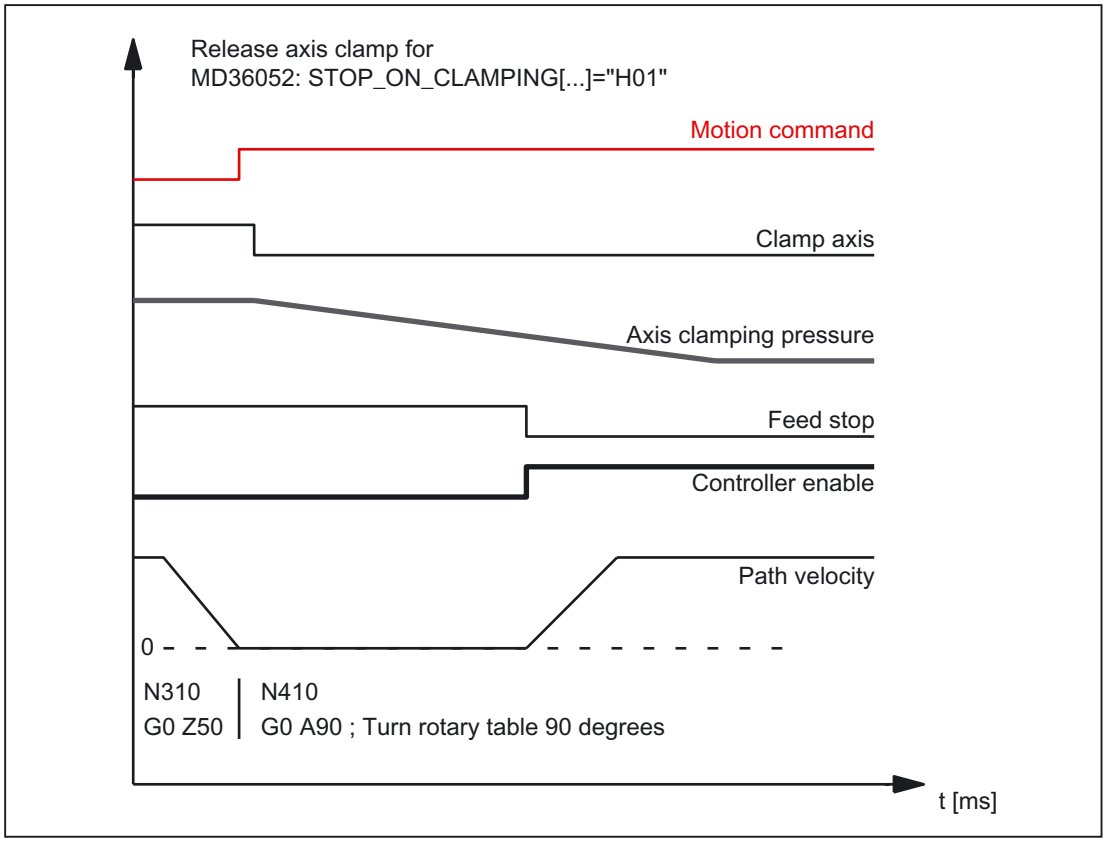


Fig. 2-3 Example 1: Interface signals and states

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

Parameterization:

MD36052 $MA_STOP_ON_CLAMPING = 'H03'

Prerequisites 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 clamp even if only during positioning (G0).

The following image shows an example of the interface signals and states upon releasing of the axis clamp. The part program blocks N220, N310 and N410 refer to the schematic example under certain boundary conditions.

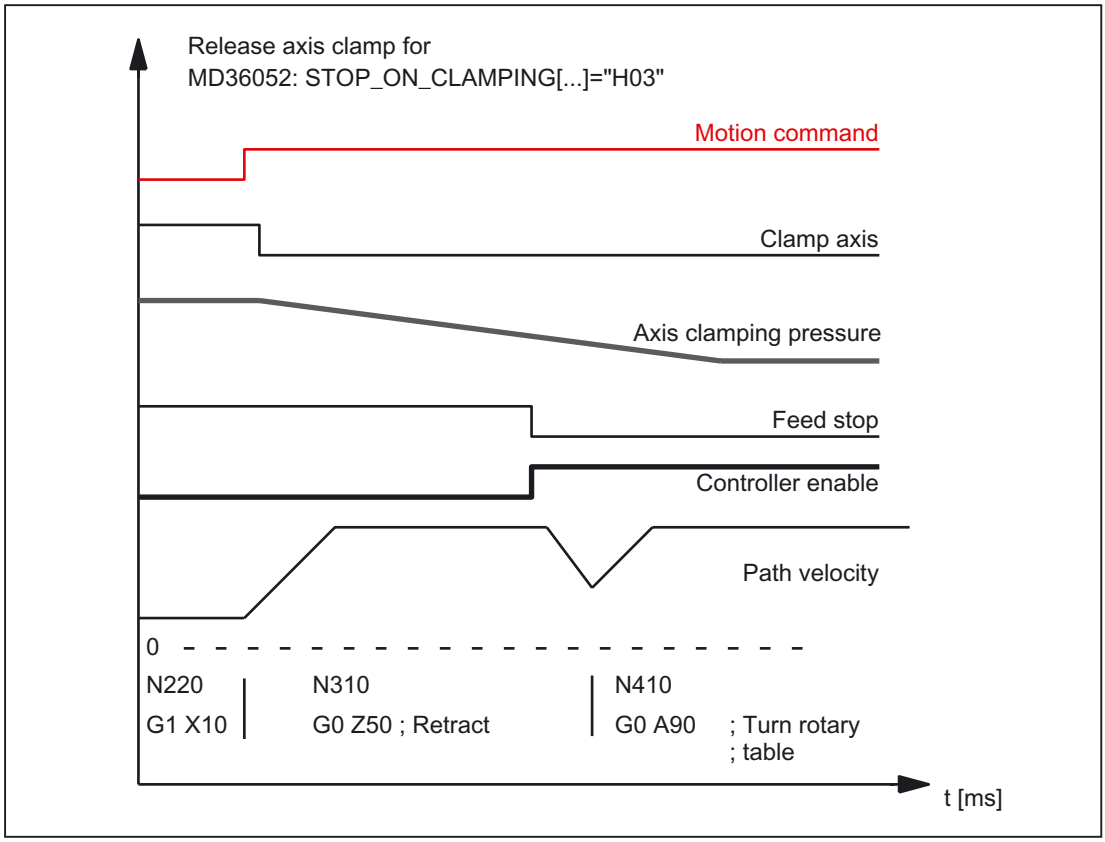


Fig. 2-4 Example 2: Interface signals and states

Automatic stopping for setting of the clamp

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 STOP_ON_CLAMPING = 'H04'

Prerequisites 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 (G0)).
- The axis is clamped if the feed rate 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 image 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 certain boundary conditions.

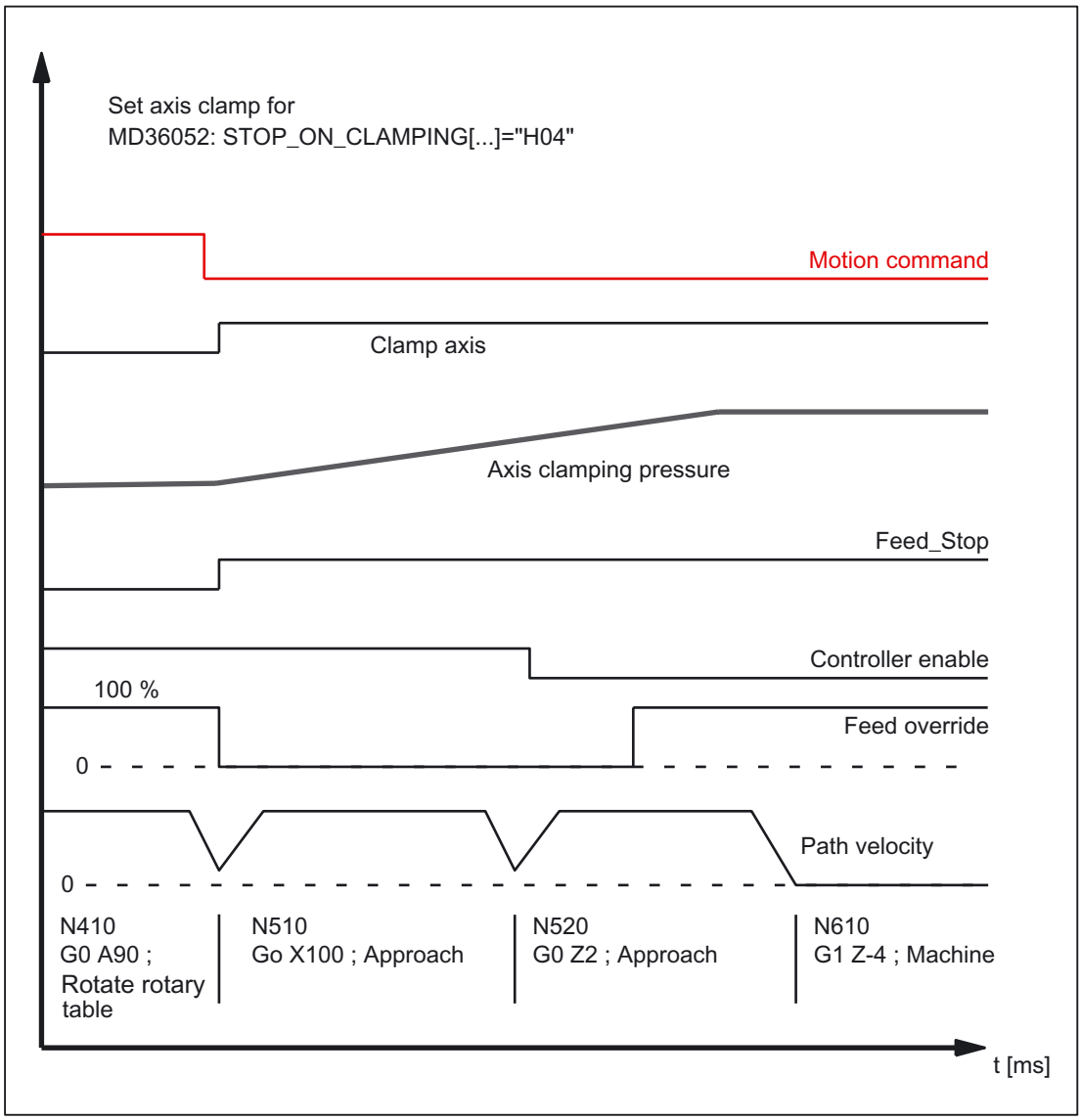


Fig. 2-5 Example 3: Interface signals and states

Constraints

Continuous-path mode

For the above-mentioned functions:

- Automatic stopping for removal of the clamp
- Optimized releasing of the axis clamp via travel command
- Automatic stopping for setting of the clamp

the "Look Ahead" function must be active.

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

Example:

After insertion of the part program blocks N320 and N420 into the part program used in the examples, the function behaves as follows:

```
N100     G0 X0 Y0 Z0 A0 G90 G54 F500
N101     G641 ADIS=.1 ADISPOS=5
N210     G1 X10                                    ; Edit
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                                    ; Edit
N620     G1 X0 Y-20
```

- MD36052 STOP_ON_CLAMPING = 'H03' is no longer operative.

  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 continuouspath mode. The Look Ahead stopping on N410 would not be necessary because stopping occurs anyway.

- MD36052 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 STOP_ON_CLAMPING = 'H01' or 'H04'

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

- MD36052 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 STOP_ON_CLAMPING = 'H04'
  generates a Look Ahead stop for the path motion if the feed rate overrate == 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 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 (clamping in progress)
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 for 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:
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 / 60.7 (position reached with exact stop coarse / fine)

2.1.7 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)

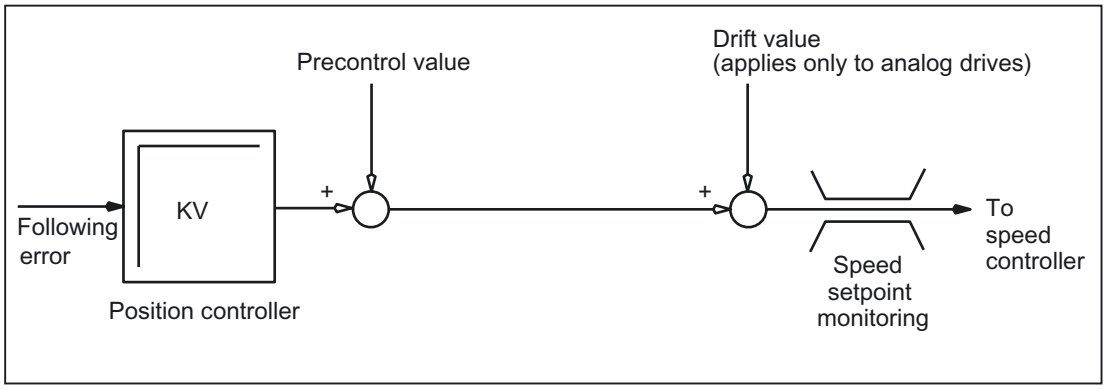


Fig. 2-6 Speed setpoint calculation

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)

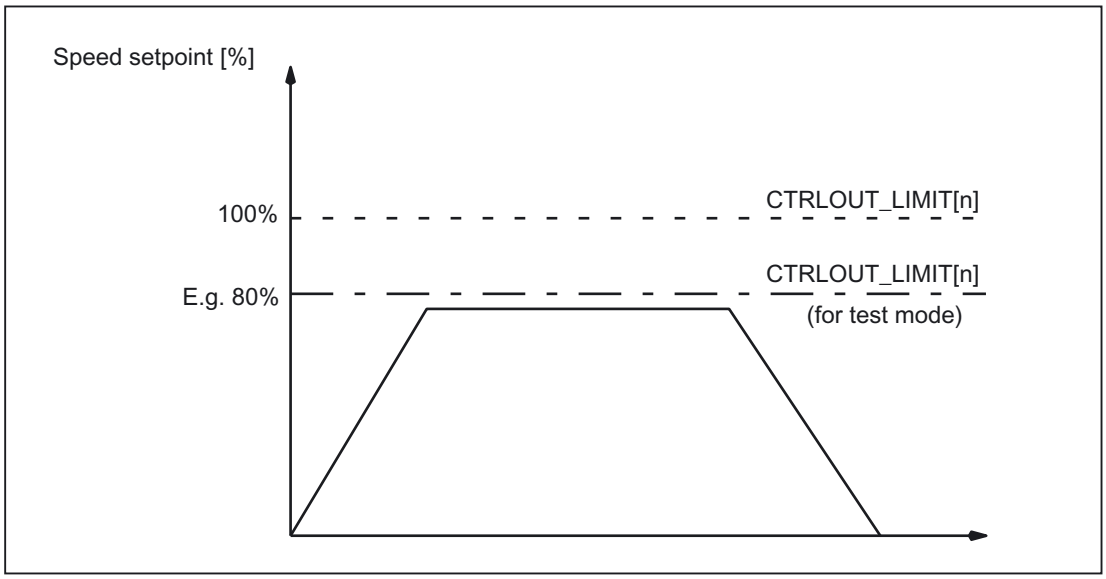


Fig. 2-7 Speed setpoint limitation

Speed setpoint monitoring delay

To prevent an error reaction from occurring in every speed limitation instance, a delay time can be configured. Only if the speed limitation is required for longer than the configured time does the corresponding error reaction occur.

MD36220 $MA_ CTRLOUT_LIMIT_TIME (Speed setpoint monitoring delay)

Effectivity

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

Effect

Upon exceeding of the configured delay time, the following occurs:

- Alarm: "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
  (Time for braking ramp when an error occurs)
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

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.

2.1.8 Actual velocity monitoring

Function

The actual velocity monitoring monitors the current actual velocity of a machine axis for the configured threshold:

MD36200 $MA_AX_VELO_LIMIT (velocity monitoring threshold)

As long as no error is present, the actual velocity of an axis/spindle never becomes greater than the configured threshold because the threshold is entered approx. 5-10% larger than the maximum traversing velocity.

Exceeding of the configured threshold of the velocity monitoring generally only occurs during drive errors.

Activation

Actual velocity monitoring is activated as soon as the active measuring system:
DB31, ... DBX1.5 / 1.6 (position measuring system 1 / 2)
supplies valid actual values (encoder limit frequency is not exceeded).

Effectivity

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

Effect

Upon exceeding of the threshold, the following occurs:

- Alarm: "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
  (Time for braking ramp when an error occurs)
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

2.2 Encoder monitoring

2.2.1 Encoder limit frequency monitoring

Function

Monitoring of the parameterized encoder limit frequency:
MD36300 $MA_ENC_FREQ_LIMIT (encoder limit frequency)

Effectivity

The encoder limit frequency always refers to the active measuring system selected in the NC/PLC interface:
DB31, ... DBX1.5 / 1.6 (position measuring system 1 / 2)

The encoder limit frequency is operative for:

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

Effect

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

  – Alarm: "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
  (Time for braking ramp when an error occurs)

  – If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

  After the axis have stopped, the position control is automatically resumed.

Note

If the encoder limit frequency is exceeded, the position-controlled machine axis must be re-referenced.

References:
/FB/ Description of Functions Basic Machine; Reference Point Approach (R1)

2.2.2 Zero mark monitoring

Function

The zero mark monitoring monitors the number of encoder impulses between two encoder zero marks of the active measuring system. In a properly functioning measuring system, the number of encoder impulses detected from encoder zero mark to encoder zero mark must remain the same.

MD36310 $MA_ENC_ZERO_MONITORING (Zero mark monitoring)

Note

The zero mark monitoring is only active below the assigned encoder limit frequency for new synchronization:
MD36302 $MA_ENC_FREQ_LIMIT_LOW

The type of zero mark monitoring depends on the type of encoder used:
MD30240 $MA_ENC_TYPE = <*encoder type*>

Encoder type	Meaning
0	Simulation
1	Raw signal generators (voltage, current, EXE etc.) -> High resolution
2	Rectangular signal encoder (standard, no. of PPRs quadrupled)
3	Encoder for stepper motor
4	Absolute value encoder - SIMODRIVE 611D: absolute encoder with EnDat interface only - PROFIBUS drives: No restriction
5	Encoder with SSI interface

The following table shows the corresponding context:

MD36310	*Encoder type*	Function
0	-	No zero mark monitoring.
100	-	No zero mark monitoring and suppress encoder alarms.
> 0	0	No zero mark monitoring.
	3	No zero mark monitoring.
	1	Number of detected changes on which an alarm is to be output. Every time the encoder is switched on, counting starts at "0".
	2	
	4	Permissible deviation between the absolute and incremental encoder track in 1/2 rough lines. It is generally sufficient to enter a 1/2 rough line. If this is exceeded, alarm 20510 is triggered. (See below: Boundary conditions)
	5	No zero mark monitoring. Power-ON-Alarm 25000 instead of Reset-Alarm 20510.

Effect

Active measuring system

The following occurs when the zero mark monitoring is initiated:

- Alarm: "25020 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
  (Time for braking ramp when an error occurs)
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).

Passive measuring system
The following occurs when the zero mark monitoring is initiated:

- Alarm: " 25020 Axis <*Axis identifier*> Zero mark monitoring of passive encoders"
  The alarm is only displayed.

Note

Absolute encoders

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. For information about adjusting absolute encoders, refer to:

References:
/FB/ Description of Functions Basic Machine; Reference Point Approach (R1);
Section: Referencing with absolute encoders

Restrictions

Absolute encoder
The monitoring of the deviation between the absolute and incremental encoder track:
MD36310 $MA_ENC_ZERO_MONITORING > 0
MD30240 $MA_ENC_TYPE == 4
only detects deviations due to contamination of the absolute track. Corruptions of the incremental track are not recognized. This type of monitoring is therefore intended only as a supplementary monitoring that can be used when disturbances of the absolute position are suspected.

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.

2.2.3 Hardware faults and contamination signal

Function

Monitoring of the measuring systems of a machine axis with regard to:

- Hardware faults
- Contamination of the measuring system

Fault case

In the event of an error:

- Alarm: "25000 Axis <*Axis identifier*> Hardware fault active encoder"
- Alarm: "25001 Axis <*Axis identifier*> Hardware fault passive encoder"
- Alarm: "25010 Axis <*Axis identifier*> Contamination of measuring system"
- Alarm: "25011 Axis <*Axis identifier*> Contamination of passive encoder"
- In the event of a fault in the active measuring system:
  - The affected axis is stopped via the configured braking ramp in follow-up mode:

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

  - If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop (set velocity = 0).
- In the event of a fault in the passive measuring system: Only alarm display
- For hardware faults, the referencing status of the measuring system is reset:

  DB31, ... DBX60.4 / 60.5 = 0 (referenced/synchronized 1 / 2)

2.3 Monitoring of static limits

2.3.1 Overview of the limit switch monitoring

Overview of possible limit switch monitoring functions:

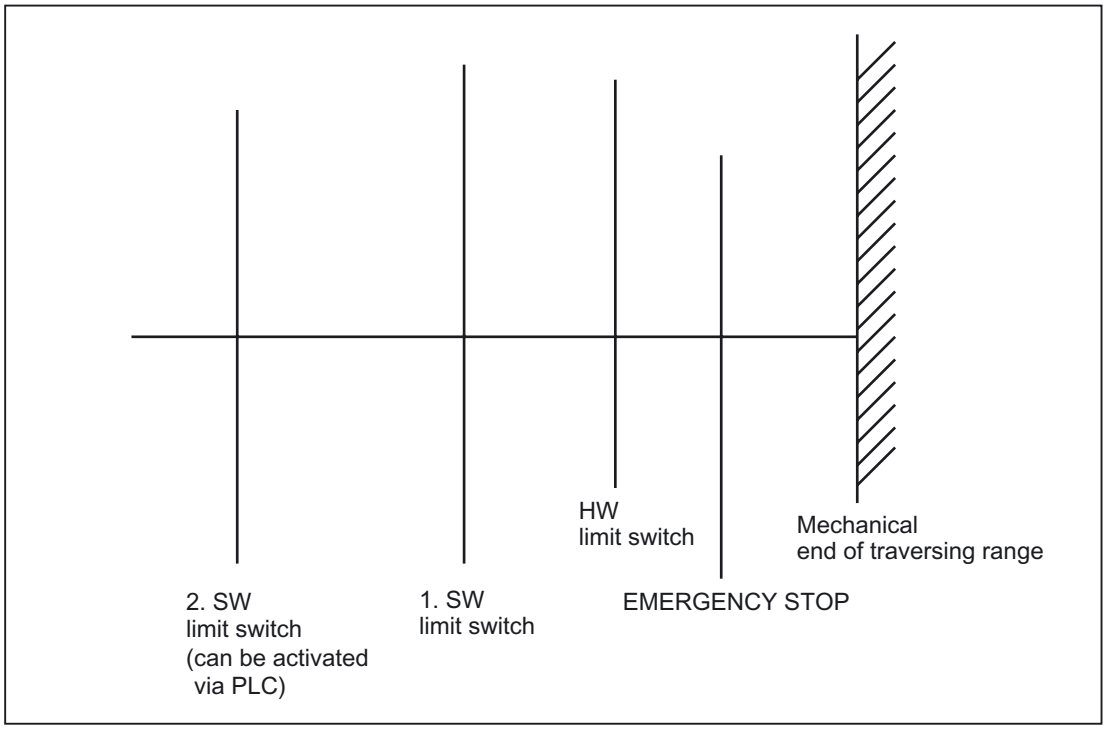


Fig. 2-8 Travel limits

2.3.2 Hardware limit switches

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 / 12.1 = 1 (hardware limit switch minus / plus)

Braking behavior

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

MD36600 $MA_BRAKE_MODE_CHOICE = <*Braking behavior*>

Braking behavior:

- 0: Braking with the configured axial acceleration
- 1: Rapid stop (set velocity = 0)

Effectivity

The hardware limit switch monitoring is active after the control 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.

2.3.3 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.

- MD36110 POS_LIMIT_PLUS (1st software limit switch plus)
- MD36110 POS_LIMIT_MINUS (1st software limit switch minus)
- MD36130 POS_LIMIT_PLUS (2nd software limit switch plus)
- MD36120 POS_LIMIT_MINUS (2nd software limit switch minus)

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)

Effectivity

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

Constraints

- 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)

Effect

Automatic operating modes (AUTOMATIC, MDI)

1. Without transformation, without overlaid movement, 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.
2. With transformation:

   Different reactions occur depending on the transformation type.

   – Behavior as under 1.

   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.
3. 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

1. JOG without transformation

   The machine axis stops at the software limit switch position.

2. 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 hardware limit switch <*Direction*>"

2.3.4 Working area limitation

Function

Working area limitation via setting data

The working area limitation serves to limit the traversing range of a geometry or special axis with respect to the basic coordinate system.

The dimensions of the working area limitation are configured via immediately effective setting data:

SD43420 $SA_WORKAREA_LIMIT_PLUS (Working area limitation plus)

SD43420 $SA_WORKAREA_LIMIT_MINUS (Working area limitation minus)

Programmable working area limitation

The working area limitation is programmable via the part program instructions `G25` / `G26` (lower / upper working area limitation). The programmed working area limitation has priority over the working area limitation set in the settings data. It overwrites the value entered in the setting date and is retained even after `NC-RESET` and program end.

Working area limitation and tool data

For traversing motions with an active tool, not only the position of the axis but also the position of the tool tip P is monitored.

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)

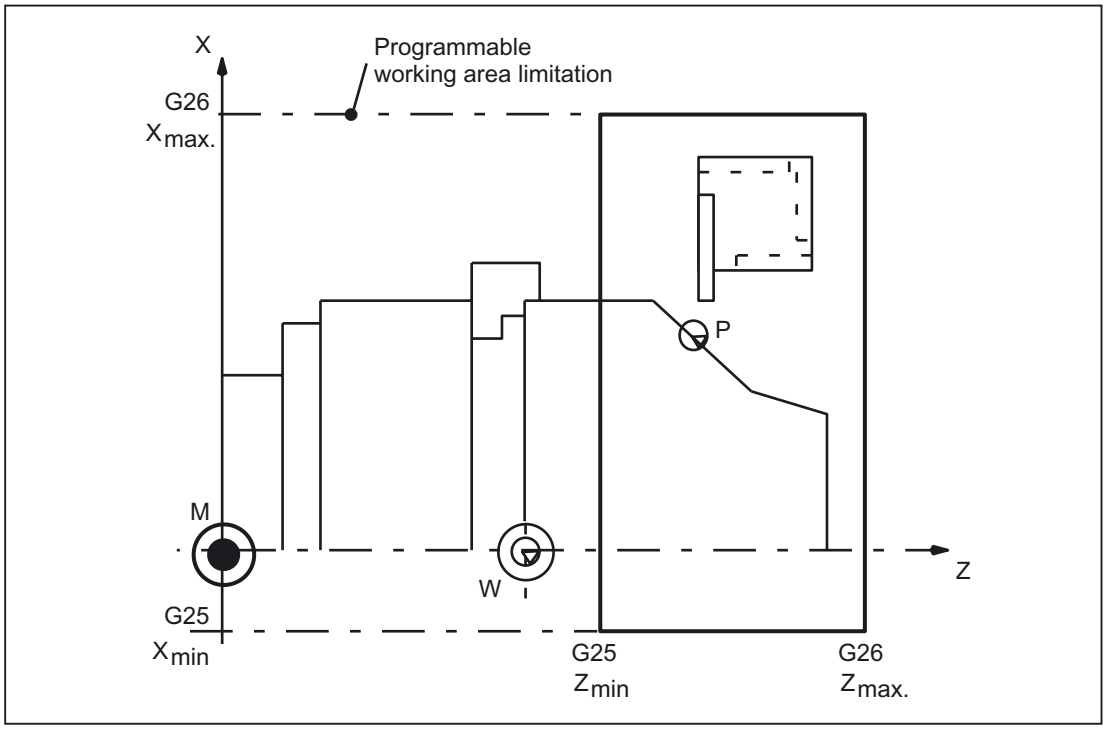


Fig. 2-9 Working area limitation

References:
/PG/ Programming Guide: Fundamentals

Effectivity

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

Activation

Working area limitation via setting data

The activation/deactivation of the working area limitation takes place direction-specifically via the immediately effective setting data:

- SD43400 $SA_WORKAREA_PLUS_ENABLE
- SD43410 $SA_WORKAREA_MINUS_ENABLE

Programmable working area limitation

The activation/deactivation of the programmable working area limitation takes place via the part program instructions:

- WALIMON (Working area limitation ON)
- WALIMOF (Working area limitation OFF)

Changing the working area limitation

- Working area limitation via setting data

  HMI user interface: Operating area "Parameter"

  – Automatic modes: Changes are possible only in the RESET state; effectivity: immediately

  – Manual operating modes: Changes are always possible, effectivity: at the start of the next traversing motion.

- Programmable working area limitation

  Reprogramming of the working area limitation via G25 / G26
  <*Axis identifier*><*Value*>

  Changes are immediately effective.

Constraints

- The working area limitation refers to the basic coordinate system.
- The working area limitation must lie within the range of the 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.

- The RESET position with regard to WALIMON / WALIMOF is configurable via:

  MD20150 $MC_GCODE_RESET_VALUES (RESET position of G groups)

- 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)

Effect

Automatic operating modes

1. With / without transformation

   The part program block with a programmed traversing motion that would lead to overrunning of the working area limitation is not started.

2. With overlaid motion

   The part program block with a programmed traversing motion that would lead to overrunning of the working area limitation is started. Axes that are traversed with overlaid motion stop at the working area limitation.

Manual operating modes

1. JOG with / without transformation

   The axis is positioned at the working area limitation and then stopped. Other machine axes participating in the traversing motion are braked. The preset path is left during this process.

General

- Activation of the working area limitation

  If the actual position of an axis after activation is located outside of the working area limitation, it is stopped with the maximum permissible acceleration.

- Overrunning of the working area limitation in JOG mode

  If the position of the working area limitation 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 moved farther:

  Alarm: "10631 Channel <Channel number> Axis <Axis identifier> is at the working area limitation <Direction>"

2.4 Protection zones

2.4.1 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 third dimension can be limited between -1 and +1.
The following four cases are possible:

- Dimension of the protection zone from -1 to +1
- Dimension of the protection zone from -1 to upper limit
- Dimension of the protection zone from lower limit to +1
- 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.

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.

Note

During machining in operating modes JOG, MDI and AUTOMATIC, the control checks whether the tool (or its protection zones) violate the protection zones of the workpiece.

2.4.2 Types of protection zone

Protection zones relating to machine or channel

- Machine-related protection zone

  A machine-related protection zone is taken into account by all channels of the NC.

- Channel-related protection zones

  A channel-related protection zone is taken into account within one channel only.

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

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

Absolute and relative protection zones

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

Below are a few examples of protection zones

- Toolrelated protection zone
- Workpiece-related protection zone
- 3-dimensional protection zone
- Relative protection zone

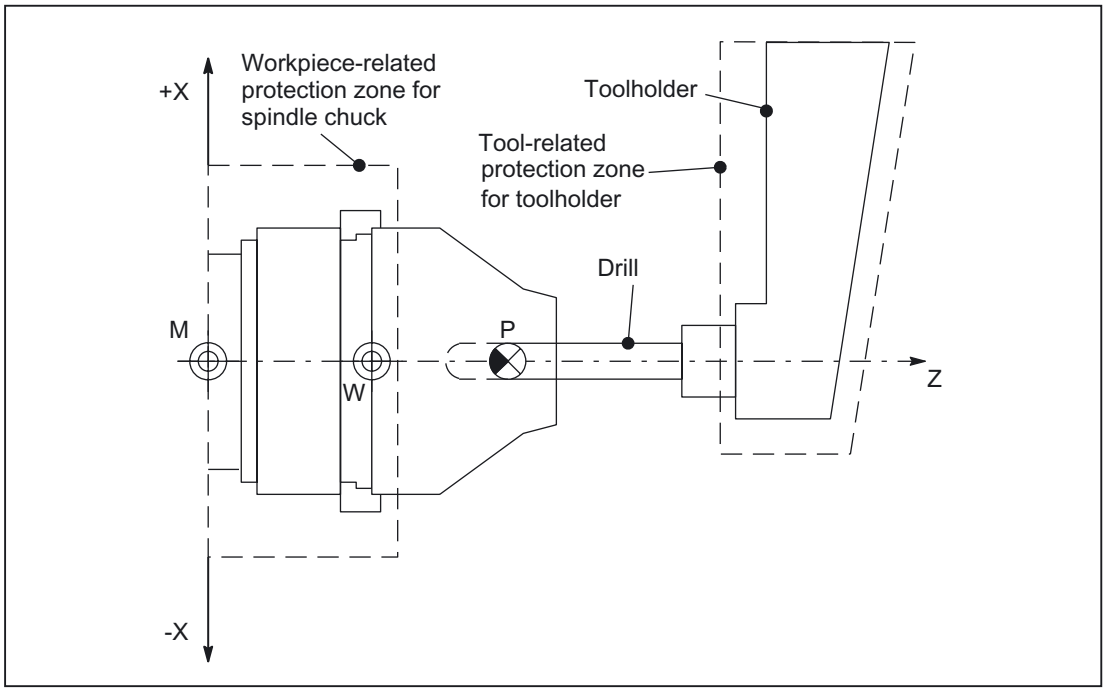


Fig. 2-10 Example of application on turning machine

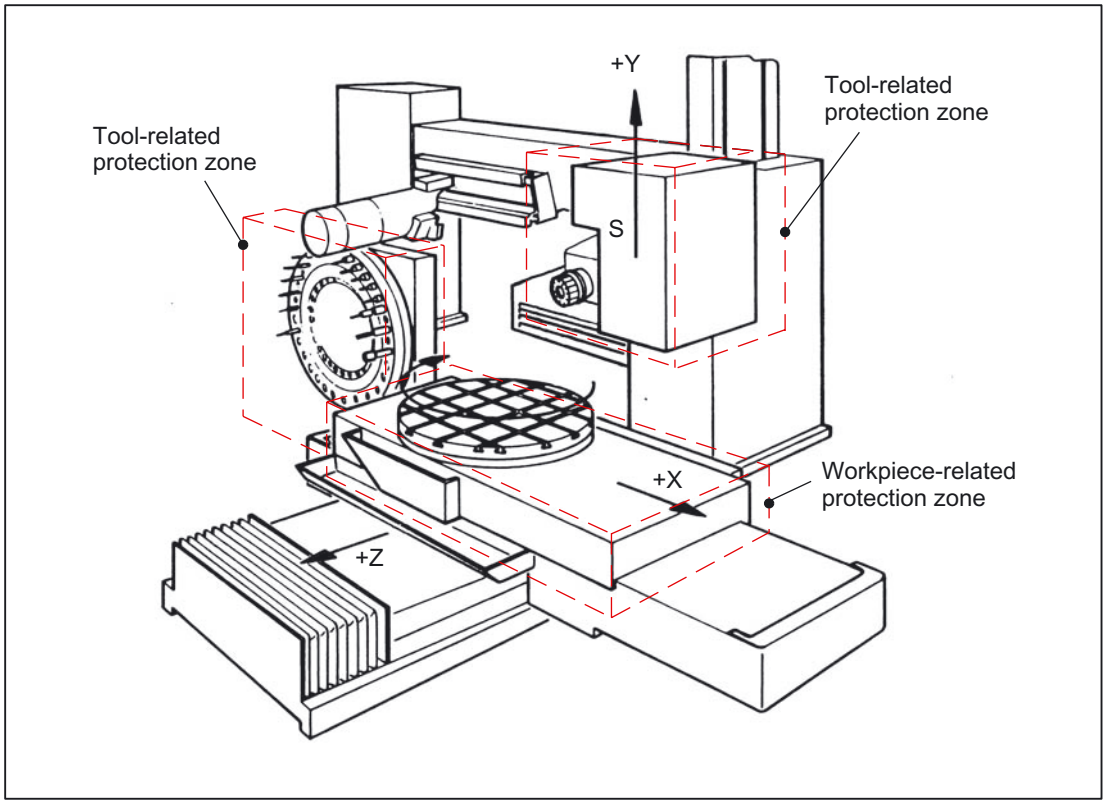


Fig. 2-11 Example of a milling machine

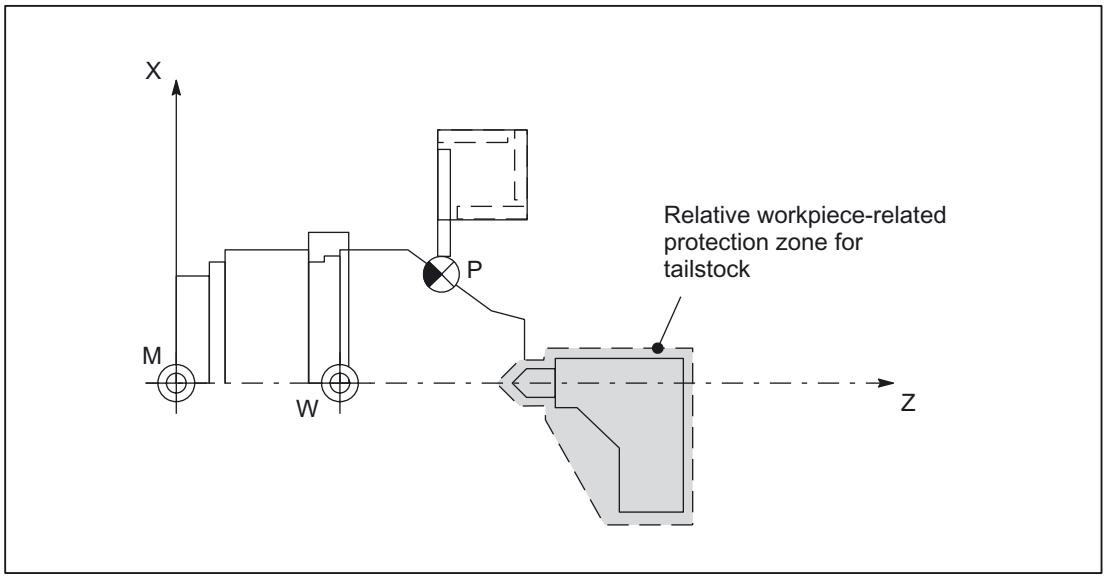


Fig. 2-12 Example of a turning machine with relative protection zone for tailstock

2.4.3 Definition via part program instruction

General

A protection zone definition must contain the following information:

- Protection zone type (workpiece- or tool-related)
- Orientation of protection zone
- Type of limitation in the third dimension
- Upper and lower limit 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 channel-specific 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 the defined protection zone
`t`	`BOOL`	Tool-related protection zone TRUE: Tool-related protection zone FALSE: Workpiece-related protection zone
`applim`	`INT`	Type of limitation in the third dimension 0: No limit 1: Limit in positive direction 2: Limit in negative 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 negative 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.

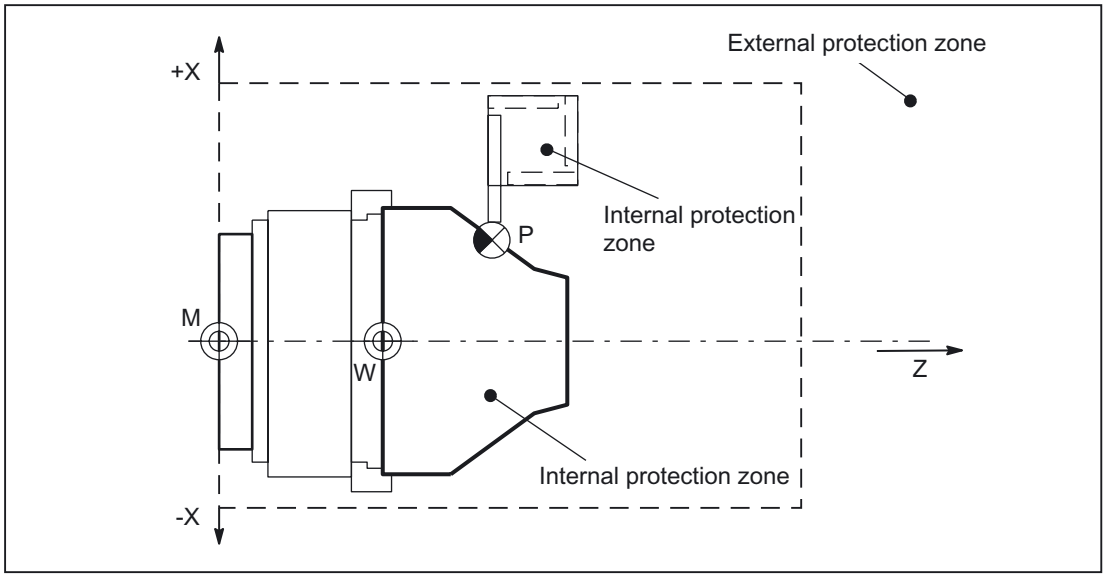


Fig. 2-13 Examples: External and internal protection zone

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.

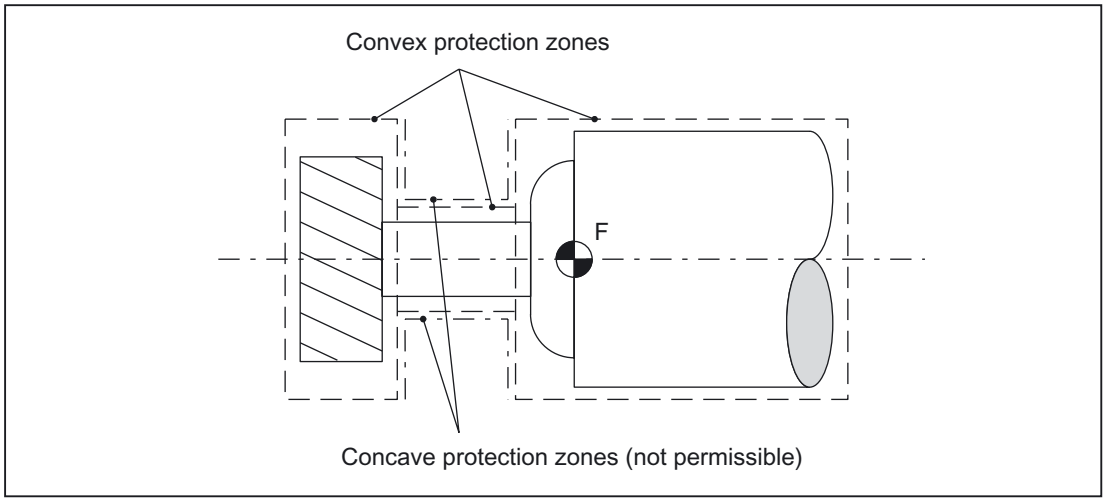


Fig. 2-14 Examples: Convex and concave tool-related protection zones

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 or channel-related protection zone is completed with the subroutine EXECUTE(n).

2.4.4 Definition via system variable

General

If the protection zones are defined with part program instructions (see Chapter: Definition by means of part program instructions), the protection zone data are 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 variable

The protection zone definitions include the 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 immediately after power up of the control and 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: 1. and 2nd geometry axis 1: 3. and 1st geometry axis 2: 2. and 3ard geometry axis
$SN_PA_LIM_3DIM[n] $SC_PA_LIM_3DIM[n]	INT	Type of limitation in the third dimension 0: No limit 1: Limit in positive direction 2: Limit in negative 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
$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

System variable	Type	Meaning
$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 elements 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 NCspecific protection zones
_N_CHAN_PRO	Data block for channelspecific protection zones in channel 1
_N_CHAN_PRO	Data block for channelspecific 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 channelspecific protection zones

2.4.5 Activation and deactivation of protection zones

General

The activation status of a protection zone is:

- Preactivated
- Preactivated with conditional stop
- Enabled
- Disabled

A protection zone is monitored for violation only when it is activated.

Activation

The activation of a protection zone can take place through:

- Part program instruction
- Automatically after the control powers up
- PLC user program

  For activation through the PLC user program, the protection zone must be first preactivated via a part program.

The preactivation, deactivation and activation of all protection zones always takes place channel-specifically. A protection zone can also be active in multiple channels simultaneously (application example: Double-slide single-spindle machine with one quill and two machining slides. Machine-related protection zones are automatically activated in all channels after the control powers up.

An activated protection zone is only taken into account after the successful referencing of all participating geometry axes.

Preactivation

Only preactivated protection zones can be activated from the PLC user program.

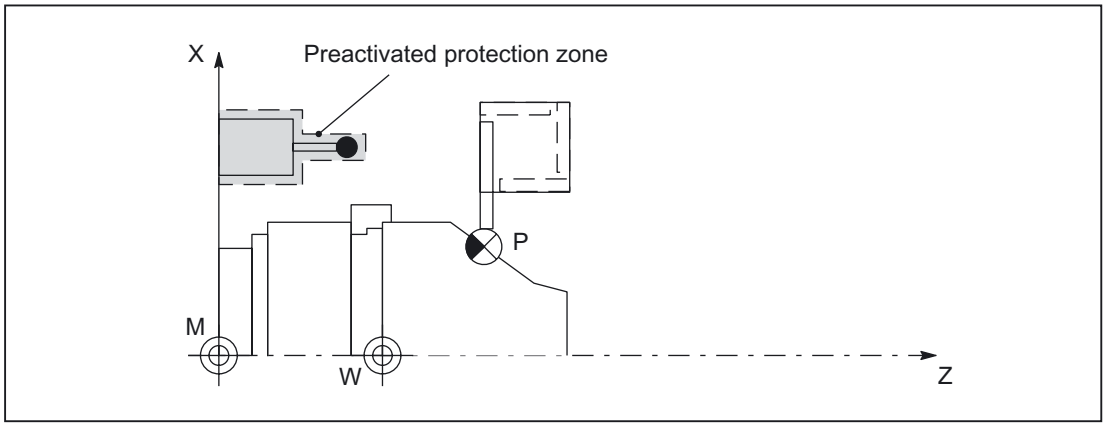


Fig. 2-15 Example: Turning machine with preactivated protection zone for a sensor

Deactivation

A protection zone can be deactivated from a part program only.

RESET response

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

Memory requirements

The memory requirements with regard to protection zones are defined with the following parameters:

- Maximum number of protection zones that can be simultaneously activated in the channel:

  MD28210 $MC_MM_NUM_PROTECT_AREA_ACTIVE

- Maximum number of contour elements that can be defined per protection zone:

  MD28212 $MC_MM_NUM_PROTECT_AREA_CONTUR

Deactivation, preactivation, activation via part program

The activation status of a channel- or machine-specific protection zone is defined by the corresponding subroutine:

- Channel-specific protection zone:

  `CPROT (n, state, xMov, yMov, zMov)`

- Machine or NC-specific protection zone:

  `NPROT (n, state, xMov, yMov, zMov)`

Parameters	Type	Description
`n`	`INT`	Number of the protection zone
`state`	`INT`	Activation status 0: Deactivated 1: Preactivated 2: Enabled
`xMov, yMov, zMov`	`REAL`	Offset values of the previously defined protection zone in the geometry axes

Offsets

During preactivation or activation of the protection zone, an offset can be entered in 0 to 3 dimensions. The offset refers to:

- Workpiece-specific protection zones: Machine zero
- Tool-specific protection zones: Tool holder reference point F

Note

A protection zone cannot be activated in a single channel with different offsets simultaneously.

Activation via PLC user program

A protection zone preactivated in the part program can be activated in the PLC user program.

Preactivated protection zones

The NC indicates the preactivated protection zones:

DB21, ... DBX272.0 - 273.1 (machine-related protection zone 1 - 10 preactivated)

DB21, ... DBX274.0 -275.1 (channel-specific protection zone 1 - 10 preactivated)

Protection zone violation

Activated and preactivated protection zones that are or would be violated by the programmed traversing motions of the current part program block if the PLC user program would activate the preactivated protection zone:

DB21, ... DBX276.0 - DBX277.1 (machine-related protection zone 1 - 10 violated)

DB21, ... DBX278.0 - DBX279.1 (channel-specific protection zone 1 - 10 violated)

Activate

The preactivated protection zones can be activated from the PLC user program:

DB21, ... DBX8.0 to DBX9.1 (activate machine-related protection zone 1 - 10)

DB21, ... DBX10.0 to DBX11.1 (activate channel-specific protection range 1 - 10)

Deactivation

Protection zones activated from the part program cannot be deactivated by the PLC user program.

Note

It follows from the rules listed above that protection zones that should be activated via the PLC user program are intended specially for this. Preactivation in the part program is only useful for these protection zones.

For protection zones that are known only in the part program and not in the PLC user program, only activation in the part program makes sense.

Automatic activation after the control powers up

The configuration for automatic activation of a protection zone after the control powers up is performed via the following system variables:

- Channel-specific protection zone:

  $SC_PA_ACTIV_IMMED[n]

- Machine or NC-specific protection zone:

  $SN_PA_ACTIV_IMMED[n]

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

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.

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 automatic modes, activated and preactivated protection zones are monitored even during program control: PROGRAM TEST.

2.4.6 Protection zone violation and temporary enabling of individual protection zones

Function

Workpiece and toolrelated protection zones that are activated or deactivated are monitored for collision. If a protection zone violation is detected, behavior in the individual operating modes is as follows.

End of temporary release
Temporary release of a protection zone is terminated after the following events:

- after `NC RESET`
- Operating modes AUTOMATIC or MDA End of block is outside the protection zone
- Manual operating modes: End of movement is outside the protection zone
- Activating a protection zone

On NC RESET all the enabled protection zones become active again. If the part program or jog mode is started again, the protection zones must be reenabled. If the current position lies within a protection zone that becomes active again after NC RESET, this protection zone must be enabled again on the first path movement.

Pre-activated protection zones
Protection zones can be pre-activated with part programs. To make them fully operative, they must also be set to the "operative" state by the PLC.

In contrast to AUTOMATIC mode, a change in the NC/PLC interface signals "Make preactivated protection zones operative" only has an effect on stationary axes in the geometry system. This means: If an inoperative protection zone is made "operative" once a motion has been started, it is not evaluated until the axes have stopped, possibly resulting in the output of an alarm.

However, the alarm 10704 "Protection zone monitoring not guaranteed" and the PLC interface signal "Protection zone monitoring not guaranteed" DB31, ...DBX39.0 are output if a preactivated protection zone is made "operative" while a traversing motion is in progress.

Deactivating tool-related protection zones
Toolrelated protection zones can be deactivated only in the part program or, if they have been preactivated, by being rendered "inoperative" by the PLC.

Geometry axis replacement
Machine data can be used to configure protection zones and working area limitations for geometry axis replacement.

- Protection zones can be activated with:

  MD10618 PROTAREA_GEOAX_CHANGE_MODE, Bit 1 = 1

- Working area limitations can be activated with:

  MD10604 WALIM_GEOAX_CHANGE_MODE, Bit 0 = 1

If functions such as protection zone or working area limitation are deactivated during geometry axis replacement, the bits should be set to zero.

For frames when changing geometry axes, see:
References:
/FB/ Description of Functions Basic Machine; Axis, Coordinate Systems, Frames (K2)

Transformation changeover
Protection zones can also be activated by changing a transformation changeover via machine data:
MD10618 PROTAREA_GEOAX_CHANGE_MODE, Bit 0 = 1.

Bit 0 = 0 deactivates this function.

Monitoring overlaid movement
Axes that have been assigned to another channel are not taken into account. The last position to be approached is taken to be the end position. It is not taken into account whether the axis has traversed after changing channels.

Behavior in the AUTOMATIC and MDI operating modes

Protection zones are not overrun in Automatic modes:

- If the movement in a block is from outside into the protection zone (N30), deceleration is executed toward the end of the previous block (N20) and the movement is stopped.

  – If the protection zone is preactivated but not activated by the PLC, machining is continued (case 1).

  – If the protection zone is activated or preactivated and activated by the PLC, machining is stopped (case 2).

- If the starting point of the block is inside the protection zone, the movement is not started.

If the protection zone is violated, alarm
10700 "NCK protection zone violated in v or MDI"
or
10701 "Channel-specific protection zone violated in Automatic or MDI"
is triggered for the workpiece-related protection zone.

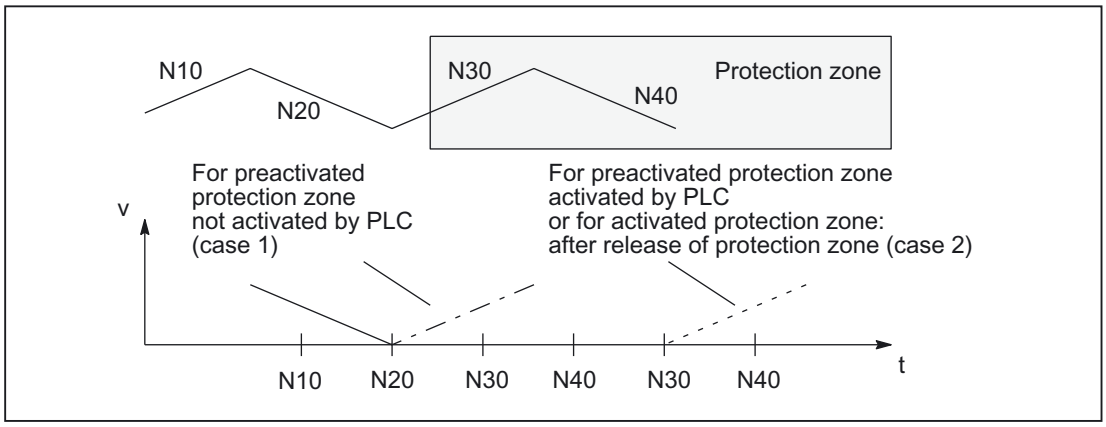


Fig. 2-16 Behavior of the path velocity when entering a protection zone

Overlaying several axis motions

Overlaid motions of external ZO (zero offsets) or DRF are taken into account if they are executed at a sufficiently early point in time.

If an overlaid motion occurs while a protection zone is active or operative, an alarm is output as a warning. This alarm has no effect on the machining operation and resets itself if the transferred motion has been fully taken into account. The PLC interface signal "Protection zone monitoring not guaranteed" DB31, ...DBX39.0 is set at the same time as alarm 10704.

Enabling of workpiece-related protection zones

When a workpiece-related protection zone has been violated, the operator can enable it temporarily with NC start in the AUTOMATIC and JOG modes so that it can be traversed. This clears the alarm and travels into the protection zone in the AUTOMATIC and MDI operating modes.

Only workpiecerelated protection zones can be enabled temporarily with NC start and traversed by all toolrelated protection zones including the programmed path.

If on NC-START the preactivated tool or workpiecerelated protection zone is deactivated by the PLC after the alarm, machining is continued without the protection zone being enabled temporarily.

If a fully operative, preactivated protection zone causes a machining interruption and the output of an alarm owing to protection zone violation, machining can be resumed on NC start if the PLC makes the zone inoperative again.

If enabling of a protection zone is to be safeguarded better than with a simple NC start, NC start must be disabled or made dependent on other conditions in the PLC user program when this alarm is triggered.

If the user does not want to permit overrunning the protection zone, he can terminate the traversing movement with NC RESET.

If several protection zones are violated at the same time by the movement, acknowledgment is required for each of these protection zones. With NC start the individual protection zones can then be enabled one after the other.

Application for temporary enabling:

Drilling a turned part: The drill is allowed to enter the protection zone of the spindle chuck.

Monitoring of overlaid motion

On preparation of the NC blocks, part of the offsets of geometry axes resulting from the overlaid motions are taken into account.

If further offsets occur that could not be taken into account on preparation of the blocks, the channelspecific PLC interface signal "Protection zone monitoring not guaranteed" DB31, ...DBX39.1 is set. This signal is set while offsets are active that cannot be taken into account. The signal can be set and reset within a block.

Simultaneously with the PLC interface signal, a selfcanceling alarm 10704 "Protection zone monitoring is not guaranteed" is output.

The following overlaid motions of geometry axes are taken into account in the preparation of blocks:

1. DRF offsets
2. Work offsets external
3. Fine tool offsets
4. Rapid retraction
5. Offsets generated by compile cycles
6. Oscillation
7. Concurrent positioning axes
8. Positioning axes

The alarm is canceled or the PLC interface signal reset when the offsets from the overlaid motions are taken into account again or when the offsets are reduced to zero again.

Note

The end position for positioning axes is taken to be a position in the whole block. This means that the alarm 10704 "Protection zones not guaranteed" is output when the positioning axis starts to move. The overlaid motions themselves are not limited, nor is there any intervention in processing of the program.

Behavior in JOG mode

Overlaying several axis movements

In JOG mode traversing can be performed simultaneously in several geometry axes even when protection zones are active. However, safe monitoring of protection zones can then no longer be guaranteed. This is indicated as follows:

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

The traversing range of the geometry axes is limited in all directions by the protection zones with the same effect as they had at the start point.

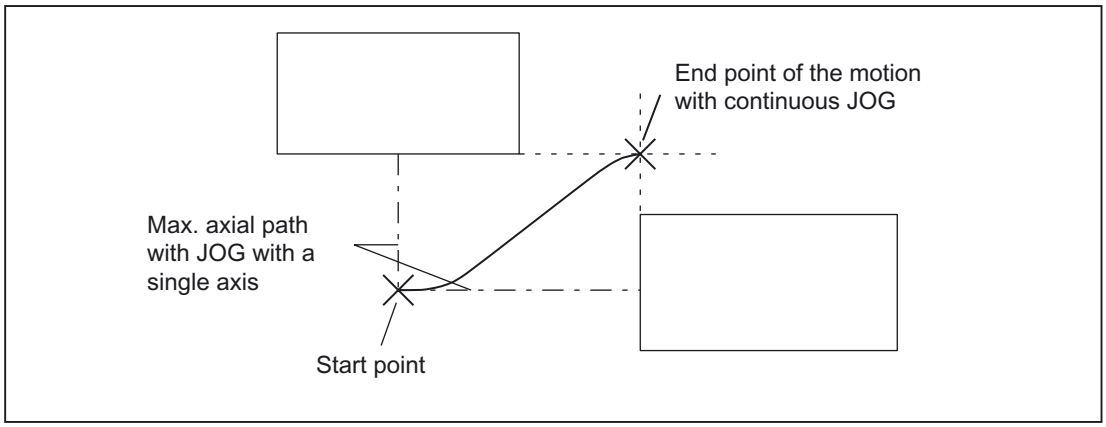


Fig. 2-17 Motion boundaries of axes

When the geometry axes have completed their movements (end of interpolation), the alarm is automatically reset and the final position checked to see whether it is within one or several protection zones.

There are three possible situations in this case:

1. If the position is outside all active protection zones, the next traversing motion can be started normally. The appropriate PLC interface signals "Machinespecific or channel-specific protection zone violated" are set for the protection zones that are enabled or just preactivated, but not yet operative.

2. If the position is within an active protection zone, the alarm "Protection zone violated in JOG" is generated, thereby disabling any traversing motions. The appropriate PLC interface signals "Machinespecific or channelspecific protection zone violated" are also set.

   The alarm is reset by:

   – Temporary enabling of the affected protection zones

   – Deactivation of the relevant protection zones if they are preactivated

   – Deactivation of the protection zone in MDI

3. If the position is on the protection zone limitation (position is still valid), no alarm is generated.

Note

While any one axis in the geometry system is still oscillating, the status "Motions of axes in geometry system completed" cannot be reached.

The warning remains active, the other axes in the geometry system can continue to traverse.

The alarm "Protection zone reached in JOG" is not output if the motion of the first axis to be started is terminated by the limitation determined prior to the motion.

Monitoring of (pre)activated protection zones
(Pre-)activated protection zones are also monitored in manual modes (JOG, INC, handwheel).

Limitations of the traversing movement of an axis
Axis motions are limited in JOG mode by means of software limit switches or working area limitation. The protection zones are an additional limiting element on the traversing motion of the geometry axes.

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 one 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.

Enabling workpiece-related protection zones
When a workpiecerelated protection zone has been violated, the operator can also enable it temporarily in JOG mode so that it can be crossed. This resets the alarm and the motion is started in the manual operating modes after a new travel command.

Temporary release of protection zones

Protection zones can be enabled in JOG mode when:

1. the current position is within a protection zone (alarm active)
2. a motion is to be started on the protection zone limit (alarm active)

A protection zone is enabled when:

- a positive signal edge arrives at the PLC interface "Temporary enabling of protection zones" (this enable resets the active alarm).
- if the axis then starts to move again into the same protection zone.

Start of the motion causes:

- the protection zone to be enabled
- the appropriate PLC interface signals "Machinespecific or channelspecific protection zone violated" to be set.
- the axis to start moving.

The enabling signal is canceled if a motion is started that does not lead into the enabled protection zone.

If the current position is located in other active protection zones or the limit for other protection zones must be crossed with the motion that has been started, then alarms 10702, 10703 or 10706, 10707 are output. The PLC interface signal "Temporary enabling of protection zones" can be set again to enable the protection zone for which an alarm is output.

The enabling signals for the individual protection zones are still valid on switchover to operating modes AUTOMATIC or MDA, and vice versa, the enabling signals of protection zones that were output in AUTOMATIC and MDA remain valid.

If the end position is located outside the relevant protection zone the next time the axes in the geometry system stop:

- the enabling signals of the individual protection zones are canceled
- the appropriate PLC interface signal "Machinespecific or channelspecific protection zone violated" is reset.

2.4.7 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 exchange

If an axis is not active in a channel because of an axis replacement, 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).

Supplementary Conditions

3

3.1 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

  The corresponding machine data are described in:
  References:
  /FB/ Description of Functions Basic Machine; Velocities, Set/Actual Value Systems, Control (G2)

Only drives with analog speed setpoint interface

- MD32260 $MA_RATED_VELO (Nominal motor speed)
- MD32250 $MA_RATED_OUTVAL (Nominal output voltage)

Examples

4

4.1 Definition and activation of protection zones

Requirement

The following internal protection zones are to be defined for a turning machine:

- One machine- and workpiecerelated protection zone for the spindle chuck, without limitation in the third dimension
- One channelspecific protection zone for the workpiece, without limitation in the third dimension
- One channelspecific, toolrelated 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 100mm in the Z axis in the positive direction.

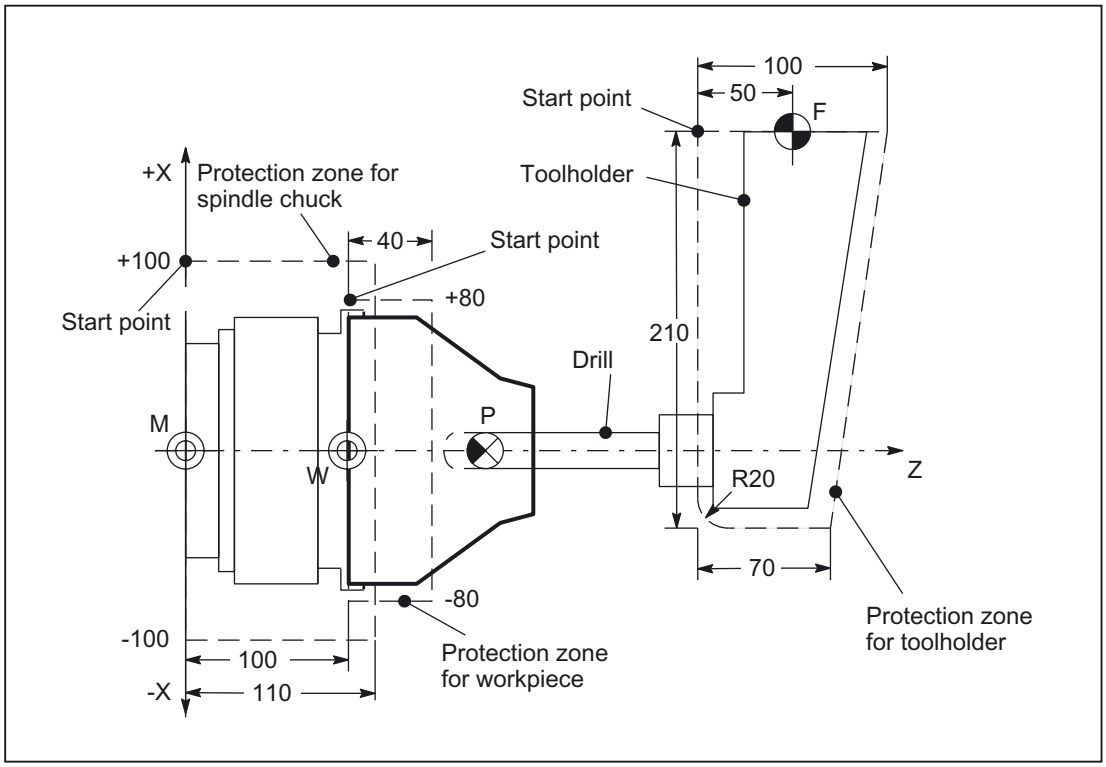


Fig. 4-1 Example of protection zones on a lathe

Protection zone definition in the part program

Table 4-1 Part program excerpt for protection zone definition:

```
DEF INT AB
G18                                    ; Definition of the working plane
NPROTDEF(0,FALSE,0,0,0)
NPROTDEF(1,FALSE,0,0,0)                ; Definition beginning: Protection zone
                                         for spindle chuck
G01 X100 Z0                            ; Contour description: 1. Contour element
G01 X-100 Z0                           ; Contour description: 2. Contour element
G01 X-100 Z110                         ; Contour description: 3. Contour element
G01 X100 Z110                          ; Contour description: 4. Contour element
G01 X100 Z0                            ; Contour description: 5. Contour element
EXECUTE(AB)                            ; End of definition: Protection zone for
                                         spindle chuck
CPROTDEF(1,FALSE,0,0,0)                ; Definition beginning: Protection zone
                                         for workpiece
G01 X80 Z0                             ; Contour description: 1. Contour element
G01 X-80 Z0                            ; Contour description: 2. Contour element
G01 X-80 Z40                           ; Contour description: 3. Contour element
G01 X80 Z40                            ; Contour description: 4. Contour element
G01 X80 Z0                             ; Contour description: 5. Contour element
```

```
EXECUTE(AB)                          ; End of definition: Protection zone for
                                       workpiece
CPROTDEF(2,TRUE,0,0,0)               ; Definition beginning: Protection zone
                                       for tool holder
G01 X0 Z-50                          ; Contour description: 1. Contour element
G01 X-190 Z-50                       ; Contour description: 2. Contour element
G03 X-210 Z-30 I-20                  ; Contour description: 3. Contour element
G01 X-210 Z20                        ; Contour description: 4. Contour element
G01 X0 Z50                           ; Contour description: 5. Contour element
G01 X0 Z-50                          ; Contour description: 6. Contour element
EXECUTE(AB)                          ; End of definition: Protection zone for
                                       tool holder
```

Protection zone definition with system variables

Table 4-2 Protection zone: Spindle chuck

System variable	Value	Comment
$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= 3. 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 4-3 Protection zone: Workpiece and tool holder

System variable	Value	Comment
$SN_PA_ACTIV_IMMED[0]	0	; Protection zone for workpiece not immediately active
$SN_PA_ACTIV_IMMED[1]	0	; Protection zone for tool holder 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= 3. and first geometry axis ; Protection zone for workpiece
$SC_PA_ORI[1]	1	; Orientation of the protection zone: 1= 3. and first geometry axis ; Protection zone for tool holder
$SC_PA_LIM_3DIM[0]	0	; Type of limitation in the third dimension: 0 = no limitation ; Protection zone for workpiece tool holder 0

$SC_PA_LIM_3DIM[1]	0	; Type of limitation in the third dimension: 0 = no limitation ; Protection zone for tool holder
$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 tool holder
$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 tool holder
$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 tool holder 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 tool holder, contour element 0
$SN_PA_CONT_TYP[1, 1]	3	; Contour type[i] : 3 = G3 f. Circuit element, counter-clockwise, ; Protection zone for tool holder, contour element 1
$SN_PA_CONT_TYP[1, 2]	1	; Contour type[i] : 1 = G1 for even, ; Protection zone for tool holder, contour element 2
$SN_PA_CONT_TYP[1, 3]	1	; Contour type[i] : 1 = G1 for even, ; Protection zone for tool holder, contour element 3
$SN_PA_CONT_TYP[1, 4]	1	; Contour type[i] : 1 = G1 for even, ; Protection zone for tool holder, contour element 4
$SN_PA_CONT_TYP[1, 5]	0	; Contour type[i] : 0 = not defined, ; Protection zone for tool holder, contour element 5
$SN_PA_CONT_TYP[1, 6]	0	; Contour type[i] : 0 = not defined, ; Protection zone for tool holder, contour element 6

$SN_PA_CONT_TYP[1, 7]	0	; Contour type[i] : 0 = not defined, ; Protection zone for tool holder, contour element 7
$SN_PA_CONT_TYP[1, 8]	0	; Contour type[i] : 0 = not defined, ; Protection zone for tool holder, contour element 8
$SN_PA_CONT_TYP[1, 9]	0	; Contour type[i] : 0 = not defined, ; Protection zone for tool holder, 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 tool holder, contour element 0
$SN_PA_CONT_ORD[1, 1]	-210	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 1
$SN_PA_CONT_ORD[1, 2]	-210	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 2
$SN_PA_CONT_ORD[1, 3]	0	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 3
$SN_PA_CONT_ORD[1, 4]	0	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 4
$SN_PA_CONT_ORD[1, 5]	0	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 5
$SN_PA_CONT_ORD[1, 6]	0	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 6
$SN_PA_CONT_ORD[1, 7]	0	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 7
$SN_PA_CONT_ORD[1, 8]	0	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 8
$SN_PA_CONT_ORD[1, 9]	0	; Endpoint of contour[i], ordinate value ; Protection zone for tool holder, 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 tool holder, contour element 0
$SN_PA_CONT_ABS[1, 1]	-30	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 1
$SN_PA_CONT_ABS[1, 2]	20	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 2
$SN_PA_CONT_ABS[1, 3]	50	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 3
$SN_PA_CONT_ABS[1, 4]	-50	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 4
$SN_PA_CONT_ABS[1, 5]	0	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 5
$SN_PA_CONT_ABS[1, 6]	0	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 6
$SN_PA_CONT_ABS[1, 7]	0	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 7
$SN_PA_CONT_ABS[1, 8]	0	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 8
$SN_PA_CONT_ABS[1, 9]	0	; Endpoint of contour[i], abscissa value ; Protection zone for tool holder, 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 tool holder, contour element 9
$SN_PA_CENT_ORD[1.0]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 0
$SN_PA_CENT_ORD[1.1]	-190	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 1
$SN_PA_CENT_ORD[1.2]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 2
$SN_PA_CENT_ORD[1.3]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 3
$SN_PA_CENT_ORD[1.4]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 4
$SN_PA_CENT_ORD[1.5]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 5
$SN_PA_CENT_ORD[1.6]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 6
$SN_PA_CENT_ORD[1.7]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 7
$SN_PA_CENT_ORD[1.8]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, contour element 8
$SN_PA_CENT_ORD[1.9]	0	; Midpoint of contour[i], ordinate value ; Protection zone for tool holder, 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 tool holder, contour element 0
$SN_PA_CENT_ABS[1.1]	-30	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 1
$SN_PA_CENT_ABS[1.2]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 2
$SN_PA_CENT_ABS[1.3]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 3
$SN_PA_CENT_ABS[1.4]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 4
$SN_PA_CENT_ABS[1.5]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 5
$SN_PA_CENT_ABS[1.6]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 6
$SN_PA_CENT_ABS[1.7]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 7
$SN_PA_CENT_ABS[1.8]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 8
$SN_PA_CENT_ABS[1.9]	0	; Midpoint of contour[i], abscissa value ; Protection zone for tool holder, contour element 9

Activation

Table 4-4 Part program excerpt for activating the three protection zones for spindle chuck, workpiece, and tool holder:

```
NPROT(1, 2, 0, 0, 0)        ; Protection zone: Spindle chuck
CPROT(1, 2, 0, 0, 100)      ; Protection zone: Workpiece with 100mm offset in the Z
                              axis.
CPROT(2, 2, 0, 0, 0)        ; Protection zone: Toolholder
```

Data Lists

5

5.1 Machine data

5.1.1 NC-specific machine data

Axis monitoring functions

None

Protection zones

Number	Identifier: $MN_	Description
10604	WALIM_GEOAX_CHANGE_MODE	Working area limitation during switchover of geometry axes
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

5.1.2 Channelspecific machine data

Axis monitoring functions

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES[n]	Reset G groups
21020	WORKAREA_WITH_TOOL_RADIUS	Allowance for tool radius with working area limitation

Protection zones

Number	Identifier: $MC_	Description
28200	MM_NUM_PROTECT_AREA_CHAN	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)

5.1.3 Axis/spindlespecific machine data

Axis monitoring functions

Number	Identifier: $MA_	Description
30310	ROT_IS_MODULO	Modulo conversion for rotary axis and spindle
32200	POSCTRL_GAIN [n]	Servo gain factor
32250	RATED_OUTVAL	Rated output voltage
32260	RATED_VELO	Rated motor speed
32300	MAX_AX_ACCEL	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 matching
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	Time delay 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	Software limit switch minus
36110	POS_LIMIT_PLUS	Software limit switch plus
36120	POS_LIMIT_MINUS2	Software limit switch minus
36130	POS_LIMIT_PLUS2	Software limit switch plus
36610	AX_EMERGENCY_STOP_TIME	Length of the braking ramp for error states
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

Number	Identifier: $MA_	Description
36300	ENC_FREQ_LIMIT	Encoder limit frequency
36302	ENC_FREQ_LIMIT_LOW	Encoder limit frequency resynchronization
36310	ENC_ZERO_MONITORING	Zero mark monitoring
36400	CONTOUR_TOL	Tolerance band contour monitoring
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover
36600	BRAKE_MODE_CHOICE	Deceleration behavior on hardware limit switch
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay servo enable

Protection zones

Number	Identifier: $MA_	Description
30800	WORK_AREA_CHECK_TYPE	Type of checking of working area limits

5.2 Setting data

5.2.1 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

5.3 Signals

5.3.1 Signals to channel

Axis monitoring functions

None

Protection zones

DB number	Byte.Bit	Description
21, ...	1.1	Enable protection zones
21, ...	4.7	Feed override
21, ...	6.0	Feed disable
21, ...	8.0	Activate machinerelated protection zone 1
	:	:
21, ...	8.7	Activate machinerelated protection zone 8
21, ...	9.0	Activate machinerelated protection zone 9
21, ...	9.1	Activate machinerelated protection zone 10
21, ...	10.0	Activate channelspecific protection zone 1
	:	:
21, ...	10.7	Activate channelspecific protection zone 8
21, ...	11.0	Activate channelspecific protection zone 9
21, ...	11.1	Activate channelspecific protection zone 10

5.3.2 Signals from channel

Axis monitoring functions

None

Protection zones

DB number	Byte.Bit	Description
21, ...	272.0	Machinerelated protection zone 1 preactivated
	:	:
21, ...	272.7	Machinerelated protection zone 8 preactivated
21, ...	273.0	Machinerelated protection zone 9 preactivated

DB number	Byte.Bit	Description
21, ...	273.1	Machinerelated protection zone 10 preactivated
21, ...	274.0	Channelspecific protection zone 1 preactivated
	:	:
21, ...	274.7	Channelspecific protection zone 8 preactivated
21, ...	275.0	Channelspecific protection zone 9 preactivated
21, ...	275.1	Channelspecific protection zone 10 preactivated
21, ...	276.0	Machinerelated protection zone 1 violated
	:	:
21, ...	276.7	Machinerelated protection zone 8 violated
21, ...	277.0	Machinerelated protection zone 9 violated
21, ...	277.1	Machinerelated protection zone 10 violated
21, ...	278.0	Channelspecific protection zone 1 violated
	:	:
21, ...	278.7	Channelspecific protection zone 8 violated
21, ...	279.0	Channelspecific protection zone 9 violated
21, ...	279.1	Channelspecific protection zone 10 violated

5.3.3 Signals to axis

Axis monitoring functions

DB number	Byte.Bit	Description
31, ...	1.4	Follow up operation
31, ...	1.5 / 1.6	Position measuring system 1 / 2
31, ...	2.1	Controller enable
31, ...	2.3	Clamping in progress
31, ...	3.6	Velocity/spindle speed limitation
31, ...	4.3	Feed stop
31, ...	12.0 / 12.1	Hardware limit switch minus/Hardware limit switch plus
31, ...	12.2 / 12.3	Software limit switch minus / 2nd software limit switch plus
31, ...	60.2 / 60.3	Encoder limit frequency exceeded 1 / 2
31, ...	60.4 / 60.5	Referenced/synchronized 1 / 2
31, ...	64.6 / 64.7	Traverse command minus / plus

Protection zones

None

Index

A

Activation of protection zones
  Example, 4-10
Axis clamping
  Operations, optimized, 2-7
Axis monitoring functions, 1-1
  Actual velocity, 2-15
  Axis/spindlespecific machine data, 5-2
  Axis/spindlespecific setting data, 5-3
  Channelspecific machine data, 5-1
  Constraints, 3-1
  Contour, 2-1
  Following error, 2-2
  Speed setpoint, 2-14
  Zero speed, 2-5

C

Contamination signal, 2-21
Contour error, 2-1
Coordinate system, 2-29

E

Encoder monitoring functions, 2-16
  Encoder frequency, 2-16
  Zero marks, 2-18

H

Hardware faults, 2-21
Hardware limit switches, 2-22

I

Interface signals|Table, 5-4

L

Limit switch monitoring, 2-22

M

MD10604, 2-43
MD10618, 2-43
MD18190, 2-31
MD20150, 2-27
MD21020, 2-26
MD28200, 2-31
MD28210, 2-40
MD28212, 2-40
MD30310, 2-24, 2-27
MD31030, 3-1
MD31050, 3-1
MD31060, 3-1
MD31070, 3-1
MD31080, 3-1
MD32000, 2-2
MD32200, 2-2
MD32200, 2-4
MD32250, 3-1
MD32260, 3-1
MD32300, 2-2
MD32610, 2-2
MD32800, 2-2
MD32810, 2-2, 3-1
MD36000, 2-6
MD36010, 2-3, 2-4, 2-6
MD36012, 2-6
MD36020, 2-3, 2-4
MD36030, 2-6
MD36040, 2-5
MD36050, 2-7, 2-13
MD36052, 2-7, 2-9, 2-10, 2-11, 2-12
MD36060, 2-5
MD36100, 2-23
MD36110, 2-23
MD36120, 2-23
MD36130, 2-23
MD36200, 2-15
MD36210, 2-14
MD36220, 2-15
MD36300, 2-16
MD36302, 2-18
MD36310, 2-18

MD36400, 2-2
MD36600, 2-23
MD36610, 2-3, 2-5, 2-6, 2-7, 2-15, 2-16, 2-17, 2-19, 2-21
Monitoring of static limits, 2-22
Motion monitoring functions, 2-1

O

Orientation, 2-30

P

Protection zone
  Activate, 2-39
  Deactivation, 2-39
  Definition, 2-33, 2-36
  Enable, 2-42
  Restrictions, 2-48
Protection zone definition
  Example, 4-1
Protection zone definition with system variables, 2-37
Protection zone violation, 2-42
Protection zones, 1-1, 2-29
  Channelspecific machine data, 5-2
  Data storage, 2-38
  General machine data, 5-1

S

SD43400, 2-26
SD43410, 2-26
SD43420, 2-25
SD43430, 2-25
Servo gain factor, 2-1
Signal distortion, 2-1
Software limit switch, 2-23
Spindle speed
  Control, 2-17

W

Working area limitation, 2-25

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Continuous-Path Mode, Exact Stop, LookAhead (B1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 General 2-1
2.1.1 Parameterization of the reset response 2-1
2.1.2 Block change and positioning axes 2-1
2.1.3 Block change delay 2-1

2.2 Exact stop 2-2

2.3 Continuous-path mode 2-6
2.3.1 General 2-6
2.3.2 Velocity reduction according to overload factor 2-8
2.3.3 Rounding according to path criterion 2-11
2.3.4 Rounding with maximum possible dynamic response on each axis 2-19
2.3.5 Smoothing the path velocity 2-21
2.3.6 Dynamic response adaptation 2-25
2.3.7 Technology G group 2-33

2.4 LookAhead 2-36

2.5 NC block compressor COMPON, COMPCURV, -CAD 2-41

3 Supplementary Conditions 3-1

3.1 Rounding and repositioning (REPOS) 3-1

3.2 Smoothing the path velocity 3-2

4 Examples 4-1

4.1 Example of jerk limitation on the path 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 General machine data 5-1
5.1.2 Channel-specific machine data 5-1
5.1.3 Axis/spindle-specific machine data 5-2

5.2 Setting data 5-2
5.2.1 Channel-specific setting data 5-2

5.3 Signals 5-3
5.3.1 Signals from channel 5-3
5.3.2 Signals to axis/spindle 5-3

Index Index-1

Brief Description

1

1.1 Brief description

Continuous-path mode

In continuous-path traversing 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.

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 in accordance with the selected exact stop criterion.

Exact stop
Exact stop is a state of a machine axis that refers to the position difference relative to its setpoint position at the end of the traversing motion. The machine axis reaches the "exact stop" state as soon as its following error is less than the specified position difference (exact stop limit).

Look Ahead

"Look ahead" is a function for optimizing 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 Look ahead, the NC only takes into consideration the block immediately following the current block to determine 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 an assignable number of blocks following the current block, a much higher path velocity can be attained under certain circumstances because the NC now has more 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.

Dynamic response adaptation

In addition to the "Smoothing the path velocity" function, "Dynamic response adaptation" is another function for avoiding high-frequency excitations of machine resonances while optimizing the dynamic path response. 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.

NC block compressor

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:

- Fewer part program blocks to describe the workpiece contour
- Higher maximum path velocities

Detailed Description 2

2.1 General

2.1.1 Parameterization of the reset response

The initial setting assigned to each channel is activated by Reset:
MD20150 $MC_GCODE_RESET_VALUES (reset of G-groups)

The initial setting can be specified for exact stop and continuous-path modes and exact stop criterion.

Detailed descriptions of initial settings can be found in:
References:
/FB1/ Description of Functions, Basic Machine; Mode Group, Channel, Program Operation, Reset Response (K1)

2.1.2 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. For a detailed description of positioning axes, refer to:

References:
/FB2/ Description of Functions, Extended Functions; Positioning Axes (P2)

2.1.3 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:

- Missing auxiliary function acknowledgement by the PLC
- Non-existent following blocks
- Active function: "Empty buffer"

Consequences
If a block change cannot be executed in continuous-path mode, all axes programmed in this part program block (except modal special axes) are stopped. In this case, contour errors do not occur. The stopping of the path axes during machining can cause undercuts on the workpiece surface.

2.2 Exact stop

Exact stop or exact stop mode

In contrast to continuous-path mode, in exact stop or exact stop mode, all path axes and non-modal special axes involved in the traversing motion are decelerated until they reach a standstill at the end of each block. 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:

- All path axes and non-modal special axes involved in the traversing motion are decelerated at the end of the block until they reach a standstill.
- 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.

Use

Exact stop mode should always be used when the programmed contour must be executed exactly.

Activation of exact stop mode

Exact stop mode is activated in the part program by programming the following G-functions:

- `G60` - Exact stop on, modal
- `G09` - Exact stop on, non-modal

Exact stop criteria: "Exact stop coarse" and "Exact stop fine"

The two exact stop criteria "Exact stop coarse" and "Exact stop fine" are used to specify the applicable tolerance window for reaching the "Exact stop" state of a machine.

State of the machine axis
The state of a machine axis that refers to the position difference relative its setpoint position at the end of a traversing motion is also designated as 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).

Parameter assignment
The tolerance windows of the exact stop criteria are specified with the following machine data:

- MD36010 STOP_LIMIT_FINE (exact stop fine)
- MD36000 STOP_LIMIT_COARSE (exact stop coarse)

Note

Tolerance window

The tolerance windows of the exact stop criteria "Exact stop coarse" and "Exact stop fine" should be assigned such that the following applies: "Exact stop coarse" > "Exact stop fine"

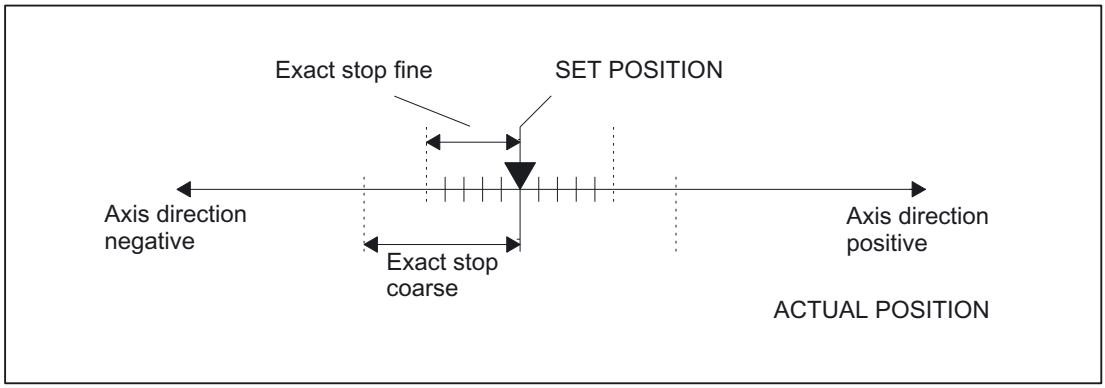


Fig. 2-1 Tolerance windows of exact stop criteria

Exact stop criterion: "Interpolator end"

With exact stop criterion "Interpolator end", the block change to the next block takes place as soon as all path axes and non-modal special axes involved in the traversing motion have reached the position programmed in the block in terms of the setpoint value. 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".

Block change depending on exact stop criteria

The figure below illustrates the block change timing in terms of the selected exact stop criterion.

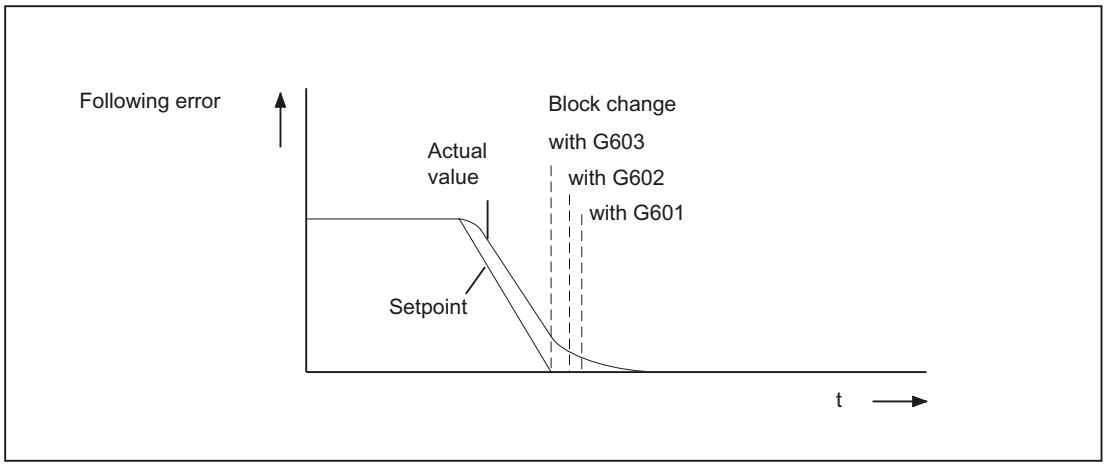


Fig. 2-2 Block change accordance to selected exact stop criterion

Activation of an exact stop criterion

An exact stop criterion is activated in the part program by programming the following G-functions:

- `G601` - Exact stop fine
- `G602` - Exact stop coarse
- `G603` - Interpolator end

Evaluation factor for exact stop criteria

A parameter-set-related evaluation of the exact stop criteria can be specified by means of the following axis-specific machine datum:
MD36012 $MA_STOP_LIMIT_FACTOR (exact stop coarse/fine factor and standstill)

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-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 made for each channel with the following machine datum:
MD20550 $MC_EXACT_POS_MODE (exact stop conditions for G00 and G01)

Coding
Each exact stop criterion is location-coded:

- Ones position E: rapid traverse
- Tens position Z: all other part program commands in the first G-group

MD20550 $MC_EXACT_POS_MODE = < *ZE* >

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 part program.

Tens position = 0:
For traversing with all other part program commands of the first G-group, the exact stop criterion programmed in the part program is active.

Assignable exact stop criterion for rapid traverse transitions in continuous path mode

The following channel-specific machine datum is used to assign the response at the block change of part program blocks before and after rapid traverse blocks:
MD20552 $MC_EXACT_POS_MODE_G0_TO_G1 = < value >

Value	Meaning
0	No additional stop at the block change
1	Stop at block change: response, as appropriate G601 (exact stop window fine)
2	Stop at block change: response same as for G602 (exact stop window coarse)
3	Stop at block change: response same as for G603 (interpolator end)
4	Like 0; in addition, the override of the next non-G00 block is taken into account with Look ahead in the G00 block during the transition from G00 to non-G00.
5	Like 0; in addition, the override of the next block is taken into account with Look ahead during the transition from G00 to non-G00 and non-G00 to G00.

2.3 Continuous-path mode

2.3.1 General

Continuous-path mode

In 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 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 operation causes:

- Contour rounding
- Shorter machining times through elimination of braking and acceleration processes that are required to comply with the exact stop criterion
- Improved 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 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 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 switchover of the acceleration profile BRISK/SOFT.

  For more information about BRISK and SOFT:
  References:
  /FB1/ Description of Functions, Basic Machine; Acceleration (B2)
- 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 have been programmed with syntax POS and have a travel time that exceeds that of the path axes. Block change occurs when the "exact stop fine" of the positioning axes is reached.
- The time taken to position a spindle programmed with syntax SPOS is longer than the travel 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

Auxiliary function output during traversal

In continuous-path mode with auxiliary-function output during motion and short traversing blocks, the path velocity is decelerated prior to PLC acknowledgment of the auxiliary functions.

The axes are decelerated to standstill respecting the acceleration limits at the end of the block. At the end of the block acknowledgment is awaited before motion can continue.

Acknowledgment during deceleration

Acknowledgment during deceleration accelerates the velocity back to the programmed path velocity.

In order to prevent this happening in continuous-path mode, machine data:
MD10110 PLC_CYCLE_TIME_AVERAGE (maximum PLC acknowledgment time)
can be used to set a time for the CNC during which the PLC will safely acknowledge the auxiliary functions for the CNC.

Acknowledgment outside of travel time

If, due to the programmed path length and the velocity of the block with auxiliary-function output, the travel time is shorter than that set in machine data:
MD10110 PLC_CYCLE_TIME_AVERAGE
LookAhead deceleration is applied to the path velocity for the block, so that the duration of the block corresponds to the time set.

If acknowledgment is not received within the time, the following prepared block cannot be processed and the axes are braked to standstill with setpoint = 0 without considering the acceleration limits.

Acknowledgment not received by end of block

If the acknowledgment is not received by the end of the block in long blocks in which the velocity has not needed to be reduced on account of the PLC acknowledgment time, the velocity is maintained until the end of the block and then reduced as described above.

Acknowledgment during braking

If the acknowledgment arrives while the axis is decelerating, the axis is not accelerated back up to the requested velocity.

MD10110

Machine data:
MD10110 PLC_CYCLE_TIME_AVERAGE
is not evaluated.

2.3.2 Velocity reduction according to overload factor

Velocity reduction according to overload factor

The function lowers the path velocity in continuous-path mode until the non-tangential block transition can be traversed in one interpolation cycle while respecting the deceleration limit and taking and overload factor into account.

With the reduction in 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. With a practically tangential block transition, the path velocity is not reduced if the permissible axial accelerations are not exceeded. In this way, very small angular changes in the contour can be overtraveled directly.

Overload factor

The overload factor restricts step changes in the machine axis velocity at block ends. So 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, which is set in machine data
MD32300 MAX_AX_ACCEL (axis acceleration),
may be exceeded for an IPO cycle.

The velocity jump is the product of the axial acceleration * (overload factor -1) * interpolator cycle.

The factor is set in machine data:
MD32210 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.

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.

Set machine data:
MD20490 IGNORE_OVL_FACTOR_FOR_ADIS
to smooth block transitions irrespective of the programmed overload factor.

Selection and deselection of velocity reduction

Continuous-path mode with velocity reduction according to overload factor can be selected modally in every NC part program block by means of program code `G64`.

Continuous-path mode `G64` can be

- Interrupted non-modally by selecting exact stop `G09`
- Deselected by selecting exact stop `G60`
- Deselected by selecting rounding `G641`

Implicit continuous-path mode

If it is not possible to insert approximate positioning blocks due to the very short block path lengths (e.g., zero-clocked blocks) in continuous-path mode with approximate positioning `G641`, the mode is switched over to continuous-path mode `G64`.

The figure below shows how the function velocity drops according to an overload factor.

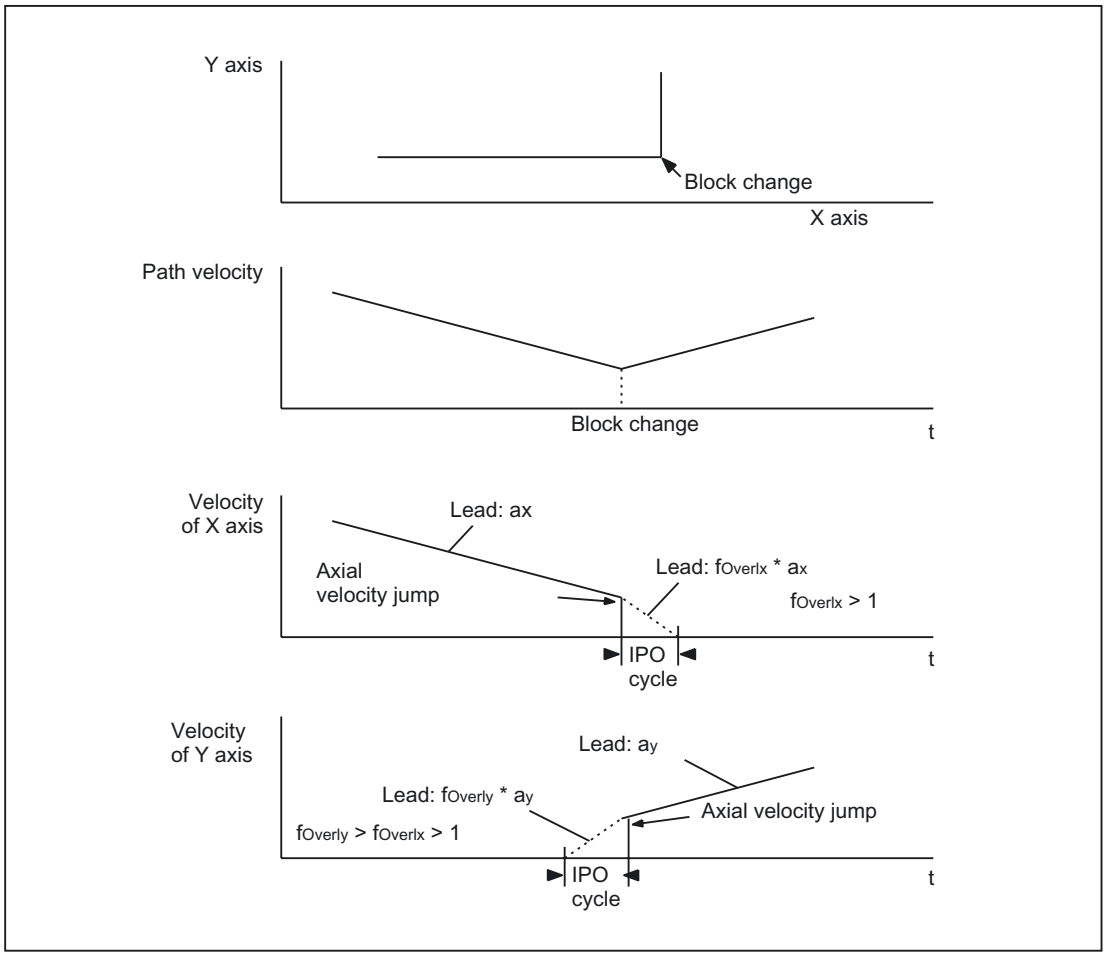


Fig. 2-3 Axial velocity change on block transition

2.3.3 Rounding according to path criterion

Blending

Rounding means that an angular block transition is changed to a tangential block transition by a local change to the programmed feedrate.

Rounding replaces the area in the vicinity of the original angular block transition (including transitions between blocks inserted by the CNC) by a continuous contour. In this case, it is not only the geometry axes that are taken into account, but all machine axes, which are traversing 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: `RND, RNDM, ASPLINE, BSPLINE, CSPLINE`.

If a rounding movement initiated by G641, G642, G643, G644 is interrupted, the corner point of the original contour will be used for subsequent repositioning, rather than the interruption point.

Rounding is initiated by 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.

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 overload factor). This means that very small knees in the contour (e.g., 0.5 degrees) can be overtraveled directly.

No intermediate rounding blocks

An intermediate rounding block is not inserted in the following situations:

1		The axis stops between the two blocks. This occurs when ...
	1	The auxiliary function output is programmed before the movement in the following block.
	2	The following block does not contain a path movement.
	3	An axis, which was previously a positioning axis traverses as a path axis for the first time in the following block.
	4	An axis, which was previously a path axis traverses as a positioning axis for the first time in the following block.
	5	The previous block moves geometry axes and the following block does not.
	6	The following block moves geometry axes and the previous block does not.
	7	Prior to thread cutting: The following block uses G33 as a preparatory function and the previous block does not.
	8	A change is made between BRISK and SOFT.
	9	Axes involved in a transformation are not completely assigned to the path movement (e.g., with oscillation, positioning axes).
2		The rounding block would slow down part program execution. This occurs when ...
	1	A rounding block is inserted between very short blocks. Since each block requires at least one interpolation cycle, the added intermediate block would double the machining time.
	2	A block transition G64 (continuous-path mode without rounding) can be traversed without speed reduction. Rounding would increase the machining time.
3		Rounding is not parameterized. This occurs when ...
	1	ADISPOS == 0 in G0 blocks. (default!)
	2	ADIS == 0 in non-G0 blocks. (default!)
	3	On transition from G0 to non-G0 or non-G0 to G0 respectively, the smaller value from ADISPOS and ADIS applies.

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		Result for
Geometry axes	Orientation axis/ synchronized axis	Rounding path
Smooth	Smooth	Defined path is traversed exactly
Smooth	Angular path	Intermediate blocks, the geometry axes follow the path exactly, all orientation/synchronized axis paths are smoothed

Rounding behavior with synchronized paths		
Angular path	Smooth	Intermediate block, the geometry axes perform rounding, all orientation/synchronized axis paths are smoothed
Angular path	Angular path	intermediate block, the geometry axes perform rounding, all orientation/synchronized axis paths are smoothed

Path criterion

The size of the rounding area can be controlled by path criteria ADIS and ADISPOS. These are the precoincidences for the block change. ADIS and ADISPOS describe the distance, which the rounding block may begin, at the earliest, before the end of the block or the distance after the end of block within which the rounding block must be terminated.

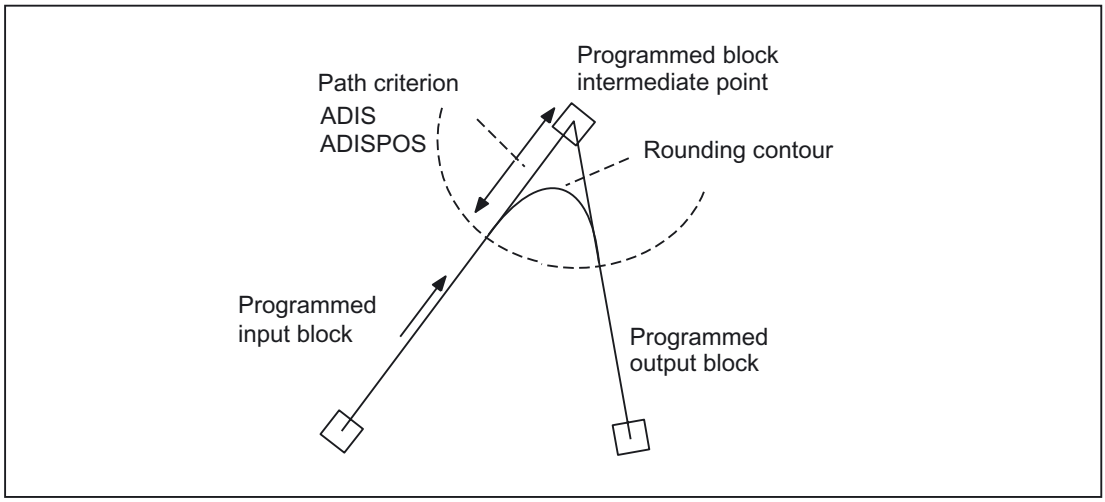


Fig. 2-4 Example for rounding an angular block transition

Acute angles produce rounding curves with a large degree of curvature and therefore cause a corresponding reduction in velocity.

Parameterization of the path criterion

- ADISPOS is programmed in the same way as ADIS, but must be used specifically for movements in rapid traverse mode (G00).
- ADIS and ADISPOS are preset in the part program.

  For example, ADIS = 0.3 indicates a path criterion with a rounding distance of 0.3 mm (ADISPOS is the same).

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 G00, the smaller rounding distance applies.
- If a very small value is used for ADIS, the control 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.

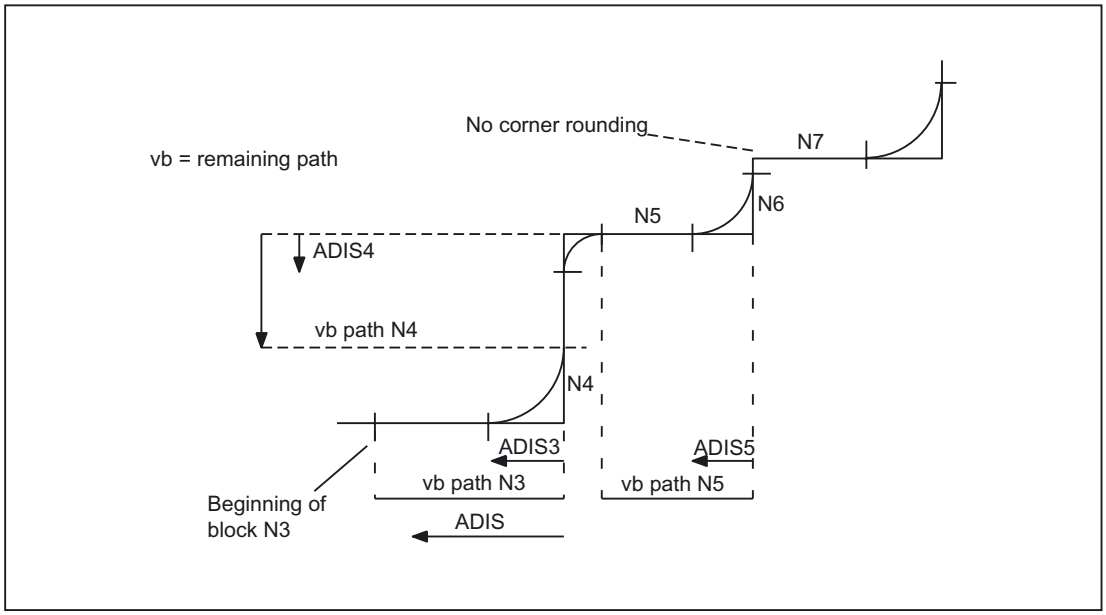


Fig. 2-5 Path with limitation of ADIS

Programming example

```
N1 G641Y50 F10 ADIS = 0.5
N2 X50
N3 X50.7
N4 Y50.7
N5 Y51.4
N6 Y51.0
N7 X52.1
```

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.

Selection and deselection of rounding blocks

Program code G641 can be inserted in any NC part program block to modally select rounding according to a path criterion. Before or on selection, the path criteria ADIS/ADISPOS must be specified.

Continuous-path mode G641 can be

- Interrupted non-modally by selecting exact stop G09
- Deselected by selecting exact stop G60
- Deselected by selecting velocity drop G64

Rounding with axial tolerances

In addition to rounding using the command G641, rounding with axial tolerances enabled modally using G642 is possible.

In this case, instead of rounding being performed within a defined ADIS range, the axial tolerances defined with machine data
MD33100 COMPRESS_POS_TOL
are used.

The mode of operation is otherwise identical to G641.

References:
/PG/ Programming Guide: Fundamentals

Extensions

G643 is not used to generate a separate rounding block, but axisspecific blockinternal rounding movements are inserted.

The expansions described below refine the response with G642 and G643 and "rounding with contour tolerance" is introduced. "

When rounding with G642 and G643, the maximum permissible deviations of each axis are normally specified.

Machine data:
MD20480 SMOOTHING_MODE
can be used to configure rounding with G642 and G643 so that instead of the axis-specific tolerances, a contour tolerance and an orientation tolerance can be specified.

In this case, the tolerance of the contour and of the orientation is set using two independent setting data, which can be programmed in the NC program. This means that different setting data can be specified for each block transition.

Contour tolerance

A contour tolerance is specified using setting data:
SD42465 SMOOTH_CONTUR_TOL

This setting data defines the maximum rounding tolerance for the contour.

Orientation tolerance

An orientation tolerance is specified using setting data:
SD42466 SMOOTH_ORI_TOL

This setting data defines the maximum rounding tolerance for the tool orientation. The data determines the maximum permissible angular deviation of the orientation of the tool. This data is only effective if an orientation transformation is active.

Possible combinations

The table below shows the effect of axis tolerances in machine data:
MD33100 COMPRESS_POS_TOL
and the above setting data for codes G642 and G643.

MD20480 SMOOTHING_MODE must be loaded with two decimal places as follows:		
Value	**G642** (setting value in ten's place of MD20480)	**G643** (setting value in unit's place of MD20480)
0	Axis-specific tolerances are used for G642. These are set using the axis-specific machine data: MD33100 COMPRESS_POS_TOL (default value).	Axis-specific tolerances are used for G643. These are set using the axis-specific machine data: MD33100 COMPRESS_POS_TOL (default value).
1	When rounding geometry axes with G642, the contour tolerance: SD42465 SMOOTH_CONTUR_TOL is used. The remaining axes are rounded using the axis-specific tolerances in machine data: MD33100 COMPRESS_POS_TOL	When rounding geometry axes with G643, the contour tolerance: SD42465 SMOOTH_CONTUR_TOL is used. The remaining axes are rounded using the axis-specific tolerances in machine data: MD33100 COMPRESS_POS_TOL
2	The orientation movement with G642 is rounded using the angle tolerance: SD42466 SMOOTH_ORI_TOL. The axis-specific tolerances in machine data: SD42466 SMOOTH_ORI_TOL are used for all other axes.	The orientation movement with G463 is rounded using the angle tolerance SD42466 SMOOTH_ORI_TOL. The axis-specific tolerances in machine data: SD42466 SMOOTH_ORI_TOL are used for all other axes.

MD20480 SMOOTHING_MODE must be loaded with two decimal places as follows:		
3	Combination of the two options 1x and 2x. i.e., the following tolerances are used for G642: SD42465 SMOOTH_CONTUR_TOL and SD42466 SMOOTH_ORI_TOL. Further axes are rounded using the axis-specific tolerance.	Combination of the two options 1x and 2x. i.e., the following tolerances are used for G643: SD42465 SMOOTH_CONTUR_TOL and SD42466 SMOOTH_ORI_TOL. Further axes are rounded using the axis-specific tolerance.
4	The rounding length programmed with ADIS= or ADISPOS= is used for G642. Any axis-specific tolerance or contour and orientation tolerance specifications are ignored.	The rounding length programmed with ADIS= or ADISPOS= is used for G643. Any axis-specific tolerance or contour and orientation tolerance specifications are ignored.

Differences between G642 - G643

With regard to their rounding behavior, commands G642 ad G643 differ as follows:

G642	G643
With G642, the rounding travel is determined based on the shortest rounding travel of all axes. This value is taken into account when generating a rounding block.	In the case of G643, the rounding travel of each axis can be different. The rounding travels are taken into account axisspecifically and block-internally.
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.

Constraints

Expansion to include contour tolerance and orientation tolerance exists only in systems with options for polynomial interpolation. The orientation-transformation option is also required for rounding orientations with angular tolerance specification.

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, MD 28530 MM_PATH_VELO_SEGMENTS must be greater than zero (> 0). If this condition is satisfied, then MD 35240 ACCEL_TYPE_DRIVE must equal FALSE for all axes.

Expansion

Rounding with G642 and G643 has been extended so that the length of the rounding distance can be specified directly instead of entering the maximum tolerances.

As with G641, the language commands ADIS = ... for G01 and ADISPOS = ... for G00 are used respectively for this purpose.

Machine data:
MD20480 SMOOTHING_MODE
determines whether G642, G643 or the maximum rounding length is specified.

The following applies:

SMOOTHING_MODE = x4	G643 with ADIS or ADISPOS
SMOOTHING_MODE = 4x	G642 with ADIS or ADISPOS

Profile for limit velocity

The hundred's place in machine data:
MD20480 SMOOTHING_MODE
can be used to control the use of a velocity profile during rounding.

Hundred's place:	
< 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

See also MD28530 MM_PATH_VELO_SEGMENTS.

2.3.4 Rounding with maximum possible dynamic response on each axis

How this type of rounding differs from existing types

Unlike existing rounding types, which are activated with G codes G641, G642 and G643, in this case, the maximum possible dynamic response on each axis takes priority.

Activation

This type of rounding is activated by G code G644. Further information needs to be programmed and/or entered in the machine data, depending on the setting.

Note

G644 is not available with an active kinematic transformation. The system switches internally to G642.

Configuration

Machine data:
MD20480 SMOOTHING_MODE
is used to configure rounding with G644.

The following options are available (the values should be entered in the thousand's place of the machine data):

Thousand's place:	
0xxx:	When rounding with G644, the maximum deviations of each axis specified with machine data: MD33100 COMPRESS_POS_TOL are used.
1xxx:	Input the maximum rounding path by programming ADIS=... or ADISPOS=... (as for G641)
2xxx:	Input the maximum possible frequencies for each axis in the rounding area with machine data: MD32440 LOOKAH_FREQUENCY. 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: The jerk is limited. BRISK: When BRISK is active, only the acceleration is limited to its maximum value. 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.

While, with 3xxx, the rounding distance results exclusively from the dynamic response data of the respective axis, in the other cases the distance to the corner at which the rounding motion begins is checked. In the first two cases, the possible rounding distance is limited to a maximum value, which results either from the defined maximum deviation or a direct programming command.

In contrast, in the case of 2xxx, the rounding distance is limited to a minimum value, which results from the maximum permissible excitation frequency and the current path velocity.

Additional limitation

Any further limitations are also effective in addition to this rounding distance limitation. The rounding distance cannot exceed half the length of the original participating blocks.

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. Consequently, a different velocity characteristic is produced for each axis, depending on whether BRISK or SOFT is active.

BRISK

With BRISK, no jerk limitation is active; the acceleration of each axis reaches its maximum value in the entire rounding area.

SOFT

With SOFT, the jerk of each axis is limited to its maximum value within the rounding area. The rounding motion thus consists of 3 general 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 (see figure below).

2.3.5 Smoothing the path velocity

Application

In some applications in mold making, especially in the case of high speed cutting, it is desirable to achieve a constant path velocity.

Response without smoothing

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.

Solution: Smoothing

In some cases, it can therefore be reasonable to sacrifice transient acceleration processes in favor of a smoother tool path velocity. The following sections describe the conditions and possible settings to avoid less effective accelerations/decelerations.

Advantages

The following improvements are possible:

- Avoidance of excitations of possible machine resonance due to continuous, transient braking and acceleration processes (in the area of less IPO cycles).
- Avoiding of constantly varying cutting rates due to acceleration which brings no significant shortening of the program running time.

Smoother path movement

If the velocity is controlled smoother and not every acceleration process is carried out, both advantages can be achieved without any undesired extended machining time.

Decision-making criteria

The control system makes a decision based on:

- Smoothing frequency for the tool path velocity (MD)
- Tolerable loss in productivity when suppressing accelerations/decelerations (MD).

Setting parameters for path smoothing

The user can define the following parameters to set the path smoothing:

- Use of MD20460 to increase machining time

  If the velocity is not accelerated, it will take longer to process this part program (see example).

- Use of MD32440 to input the resonance frequencies of the participating axes

  Only accelerations in velocity, which clearly excite machine resonances, should be removed

- Use of MD20462 to take into account the programmed feedrate for smoothing

MD20460

A percentage value corresponding to the permissible extension must be specified in machine data:
MD20460 LOOKAH_SMOOTH_FACTOR.

The percentage value defines how much longer a machining step without accelerations may be than the corresponding step when performing the 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.

MD20462

Machine data:
MD20462 LOOKAH_SMOOTH_WITH_FEED
defines that the programmed feedrate will also be taken into account in the rounding.

If the MD is set to 1 (default), the smoothing factor is observed with particular precision if the override is set to 100%.

MD32440

Machine data:
MD32440 LOOKAH_FREQUENCY
contains a smoothing frequency for LookAhead.

Acceleration and braking operations, which take place at a higher frequency than that set in this MD, are smoothed in machine data
MD20460 LOOKAH_SMOOTH_FACTOR
and
MD20465 ADAPT_PATH_DYNAMIC
on the basis of the parameter settings or the dynamic response is reduced.
The minimum value for all axes involved in the path is always calculated.

If vibration is excited in the mechanical system of this axis and if the corresponding frequency is known, this MD should be set to a value smaller than this frequency.

The required resonance frequencies, for example, can be calculated using the built-in measuring functions.
The minimum value for this MD32440 is calculated as *fPath* on the basis of the axes involved in the path.
Only accelerations, during which the initial or final velocity of this motion is returned to within time:
$t = t2-t1 = 2/fPath$
are taken into account for smoothing.

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 at this point.

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.

Activation

The smoothing of the path velocity is activated with machine data:
MD20460 LOOKAH_SMOOTH_FACTOR,
which is active with `NewConfig`.
With the default value 0, the function is deactivated.

Example

The following parameters are assumed:

MD20460 LOOKAH_SMOOTH_FACTOR = 10%

MD32440 LOOKAH_FREQUENCY[AX1] = 20Hz

MD32440 LOOKAH_FREQUENCY[AX2] = 20Hz

MD32440 LOOKAH_FREQUENCY[AX3] = 10Hz

The path involves the 3 axes X = AX1, Y = AX2, Z = AX3.
The minimum value of MD32440 LOOKAH_FREQUENCY for these 3 axes is thus 10 Hz. This means that any acceleration, which is completed within a period of t2-t1 = 2/10Hz = 200 ms, is examined. The time t2 is the time after which, following acceleration from velocity v1, the velocity returns to this velocity v1. The extending of the execution time is also only considered within this range.

A time t2-t1 in excess of 200 ms or additional program processing time t3-t2 greater than 10% (= MD20460 LOOKAH_SMOOTH_FACTOR) of t2-t1, produces the following time characteristic:

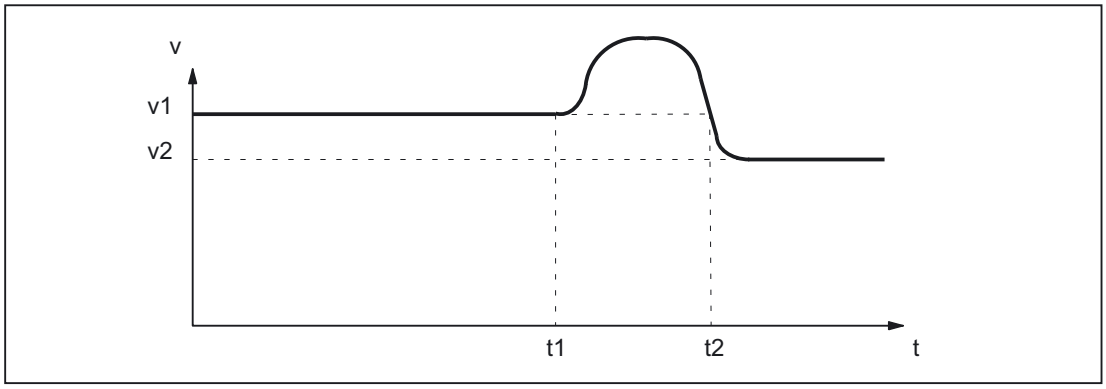


Fig. 2-6 Characteristic of time-optimum path velocity (without smoothing)

However, a time t2-t1 below 200 ms and additional program processing time t3-t2 no greater than 10% of t2-t1 produces this time characteristic:

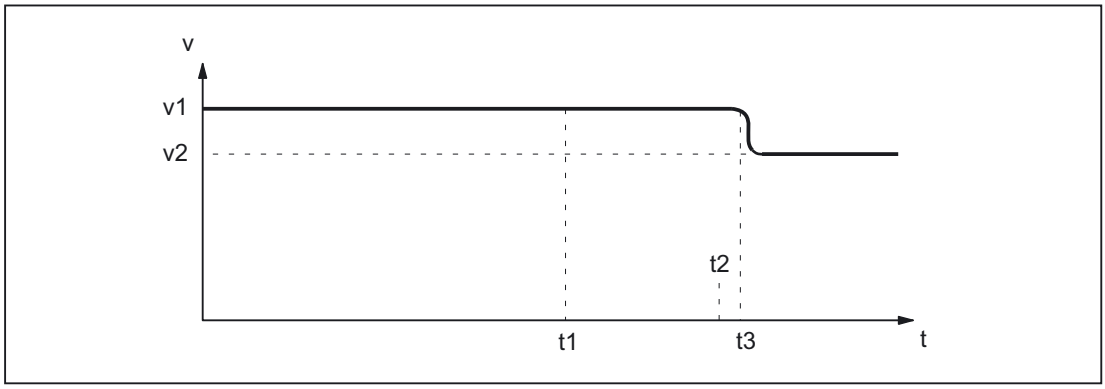


Fig. 2-7 Characteristic of the smoothed path velocity

2.3.6 Dynamic response adaptation

Key statement

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 function: "Dynamic response adaptation" therefore provides the option of adapting the dynamic response of acceleration and deceleration processes to the machine conditions by means of the following parameters:

- Path dynamic response adaptation factor
  With the adaptation factor of the path dynamic response, temporary changes in the path velocity are executed with smaller dynamic response limit values. This adaptation factor can be set for each channel for traversing motions with acceleration without (BRISK) and with (SOFT) jerk limiting:

  – Traversing motions with acceleration without jerk limiting:
  MD20465 $MC_ADAPT_PATH_DYNAMIC[0] (path dynamic response adaptation)
  The adaptation factor affects the acceleration.

  – Traversing motions with acceleration with jerk limiting:
  MD20465 $MC_ADAPT_PATH_DYNAMIC[1]
  The adaptation factor affects the jerk.

- Excitation frequency
  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 excitation frequency from which the dynamic response limiting takes effect can be specified for each axis by means of:
  MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency with Look ahead)
  During machining, the NC identifies cyclically among all axes involved in the path the minimum excitation frequency as the limiting frequency relevant for the dynamic response adaptation.

Constraints

The dynamic response adaptation considers only 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.

Activcation

The dynamic response adaptation is only active 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 dynamic response adaptation is **not** active under the following boundary conditions:

- Programmed rapid traverse (`G0`)
- Changes in the override value
- Stop requests during motion, e.g `NC-STOP, NC-RESET`
- Active function: "Velocity-related path acceleration" (`DRIVE`)

Activation

The dynamic response adaptation is activated by an adaptation factor greater than 1:
MD20465 $MC_ADAPT_PATH_DYNAMIC > 1

Automatic activation of the "Smoothing the path velocity" function

When the dynamic response adaptation is activated in continuous-path mode, the "Smoothing the path velocity" function is always activated as well. With an assigned smoothing factor of 0% (default: deactivated), a smoothing factor of 100% is used. With a smoothing factor other than 0%, the assigned value is used:
MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor with Look ahead)

Adaptations

In order to clarify the adaptation processes sketched below, please note the following basic principles:

The size of the time window is t_{adapt} = 1 / *f*.

1. The time needed to change the velocity is less than t_{adapt}:

   The acceleration rates are increased by a factor of 1 and decreased by the value written in machine data:
   MD20465 ADAPT_PATH_DYNAMIC.
   The reduction in acceleration rate increases the time taken to change the velocity.
   The following cases are possible:

   – 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.
   In spite of the lower acceleration rate, the process takes less time than t_{adapt}.
   The permissible reduction was fully utilized.

2. The time needed to change the velocity is greater than t_{adapt}:

   No dynamic response adaptation is required.

Example 1: Effect of dynamic response adaptation; acceleration mode: BRISK

Parameter assignment

Machine data

$MC_ADAPT_PATH_DYNAMIC[0] = 1.5	
$MC_LOOKAH_SMOOTH_FACTOR = 1.0	See note
$MA_LOOKAH_FREQUENCY[AX1] = 20 Hz	T_{AX1} = 1/20 Hz = 50 ms
$MA_LOOKAH_FREQUENCY[AX2] = 10 Hz	T_{AX2} = 1/10 Hz = 100 ms
$MA_LOOKAH_FREQUENCY[AX3] = 20 Hz	T_{AX3} = 1/20 Hz = 50 ms

Note
A smoothing factor other than 0% prevents the default value of 100% from being used to smooth the path velocity. To illustrate this effect, in contrast to Example 2 the smoothing of the path velocity is practically deactivated.

AX2

For path motions in which axis AX2 is involved, all deceleration and acceleration processes that would last less than T_{AX2} are adapted.

AX1, AX3

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. This time is designated $t_{adaptxy}$ in the following diagrams:

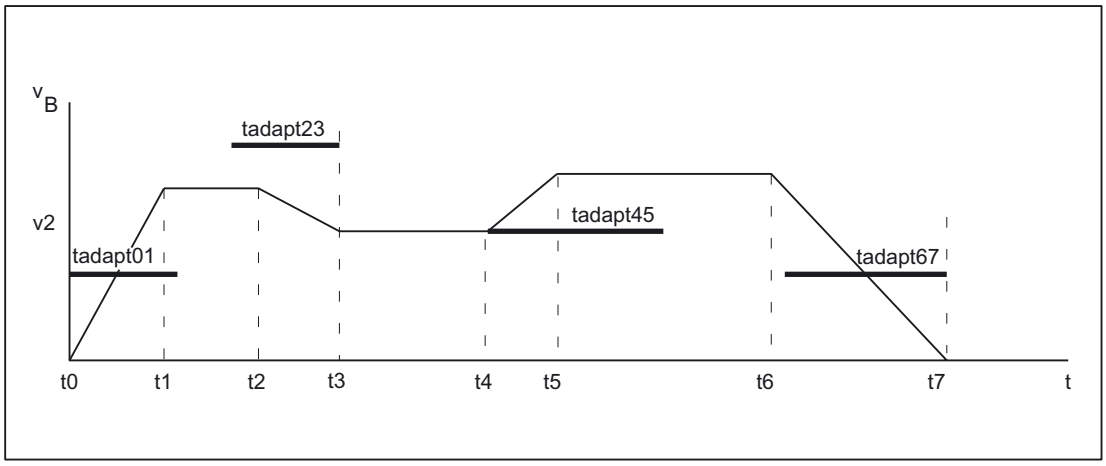


Fig. 2-8 Path velocity profile optimized for time without smoothing or dynamic adaptation response

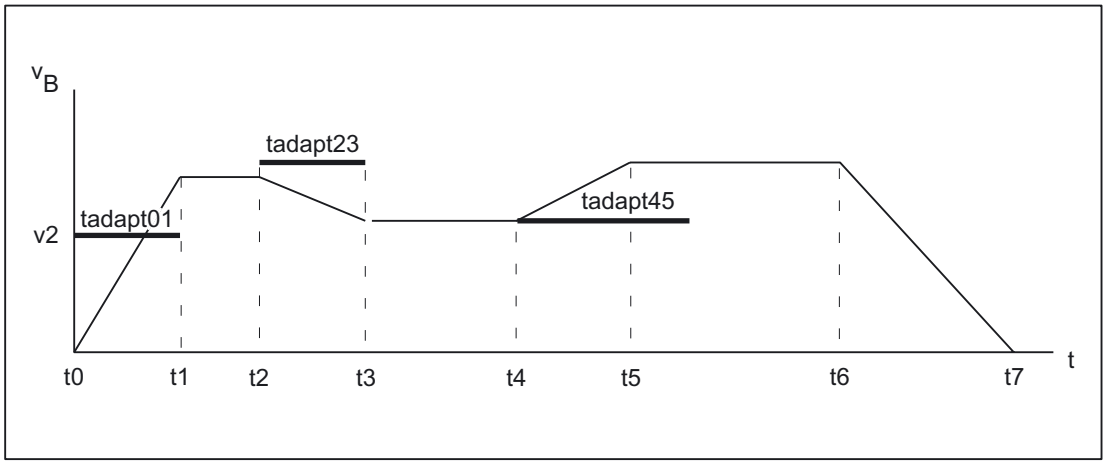


Fig. 2-9 Path velocity profile with adaptation of dynamic response

- Interval: t0 - t1 and t2 - t3
  The acceleration process between t0 - t1 and the deceleration process between t2 - t3 are lengthened in time due to an adaptation of the acceleration to time $t_{adapt01}$ and $t_{adapt23}$.
- Interval: t4 - t5
  The acceleration process between t4 - t5 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: t6 - t7
  The deceleration process between t6 - t7 remains unchanged, as it lasts longer than $t_{adapt67}$.

Example 2: Effect of smoothing the path velocity and dynamic response adaptation: acceleration mode: `BRISK`

Parameter assignment

Machine data

$MC_ADAPT_PATH_DYNAMIC[0] = 3	
$MC_LOOKAH_SMOOTH_FACTOR = 80%	
$MA_LOOKAH_FREQUENCY[AX1] = 20 Hz	T_{AX1} = 1/20 Hz = 50 ms
$MA_LOOKAH_FREQUENCY[AX2] = 20 Hz	T_{AX2} = 1/20 Hz = 50 ms
$MA_LOOKAH_FREQUENCY[AX3] = 20 Hz	T_{AX3} = 1/20 Hz = 50 ms

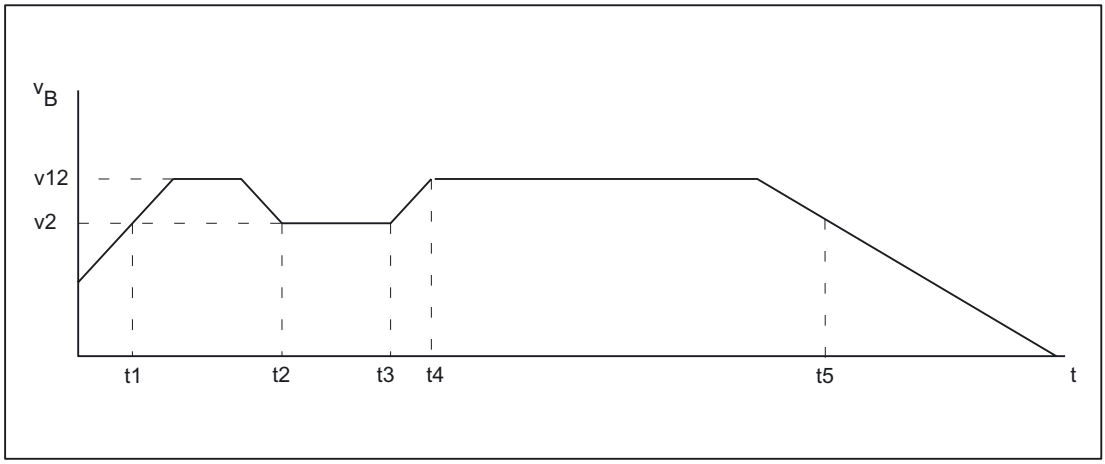


Fig. 2-10 Path velocity profile optimized for time without smoothing or dynamic adaptation response

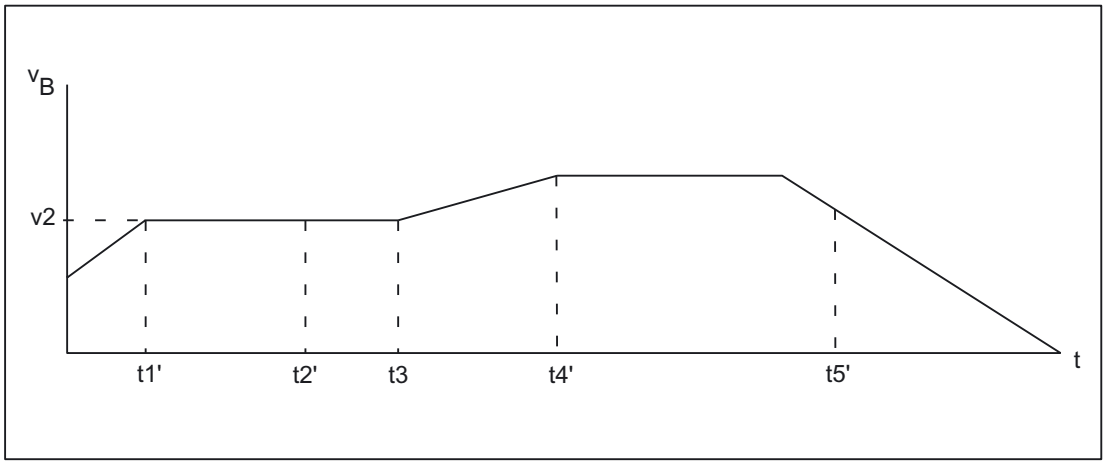


Fig. 2-11 Path velocity profile with path smoothing and adaptation of dynamic response

Effects of path smoothing

- Interval: t1 - t2
  The acceleration and deceleration process between t1 - t2 does not take place because the lengthening of the machining time without the acceleration process to v12 is less than the resulting time if a smoothing factor of 80% is applied.
- Interval: t3 - t5
  The acceleration and braking profile between t3 and t5 does not fulfill this condition or takes longer the assigned smoothing time: T_{AXn} = 2/20 Hz = 100 ms.

Effects of the dynamic response adaptation

- Interval: t3 - t4
  The acceleration process between t3 - t4 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 t1
  The acceleration up to t1 left over after path smoothing is extended to the time period up to t1' by the dynamic response adaptation.

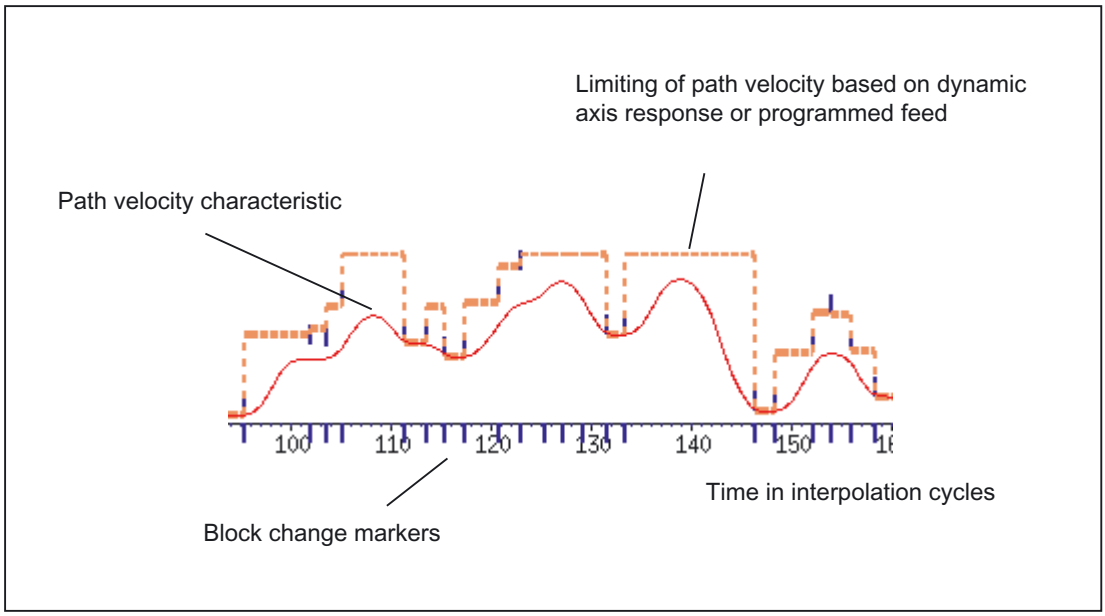
Conclusion
This example shows the interaction of path smoothing and dynamic response adaptation in continuous path mode. Only those acceleration or deceleration processes that were not eliminated by the path smoothing are subsequently optimized by the dynamic response adaptation. For this reason, both functions should always be activated, if possible.

Example 3: Effect of adaptation and smoothing the path velocity; acceleration mode: `BRISK`

Parameter assignment

$MC_ADAPT_PATH_DYNAMIC[1] = 4	
$MC_LOOKAH_SMOOTH_FACTOR = 100%	
$MA_LOOKAH_FREQUENCY[AX1] = 10 Hz	T_{AX1} = 1/10 Hz = 100 ms
$MA_LOOKAH_FREQUENCY[AX2] = 10 Hz	T_{AX2} = 1/10 Hz = 100 ms
$MA_LOOKAH_FREQUENCY[AX3] = 20 Hz	T_{AX3} = 1/20 Hz = 50 ms

Without path dynamic response adaptation or path smoothing

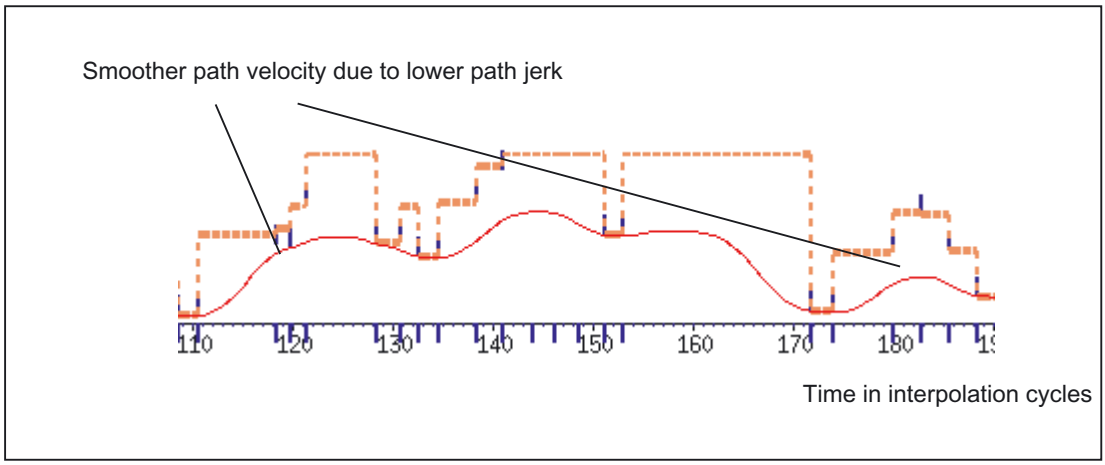


The path-velocity characteristic has been obtained through deselection of path dynamic response adaptation and path smoothing.

This corresponds to the following parameter settings:

```
$MC_ADAPT_PATH_DYNAMIC[1] = 1
$MC_LOOKAH_SMOOTH_FACTOR = 0%
```

With path dynamic response adaptation, without path smoothing

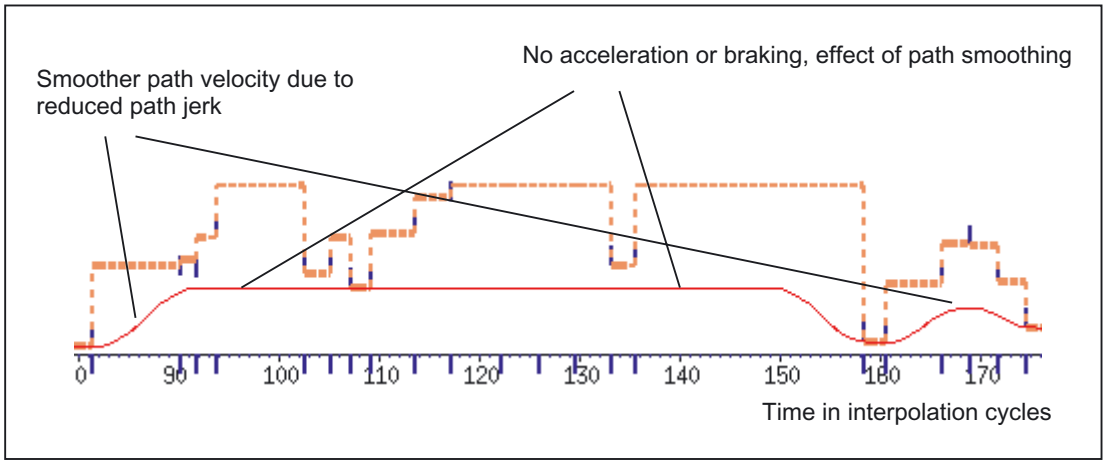


The path-velocity characteristic has been obtained through selection of path dynamic response adaptation with minimum, and thus virtually inactive, path smoothing.

The following parameter settings were made:

```
$MC_ADAPT_PATH_DYNAMIC[1] = 4
$MC_LOOKAH_SMOOTH_FACTOR = 1%
```

With path dynamic response adaptation and path smoothing



The path-velocity characteristic has been obtained through selection of path dynamic response adaptation and path smoothing.

The standard path rounding parameter settings for deselected path smoothing and active path dynamic response adaptation were selected:

```
$MC_ADAPT_PATH_DYNAMIC[1] = 4
$MC_LOOKAH_SMOOTH_FACTOR = 0%
(same meaning as $MC_LOOKAH_SMOOTH_FACTOR = 100%)
```

Installation and startup

The basic prerequisites for starting up the path dynamic response adaptation function are as follows:

- Determination of the natural frequency of the path axes for parameter assignment of dynamic response adaptation:
  MD32440 $MA_LOOKAH_FREQUENCY
- Determination of the dynamic response limiting values: velocity, acceleration, and jerk

Determination of dynamic response limiting values
The determination of the dynamic response limiting values for the traversing of path axes by means of acceleration with jerk limiting (SOFT) is described below. This procedure can be transferred analogously to the case of acceleration without jerk limiting (BRISK).

1. Deactivate the dynamic response adaptation:
   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. For all path axes, determine the F_{APD} factor with:
   F_{APD} = (largest determined jerk) / (smallest determined jerk)
   Note
   The smallest determined jerk is the value for the jerk for the most critical traversing velocity.
5. Enter the largest F_{APD} factor determined over all path axes as the value for the dynamic response adaptation:
   MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = F_{APD}

2.3.7 Technology G group

Dynamic response settings for technology G groups

Different settings can be stored and programmed for five different machining sections, e.g. tapping, roughing, smoothing, and smooth-finishing. So, for example, roughing can be performed optimized for time and smoothing, optimized for surface.

With machine data:
MD20150: GCODE_RESET_VALUES,
the basic setting Technology G group after RESET can be set.

Only the dynamic response of the path axes is determined when the technology group G code is activated. This has **no effect** on:

- Positioning axes
- PLC axes
- Command axes
- Movements based on axis coupling
- Overlaid movements with handwheel
- JOG movements
- Reference point approach (G74)
- Fixed point travel (G75)
- Rapid traverse movements (G0)

The dynamic response for these axis movements is still derived from the machine data of the DYNNORM default setting.

Technology G group

Five dynamic response settings are available in G code group 59 technology:

- DYNORM for standard dynamic response
- DYNPOS for positioning mode, tapping
- DYNROUGH for roughing
- DYNSEMIFIN for finishing
- DYNFINISH for smooth-finishing

For further information about programming, please refer to
References:
/PG/, Programming Guide Fundamentals, Path traversing behavior.

Enabling selected dynamics groups only

Some G codes can be deactivated by the machine manufacturer. If deactivated G codes are programmed alarm 14011 is output. G codes DYNPOS for positioning mode and DYNSEMIFIN for finishing, for example, can be deactivated with the following setting.

MD10712: NC_USER_CODE_CONF_NAME_TAB[0]="DYNPOS"
MD10712: NC_USER_CODE_CONF_NAME_TAB[1]=" "
MD10712: NC_USER_CODE_CONF_NAME_TAB[2]="DYNSEMIFIN"
MD10712: NC_USER_CODE_CONF_NAME_TAB[3]=" "

Configuring the dynamic response values.

The values for each technology group are stored in the following machine data:

- Axis
  MD32300: MAX_AX_ACCEL[n]
  MD32431: MAX_AX_JERK[n]
  MD20600: MAX_PATH_JERK[n]
  MD32432: PATH_TRANS_JERK_LIM[n]
  MD32310: MAX_ACCEL_OVL_FACTOR[n]
  MD32433: SOFT_ACCEL_FACTOR[n]
- Path
  MD20602: CURV_EFFECT_ON_PATH_ACCEL[n]
  MD20603: CURV_EFFECT_ON_PATH_JERK[n]

Value frange for index n = 0 to 4

Example of programming with index:
MD32300: MAX_AX_ACCEL[3, AX1]=1
R1=MD20602: CURV_EFECT_ON_PATH_ACCEL[4]

Notice

Writing the machine data **without an index** places the same value in all field elements of the machine data in question.

Reading the machine data **without an index** always supplies the value of the field with index 0.

These constellations can occur:
1. Write from part program MD32300: MAX_AX_ACCEL[AX1]=1 sets all five field elements to value 1 (unit of measurement metric or inch depending on axis type and reference system).

2. Read in archive files from software versions earlier than SW 7.2.

Activate parameter sets for dynamic response

By programming G codes, the machine data affecting dynamics are activated and assigned as follows:

Active parameter set	**G CODE of group 59 technology:**
Index 0: Value for:	DYNNORM
Index 1: Value for:	DYNPOS
Index 2: Value for:	DYNROUGH
Index 3: Value for:	DYNSEMIFIN
Index 4: Value for:	DYNFINISH

Note

We recommend suppressing **G codes that are not used**
with MD10712: NC_USER_CODE_CONF_NAME_TAB[n]. This will prevent these G codes from being programmed accidentally and activating unconfigured machine data.

2.4 LookAhead

Function

LookAhead is a procedure in continuous-path mode (G64, G641) that achieves velocity control with LookAhead over several NC part program blocks beyond the current block.

If the program blocks only contain very small paths, a velocity per block is achieved that permits deceleration of the axes at the block end point without violating acceleration limits. This means that the programmed velocity was not actually reached although a sufficient number of prepared blocks with virtually tangential path transitions was available.
With the LookAhead function it is possible to plan the acceleration and deceleration phase with approximately tangential path transitions in order to achieve a higher feedrate with shorter distances. Deceleration to velocity limits is possible with LookAhead such that violation of the acceleration and velocity limit is prevented.

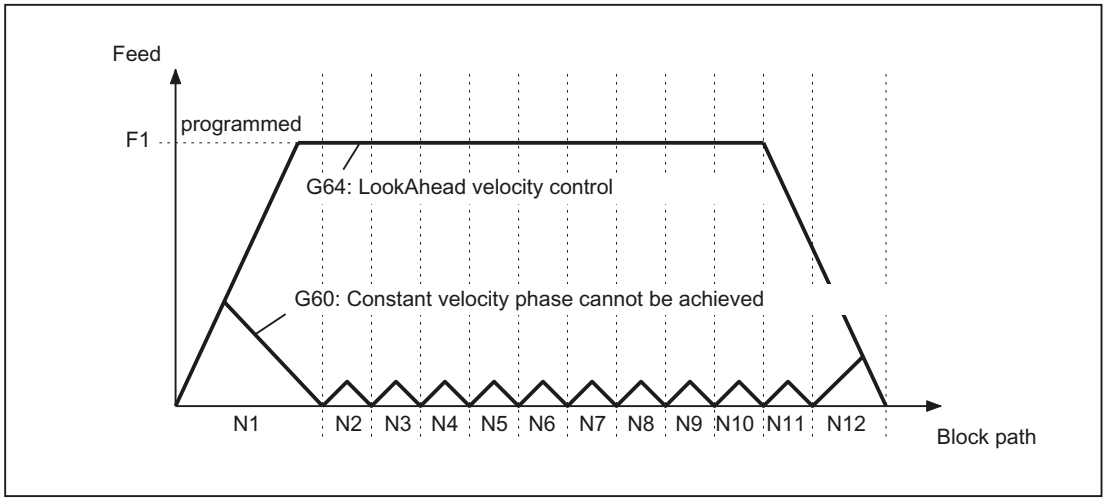


Fig. 2-12 Comparison of velocity response with exact stop G60 and continuous-path mode G64 with LookAhead with short traversing paths.

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

Scope

The LookAhead function is only available for path axes and not for spindles and positioning axes.

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.

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 standard 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 (see fig. below). It is enough 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 = 1m/s^2 and a high feedrate of v_{path} = 10 m/min, the following number of blocks for the control is obtained with an attainable block cycle time of the control of TB = 10 ms.

$n_{LookAhead}$= Deceleration path/Block length = (v_{path}^2 / (2a)) / (v_{path} * TB) = 9

Considering these aspects, 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.

As, in a program, the machining velocity is very often set to a lower value than the maximum velocity, more blocks than are required would be predicted, overloading the processor unnecessarily. The number of blocks required is therefore derived from the velocity, which is calculated by multiplying the programmed speed by the value of machine data:
MD12100 OVR_FACTOR_LIMIT_BIN
(limit of binary-coded override switch)
or by the 31st override value of machine data:
MD12030 OVR_FACTOR_FEEDRATE
(analysis of path feedrate override switch).

The 31st override value must tally with the override factor most frequently used.
The number of blocks considered by the LookAhead function is limited by the possible number of NC blocks in the IPO buffer.

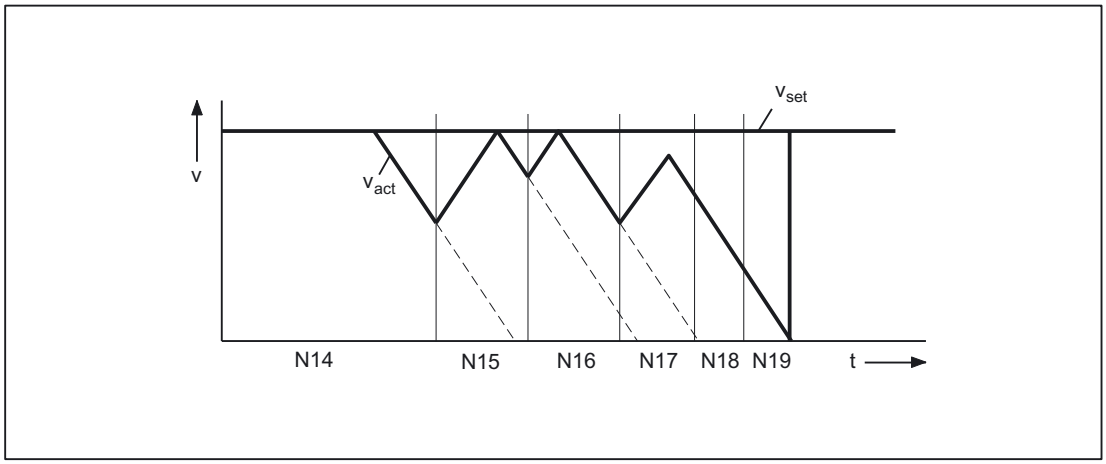


Fig. 2-13 Example for modal velocity control (number of blocks considered by the LookAhead function = 2)

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.

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.

This function is activated using machine data:
MD20400 LOOKAH_USE_VELO_NEXT_BLOCK
(LookAhead for following block velocity).

Override points

If the velocity profile of the following block velocity is not sufficient because, for example, very high override values, e.g., 200%, or constant cutting rate G96/G961 are being used, 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 then 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 higher value must take account of the velocity range reached by the maximum value of machine data:
MD12030 OVR_FACTOR_FEEDRATE
(analysis of path feedrate override switch)
or the value of machine data:
MD12100 OVR_FACTOR_LIMIT_BIN
(limit of binary-coded override switch)
respectively.

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 already at 100% override, a point must be set in the lower override range as well.

The number of override points used for each channel is specified in machine data:
MD20430 LOOKAH_NUM_OVR_POINTS
(number of override switch points for LookAhead).

The corresponding points are written to machine data:
MD20440 LOOKAH_OVR_POINTS
(override switch points for LookAhead).

A combination of both procedures can be used to calculate the velocity profile and is generally recommended, because the preset machine data for these functions already takes the widest range of override-specific velocity limits into account.
Plannable velocity limits restrict override-specific velocity limits.
If neither of the two procedures has been activated, the setpoint velocity is always applied in the current block.

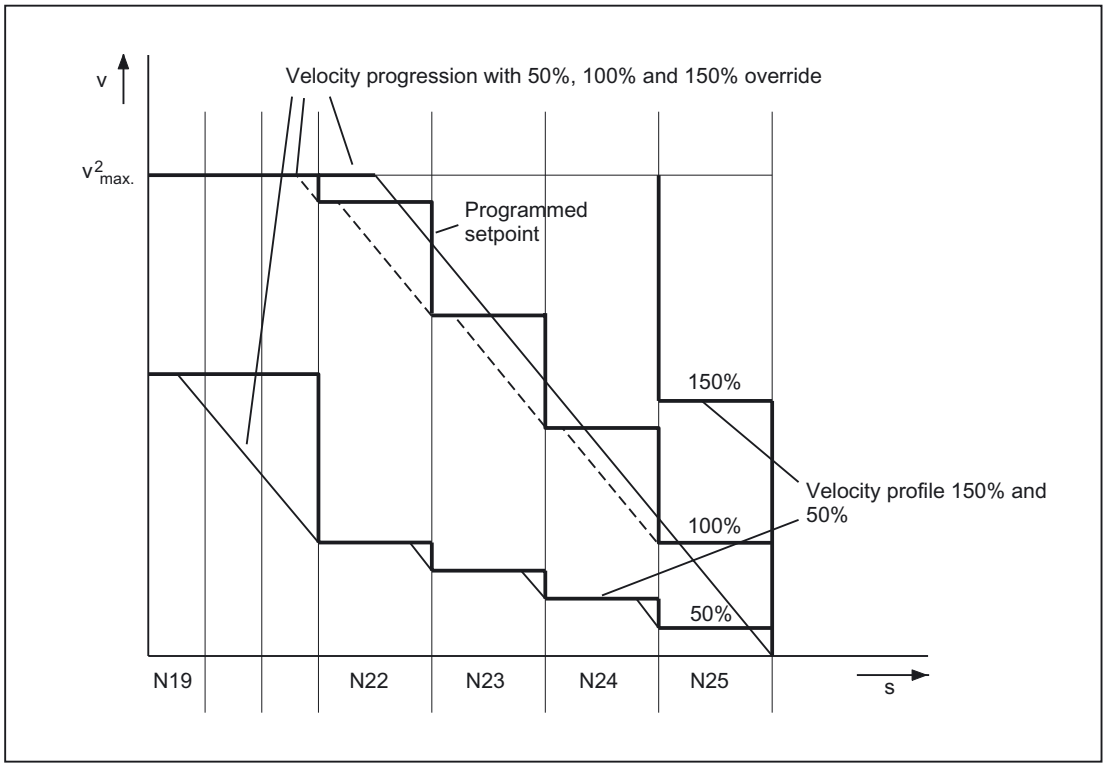


Fig. 2-14 Example for limiting velocity characteristics with number of LookAhead blocks = 4

... and the following settings:

MD20430 LOOKAH_NUM_OVR_POINTS = 2

MD20440 LOOKAH_OVR_POINTS = 1.5, 0.5

MD20400 LOOKAH_USE_VELO_NEXT_BLOCK = 1

Block cycle problem

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.

Machine data:
MD20450 LOOKAH_RELIEVE_BLOCK_CYCLE
(relief factor for the block cycle time)
can be set to smooth such velocity fluctuations.

Selection and deselection of LookAhead

The LookAhead function is selected and deselected with continuous-path mode (G64 or G641 respectively).

Special cases of LookAhead

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 movement 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: /FB1/, A3, Axis clamping.

2.5 NC block compressor COMPON, COMPCURV, -CAD

COMPON, COMPCURV

The modal G code COMPON or COMPCURV can be used to activate an "NC block compressor".
This function collects a series of linear blocks during linear interpolation (the number is limited to 10) and approximates them within a tolerance specified in machine data via a 3rd degree (COMPON) or 5th degree (COMPCURV) polynomial.
One larger traversing block is processed by the NC instead of a large number of small blocks.

COMPCAD

The COMPCAD G code can be used to select a further compression, which optimizes the surface quality and velocity. The interpolation accuracy can again be specified in machine data.

COMPCAD is processor and memoryintensive. It should only be used if surface quality enhancement measures cannot be incorporated in the CAD/CAM program.

The programming procedure is described in:
References:
/PGA/ Programming Guide, Advanced.

The compressor for orientation transformation is described in:
References:
/FB3/ Description of Functions, Special Functions; 3 to 5-axis transformation (F2).

The following three machine data are available for the compressor function:

MD20170 COMPRESS_BLOCK_PATH_LIMIT	This MD specifies the maximum path length for block compression. Longer blocks are not compressed.
MD33100 COMPRESS_POS_TOL	A tolerance can be specified for each axis. This value specifies the maximum deviation of the generated spline curve from the programmed end points. The higher the values, the more blocks can be compressed.
MD20172 COMPRESS_VELO_TOL	The maximum permissible deviation of the path feedrate while the compressor is active in conjunction with FLIN and FCUB can be specified here (does **not** apply to the COMPCAD command).

Recommendation for MD settings

The following machine data affect the compressor function and should contain the following values (specified in mm):

Machine data	Recommended value
MD18360 MM_EXT_PROG_BUFFER_SIZE	100
MD28520 MM_MAX_AXISPOLY_PER_BLOCK	3
MD28530 MM_PATH_VELO_SEGMENTS	5
MD28540 MM_ARCLENGTH_SEGMENTS	10
MD28070 MM_NUM_BLOCKS_IN_PREP	60
MD28060 MM_IPO_BUFFER_SIZE	100
SD42470 CRIT_SPLINE_ANGLE	36
MD20170 COMPRESS_BLOCK_PATH_LIMIT	20
MD20172 COMPRESS_VELO_TOL	100
MD32310 MAX_ACCEL_OVL_FACTOR[AX1]	<Value for G64 operation>
MD32310 MAX_ACCEL_OVL_FACTOR[AX2]	<Value for G64 operation>
MD32310 MAX_ACCEL_OVL_FACTOR[AX3]	<Value for G64 operation>
MD20490 IGNORE_OVL_FACTOR_FOR_ADIS	1

A tolerance can be specified for each axis: This value specifies the maximum deviation of the generated spline curve from the programmed end points. The higher the values, the more blocks can be compressed.

Experience has shown that a value of 0.01 is suitable for most applications:

```
MD33100 COMPRESS_POS_TOL[AX1] = 0.01
MD33100 COMPRESS_POS_TOL[AX2] = 0.01
MD33100 COMPRESS_POS_TOL[AX3] = 0.01
```

If not, the value can be increased to 0.02, for example:

```
MD33100 COMPRESS_POS_TOL[AX1] = 0.02
MD33100 COMPRESS_POS_TOL[AX2] = 0.02
MD33100 COMPRESS_POS_TOL[AX3] = 0.02
NewConfig
```

Activating the MD values

The new values are activated after the `NewConfig` command.

The rounding function `G642` and jerk limitation `SOFT` can be used to achieve further improvements in surface quality. These commands must be entered at the start of the program:

`COMPCAD SOFT G642`

`COMPOF` terminates the compressor function.

All blocks are compressed for which a simple syntax is sufficient:

`N... G1X... Y... Z... F...`

where "..." is a number and X, Y, Z are axis names.

Blocks with e.g., extended addresses such as `C=100` or `A=AC(100)` or blocks with auxiliary functions are also compressed.

Supplementary Conditions

3

3.1 Rounding and repositioning (REPOS)

Repositioning within the rounding area

If the traversing motion of the path axes within the corner rounding area is interrupted for traversing blocks with programmed rounding (part program command G641, G642, G643 or G644), repositioning occurs as follows in the event of a subsequent REPOS operation, depending on the current REPOS mode:

REPOS mode	
RMB	Block start of interrupted traversing block
RMI	Block end of interrupted traversing block
RME	Block end of interrupted traversing block
RMN	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.

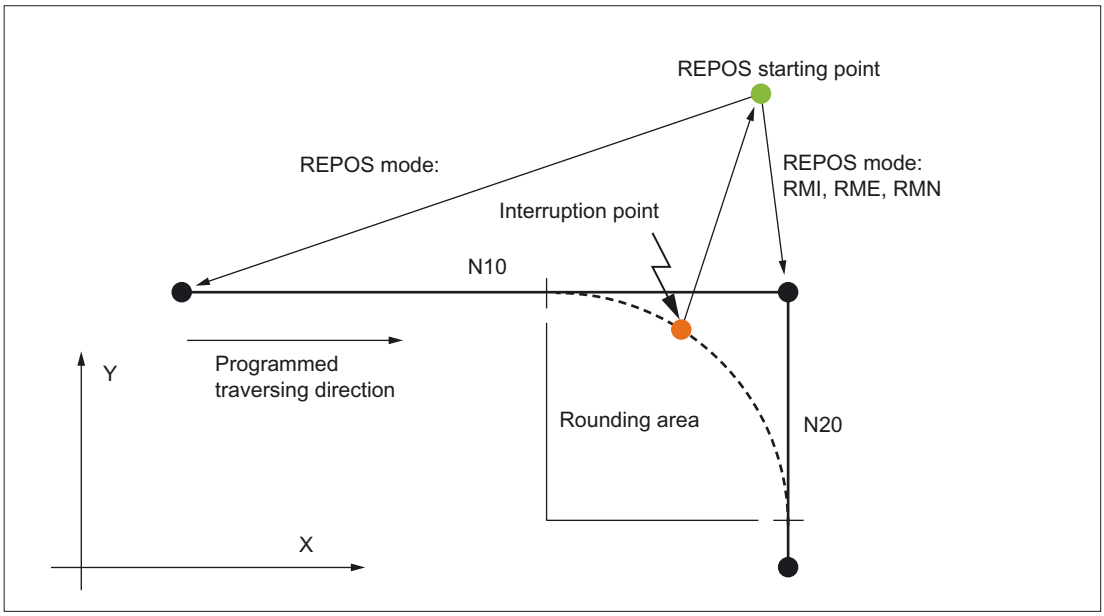


Fig. 3-1 Example of Rounding and REPOS

3.2 Smoothing the path velocity

Several blocks with SOFT and BRISK

Smoothing of the path velocity is only effective in continuous-path mode with LookAhead over several blocks with `SOFT` and `BRISK`, but not with `G0`.

The cycle times of the control must be parameterized such that the preprocessing is provided with enough blocks to be able to analyze an acceleration process.

Examples

4

4.1 Example of jerk limitation on the path

```
...
N1000 G64 SOFT              ; Continuous-path mode with SOFT acceleration
                              characteristics
N1004 G0 X-20 Y10
N1005 G1 X-20 Y0            ; Straight
N1010 G3 X-10 Y-10 I10      ; Block transition with jump in path curvature
                              (straight - circular)
N1011 G3 X0 Y0 J10          ; Block transition with continuous-path curvature
N1020 G2 X5 Y5 I5           ; Block transition with jump in path curvature
                              (circular - circular)
N1021 G2 X10 Y0 J-5
...
```

Data Lists

5

5.1 Machine data

5.1.1 General machine data

Number	Identifier: $MN_	Description
10110	PLC_CYCLE_TIME_AVERAGE	Maximum PLC acknowledgment time
18360	MM_EXT_PROG_BUFFER_SIZE	FIFO buffer size for processing from external

5.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
20400	LOOKAH_USE_VELO_NEXT_BLOCK	Look Ahead for constant acceleration velocity control
20430	LOOKAH_NUM_OVR_POINTS	Number of override switch points for Look Ahead
20440	LOOKAH_OVR_POINTS	Override switch points for 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 takes feedrate into account
20465	ADAPT_PATH_DYNAMIC	Adaptation of path dynamic response
20480	SMOOTHING_MODE	Rounding behavior with G642, G643, G644
20490	IGNORE_OVL_FACTOR_FOR_ADIS	G641/G642 irrespective of overload factor
20550	EXACT_POS_MODE	Exact stop conditions with G0/G1
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
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)

Number	Identifier: $MC_	Description
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
28540	MM_ARCLENGTH_SEGMENTS	Number of storage elements for arc length function representation per block

5.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
32310	MAX_ACCEL_OVL_FACTOR	Overload factor for velocity jump
32431	MAX_AX_JERK	Maximum axis jerk for path motion
32432	PATH_TRANS_JERK_LIM	Maximum 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
32440	LOOKAH_FREQUENCY	Smoothing limit frequency for LookAhead
33100	COMPRESS_POS_TOL	Maximum deviation with compensation
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

5.2 Setting data

5.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42465	SMOOTH_CONTUR_TOL	Max. contour deviation on rounding
42466	SMOOTH_ORI_TOL	Max. deviation of the tool orientation on rounding
42470	CRIT_SPLINE_ANGLE	Limit angle for spline and polynomial interpolation and compressor

5.3 Signals

5.3.1 Signals from channel

DB number	Byte.Bit	Description
21, ...	36.3	All axes stationary

5.3.2 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	60.6	Position reached with exact stop coarse
31, ...	60.7	Position reached with exact stop fine

Index

A

Adaptation
  Dynamic Response, 1-2
Auxiliary function output, 2-7

B

Blending, 2-11
Block-change point, 2-6

C

COMPCAD, 2-41
COMPCURV, 2-41
COMPON, 2-41
Compressor for orientation transformation, 2-41
Continuous-path mode, 2-6

E

Exact stop criteria, 2-2, 2-3, 2-4
  Exact stop coarse, 2-4
  Exact stop fine, 2-4
Exact stop criterion, 2-6

G

Geometry axes, 2-7, 2-11

I

Implicit continuous-path mode, 2-10
Implicit exact stop, 2-6
Interpolator end, 2-4

L

LookAhead, 2-6, 2-36
  Block cycle problem, 2-40
  Following block velocity, 2-38
  Number of blocks, 2-37
  Override, 2-39
  Selection and deselection, 2-40
  Velocity profiles, 2-38

M

MD settings
  NC block compressor, 2-42
MD10110, 2-7, 2-8
MD12030, 2-38, 2-39
MD12100, 2-38, 2-39
MD18360, 2-42
MD20150, 2-33
MD20170, 2-42
MD20172, 2-42
MD20400, 2-38
MD20430, 2-39
MD20440, 2-39
MD20450, 2-40
MD20460, 2-22, 2-23, 2-24, 2-26
MD20462, 2-22
MD20465, 2-23, 2-25, 2-26, 2-27, 2-28, 2-30, 2-33
MD20480, 2-15, 2-18, 2-19
MD20490, 2-9, 2-42
MD28060, 2-42
MD28070, 2-42
MD28520, 2-42
MD28530, 2-18, 2-42
MD28540, 2-42
MD32210, 2-9
MD32300, 2-9
MD32310, 2-42
MD32431, 2-33
MD32440, 2-19, 2-23, 2-24, 2-25
MD33100, 2-15, 2-16, 2-19, 2-42, 2-43

N

NC block compressor
  COMPON, COMPCURV, COMCAD, 2-41

O

Orientation axes, 2-11
Overload factor, 2-9

P

Path criterion, 2-13
POS, 2-7

R

Rounding with contour tolerance, 2-15

S

SD42465, 2-16
SD42466, 2-16
SD42470, 2-42
SPOS, 2-7
Synchronized axes, 2-7, 2-12

V

Velocity reduction according to overload factor, 2-8

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Acceleration (B2)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Customer benefit 1-1

1.2 Features 1-1

1.3 Prerequisites 1-3

2 Detailed Description 2-1

2.1 Acceleration without jerk limitation (BRISK/BRISKA) (channel-/axis-specific) 2-1
2.1.1 Detailed description 2-1
2.1.1.1 General information 2-1
2.1.1.2 Programmable maximum value (axis-specific) 2-3
2.1.2 Activation 2-3
2.1.2.1 Parameterization 2-3
2.1.3 Programming 2-3
2.1.3.1 Path acceleration without jerk limitation (BRISK) 2-3
2.1.3.2 Single-axis acceleration without jerk limitation (BRISKA) 2-4

2.2 Constant travel time (channel-specific) 2-5
2.2.1 Detailed description 2-5
2.2.1.1 General information 2-5
2.2.2 Activation 2-6
2.2.2.1 Parameterization 2-6
2.2.3 Programming 2-6

2.3 Acceleration matching (ACC) (axis-specific) 2-7
2.3.1 Detailed description 2-7
2.3.1.1 General information 2-7
2.3.2 Activation 2-7
2.3.3 Programming 2-7

2.4 Acceleration margin (channel-specific) 2-8
2.4.1 Detailed description 2-8
2.4.1.1 General information 2-8
2.4.2 Activation 2-9
2.4.2.1 Parameterization 2-9
2.4.3 Programming 2-9

2.5 Path-acceleration limitation (channel-specific) 2-9
2.5.1 Detailed description 2-9
2.5.1.1 General information 2-9
2.5.2 Activation 2-10
2.5.2.1 Parameterization 2-10
2.5.3 Programming 2-10
2.5.3.1 Limit value 2-10
2.5.3.2 Activation/Deactivation 2-10

2.6 Path acceleration for real-time events (channel-specific) 2-11
2.6.1 Detailed description 2-11
2.6.1.1 General information 2-11

2.6.2 Activation ... 2-12
2.6.3 Programming ... 2-12

2.7 Acceleration with programmed rapid traverse (G00) (axis-specific) ... 2-13
2.7.1 Detailed description ... 2-13
2.7.1.1 General information ... 2-13
2.7.2 Activation ... 2-14
2.7.2.1 Parameterization ... 2-14
2.7.3 Programming ... 2-14

2.8 Acceleration with active jerk limitation (SOFT/SOFTA) (axis-specific) ... 2-14
2.8.1 Detailed description ... 2-14
2.8.1.1 General information ... 2-14
2.8.2 Activation ... 2-15
2.8.2.1 Parameterization ... 2-15
2.8.3 Programming ... 2-15

2.9 Excessive acceleration for non-tangential block transitions (axis-specific) ... 2-16
2.9.1 Detailed description ... 2-16
2.9.1.1 General information ... 2-16
2.9.2 Activation ... 2-16
2.9.2.1 Parameterization ... 2-16
2.9.3 Programming ... 2-17

2.10 Acceleration margin for radial acceleration (channel-specific) ... 2-17
2.10.1 Detailed description ... 2-17
2.10.1.1 General information ... 2-17
2.10.2 Activation ... 2-19
2.10.2.1 Parameterization ... 2-19
2.10.3 Programming ... 2-19

2.11 Jerk limitation with path interpolation (SOFT) (channel-specific) ... 2-19
2.11.1 Detailed description ... 2-19
2.11.1.1 General information ... 2-19
2.11.1.2 Maximum jerk value (axis-specific) ... 2-21
2.11.1.3 Maximal jerk value (canal-specific) ... 2-21
2.11.2 Activation ... 2-22
2.11.2.1 Parameterization ... 2-22
2.11.3 Programming ... 2-22

2.12 Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) ... 2-23
2.12.1 Detailed description ... 2-23
2.12.1.1 General information ... 2-23
2.12.2 Activation ... 2-23
2.12.2.1 Parameterization ... 2-23
2.12.3 Programming ... 2-23

2.13 Path-jerk limitation (channel-specific) ... 2-24
2.13.1 Detailed description ... 2-24
2.13.1.1 General information ... 2-24
2.13.2 Activation ... 2-25
2.13.2.1 Parameterization ... 2-25
2.13.3 Programming ... 2-25
2.13.3.1 Maximum path-jerk ... 2-25
2.13.3.2 Activation/Deactivation ... 2-25

2.14 Path jerk for real-time events (channel-specific) ... 2-26
2.14.1 Detailed description ... 2-26
2.14.1.1 General information ... 2-26
2.14.2 Activation ... 2-27
2.14.3 Programming ... 2-27

2.15 Jerk with programmed rapid traverse (G00) (axis-specific) ... 2-28
2.15.1 Detailed description ... 2-28
2.15.1.1 General information ... 2-28
2.15.2 Activation ... 2-29
2.15.2.1 Parameterization ... 2-29
2.15.3 Programming ... 2-29

2.16 Excessive jerk for block transitions without constant curvature (axis-specific) ... 2-29
2.16.1 Detailed description ... 2-29
2.16.1.1 General information ... 2-29
2.16.2 Activation ... 2-30
2.16.2.1 Parameterization ... 2-30
2.16.3 Programming ... 2-30

2.17 Jerk filter (axis-specific) ... 2-30
2.17.1 Detailed description ... 2-30
2.17.1.1 General information ... 2-30
2.17.2 Activation ... 2-34
2.17.2.1 Parameterization ... 2-34
2.17.3 Programming ... 2-34

2.18 Knee-shaped acceleration characteristic curve ... 2-34
2.18.1 Detailed description ... 2-34
2.18.1.1 Adaptation for the motor characteristic curve ... 2-34
2.18.1.2 Effect on the path acceleration ... 2-37
2.18.1.3 Substitute characteristic curve ... 2-38
2.18.2 Activation ... 2-40
2.18.2.1 Parameterization ... 2-40
2.18.2.2 Activation ... 2-41
2.18.3 Programming ... 2-41
2.18.3.1 Channel-specific activation (DRIVE) ... 2-41
2.18.3.2 Axis-specific activation (DRIVEA) ... 2-42

3 Supplementary Conditions ... 3-1

3.1 Acceleration and jerk ... 3-1

3.2 Knee-shaped acceleration characteristic curve ... 3-1
3.2.1 Active kinematic transformation ... 3-1

4 Examples ... 4-1

4.1 Acceleration ... 4-1
4.1.1 Path velocity characteristic ... 4-1

4.2 Jerk ... 4-2
4.2.1 Path velocity characteristic ... 4-2

4.3 Acceleration and jerk ... 4-4

4.4 Knee-shaped acceleration characteristic curve ... 4-5
4.4.1 Activation ... 4-5

5 Data Lists ... 5-1

5.1 Machine data ... 5-1
5.1.1 Channel-specific machine data ... 5-1
5.1.2 Axis-/spindle-specific machine data ... 5-1

5.2 Setting data ... 5-2
5.2.1 Channel-specific setting data ... 5-2

5.3 System variables ... 5-2

Index ... Index-1

Brief Description

1

1.1 Customer benefit

Key statement

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.

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)

1.3 Prerequisites

Key statement

There are no specific requirements to be met.

Detailed Description 2

2.1 Acceleration without jerk limitation (BRISK/BRISKA) (channel-/axis-specific)

2.1.1 Detailed description

2.1.1.1 General information

Key statement

In the case of acceleration without jerk limitation (jerk = infinite) the maximum value is applied for acceleration immediately. As regards 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

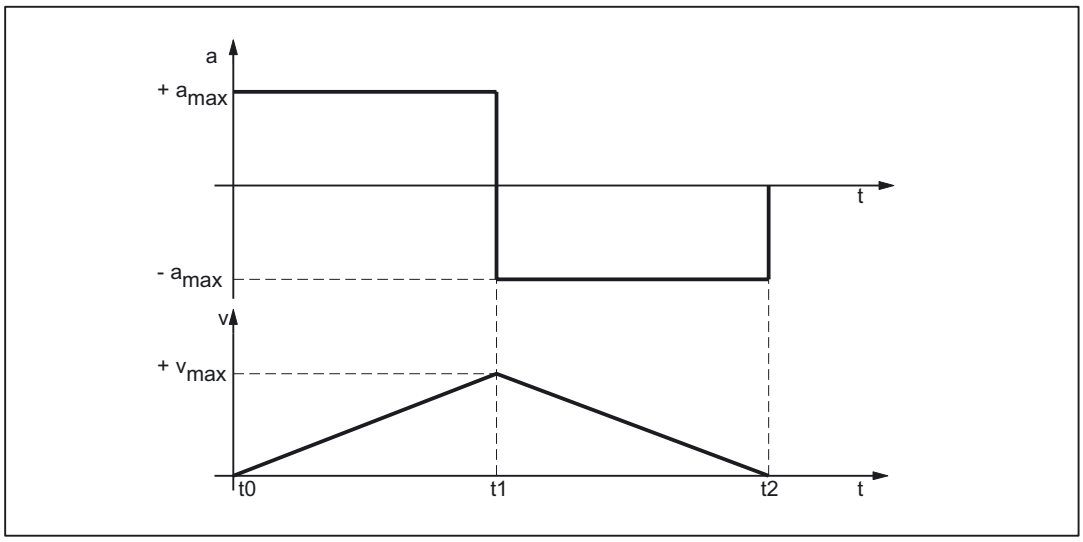


Fig. 2-1 Velocity and acceleration schematic for stepped acceleration profile

a_{max}:	Maximum acceleration value
v_{max}:	Maximum velocity value
t:	Time

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

Sudden acceleration can generally be avoided by setting a constant travel time (see chapter: "Abrupt acceleration changes with constant travel phase").

- Interval: t_1 - t_2

  Constant acceleration with $-a_{max}$; linear decrease in velocity

2.1.1.2 Programmable maximum value (axis-specific)

Key statement

The maximum acceleration value can be set for each specific machine axis:

MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)

The path parameters are calculated by the path planning component during preprocessing so that the programmed maximum values of the machine axes that are of relevance for the path are not exceeded.

Exceeding of maximum value

It is possible for the maximum value to be exceeded in connection with specific machining situations (see Chapter: "Acceleration matching (ACC) and system variables ($AC_PATHACC)).

2.1.2 Activation

2.1.2.1 Parameterization

Key statement

The maximum values are parameterized for specific axes using machine data:

MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)

2.1.3 Programming

2.1.3.1 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.

2.1.3.2 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

2.2 Constant travel time (channel-specific)

2.2.1 Detailed description

2.2.1.1 General information

Key statement

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.

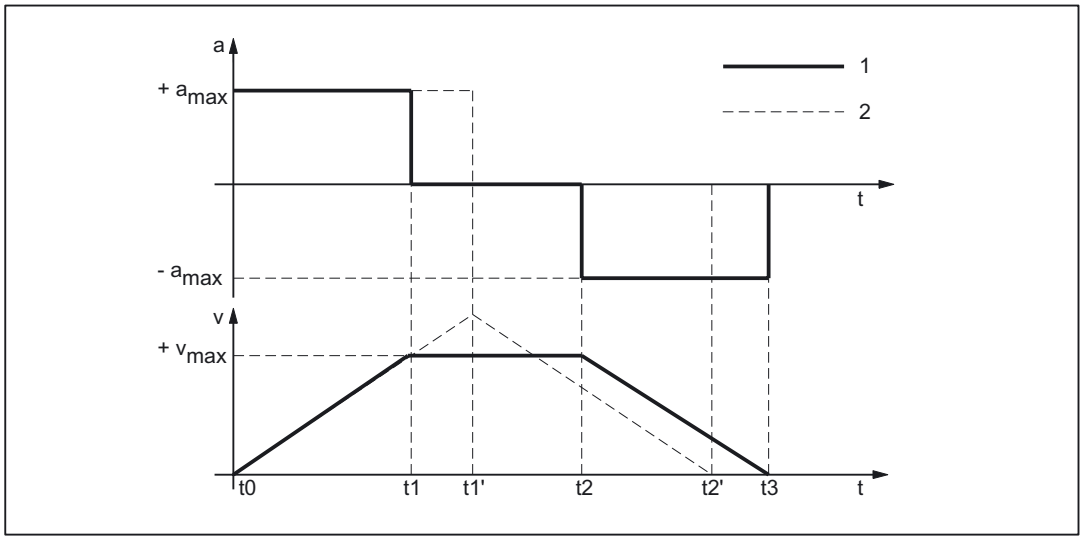


Fig. 2-2 Schematic for abrupt acceleration

1: Characteristic with constant travel time
2: Characteristic without constant travel time
a_{max}: Maximum acceleration value
v_{max}: Maximum velocity value
t: Time

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.

2.2.2 Activation

2.2.2.1 Parameterization

Key statement

The constant travel time is parameterized for specific channels using machine data:

MD20500 $MC_CONST_VELO_MIN_TIME
(minimum time with constant velocity)

2.2.3 Programming

Key statement

The function is not programmable.

2.3 Acceleration matching (ACC) (axis-specific)

2.3.1 Detailed description

2.3.1.1 General information

Key statement

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).

2.3.2 Activation

Key statement

The function does not need to be activated.

2.3.3 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

Acceleration matching is deactivated via a reset, or by setting the adjustment factor to 100% within the control.

2.4 Acceleration margin (channel-specific)

2.4.1 Detailed description

2.4.1.1 General information

Key statement

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.

2.4.2 Activation

2.4.2.1 Parameterization

Key statement

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)

2.4.3 Programming

Key statement

The function is not programmable.

2.5 Path-acceleration limitation (channel-specific)

2.5.1 Detailed description

2.5.1.1 General information

Key statement

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

2.5.2 Activation

2.5.2.1 Parameterization

Key statement

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)

2.5.3 Programming

2.5.3.1 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

2.5.3.2 Activation/Deactivation

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

2.6 Path acceleration for real-time events (channel-specific)

2.6.1 Detailed description

2.6.1.1 General information

Key statement

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
- Feedrate override modifications
- Modification of the velocity default for "safely reduced velocity" within the context of the "Safely Integrated" function

the path acceleration can be set for the real-time events specified above 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 control 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

Real-time-event path acceleration is enabled, irrespective of the radial acceleration.

Effective

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 modifications • Modification of the velocity default for "safely reduced velocity" within the context of the "Safely Integrated" function
Ineffective	Real-time-event path acceleration 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 Chapter: Programming.

2.6.2 Activation

Key statement

The function does not need to be activated.

2.6.3 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/s^2

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`).

2.7 Acceleration with programmed rapid traverse (G00) (axis-specific)

2.7.1 Detailed description

2.7.1.1 General information

Key statement

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.

2.7.2 Activation

2.7.2.1 Parameterization

Key statement

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
```

2.7.3 Programming

Key statement

The function is not programmable.

2.8 Acceleration with active jerk limitation (SOFT/SOFTA) (axis-specific)

2.8.1 Detailed description

2.8.1.1 General information

Key statement

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

2.8.2 Activation

2.8.2.1 Parameterization

Key statement

The maximum value for acceleration with active jerk limitation (S0FT/S0FTA) is parameterized using the axis-specific machine data:

MD32434 $MA_SOFT_ACCEL_FACTOR
(scaling of the acceleration limitation with SOFT)

2.8.3 Programming

Key statement

The function is not programmable.

2.9 Excessive acceleration for non-tangential block transitions (axis-specific)

2.9.1 Detailed description

2.9.1.1 General information

Key statement

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
```

2.9.2 Activation

2.9.2.1 Parameterization

Key statement

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)

2.9.3 Programming

Key statement

The function is not programmable.

2.10 Acceleration margin for radial acceleration (channel-specific)

2.10.1 Detailed description

2.10.1.1 General information

Key statement

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)

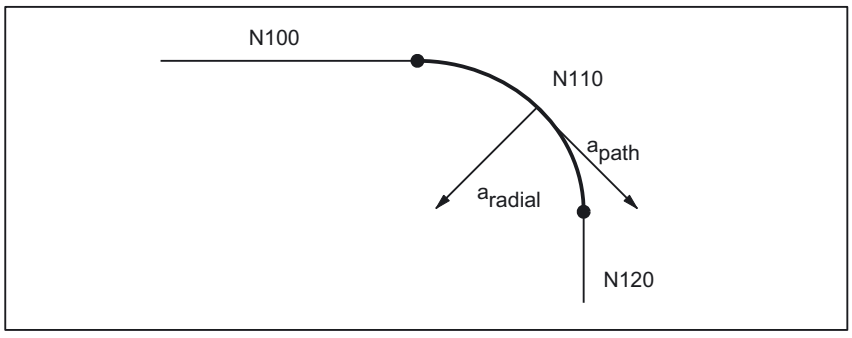


Fig. 2-3 Radial and path acceleration on curved contours

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:

MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL =

$$\frac{a_{radial}\ [m/s^2]}{MD32300\ \$MA_MAX_AX_ACCEL\ [m/s^2]} = \frac{0.694}{3} \approx 0.23$$

Linear motions

The acceleration margin referred to above is ineffective in the case of linear motions (linear interpolation) without active kinematic transformation.

2.10.2 Activation

2.10.2.1 Parameterization

Key statement

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)

2.10.3 Programming

Key statement

The function is not programmable.

2.11 Jerk limitation with path interpolation (SOFT) (channel-specific)

2.11.1 Detailed description

2.11.1.1 General information

Key statement

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

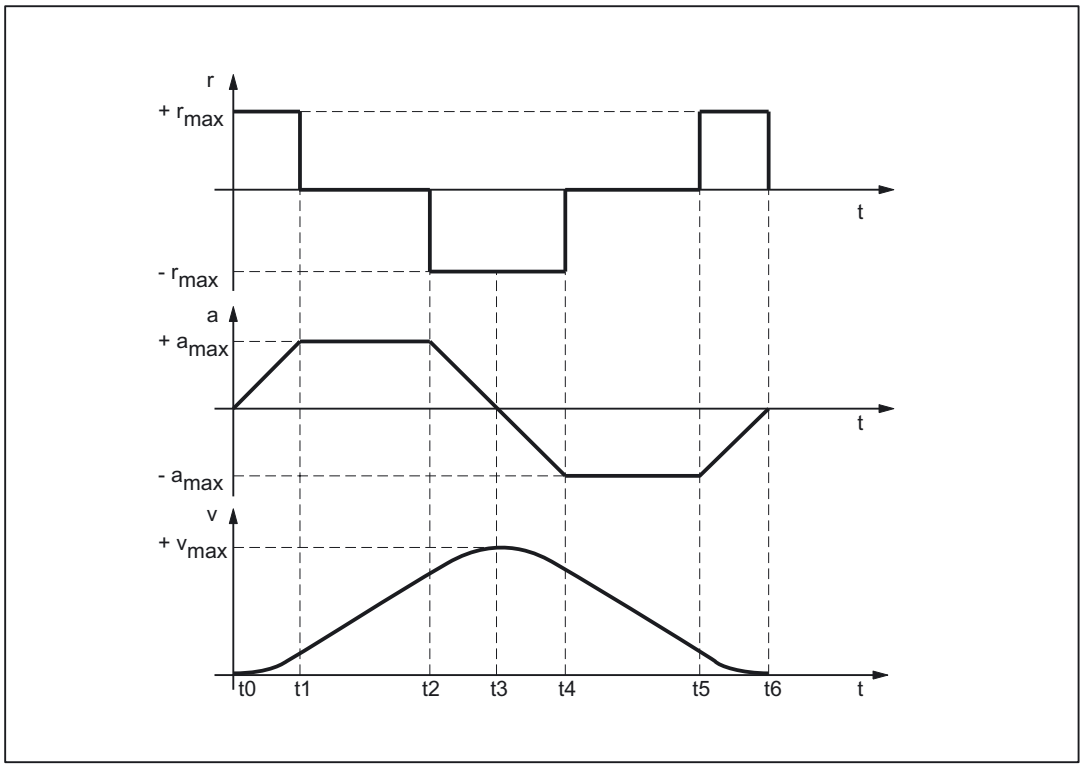


Fig. 2-4 Jerk, acceleration and velocity schematic with jerk limitation acceleration profile

r_{max}:	Maximum jerk value
a_{max}:	Maximum acceleration value
v_{max}:	Maximum velocity value
t:	Time

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

2.11.1.2 Maximum jerk value (axis-specific)

Key statement

The maximum jerk value can be set for each specific machine axis using the following machine data:
MD32431 $MA_MAX_AX_JERK (maximum axis jerk)

The path parameters are calculated by the path planning component during preprocessing so that the programmed maximum values of the machine axes that are of relevance for the path are not exceeded.

Exceeding of maximum value

It is possible for the maximum value to be exceeded in connection with specific machining situations (see next Chapter: "System variables ($AC_PATHJERK)")

2.11.1.3 Maximal jerk value (canal-specific)

Key statement

As well as it being possible to set the maximum jerk value for specific axes, it can also be assigned as a channel-specific path parameter using the following machine data:
MD20600 $MC_MAX_PATH_JERK (path-related maximum jerk)

To prevent the axis and channel-specific maximum jerk values interfering with one another, the channel-specific value must be set to a value greater then the maximum axial values.

2.11.2 Activation

2.11.2.1 Parameterization

Key statement

The axis- and channel-specific maximum values are parameterized using the following machine data:

MD32431 $MA_MAX_AX_JERK (maximum axis jerk)

MD20600 $MC_MAX_PATH_JERK (path-related maximum jerk)

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]

Supplementary 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.

2.12 Jerk limitation with single-axis interpolation (SOFTA) (axis-specific)

2.12.1 Detailed description

2.12.1.1 General information

Key statement

The maximum jerk value can be set for each specific machine axis for single-axis movements (e.g., JOG, JOG/INC, positioning axis, reciprocating axis, setup modes, etc.):

MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axis jerk)

Initial setting

Acceleration with jerk limitation can be set as the axial initial setting:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE
(initial setting of axial jerk limitation)

2.12.2 Activation

2.12.2.1 Parameterization

Key statement

The function's initial setting and the maximum values are parameterized for specific axes using machine data:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE
(initial setting of axial jerk limitation)

MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axis jerk)

2.12.3 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

2.13 Path-jerk limitation (channel-specific)

2.13.1 Detailed description

2.13.1.1 General information

Key statement

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

2.13.2 Activation

2.13.2.1 Parameterization

Key statement

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)

2.13.3 Programming

2.13.3.1 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

2.13.3.2 Activation/Deactivation

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

2.14 Path jerk for real-time events (channel-specific)

2.14.1 Detailed description

2.14.1.1 General information

Key statement

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
- Feedrate override modifications
- Modification of the velocity default for "safely reduced velocity" within the context of the "Safely Integrated" function

the path jerk can be set for the real-time events specified above 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.

Effective

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 modifications • Modification of the velocity default for "safely reduced velocity" within the context of the "Safely Integrated" function
Ineffective	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/s^2]

  Smoothing time: Freely selectable, e.g., 0.02 s

For information about programming system variables in the part program or synchronized actions, see Chapter: Programming.

2.14.2 Activation

Key statement

The function does not need to be activated.

2.14.3 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/s^3

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 (ST0PRE).

2.15 Jerk with programmed rapid traverse (G00) (axis-specific)

2.15.1 Detailed description

2.15.1.1 General information

Key statement

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.

2.15.2 Activation

2.15.2.1 Parameterization

Key statement

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

2.15.3 Programming

Key statement

The function is not programmable.

2.16 Excessive jerk for block transitions without constant curvature (axis-specific)

2.16.1 Detailed description

2.16.1.1 General information

Key statement

In the case of block transitions without constant curvature (e.g., straight > circle), 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 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.

2.16.2 Activation

2.16.2.1 Parameterization

Key statement

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)

2.16.3 Programming

Key statement

The function is not programmable.

2.17 Jerk filter (axis-specific)

2.17.1 Detailed description

2.17.1.1 General information

Key statement

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 has its advantages, where very large filter time constants are concerned but the 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), e.g., in the HMI Advanced "Axes" service screen form, 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 2 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:

$$f_{Zmax} = \frac{1}{2 * \pi * T_{Zmin}} = \frac{1}{2 * \pi * T_{Position\text{-}control\ cycle}} \quad \text{(Shannon Theorem)}$$

The limit is enabled for a specific axis using the machine data:
MD32400 AX_JERK_ENABLE (axial jerk limitation)
and set (with time specification) for the smoothing filter via the machine data:
MD32410 AX_JERK_TIME (time constant for axial jerk filter).

With **SW 5.1** and higher, it is also possible to control the jerk limitation in the position controller with a new filter based on a smoothing method that incurs few contour errors:

MD32402 AX_JERK_MODE = 1	;	2nd-order filter (default)
		corresponds to SW 1 to SW 4.4
MD32402 AX_JERK_MODE = 2	;	Sliding mean value generation
		(New jerk filter with SW 5.1 and higher).
MD32402 AX_JERK_MODE = 3	;	Bandstop filter with SW 6.3 and higher

Modus 2 requires a bit more computation time, but with the same smoothing effect, it results in lower contour errors or, with the same accuracy, in a smoother contour with smoother movements. Mode 2 is recommended. Mode 1 is the default mode in SW 1 to SW 4.4 for compatibility reasons.

For more information about how the jerk filter available with SW 5.1 and higher works (balancing filter for improving the position setpoints of the position controller), please refer to:
References:
/FB1/ Description of Functions, Basic Machine; Velocities, Setpoint/Actual-Value Systems, Regulatory Control (G2), Chapter: "Control Optimization"

Mode 3

There are 2 parameterization options for the bandstop filter:

- "Real bandstop filter":

  When identical numerator and denominator natural frequencies are selected (=blocking frequency). If you select (numerator) damping setting zero, the blocking frequency is equivalent to complete attenuation.

  In this case the 3 dB bandwidth is determined on the following basis:

  $f_{bandwidth} = 2 * f_{block.}$

  If instead of complete attenuation, a reduction by a factor of k is all that is required, then the numerator damping should be selected in accordance with k.

- "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.

2.17.2 Activation

2.17.2.1 Parameterization

Activation

The jerk filter is activated using the machine data:

MD32400 $MA_AX_JERK_ENABLE = TRUE

The jerk filter is active in all operating modes and with all types of interpolation.

Filter mode

The mode is selected using the machine data:

MD32402 $MA_AX_JERK_MODE = *mode*

Mode = 1	(2nd-order filter)
Mode = 2	(Sliding mean value generation)
Mode = 3	(Bandstop filter)

2.17.3 Programming

Key statement

The function is not programmable.

2.18 Knee-shaped acceleration characteristic curve

2.18.1 Detailed description

2.18.1.1 Adaptation for the motor characteristic curve

Key statement

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.

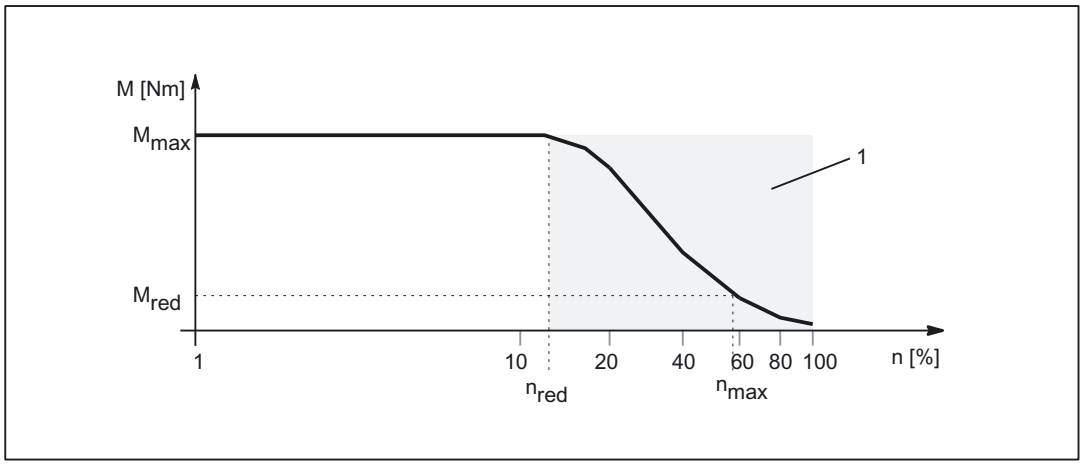


Fig. 2-5 Torque characteristic curve of a motor with torque characteristic that is highly dependent upon speed

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)

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

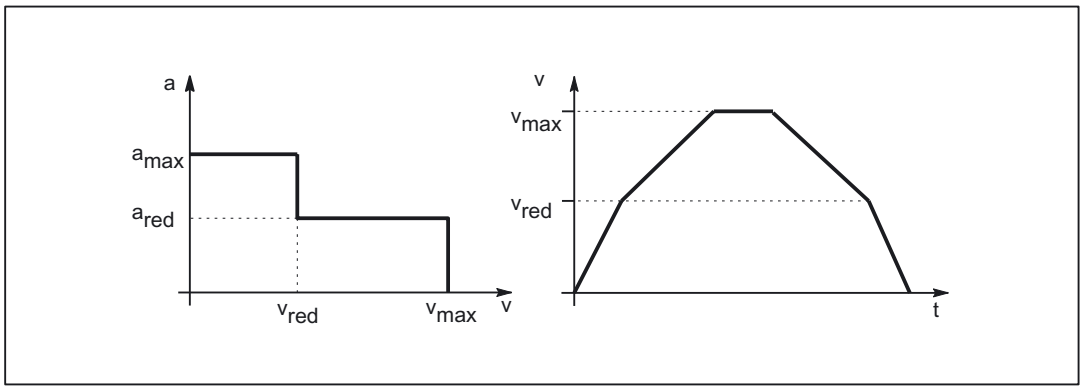


Fig. 2-6 Acceleration and velocity characteristic with acceleration reduction: 0 = constant

Hyperbolic characteristic

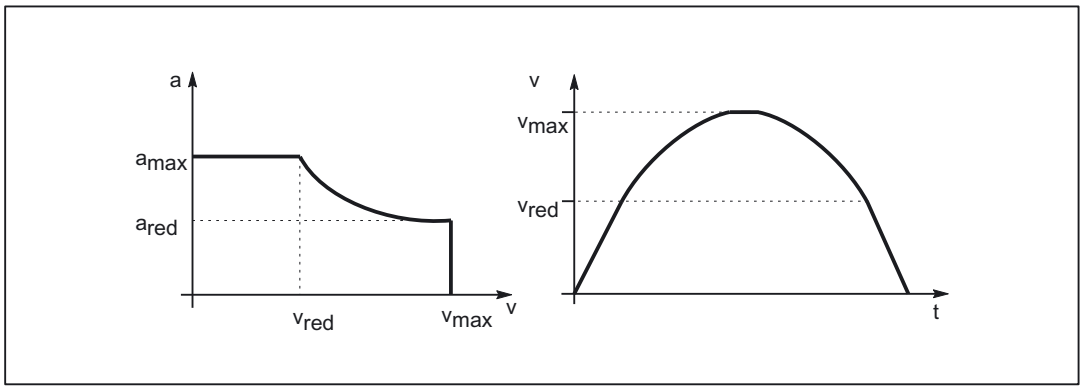


Fig. 2-7 Acceleration and velocity characteristic with acceleration reduction: 1 = hyperbolic

Linear characteristic

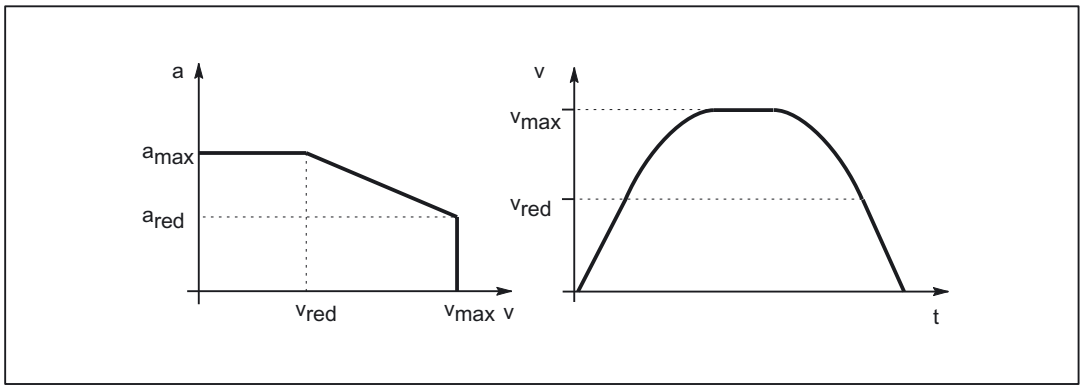


Fig. 2-8 Acceleration and velocity characteristic with acceleration reduction: 2 = linear

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

2.18.1.2 Effect on the path acceleration

Key statement

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.

2.18.1.3 Substitute characteristic curve

Key statement

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.

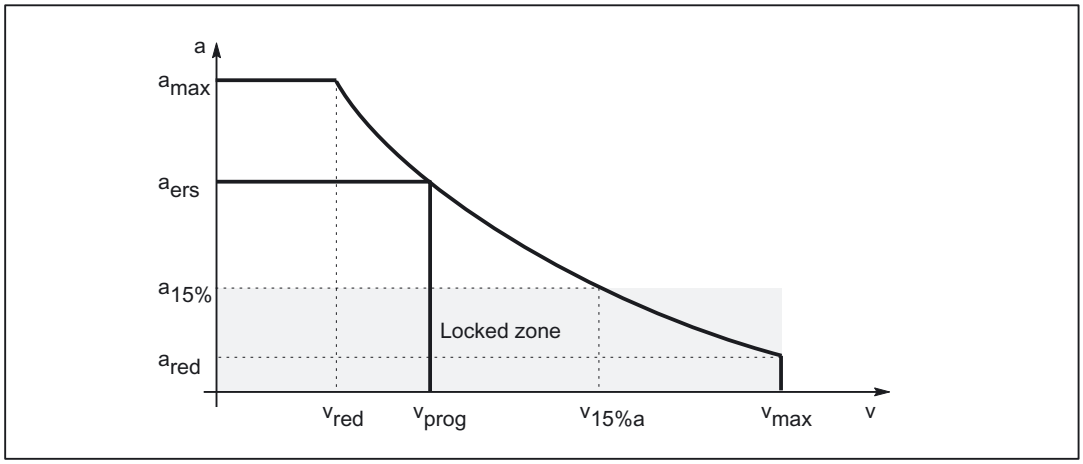


Fig. 2-9 Substitute path characteristic curve: Linear path

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 $a_{15\%}$

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.

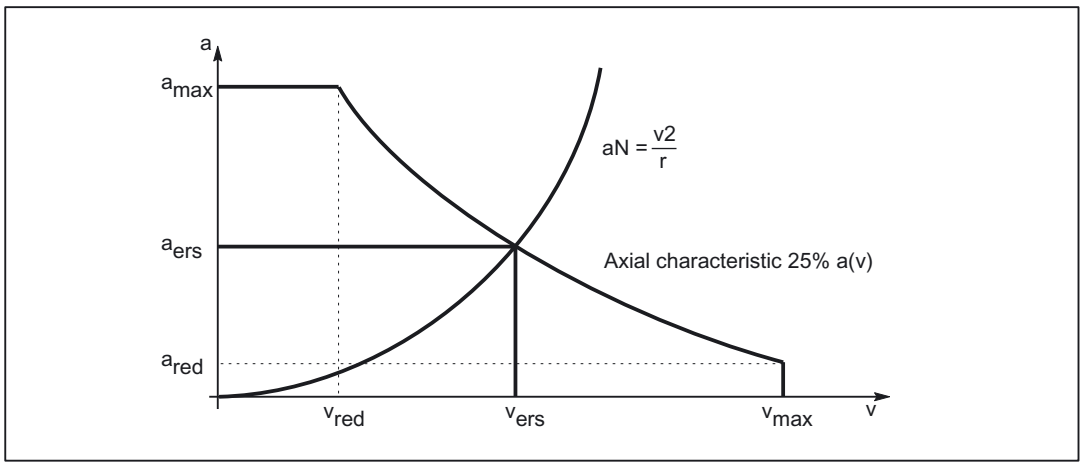


Fig. 2-10 Substitute path characteristic curve: Curved path

a_N:	Normal acceleration
a_{ers}:	Substitute characteristic curve constant acceleration
v_{ers}:	Substitute characteristic curve velocity
r:	Path radius

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.

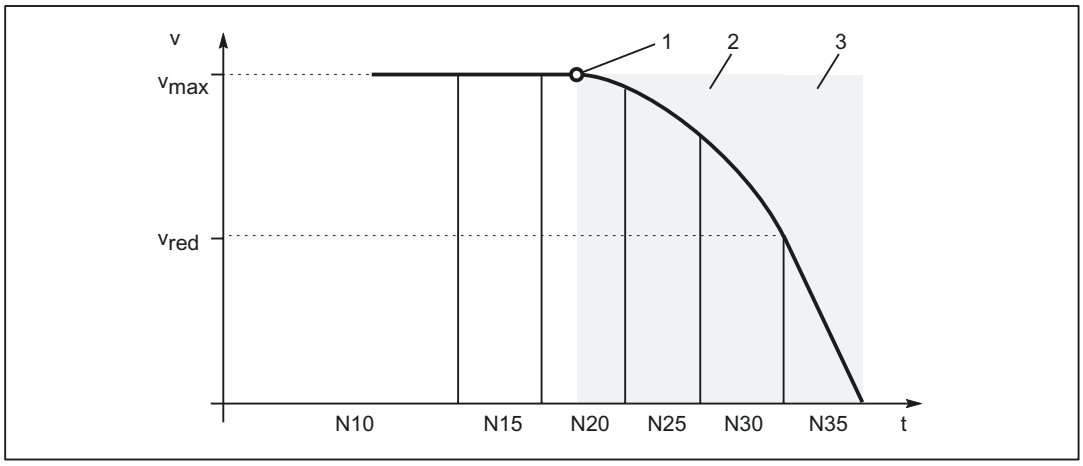


Fig. 2-11 Deceleration with LookAhead

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

2.18.2 Activation

2.18.2.1 Parameterization

Key statement

The knee-shaped acceleration characteristic curve is parameterized for specific axes using the following 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)

2.18.2.2 Activation

Key statement

The knee-shaped acceleration characteristic curve is activated for a specific machine axis using the machine data:

MD35240 $MA_ACCEL_TYPE_DRIVE = TRUE

Single-axis interpolation

As soon as the knee-shaped acceleration characteristic curve is activated, in the case of single-axis interpolations (positioning axis, reciprocating axis, manual travel, etc.), traversing is performed exclusively in DRIVEA mode.

Switching of the acceleration profile via the part-program instructions:

- Abrupt acceleration changes (BRISKA)
- Acceleration with jerk limitation (SOFTA)

is not possible.

Path interpolation

If the knee-shaped acceleration characteristic curve is parameterized for a machine axis that is of relevance for a programmed path, but the DRIVE part-program instruction has not been activated, then a substitute characteristic curve with lower dynamic limit values is generated for the path.

2.18.3 Programming

2.18.3.1 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.

2.18.3.2 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 the 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.

Supplementary Conditions

3

3.1 Acceleration and jerk

There are no other supplementary conditions to note.

3.2 Knee-shaped acceleration characteristic curve

3.2.1 Active kinematic transformation

Key statement

The knee-shaped acceleration characteristic curve is not taken into account in connection with an active kinematic transformation. The control switches to acceleration without jerk limitation (BRISK) and a substitute characteristic curve is adopted for path acceleration.

Examples

4

4.1 Acceleration

4.1.1 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)

```
; 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
```

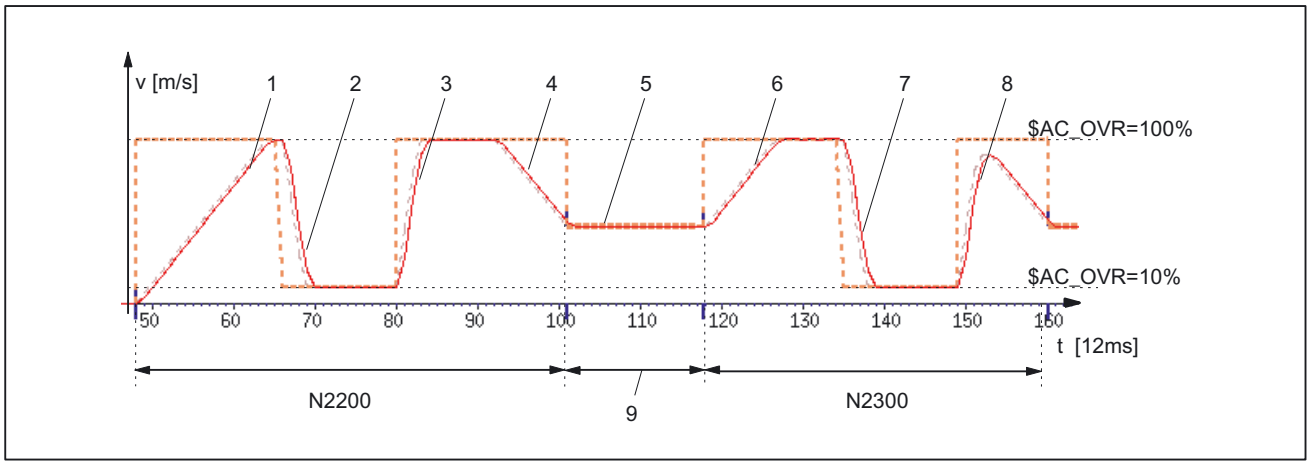


Fig. 4-1 Switching between path acceleration specified during preprocessing and real-time acceleration

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...)

4.2 Jerk

4.2.1 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)

```
; 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
```

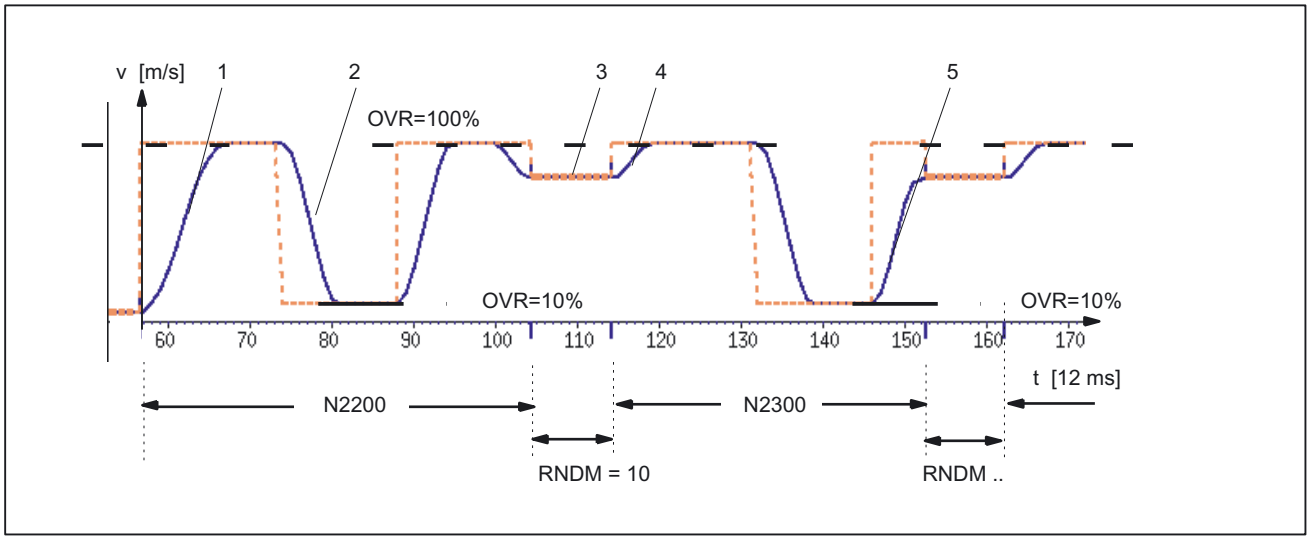


Fig. 4-2 Switching between path jerk specified during preprocessing and $AC_PATHJERK

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

4.3 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

```
N90 F5000 SOFT G64          ; Continuous-path mode, acceleration with jerk
                            limitation
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, 15° bend
N200 M30
```

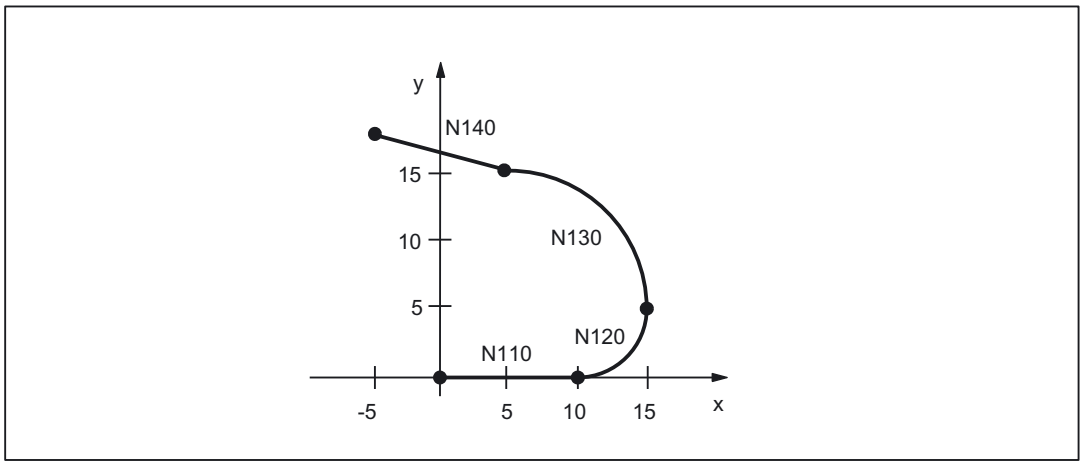


Fig. 4-3 Part program contour

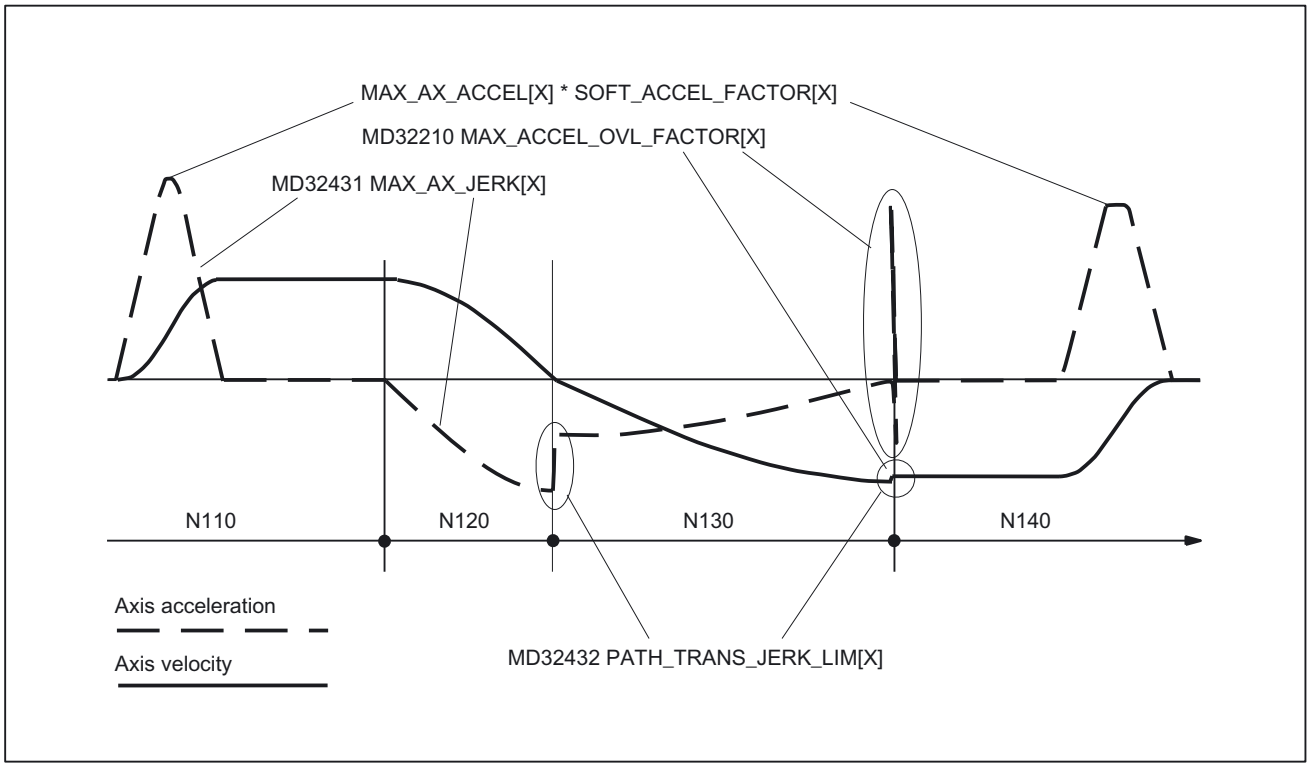


Fig. 4-4 X axis: Velocity and acceleration characteristic

4.4 Knee-shaped acceleration characteristic curve

4.4.1 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)

```
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)

```
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
...
```

Data Lists

5

5.1 Machine data

5.1.1 Channel-specific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of G groups
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity
20600	MAX_PATH_JERK	Pathrelated maximum jerk
20602	CURV_EFFECT_ON_PATH_ACCEL	Influence of path curvature on dynamic path response
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements

5.1.2 Axis-/spindle-specific machine data

Number	Identifier: $MA_	Description
32000	MAX_AX_VELO	Maximum axis velocity
32300	MAX_AX_ACCEL	Maximum axis acceleration
32310	MAX_ACCEL_OVL_FACTOR	Overload factor for velocity jump
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	Enabling axial jerk limitation
32430	JOG_AND_POS_MAX_JERK	Max. axial jerk for JOG and POS
32431	MAX_AX_JERK	Maximum axis jerk for path motion
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
35220	ACCEL_REDUCTION_SPEED_POINT	Speed for reduced acceleration
35230	ACCEL_REDUCTION_FACTOR	Acceleration reduction factor

Number	Identifier: $MA_	Description
35240	ACCEL_TYPE_DRIVE	"DRIVE" acceleration characteristic curve: ON/OFF
35242	ACCEL_REDUCTION_TYPE	Type of acceleration reduction

5.2 Setting data

5.2.1 Channel-specific 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

5.3 System variables

Identifier	Description
$AC_PATHACC	Path acceleration for real-time events
$AC_PATHJERK	Path jerk for real-time events

Index

$

$AC_PATHACC, 2-12, 2-27
$AC_PATHJERK, 2-26, 2-27, 2-28
$SC_IS_SD_MAX_PATH_ACCEL, 2-10
$SC_IS_SD_MAX_PATH_JERK, 2-25
$SC_SD_MAX_PATH_ACCEL, 2-10
$SC_SD_MAX_PATH_JERK, 2-25

A

ACC[axis], 2-7

B

BRISK, 2-3
BRISKA, 2-4

D

DRIVE, 2-41
DRIVEA, 2-42

J

Jerk limiting
    Smoothing method, 2-33

M

MD20150, 2-4, 2-22, 2-42
MD20500, 2-5, 2-6
MD20600, 2-21, 2-22
MD20602, 2-17, 2-18, 2-19
MD20610, 2-9
MD32000, 2-40
MD32300, 2-3, 2-14, 2-15, 2-16, 2-17, 2-18, 2-40
MD32310, 2-16
MD32400, 2-32, 2-34
MD32402, 2-34
MD32410, 2-32
MD32420, 2-4, 2-23, 2-24
MD32430, 2-23
MD32431, 2-21, 2-22, 2-29
MD32432, 2-30
MD32433, 2-15
MD32434, 2-14, 2-15, 2-29
MD32435, 2-29
MD35220, 2-37, 2-40
MD35230, 2-37, 2-40
MD35240, 2-41
MD35242, 2-35, 2-40

S

SD42500, 2-9, 2-10
SD42502, 2-9, 2-10
SD42510, 2-24, 2-25
SD42512, 2-24, 2-25
SOFT, 2-22
SOFTA, 2-23

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Diagnostic Tools (D1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Function 1-1

2 Detailed Description 2-1

2.1 Description of diagnostic tools 2-1

2.2 Service displays 2-4

2.3 Axis/spindle service display 2-5

2.4 Drive service display (for digital drives only) 2-14

2.5 Service display PROFIBUS DP 840Di 2-25

2.6 Communication log 2-28

2.7 PLC status 2-29

2.8 Other diagnostics tools 2-30

2.9 Identifying defective drive modules 2-31

3 Supplementary Conditions 3-1

3.1 Secondary conditions for diagnostic tools 3-1

4 Examples 4-1

4.1 Examples 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Axis/spindle-specific machine data 5-1
5.1.3 Drive-specific machine data 5-2

5.2 Setting data 5-2
5.2.1 Axis/spindle-specific setting data 5-2

5.3 Signals 5-3
5.3.1 Signals to axis/spindle 5-3
5.3.2 Signals from axis/spindle 5-3

Index Index-1

Tables

Table 2-1 Diagnostic screen PROFIBUS configuration 2-26
Table 2-2 Diagnostic screen information on the slaves 2-26
Table 2-3 Diagnostic screen detailed information on the slave 2-27
Table 2-4 Diagnostic screen AxisInfo 2-28
Table 2-5 Bus configuration example 2-33

Brief Description

1

1.1 Function

Diagnostic tools

Integrated and external diagnostic tools are available for operating the SINUMERIK control. In addition, the NC assists with error delimitation for drive problems by providing the option of simulating the drive interface of machine axes.

Integrated diagnostic tools

The following information is displayed via the HMI user interface:

- Display of alarms and messages from the control system or drives in plaintext
- Status displays for:
  - Interface signals from NC, MMC, PLC and I/O modules
  - Data blocks
  - PLC flags, timers and counters
  - Inputs and outputs of the PLC5
- Service displays
  - Nominal values, actual values and status data for axes/spindles
  - Communication error log for NC, PLC and MMC
  - Logbook
  - Display showing version of system software installed

External diagnostic tools

The 611D startup software (to be installed on an external computer) is used to configure and set parameters for SIMODRIVE 611-D drives.

The 611D startup software provides the following functions:

- Initial startup through direct input of drive parameters
- Startup by transferring standard data records that are derived from motor/power section combinations
- Automatic controller setting (self-optimization)
- Measurement functions for evaluating speed and position control loops and regulating torque in the time and frequency range without external measurement equipment.
- Fast Fourier Transformation (FFT) for analysis of the control loop setting and machine kinematics
- Circularity test
- Archiving drive and control data

By means of the DAC configuration, the 611D drive modules enable the output of all important control loop variables of the position, speed and torque levels to external measurement devices (e.g. oscilloscope, signal plotter) via measuring sockets.

References

A detailed description of the 611D startup software can be found in:
/FBA/ functional description of drive functions, speed control loop (DD2)

A detailed description of the circuit test can be found in:
/FB/ functional description of extended functions, compensations (K3)

Detailed Description 2

2.1 Description of diagnostic tools

Scope

The functional description deals with displays of the user interface, system functions, procedures for determining system statuses and if necessary measures for avoiding undesirable conditions for the NC control, PLC and drives.

General

Alarm and signal status displays

The currently active or not yet acknowledged alarms and messages are displayed in the Diagnostics operating area.

Alarm log

The alarm log contains the alarms that have occurred and the time. Detailed information on the individual alarms can be found in:

References: /DA/ Diagnosis instructions, in particular for systems with HMI Advanced in the **Online Help**.

Note

The corresponding explanations for alarms and messages which the machine tool manufacturer issues (range of values .....),

can be found in the machine tool manufacturer's documentation.

Alarm handler

Application

The alarm handler provides an infrastructure for activating and managing alarms on the NCK.

Functions

- Buffering of a maximum of 16 alarms that have been activated since system powerup and which have not yet been reset.
- Alarm reactions can be programmed as channel-specific, modegroupspecific or NCK-specific reactions.
- The "NoReady" alarm reaction can also be programmed as a channel-specific reaction.

Activation

The alarm handler is activated when an error status is detected in the NCK, causing an alarm to be output.

It is possible to trigger an alarm in a part program using the **SETAL** command.
Reference: /PGA/ Programming Guide Advanced

Note

The currently active alarms in the NCK are read via the operator panel interface.

It is not possible to set alarms externally in the NCK.

Alarms with an alarm ID in the 60000 to 60999 range can be activated in a part program.

Data backup

At POWER ON, the alarm handler data is completely reinitialized since it is not stored in the buffered SRAM.

Compatibility

As of SW 4

As of SW 4.1 and later, it is possible to set the channel-specific signal CHANNEL_NOREADY in the VDI interface in response to alarms.

Up to SW 3.x

MD11412 $MN_ALARM_REACTION_CHAN_NOREADY
controls whether the function channel-specific signal CHANNEL_NOREADY is used. This ensures that earlier PLC versions remain compatible.

Default setting: CHANNEL_NOREADY signal is not used.
Alarms that have specified a channel-specific NOREADY signal are reconfigured to the modegroupspecific NOREADY signal.

Clearing criterion

For each alarm, you must specify how the alarm can be cleared again. The following clearing criteria are possible:

- POWERONCLEAR
  The alarm is cleared by switching the control off and then on again.
- RESETCLEAR
  When the Reset key is pressed, the alarm is cleared in the channel in which it occurred.
- CANCELCLEAR
  The alarm is cleared in any channel when the Cancel key is pressed. The alarm can also be cleared by means of an NC start or Reset.
- NCSTARTCLEAR
  The alarm is cleared in the channel in which the alarm occurred by starting a program. The alarm can also be cleared by means of a Reset.
- CLEARHIMSELF
  The alarm is not cleared by an operator input or action, but explicitly by a "clearAlarm" programmed in the NCK source code.
- BAGRESETCLEAR
  The alarm is cleared by a "BAGRESETCLEAR" command or by executing a Reset in every channel of this mode group.
- NCKRESETCLEAR
  The alarm is cleared by an "NCKRESETCLEAR" command or by executing a Reset in every channel.

Alarm display control

The scope of the alarm outputs can be modified using machine data.

- MD11410 $MN_SUPPRESS_ALARM_MASK
  Mask for suppressing special alarm outputs
- MD11411 $MN_ENABLE_ALARM_MASK
  Mask for enabling special alarm outputs

For details of this machine data, please refer to chapter 4.

2.2 Service displays

Conditions of use

Conditions for the use of service displays are specified. Service displays are differentiated between in terms of axis/spindle, drive and profibus DP

Operation

For how to operate the service displays see:
References: /BAD/, "HMI Advanced Operator's Guide"
/BEM/, "HMI Embedded Operator's Guide"

Note

On HMI Advanced, it is possible to switch between the displays using the vertical soft key for Part view/Overall view. The data in the partial view are updated at significantly shorter intervals.

General

In principle, the following service displays are available:

- Axis/spindle service displays
- Drive service displays
- Profibus-DP service displays

Note

System dependencies

The availability of individual service displays depends on the particular system, e.g.:

- Drive service displays: for digital drives only
- Profibus-DP service displays: for SINUMERIK 840Di only

2.3 Axis/spindle service display

Values and statuses

Displays showing values and statuses on the control's user interface allow the operating status of the axes and spindles to be evaluated.

Accessing the diagnostic options

For the purposes of installing and diagnosing

- axes and
- spindles,

the information shown in the following figure can be called up for each axis/spindle in the "Diagnostics" operating area via the operator panel.

This information is used for

- Checking the setpoint branch
  (e.g. **position setpoint, speed setpoint, spindle speed setpoint prog.**)
- Checking the actual value branch
  (e.g. **actual position value measuring system 1/2, actual speed value**)
- Optimizing the position control loop
  (e.g. **following error, control deviation, servo gain factor**)
- Checking the whole control loop of the axis
  (e.g. by comparing the position setpoint and the actual position value, speed setpoint and the actual speed value)
- Checking hardware errors
  (e.g. by checking the encoder: If the axis is physically moved, a change in the actual position value must result)
- Setting and checking axis monitoring functions.

Descriptions on how to select and operate the "Diagnostics" area can be found in
References: /BAD/, "HMI Advanced Operator's Guide"
/BEM/, "HMI Advanced Operator's Guide"

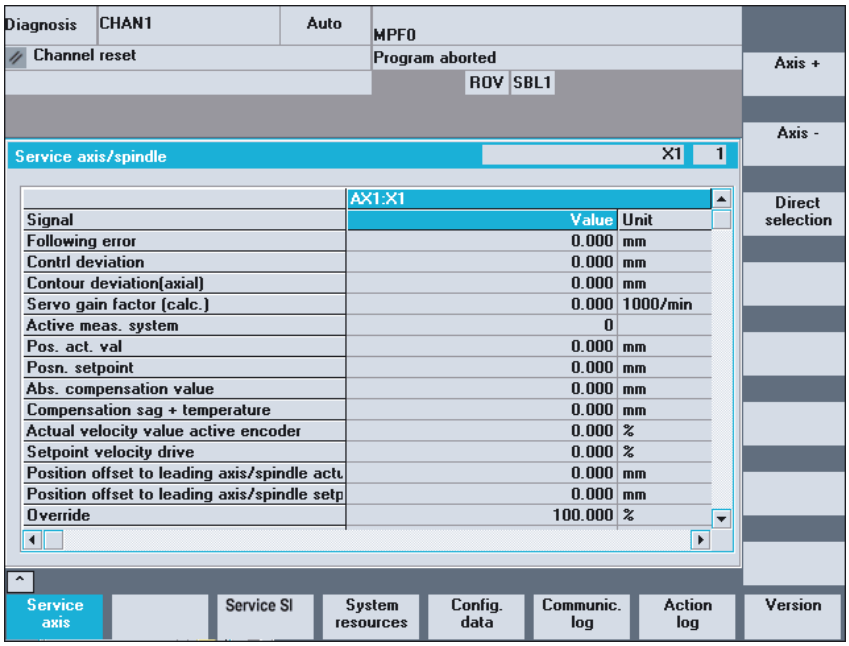


Fig. 2-1 Example for service axis/spindle HMI Advanced

Following error

The difference between the position setpoint and the actual position value of active measuring system 1 or 2.

Unit: mm, inch or degrees

Error signal

The difference between the position setpoint at the position controller input and the actual position value of active measuring system 1 or 2.

Unit: mm, inch or degrees

Contour deviation

The current contour deviation is displayed with this value (variations of the following error caused by settling operations on the speed controller due to load changes).

The contour deviation results from the difference between an actual position pre-calculated from the position setpoint and the actual position value of active measuring system 1 or 2.

Unit: mm, inch or degrees

Servo gain factor (calculated)

The servo gain factor in the display is calculated by the NC according to the following equation:

$$\text{Servo gain factor} = \frac{\text{Speed setpoint}}{\text{Following error}} \;;$$

$$\text{Unit (with standard setting):} \quad \frac{[\text{m/min}]}{[\text{mm}]} \;;$$

Velocity setpoint = setpoint currently being output to the axis/spindle.

References: /FB1/, G2, "Velocities, Setpoint/Actual Value Systems, Control Loop Control"

Active meas. system

This line indicates whether measuring system 1 or 2 is active.

Position actual value measuring system 1/2

The actual position of the axis as measured via measuring system 1/2. The position is displayed in the machine coordinate system (no work offsets or tool offsets included).

Unit: mm, inches or degrees

Position reference value

Specified position transferred from the interpolator to the position control.

Unit: mm, inches or degrees

Compensation value meas. system 1 or 2

Display of absolute compensation value for measuring system 1 or 2.

The compensation value consists of the sum of backlash and leadscrew error compensation for the actual axis position.

Unit: mm, inches or degrees

Sag and temperature compensation

Display of the compensation value calculated for the current axis position based on the total of the sag and temperature compensations.

Unit: mm, inches or degrees

Velocity actual value of active encoder (only 840Di)

Display of velocity actual value of the currently active encoder.

Velocity setpoint of drive (only 840Di)

Display of velocity setpoint of drive.

Speed actual value

The pulses supplied by the encoder are evaluated by the NC and displayed.

Unit: %

100% means maximum speed (corresponds to 10 V for analog interface; maximum speed for SIMODRIVE 611 digital, specified by machine datum:
MD1401 $MD_MOTOR_MAX_SPEED
(speed for maximum useful motor speed)).

Speed setpoint

Speed setpoint transferred to the drive (= speed setpoint from position controller and feed forward control).

Unit: %

100% corresponds to the maximum speed setpoint (10 V for an analog interface, maximum speed for SIMODRIVE 611 digital).

Spindle speed setpoint prog.

Speed setpoint programmed by the user.

Unit: rpm
e.g.: input: S1000; display: 1000 rpm

Display applies to spindles only.

Spindle speed setpoint current

Current active speed setpoint with correct sign, including calculated compensation value and any operative speed limitation (programmed by means of setting or machine data).

Unit: rpm

Display applies to spindles only.

Override

The effective correction factor of the feed or spindle correction switch is displayed.

Unit: %

Position offset for master axis/spindle actual value

The currently applicable position offset value is displayed here (relative to the actual value) if such a position offset (angular offset between master and slave axes) has been programmed for the "Synchronous spindle" function.

Unit: mm, inches, degrees

Reference: /FB/, S3, Synchronous spindle

Position offset for master axis/spindle setpoint value

The currently applicable position offset value is displayed here (relative to the setpoint) if such a position offset (angular offset between master and slave axes) has been programmed for the "Synchronous spindle" function.

Unit: mm, inches, degrees

Reference: /FB/, S3, synchronous spindle

Current gear stage

The current actual gear stage is displayed here.

With axes, this is only displayed if a spindle is assigned to the axis. This display corresponds to the IS "actual gear stage" (DB31, ... DBX16.0 to 16.2).

References: /FB1/, S1, "Spindles"

Displays which of the 6 parameter sets of the position controller is active. A parameter set changeover takes place, for example, with a gear shift.

References: /FB1/, G2, "Velocities, Setpoint/ActualValue Systems, ControlLoop Control"

Controller mode

Displays the current controller modes.

0:	Position Control
1:	Speed Control
2:	Stopping
3:	Parking
4:	Follow-up
5:	Braking

For a more detailed description of controller modes, refer to :
References: /FB1/, A2, "Various Interface Signals"

"Referenced" status display

Status display for reference point approach (axis).

Bit0=Status 0:	The machine axis is not cross-referenced using position measurement system 1 or 2.
Bit0=Status 1:	The machine axis has reached the reference point (incremental measuring system) and/or target point (length measuring system with distance coded reference marks) during reference point approach.
Bit1=Status 0:	No obligation to cross-ref. (NC start possible without cross-ref. this axis)
Bit1=Status 1:	There is no obligation to cross-reference for the NC start.

Depending on machine data
MD34110 $MA_REFP_CYCLE_NR
and
MD20700 $MC_REFP_NC_START_LOCK

Display corresponds to
IS "Referenced/synchronized" 1 or 2 (DB31, ... DBX60.4 and 60.5).

References: /FB1/, R1, "Reference Point Approach"

QEC learning active

Indicates whether or not the learning process for quadrant error compensation for the axis is active.

Fixed stop reached

Indicates whether or not the axis has fulfilled the conditions for "Fixed stop reached" when the "Travel to fixed stop" function is active (IS DB31..., DBX62.5)
References: /FB/, F1, Travel to Fixed Stop

Torque limitation value

Indicates the value programmed by means of
FXST[x] or
SD43510 $SA_FIXED_STOP_TORQUE
or the value defined with machine datum:
MD37010 $MA_FIXED_STOP_TORQUE_DEF
for the clamping torque for "Travel to fixed stop".

Unit: % of maximum torque

References: /FB/, F1, Travel to Fixed Stop

Safe actual position of the axis

Displays the current actual axis position that has been measured via the NC. This actual position should correspond in value to "Safe actual position of drive".
References: /FBSI/ Description of Functions Safety Integrated

Safe actual position of drive

Displays the current actual axis position that has been measured via the drive. This actual position should correspond in value to "Safe actual position of axis".
References: /FBSI/ Description of Functions Safety Integrated

Safe input signals of the axis

Displays the safe input signals of the PLC defined for the "Safety Integrated" function. The status of these input signals should correspond to the "Safe input signals drive" status.
References: /FBSI/ Description of Functions Safety Integrated

Safe input signals of the drive

Displays the safe input signals of the drive (DMP on drive bus) defined for the "Safety Integrated" function. The status of these input signals should correspond to the "Safe input signals of axis" status.
References: /FBSI/ Description of Functions Safety Integrated

Safe output signals of the axis

Displays the safe output signals of the PLC defined for the "Safety Integrated" function. The status of these output signals should correspond to the "Safe output signals of drive" status.
References: /FBSI/ Description of Functions Safety Integrated

Safe output signals of the drive

Displays the safe output signals of the drive (DMP on drive bus) defined for the "Safety Integrated" function. The status of these output signals should correspond to the "Safe output signals of axis" status.
References: /FBSI/ Description of Functions Safety Integrated

The following description contains more information on problems and questions that may arise.

Control technology concept

The figure below shows at which points in the controlloop the axis and spindle information is read off.

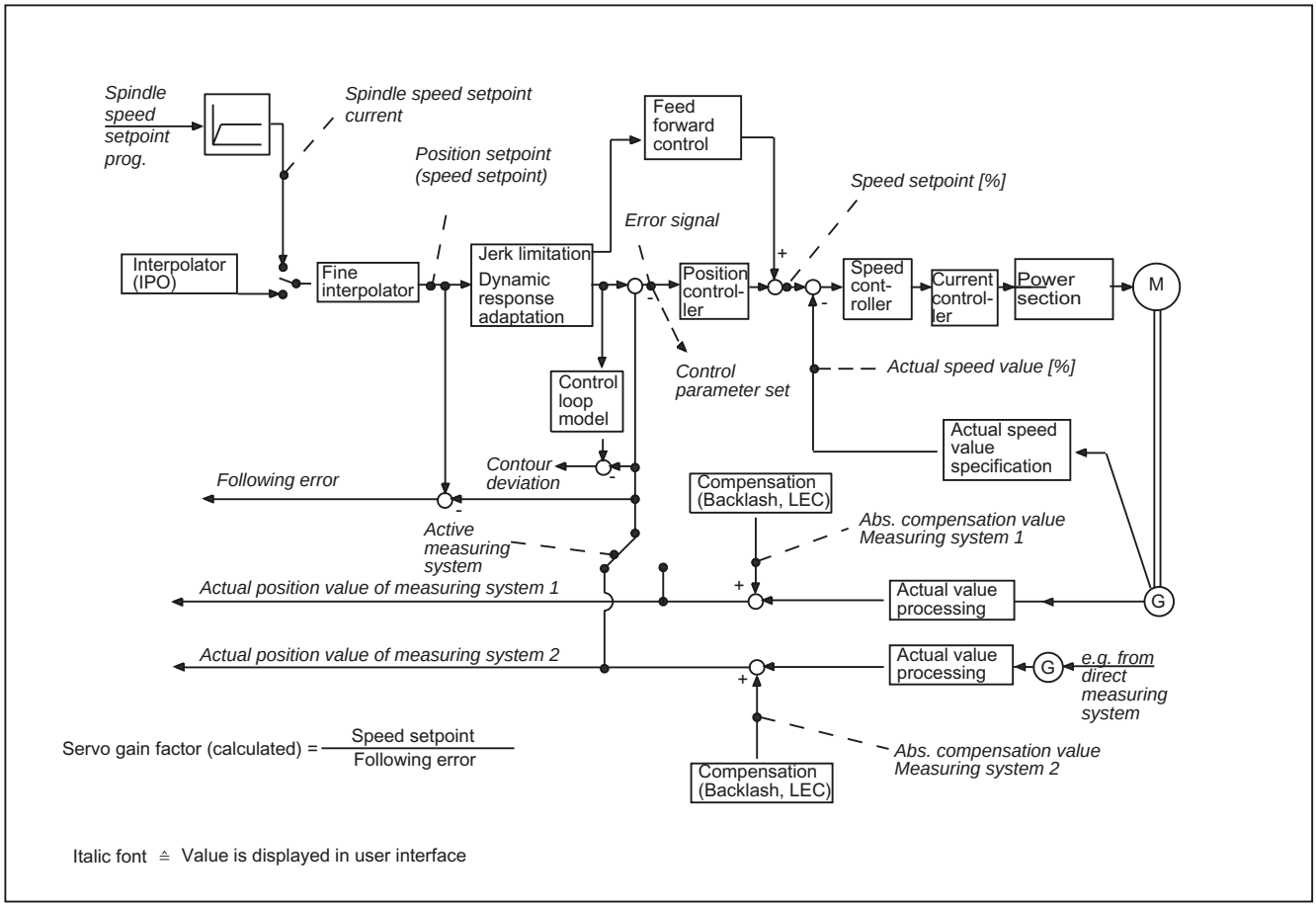


Fig. 2-2 Overview diagram of axis and spindle information

Checks

Check of the position controller setting

A simple check of the position controller setting can be made by means of the service axis screen.
To this end, the number 1 (corresponding to KV = 1) is entered in machine datum: MD32200 $MA_POSCTRL_GAIN [n] (KV factor).
The change takes effect immediately.

Because the servo gain factor is defined as

$$\text{Servo gain factor} = \frac{\text{Speed setpoint}}{\text{Following error}} \;;\; \frac{[\text{m/min}]}{[\text{mm}]} \;;\; \text{(Default)}$$

a **following error** of 1 mm must be measured (with KV = 1 and constant velocity) at a feedrate of 1 m/min.

If the desired servo gain (KV) factor does not correspond to the actual factor, the possible causes and remedial optimization options are as follows:

- Speed or torque feedforward control is activated.
  The servo gain factor displayed is higher than the setting in MD32200 $MA_POSCTRL_GAIN [n] (servo gain (KV) factor).

- Filter for jerk limitation or dynamic response adaptation is activated.
  The servo gain factor displayed is lower than the setting in MD32200 $MA_POSCTRL_GAIN [n] (servo gain (KV) factor).

Diagnostics for alarms

This information is also provided as a diagnostic tool for diagnosing the causes of alarms such as:

- "Standstill monitoring"
  ⇒ **following error >**
  MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)
- "Contour monitoring"
  ⇒ **contour deviation >**
  MD36400 $MA_CONTOUR_TOL (tolerance band for contour monitoring)
- "Speed setpoint limiting"
  ⇒ **Speed setpoint >**
  MD36210 $MA_CTRLOUT_LIMIT
  (maximum speed setpoint)
- "Positioning monitoring"
  ⇒ **following error >**
  MD36010 $MA_STOP_LIMIT_FINE
  (exact stop fine)
- "Measuring system changeover not available"
  ⇒ difference between **actual position value measuring system 1 and 2 >**
  MD36500 $MA_ENC_CHANGE_TOL
  (maximum tolerance for actual position value measurement)
- "Clamping monitoring"
  ⇒ **following error >**
  MD36050 $MA_CLAMP_POS_TOL
  (clamping tolerance for "Clamping active" interface signal)

For details of the behavior of the NC control in response to individual alarms, and remedial action, please refer to:
References: /DA/, "Diagnostics Guide"

Diagnostics of operational state errors

Moreover, with this information, faulty operating modes can be investigated, for example:

- Despite an active motion command, the axis does not move.
  ⇒ Check whether controller is enabled. In **controller mode,** position control or speed control (with spindle control) must be activated.
- Occurrence of feed fluctuations.
  ⇒ Detection via **following error** or **actual speed value.**
- Incorrect positioning.
  ⇒ Compare **position setpoint** with **actual position value of measuring system 1/2** and **absolute compensation value of measuring system 1 or 2**.
- The cam is not detected by the PLC during referencing.
  ⇒ Check **status display "referenced"**
- An incorrect reference point value was displayed.
  ⇒ The wrong measuring system may have been used for referencing.
- Large fluctuations in the **actual speed value** are occurring in the main spindle drive.
  ⇒ The selected actual speed range for encoders is too high or the machine datum: MD36300 $MA_ENC_FREQ_LIMIT [n] (encoder limiting frequency) is set higher than specified in the encoder data sheet.

Incorrect spindle positioning occurs.
⇒ The wrong measuring system may be selected or synchronization may have been performed with the wrong zero mark.

2.4 Drive service display (for digital drives only)

Displays

Displays on the control's user interface that show values and statuses allow for evaluation of the operating statuses of the digital drives.

Access

Accessing diagnostic options:

For the purposes of installing and diagnosing

- Feed drives (FDD) and
- Main spindle drives (MSD)

the information shown in the following figure can be called up for each axis/spindle in the "Diagnostics" operating area via the operator panel.

Note

The parameters in the "Drive" service display are not necessary for connecting drives via the PROFIBUS DP. For SINUMERIK 840Di, the drives are defined as PROFIBUS nodes. The appropriate service data is displayed in 840DiStartup in the menu Diagnostics --> PROFIBUS.

Application

The diagnostic options are used for:

- Checking the status of enabling and control signals
  (e.g. **pulse enable**, **drive enable**, **motor selection**, **setpoint parameter set**)
- Checking the status of FDD/MSD operating modes
  (e.g. **setup mode**, **parking axis**)
- Displaying temperature warnings
- Checking the current setpoint/actual value display
  (e.g. **actual position value measuring system 1/2**, **speed setpoint**, **actual speed value**)
- Checking the drive status (drive ready)
- Displaying the current ramp-up phase
- Displaying the group error message (**message status class 1**)
- Displaying the drive status messages
  (e.g. **threshold torque not reached, minimum speed not reached**, **actual speed = set speed**)

Service HMI Advanced Drive

Descriptions on how to select and operate the "Diagnostics" area can be found in
References: /BAD/, "HMI Advanced Operator's Guide"
/BEM/, "HMI Embedded Operator's Guide"

Note

The individual status displays, warnings, messages, etc., are explained in the following sections. For HMI SW 6 and higher, the status is shown in plain text as "yes" or "no" instead of "0" and "1".

For additional information, see:
References: /IAD/, "Installation and Start-Up Guide"

Explanations/Terms

Drive enable (terminal 64/63)

The display corresponds to the status of terminal 64/63 on the SIMODRIVE611 digital infeed/regenerative feedback unit.

State 1: Central drive enable
State 0: Central drive disable

Display corresponds to machine datum:
MD1700 $MD_TERMINAL_STATE
(status of binary inputs).

Pulse enable (terminal 63/48)

The display corresponds to the status of terminal 63/48 on the SIMODRIVE611 digital infeed/regenerative feedback unit.

State 1: Central pulse enable
State 0: Central pulse disable

Display corresponds to machine datum:
MD1700 $MD_TERMINAL_STATE
(status of binary inputs).

Pulse enable (terminal 663)

The display corresponds to the status of terminal 663 (relay: safe operational stop) on the drive module.

State 1: Module-specific pulse enable
State 0: Module-specific pulse disable

Display corresponds to machine datum:
MD1700 $MD_TERMINAL_STATE
(status of binary inputs).

Setup mode (terminal 112) HMI SW 6.3 and later

The display corresponds to the status of terminal 112 on the SIMODRIVE 611 digital infeed/regenerative feedback unit.

State 1: Central drive disable for setup mode
State 0: Central drive enable, no setup mode

Display corresponds to machine datum:
MD1700 $MD_TERMINAL_STATE
(status of binary inputs).

PLC pulse enable

Indicates whether the pulse enable from the PLC is available for the drive.

State 1: The pulses for the drive module have been disabled by the PLC.
State 0: Pulse enable for this drive is activated by the PLC.

Display corresponds to IS "Pulse enable" (DB31, ... DBX21.7).
References: /FB1/, A2, "Various Interface Signals"

Speed controller enable NC

This display indicates whether the speed controller for the drive has been enabled by the NC.

State 1: Speed controller enable = OFF
State 0: Speed controller enable = ON

Rampup function generator quick stop

Status display for rampup function generator quick stop.

State 1:	Ramp-up function generator quick stop is not active for the drive.
State 0:	Ramp-up function generator quick stop is active. The drive is stopped without a ramp function with speed setpoint = 0 and without pulse suppression.

Display corresponds to IS "Ramp-up function generator quick stop" (DB31, ... DBX92.1).
References: /FB1/, A2, "Various Interface Signals"

DC link state (on/off)

The display contains the following drive warning:

State 0: DC link voltage = ON
State 1: DC link voltage is below warning threshold.

The warning threshold corresponds to machine datum:
MD1604 $MD_LINK_VOLTAGE_WARN_LIMIT (DC link undervoltage warning threshold).

Enable pulses

Message indicating whether the drive pulses have been enabled.

State 0:	The drive module pulses are suppressed. The axis/spindle can therefore not be traversed.
State 1:	The drive module pulses are enabled. The axis/spindle can now be traversed.

Display corresponds to IS "Enable pulses" (DB31, ... DBX93.7).
References: /FB1/, A2, "Various Interface Signals"

Drive ready

Display of the current status of the selected drive.

State 0: The drive is not ready.
State 1: The drive is ready.

Display corresponds to IS "Drive Ready" (DB31, ... DBX93.5).
References: /FB1/, A2, "Various Interface Signals"

Display indicating the current rampup phase of the selected drive.

Meaning:

CRC error

Display of communications errors detected in hardware between NC and drive.

Note

If the display shows a value other than "0", please contact your SIEMENS Regional Office!

ZK1 Messages

Display indicates whether messages of status class 1 are active.

State 0: No status class 1 message is active.
State 1: One or several status class 1 messages are active.

Status class 1 messages are alarms with the following characteristics:

- They cause internal responses
  (e.g. regenerative braking, immediate pulse suppression)
- They are modal

DC link voltage

Indicates the current DC link voltage level within the drive grouping.
Unit: Volts

Speed setpoint

The displayed speed setpoint represents the unfiltered total setpoint value. It is made up of the position controller output component and the speed feed forward branch.
Unit: rpm

Display corresponds to machine datum:
MD1706 $MD_DESIRED_SPEED (desired speed).

Speed actual value

The actual value displayed represents the unfiltered actual speed value.
Unit: rpm

Display corresponds to machine datum:
MD1707 $MD_ACTUAL_SPEED (actual speed value).

Smoothed actual current value

Display of the smoothed actual current value. The torquegenerating actual current value is smoothed by a PT1 element with parameterizable time constant.
Unit: %
100 % corresponds to the maximum current of the power section.

Display corresponds to machine datum:
MD1708 $MD_ACTUAL_CURRENT (smoothed actual current value).

Motor temperature

Display of motor temperature measured via temperature sensors.
Unit: Degrees Celsius

Display corresponds to machine datum:
MD1702 $MD_MOTOR_TEMPERATURE (motor temperature).

Speed setpoint filter 1

Status display of speed setpoint smoothing function.

State 0:	No speed setpoint smoothing is active.
State 1:	Smoothing of the speed setpoint as requested by the PLC using IS "Speed setpoint smoothing" (DB31, ... DBX20.3) takes effect because speed setpoint filter 1 is configured as the low pass.

Display corresponds to IS "Speed setpoint smoothing active"
(DB31, ... DBX92.3)
and is shown in an updated form as of software version 6.3 and later.
References: /FB1/, A2, "Various Interface Signals"

2. Torque Limit

Display of active torque limit

State 0: Torque limit 1 is active.
State 1: Torque limit 2 is active.

Display corresponds to IS "Torque limit 2 active"
(DB31, ... DBX92.2).
References: /FB1/, A2, "Various Interface Signals"

Integrator disabling

This display indicates whether the speed controller integrator is active.

State 0:	The integrator of the speed controller is enabled. The speed controller functions as a PI controller.
State 1:	Deactivation of the speed controller integrator as requested by the PLC using IS "Integrator disable n-controller" (DB 31, ... DBX 21.6) is active for the drive module. The speed controller has therefore switched from a PI to a P controller.

Display corresponds to IS "n-controller integrator disabled"
(DB31, ... DBX93.6).
References: /FB1/, A2, "Various Interface Signals"

Setup mode

Mode display of the SIMODRIVE 611 digital.

State 0: Normal mode is active for the drive.
State 1: Setup mode is active for the drive.

Display corresponds to IS "Setup mode active"
(DB31, ... DBX92.0).
References: /FB1/, A2, "Various Interface Signals"

Parking axis

Mode display of the SIMODRIVE 611 digital.

State 0:	Axis/spindle in normal mode
State 1:	Axis/spindle in parking position, i.e. all encoder-specific monitoring and evaluating functions are disabled. This allows the encoder to be withdrawn without initiating an alarm.

Setpoint parameter set (drive)

Indicates which of the 8 drive parameter sets of the SIMODRIVE 611 digital are to be activated by the PLC.

Display corresponds to IS "Parameter set selection A,B,C"
(DB31, ... DBX21.0 to 21.2).
References: /FB1/, A2, "Various Interface Signals"

Actual parameter set (drive)

Display indicating which of the 8 drive parameter sets of the SIMODRIVE 611 digital is currently active.

Display corresponds to IS "Active parameter set A,B,C"
(DB31, ... DBX93.0 to 93.2).
References: /FB1/, A2, "Various Interface Signals"

Operating mode

Display indicating whether the motor is operating as a feed drive or main spindle drive.

Motor selection (star/delta)

Display indicating which motor data set is to be activated by the PLC. At the moment the motor data record is used for the star/delta switchover on main spindle drives.

The following assignment applies:

Motor selection	Application	Coding	
Motor 1	MSD: Star mode	0	0
Motor 2	MSD: Delta mode	0	1
Motor 3	Reserved	1	0
Motor 4	Reserved	1	1

The display applies only to MSD drives.
The display corresponds to IS "Motor selection A,B"
(DB31, ... DBX21.3 to 21.4).
References: /FB1/, A2, "Various Interface Signals"

Actual motor (star/delta)

Display indicating which of the motor data sets is currently active. At the moment the motor data record is used for the star/delta switchover on main spindle drives.

The following assignment applies:

Motor selection	Application	Coding	
Motor 1	MSD: Star mode	0	0
Motor 2	MSD: Delta mode	0	1
Motor 3	Reserved	1	0
Motor 4	Reserved	1	1

The display applies only to MSD drives.
The display corresponds to IS "Active motor A,B"
(DB31, ... DBX93.3 to 93.4).
References: /FB1/, A2, "Various Interface Signals"

Position actual value measuring system 1/2

The actual position of the axis as measured via measuring system 1/2. The position is displayed in the machine coordinate system (no work offsets or tool offsets included).

Unit: mm, inches or degrees

Power section in i²t limitation HMI SW 6.3 and later

Limitation for protecting the power section against continuous overloading of the SIMODRIVE 611 drives.

State 1: i^2t power section limitation has responded
State 0: i^2t power section limitation has not responded

The display applies to SIMODRIVE universal and SIMODRIVE digital.
The display corresponds to IS "i^2t monitoring"
(DB31, ... DBX95.7).
References: /FB1/, A2, "Various Interface Signals"

Heatsink temperature warning

Warning signal output by drive:

State 0: The heatsink temperature monitoring has not responded.
State 1: The heatsink temperature monitoring has responded.

Display corresponds to IS "Heatsink-temperature prewarning"
(DB31, ... DBX94.1).
References: /FB1/, A2, "Various Interface Signals"

Motor temperature warning

Warning signal output by drive:

State 0: The motor temperature is below the warning threshold.
State 1: The motor temperature has exceeded the defined warning threshold.

The warning threshold corresponds to machine datum:
MD1602 $MD_MOTOR_TEMP_WARN_LIMIT (maximum motor temperature).

Display corresponds to IS "Motor-temperature prewarning"
(DB31, ... DBX94.0).
References: /FB1/, A2, "Various Interface Signals"

Ramp-up function completed

Status display of drive.

State 0:	The ramp-up function has not yet been completed after a new speed setpoint was defined.
State 1:	The actual speed value has reached the speed tolerance band after a new speed setpoint was defined.

The speed tolerance band corresponds to machine datum:
MD1426 $MD_SPEED_DES_EQ_ACT_TOL (tolerance band for '$n_{desired}$-n_{actual}' message).

Display corresponds to IS "Ramp-up function completed"
(DB31, ... DBX92.2).
References: /FB1/, A2, "Various Interface Signals"

Torque lower than threshold setting

Status display of drive.

State 0:	In the stationary condition (i.e. ramp-up procedure completed), the torque setpoint is greater than the threshold torque.
State 1:	In the stationary condition, the torque setpoint has not reached the threshold torque.

The threshold torque corresponds to machine datum:
MD1428 $MD_TORQUE_THRESHOLD_X (threshold torque).

The display corresponds to IS "$M_d < M_{dx}$"
(DB31, ... DBX94.3).
References: /FB1/, A2, "Various Interface Signals"

Speed lower than minimum setting

Status display of drive.

State 0: The actual speed value is greater than the minimum speed.
State 1: The actual speed value is smaller than the minimum speed.

The minimum speed corresponds to machine datum:
MD1418 $MD_SPEED_THRESHOLD_MIN ($n_{min}$ for '$n_{actual} < n_{min}$' message).

The display corresponds to IS "$|n_{actual}| < n_{min}$"
(DB31, ... DBX94.4).
References: /FB1/, A2, "Various Interface Signals"

Speed lower than threshold setting

Status display of drive.

State 0: The actual speed value is greater than the threshold speed.
State 1: The actual speed value is smaller than the threshold speed.

The threshold speed corresponds to machine datum:
MD1417 $MD_SPEED_THRESHOLD_X
(n_x for '$n_{actual} < n_x$' message).

The display corresponds to IS "$|n_{actual|} < n_x$"
(DB31, ... DBX94.5).
References: /FB1/, A2, "Various Interface Signals"

Actual speed = set speed

Status display of drive.

State 0:	The actual speed value is outside the speed tolerance band after a new speed setpoint was defined.
State 1:	The actual speed value has reached the speed tolerance band after a new speed setpoint was defined.

The speed tolerance band corresponds to machine datum:
MD1426 $MD_SPEED_DES_EQ_ACT_TOL (tolerance band for 'n_{set}-n_{actual}' message).

The display corresponds to IS "$|n_{actual}|= n_{setpoint}$"
(DB31, ... DBX94.6).
References: /FB1/, A2, "Various Interface Signals"

Variable signal 1

Status display of variable signaling function of 611D.
The variable signal function monitors a memory location for violation of the defined threshold. In addition to the threshold, a tolerance band can be defined which is also taken into account when scanning for violation of the threshold value. The signal can be combined with an ON delay or OFF delay time.

State 0: Threshold value not reached
State 1: Threshold value exceeded

Parameters for the variable signal function are set using the following 611D machine data:

MD1620 $MD_PROG_SIGNAL_FLAGS	(Bits variable signal function)
MD1621 $MD_PROG_SIGNAL_NR	(Signal number variable signal function)
MD1622 $MD_PROG_SIGNAL_ADDRESS	(Address variable signal function)
MD1623 $MD_PROG_SIGNAL_THRESHOLD	(Threshold variable signal function)
MD1624 $MD_PROG_SIGNAL_HYSTERESIS	(Hysteresis variable signal function)
MD1625 $MD_PROG_SIGNAL_ON_DELAY	(ON Delay variable signal function)
MD1626 $MD_PROG_SIGNAL_OFF_DELAY	(OFF Delay variable signal function)

Diagnostics for alarms

This information is also provided as a diagnostic tool for diagnosing the causes of alarms such as:

- "Ramp-up error"
  ⇒ Check the **ramp-up phase** to see which ramp-up phase the drive has reached.
- "Drive malfunction"
  ⇒ **Message ZK1** is set.
  ⇒ Check **specified parameter set**, **motor selection**, **DC link status**.
  Check following alarms 300500 to 300515.

- "Zero speed monitoring",
  25050 "Contour monitoring",
  25060 "Speed setpoint limitation"
  25080 "Positioning monitoring"
  ⇒ The enabling for the drive may have been omitted (e.g. **pulse enable**, **drive enable**, **pulse enable for PLC** not available); this leads to display **pulses enabled** = off.
- "Motor temperature exceeded"
  ⇒ Check the current motor temperature.

For details on the behavior of the NC control in response to individual alarms, and remedial action, please refer to:

References: /DA/, Diagnostics Guide.

2.5 Service display PROFIBUS DP 840Di

The user interface 840Di StartUp provides diagnostic screen forms for PROFIBUS DP and its nodes. These diagnostic screens are only intended for information. You cannot modify them.

The following detailed information is displayed:

- PROFIBUS configuration
- Information on the slaves regarding their assignment to PLC/NC
- Detailed information on the slaves and the corresponding slots
- Information on the axes

To obtain a quick overview, the current states of certain functions are represented by colored lamps. The following general conventions are used for the meaning of the individual colors:

- Green: Function is OK
- Red: Failure or no communication at the moment
- Gray: Function is not available for the present communication

Diagnostic screen PROFIBUS DP Configuration

The diagnostic screen PROFIBUS Configuration provides general information on PROFIBUS DP.

The following parameters are displayed:

Table 2-1 Diagnostic screen PROFIBUS configuration

Function/subfunction	Explanation/meaning
Bus configuration	
Baud rate in MBd	Data transfer rate
Cycle time in msec	Configured bus cycle time; also defines the position controller cycle
Synchronous portion (TDX) in msec	Configured time for cyclic data exchange within a PROFIBUS DP cycle
Status	
Configuration OK.	Status of configuration • Green lamp: DP master has powered up. • Red lamp: Failure or no communication.
Bus status	Current bus status is displayed in this field. Each bus status is explained in the screen form in brief. Possible states are: • POWER_ON • OFFLINE • CLEAR • OPERATE • ERROR

Diagnostic screen of the DP slaves

These diagnostic screen forms provide an overview of the DP slaves configured and detected on the bus.

The following information is provided:

Table 2-2 Diagnostic screen information on the slaves

Function/subfunction	Explanation/meaning
Slave No. [DP address]	Configured DP address of DP slave
Assignment	It is displayed whether the DP slave is assigned to the NC or to the PLC. • NC: e.g. one or several drives controlled by the NC. • PLC: e.g. I/O modules or an axis controlled by the PLC.
Active on the bus	Displays whether the DP slave has been detected on the bus • Green lamp: DP slave has been detected on PROFIBUS DP and the data exchange with the assigned component (NC or PLC) operates. • Red lamp: Failure or no communication.
Sync. with NC	Displays whether DP slave operates on the bus synchronously to the NC. • Green lamp: DP slave operates on PROFIBUS DP synchronously to the NC, i.e. the equidistant data exchange takes place. • Gray lamp: DP slave is not assigned to the NC, but to the PLC. • Red lamp: Failure or no communication.
Number of slots	Number of configured slots within DP slave
Details	Pressing this button will open another diagnostic screen that provides detailed information on the corresponding DP slave.

Detailed information of the slots within a slave

The **Details** button opens the diagnostics dialog box Detailed information on the slave. This screen form provides detailed information on the slots assigned to the DP slave.

In addition, the Slave dialog box displays important information on the DP slave currently selected.

The following information is displayed for the slots:

Table 2-3 Diagnostic screen detailed information on the slave

Function/subfunction	Explanation/meaning
Slaves	
Slave No. [DP address]	Configured DP address of DP slave
Assignment	Displays whether the DP slave is assigned to the NC or to the PLC. • NC: e.g. one or several drives controlled by the NC. • PLC: e.g. I/O modules or an axis controlled by the PLC.
Active on the bus	Displays whether the DP slave has been detected on the bus • Green lamp: DP slave has been detected on PROFIBUS DP and the data exchange with the assigned component (NC or PLC) operates. • Red lamp: Failure or no communication.
Synchr.	Displays whether DP slave operates on the bus synchronously to the NC. • Green lamp: DP slave operates on PROFIBUS DP synchronously to the NC, i.e. the equidistant data exchange takes place. • Gray lamp: DP slave is not assigned to the NC, but to the PLC. • Red lamp: Failure or no communication.
Slots	
No.	Slot number within DP slave
I/O address	I/O address in the I/O address space of the PLC assigned to this slot. For NC axes, setpoint and actual value must always be configured using the same I/O address.
Logical drive no.	Drive number assigned for the appropriate axis in the NC machine data.
Length [bytes]	Length of the I/O area in the STEP7 I/O address space, which is reserved for the slot
Type	Specification whether the slot is an input, output or diagnostic slot. If the slot is assigned to an NC axis, an output is always designated as a setpoint and an input always as an actual value.
Machine axes	Display of the name defined for this slot in the machine data. If the slot is not assigned to any NC axis, <No NC axis> is displayed.
Message frame type	Message frame type configured in the NC machine data. If the slot is not assigned to any NC axis, the message frame type will not be occupied (-).
Status	Current slot status is only displayed for NC axes • Green: Slot is used by the NC; communication active • Gray: No NC axis • Red: Slot is used by the NC; communication currently not active

Diagnostic screen for the axes

The diagnostic screen AxisInfo displays axisspecific detailed information. The diagnostic screen provides an NCoriented view of the axis information.

The following information is displayed for the axes:

Table 2-4 Diagnostic screen AxisInfo

Function/subfunction	Explanation/meaning
Machine axes	Name of the axis defined in the NC machine data
Output	
Slave/slot	Configured assignment
Status	Current slot status Green lamp: Cyclic communication Red lamp: No cyclic communication (as yet)
Message frame failures	The display shows the number of message frame failures since the NC was booted. This value indicates the quality (fault susceptibility) of the PROFIBUS DP line.
Encoder 1	
Slave/slot	Configurable assignment
Status	Current slot status Green lamp: Cyclic communication Red lamp: No cyclic communication (as yet)
Message frame failures	The display shows the number of message frame failures since the NC was booted. This value indicates the quality (fault susceptibility) of the PROFIBUS DP line.
Type	Display of encoder type configured in the NC machine data ABS: Absolute encoder INC: Incrementalvalue encoder
Encoder 2	(If configured, the same display as for encoder 1)

2.6 Communication log

Log assistance

In event of a fault and when developing OEM applications, control logs may assist with the analysis.

Logs and version

Communication log

The communication errors which have occurred between the HMI and NC are displayed in chronological order under the soft key **Comm. log** in the "Diagnostics" operating area. This error list assists developers of OEM applications in localizing sporadic errors. The list has no relevance for normal operation.

Logbook

In the "Diagnostics" operating area, under the soft key **Logbook**, all system-relevant modifications to the controller are automatically noted in the Logbook display (e.g. changes to access levels).

For SINUMERIK 840Di, the log book is displayed in 840Di StartUp.

Version

In the event of service work, ("Diagnostics" operating area, under the soft key **Version**), the built-in HMI or NC software status is read out.
The software status of each software module is also kept as a list.

2.7 PLC status

PLC status signals can be checked and altered via the operator panel in the "Diagnostics" operating area.

Application

The end customer or service personnel can use this function on site without a programming device to do the following:

- Check the input and output signals of the PLC I/Os.
- Carry out limited troubleshooting.
- Check the interface signals for diagnostic purposes.

Operation

For information on how to operate the service displays and alter them, see:
References: /BA/, "Operator's Guide"

Status display

The status of the following data can be displayed on the operator panel.

- Interface signals from the machine control panel
- Interface signals to the machine control panel
- Interface signals between the NCK and PLC
- Interface signals between the MMC and PLC
- Data blocks (DB 0-127)
- Flags (FB 0-255)
- Timers (T 0-127)
- Counters (C 0-63)
- Inputs (IB 0-127)
- Outputs (QB 0-127)

For a breakdown of the interface signals (DBx, DBBy), refer to:
References: /IAD/, "Installation and Start-Up Guide"
/IAF/, "Installation and Start-Up Guide"

Change in status

The status of the above signals can be changed for test purposes. Signal combinations are also possible. A maximum of ten operands can be altered at any one time.

2.8 Other diagnostics tools

611D startup tool

Using the 611D startup tool and archiving software, the control can be evaluated and the control status can be saved.

611D startup tool

One of the functions of this program is to provide a tool

- For evaluating the most important values for the position, speed and current control
- For archiving drive and control data, and
- For analyzing the stated physical properties.

For handling and complete range of functions see:
References: /IAD/, "Installation and Start-Up Guide"

Archiving of data

The PCIN software package can be used to archive machine data, setting data, part programs, etc.

A description of how to use this can be found in the associated documentation.
References: /PI/ PCIN 4.3

840Di StartUp

For diagnosing the SINUMERIK 840Di, the WINDOWS program 840Di StartUp can be used. This provides information, e.g. on the current operating mode and the nodes of PROFIBUS DP.

2.9 Identifying defective drive modules

Deactivate drives

Drives can be removed from the NC configuration using a piece of machine data.

Troubleshooting may involve a situation where a drive module (SIMODRIVE 611 digital) displayed in an alarm text needs to be removed from the bus in order to determine whether this module has caused the displayed error.

With machine datum:
MD13030 $MN_DRIVE_MODULE_TYPE,
individual modules can be removed from the NC-side drive bus configuration (the affected axes are switched to simulation).

Note

You must remove the desired module from the drive bus configuration (SIMODRIVE 611 digital) before you activate the function. To do this, connect the drive so as to exclude the module.

Since this internal modification to the machine configuration can result in damage to the machine if implemented incorrectly, the axes are prevented from moving.

If Safety Integrated has been activated for the modules concerned, you must disable it manually (safety, logged, EMERGENCY STOP scheme).

Remove drive module at NC end

A drive module (SIMODRIVE 611 digital) specified in an alarm text must be removed from the bus:

1. Remove the module from the drive bus network.
2. Set entries of the drive module in machine datum:
   MD13030 $MN_DRIVE_MODULE_TYPE
   to zero (zero-axis module).
3. Perform an NC RESET.

The axes which were controlled by the removed drive modules are now replaced by simulated axes. The 611D bus with its drive modules is now in a state in which it could normally move axes, but axis traversal has been disabled internally.
Alarm 300020 "Drive %1 removed for diagnostics" is displayed to indicate this status to the operator.

Restoring the initial configuration

After completing the diagnostics, the initial configuration on the drive bus must be restored:

1. Replace or re-install the removed drive module.
2. Change entries of the drive module in machine datum:
   MD13030 $MN_DRIVE_MODULE_TYPE
   back to the original values.
3. Perform an NC RESET.

Example

The 2axis module with drive numbers "1" and "2" must be removed from a drive grouping.

Note

Before activating the function, the module in question must be removed from the drive bus configuration

(SIMODRIVE 611 digital). To do this, connect the drive bus so as to exclude the module.

If Safety Integrated has been activated for the modules concerned, you must disable it manually (safety, logged, EMERGENCY STOP scheme).

Table 2-5 Bus configuration example

Module	Drive no.	Active	Type	Module type	Power section code
1	10	1	ARM/MSD	Axis	6
Left	1	1	SRM/FDD	Axis	14
Right	2	1	SRM/FDD	Axis	14
Left	4	1	HLA	Axis	
Right	5	1	ANA	Axis	
4	12	1	SLM	Axis	11
5	11	1	PER	DMPC	

Module "2" must now be removed:

- Machine datum:
  MD13030 $MN_DRIVE_MODULE_TYPE
  is to be selected on the "General MD" MD screen.
- DRIVE_MODULE_TYPE[0] = 1
  DRIVE_MODULE_TYPE[1] = 2 <- set this entry to zero
  DRIVE_MODULE_TYPE[2] = 2 <- set this entry to zero
  DRIVE_MODULE_TYPE[3] = 2
  DRIVE_MODULE_TYPE[4] = 2
  DRIVE_MODULE_TYPE[5] = 1
  DRIVE_MODULE_TYPE[6] = 9
- After the changes, the table looks like this:
  DRIVE_MODULE_TYPE[0] = 1
  DRIVE_MODULE_TYPE[1] = 0
  DRIVE_MODULE_TYPE[2] = 0
  DRIVE_MODULE_TYPE[3] = 2
  DRIVE_MODULE_TYPE[4] = 2
  DRIVE_MODULE_TYPE[5] = 1
  DRIVE_MODULE_TYPE[6] = 9
- Alarms 300020 "Drive 1 removed for diagnostics" and 300020 "Drive 2 removed for diagnostics" are displayed.

Internally simulated drives are used for all axes which had settings on the removed drive numbers. If the controller is engaged for the drives that are still installed, these drives operate in the normal way. Interpolative traversal of all axes is disabled.

Note

If alarm 300003 "Axis xx drive yy incorrect module type zz" appears, then you have removed only one part of a 2axis module. In this case, you should check the module type in the drive configuration display. For removed modules, "NO" axis type is shown.

Supplementary Conditions

3

3.1 Secondary conditions for diagnostic tools

Availability of "Identification of defective drive modules"

Availability

Identification of defective drive modules

The "Identification of defective drive modules" function is available in SW 6.3 and later for SINUMERIK 840D/810D.

Examples

4

4.1 Examples

No examples are available.

Data Lists

5

5.1 Machine data

5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
11410	SUPPRESS_ALARM_MASK	Mask for suppressing special alarms
11411	ENABLE_ALARM_MASK	Activation of special alarms
11412	ALARM_REACTION_CHAN_NOREADY	Alarm reaction CHAN_NOREADY permitted
11413	ALARM_PAR_DISPLAY_TEXT	Texts as alarm parameters (Siemens Rights)
11420	LEN_PROTOCOL_FILEX	File size for protocol files (KB)
13030	DRIVE_MODULE_TYPE	Module identifier (SIMODRIVE 611 digital)

5.1.2 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
32200	POSCTRL_GAIN [n]	Servo gain factor
32250	RATED_OUTVAL	Rated output voltage
32260	RATED_VELO	Rated motor speed
36010	STOP_LIMIT_FINE	Exact stop fine
36030	STANDSTILL_POS_TOL	Zero speed tolerance
36050	CLAMP_POS_TOL	Clamping tolerance
36210	CTRLOUT_LIMIT	Maximum speed setpoint
36300	ENC_FREQ_LIMIT[n]	Encoder limit frequency
36400	CONTOUR_TOL	Tolerance band contour monitoring
36500	ENC_CHANGE_TOL	Max. tolerance for actual position value acquisition
37010	FIXED_STOP_TORQUE_DEF	Clamping torque
43510	FIXED_STOP_TORQUE	Fixed stop clamping torque

5.1.3 Drive-specific machine data

Number	Identifier: $MD_	Description
1401	MOTOR_MAX_SPEED	Speed for max. useful motor speed
1417	SPEED_THRESHOLD_X	n_x for '$n_{act} < n_x$' signal
1418	SPEED_THRESHOLD_MIN	n_{min} for '$n_{act} < n_{min}$' signal
1426	SPEED_DES_EQ_ACT_TOL	Tolerance band for 'n_{set}-n_{act}' signal
1428	TORQUE_THRESHOLD_X	Threshold torque
1602	MOTOR_TEMP_WARN_LIMIT	Maximum motor temperature
1604	LINK_VOLTAGE_WARN_LIMIT	DC link under voltage warning threshold
1620	PROG_SIGNAL_FLAGS	Bits variable signal function
1621	PROG_SIGNAL_NR	Signal number variable signal function
1622	PROG_SIGNAL_ADDRESS	Address variable signal function
1623	PROG_SIGNAL_THRESHOLD	Threshold variable signal function
1624	PROG_SIGNAL_HYSTERESIS	Hysteresis variable signal function
1625	PROG_SIGNAL_ON_DELAY	ON Delay variable signal function
1626	PROG_SIGNAL_OFF_DELAY	OFF Delay variable signal function
1700	TERMINAL_STATE	Status of binary inputs
1702	MOTOR_TEMPERATURE	Motor temperature
1706	DESIRED_SPEED	Speed setpoint
1707	ACTUAL_SPEED	Speed actual value
1708	ACTUAL_CURRENT	Smoothed current actual value

5.2 Setting data

5.2.1 Axis/spindle-specific setting data

Axis-specific setting data

Number	Identifier	Name
Axis-specific setting data ($SA_....)		
43510	FIXED_STOP_TORQUE	Fixed stop clamping torque

5.3 Signals

5.3.1 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	16.0 - 16.2	Actual gear stages A, B, C
31, ...	21.0 - 21.2	Parameter set selection A, B, C
31, ...	21.3 - 21.4	Motor selection A, B
31, ...	21.7	Pulse enable

5.3.2 Signals from axis/spindle

DB number	Byte.Bit	Description
31, ...	60.4	Referenced/synchronized 1
31, ...	60.5	Referenced/synchronized 2
31, ...	62.5	Fixed stop reached
31, ...	92.0	Setup mode active
31, ...	92.1	Rampup function generator quick stop
31, ...	92.2	Torque limit 2 active
31, ...	92.3	Speed setpoint smoothing active
31, ...	93.0 - 93.2	Active parameter set A, B, C
31, ...	93.3 - 93.4	Active motor A, B
31, ...	93.5	Drive Ready
31, ...	93.6	Speed controller integrator disabled
31, ...	93.7	Pulses enabled
31, ...	94.0	Motor temperature prewarning
31, ...	94.1	Heat-sink temperature prewarning
31, ...	94.2	Ramp-up function completed
31, ...	94.3	_'$M_{d_}$'< M_{dx}
31, ...	94.4	_'$n_{act_}$'< n_{min}
31, ...	94.5	_'$n_{act_}$'< n_x
31, ...	94.6	_'$n_{act_}$'= n_{set}

Index

6

611D startup tool, 2-30

A

Axis/spindle service display, 2-5

C

Communication log, 2-29

D

DB31, ...
- DBX16.0, 2-9
- DBX16.1, 2-9
- DBX16.2, 2-9
- DBX20.3, 2-19
- DBX21.0, 2-20
- DBX21.1, 2-20
- DBX21.2, 2-20
- DBX21.3, 2-21
- DBX21.4, 2-21
- DBX21.7, 2-17
- DBX60.4, 2-10
- DBX60.5, 2-10
- DBX62.5, 2-10
- DBX92.0, 2-20
- DBX92.1, 2-17
- DBX92.2, 2-19
- DBX92.3, 2-19
- DBX93.0, 2-20
- DBX93.1, 2-20
- DBX93.2, 2-20
- DBX93.3, 2-21
- DBX93.4, 2-21
- DBX93.5, 2-18
- DBX93.6, 2-20
- DBX93.7, 2-17
- DBX94.0, 2-22
- DBX94.1, 2-22
- DBX94.2, 2-22
- DBX94.3, 2-23
- DBX94.4, 2-23
- DBX94.5, 2-23
- DBX95.7, 2-22

DB31, ... DBX94.6, 2-24
Diagnostic tools (D1), 1-1
- Interrupts, 2-1

Diagnostics, 2-5
Drive service display, 2-14

L

Logbook, 2-29

M

MD13030 $MN_DRIVE_MODULE_TYPE, 2-31, 2-32, 2-33
MD1401 $MD_MOTOR_MAX_SPEED, 2-8
MD1417 $MD_SPEED_THRESHOLD_X, 2-23
MD1418 $MD_SPEED_THRESHOLD_MIN, 2-23
MD1426 $MD_SPEED_DES_EQ_ACT_TOL, 2-22, 2-24
MD1428 $MD_TORQUE_THRESHOLD_X, 2-23
MD1604 $MD_LINK_VOLTAGE_WARN_LIMIT, 2-17
MD1620 $MD_PROG_SIGNAL_FLAGS, 2-24
MD1621 $MD_PROG_SIGNAL_NR, 2-24
MD1622 $MD_PROG_SIGNAL_ADDRESS, 2-24
MD1623 $MD_PROG_SIGNAL_THRESHOLD, 2-24
MD1624 $MD_PROG_SIGNAL_HYSTERESIS, 2-24
MD1625 $MD_PROG_SIGNAL_ON_DELAY, 2-24
MD1626 $MD_PROG_SIGNAL_OFF_DELAY, 2-24
MD1700 $MD_TERMINAL_STATE, 2-16
MD1702 $MD_MOTOR_TEMPERATURE, 2-19
MD1706 $MD_DESIRED_SPEED, 2-18
MD1707 $MD_ACTUAL_SPEED, 2-19
MD1708 $MD_ACTUAL_CURRENT, 2-19
MD20700 $MC_REFP_NC_START_LOCK, 2-10
MD32200 $MA_POSCTRL_GAIN, 2-12, 2-13
MD34110 $MA_REFP_CYCLE_NR, 2-10
MD36010 $MA_STOP_LIMIT_FINE, 2-13
MD36030 $MA_STANDSTILL_POS_TOL, 2-13
MD36050 $MA_CLAMP_POS_TOL, 2-13

MD36210 $MA_CTRLOUT_LIMIT, 2-13
MD36300 $MA_ENC_FREQ_LIMIT, 2-14
MD36400 $MA_CONTOUR_TOL, 2-13
MD36500 $MA_ENC_CHANGE_TOL, 2-13
MD37010 $MA_FIXED_STOP_TORQUE_DEF, 2-10

N

NCK alarm handler, 2-2

O

Of defective drive modules
    Identification, 2-31

P

PCIN, 2-31
PLC status, 2-29

R

Ramp-up phase, 2-24

S

Service display PROFIBUS DP, 2-25
Servo gain factor (Kv), 2-13

V

Version, 2-29

Z

ZK1 Messages, 2-24

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Travel to Fixed Stop (F1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 General functionality 2-1
2.1.1 Functional sequence, programming, parameterization 2-1
2.1.2 Response to RESET and function abort 2-8
2.1.3 Block search response 2-9
2.1.4 Miscellaneous 2-14
2.1.5 Supplementary conditions for expansions 2-17
2.1.6 Travel with limited moment/force FOC: 2-20

2.2 Travel to fixed stop with analog drives 2-23
2.2.1 SIMODRIVE 611 digital (VSA/HSA) 2-23
2.2.2 Travel to fixed stop with hydraulic drives SIMODRIVE 611 digital (HLA module) 2-27

2.3 Travel to fixed stop with analog drives 2-27
2.3.1 SIMODRIVE 611 analog (FDD) 2-27
2.3.2 SIMODRIVE 611 analog (FDD) 2-30
2.3.3 Diagrams for travel to fixed stop with analog drives 2-34

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Travel to fixed stop 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 Axis/spindle-specific machine data 5-1

5.2 Setting data 5-2
5.2.1 Axis/spindle-specific setting data 5-2

5.3 Signals 5-2
5.3.1 Signals to axis/spindle 5-2

Index Index-1

Tables

Table 2-1 Controlling the initial setting of the modal limitation of torque/force 2-20

Brief Description

1

1.1 Brief description

Customer benefit

The "Travel to fixed stop" function can be used for operations such as traversing tailstocks or sleeves to an end limit position in order to clamp workpieces.

Features

- The clamping torque and a fixed stop monitoring window can be programmed in the parts program and can also be altered via setting data once the fixed stop has been reached.
- The "travel to fixed stop" function can be implemented for axes as well as for spindles with axistraversing capability.
- The function can be implemented for several axes simultaneously and parallel to the motion of other axes.
- Torques or the power can be adjusted to a specific setting.
- Travel with limited torque/power (Force Control, FOC) can be activated.
- The "travel to fixed stop" functions can be enabled from synchronized actions.
- Block search with calculation, multichannel (SERUPRO).

  Move axes with FXS and FOC in simulation mode.
- Vertical axes can also be moved with FXS to a fixed stop.
- With SW 6.4 and higher, VDI signals can be used to set a REPOS offset for each axis and display the FXS status currently active on the machine after the search target has been located.

Detailed Description 2

2.1 General functionality

2.1.1 Functional sequence, programming, parameterization

Programming

Travel to fixed stop is selected or deselected with the following commands:

`FXS[Machine axis identifier]=1` (selected)

`FXS[Machine axis identifier]=0` (deselected)

The commands are modal.

The clamp torque is set with the command:
`FXST[Machine axis identifier] = <torque>.`

It is entered in % of the static torque for feed spindle drives:
(MD1118 MOTOR_STANDSTILL_CURRENT)
or in % of motor torque for main spindle drives:
(MD1103 MOTOR_NOMINAL_POWER).

The command is used for setting the width of the fixed stop monitoring window.

`FXSW[machine axis identifier] = <monitoring window>`

The unit is dependent on the default setting: mm, inch or degrees.

Channel axis identifier

Instead of the machine axis identifiers, it is also possible to use channel axis identifiers if the channel axis identifiers are assigned exactly to one machine axis.

Restrictions:

Channel identifiers may not be used (option disabled) for machine axes which have an active transformation or frame.

In cases where the machine axis is acting as a coupled axis (e.g. slave axis), the system disables programming of channel axis identifiers and alarm 14092 "Wrong axis type" is displayed.

The movement to the destination point can be described as a path or positioning axis movement. With positioning axes, the function FXS can be performed across block boundaries. The function may also be selected for several machine axes simultaneously. The FXST and FXSW commands are optional.
The travel path and the command which activates that function **must be programmed in one block** (exception: Synchronized actions).

Examples

With machine axis identifiers:

```
X250 Y100 F100 FXS[X1]=1
X250 Y100 F100 FXS[X1]=1 FXST[X1]=12.3
X250 Y100 F100 FXS[X1]=1 FXST[X1]=12.3 FXSW[X1]=2                ; mm
X250 Y100 F100 FXS[X1]=1 FXSW[X1]=2                              ; mm
```

References:
/PG/ "Programming Guide: Fundamentals"

Channel axis identifier with unambiguous machine axis assignment:

For the purpose of illustrating the differences in programming, chanel axis X is programmed as the image of machine axis AX1 [or X1 (name in machine parameter: MD10000 $MN_AXCONF_MACHAX_NAME_TAB)].

Programming with machine axis identifiers

```
FXS[X1] = 1                ;  Selecting X1
FXST[X1] = 10              ;  New torque for X1
FXSW[X1] = 5               ;  New window for X1
```

Programming with channel axis identifiers

```
FXS[X] = 1                 ;  Selecting X1 -> X1
FXST[X] = 10               ;  New torque X1 -> X1
FXSW[X] = 5                ;  New window for X -> X1
```

All four of the following programming lines have the same effect when the channel axis X is imaged on the machine axis AX1, X1:

```
Z250 F100 FXS[AX1]=1 FXST[AX1]=12.3 FXSW[AX1]=2000
Z250 F100 FXS[X1]=1 FXST[X1]=12.3 FXSW[X1]=2000
Z250 F100 FXS[X]=1 FXST[X]=12.3 FXSW[X]=2000
Z250 F100 FXS[X]=1 FXST[X1]=12.3 FXSW[AX1]=2000
```

Functional sequence

The function is explained by the example below (sleeve is pressed onto workpiece).

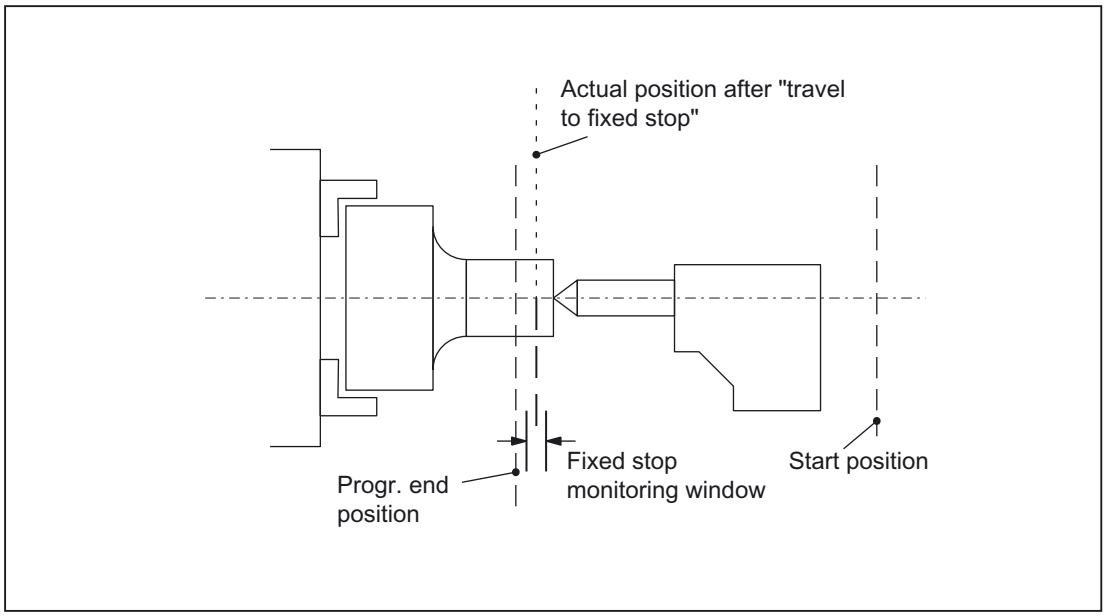


Fig. 2-1 Example of travel to fixed stop

Selection

The NC detects that the function "travel to fixed stop" is selected via the command `FXS[x]=1` and signals the PLC via the IS "Activate travel to fixed stop" (DB31, ... DBX62.4) that the function has been selected.

When the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
(register PLC acknowledgments for travel to fixed stop)
is set accordingly, the IS waits for acknowledgment by the PLC via IS "Activate travel to fixed stop" (DB31, ... DBX3.1).

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 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 was programmed in the block or since the beginning of program, the value entered in the axis specific machine parameter:
MD37010 FIXED_STOP_TORQUE_DEF
(default setting for clamping torque)
applies.

Fixed stop reached

As soon as the axis comes into contact with the mechanical fixed stop (workpiece), the closed-loop control in the drive raises the torque so that the axis can move on. The torque rises up to the programmed limit value and then remains constant.

The status "fixed stop reached" can, depending on the machine parameter:
MD37040 FIXED_STOP_BY_SENSOR (fixed stop recognitioner by sensor)
be determined in the following ways:

- FIXED_STOP_BY_SENSOR = 0

  The status "fixed stop reached" is active when the axial contour deviation (= difference between actual and expected following error) has exceeded the value of the machine parameter:
  MD37030 FIXED_STOP_THRESHOLD (threshold for fixed stop recognition).

- FIXED_STOP_BY_SENSOR = 1

  External sensor transmits "Fixed stop reached" status to NC via the PLC by means of IS "Sensor fixed stop" (DB31, ... DBX1.2).

- FIXED_STOP_BY_SENSOR = 2

  The "Fixed stop reached" status is active either when the status has been detected by the contour monitoring function or when the external sensor signals this status by a signal change from 0 " 1 to (DB31,...DBX 1.2).

The axial contour deviation is explained in:
References:
/FB1/ Function descriptions machine basics; Diagnostic aid (D1)

Internal processes

Once the NC has detected the "Fixed stop reached" status, it deletes the distancetogo and the position setpoint is made to follow. The controller enabling command remains active.

The PLC is then informed of the state by means of IS "Fixed stop reached"
(DB31, ... DBX62.5).

If the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
is set accordingly, then the IS waits for acknowledgment by the PLC in the form of IS "Acknowledge fixed stop reached" (DB31, ... DBX1.1).

The NC then executes a block change or considers the positioning motion to be completed, but still leaves a setpoint applied to the drive actuator 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 torque was programmed in the block or since the beginning of program, the value entered in the axis specific machine parmeter:
MD37010 FIXED_STOP_TORQUE_DEF
(default setting for clamping torque)
applies.

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.

Enabling the fixed stop alarms

The machine parameter:
MD37050 FIXED_STOP_ALARM_MASK
can be used to set the enabling of the fixed stop alarms as follows:

MD 37050 = 0	Fixed stop not reached (suppress Alarm 20091)
MD 37050 = 2	Fixed stop not reached (suppress alarm 20091) and fixed stop aborted (suppress alarm 20094)
MD 37050 = 3	Fixed stop aborted (suppress alarm 20094)

All other permissible values of 7 or less do not suppress any alarms.

Using the parts program command `NEWCONF` a new setting can be activated.

Fixed stop is not reached

If the programmed end position is reached without the status "fixed stop reached" having been recognized, the alarm 20091 "fixed stop not reached" is activated, depending on the status of the machine parameter:
MD37050 FIXED_STOP_ALARM_MASK (enabling fixed stop alarms).

Function abort

If the "Travel to fixed stop" function is aborted owing to a pulse disable command, cancellation of PLC acknowledgments or a Reset in the approach block, the display or suppression of alarm 20094 can be controlled via:
MD37050 FIXED_STOP_ALARM_MASK.

Abort without alarm

The PLC can initiate travel to fixed stop without an alarm in the approach block (for example, when the operator actuates a key) if alarm 20094 is suppressed in machine parameter:
MD37050 FIXED_STOP_ALARM_MASK.

The Travel to fixed stop function is deselected in response to both "Fixed stop not reached" and "Fixed stop aborted".

Interrupts

If the fixed stop position is not reached when the function is active, alarm 20091 "Fixed stop not reached" is output and a block change executed.

If a traversing request (e.g. from the parts program, the PLC, from compile cycles or from the operator panel) is provided for an axis after the fixed stop has been reached, the alarm 20092 "Travel to fixed stop still active" is output and the axis is not moved.

If after reaching the fixed stop an axis is pushed out of position more than the indicated value in SD FIXED_STOP_WINDOW (fixed stop monitoring window),
the alarm 20093 "fixed stop monitoring has responded" is output; the "travel to fixed stop" function is then deselected for this axis and system variable $AA_FXS[x] is set to 2.

No termination during alarm

"Travel to fixed stop" continues to remain active during alarms,
if the bit value in the machine parameter:
MD37052 FIXED_STOP_ALARM_REACTION
has been set.

IS "Mode group ready" (DB11, ... DBX6.3) remains active.

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 parts program commands.

Sequence in case of a fault or abnormal termination

The IS "Activate travel to fixed stop" (DB31, ... DBX62.4) is reset.

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1).

The torque limitation is then canceled and a block change executed.

Deselection

The NC recognizes the function deselection via programming of the command FXS[x]=0. Then an advance stop (STOPRE) is internally released, since it can't be forseen where the axis will be after deselection.

The torque limitation and the monitoring of the fixed stop monitoring window is cancelled. The IS "Activate travel to fixed stop" (DB31, ... DBX62.4) and "Fixed stop reached" (DB31, ... DBX62.5) are reset.

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1) and/or
"Acknowledge fixed stop reached" (DB31, ... DBX1.1).

The axis will then change to position control. The followup mode of the position setpoints is ended and a synchronization to the new actual position is carried out.

A programmed traverse motion can then be executed. This motion must lead away from the fixed stop or else the stop or even the machine may sustain damage.

A block change is executed after the target position has been reached.

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.

Blockrelated synchronized actions

By programming a blockrelated synchronized action, travel to fixed stop can be connected during an approach motion.

Programming example:

```
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                      ; 17mm FXS is activated
```

Changing the clamping torque and fixed stop monitoring window

The clamping torque and the monitoring window can be changed with the commands FXST[x] and FXSW[x]. The changes take effect before traversing movements in the same block.

Programming of a new fixed stop monitoring window causes a change not only in the window width but also in the reference point for the center of the window if the axis has moved prior to reprogramming. The actual position of the machine axis when the window is changed is the new window center point.

Terminal 663 with MD37002 controllable

With the machine parameter:
MD37002 FIXED_STOP_CONTROL
the response to pulse blocking on the fixed stop is controllable.

Deleting the pulses by terminal 663 or the IS "Pulse enable" DBX31, ...DBX21.7 will not abort the function. As a result, the drive will press against the fixed stop again without any further operating action when the machine is restarted.

The rise time of the torque corresponds to the time needed by the current controller of the drive to reach the limitation again.

If the pulses are deleted when a deselection is active (waiting for PLC acknowledgments), the torque limit will be reduced to zero. If the pulses are reactivated during this phase, torque is no longer built up. Once the deselection has been completed, you can continue traversing as normally.

FXS commands programmable in synchronized actions

The parts program commands FXS, FXST and FXSW can be programmed in synchronized actions/technology cycles.

The function FXS[x]=1 can also be activated without movement; the torque is limited immediately. As soon as the axis is moved via a setpoint, the limit stop monitor is activated.

In static and blockrelated synchronized actions, the same commands FXS, FXST, FXSW can be used as in the normal parts program run. The values assigned can result from a calculation.

Ramp for torque limitation with MD37012

A ramp has been implemented so that the setting of a torque limit is not to jerky.

For this purpose, the amount of time required to reach the new torque limit is specified in machine data element:
MD37012 FIXED_STOP_TORQUE_RAMP_TIME.

2.1.2 Response to RESET and function abort

Response to RESET

During selection (fixed stop not yet reached) the function FXS can be aborted with RESET. The termination is carried out such that an "almost achieved" fixed stop (setpoint already beyond the fixed stop, but still within the threshold for the fixed stop detection) will not result in damage.

This is achieved by synchronizing the position setpoint to the new actual position. As soon as the fixed stop is reached, the function remains operative even after RESET.

Function abort

A function abort can be triggered by the following events:

- EMERGENCY STOP:

  – With an 840D control, the NC and drive are disconnected from the supply after EMERGENCY STOP, i.e. the PLC must react.

  – With an 840Di control, the NC and drive are disconnected from the supply after EMERGENCY STOP, i.e. the PLC must react.



Caution

Make sure that after cancelling the function "travel to fixed stop" by "EMERGENCY STOP" no dangerous machine situation (MD37002 FIXED_STOP_CONTROL e.g. pulse blocking cancel) can arise.

- The fixed stop monitoring function responds in the case of:

  – Exit from fixed stop position by axis

  – Tool breakage

  – Pulse disable

2.1.3 Block search response

Block search with calculation

The response is as follows:

- 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 desired fixed stop status is reached, alarm message 10208 "press NC Start to continue the program" is output. The program can be continued after pressing NC Start to acknowledge.
- At the start of the target block, `FXST[x]` and `FXSW[x]` are set to the same value as they would have during normal program processing.

Block search without calculation

The commands `FXS`, `FXST` and `FXSW` are ignored.

FOC

`FOCON`/`FOCOF` is activated modally. It is already active in the approach block.

SERUPRO

Block search with calculation, multichannel

The block search in program test mode is designated SERUPRO and is derived from the "Search-Run by Program test." This search mode allows the user a multichannel block search with calculation of all required status data from the previous history.

The PLC interface is updated in this block search and matching processes, which cover the interaction of several channels executed within the framework of this block search correctly.

Search process with FXS and FOC

The user selects FXS or F0C in a program area of the searched target block in order to acquire all states and functions of this machining last valid. The NC will start 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 "travel to fixed stop" is located between the beginning of the program and the search target, the instruction is not really executed by the NC. The motion is only simulated up to the programmed end point.



Caution

SERUPRO approach does not really take the statementFXS 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 parts program. If necessary, the user can start an ASUB in order to activate or deactivate FXS in this SERUPRO-ASUP.

$AA_FXS and $VA_FXS

In SW 6.2 and higher, the meaning of system variable $AA_FXS is redefined for SERUPRO only and completely replaced by variable $VA_FXS. Variables $AA_FXS and $VA_FXS have the same values continuously outside the SERUPRO function.

Description	NCK Variables
Axis not at fixed stop	$AA_FXS = 0 and $VA_FXS = 0
Fixed stop successfully approached	$AA_FXS = 1 and $VA_FXS = 1
Approach to fixed stop failed	$AA_FXS = 2 and $VA_FXS = 2
Travel to fixed stop selection active	$AA_FXS = 3 and $VA_FXS = 3
Fixed stop detected	$AA_FXS = 4 and $VA_FXS = 4
Travel to fixed stop deselection active	$AA_FXS = 5 and $VA_FXS = 5

Course of values

Course of values of system variables $VA_FXS[ ] with values 1 to 5

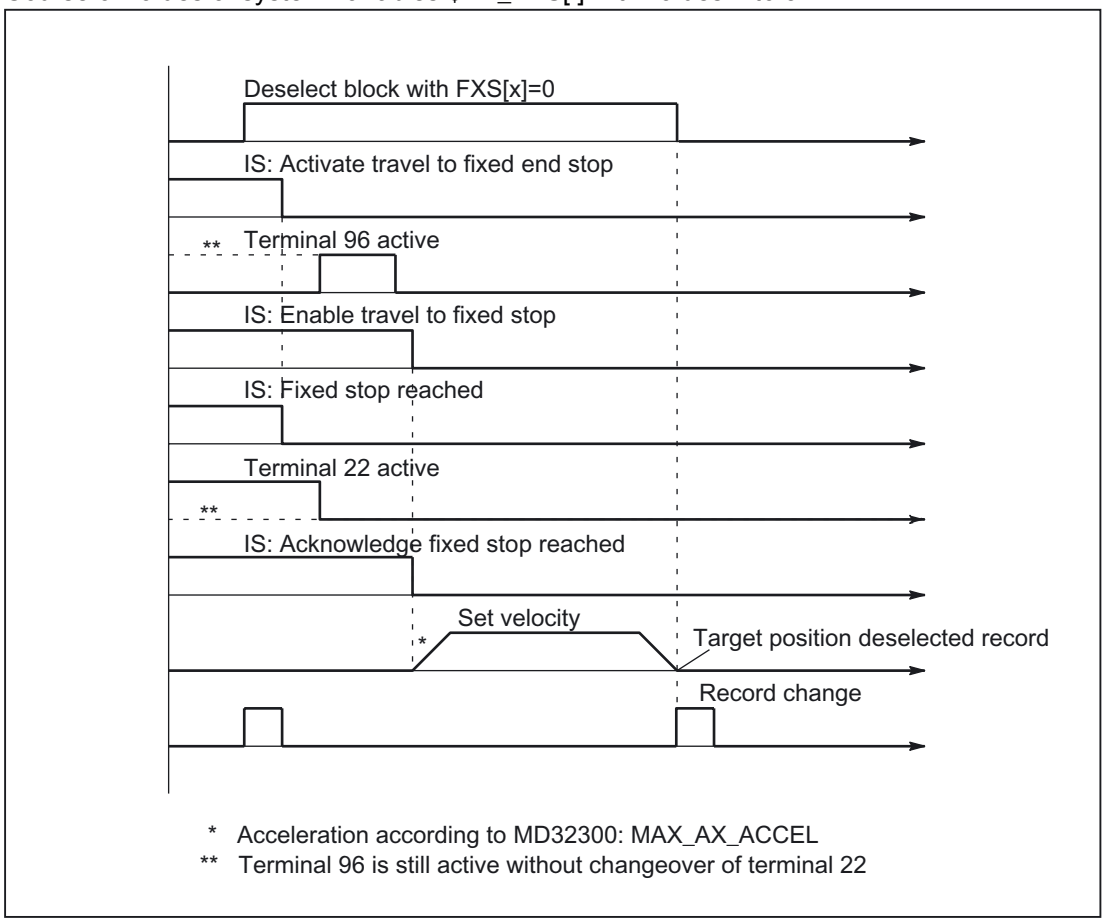


Fig. 2-2 Diagram for FXS with a digital drive (611 digital)

$AA_FXS Simulate axis traversal

System variable $AA_FXS displays the current status of program simulation "program-sensitive system variable."

Example:

If in the SERUPRO process axis Y traversal is simulated with `FXS[Y]=1`, then $AA_FXS has a value of 3.

If in the SERUPRO process axis Y traversal is simulated with `FXS[Y]=0`, then $AA_FXS has a value of 0.

During simulation with SERUPRO, $AA_FXS cannot have the values 1, 2, 4, 5, since the actual status of $VA_FXS "travel to fixed stop" can never be detected.

Note

The state $AA_FXS = 1 is never reached during the SERUPRO operation. This means that other program branches can be processed which will produce different results due to the simulation.

If after the SERUPRO process axis Y is traversed again, then variables $AA_FXS and $VA_FXS are assigned the same values again.

$VA_FXS Real machine status

Variable $VA_FXS always describes the real machine status.

In this way, the actual existing machine status of the corresponding axis with $VA_FXS is displayed.

Setpoint/actual status comparison

The two system variables $AA_FXS and $VA_FXS can be used to compare the setpoint and actual states in the parts program. The SERUPRO ASUB routine is then as follows:

SERUPRO ASUP

Asup fxsSeruproAsup.mpf

;The setpoint and actual states are compared for the
;REPOSA block FXS to correspondingly activate or deactivate.

```
N1000 WHEN ($AA_FXS[X]==3) AND ($VA_FXS[X]==0) DO FXS[X]=1
N2000 WHEN ($AA_FXS[X]==0) AND ($VA_FXS[X]==1) DO FXS[X]=0
N1020 REPOSA
```

REPOS display Offset

Once the search target has been found, the FXS status active on the machine is displayed for each axis via the axial VDI signals:
IS "Activate travel to fixed stop" (DB31, ... DBX62.4)
and
IS "Fixed stop reached" (DB31, ... DBX62.5).

Example:

If the machine is at the fixed stop and the block search has reached a block after deselection ofFXS, the new target position is displayed as the REPOS offset via IS "Fixed stop reached" (DB31, ... DBX62.5) as REPOS offset.

FXS fully automatically in REPOS

With REPOS the functionality of FXS is repeated automatically and designated FXS-REPOS below. This sequence is comparable to the program "fxsSeruproAsup.mpf." Every axis is taken into account and the torque last programmed before the search target is applied.

The user can treat FXS separately in a SERUPRO-ASUP.

The following then applies:

Every FXS action in SERUPRO-ASUP automatically ensures
$AA_FXS[X] == $VA_FXS[X].

This deactivates FXS-REPOS for axis X.

Deactivate FXS REPOS

FXS-REPOS is deactivated by:

- An FXS synchronous action which refers to REPOSA

  or

- $AA_FXS[X] == $VA_FXS[X] in SERUPRO_ASUP.

Note

A SERUPRO-ASUP **without** FXS handling or no SERUPRO-ASUP results in fully automatic FXS-REPOS.



Caution

FXS-REPOS moves all path axes in a path grouping to the target position. Axes with and without FXS treatment thus traverse together with the G code and feedrate valid at the time the target block is processed. As a result, the fixed stop may be approached in rapid traverse (G0) or higher velocity.

FOC fully automatically in REPOS

The FOC-REPOS function behaves analogously to the FXS-REPOS function.

Note

A continuously changing torque characteristic cannot be implemented with FOC-REPOS.

Example:

A program moves axis X from 0 to 100 and activates FOC every 20 millimeters for 10 millimeters at a time. This torque characteristic is generated with nonmodal FOC and therefore cannot be traced by FOC-REPOS. FOC-REPOS will traverse axis X from 0 to 100 with or without FOC.

For programming examples of FXS "travel to fixed stop," please see:
References:
/FB1/ description of functions machine basics; BAG, channel, program operation (K1);
Chapter: Program test

2.1.4 Miscellaneous

Setting data

The following axisspecific setting data are provided for the "Travel to fixed stop" function:

SD43500 FIXED_STOP_SWITCH (selection of travel to fixed stop)

SD43510 FIXED_STOP_TORQUE (clamping torque for travel to fixed stop)

SD43520 FIXED_STOP_WINDOW (fixed stop monitoring window)

The setting data are effective only when the axis has reached the fixed stop.

The status of the setting data is displayed via the operator panel in the "Parameters" area.

The commands FXS[x], FXST[x] and FXSW[x] effect a block-synchronous change in these setting data.

If FXST[x] and FXSW[x] are not programmed, the default settings from the machine data:
MD37010 FIXED_STOP_TORQUE_DEF
and
MD37020 FIXED_STOP_WINDOW_DEF
are taken over in the corresponding setting dates if "travel to fixed stop" is activated.

The setting data for clamping torque and fixed stop monitoring window can be changed by the operator and via the PLC. It is thus possible to specify a higher or lower clamping torque or a modified fixed stop monitoring window after the fixed stop has been reached.

Changing the clamping torque using the ramp and values greater than 100%

A clamping torque change is transferred to the drive steplike. It is possible to specify a ramp for always reaching a changed torque limit via the machine parameter:
MD37012 FIXED_STOP_TORQUE_RAMP_TIME.

Clamping torque greater than 100%

Values greater than 100% for the setting data:
SD43510 FIXED_STOP_TORQUE
are only practical for a short time.

Irrespectively, the maximum torque is limited by the drive.

The drive machine dates have a limiting effect, i.e.:

MD1103 Motor current

MD1104 Maximum motor current

MD1105 Reduction in max. motor current

MD1230/1231 Torque limiting value 1/2

For further information, please refer to the Planning Guide SIMODRIVE ThreePhase Motors for Feed and Main Spindle Drives and to the appropriate document regarding the hydraulic module /FBHLA/.
References:
/FBHLA/ Description of functions HLA-Module

Status query in the parts program

System variable $AA_FXS[x] indicates the status of the "travel to fixed stop" function.

It has the following coding:

$AA_FXS=0	Axis not at fixed stop.
$AA_FXS=1	Fixed stop has been approached successfully (axis is within fixed stop monitoring window).
$AA_FXS=2	Approach to fixed stop has failed (axis is not at fixed stop).
$AA_FXS=3	Travel to fixed stop activated.
$AA_FXS=4	Fixed stop detected.
$AA_FXS=5	Travel to fixed stop is deselected. The deselection is not yet completed.

Query of the system variable in the parts program initiates a block search stop.

As a result of the status query in the parts program, it is possible, for example, to react to an erroneous operational sequence of the "Travel to fixed stop" function.

The following example applies:
MD37050 FIXED_STOP_ALARM_MASK = 0
⇒ An alarm is not generated in the error scenario, and a block change therefore takes place and the error scenario can be evaluated via the system variable.

Example

```
X300 Y500 F200 FXS[X1]=1 FXST[X1]=25 FXSW[X1]=5
IF $AA_FXS[X1]=2 GOTOF FXS_ERROR
G01 X400 Y200
```

Inoperative IS signals

The following IS signals (PLC ! NCK) are inoperative for axes at the fixed stop until the function is deselected (incl. traversing motion):

- IS "Axis/spindle lock" (DB31, ... DBX1.3)
- IS "Controller enable" (DB31, ... DBX2.1)

Actual position at fixed stop

System variable $AA_IM[x] can determine the actual position of the machine axis, e.g. for test purposes after successful "travel to fixed stop."

Combination with other functions

"Measure with deletion of distance to go" (command MEAS) and "travel to fixed stop" cannot be programmed at the same time in one block.

Exception:

One function is acting on a path axis while the other is acting on a positioning axis or both functions are acting on positioning axes.

Contour monitoring

The axis contour monitoring function is inoperative while "Travel to fixed stop" is active.

Positioning axes

When "Travel to fixed stop" is applied toPOSA axes, block changes are made independently of the fixed stop motion.

Vertical axes

The "Travel to fixed stop" function can be used for vertical axes even when alarms are active.

Should the traversal of vertical axes be aborted as a result of an FXS alarm, the relevant drives are not disconnected from the supply via the VDI interface.

This functionality has the same effect on vertical axes as an electronic weight compensation and can be configured via machine parameter:
MD37052 FIXED_STOP_ALARM_REACTION.

Note

For further details about adaptations for SIMODRIVE 611 digital or digital (HLA module), please see:
References:
/FB2/ Expansion functions; Compensation (K3); Chapter: Electronic weight compensation

MD37052

The machine parameter:
MD37052 FIXED_STOP_ALARM_REACTION
does not result in disconnection of the drive from the power supply when an alarm is generated, as interface signal "Mode group ready" DB11,... DBX6.3 remains active.

Bit value=0:

The alarms have an effect on FXS (drive becomes disconnected as previously).
IS "Mode group ready" DB11, ... DBX6.3 is reset.

Bit value=0:

The alarms have no effect on FXS.
IS "Mode group ready" DB11, ... DBX6.3 remains active.

Bit 0: Alarm 20090	Travel to fixed stop cannot be programmed.
Bit1: Alarm 20091	Fixed stop is not reached.
Bit 2: Alarm 20092	Travel to a fixed stop active.
Bit 3: Alarm 20093	Standstill monitoring on fixed stop is triggered.
Bit 4: Alarm 20094	Travel to fixed stop aborted.

2.1.5 Supplementary conditions for expansions

Response to pulse blocking

The cancellation of the pulse enable, either by terminal 663 or by NST "Pulse enable" (DB31, ... DBX21.7) is referred to as pulse blocking below.

The machine parameter:
MD37002 FIXED_STOP_CONTROL
can be used to influence the interaction of "travel to fixed stop" and pulse blocking.

Bit 0: Response in case of pulse blocking at stop as follows:

Bit 0 = 0	Travel to fixed stop aborted.
Bit 0 = 1	Travel to fixed stop is aborted, i.e. the drive becomes weak.

Once the pulse blocking is canceled again, the drive will press at the limited torque again. The torque is actuated steplike.

At the fixed stop, the drive can be controlled either via:

- IS "Pulse enable" (DB31, ... DBX21.7)

  or

- Drive monitor 663 pulse enable

The NC evaluates IS "Pulse enable" (DB31, ... DBX21.7) itself.

Depending on the drive machine parameter:
MD1012 FUNC_SWITCH
the NC reacts with FXS to a change in the status of terminal 663 as follows:

Bit 2 = 0	The NC does **not** receive the status of terminal 663.
Bit 2 = 1	The NC does receive the status of terminal 663.

Terminal 663

When pulse enabling is canceled by terminal 663, the drive is deenergized and coasts to a standstill immediately.

With:
MD1012 FUNC_SWITCH, Bit 2 = 0,
this is not signaled to NC.

The status can be checked in the line "Pulse enable" (terminal 663) in service display service drive.

Note

Travel to fixed stop only can be stopped by blocking of the pulses from IS "Pulse enable" (DB31, ... DBX21.7) or Terminal 663, if:
MD37002 FIXED_STOP_CONTROL, Bit 0 = 0
and
MD1012 FUNC_SWITCH, Bit 2 = 1
are fulfilled.

Both interrupting FXS and stopping "travel to fixed stop" is achieved by the following settings:
MD37002 FIXED_STOP_CONTROL, Bit 0 = 0
and
MD1012 FUNC_SWITCH, Bit 2 = 0.

Programming FXS in synchronized actions

The function is **not** available for analog axes (PLC acknowledgment cannot be awaited).

Select FXS[ ]=1:

The following signal interfaces are set:

Message to PLC:
IS "Activate travel to fixed stop" (DB31, ... DBX62.4)

The FXS selection command can **only** be used in systems with digital drives (VSA, HSA, HLA).

The following condition **must** be observed:
MD37060 FIXED_STOP_ACKN_MASK, Bit 0 = 0

Bit 0 = 1 (waiting for PLC acknowledgment) must **not** be set, otherwise, an interpolator stop would be required to acknowledge the signal, interrupting the movement.

Deselect FXS[ ]=0:

The following signal interfaces are reset:

Message to PLC:
IS "Activate travel to fixed stop" (DB31, ... DBX62.4)

The machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
must have the value null for signal deselection **without** motion stop.

Without ramp

The torque limit is changed without taking into account the ramp if:

- FXS is activated with (FXS[ ]=1), to make sure that the reduction is activated immediately (especially for synchronized actions).
- The drive needs to be de-energized as quickly as possible in the event of a fault.

Selection of FXS with G64

In the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
bit 0 must be 0 (do not wait for PLC input signal "travel to fixed stop active"), since the selection of FXS may not initiate a motion stop.

Modify the torque FXST

The clamping torque can already be modified when approaching the stop.

The torque limit FXST acts in addition to the acceleration limitation with ACC.

The new torque in consideration of the ramp
(MD37012 FIXED_STOP_TORQUE_RAMP_TIME)
interpolation cycle takes effect after the change in the drive.
A change of the effective torque can be checked by reading the synchronized action variable $VA_TORQUE[axis].

2.1.6 Travel with limited moment/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 following functionality FOC (Force Control) is provided.

Force/travel or force/time profiles are thus possible using the "Interpolation cycle" resolution.

The function allows torque/force to be modified at any time using synchronized actions.

The function can be activated modally or blockrelated.

Modal activation (FOCON/FOCOF)

The activation of the function after POWER_ON and RESET is determined by the machine parameter:
MD37080 FOC_ACTIVATION_MODE.

Table 2-1 Controlling the initial setting of the modal limitation of torque/force

	After	Value	Effect
Bit 0	Power On	0	FOCOF
		1	FOCON
Bit 1	RESET	0	FOCOF
		1	FOCON

FOCON: Activation of the modally effective torque/force limitation

FOCOF: Disable torque/force limitation

The modal activation acts beyond the program end.

If already programmed, it is effective with FXST set torque/force.

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:

```
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 parts 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 parts 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 PowerOn, the machine data:
MD37010 FIXED_STOP_TORQUE_DEF. is effective in an activation.

This torque can be modified 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.

0: FOC not active
1: FOC modal active
2: FOC block-related active

Determine torque limit status

The system variables $VA_TORQUE_AT_LIMIT can be used at any time to read in systems with digital drives (FDD, MSD, HLA) whether the currently active torque corresponds to the given torque limit.

0: Effective torque less than torque limit value
1: Effective torque has reached the torque limit value

Restrictions

The function FOC is subject to the following restrictions:

- The change of the torque/force limitation representing itself as an acceleration limitation is **only taken into account in the traversing movement at block limits** (see command ACC).
- Only FOC:

  No monitoring is possible from the VDI interface to check that the active torque limit has been reached.
- If the acceleration limitation is not adapted accordingly, an increase of the drag distance during the traversing motion occurs.
- If the acceleration limitation is not adapted accordingly, the end-of-block position is possibly reached later than specified in:
  MD36040 STANDSTILL_DELAY_TIME.

  The machine parameter:
  MD36042 FOC_STANDSTILL_DELAY_TIME
  is introduced for this and monitored in this status.

Possible application 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:
/FB2/ Expansion functions; several control panels on multiple NCUs, decentralized systems (B3)

The status of the machine axis is kept in the case of a container switch, 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 switch.

2.2 Travel to fixed stop with analog drives

2.2.1 SIMODRIVE 611 digital (VSA/HSA)

Selection

The NC detects that the function "travel to fixed stop" is selected via the command FXS[x]=1 and signals the PLC via the IS "Activate travel to fixed stop" (DB31, ... DBX62.4) that the function has been selected.

If the machine parameter:
MD37050 FIXED_STOP_ACKN_MASK
is set accordingly, then the IS waits for acknowledgment by the PLC in the form of IS "Activate travel to fixed stop" (DB31, ... DBX3.1)
and then the function is started.
This acknowledgment is not required by the NC for digital drives.

The axis now traverses to the target position at the programmed velocity. At the same time, the clamping torque (specified via FXST[x]) is transferred to the drive via the digital interface, and this limits its effective torque. In addition, the acceleration is reduced accordingly in the NC automatically acc. to FXST[x]).

Fixed stop reached

As soon as the axis reaches the fixed stop, the axial contour deviation increases.

If the recorded threshold in the machine parameter:
MD37030 FIXED_STOP_THRESHOLD
is exceeded or the IS signal "Sensor fixed stop" (DB31, ... DBX1.2) is set, the controller recognizes that the fixed stop has been reached.

The NC will then delete the distance to go and will synchronize the position setpoint to the current actual position value. The controller enabling command remains active.

The NC then outputs IS "Fixed stop reached" (DB31 , ... DBX62.5) to the PLC.

If the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
is set accordingly, then the IS waits for acknowledgment by the PLC in the form of IS "Acknowledge fixed stop reached" (DB31, ... DBX1.1).

This acknowledgment is not required by the NC for digital drives.

A block change is executed. The clamping torque continues to be applied.

Fixed stop is not reached

If the programmed end position is reached without the "Fixed stop reached" status being recognized, then the torque limitation in the drive is canceled via the digital interface and IS "Activate travel to fixed stop" (DB31, ... DBX62.4).

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1)
and then the block change is performed.

Deselection

The NC recognizes the function deselection via programming of the command `FXS[x]=0`. Then an advance stop (`STOPRE`) is internally released, since it cannot be forseen where the axis will be after deselection.

The torque limitation and monitoring of the fixed stop monitoring window are canceled.
IS "Activate travel to fixed stop" (DB31, ... DBX62.4) and "Fixed stop reached"
(DB31, ... DBX62.5) are reset.

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1) and/or
"Acknowledge fixed stop reached" (DB31, ... DBX1.1).

The axis will then change to position control. The followup mode of the position setpoints is ended and a synchronization to the new actual position is carried out.

A programmed traverse motion can then be executed. This motion must lead away from the fixed stop or else the stop or even the machine may sustain damage.

A block change is executed after the target position has been reached.

Enabling the fixed stop alarms

The machine parameter:
MD37050 FIXED_STOP_ALARM_MASK
can be used to set the enabling of the fixed stop alarms as follows:

MD 37050 = 0	Fixed stop not reached (suppress Alarm 20091)
MD 37050 = 2	Fixed stop not reached (suppress alarm 20091) **and** fixed stop aborted (suppress alarm 20094)
MD 37050 = 3	Fixed stop aborted (suppress alarm 20094)

All other permissible values of 7 or less do not suppress any alarms.

Using the parts program command `NEWCONF` a new setting can be activated.

Terminal 663 with MD37002 controllable

With the machine parameter:
MD37002 FIXED_STOP_CONTROL
the response to pulse blocking on the fixed stop is controllable.

Deleting the pulses by terminal 663 or the IS "Pulse enable" DBX31, ...DBX21.7 will not abort the function. As a result, the drive will press against the fixed stop again without any further operating action when the machine is restarted.

The rise time of the torque corresponds to the time needed by the current controller of the drive to reach the limitation again.

If the pulses are deleted when a deselection is active (waiting for PLC acknowledgments), the torque limit will be reduced to zero. If the pulses are reactivated during this phase, torque is no longer built up. Once the deselection has been completed, you can continue traversing as normally.

Diagram

The following diagram shows the progress of the motor current, the following error and IS signals for "Activate travel to fixed stop" (DB31, ... DBX62.4) and "Fixed stop reached" (DB31, ... DBX62.5) with digital drive (SIMODRIVE 611 digital).

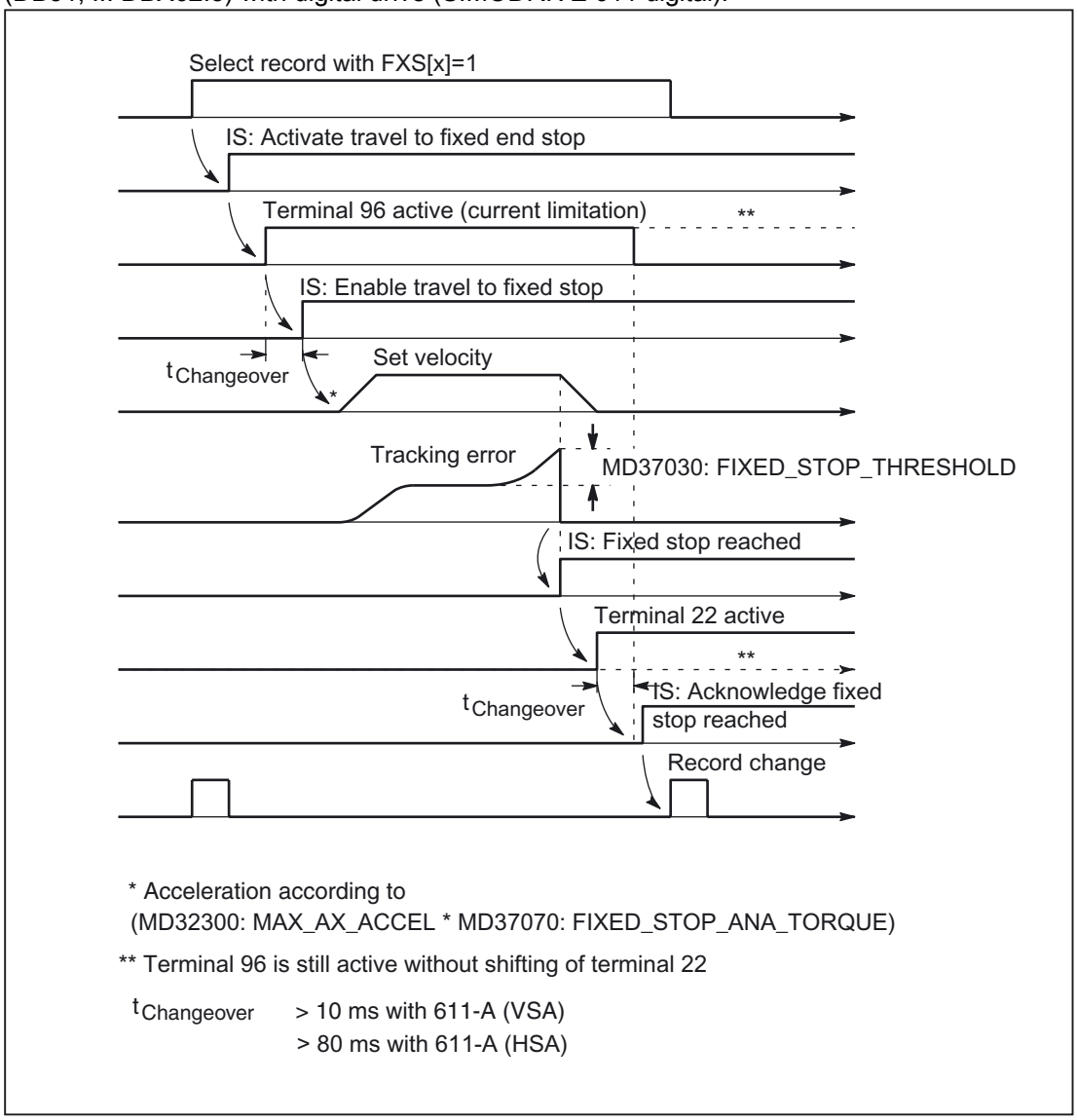


Fig. 2-3 Diagram for FXS with a digital drive (611 digital)

2.2.2 Travel to fixed stop with hydraulic drives SIMODRIVE 611 digital (HLA module)

Velocity/force control

If the function FXS (FXS[x]=1) is activated for the hydraulic module 611 digital (HLA module), only a change from velocity control to force control takes place. Positioning from the NC is no longer possible in this case.

NC

When traveling to fixed stop, the NC evaluates a torque/force limit acting in addition to the limits set on the drive:

- Current
- Force/torque
- Power, pull-out power
- Setup mode

Note

For a description of velocity and force control and special information about the adjustments on SIMODRIVE 611 digital and digital (HLA module), please refer to:
References:
/FBHLA/ Description of Functions HLA Module; Drive Functions Firmware
/FB2/ Expansion functions; Compensation (K3); Chapter: Electronic weight compensation

2.3 Travel to fixed stop with analog drives

2.3.1 SIMODRIVE 611 analog (FDD)

Current/torque control

With the 611 analog (FDD), the torque control and torque limitation 611 analog (VSA) has been realized as a current control/current limitation.

Fixed clamping torque

A fixed current limitation is preset in the drive actuator by means of a resistor circuit (or via R12). This current limitation is activated by the control via a PLC output (which acts on terminal 96 of the actuator), thus ensuring that a fixed clamping torque is available on the axis.

Setpoints can be injected via terminals 56/14 or 24/20.

Programmable clamping torque

In this case, the PLC switches the drive actuator from speed-controlled into current-controlled operation as soon as the fixed stop is reached.

Activation of terminal 22 causes the voltage level applied to terminals 20/24 to take effect as a current setpoint rather than a speed setpoint.

The NC is thus able to specify a variable clamping torque.

Setpoints must be injected via terminals 24/20.

References:
/IAA/ Installation and start-up guide SIMODRIVE 611 analog system

Selection

The NC detects that the function "travel to fixed stop" is selected via the command `FXS[x]=1` and signals the PLC via the IS "Activate travel to fixed stop" (DB31, ... DBX62.4) that the function has been selected.

The PLC must then activate the current limitation on the actuator (terminal 96).

If the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
is set accordingly, then the IS waits for acknowledgment by the PLC in the form of IS "Activate travel to fixed stop" (DB31, ... DBX3.1)
and then the function is started.

Note

This acknowledgment should always be programmed for analog drives for safety reasons, i.e. so that the motion is not started before the torque has been limited.

The NC then internally sets the torque limit to the value specified via the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE
(torque limit on fixed stop approach for analog drives).
This value must correspond to the torque limit value activated via terminal 96.

In addition, the acceleration is automatically reduced in the NC according to the value in the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE.

The axis now traverses to the target position at the programmed velocity.

Fixed stop reached

As soon as the axis reaches the fixed stop, the axial contour deviation increases.

If the recorded threshold in the machine parameter:
MD37030 FIXED_STOP_THRESHOLD
is exceeded or the IS signal
"Sensor fixed stop" (DB31, ... DBX1.2) is set,
the controller recognizes that the fixed stop has been reached.

The position controller then responds by outputting a speed setpoint corresponding to the value set in the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE.

The speed controller's output is limited by terminal 96, forcing the drive to the current limit by means of this continuously applied setpoint.

The NC then deletes the remaining distancetogo and forces the position setpoint to follow. The controller enabling command remains active.

The NC then outputs IS "Fixed stop reached" (DB31 , ... DBX62.5) to the PLC.

If a **programmable clamping torque** input (via FXST[x] or setting data), is required from the NC, then the PLC must switch the drive over from speed controlled to current controlled operation.

To do so, it activates terminal 22 and switches off the current limitation (terminal 96) after a period of 10 ms.

As a result, the torque now takes effect on the drive from the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE.

If the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
is set accordingly, then the IS waits for acknowledgment by the PLC in the form of IS "Acknowledge fixed stop reached" (DB31, ... DBX1.1) and then suddenly transfers the required clamping torque from the selection block to the drive.

A block change is executed. The clamping torque continues to be applied.

Fixed stop is not reached

If the programmed end position is reached without the "fixed stop reached" status being recognized, then the internal torque limit in the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE
is cancelled and the IS signal "activate travel to fixed stop"
(DB31, ... DBX62.4).

The PLC must then deactivate the current limitation (terminal 96).

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1)
and then the block change is performed.

Deselection

The NC detects that the function has been deselected on the basis of command`FXS[x]=0` and specifies a speed or current setpoint of "0", i.e. zero clamping torque.

The NC then resets IS "Activate travel to fixed stop" (DB31, ... DBX62.4) and "Fixed stop reached" (DB31, ... DBX62.5).

If current-controlled operation is activated, the PLC must first switch on the current limitation (terminal 96) and switch the drive over to speed-controlled operation (terminal 22) (a speed setpoint of "0" is applied by NC).

Then current limitation must then be deactivated (terminal 96).

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1) and/or
"Acknowledge fixed stop reached" (DB31, ... DBX1.1).

The axis then switches over to position control mode (follow-up mode is terminated) and synchronization with the new actual position takes place.

The programmed travel motion is then executed.

A block change is executed after the target position has been reached.

2.3.2 SIMODRIVE 611 analog (FDD)

Fixed clamping torque

A fixed clamping torque is implemented by entering a fixed torque limitation in a free gear stage in the drive actuator (setting parameter 039). When the "Travel to fixed stop function" is selected, the PLC switches over to the unassigned gear stage of the drive actuator, thus activating the torque limitation.

Setpoints must be injected via terminals 56/14.

Programmable clamping torque

In this case, the PLC switches the drive actuator from speed-controlled into torque-controlled operation after the fixed stop is reached. The NC can therefore input a variable clamping torque.

Setpoints must be injected via terminals 56/14.

References:
/IAA/ Installation and start-up guide SIMODRIVE 611 analog system

C-axis operation

The control system has to switch the spindle into C-axis mode before the "Travel to fixed stop" function is selected. The PLC does this by activating one of the programmable terminals E1-D9 (e.g. terminal E1) of the drive actuator.

Selection

The NC detects that the function "travel to fixed stop" is selected via the command `FXS[x]=1` and signals the PLC via the IS "Activate travel to fixed stop" (DB31, ... DBX62.4) that the function has been selected.

As a result, the PLC activates the unassigned gear stage, in which the torque limitation is effective, by means of programmable terminals E1 - E9 of the drive actuator. It then switches the speed controller monitor off (one terminal (E1-E9) to deactivate error F11 of drive actuator).

If the machine paramete:
MD37060 FIXED_STOP_ACKN_MASK
is set accordingly, then the IS waits for acknowledgment by the PLC in the form of IS "Activate travel to fixed stop" (DB31, ... DBX3.1) and then the function is started.

Note

This acknowledgment should always be programmed for analog drives for safety reasons, i.e. so that the motion is not started before the torque has been limited.

The NC then internally sets the torque limit to the value specified via the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE
(torque limit on fixed stop approach for analog drives).
This value must correspond to the torque limit value set in the actuator.

In addition, the acceleration is automatically reduced in the NC according to the value in the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE.

The rotary axis now traverses to the target position at the programmed velocity.

Fixed stop reached

As soon as the Caxis reaches the fixed stop, the axial contour deviation increases.

If the recorded threshold in the machine parameter:
MD37030 FIXED_STOP_THRESHOLD
is exceeded or the IS signal
"Sensor fixed stop" (DB31, ... DBX1.2) is set,
the controller recognizes that the fixed stop has been reached.

The position controller then responds by outputting a speed setpoint corresponding to the value set in the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE.

The speed controller forces the drive to the torque limit by means of this continuously applied setpoint.

The NC then deletes the remaining distancetogo and forces the position setpoint to follow. The controller enabling command remains active.

The NC then outputs IS "Fixed stop reached" (DB31 , ... DBX62.5) to the PLC.

If a **programmable clamping torque** input (via FXST[x] or setting parameter) is required from the NC, then the PLC must switch the drive over from speed controlled to current controlled operation. To do so, it activates one of the programmable terminals E1-E9 (e.g. terminal E5) and switches off the torque limitation after a period of 80 ms by selecting the preceding gear stage.
As a result, the torque now takes effect on the drive from the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE.

If the machine parameter:
MD37060 FIXED_STOP_ACKN_ASK
is set accordingly, then the IS waits for acknowledgment by the PLC in the form of IS "Acknowledge fixed stop reached" (DB31, ... DBX1.1) and then suddenly transfers the required clamping torque from the selection block to the drive.

A block change is executed. The clamping torque continues to be applied.

Fixed stop is not reached

If the programmed end position is reached,
without the "fixed stop reached" status being detected,
then the internal torque limit in the machine parameter:
MD37070 FIXED_STOP_ANA_TORQUE
is cancelled and the IS signal "activate travel to fixed stop"
(DB31, ... DBX62.4).

The PLC then activates the preceding gear stage, thus deactivating the torque limitation. It also switches on the speed control monitoring function again.

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1)
and then the block change is performed.

Deselection

The NC detects that the function has been deselected on the basis of command `FXS[x]=0` and specifies a speed or torque setpoint of "0", i.e. zero clamping torque.

The NC then resets IS "Activate travel to fixed stop"
(DB31, ... DBX62.4) and "Fixed stop reached" (DB31, ... DBX62.5).

If torque-controlled operation is activated, the PLC must first select the unassigned gear stage in which the torque limitation is effective and switch the drive over to speed-controlled operation (a speed setpoint of "0" is applied by NC). The PLC must also deactivate the speed controller monitoring function.

The PLC then activates the preceding gear stage, thus deactivating the torque limitation. It also switches on the speed control monitoring function again.

Depending on the machine parameter:
MD37060 FIXED_STOP_ACKN_MASK
the NC waits for the PLC to acknowledge by resetting IS
"Activate travel to fixed stop" (DB31, ... DBX3.1) and/or
"Acknowledge fixed stop reached" (DB31, ... DBX1.1).

The C-axis then switches over to position control mode (follow-up mode is terminated) and synchronization with the new actual position takes place.

The programmed travel motion is then executed.

A block change is executed after the target position has been reached.

2.3.3 Diagrams for travel to fixed stop with analog drives

FXS selection (fixed stop is reached)

The following diagram shows the sequence of the following error and interface signals for "FXS selection" (fixed stop is reached) on analog drives.

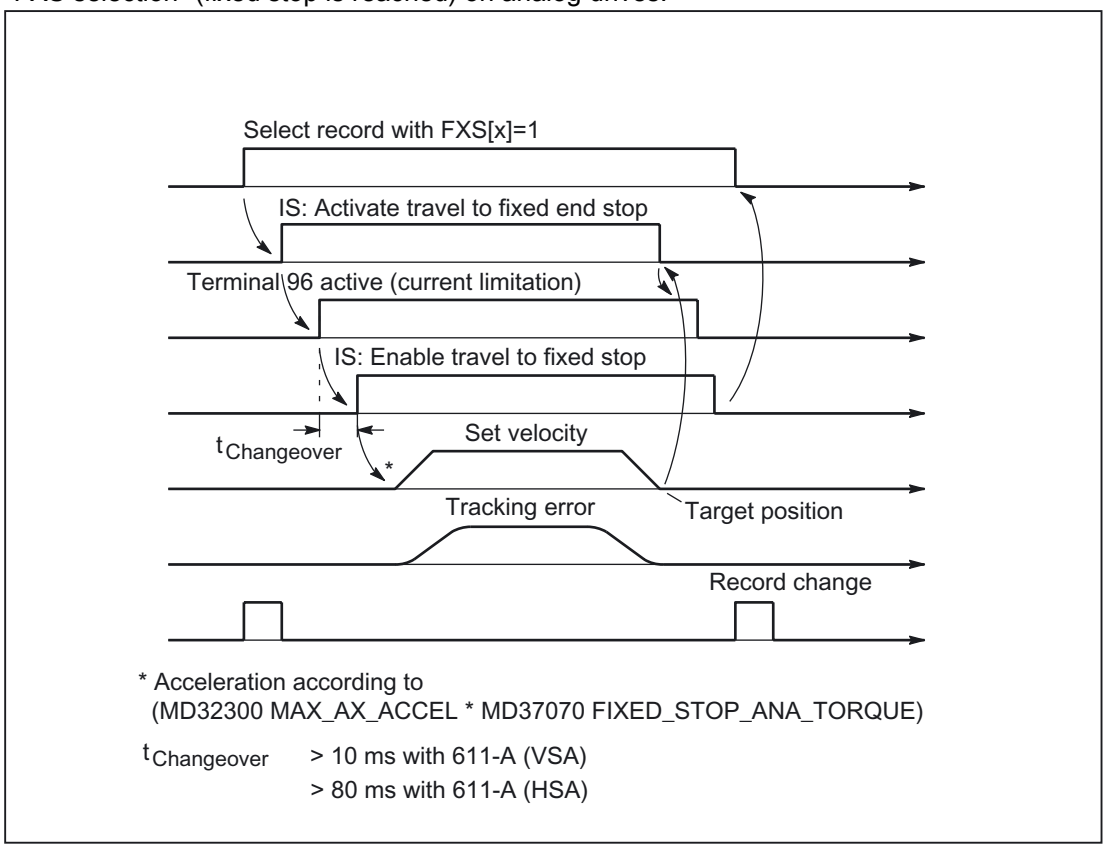


Fig. 2-4 Diagram for FXS selection (fixed stop is reached) with analog drive

FXS selection (fixed stop is not reached)

The following diagram shows the sequence of the following error and interface signals for "FXS selection" (fixed stop is **not** reached) on analog drives.

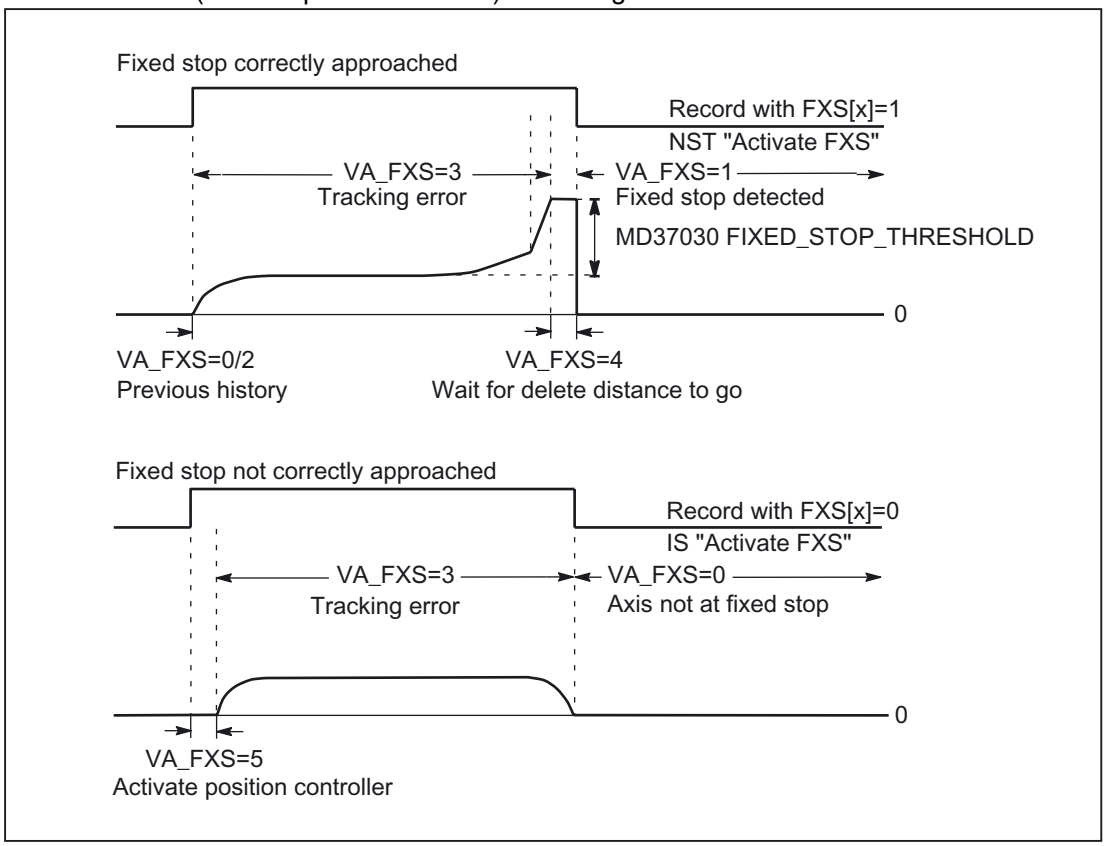


Fig. 2-5 Diagram for FXS selection (fixed stop is not reached) with analog drive

FXS deselection

The following diagram shows the sequence of the following error and interface signals for "FXS Deselection" on analog drives.

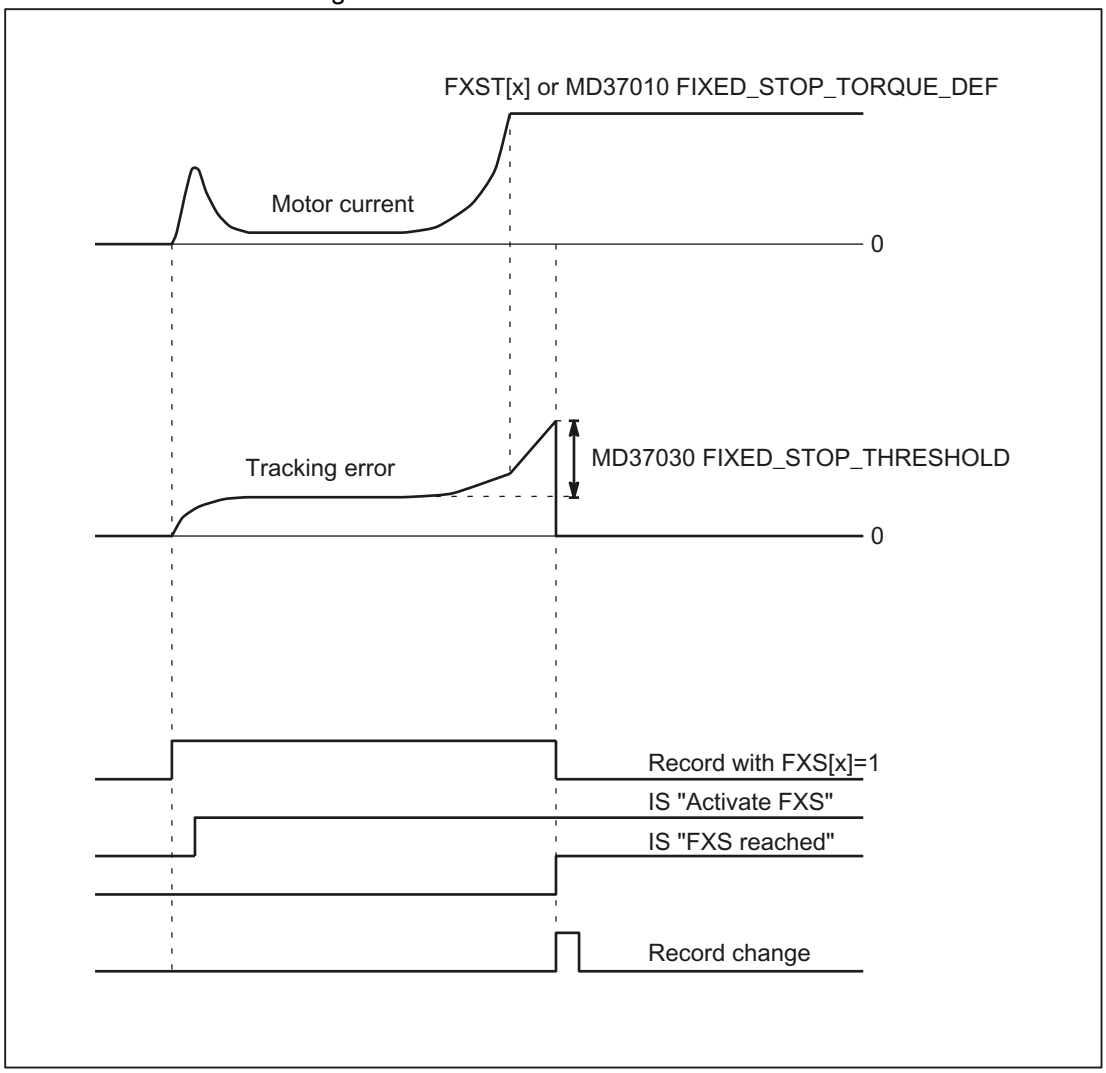


Fig. 2-6 Diagram for FXS deselection with analog drive

Supplementary Conditions

3

3.1 Supplementary conditions

There are no supplementary conditions to note.

4 Examples

4.1 Travel to fixed stop

Static synchronized actions

Travel to fixed stop (FXS), initiated by a synchronized action.

```
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 in dependence
(($R3==1) AND                   ; On R3 and
($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)                  ; Axis Y for the movement to
                                ; Activate synchronized action

N40 G1 F1000 X100               ; Movement of another axis
N50 ......
```

```
N60 GET(Y)                    ; Include axis Y again in the
                              ; Pathgroup
```

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 flag (here R1) in the condition will ensure that the function is not activated more than once.

Programming example (parts program fragment):

```
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 blockrelated synchronized action, travel to fixed stop can be connected during an approach motion.

Programming example:

```
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                      ; 17mm FXS is activated
```

Data Lists

5

5.1 Machine data

5.1.1 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
36042	FOC_STANDSTILL_DELAY_TIME	Delay time 0 monitoring with FOC and FXS
37000	FIXED_STOP_MODE	Travel to fixed stop mode
37002	FIXED_STOP_CONTROL	Special function when traveling to fixed stop
37010	FIXED_STOP_TORQUE_DEF	Default setting for clamping torque
37012	FIXED_STOP_TORQUE_RAMP_TIME	Virtual time until reaching the new clamping torque when traveling to fixed stop
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	Status of activation of the modal function FOC

5.2 Setting data

5.2.1 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43500	FIXED_STOP_SWITCH	Selection of travel to fixed stop
43510	FIXED_STOP_WINDOW	Clamping torque when traveling to fixed stop extended to a torque greater than 100%
43520	FIXED_STOP_TORQUE	Fixed stop monitoring window

5.3 Signals

5.3.1 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	1.1	Acknowledge fixed stop reached
31, ...	1.2	Sensor for fixed stop
31, ...	1.3	Axis/spindle disable
31, ...	2.1	Controller enable
31, ...	3.1	Enable travel to fixed stop
31, ...	62.4	Activate travel to fixed end stop
31, ...	62.5	Fixed stop reached

Index

B

Block-related limit (FOC), 2-21

C

Channel axis identifiers with FXS, 2-1

F

FXS REPOS, 2-13

M

MD1012, 2-18
MD1103, 2-15
MD1104, 2-15
MD1105, 2-15
MD1118, 2-1
MD1130:, 2-1
MD1230/1231, 2-15
MD36040:, 2-22
MD36042, 2-22
MD37002, 2-8, 2-9, 2-17, 2-18, 2-25
MD37010, 2-3, 2-14, 2-21
MD37012, 2-8, 2-15, 2-20
MD37020, 2-5, 2-14
MD37030, 2-4, 2-23, 2-29, 2-31
MD37040, 2-4
MD37050, 2-5, 2-15, 2-23, 2-25
MD37052, 2-6, 2-16, 2-17
MD37060, 2-3, 2-4, 2-6, 2-7, 2-19, 2-23, 2-24, 2-28, 2-29, 2-30, 2-31, 2-32, 2-33
MD37070, 2-28, 2-29, 2-31, 2-32
MD37080, 2-20
Modal activation (FOCON/FOCOF), 2-20

R

REPOS Offset, 2-13

S

SD43500, 2-14
SD43510, 2-14, 2-15
SD43520, 2-14
SERUPRO, 2-10
SERUPRO ASUP, 2-12

T

Travel to fixed stop
- Analog drives, 2-27
- Analog drives, diagrams, 2-34
- Analog drives, FXS deselection, 2-36
- Analog drives, FXS selection, 2-34
- Analog drives, SIMODRIVE 611A (FDD), 2-28
- Analog drives, SIMODRIVE 611A (MSD), 2-30
- Block search, 2-9
- Contour monitoring, 2-16
- Customer benefit, 1-1
- Deselection, 2-7
- Function abort, 2-9
- Functional sequence, 2-3
- Monitoring window, 2-5
- Positioning axes, 2-16
- Programming, 2-1
- RESET, 2-8
- Selection, 2-3
- Setting data, 2-14, 2-25
- Status query, 2-15

Travel to fixed stop (F1)
- Interrupts, 2-6
- Setting data, 2-5

V

Vertical axes, 2-16

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 Velocities, traversing ranges, accuracies 2-1
2.1.1 Velocities 2-1
2.1.2 Traversing ranges 2-3
2.1.3 Positioning accuracy of the control system 2-4
2.1.4 Block diagram of resolutions and scaling values 2-5
2.1.5 Input/display resolution, computational resolution 2-6
2.1.6 Scaling of physical quantities of machine and setting data 2-7

2.2 Metric/inch measuring system 2-11
2.2.1 General 2-11
2.2.2 Conversion of basic system by parts program 2-12
2.2.3 Manual switchover of the basic system 2-16
2.2.4 FGROUP and FGREF 2-20

2.3 Setpoint/actual-value system 2-23
2.3.1 General 2-23
2.3.2 Speed setpoint and actual-value routing 2-25
2.3.3 Configuration of drives 2-32
2.3.4 Adapting the motor/load ratios 2-33
2.3.5 Speed setpoint output 2-37
2.3.6 Actual-value processing 2-39
2.3.7 Adjustments to actual-value resolution 2-42

2.4 Closed-loop control 2-49
2.4.1 General 2-49
2.4.2 Parameter sets of the position controller 2-54
2.4.3 Extending the parameter set 2-56

2.5 Optimization of the control 2-59
2.5.1 Position controller: injection of positional deviation 2-59
2.5.2 Position controller position setpoint filter: new balancing filter 2-61
2.5.3 Position controller position setpoint filter: new jerk filter 2-68
2.5.4 Position control with proportional-plus-integral-action controller 2-71
2.5.5 System variable for status of pulse enable 2-73
2.5.6 Expansions for "deceleration axes" 2-74

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Examples 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 Memory specific machine data 5-1
5.1.2 NC-specific machine data 5-1
5.1.3 Channel-specific machine data 5-2
5.1.4 Axis/spindle-specific machine data 5-2

Index Index-1

Tables

Table 2-1 Traversing ranges of axes 2-4

Table 2-2 Special features of the feedrate weighting for rotary axes in FGROUP: 2-21

Brief Description

1

1.1 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

Detailed Description

2

2.1 Velocities, traversing ranges, accuracies

2.1.1 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 defined via machine data:
MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed).

For more information, see:
References:
/FB1/Description of Functions, Basic Machine; Spindles (S1)

As well as limiting using MD32000, the control limits the maximum path velocity in relation to the situation and according to the following formula:

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{\text{ENC_RESOL [n]} * \text{internal multiplication}} * \frac{\text{DRIVE_ENC_RATIO_NUMERA [n]}}{\text{DRIVE_ENC_RATIO_DENOM [n]}} * \frac{\text{DRIVE_AX_RATIO_DENOM [n]}}{\text{DRIVE_AX_RATIO_NUMERA [n]}} * \text{LEADSCREW_PITCH} * \text{INT_INCR_PER_MM}$$

For setting the interpolation cycle, see:
References:
/FB3/Description of Functions, Special Functions; Cycle Times (G3)

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

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 via 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.

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{\text{ENC_RESOL [n]} * \text{internal multiplication}} * \frac{\text{DRIVE_ENC_RATIO_NUMERA [n]}}{\text{DRIVE_ENC_RATIO_DENOM [n]}} * \frac{\text{DRIVE_AX_RATIO_DENOM [n]}}{\text{DRIVE_AX_RATIO_NUMERA [n]}} * \text{LEADSCREW_PITCH} * \text{INT_INCR_PER_MM}$$

Example:

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 range of values of the feedrates depends on the computational resolution selected.
If
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./degree)
is assigned its default values, the following range of values can be programmed with the input resolution:

Range of values for path feedrate 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 of 10, the ranges of values change accordingly.

2.1.2 Traversing ranges

Range of values of the traversing ranges

The range of values of the traversing range depends on the computational resolution selected.

If 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./degree)
are assigned their default values, the following range of values can be programmed with the input resolution:

Table 2-1 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.

References:
/FB1/Description of Functions, Basic Machine; Axis Monitoring, Protection Zones (A3)

The traversing range for rotary axes can be limited via machine data.

References:
/FB2/Description of Functions, Extended Functions; Rotary Axes (R2)

For special features of linear and rotary axes with a large traversing range, see:
References:
/FB1/Description of Functions, Basic Machine; Reference Point Approach (R1)

2.1.3 Positioning accuracy of the control system

Actual-value resolution and computational resolution

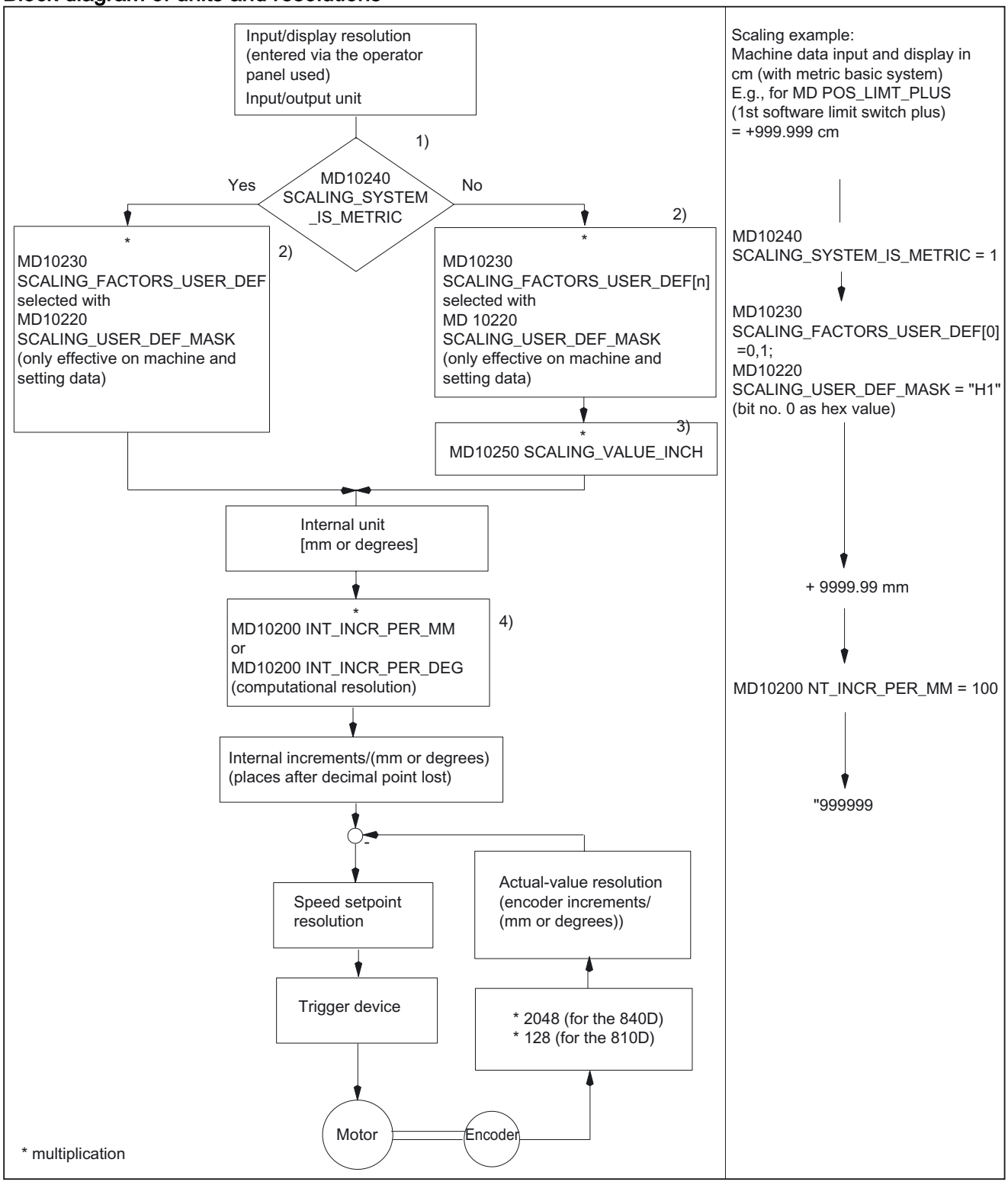
The positioning accuracy of the control depends on the actual-value resolution (= encoder increments/(mm or degrees)) and the computational resolution (= internal increments/(mm or degrees)).

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.

2.1.4 Block diagram of resolutions and scaling values

Block diagram of units and resolutions



This diagram shows how input values are converted into internal units.

It also shows the following conversion to internal increments/(mm or degrees), which can cause loss of decimal places if the computational resolution was selected to be coarser than the input resolution.

In addition, it provides an overview of the following topics:

- Selection of measuring system (metric/inch)
- Scaling of physical quantities of machine and setting data
- Conversion of basic system
- Setting of computational resolution

One example is the conversion of a physical quantity
(MD36110 $MA_POS_LIMIT_PLUS)
depending on the machine data parameterization
(MD10230 $MN_SCALING_FACTORS_USER_DEF,
MD10220 $MN_SCALING_USER_DEF_MASK).

2.1.5 Input/display resolution, computational resolution

Resolutions: Differences

The following types of resolution, e.g. resolution of linear and angular positions, velocities, accelerations, and jerk, must be differentiated as follows:

- Input resolution

  Data is input via the control panel or parts programs.

- Display resolution

  Data is displayed via the control panel.

- Computational resolution

  Data input via the control panel or parts program is displayed internally.

The input and display resolution is defined via the control panel being used, whereby the display resolution of position values can be changed with machine data:
MD9004 $MM_DISPLAY_RESOLUTION.

Machine data:
MD9011 $MM_DISPLAY_RESOLUTION_INCH
can be used to configure the display resolution of position values using the inch system. This allows you to display up to six decimal places with the inch setting.

For the programming of parts programs, the input resolutions listed in the Programming Guide apply.

The desired computational resolution is defined via 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 parts 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 computational resolution.

Example of rounding:

Computational resolution: 1000 increments/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 parts program:

```
N20 G0 X 1.0000 Y 1.0000             ;  Traverse axes to position
                                        X=1.0000 mm, Y=1.0000 mm
N25 G0 X 5.0002 Y 2.0003             ;  Traverse axes to position
                                        X=5.0002 mm, Y=2.0003 mm
```

2.1.6 Scaling of physical quantities of machine and setting data

Input/output units

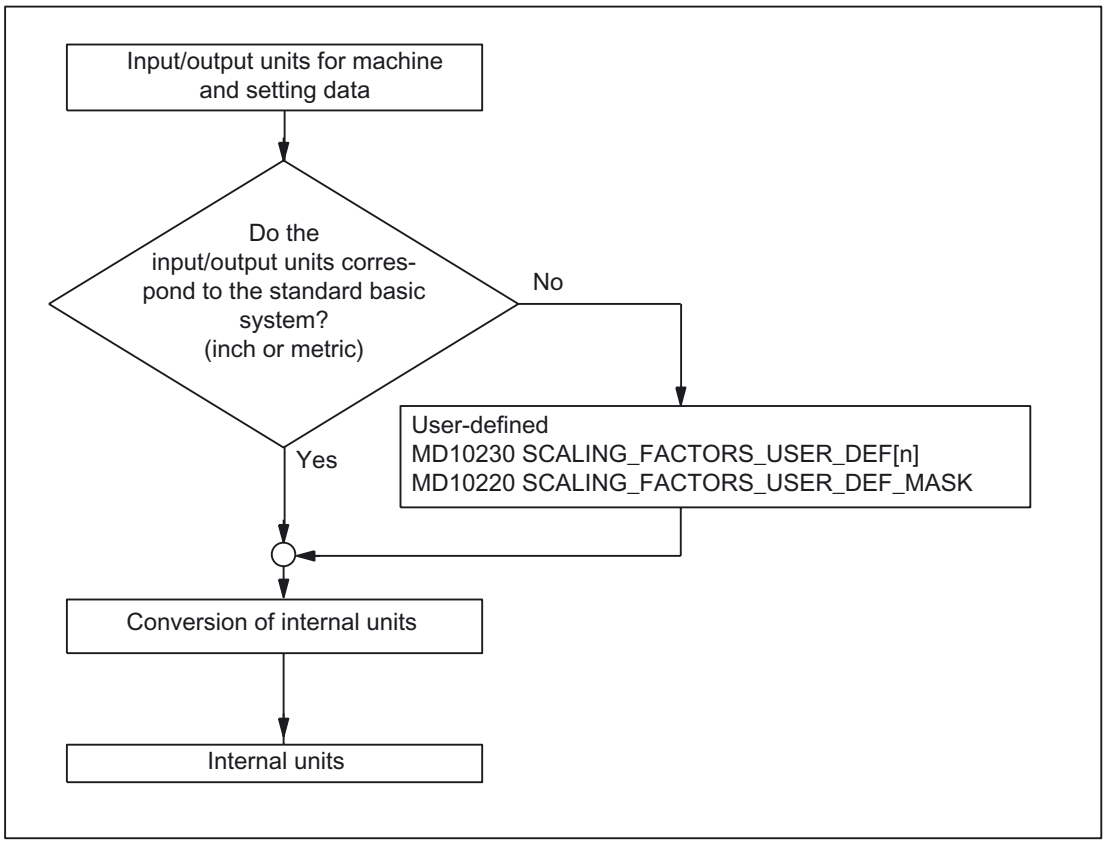
Machine and setting data that possess a physical quantity are interpreted in the input/output units below depending on whether the metric or inch system is selected:

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 $rev./s^2$	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 1	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 degree/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.
In order to achieve this, an adjustment must be made between the newly selected input/output units and the internal units, via 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).



The following applies:

Selected input/output unit = MD10230 * internal unit

The selected input/output unit is input into machine data:
MD10230 $MN_SCALING_FACTORS_USER_DEF[n],
expressed in internal units of 1 mm, 1 degree or 1 s.

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. In this case, bit number 2 must be set in machine data:
MD10220 $MN_SCALING_USER_DEF_MASK.

⇒ 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. In this case, bit number 2 must be set in machine data:
MD10220 $MN_SCALING_USER_DEF_MASK.

⇒ MD10220 $MN_SCALING_USER_DEF_MASK = 'H4'; (bit no. 2 as hex value)

⇒ The scaling factor is calculated using the following formula:

$$\text{MD10230 SCALING_FACTORS_USER_DEF[n]} = \frac{\text{Input/output unit selected}}{\text{Internal unit}}$$

$$\text{MD10230 SCALING_FACTORS_USER_DEF[n]} = \frac{1\,\frac{\text{m}}{\text{min}}}{1\,\frac{\text{mm}}{\text{s}}} = \frac{\frac{1000\ \text{mm}}{60\ \text{s}}}{1\,\frac{\text{mm}}{\text{s}}} = \frac{1000}{60} = 16.667;$$

⇒ MD10230 SCALING_FACTORS_USER_DEF[n] = 16.667

Index 2 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 is to be performed 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)

$$\Rightarrow \text{MD10220 SCALING_FACTORS_USER_DEF[n]} = \frac{1\,\frac{\text{ft}}{\text{s}^2}}{1\,\frac{\text{mm}}{\text{s}^2}} = \frac{12*25.4\,\frac{\text{mm}}{\text{s}^2}}{1\,\frac{\text{mm}}{\text{s}^2}} = 304.8;$$

⇒ MD10230 SCALING_FACTORS_USER_DEF[4] = 304.8

Index 4 defines the "Linear acceleration" in the "Scaling factors of physical quantities" list.

2.2 Metric/inch measuring system

2.2.1 General

The control system can operate with the inch or the metric system of measurement.

Initial state

The initial state is defined via the following machine data element:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC (basic metric system).

Depending on the setting in the MD, all geometric values are interpreted either as metric or inch values.

All manual settings also refer to this initial state (e.g. handwheel, INC, JOG feedrate), as do zero offsets, tool offsets, FRAMES, etc., and the associated displays.

Supplements

The setting:
MD10260 $MN_CONVERT_SCALING_SYSTEM=1
greatly simplifies a change in the measuring system.

- HMI user interface: Softkey in the "Machine" operating area for changing the measuring system.
- Automatic conversion of active NC data when the measuring system is changed.
- Data backup is performed with the actual measuring system identifier.
- The measuring system for sag compensation is configured in machine data element:

  MD32711 $MA_CEC_SCALING_SYSTEM_METRIC

  References:
  /FB2/Description of Functions, 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:
  /FB2/Description of Functions, Extended Functions; Software Cams, Limit Switching Signals (N3)/Indexing Axes (T1)

2.2.2 Conversion of basic system by parts program

Conversion factor

The factor for inch/metric conversion for data input and output can be changed via machine data element:
MD10250 $MN_SCALING_VALUE_INCH (conversion factor for changing to the inch system).

Note

The machine data element is not visible unless the password of protection level "Siemens" is set.

By changing the default value, the control can be adapted to a customer-specific measuring system.

When programming, it is possible to change over between measuring systems for some workpiece-related specifications with G70/G71/G700/G710. The data that can be influenced by G70/G71/G700/G710 is described in the Programming Guide.

The initial state of G70/G71/G700/G710 can be defined channel-specifically via the G group initial state (reset) in machine data:
MD20150 $MC_GCODE_RESET_VALUES[12].

When changing the measuring system via the relevant soft key on the HMI, these initial settings are automatically predefined to appropriate settings for the new measuring system with G700 or G710.

The axis-specific machine data element:
MD31200 $MA_SCALING_FACTOR_G70_G71
(factor for converting values with active G70/G71/G700/G710)
can be used to freely select the conversion factor between metric and inch conversion with G70/G71/G700/G710 programming.
The controller can be adapted to a customized measuring system by changing the predefined settings. The factor only takes effect if parts program programming differs from the initial state.

Example:

Initial state: inch; (MD10240 $MN_SCALING_SYSTEM_IS_METRIC=0)

Parts program:

```
N15 G70             ; Factor is not effective, since the setting is exactly the
                      same as the initial state.
N20 G71             ; Factor is effective.
```

The current setting (selected with G70/G71) and the initial state can be the same or differ at any given time. The current setting is channel-specific, the initial state applies to all channels.

Application:

With this function it is possible, for example, with a metric basic system, to machine an inch thread in a metric parts program. Tool offsets, zero offsets and feedrates remain metric.

Machine data are displayed on the screen using the basic system selected in machine data: MD10240 $MN_SCALING_SYSTEM_IS_METRIC (metric basic system).

Machine coordinates, tool data and zero offsets are always displayed in the initial state, workpiece coordinate displays are in the current setting.

If programs including data sets (zero offset, tool offset) programmed in the measuring system, which differs from the basic system, are read in from an external source, the initial state must first be changed via machine data MD10240.

Data exchange with the PLC of interface signals containing dimension information, e.g. feedrate for path and positioning axes, is always carried out in the selected basic system.

Supplements

G700/G710 extends 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 parts program are read and written in the programmed measuring system.

This allows you to implement parts programs that are independent of the current default measuring system.

Comparison of the effect of G70/G71 and G700/G710 on machine data and system variables in the parts program:

- With G70/G71: reading/writing in the basic system
- With G700/G710: reading/writing in the programmed measuring system

Examples:

If MD10240 $MN_SCALING_SYSTEM_IS_METRIC=1,
both part programs are executed with a metric setting.

```
N100 R1=0 R2=0
N120 G01 G70 X1 F1000
N130 $MA_LUBRICATION_DIST[X]=10
N140 NEWCONF
N150 IF ($AA_IW[X]>$MA_LUBRICATION_DIST[X])
N160 R1=1
N170 ENDIF
N180 IF ($AA_IW[X]>10)
N190 R2=1
N200 ENDIF
N210 IF ( (R1<>0) OR (R2<>0))
N220 SETAL(61000)
N230 ENDIF
N240 M30
```

Note

Alarm 61000 is not output if `G70` is replaced by `G700` in block `N120`.

Synchronized actions

In order to prevent the current parts program context from changing the positioning behavior of a synchronized action arbitrarily in response to asynchronous trigger conditions, the measuring system must be defined at the time of interpretation. This is the only way to achieve a defined and reproducible positioning behavior of a synchronized action.

Example 1:

```
N100 R1=0
N110 G0 X0 Z0
N120 WAITP(X)
N130 ID=1 WHENEVER $R1==1 DO POS[X]=10
N140 R1=1
N150 G71 Z10 F10              ;Z=10 mm          X=10 mm
N160 G70 Z10 F10              ;Z=254 mm         X=254 mm
N170 G71 Z10 F10              ;Z=10 mm          X=10 mm
N180 M30
```

Example 2:

The definition is made here by programming G71 in the synchronized action.

```
N100 R1=0
N110 G0 X0 Z0
N120 WAITP(X)
N130 ID=1 WHENEVER $R1==1 DO G71 POS[X]=10
N140 R1=1
N150 G71 Z10 F10              ;Z=10 mm          X=10 mm
N160 G70 Z10 F10              ;Z=254 mm         X=10 mm (X posit. always at 0 mm)
N170 G71 Z10 F10              ;Z=10 mm          X=10 mm
N180 M30
```

Comparison: G70/G71-G700/G710

Where:

P:	Data is read/written in the programmed measuring system
G:	Data is read/written in the basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC).
R/W:	Read/Write

Comparison:

Range	G70/G71	G700/G710
	Parts program	Parts program
	R/W	R/W
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
Pitch	P/P	P/P
Programmable FRAME	P/P	P/P
Settable FRAMES	G/G	P/P
Basic frames	G/G	P/P
Work offsets external	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

Range	G70/G71	G700/G710
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

References:
/PG/Programming Guide, Fundamentals; List of Addresses

2.2.3 Manual switchover of the basic system

General

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 boundary 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 doe 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.

Machine data:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC
and the corresponding G70/G71/G700/G710 settings in machine data:
MD20150 $MN_GCODE_RESET_VALUES
are switched automatically and consistently for all configured channels.

The value of 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 soft key 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 prerequisites for a switchover are not fulfilled on one of the NCUs linked, no switchover will take place on none 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:
/FB2/Description of Functions, Extended Functions; Several Control Panels on Multiple NCUs, Decentralized 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
- Acceleration rates
- Jerk
- Tool offsets
- Programmable, settable and work offsets external 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)
- PUD (**P**rogram global **U**ser **D**ata)
- Analog inputs/outputs
- Data exchange via FC21

The user is prompted to take the current valid measuring system:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC
into account.

The current measuring system setting can be read at the PLC interface via the "inch measuring system" signal:
DB10 DBX107.7.

Signal:
DB10 DBB71 (inch/metric measuring system modification counter)
can be used for reading the "measuring system modification counter".

User tool data

Additional machine data sets are introduced for user-defined tool data:
MD18094 $MN_MM_NUM_CC_TDA_PARAM
and cutting edge data:
MD18096 $MN_MM_NUM_CC_TOA_PARAM:

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 lengthrelated userdefined tool data are 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 control via 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

Machine data:
MD31090 $MA_JOG_INCR_WEIGHT
consists of two values, which contain the axial increment factors for both measuring systems.

Depending on the current setting in machine data:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC,
the control selects the correct value automatically.

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 read separately from the control and have access to data relevant to the measuring system, receive an INCH or METRIC identifier during the read action, depending on machine data:
MD10260 $MN_CONVERT_SCALING_SYSTEM.
The identifier corresponds to machine data:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC.
This records 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 parts programs, these can also be "protected" against operator errors as described above. You can, therefore, prevent parts programs containing only metric data, for example, from running on an inch measuring system.

Archives and machine data sets are downward compatible if:
MD11220 $MN_INI_FILE_MODE = 2.

Note

The INCH/METRIC instruction is only generated if compatibility machine data:
MD10260 $MN_CONVERT_SCALING_SYSTEM
is set.

Rounding machine data

All length-related machine data are 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.

2.2.4 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 [*<axis name>*] = 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 factor 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 setting is independent of the active basic system:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC
and of the currently active inch/metric G code.

Table 2-2 Special features of the feedrate weighting for rotary axes in FGROUP:

```
N100 FGROUP(X,Y,Z,A)
N110 G1 G91 A10 F100
N120 G1 G91 A10 X0.001 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.
The time required to execute the two blocks can be differ greatly.

Note

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 setting, when:

- G71/G710:
  FGREF[A]=57.296
- G70/G700:
  FGREF[A]=57.296/25.4

Example:

The following example illustrates the effect of FGROUP on the path and the path feedrate.

```
N100 R1=0
N110 FGROUP(X,A)
N120 G91 G1 G710 F100              ;  Feedrate=100 mm/min or 100 deg/min
N130 DO $R1=$AC_TIME
N140 X10                           ;  Feedrate=100 mm/min
                                   ;  Path=10 mm
                                   ;  R1=6 s approx.
N150 DO $R2=$AC_TIME
N160 X10 A10                       ;  Feedrate=100 mm/min
                                   ;  Path=14.14 mm
                                   ;  R2=8 s approx.
```

```
N170 DO $R3=$AC_TIME
N180 A10                          ;  Feedrate=100 degrees/min
                                  ;  Path=10 degrees
                                  ;  R3=6 s approx.
N190 DO $R4=$AC_TIME
N200 X0.001 A10                   ;  Feedrate=100 mm/min
                                  ;  Path=10 mm
                                  ;  R4=6 s approx.
N210 G700 F100                    ;  Feedrate=2540 mm/min or 100 deg/min
N220 DO $R5=$AC_TIME
N230 X10                          ;  Feedrate=2540 mm/min
                                  ;  Path=254 mm
                                  ;  R5=6 s approx.
N240 DO $R6=$AC_TIME
N250 X10 A10                      ;  Feedrate=2540 mm/min
                                  ;  Path=254.2 mm
                                  ;  R6=6 s approx.
N260 DO $R7=$AC_TIME
N270 A10                          ;  Feedrate=100 degrees/min
                                  ;  Path=10 degrees
                                  ;  R7=6 s approx.
N280 DO $R8=$AC_TIME
N290 X0.001 A10                   ;  Feedrate=2540 mm/min
                                  ;  Path=10 mm
                                  ;  R8=0.288 s approx.
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=6 s approx.
N330 M30
```

Note

The variable $AC_TIME contains the time from the start of the block in seconds. It can only be used in synchronized actions.

2.3 Setpoint/actual-value system

2.3.1 General

Control loop

A control loop with the following structure can be configured for every closed-loop controlled axis/spindle:

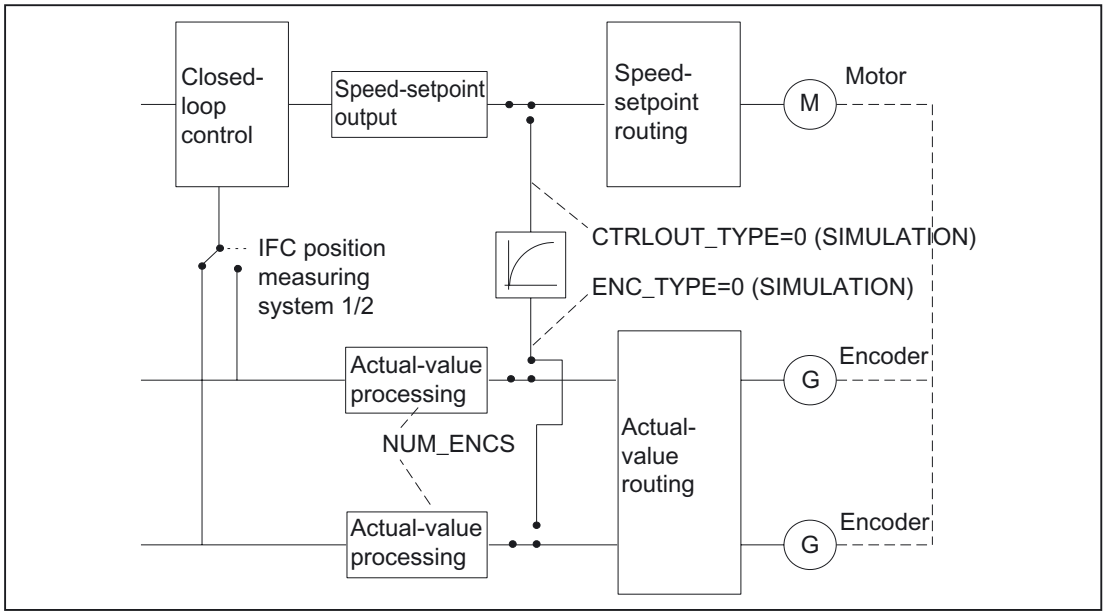


Fig. 2-1 Block diagram of a control loop

Setpoint output

A setpoint can be output for each axis/spindle. Setpoints are output digitally to the actuator on SINUMERIK 840D/810D.

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 highspeed positioning tasks.

Enter the number of encoders used in 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 control.

Reference point approach is executed by the selected measuring system.
Each positioning measuring system must be referenced separately.

For an explanation of actual-value acquisition compensation functions, see:
References:
/FB2/Description of Functions, Extended Functions; Compensations (K3)

For an explanation of encoder monitoring, see:
References:
/FB1/Description of Functions, Basic Machine; Axis Monitoring, Protection Zones (A3)

Switching between measuring systems

You can switch between the two measuring systems using the NC/PLC interface signals:
DB31, ... DBX1.5 (positioning measuring system 1)
and
DB31, ... DBX1.6 (positioning measuring system 2).

References
/FB1/Description of Functions, Basic Machine; Various Interface Signals (A2)

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 related tolerance is entered in machine data:
MD36500 $MA_ENC_CHANGE_TOL (maximum tolerance for position actual-value switchover).

On switchover, the current difference between position measuring system 1 and 2 is traversed immediately.

The permissible difference between the actual values of the two measuring systems must be entered in machine data:
MD36510 $MA_ENC_DIFF_TOL.

This tolerance setting must not be exceeded during cyclical comparison of the two measuring systems or an error message will be generated.

Related monitoring is not active if MD35510=0, if 2 measuring systems are not active/available on the axis or if the axis is not referenced (at least current control measuring system).

Types of actual-value acquisition

The encoder type used must be defined in machine data:
MD30240 $MA_ENC_TYPE (type of actual-value acquisition (position actual 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 machine data:
MD30130 $MA_CTRLOUT_TYPE[n] (setpoint output type)
and

MD30240 $MA_ENC_TYPE[n] (actual-value acquisition type)
to "0".

As soon as the standard machine data have been loaded, the axes become simulation axes.

The setpoint and actual value can be set to the reference point value with reference point approach.

Machine data:
MD30350 $MA_SIMU_AX_VDI_OUTPUT (axis signal output on simulation axes)
can be used to define if the axis-specific interface signals are output to the PLC during the simulation.

Actual-value correction

If actual-value corrections performed by the NC on the encoder selected for position control are not to influence the actual value of any other encoder defined in the same axis, then the position control encoder must be declared to be "independent" via machine data:
MD30242 $MA_ENC_IS_INDEPENDENT.

Actual-value corrections include the following:

- Modulo treatment
- Reference point approach
- Measuring system comparison
- PRESET

2.3.2 Speed setpoint and actual-value routing

General

In order to carry out speed setpoint and actual-value routing, the following must be defined for each axis/spindle:

- Assignment of 1st measuring circuit
- Assignment of 2nd measuring circuit (if present)
- Assignment of setpoint branch

Multiple assignment is also possible, e.g. using a measuring circuit for position actual-value acquisition for the alternating control of several axes/spindles.

Note

When a SIMODRIVE 611 universal is operated via the PROFIBUS DP, various machine data that need to be parameterized for digital and analog drives are not assigned.

Prerequisite for routing

All NC machine axes must be uniquely defined in machine data:
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[n] (machine axis name).

This name must be unique throughout the system (all mode groups and channels).

References:
/FB1/Description of Functions, Basic Machine;
Axes, Coordinate Systems, Frames (K2)/
Mode Group, Channel, Program Operation, Reset Behavior (K1)

Speed setpoint routing

For speed setpoint routing, the following setpoint assignments for parameterizing the relevant machine data must be made:

Setpoint assignment	Number
Drive type:	Of the bus segment
Drive number/module number:	Of the module within the bus segment
Output on drive module/module:	Of the setpoint output
Output type of setpoint:	Type of speed setpoint output
Setpoint output is unipolar (only active in conjunction with PROFIBUS DP)	Polarity of the output driver for speed setpoint output

Index of MD for speed setpoint routing

The index [n] of the machine data for setpoint routing is coded with 0 for setpoint assignment with default setting 1.

Speed setpoint routing

The following machine data need to be parameterized for each setpoint branch:

MD30100 $MA_CTRLOUT_SEGMENT_NR[n] (setpoint assignment of bus segment):	
The number of the bus segment, via which the output is addressed, is entered here. Depending on the SINUMERIK version, certain bus segments are preassigned.	
• Local bus (similar to SINUMERIK FM NC, for example)	= 0
• 611D I/O bus (1st DCM, e.g. SINUMERIK 810D)	= 1
• Local I/O bus	= 2
• 611D I/O bus (2nd DCM)	= 3
• Reserved for virtual buses	= 4
• PROFIBUS-DP (NCU 573.2 and higher) (PROFIBUS line for ProfiSafe at PLC end)	= 5
• PROFIBUS DP link module (NCU 573.2 and higher)	= 6

MD30110 $MA_CTRLOUT_MODULE_NR[n] (setpoint assignment: drive number/module number):	
The number of the module in the bus segment, via which the output is to be addressed, is entered here. The logical drive number of the axis module can be set via machine data: MD13010 $MN_DRIVE_LOGIC_NR[n] for:	
• SINUMERIK 810D	Range of values 0 - 15
• SINUMERIK 840D as 611 digital drive number	Range of values 0 -31
• SINUMERIK 840D PROFIBUS on link module	Range of values 0 -125

MD30120 $MA_CTRLOUT_NR[n]) (setpoint assignment: setpoint output on drive module/module):
The number of the setpoint output must be entered here (always output 1 for SINUMERIK 840D/810D).

MD30130 $MA_CTRLOUT_TYPE[n] (setpoint output type):	
The speed setpoint output type is entered here.	
0:	Simulation (HW not required)
1:	Standard (differentiated via HW configuration)
2:	Stepper motor and 3: FM module (for SINUMERIK FM NC only)

MD30134 IS_UNIPOLAR_OUTPUT[n] (setpoint output is unipolar):
The unipolar speed setpoint output works only in conjunction with PROFIBUS DP.

Actual-value routing

For actual-value routing, the following actual-value assignments for parameterizing the associated machine data must be made:

Actual-value assignment	Number:
Drive type:	Of the bus segment
Drive number/module number:	Of the module within the bus segment
Input on drive module/measuring circuit module:	Of the setpoint input
Type of actual-value acquisition (position actual value):	The encoder type used
Encoder set independently/dependently	The encoder is independent Actual-value corrections by the NC

Machine data actual-value routing

The following machine data must be parameterized for each actual-value branch:

MD30210 $MA_ENC_SEGMENT_NR[n] (actual-value assignment of bus segment):	
The number of the bus segment, via which the encoder is addressed, is entered here. Depending on the SINUMERIK version, certain bus segments are preassigned.	
• Local bus (similar to SINUMERIK FM NC, for example)	= 0
• 611D I/O bus (1st DCM, e.g. SINUMERIK 810D)	= 1
• Local I/O bus	= 2
• 611D I/O bus (2nd DCM)	= 3
• Reserved for virtual buses	= 4
• PROFIBUS-DP (NCU 573.2 and higher) (PROFIBUS line for ProfiSafe at PLC end)	= 5
• PROFIBUS DP link module (NCU 573.2 and higher)	= 6

MD30220 $MA_ENC_MODULE_NR[n]) (actual-value assignment: drive module number/measuring circuit number):	
The number of the module in the bus segment, via which the encoder is addressed, is entered here. The logical drive number of the axis module can be set via machine data: MD13010 $MN_DRIVE_LOGIC_NR[n] for:	
• SINUMERIK 810D	Range of values 0 - 15
• SINUMERIK 840D as 611 digital drive number	Range of values 0 -31
• SINUMERIK 840D PROFIBUS on link module	Range of values 0 -125

MD30230 $MA_ENC_INPUT_NR[n]) (actual-value assignment: input on drive module/measuring circuit module):	
The number of the input, to which the position actual-value encoder is connected, is entered here.	
• SINUMERIK 840D/810D:	= 1 or 2 (counting down)
• SINUMERIK FM NC:	= 1 - 4, in accordance with input selected X3 - X6

MD30240 $MA_ENC_TYPE[n] (type of actual-value acquisition):
Enter the encoder type used here.

MD30242 $MA_ENC_IS_INDEPENDENT[n] (encoder is independent):	
To prevent actual-value corrections influencing the actual value of an encoder defined in the same axis, the latter must be declared independent.	
0:	Encoder is independent
1:	Encoder is dependent

MD index for actual-value routing

The coding of the machine data index [n] for actual-value routing is:

[Encoder no.]	
0	For first encoder
1	For second encoder

Examples of setpoint/actual-value routing

SINUMERIK 840D/810D with SIMODRIVE 611 digital

For machine axis "X1", the setpoint should be output digitally and actual values acquired on drive module 4 (4th slot = index [3]).

The "logical drive number" of this module is 7.

Encoder number: 1, 2

Therefore, actual values are acquired via a direct and indirect measuring system.

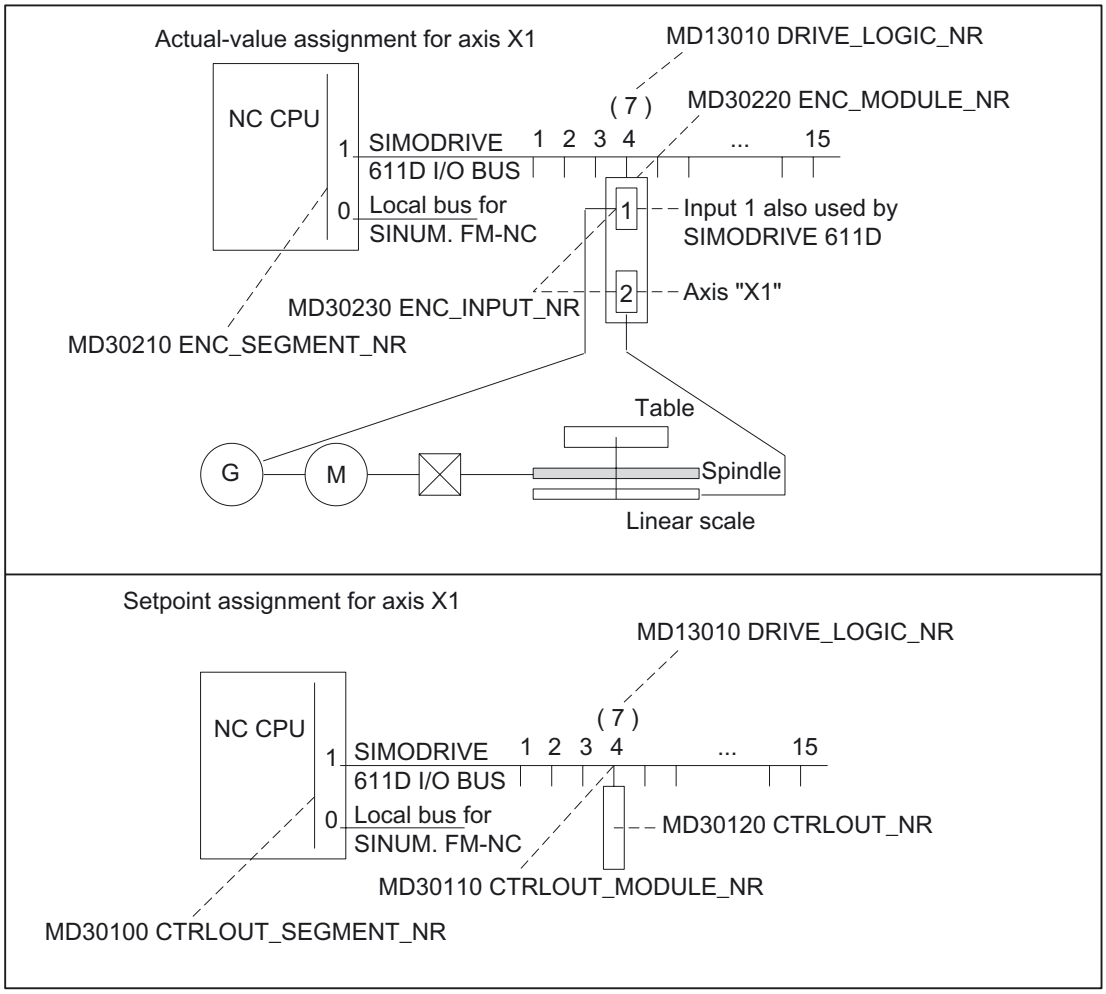


Fig. 2-2 Example of setpoint/actual-value routing

Special features of SINUMERIK 840D/810D with SIMODRIVE 611 digital:

- MD30110 $MA_CTRLOUT_MODULE_NR[n]
  and
  MD30220 $MA_ENC_MODULE_NR[n]
  always have the same logical drive number with either indirect measuring systems or if the motor encoder has to be evaluated in the NC.
- If direct measuring systems are installed, you can also configure encoders that are connected to other drives.

SINUMERIK 810D with axis expansion interface

Integrated drive no. 4 should be assigned to machine axis "X1":

Motor measurement channel	X414
Pulse interface	X304

Measurement channel no. 6 should be assigned to machine axis "X1":
X416 as direct measuring system.

The logical number of the drive is set automatically to 4.

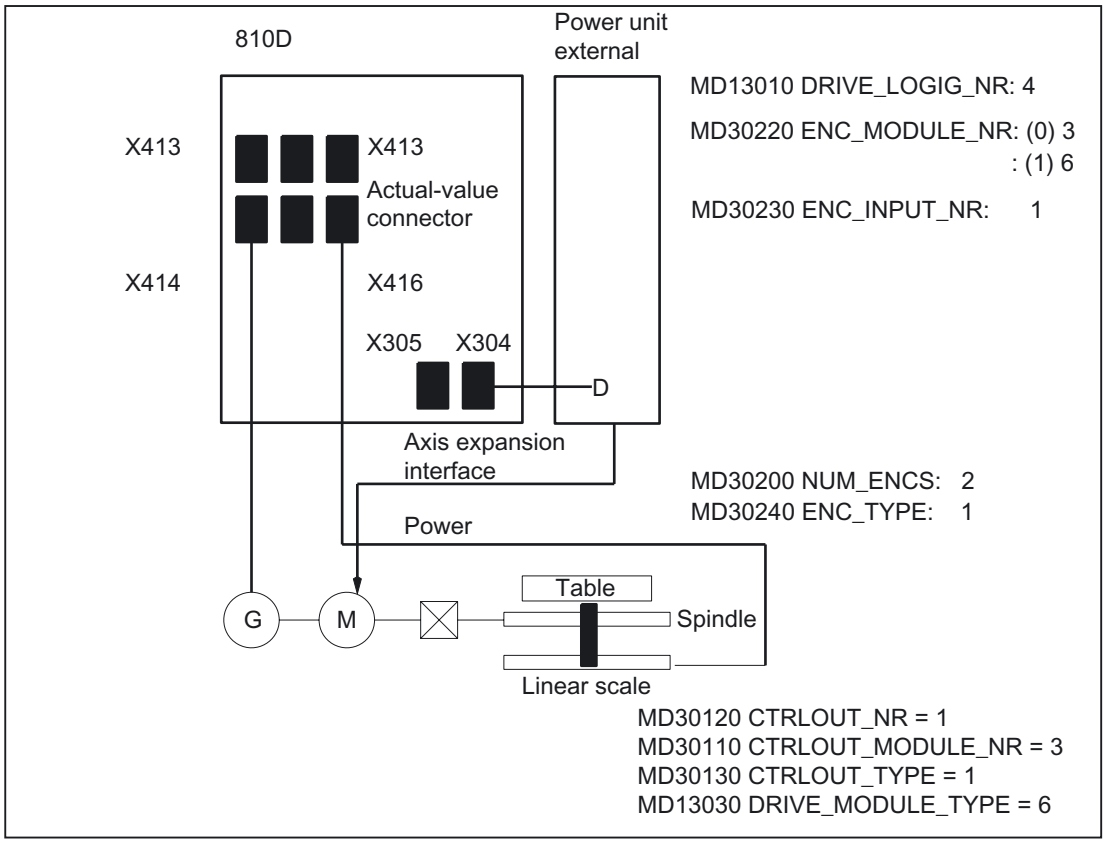


Fig. 2-3 Example of setpoint/actual-value routing with axis expansion interface

Actual-value assignment		Machine data parameterization for 810D	
1st axis actual value from motor encoder	(X414)	MD30220 $MA_ENC_MODULE_NR[0] MD30230 $MA_ENC_INPUT_NR[0]	= 4 = 1
2nd axis actual value from linear scale	(X416)	MD30220 $MA_ENC_MODULE_NR[1] MD30230 $MA_ENC_INPUT_NR[1]	= 3 = 1
Number of encoders:		MD30200 $MA_NUM_ENCS	= 2
Actual-value acquisition modes	1:	MD30240 $MA_ENC_TYPE[0]	= 1
	2:	MD30240 $MA_ENC_TYPE[1]	= 1

Setpoint assignment	
MD30110 $MA_CTRLOUT_MODULE_NR[0]	= 4
MD30120 $MA_CTRLOUT_NR[0]	= 1
MD13010 $MN_DRIVE_LOGIC_NR[3]	= 4
MD30130 $MA_CTRLOUT_TYPE[0]	= 1

2.3.3 Configuration of drives

SINUMERIK 840D/810D with SIMODRIVE 611 digital

SINUMERIK 840D/810D with 611D drive bus

You can configure the drive in the "Diagnostics" operating area on the operator panel (HMI; Human Machine Interface).

The following machine data are automatically parameterized for each real drive:

- MD13010 $MN_DRIVE_LOGIC_NR[n] (logical drive number)
- MD13000 $MN_DRIVE_IS_ACTIVE[n] (activate SIMODRIVE 611 digital drive)
- MD13030 $MN_DRIVE_MODULE_TYPE[n] (module identifier)
- MD13040 $MN_DRIVE_TYPE[n] (drive type identifier)
- MD13020 $MN_DRIVE_INVERTER_CODE[n] (power section code of drive module)

The index [n] used in the machine data is the slot number of the real drives. The number is automatically assigned by the NC for all connected drive modules on POWER ON. The index is numbered starting with "0" at the beginning of the drive bus (1st real drive available) and continues in ascending order to the end. Multi-axis modules are assigned consecutive, physical drive numbers (counted from left to right). SINUMERIK 810D uses the first 6 slots (indices 0-5).

Machine data:
MD13010 $MN_DRIVE_LOGIC_NR (logical drive number)
can be used to create dummies for modules, which are not yet available.

SINUMERIK 840Di with SIMODRIVE 611 universal

SINUMERIK 840Di with PROFIBUS DP

When a SINUMERIK 840Di is operated with the PROFIBUS DP drive 611 universal, the following MD are **not** used:

- MD13000 $MN_DRIVE_IS_ACTIVE[n] (activate SIMODRIVE 611 digital drive)
- MD13010 $MN_DRIVE_LOGIC_NR[n] (logical drive number)
- MD13020 $MN_DRIVE_INVERTER_CODE[n] (power section code of drive module)
- MD13030 $MN_DRIVE_MODULE_TYPE[n] (module identifier)
- MD13040 $MN_DRIVE_TYPE[n] (drive type identifier)

The following machine data are used instead of MD13000 to MD13040:

- MD13050 $MN_DRIVE_LOGIC_ADDRESS[n] (drive address)
- MD13060 $MN_DRIVE_TELEGRAM_TYPE[n]

  (Message frame type for drives connected to PROFIBUS DP)
- MD13070 $MN_DRIVE_FUNCTION_MASK[n]

  (DP functions used for drives connected to PROFIBUS DP)

  Allows adaptation of certain nonstandardized PROFIBUS control bits of the SIMODRIVE 611 universal.
- MD13080 $MN_DRIVE_TYPE_DP[n] (PROFIBUS DP drive type)

  Selection of external slaves, synchronous, asynchronous or linear drives.

2.3.4 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

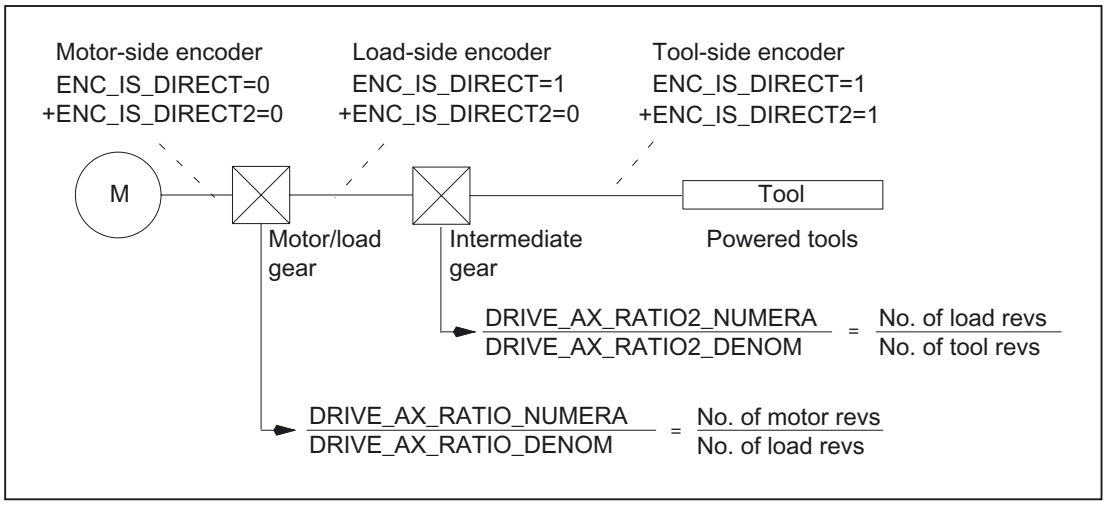


Fig. 2-4 Gear unit types and encoder locations

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 control 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 more information about parameter sets during gear stage change, see:
References:
/FB1/Description of Functions, Basic Machine; Spindles (S1).

Intermediate gear

Additional, configurable load intermediate gears are also supported by the control:

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 parts program or PLC (VDI 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 movement, compensations **cannot** be ruled out due to jumps in the scaling factors. These are not monitored for violation of the maximum acceleration.

Encoder directly on 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 on 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 control partially cancels the status "Axis referenced/synchronized".

If the position reference to the machine, tool, etc., has been lost, it must be restored through appropriate adjustment or referencing. This is especially important for the functions Travel to fixed stop, Referencing to Bero, Cam and Zero marker.



Caution

The control 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

To facilitate re-referencing without a disruptive RESET,
machine data:
MD34080 $MA_REFP_MOVE_DIST
and
MD34090 $MA_REFP_MOVE_DIST_CORR
should be reset to NewConfig efficacy.

For more information about referencing, please see:
References:
/FB1/Description of Functions, Basic Machine; Reference Point Approach (R1)

2.3.5 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

Machine data:
MD32100 $MA_AX_MOTION_DIR (travel direction)
can be used to reverse the direction of movement of the axis,
without affecting the control direction of the position control.

Speed setpoint adjustment

SINUMERIK 840D/810D

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.

SINUMERIK 840D with PROFIBUS DP

It is necessary to change machine data:
MD32250 $MA_RATED_OUTVAL[n] (rated output voltage)
from 80% to 0%, corresponding to value "0", for PROFIBUS DP drives with automatic comparison of speed setpoint scaling.
This value must also be set in the case of adjustment in the NC.
Alternatively for PROFIBUS-DP drives, manual speed setpoint comparison is also possible.

- **Manual comparison**

  A value other than zero is entered in machine data:
  MD32250 $MA_RATED_OUTVAL.

  For more information about speed setpoint adjustment, please refer to:
  References:
  /HBI/SINUMERIK 840Di Manual; "Axes and Spindles".

SINUMERIK 840Di with SIMODRIVE 611 universal

The speed setpoint comparison for SINUMERIK 840Di with SIMODRIVE 611 universal drives can be performed automatically or manually.

- **Automatic adjustment**
  Configuration values for setpoint scaling are adjusted automatically,
  provided that machine data:
  MD32250 $MA_RATED_OUTVAL[n] = 0.
  The speed setpoint comparison through acyclic services at PROFIBUS DP can be performed automatically.

SINUMERIK 840D/810D with SIMODRIVE digital

Note

Velocity adjustment and maximum speed setpoint

The velocity does not need to be adjusted on a SINUMERIK 840D/810D due to automatic speed setpoint comparison.

Maximum speed setpoint

With SINUMERIK 840D/810D, the maximum speed setpoint is the largest value,
which can be output on the SIMODRIVE 611 digital drive, due to the maximum speed set in drive machine data:
MD1401/2401 $MD_MOTOR_MAX_SPEED
(maximum useful motor speed).

MD1401 corresponds to the maximum motor speed on the spindle drive. The desired speed at the spindle is reached via the mechanical gear stage.

The output of the spindle speed is implemented in the NC for SINUMERIK 840D/840Di.
Data for 5 gear stages is implemented in the control.
The gear stages are defined by a minimum and maximum speed for the gear stage and a minimum speed and a maximum speed for the automatic gear stage change. A new set gear stage is output only if the new programmed speed cannot be traversed in the current gear stage.

Machine data:
MD36210 $MA_CTRLOUT_LIMIT[n] (maximum speed setpoint)
is used to limit the speed setpoint to a certain percentage.
Values up to 200% are possible.
An alarm is output if the limit is exceeded.

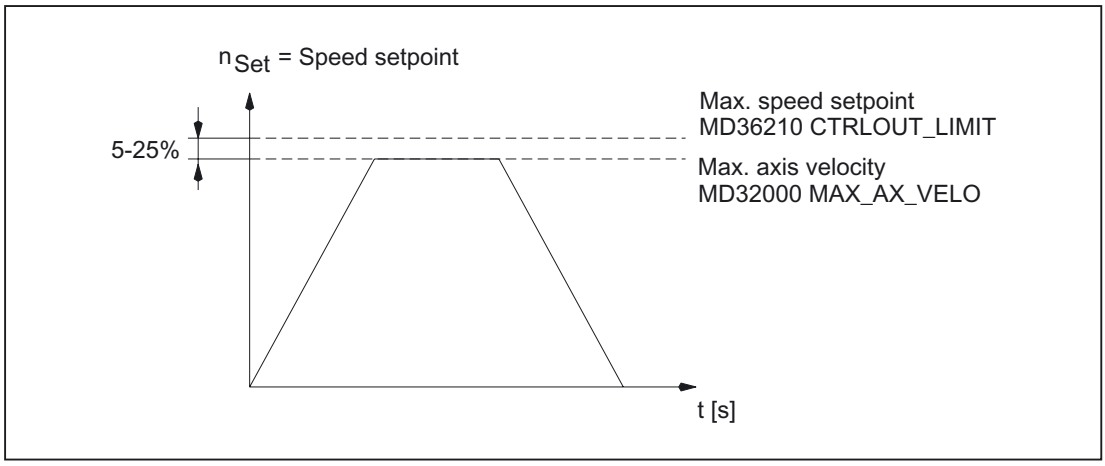


Fig. 2-5 Maximum speed setpoint

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%.

For axes, which reach their maximum velocity at around 80% of the speed setpoint range, the default setting (80%) of machine data:
MD32000 $MA_MAX_AX_VELO (maximum axis velocity)
should be applied.

With SINUMERIK 840D/810, machine data:
MD36210 $MA_CTRLOUT_LIMIT[n] (maximum speed setpoint)
and
MD1405/2405 $MD_MOTOR_SPEED_LIMIT (motor monitoring speed)
must agree.

Note

For more information about setpoint adjustment for SIMODRIVE digital drives, see:
References:
/IAD/ Installation & Startup Guide; "Axes and Spindles".

For more information about setpoint scaling for SIMODRIVE analog drives, see:
References:
/FB3/ Description of Functions, Special Functions; Analog Axis (TE2)

2.3.6 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 axis-specific machine data below are used for this.

The control calculates the actual-value resolution from the settings made in the MD. 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]	Opposite direction	-	-	-	-
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/ rev	Pulses/ rev
MD31030 $MA_LEADSCREW_PITCH	-	mm/rev.	mm/rev.	-	-
MD31050 $MA_DRIVE_AX_RATIO_DENOM[n]	-	Load rev.	-	Load rev.	See note
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[n]	-	Motor rev. if infeed gear available	-	Motor rev.	See note
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.

Note

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 (K_V) factor will not be set.

Load revolutions are entered in machine data:
MD31050 $MA_DRIVE_AX_RATIO_DENOM
and motor revolutions in machine data:
MD31060 $MA_DRIVE_AX_RATIO_NUMERA.

Coding of the machine data

The indices of the following machine data are encoder-coded [encoder no.]:
encoder 0 or 1

Encoder-dependent machine data	Meaning
MD31070 $MA_DRIVE_ENC_RATIO_DENOM[n]	(Measuring gear denominator)
MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[n]	(Measuring gear numerator)
MD31000 $MA_ENC_IS_LINEAR[n]	(Direct measuring system, linear scale)
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)
MD34320 $MA_ENC_INVERS[n]	(Length measurement system is inverse)
Further machine data without index	
MD30200 $MA_NUM_ENCS	(Number of encoders)
MD30300 $MA_IS_ROT_AX	(Rotary axis)
MD31030 $MA_LEADSCREW_PITCH	(Leadscrew pitch)

Index [n] of the following machine data depend on the servo parameter sets of the position controller, with which the actual-value resolution is calculated automatically in the control:

MD DRIVE_AX_...[servo parameter set no.] : 0-5

Parameter-set-dependent machine data	Meaning
MD31050 $MA_DRIVE_AX_RATIO_DENOM[n]	(Denominator load gear unit)
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[n]	(Numerator load gear unit)

For the following machine data, the control does not consider any parameter set nor any indices for coded encoders.

NewConfig-dependent machine data	Meaning
MD31064 $MA_DRIVE_AX_RATIO2_DENOM	(Intermediate gear denominator)
MD31066 $MA_DRIVE_AX_RATIO2_NUMERA	(Intermediate gear numerator)
MD31044 $MA_ENC_IS_DIRECT2	(Encoder on intermediate gear)
MD32000 $MA_MAX_AX_VELO	(Maximum axis velocity)
MD34080 $MA_REFP_MOVE_DIST	(Reference point distance)
MD34090 $MA_REFP_MOVE_DIST_CORR	(Reference point offset)

Note

These machine data can be activated in parts programs with the command `NEWCONF` or via the HMI operator panel using a soft key.

Variants of actual-value acquisition

The relevant machine data and relational calculations for the different methods of actual-value acquisition are described in the following.

2.3.7 Adjustments to actual-value resolution

Calculating the ratio

The calculation of the ratio is obtained from the associated machine data and is defined for incremental encoders as follows:

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} = \frac{\text{Internal increments/(mm)}}{\text{Encoder increments/(mm)}}$$

For incremental encoders with rotary axis, the following applies:

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} = \frac{\text{Internal increments/(degrees)}}{\text{Encoder increments/(degrees)}}$$

The internal pulse multiplication factor provided by the measuring system logic module is

- 2048

For raw signal generators on 840D with SIMODRIVE 611 digital

- 128

  For raw signal generators on 810D

Linear axis with linear scale

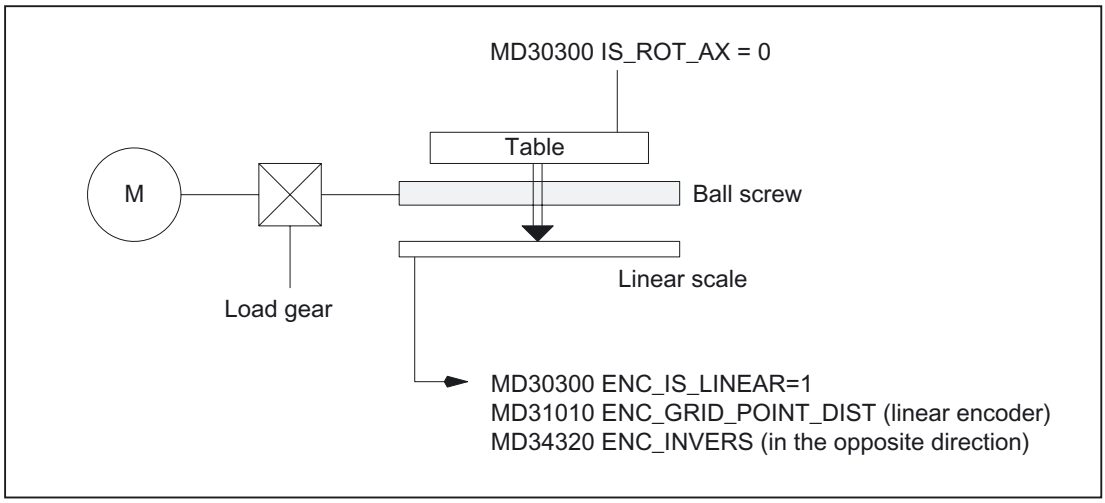


Fig. 2-6 Linear axis with linear scale

In order to adapt the actual-value resolution to the calculation resolution, the control calculates the quotients from the "internal increments/mm" and the "encoder increments/mm" as follows:

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{\text{ENC_GRID_POINT_DIST [n] * INT_INCR_PER_MM}}{\text{Internal multiplication}}$$

The distance for linear encoders is based on the pulse increments.

Linear axis with rotary encoder on motor

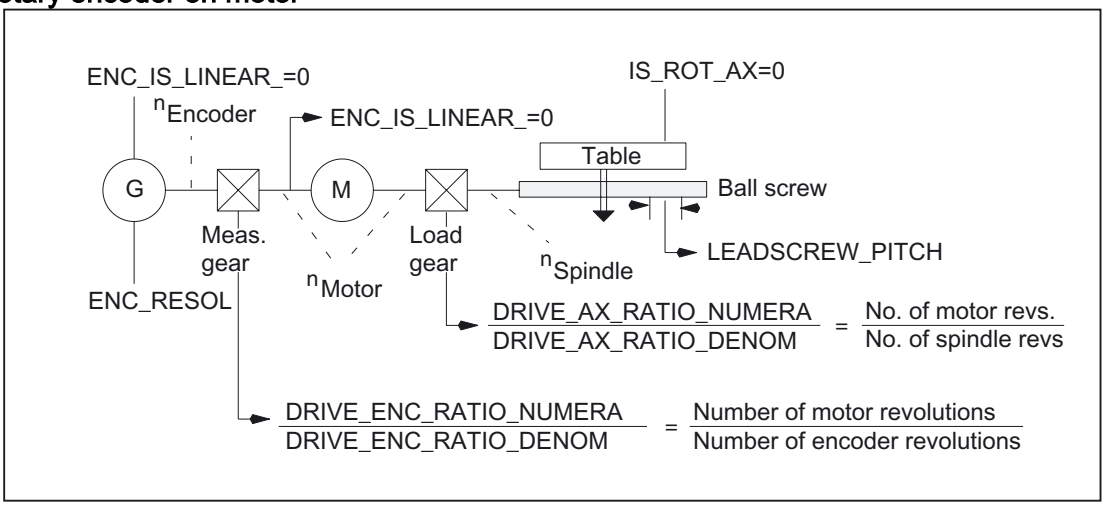


Fig. 2-7 Linear axis with rotary encoder on motor

In order to adapt the actual-value resolution to the calculation resolution, the control calculates the quotients from the "internal increments/mm" and the "encoder increments/mm" as follows:

SINUMERIK example

Linear axis with rotary encoder (2048 pulses) **on motor**;
internal multiplication (2048)

Gear: Motor/leadscrew 5
Pitch 10 mm
10000 increments per mm

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{\text{ENC_RESOL [n]} * \text{internal multiplication}} * \frac{\text{DRIVE_ENC_RATIO_NUMERA [n]}}{\text{DRIVE_ENC_RATIO_DENOM [n]}} * \frac{\text{DRIVE_AX_RATIO_DENOM [n]}}{\text{DRIVE_AX_RATIO_NUMERA [n]}} * \text{LEADSCREW_PITCH} * \text{INT_INCR_PER_MM}$$

⇒	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
	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
	MD10200 $MN_INT_INCR_PER_MM	= 10000

$$\Rightarrow \frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{2048 * 2048} * \frac{1}{1} * \frac{1}{5}$$

$$* 10\ \text{mm} * 10000\ \text{incr./mm} = 0.004768$$

Result:

1 encoder increment corresponds to 0.004768 increments of the internal unit. In practice, the available encoder resolution should not be resolved more accurately than the internal computational resolution.

Linear axis with rotary encoder on the machine

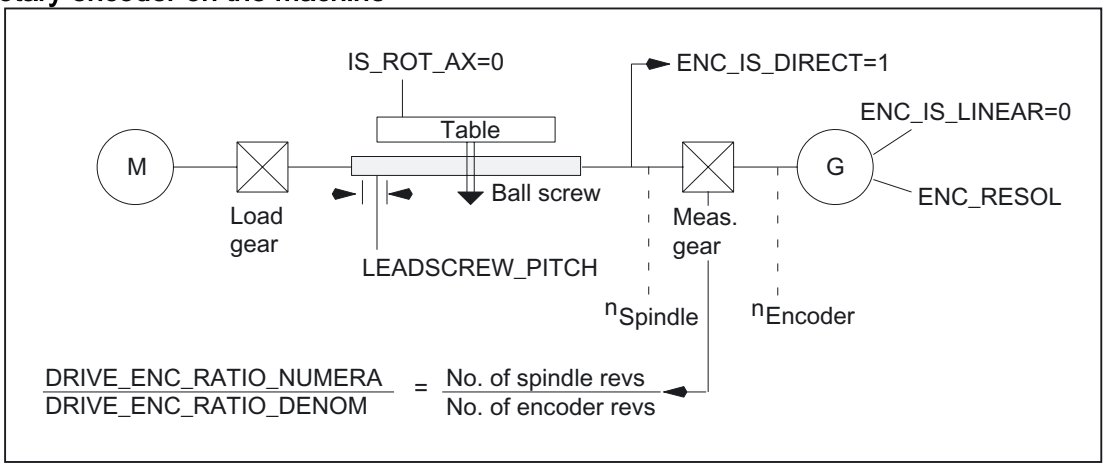


Fig. 2-8 Linear axis with rotary encoder on the machine

In order to adapt the actual-value resolution to the calculation resolution, the control calculates the quotients from the "internal increments/mm" and the "encoder increments/mm" as follows:

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{\text{ENC_RESOL [n] * internal multiplication}} * \frac{\text{DRIVE_ENC_RATIO_NUMERA [n]}}{\text{DRIVE_ENC_RATIO_DENOM [n]}} * \text{LEADSCREW_PITCH} * \text{INT_INCR_PER_MM}$$

Rotary axis with rotary encoder on motor

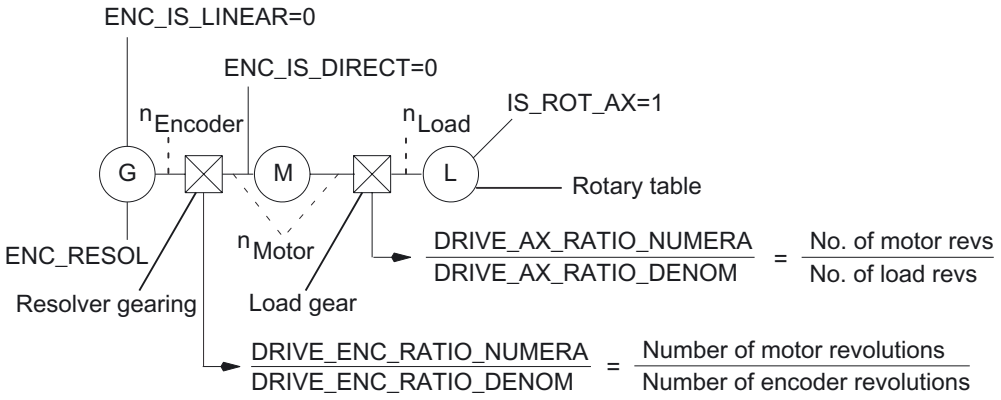


Fig. 2-9 Rotary axis with rotary encoder on motor

In order to adapt the actual-value resolution to the calculation resolution, the control calculates the quotients from the "internal increments/degree" and the "encoder increments/degree" as follows:

Example for rotary axis with encoder on motor

Rotary axis with rotary encoder (2048 pulses) **on motor**;
internal multiplication (2048)

Gear: Motor/rotary axis 5
1000 increments per degree

$$\frac{\text{Internal increments/degrees}}{\text{Encoder increments/degrees}} = \frac{\text{360 degrees}}{\text{ENC_RESOL [n] * internal multiplication}} * \frac{\text{DRIVE_ENC_RATIO_NUMERA [n]}}{\text{DRIVE_ENC_RATIO_DENOM [n]}} * \frac{\text{DRIVE_AX_RATIO_DENOM [n]}}{\text{DRIVE_AX_RATIO_NUMERA [n]}} * \text{INT_INCR_PER_DEG}$$

⇒ 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
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

$$\Rightarrow \frac{\text{Internal increments/degrees}}{\text{Encoder increments/degrees}} = \frac{\text{360 degrees}}{\text{2048 * 2048}} * \frac{1}{1} * \frac{1}{5}$$

* 1000 incr./degrees = 0.017166

Result:

1 encoder increment corresponds to 0.017166 increments of the internal unit.
The encoder resolution is thus coarser than the computational resolution by a factor of 58.

Rotary axis with rotary encoder on the machine

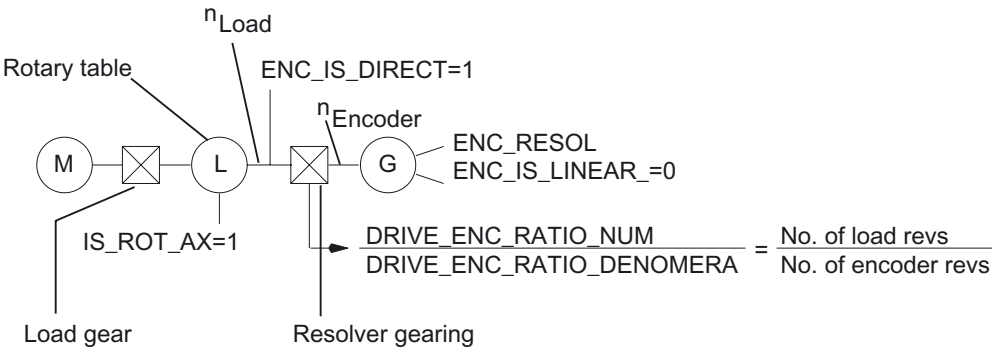


Fig. 2-10 Rotary axis with rotary encoder on the machine

In order to adapt the actual-value resolution to the calculation resolution, the control calculates the quotients from the "internal increments/degree" and the "encoder increments/degree" as follows:

$$\frac{\text{Internal increments/degrees}}{\text{Encoder increments/degrees}} = \frac{360\ \text{degrees}}{\text{ENC_RESOL [n]} * \text{internal multiplication}} * \frac{\text{DRIVE_ENC_RATIO_NUMERA [n]}}{\text{DRIVE_ENC_RATIO_DENOM [n]}} * \text{INT_INCR_PER_DEG}$$

Intermediate gear encoder on tool

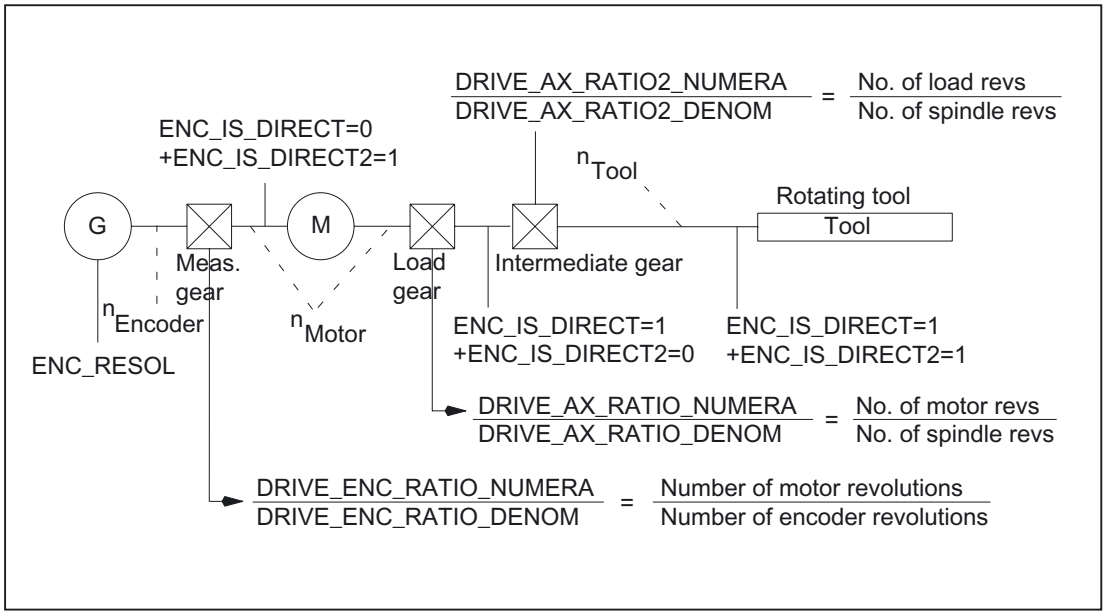


Fig. 2-11 Intermediate gear with encoder directly on the rotating tool

In order to adapt the actual-value resolution to the calculation resolution, the control calculates the quotients from the "internal increments/mm" and the "encoder increments/mm" as follows:

$$\frac{\text{Internal increments/degrees}}{\text{Encoder increments/degrees}} = \frac{360\ \text{degrees}}{\text{ENC_RESOL [n]} * \text{internal multiplication}} * \frac{\text{DRIVE_ENC_RATIO_NUMERA [n]}}{\text{DRIVE_ENC_RATIO_DENOM [n]}} * \text{INT_INCR_PER_DEG}$$

2.4 Closed-loop control

2.4.1 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 speed and current control systems for SIMODRIVE 611 are described in:
References:
/IAD/ "Installation & Startup Guide" SINUMERIK 840D/611D digital
/IAC/ "Installation & Startup Guide" SINUMERIK 810D
/PJU/ "Planning Guide" Converters

The basic structure of an axis/spindle position control is illustrated below:

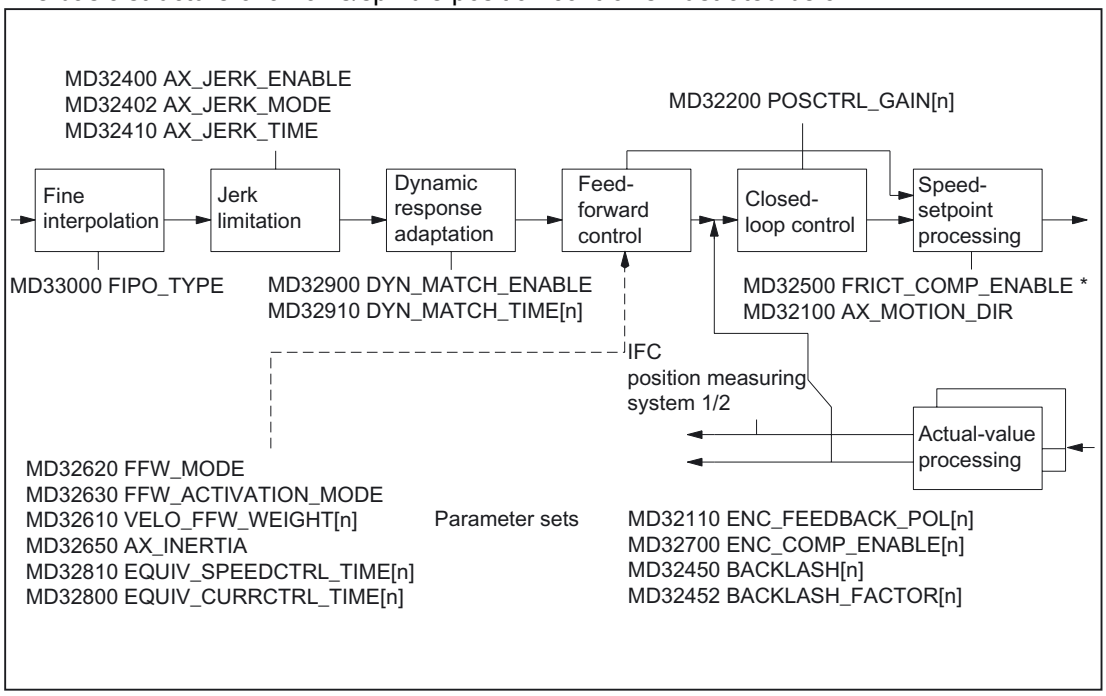


Fig. 2-12 Additional servo parameter sets for position control

For a description of the feedforward control, backlash, friction compensation with further machine data, and leadscrew error compensation, see:
References:
/FB2/Description of Functions, Extended Functions; Compensations (K3).

For a description of jerk limitation, see:
References:
/FB1/Description of Functions, Basic Machine; Acceleration (B2).

Fine Interpolation

The fine interpolator (FIPO) is used to adjust the setpoint of the (generally lower) interpolator cycle clock to the later position-control cycle.

Fine interpolation further improves the quality of the contour (decreasing the step effect of the speed setpoint).

There are three types of FIPOs:

1:	Differential FIPO
2:	Cubic FIPO
3:	Cubic FIPO, optimized for operation with feedforward control

The type of fine interpolation can be defined via machine data:
MD33000 $MA_FIPO_TYPE (fine interpolation type).

A differential FIPO not only performs cycle matching but also calculates a mean value (smoothing) from an IPO cycle.

The cubic FIPO, type 3, supplies the best contour accuracy in addition to the cycle adaptation.

K_v factor

To ensure that only low contour deviations occur in continuous-path mode, a high servo gain factor (K_v):
MD32200 $MA_POSCTRL_GAIN[n]
is required.

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

The servo gain factor (K_v) is defined as follows:

$$KV = \frac{\text{Velocity}}{\text{Following error}} \; ; \; \frac{[\text{m/min}]}{[\text{mm}]}$$

[m/min]/[mm]: Unit of servo gain factor (K_V) according to VDI standard

Servo gain factor (K_{v}) setting for SINUMERIK 840D/810D

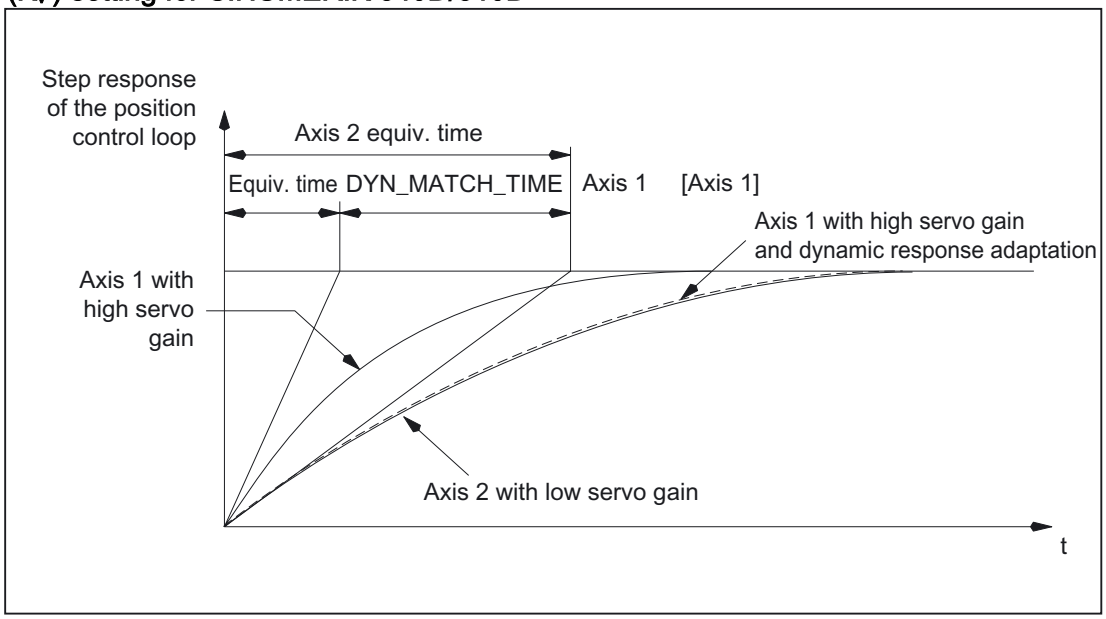


Fig. 2-13 Dynamic response adaptation

Dynamic response adaptation

The purpose of dynamic response adaptation is to set an identical following error for axes with different servo gain factors (K_v). The optimum contour accuracy for axes interpolating with each other can thus be achieved without reduced control quality. A high servo gain factor (K_v) for an axis can be maintained, which guarantees optimum disturbance suppression of the axis.

The function is activated via machine data:
MD32900 $MA_DYN_MATCH_ENABLE (dynamic response adaptation).

Axes are adapted via machine data:
MD32910 $MA_DYN_MATCH_TIME[n] (dynamic response adaptation time constant).
With this MD, the equivalent time constant of the position control loop of the axes with a higher servo gain factor (K_v) is matched to the axis with the lowest servo gain factor (K_v).

The difference between the equivalent time constants of the "slowest" control loop and the relevant axis is entered in machine data:
MD32910 $MA_DYN_MATCH_TIME[n].

Example of dynamic response adaptation of axes 1, 2 and 3 (without speed feedforward control):

The equivalent time constant of the position control loop is as follows:	
For axis 1:	30 ms
For axis 2:	20 ms
For axis 3:	24 ms

⇒ Axis 1 is dynamically the slowest axis

⇒ For machine data:
MD32910 $MA_DYN_MATCH_TIME[n] (dynamic response adaptation time constant)
the following values are achieved:

Axis 1:	0 ms
Axis 2:	10 ms
Axis 3:	6 ms

Approximation formulae

The equivalent time constant of the position control loop of an axis is calculated according to the following formula:

- No feedforward control is active

$$T_{Spares} \approx \frac{1}{\text{MD32200 POSCTRL_GAIN [1/s]}}$$

- Speed feedforward control

$$T_{Spares} \approx \text{MD32810 EQUIV_SPEEDCTRL_TIME}$$

- Torque feedforward control

$$T_{Spares} \approx \text{MD32800 EQUIV_CURRCTRL_TIME}$$

Note

If a geometry axis is subject to dynamic response adaptation, the same dynamic behavior is also required of all other geometry axes and can be activated with the setting:
MD32900 $MA_DYN_MATCH_ENABLE= 1.

References:
/IAD/ "Installation & Startup Guide", SINUMERIK 840D/611D
/IAC/ "Installation & Startup Guide", SINUMERIK 810D

2.4.2 Parameter sets of the position controller

Six different parameters sets

The position control can operate with 6 different servo parameter sets.

They are used as follows

1. Fast adaptation of the position control to altered machine characteristics during operation, e.g. a gear change of the spindle.
2. Matching the dynamics of an axis to another axis, e.g. during tapping.

Parameter set changeover

The following machine data can be changed by switching over the parameter set during operation:	
Denominator load gearbox	MD31050 $MA_DRIVE_AX_RATIO_DENOM[n]
Numerator load gearbox	MD31060 $MA_DRIVE_AX_RATIO_NUMERA[n]
K_v factor	MD32200 $MA_POSCTRL_GAIN[n]
Backlash compensation	MD32452 $MA_BACKLASH_FACTOR[n]
Feedforward control factor	MD32610 $MA_VELO_FFW_WEIGHT[n]
Exact stop limits	MD36012 $MA_STOP_LIMIT_FACTOR[n]
	MD36000 $MA_STOP_LIMIT_COARSE
	MD36010 $MA_STOP_LIMIT_FINE
and zero-speed window	MD36030 $MA_STANSTILL_POS_TOL
Equivalent time constant current control loop for torque feedforward control	MD32800 $MA_EQUIV_CURRCTRL_TIME[n]
Equivalent time constant speed control loop for speed feedforward control	MD32810 $MA_EQUIV_SPEEDCTRL_TIME[n]
Time constant for dynamic response adaptation	MD32910 $MA_DYN_MATCH_TIME[n]
Threshold value for velocity monitoring	MD36200 $MA_AX_VELO_LIMIT[n]

Tapping or thread cutting

The following applies to **parameter sets for axes**:

- For machine axes **not** involved in tapping or thread cutting, parameter set 1 (index=0) is always used.

  The further parameter sets need not be considered.

- For machine axes involved in tapping or thread cutting, the same parameter set number as for the current gear stage of the spindle is activated.

  All parameter sets correspond to the gear stages and must therefore be parameterized.

The current parameter set is displayed in operating area "Diagnostics" in the "Service axis" display. The parameter sets for gear stages 1 to 5 are referred to as interpolation parameters.

Parameter sets during gear stage change

Interpolation parameter sets during gear stage change:

In the case of spindles, each gear stage is assigned its own parameter set.
The appropriate parameter set is activated, depending on the NC/PLC interface signal: DB31, ... DBX16.0 - 16.2 (actual gear stage).

DB31, ... DBX16.0 - 16.2 (actual gear stage)		Active parameter set	
000	1st gear stage	2	(Index=1)
001	1st gear stage	2	(Index=1)
010	2nd gear stage	3	(Index=2)
011	3rd gear stage	4	(Index=3)
100	4th gear stage	5	(Index=4)
101 110 111	5th gear stage	6	(Index=5)

For more information about gear stages for spindles, see:
References:
/FB1/ Description of Functions, Basic Machine; Spindles (S1).

2.4.3 Extending the parameter set

Application

Some machines use the same drive for moving various machine parts, which, in view of considerably varying speeds, results in a gear stage change. With each gear stage change, the corresponding parameter set is also switched over.

Now several parameter sets are provided for further practical applications and for setting the feedforward control of the control loop.

Functionality

To optimize closed-loop control during startup, these codable parameter sets support practiceoriented startup by substantially reducing configuration expenditure with regard to the new functions, such as backlash compensation, feedforward control factor, exact stop limits, and zero-speed window.

New parameter sets

Until now it has been possible to change the gear ratio and other control loop parameters (such as achievable control servo gain) by switching over the servo parameter set. The already existing machine data with parameter set coding are extended as follows:

Parameter set coding (modifiable as a function of parameter set)	
Backlash compensation	MD32450 $MA_BACKLASH[n] (max. number of encoders)
Feedforward control factor	MD32610 $MA_VELO_FFW_WEIGHT[n]
Exact stop limits	MD36000 $MA_STOP_LIMIT_COARSE MD36010 $MA_STOP_LIMIT_FINE
Zero-speed window	MD36030 $MA_STANSTILL_POS_TOL

Weighting factor for parameter set changeover	
Weighting factor for:	**MD32452 $MA_BACKLASH_FACTOR[n]**
• Backlash compensation	MD32450 $MA_BACKLASH[n]
Weighting factor for:	**MD36012 $MA_STOP_LIMIT_FACTOR[n]**
• Exact stop limits	MD36000 $MA_STOP_LIMIT_COARSE MD36010 $MA_STOP_LIMIT_FINE
• Zero-speed window	MD36030 $MA_STANSTILL_POS_TOL

Machine data tried and tested to date

Further machine data with parameter set coding

The following existing machine data can be coded using parameter sets and have already been tried and tested during the startup of the NC:

Denominator load gearbox	MD31050 $MA_DRIVE_AX_RATIO_DENOM
Numerator load gearbox	MD31060 $MA_DRIVE_AX_RATIO_NUMERO
Equivalent time constant current control loop	MD32800 $MA_EQUIV_CURRCTRL_TIME
Equivalent time constant speed control loop	MD32810 $MA_EQUIV_SPEEDCTRL_TIME
Time constant for dynamic response adaptation	MD32910 $MA_DYN_MATCH_TIME
Threshold value for velocity monitoring	MD36200 $MA_AX_VELO_LIMIT

Advantages

- The indirect switchover **of a single** function: (e.g. MD32452 not 1)

  The parameter set-dependent function is only relevant if required.

- The indirect switchover of **several** functions: (e.g. MD36012 not 1)

  The common weighting factor forces the ratios of all parameter set-dependent functions within various machine data to remain constant. To switch over several functions, one single machine data can be sufficient.

In this way, configuration errors are avoided and input work reduced.

Activating the parameter set coding

Default setting without parameter set coding

Provided that the machine data below retain a value of (1), the control is compatible with earlier software versions:

MD32452 $MA_BACKLASH_FACTOR = 1

MD32610 $MA_VELO_FFW_WEIGHT = 1

MD36012 $MA_STOP_LIMIT_FACTOR = 1

Activating the parameter set coding

If the default setting in machine data:
MD32452 $MA_BACKLASH_FACTOR
is changed to something other than (1.0), the backlash compensation is also changed, depending on the parameter set.

If the default setting in machine data:
MD32610 $MA_VELO_FFW_WEIGHT
is changed to something other than (1.0), the feedforward control factor is also changed, depending on the parameter set.

If the default setting in machine data:
MD36012 $MA_STOP_LIMIT_FACTOR
is changed to something other than (1.0), the exact limit stops and the zero-speed window are also changed, depending on the parameter set.

When loading previous archives (from data stored in earlier software versions), machine data:
MD32610 $MA_VELO_FFW_WEIGHT
is automatically set to the same value for all indices, to ensure that the control remains compatible.

Machine data:
MD32452 $MA_BACKLASH_FACTOR
and
MD36012 $MA_STOP_LIMIT_FACTOR
are not even available in previous archives.
In this case, the default setting remains active automatically.
The response is also compatible in this case.

Example

Effects of various parameter sets with backlash compensation:

MD32450 $MA_BACKLASH[AX1]	= 0.01	
MD32452 $MA_BACKLASH_FACTOR[0,AX1]	= 1.0	Parameter set 1
MD32452 $MA_BACKLASH_FACTOR[1,AX1]	= 2.0	Parameter set 2
MD32452 $MA_BACKLASH_FACTOR[2,AX1]	= 3.0	Parameter set 3
MD32452 $MA_BACKLASH_FACTOR[3,AX1]	= 4.0	Parameter set 4
MD32452 $MA_BACKLASH_FACTOR[4,AX1]	= 5.0	Parameter set 5
MD32452 $MA_BACKLASH_FACTOR[5,AX1]	= 6.0	Parameter set 6

In parameter set 1 (index 0) of the first axis (AX1), a backlash compensation factor with the value 1.0 has the following effect:

1.0 * MD32450 = 0.01 mm (or inch or degrees)

The backlash compensation is twice as large in parameter set 2, three times as large in parameter set 3, etc.

The maximum value is 100.

Supplementary conditions

The functional expansion is available for all control variants.

Control response on POWER ON, RESET, REPOS

The new or modified data are activated via the "Enable machine data" soft key or by a RESET or POWER ON. Mode switchover, block search or REP0S have no influence.

2.5 Optimization of the control

2.5.1 Position controller: injection of positional deviation

Application

The stability and positioning response of axes with a low natural frequency (up to approx. 20 Hz) and a mechanical design capable of generating oscillations is improved by active oscillation damping with simultaneous use of the feedforward control.

To illustrate this, the difference in position between two measuring systems is generated and injected as an additional current setpoint for the feedforward control, according to the weighting of machine data:
MD32950 $MA_POSCTRL_DAMPING.

The function is used predominantly for axes with strong tendency to vibrate.

Functionality

The positional deviation is injected in the NC in the position controller cycle of the higher-level position control loop in the NC.

The difference in position between a direct and an indirect measuring system is generated and injected as an additional current setpoint, depending on the weighting of machine data:
MD32950 $MA_POSCTRL_DAMPING.

Direct measuring system:

MD31040 $MA_ENC_IS_DIRECT[1]=1

The encoder for position actual-value acquisition is connected directly to the machining process of the machine (load encoder).

Indirect measuring system:

MD31040 $MA_ENC_IS_DIRECT[0]=0

The encoder for position actual-value acquisition is located on the motor (motor encoder).

MD32950

The function is activated via the following machine data setting:
MD32950 $MA_POSCTRL_DAMPING = 1.

It is possible to enter both positive and negative values,
which will then serve to scale the injection of the positional deviation.

Standard setting:
MD32950 $MA_POSCTRL_DAMPING = 0.
In this case, the injection of positional deviation is inactive.

Note

The weighting of machine data:
MD32950 $MA_POSCTRL_DAMPING
can be set on the basis of step responses, for example.

If the control approaches the stability limit (vibration inclination increases), then the parameter is too large.

Supplementary condition

1. The functional expansion is available for all control variants, which use SIMODRIVE 611 digital drives.
2. The function can only be used on axes with two encoders:
   MD30200 $MA_NUM_ENCS = 2

   One of the encoders **must** be parameterized as an indirect measuring system and the other as a direct measuring system.

If these conditions are not met, reset alarm 26016 will acknowledge the function when an attempt is made to activate machine data:
MD32950 $MA_POSCTRL_DAMPING.
The function remains deactivated internally when this alarm is active.

2.5.2 Position controller position setpoint filter: new balancing filter

For speed and torque feedforward control

Application

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 feedforward control the speed setpoint at 100%, without resulting in overshoots when positioning.

Functionality

Until now, it was possible to compensate the negative effects of the balancing filter (a low pass with settable time constant), which has remained unchanged since SW 1 and which has admitted instead of an overshoot also undesired undershoots of some 10 micrometers, as follows:

- Setting a compromise of overshoots and undershoots
- Setting the positioning slightly overshooting and reducing the amplitude of these overshoots by way of position setpoint signal smoothing and jerk limitation
- Setting a speed controller with reference model instead of PI behavior (only possible with 840D)
- Setting the filter time to zero and setting the feedforward control factor to a value less than 100%
- Sacrificing the feedforward control and bringing the machine to a very high position controller gain using dynamic stiffness control.
  This measure requires a stiff machine.

With SW 5 and higher, a second, improved balancing filter is therefore available.
The existing filter is still installed and is active with the same function when transferring existing archives (e.g. in the case of upgrades).

Advantages

The new balancing filter provides the following improvements:

- An axis with feedforward control has a considerably lower inclination to undershoots when positioning.
- Achieves a higher accuracy at bent contours (can be measured, e.g. using the circularity test) and can be set more easily.
- A part of the setting is carried out by the control system automatically.

Filter activation with MD32620

The new filter is activated by changing the axial machine data:
MD32620 $MA_FFW_MODE
by selecting values 3 and 4.

The desired active feedforward control variant with new balancing
is selected as follows via MD32620:

3 =	Speed feedforward control with new balancing
4 =	Torque feedforward control (only possible with SINUMERIK 840D) with new balancing

For reasons of a compatible response of archives that contain only changes compared with the default settings, no default value can be set for these new values 3 and 4 in the new software versions.

Activation of feedforward control

Parts programs can be used to activate and deactivate the feedforward control for all axes, using instructions `FFWON` and `FFWOF`, which does not affect machine data:
MD32630 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 are read in anew (see the appropriate values of the machine data). Mode change, block search and repositioning have **no** influence on the feedforward control.

Supplementary conditions

The functional expansion is available for all control variants.

New setting rule for MD32810 and MD32800

If the new filter is active, the setting rule for machine data:
MD32810 $MA_EQUIV_SPEEDCTRL_TIME
and
MD32800 $MA_EQUIV_CURRCTRL_TIME are modified.

This means that, if the old balancing filter had previously been active and is to be changed to the new filter, the following actions must be considered:

Setting the equivalent time constant with speed feedforward control

If the previous setting was MD32620 $MA_FFW_MODE = 1:

1. Set MD32620 $MA_FFW_MODE = 3.
2. Set MD32610 $MA_VELO_FFW_WEIGHT = 1.
3. Reset MD32810 $MA_EQUIV_SPEEDCTRL_TIME.

Setting the equivalent time constant with torque feedforward control

If the previous setting was MD32620 $MA_FFW_MODE = 2:

1. Set MD32620 $MA_FFW_MODE = 4.
2. Set MD32610 $MA_VELO_FFW_WEIGHT = 1.
3. Reset MD32800 $MA_EQUIV_CURRCTRL_TIME.

This also applies when you load an older **archive** into the control system, e.g. from a previous version. For example, when changing MD32620 $MA_FFW_MODE from 1 to 3, MD32810 $MA_EQUIV_SPEEDCTRL_TIME may **not** simply retain its old value, but must be **reset**. Otherwise, no improvement will be achieved, but rather a worsening of the positioning response.

Recommended setting in case of recommissioning

If recommissioning, or if previous standard values are loaded (switch position 1 on commissioning switch and POWER ON), the following machine data default values apply:

MD32620 $MA_FFW_MODE = 1

MD32610 $MA_VELO_FFW_WEIGHT = 1

These are **not** the recommended settings, but are chosen for reasons of compatibility.

However, if recommissioning, the following setting is recommended:
MD32620 $MA_FW_MODE = 3

The balancing time for the speed feedforward control then just has to be adjusted in 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 servo trace (HMI Advanced or programming device). 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 proportional-plus-integral-action controller with special value smoothing, an approximate equivalent time can be read from drive machine data 1500-1503.
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 initial value (e.g. 1.5 ms) is then 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:	**Magnify** MD32810 $MA_EQUIV_SPEEDCTRL_TIME.
Excessively slow approach monitored:	Reduce MD32810 $MA_EQUIV_SPEEDCTRL_TIME.

Magnifying MD32810

Magnifying the value in machine data:
MD32810 $MA_EQUIV_SPEEDCTRL_TIME
slows the axis down and increases the geometric contour error on curves.

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 in machine data:
MD32810 $MA_EQUIV_SPEEDCTRL_TIME
speeds the axis up.
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 must have **identical settings** in the following data:

MD32200 $MA_POSCTRL_GAIN

MD32620 $MA_FFW_MODE

MD32610 $MA_VELO_FFW_WEIGHT

MD32810 $MA_EQUIV_SPEEDCTRL_TIME (or MD32800 $MA_EQUIV_CURRCTRL_TIME)

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.
If identical values are not possible for the above data,
machine data:
MD32910 $MA_DYN_MATCH_TIME
can be used to make an adjustment.
This allows the same servo gain value (K_V) to be displayed;
however, this rarely occurs.

Different servo gain 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.
- The jerk filter setting data is incorrect at some point.

Automatic switchover when changing the position-control cycle

Previously, if the position-control cycle (MD10050 $MN_SYSCLOCK_CYCLE_TIME) changed or the acceptance time of the speed setpoints was modified in order to increase the servo gain (K_V) (MD10082 $MN_CTRLOUT_LEAD_TIME), or dynamic stiffness control was enabled (MD32640 $MA_STIFFNESS_CONTROL_ENABLE), the adjustment of MD32810 $MA_EQUIV_SPEEDCTRL_TIME had to be repeated,
as the optimum value changed significantly.

With setting:
MD32620 $MA_FFW_MODE = 3 or 4,
the control takes these changes into account automatically,
so that machine data:
MD32810 $MA_EQUIV_SPEEDCTRL_TIME
no longer has to be reset.

After major changes, however, you should nevertheless check the positioning behavior (via servo trace).

Example of speed feedforward control

Programming sample of a selection of the speed feedforward control with new balancing and default setting: FFWON and FFWOF are active.

```
MD32620 FFW_MODE[X1] = 3                    ;  New mode for speed feedforward
                                               control
MD32630 FFW_ACTIVATION_MODE[X1] = 1         ;  FFWON and FFWOF are active in NC
                                               program.
```

FFWON now enables the speed feedforward control in the program
(with all axes of the channel with the same settings as X1);
FFWOF will disable them again.

Machine data:
MD20150 $MC_GCODE_RESET_VALUES[23] (G group initial setting)
can be used to set a default value for FFWON for every channel.

```
MD32620 $MA_FFW_MODE[X1] = 3                    ; New mode for speed feedforward
                                                  control
MD32630 $MA_FFW_ACTIVATION_MODE[X1] = 0         ; FFWON and FFWOF are active in NC
                                                  program.
```

In this case, the speed feedforward control with X1 is enabled continuously, also in JOG mode. MD20150 $MC_GCODE_RESET_VALUES[ ], FFWON and FFWOF have no effect on X1. This can be useful if the machine is only permitted to run with feedforward control, e.g. for reasons of accuracy, or if you also want to test the feedforward control without a program during startup.

Note

The setting of the feedforward control must be the same for all axes of an interpolation group.

Setting the equivalent time constant of the current control loop

MD32800 torque feedforward control for each additional option

The same rules and recommendations apply to setting the time constant of the current control loop as to the speed feedforward control.

However, as previously, activation of the torque feedforward control filter via:
MD32620 $MA_FFW_MODE = 4
must be enabled both in the drive and **via the option** in order to set the time constant via:
MD32800 $MA_EQUIV_CURRCTRL_TIME.

As with activating the previous filter via:
MD32620 $MA_FFW_MODE = 2,
all other machine data are set in consideration of the corresponding elasticity limits of the machine.

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 trialing the use of dynamic stiffness control as an alternative:
MD32640 $MA_STIFFNESS_CONTROL_ENABLE.

Example of torque feedforward control

Programming sample of a selection of the torque feedforward control with new balancing and default setting: FFWON and FFWOF are active.

```
MD32620 $MA_FFW_MODE[X1] = 4                  ; New mode for torque feedforward
                                                control
MD32630 $MA_FFW_ACTIVATION_MODE[X1] = 1       ; FFWON and FFWOF are active.
```

FFWON now enables the torque feedforward control in the program
(with all axes of the channel with the same settings as X1);
FFWOF will disable them again.

Machine data:
MD20150 $MC_GCODE_RESET_VALUES[23] (G group initial setting)
can be used to set a default value for FFWON for every program.

```
MD32620 $MA_FFW_MODE[X1] = 4                  ; New mode for torque feedforward
                                                control
MD32630 $MA_FFW_ACTIVATION_MODE[X1] = 0       ; FFWON and FFWOF are active.
```

In this case, the torque feedforward control with X1 is enabled continuously, also in JOG mode. MD20150 $MC_GCODE_RESET_VALUES[ ], FFWON and FFWOF have no effect on X1. This can be practical if the machine is only permitted to run with feedforward control, e.g. for reasons of accuracy, or if you also want to test the feedforward control without a program during startup.

For more information about the effect of the feedforward control relating to the speed and torque position controller setpoints, please refer to:
References:
/FB2/Description of Functions, Extended Functions; Compensations (K3),
Chapter: "Description of machine data"

Note

The setting of the feedforward control must be the same for all axes of an interpolation group.

2.5.3 Position controller position setpoint filter: new 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 traveled in both forward and reverse, 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.

Advantages

The filter available since software version 1, which is set via:
MD32400 $MA_AX_JERK_ENABLE = 1
and
MD32410 $MA_AX_JERK_TIME = filter time,
meets the requirements for a strong filter effect with smoothing behavior that is as symmetrical as possible for small time constants of up to approximately 10 ms.

A considerably better result is achieved using this new filter type for axial jerk limitation with floating averaging. In this case, filter time constants of approx. 20 to 40 ms can be used, depending on the particular machine.

Thanks to the floating averaging, the new filter achieves lower contour errors with the same smoothing effect, compared with the previous 2nd degree filter type. The contour of the smoothing behavior is to a large degree balanced.

Filter enable with MD32402

Machine data:
MD32402 $MA_AX_JERK_ENABLE
is used to enable the new position setpoint filter,
and is defined as follows:

MD32402 $MA_AX_JERK_MODE	**= 2**	**Select new jerk filter mode**
MD32410 $MA_AX_JERK_TIME	= 0.02	Set filter time in seconds (20 ms)
MD32400 $MA_AX_JERK_ENABLE	= 1	Enable filter calculation

If no filter mode was previously activated:
MD32402 $MA_AX_JERK_MODE = 2,
POWER ON has to be triggered once.
Otherwise, "Enable machine data" or RESET on the machine control panel are sufficient.

For reasons of compatibility, the default setting is:
MD32402 $MA_AX_JERK_MODE = 1.

Note

For new machines, the new filter is generally recommended:
MD32402 $MA_AX_JERK_MODE = 2.

Only if very high filter times are needed and contour accuracy plays a minor part (e.g. as sometimes occurs when positioning axes), can the old filter be more advantageous.

Fine adjustment

The fine adjustment of the position setpoint 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 position setpoint 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 position setpoint filter is available in all control system variants 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 more information about jerk limitation on an interpolator level, please refer to:
References:
/FB1/ Description of Functions, Basic Machine; Acceleration (B2)
Chapter: "Axis-oriented jerk limitation"
Chapter: "Axisspecific machine data"

2.5.4 Position control with proportional-plus-integral-action controller

Function

For standard applications, the core of the position controller is a proportional-action controller with the manipulation options described above connected in series upstream.
It is possible to connect an I component for special applications (e.g. 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. Expert knowledge of control engineering and measurements taken using servo trace are absolutely essential when using this function. If incorrect settings are made for the relevant machine data, there is a risk of damaging the machine, due to possible instability.

Background

The transfer function of the proportional-plus-integral-action controller is as follows:

$$G_R(s) = K_R + \frac{1}{T_n} * s$$

K_R: P component gain
T_n: Integral time

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. Find the appropriate compromise for T_n for your application, using these two extreme cases as outer limits.
   T_n must not be set too close to the instability limit, since there is a risk of damage to the machine if instability should occur.
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 K_R 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

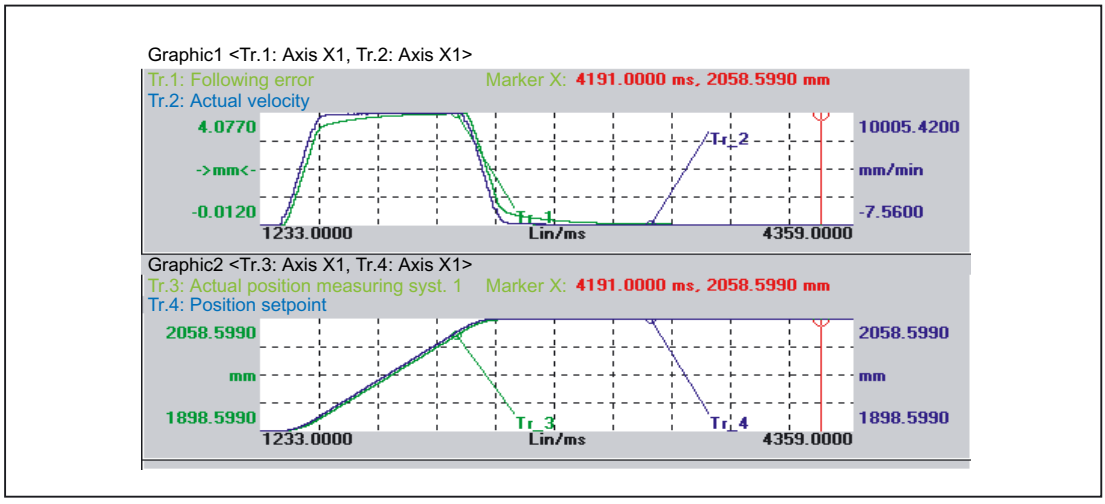


Fig. 2-14 Following error (1), actual velocity (2), position actual value (3), position setpoint (4)

2.5.5 System variable for status of pulse enable

Application

For all applications that must quickly react to pulse enabling, the status of the pulse enable is imaged to a new system variable in order to accelerate the braking signal.

This system variable is preferably evaluated in synchronized actions. Using the synchronized action, either a direct output to an NCK output can be carried out or a faster transfer to the PLC.

Functionality

Since 611D digital drives have no integrated braking signal, the brakes are normally controlled from the PLC. The brake can be closed again by deleting the pulse enable in the PLC.

If the pulse enable is deleted due to external events (611D interface, terminal 663 to the PLC) or due to drive or axis errors, the PLC can close the brake only with a delay, since the transport of the pulse enable signal via servo and interpolator requires 2 to 3 interpolator cycle clocks. In the worst case, the PLC needs another two PLC cycles. With hanging axes and linear motors, this is often slow.

System variable for enabling the drive power

Since the function must be available for all kinds of drives in the same form (also for non-electrical drives), the variable is given the name "Drive power enable".

Description	NCK variable
Status of the power enable (pulse enable) of a machine axis Predefined range of values: FALSE: no power enable TRUE: power enable exists	$VA_DPE [machine axis]

Activation

The variable is predefined and can be used at any time acc. to the data type BOOL with FALSE or TRUE. While communication with the drive is not yet established, the value 0 (FALSE) is supplied.

The synchronized actions, in which the variable is evaluated, should preferably be carried out as soon as the control powers up, e.g. by starting an ASUB from the PLC. To make sure that the synchronized actions remain continuously active and are not affected by the mode change, we recommend that they are programmed as "static synchronized actions" (using IDS as the vocabulary word).

Example

Output of the pulse enable of machine axis X1 to the first digital NCK output in all modes

```
IDS = 1 DO $A_OUT[1] = $VA_DPE[X1]
```

Supplementary conditions

The functional expansion is available for digital drives in all control variants providing synchronized actions.

Control response with POWER ON, RESET, REPOS, etc.

After POWER ON, the value 0 is supplied while communication with the drive is not yet established. Then, the system variable $VA_DPE always specifies the pulse enable value acquired at the beginning of the interpolation cycle. The modeindependent evaluation or retransmission should be carried out using static synchronized actions (vocabulary word IDS).

2.5.6 Expansions for "deceleration axes"

Application

In the case of designconditioned nonlinearities and elasticities, as often occurs in material handling and highbay racking technology, it is often necessary to sacrifice the position control due to the unstable position control loop. The axes are, therefore, traversed closed-loop controlled and not open-loop controlled. To this aim, the WF723 module offers the special function of the "deceleration axes".

Functionality

This "deceleration axes" function blocks any path movements, but allows positioning with approach to the target position by reducing the velocity step by step. The response corresponds to that of a multipoint controller.

For reasons relating to position control loop stability, the position controller algorithm is extended instead of the special positioning function "deceleration axes". It is thus also possible to operate drives that might cause problems using the position control.

Advantages

Compared with the existing method "deceleration axes", the expansion of the position controller algorithm provides the following advantages:

- The structure of the interpolation and control is extended only slightly.
  Thus, this axis automatically has all the properties of a normal axis.
- Traversing on the path also becomes possible; exactly which path can be foreseen considerably earlier, e.g. a highbay racking cage will take at different velocities. A possible collision can thus be avoided.
- It is much easier to configure the function.
- In particular, with feedforward control enabled, the positioning processes are faster.

Low-pass filter for analog drives

A lowpass filter connected at the position controller output now also allows the position controller gain to be reduced quickly with increasing frequency for analog drives (axes). This suppresses the effect of strong natural frequencies in the form of settable speed setpoint filters.

This functionality is provided as standard for digital drives.

MD32930/MD32940

The low-pass filter is activated via setting:
MD32930 $MA_POSCTRL_OUT_FILTER_ENABLE = 1.

The filter time constant is input via machine data:
MD32940 $MA_POSCTRL_OUT_FILTER_TIME.

MD32960 "dead zone"

Non-linearities close to zero speed, such as those, which can occur when simple frequency converters are used, are inhibited by a "dead zone" in the controller.

The threshold for system deviation, under which a speed setpoint of "zero" is output, can be set via machine data:
MD32960 $MA_POSCTRL_ZERO_ZONE.

The "dead zone" is input in MD32960 for each individual encoder.

Stability risk

The expanded position controller algorithm will reduce the risks that a stable control admits only a rather poor setting of the gain, or that the position control loop does not remain stable despite the effective extensions, to a minimum.

Supplementary conditions

The functional expansion is available for all control variants.

The low-pass filter is only activated if dynamic stiffness control is inactive:
MD32640 $MA_STIFFNESS_CONTROL_ENABLE = 0.

If the low-pass filter is active, the modeled following error is larger than usual during acceleration phases.
It can, therefore, be necessary to increase machine data:
MD36400 $MA_CONTOUR_TOL
above the standard value,
in order to prevent activation of axial contour monitoring (alarm 25050).

If the adjustable "dead zone" of the position controller retains its standard setting:
MD32960 $MA_POSCTRL_ZERO_ZONE = 0,
this corresponds to an input value, which is the same size as the current encoder fine resolution.
Therefore, the standard setting is compatible with earlier software versions.

If the "dead zone":
MD32960 $MA_POSCTRL_ZERO_ZONE
is configured to be larger than the zero-speed tolerance:
MD36030 $MA_STANDSTILL_POS_TOL,
zero-speed monitoring (alarm 25040) could be activated on termination of a positioning process.

If the "dead zone":
MD32960 $MA_POSCTRL_ZERO_ZONE
is larger than the exact stop limits:
MD36000 $MA_STOP_LIMIT_COARSE,
no exact stop signals are output.
This can lead to activation of position monitoring (alarm 25080)
and prevent block change.

Supplementary Conditions

3

3.1 Supplementary conditions

None

Examples

4

4.1 Examples

No examples are available.

Data Lists

5

5.1 Machine data

5.1.1 Memory specific 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

5.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB[n]	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
10080	SYSCLOCK_SAMPL_TIME_RATIO	Division factor of position-control cycle for actual-value acquisition
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[n]	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
13000	DRIVE_IS_ACTIVE[n]	Drive activation

Number	Identifier: $MN_	Description
13010	DRIVE_LOGIC_NR[n]	Logical drive number
13020	DRIVE_INVERTER_CODE[n]	Power section code of drive module
13030	DRIVE_MODULE_TYPE[n]	Module identifier
13040	DRIVE_TYPE[n]	Identifier of drive type
13050	DRIVE_LOGIC_ADDRESS[n]	Logical drive addresses
13060	DRIVE_TELEGRAM_TYPE[n]	Standard message frame type for PROFIBUS DP
13070	DRIVE_FUNCTION_MASK[n]	DP function used
13080	DRIVE_TYPE_DP[n]	Drive type PROFIBUS DP

5.1.3 Channel-specific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES[n]	Reset G groups

5.1.4 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30100	CTRLOUT_SEGMENT_NR[n]	Setpoint assignment: Drive type
30110	CTRLOUT_MODULE_NR[n]	Setpoint assignment: Drive number/module number
30120	CTRLOUT_NR[n]	Setpoint assignment: Setpoint output on drive module/module
30130	CTRLOUT_TYPE[n]	Output type of setpoint
30134	IS_UNIPOLAR_OUTPUT[n]	Setpoint output is unipolar
30200	NUM_ENCS	Number of encoders
30210	ENC_SEGMENT_NR[n]	Actual-value assignment: Drive type
30220	ENC_MODULE_NR[n]	Actual-value assignment: Drive module number/measuring circuit number
30230	ENC_INPUT_NR[n]	Actual-value assignment: Input on drive module/measuring circuit module
30240	ENC_TYPE[n]	Type of actual-value acquisition (position actual value)
30242	ENC_IS_INDEPENDENT	Encoder is independent
30300	IS_ROT_AX	Rotary axis
30350	SIMU_AX_VDI_OUTPUT	Output of axis signals for simulation axes
31000	ENC_IS_LINEAR[n]	Direct measuring system (linear scale)
31010	ENC_GRID_POINT_DIST[n]	Distance between reference marks on linear scales
31020	ENC_RESOL[n]	Encoder pulses per revolution
31030	LEADSCREW_PITCH	Leadscrew pitch

Number	Identifier: $MA_	Description
31040	ENC_IS_DIRECT[n]	Encoder is connected directly to the machine
31044	ENC_IS_DIRECT2	Encoder on intermediate gear
31050	DRIVE_AX_RATIO_DENOM[n]	Denominator load gearbox
31060	DRIVE_AX_RATIO_NUMERA[n]	Numerator load gearbox
31064	DRIVE_AX_RATIO2_DENOM	Intermediate gear denominator
31066	DRIVE_AX_RATIO2_NUMERA	Intermediate gear numerator
31070	DRIVE_ENC_RATIO_DENOM[n]	Measuring gear denominator
31080	DRIVE_ENC_RATIO_NUMERA[n]	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[n]	Sign actual value (feedback polarity)
32200	POSCTRL_GAIN [n]	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[n]	Rated output voltage
32260	RATED_VELO[n]	Rated motor speed
32450	BACKLASH[n]	Backlash
32500	FRICT_COMP_ENABLE	Friction compensation active
32610	VELO_FFW_WEIGHT	Feedforward control factor for speed feedforw. 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
32711	CEC_SCALING_SYSTEM_METRIC	System of measurement of sag compensation
32800	EQUIV_CURRCTRL_TIME [n]	Equivalent time constant current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME [n]	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	Activ. of low-pass filter at position controller output
32940	POSCTRL_OUT_FILTER_TIME	Time constant of low-pass filter at positon controller output
32950	POSCTRL_DAMPING	Factor for additional damping of position control loop
32960	POSCTRL_ZERO_ZONE[n]	Dead zone position controller
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

Number	Identifier: $MA_	Description
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

Index

$

$AC_TIME, 2-22
$VA_DPE, 2-74

A

Actual-value acquisition, 2-23
Actual-value correction, 2-25
Actual-value processing, 2-37, 2-56, 2-59
Actual-value resolution, 2-39
Actual-value routing, 2-25
Adapting the motor/load ratios, 2-33
Adjustments to actual-value resolution, 2-42
Axis parameter sets, 2-55

B

Basic system, 2-11

C

Closed-loop control, 2-49
Computational resolution, 2-6
Configuration of drives, 2-32
Control direction, 2-37
Conversion of basic system, 2-12

D

DB 31, ...
  DBX16.0 - 16.2, 2-55
DB10
  DBB71, 2-18
  DBX107.7, 2-18
DB31
  DBX1.5, 2-24
  DBX1.6, 2-24
Display resolution, 2-6
Drive configuration, 2-32
Dynamic response adaptation, 2-52

E

Encoder coding, 2-41
Encoder directly on tool, 2-35

F

Fine Interpolation, 2-50
FIPO, 2-50
Following error compensation (feedforward control)
  Speed feedforward control, 2-61
Function overview of inch/metric switchover, 2-20
  Data backup, 2-19
  Rounding machine data, 2-20
  Synchronized actions, 2-14

I

Inch measuring system, 2-11
Input resolution, 2-6
Intermediate gear, 2-35
Interpolation parameter sets, 2-55

L

Linear axis
  With linear scale, 2-43
  With rotary encoder on motor, 2-44
  With rotary encoder on the machine, 2-45

M

Manual switchover of the basic system
  General, 2-16
  Input resolution and computational resolution, 2-18
  JOG and handwheel factor, 2-19
  Reference point, 2-18
  System data, 2-17
  Tool data, 2-18
MD10000, 2-26
MD10050, 2-65
MD10082, 2-65

MD10200, 2-2, 2-3, 2-7, 2-18, 2-45
MD10210, 2-2, 2-3, 2-7, 2-47
MD10220, 2-6, 2-8
MD10230, 2-6, 2-8, 2-9
MD10240, 2-11, 2-13, 2-16, 2-18, 2-19, 2-20, 2-21
MD10260, 2-11, 2-16, 2-17, 2-19, 2-20
MD10270, 2-11
MD10290, 2-18
MD10292, 2-18
MD11220, 2-20
MD13000, 2-32, 2-33
MD13010, 2-27, 2-28, 2-32, 2-33
MD13020, 2-32, 2-33
MD13030, 2-32, 2-33
MD13040, 2-32, 2-33
MD13050, 2-33
MD13060, 2-33
MD13070, 2-33
MD13080, 2-33
MD1401, 2-38
MD1405, 2-39
MD18094, 2-18
MD18096, 2-18
MD20150, 2-12, 2-16, 2-66
MD2401, 2-38
MD2405, 2-39
MD30100, 2-26
MD30110, 2-27, 2-30, 2-32
MD30120, 2-27, 2-32
MD30130, 2-25, 2-27, 2-32
MD30134, 2-27
MD30200, 2-23, 2-31, 2-41, 2-61
MD30210, 2-28
MD30220, 2-28, 2-30, 2-31
MD30230, 2-28, 2-31
MD30240, 2-24, 2-25, 2-28, 2-31
MD30242, 2-25, 2-29
MD30300, 2-40, 2-41, 2-45, 2-47
MD30350, 2-25
MD31000, 2-40, 2-41, 2-45, 2-47
MD31010, 2-40, 2-41
MD31020, 2-40, 2-41, 2-45, 2-47
MD31030, 2-40, 2-41, 2-45
MD31040, 2-40, 2-41, 2-45, 2-47, 2-60
MD31044, 2-35, 2-40, 2-42
MD31050, 2-34, 2-40, 2-41, 2-45, 2-47, 2-54, 2-57
MD31060, 2-34, 2-40, 2-41, 2-45, 2-47, 2-54, 2-57
MD31064, 2-35, 2-42
MD31066, 2-35, 2-42
MD31070, 2-40, 2-41, 2-45, 2-47
MD31080, 2-40, 2-41, 2-45, 2-47
MD31090, 2-19
MD31200, 2-12
MD32000, 2-1, 2-39, 2-42
MD32100, 2-37
MD32200, 2-51, 2-54, 2-64, 2-65, 2-72
MD32210, 2-72
MD32220, 2-71, 2-72
MD32250, 2-37, 2-38
MD32400, 2-65, 2-69
MD32402, 2-65, 2-69
MD32410, 2-65, 2-69, 2-70
MD32450, 2-56, 2-59
MD32452, 2-54, 2-56, 2-57, 2-58, 2-59
MD32610, 2-54, 2-56, 2-58, 2-63, 2-65
MD32620, 2-62, 2-63, 2-65, 2-66, 2-67
MD32630, 2-62, 2-66, 2-67
MD32640, 2-65, 2-67, 2-76
MD32711, 2-11
MD32800, 2-54, 2-57, 2-63, 2-65, 2-67
MD32810, 2-54, 2-57, 2-63, 2-64, 2-65, 2-66
MD32900, 2-52, 2-53
MD32910, 2-52, 2-54, 2-57, 2-65
MD32930, 2-75
MD32940, 2-75
MD32950, 2-60, 2-61
MD32960, 2-75, 2-76
MD33000, 2-51
MD34080, 2-36, 2-42
MD34090, 2-36, 2-42
MD34320, 2-40, 2-41
MD35100, 2-1
MD36000, 2-54, 2-56, 2-76
MD36010, 2-54, 2-56
MD36012, 2-54, 2-56, 2-57, 2-58
MD36020, 2-71, 2-72
MD36030, 2-54, 2-56, 2-71, 2-72, 2-76
MD36040:, 2-71, 2-72
MD36110, 2-6
MD36200, 2-54, 2-57
MD36210, 2-38, 2-39
MD36400, 2-71, 2-72, 2-76
MD36500, 2-24
MD36510, 2-24
MD9004, 2-6
MD9011, 2-6
Measuring systems, 2-24
Metric measuring system, 2-11
Motor/load gear, 2-34

P

Parameter set changeover, 2-54
Parameter sets of the position controller, 2-54
Path feedrate, 2-3
Physical quantities, 2-7
Position control, 2-54

Position control loop, 2-49
Positioning accuracy, 2-4
Positioning axes, 2-3
PROFIBUS DP, 2-33
Pulse multiplication factor, 2-42

R

Resolutions, 2-6
Rotary axis
With rotary encoder on motor, 2-46
With rotary encoder on the machine, 2-48

S

Scaling, 2-6
Scaling of machine and setting data, 2-7
Servo gain factor, 2-51, 2-52
Setpoint output, 2-23
Setpoint system, 2-23
Setpoint/actual-value system|Configuration of drives for SINUMERIK 840Di, 2-32, 2-33
Simulation axes, 2-24
Speed control loop, 2-49
Speed setpoint adjustment, 2-37
Speed setpoint output, 2-37, 2-56, 2-59
Speed setpoint routing, 2-25
Spindle speed, 2-3

T

Traversing ranges, 2-3

V

Velocities, 2-1

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Auxiliary Function Output to PLC (H2)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Function 1-1

1.2 Overview of auxiliary functions 1-3

2 Detailed Description 2-1

2.1 Predefined auxiliary functions 2-1
2.1.1 Predefined auxiliary functions 2-1
2.1.2 Parameter: Group assignment 2-3
2.1.3 Parameter: Type, address extension, and value 2-3
2.1.4 Parameter: Output behavior 2-5
2.1.5 Examples of output behavior 2-8

2.2 User-defined auxiliary functions 2-11
2.2.1 User-specific and extended predefined auxiliary functions 2-11
2.2.2 Maximum number of user-defined auxiliary functions 2-12
2.2.3 Extension of predefined auxiliary functions 2-12
2.2.4 User-specific auxiliary functions 2-14
2.2.5 Parameterization 2-14
2.2.5.1 Parameter: Group assignment 2-14
2.2.5.2 Parameter: Type, address extension, and value 2-14
2.2.5.3 Parameter: Output behavior 2-16
2.2.5.4 Examples 2-16

2.3 Type-specific output behavior 2-18

2.4 Programmable output duration 2-20

2.5 Priorities of the output behavior 2-21

2.6 Auxiliary function output to the PLC 2-23

2.7 Programming 2-23

2.8 Auxiliary functions without block change delay 2-25

2.9 Associated auxiliary functions 2-25

2.10 M function with implicit preprocessing stop 2-27

2.11 Response to overstore 2-28

2.12 Block search 2-29
2.12.1 Behavior on block search with calculation 2-29
2.12.2 Output suppression of spindle-specific auxiliary functions 2-30

2.13 Querying and displaying of output M help functions 2-34
2.13.1 Possibilities of information 2-34
2.13.1.1 Status display on the operator interface 2-34
2.13.1.2 Programming status query 2-36

3 Supplementary conditions 3-1
3.1 General constraints 3-1
3.2 Output behavior 3-2
4 Examples 4-1
4.1 Defining auxiliary functions 4-1
5 Data Lists 5-1
5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Channel-specific machine data 5-1
5.2 Signals 5-2
5.2.1 Signals to channel 5-2
5.2.2 Signals from channel 5-2
5.2.3 Signals to axis/spindle 5-4
5.2.4 Signals from axis/spindle 5-5
Index Index-1

Brief Description

1

1.1 Function

General

Auxiliary functions permit activation of NC system functions and PLC user functions. Auxiliary functions can be programmed in part program blocks in the following:

- Part programs
- Synchronized actions
- User cycles

Detailed information on using auxiliary function output in synchronized actions is to be found in:

References:
/FBSY/ Description of Functions 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	Miscellaneous (i.e. special) function	M30	End of program
S	Spindle function	S100	Spindle speed 100, e.g. 1 rpm
T	Tool number	T3	Tool number 3
D, DL	Tool offset	D1	Tool cutting edge number 1
F	Feed	F1000	Feedrate 1000, e.g. mm/min

User-defined 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

Extension of predefined auxiliary functions refers to the parameter: "Address extension". The address extension defines the number of the spindle to which the auxiliary function applies. For example, 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	Miscellaneous (i.e. special) function	M2=3	2. spindle: spindle right
S	Spindle function	S2=100	2. spindle: spindle speed = 100, e.g. 1 rpm
T	Tool number	T2=3	

User-specific auxiliary functions

One feature of user-specific auxiliary functions is that the "Type" parameter must always be H. User-specific auxiliary functions do not activate system functions. User-specific auxiliary functions are output by the NC to the NC/PLC interface only. The functionality of the auxiliary functions must be implemented by the machine manufacturer/user via the PLC user program.

Type	Function	Example	Meaning
H	Auxiliary function	H2=5	User-specific function

Definition of an auxiliary function

An auxiliary function is defined by the following parameters:

- Type, address extension and value
  The three parameters are output at the NC/PLC interface.
- Output behavior
  The auxiliary-function-specific output behavior defines how long an auxiliary function is output to the NC/PLC interface and when it is output relative to the traverse movement programmed in the same part program block.
- Group assignment
  An auxiliary function can be assigned to an 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:

References:
/FB/ Description of Function Basic Machines, PLC Basic Program (P3)

1.2 Overview of auxiliary functions

M functions

M (special function)					
Address extension		**Value**			
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
0 (implicit)	- - -	Up to 8 digits	INT	Function	5
Remarks: - - -					
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
1 - 12	Spindle number	1 – 99	INT	Function	5
Remarks: Example: "Spindle stop" for 2nd spindle of the channel: M2=5. The master spindle of the channel is addressed if no an address extension is specified.					
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
0 - 99	Any	2147483647	INT	Function	5
Remarks: User-specific M function					

Application

Controlling machine functions in synchronism with the part program.

General remarks

- 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.
  References
  /FB/ Description of Function Basic Machines; PLC Basic Program (P3)
- For subroutines, there are machine data in which you can set whether the M function for the end of part program (M17, M2, and M30) will be output to the PLC:
  MD20800 $MC_SPF_END_TO_VDI
- 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**			
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
0 - 12	Spindle number [5)]	0 - +/-3.4028 ex 38 [3)]	REAL	Spindle speed	3
Remarks: The master spindle of the channel is addressed if no address extension is specified.					

Application

Spindle speed.

Remarks

- 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

H functions

H (aux. function)					
Address extension		**Value**			
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
0 - 99	Any	-2147483648 - +2147483647	INT	Any	3
		0 - +/-3.4028exp38 [2) 3) 4)]	REAL		
Remarks: The functionality must be implemented by the user in the PLC user program.					

Application

User-specific auxiliary functions.

Remarks

- H-function-specific machine data:
  - MD22110 $MC_AUXFU_H_TYPE_INT
  - MD22230 $MC_AUXFU_H_SYNC_TYPE

T functions

T (tool number) [5) 6)]					
Address extension		**Value**			
Value range	**Meaning**	**Value range**	**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)]					

Application

Tool selection.

Remarks

- Identification of tools either by means of tool number or location number.
  References
  /FBW/ Tool Management
  /FB/ Description of Functions Basic Machine; Tool Compensation (W1)
- When T0 is selected, the current tool is removed from the tool holder but not replaced by a new tool (default setting).
- T-function-specific machine data:
  – MD22220 $MC_AUXFU_T_SYNC_TYPE

D functions

D (tool offset)					
Address extension		**Value**			
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
- - -	- - -	0 - 9	INT	Selection of the tool offset	1
Remarks: Deselection of the tool offset with D0; the default is D1					

Application

Selection of the tool offset.

Remarks

- Initial setting: D1
- The default tool cutting edge after tool change can be parameterized via:
  MD20270 $MC_CUTTING_EDGE_DEFAULT
- Deselection of the tool offset: D0
- D-function-specific machine data:
  – MD22250 $MC_AUXFU_D_SYNC_TYPE

DL functions

DL (additive tool offset)					
Address extension		**Value**			
Value range	**Meaning**	**Value range**	**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.					

Application

Selection of the additive tool offset with reference to an active tool offset.

Remarks

- 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
- Deselection of the additive tool offset: DL = 0
- DL-function-specific machine data:
  - MD22252 $MC_AUXFU_DL_SYNC_TYPE

F functions

F (feedrate)					
Address extension		**Value**			
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
- - -	- - -	0.001 - 999 999.999	REAL	Path feed	6
Remarks: - - -					

Application

Path velocity.

Remarks

- F-function-specific machine data:
  - MD22240 $MC_AUXFU_F_SYNC_TYPE

FA functions

FA (axial feedrate)					
Address extension		**Value**			
Value range	**Meaning**	**Value range**	**Type**	**Meaning**	**Number**[8)]
1 - 31	Axis number	0.001 - 999 999.999	REAL	Axial feedrate	6
Remarks: - - -					

Application

Axial velocity.

Remarks

- F-function-specific machine data:
  – MD22240 $MC_AUXFU_F_SYNC_TYPE

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 MD 22110: 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 full precision.
4) If the setting is MD22110 $MC_AUXFU_H_TYPE_INT = 1, the REAL values are rounded before output to the PLC. 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: Value range of address extension
   - without tool management: 0 – 99
   - with tool management: 0 – maximum spindle number

   0: replace by the value of the master spindle number or the master tool holder
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.

Detailed Description 2

2.1 Predefined auxiliary functions

2.1.1 Predefined auxiliary functions

Function

Predefined auxiliary functions are auxiliary functions for activating system functions. The assignment of a predefined auxiliary function to system function cannot be changed.

During execution of a predefined auxiliary function, the corresponding system function is activated and the auxiliary functions are output to the NC/PLC interface.

Definition of a predefined auxiliary function

The parameters of a predefined auxiliary function are stored in machine data and can be changed in some cases. All parameters of an auxiliary function have the same index.

- MD22040 $MC_AUXFU_PREDEF_GROUP[index] (group assignment)
- MD22050 $MC_AUXFU_PREDEF_TYPE[index] (type)
- MD22060 $MC_AUXFU_PREDEF_EXTENSION[index] (address extension)
- MD22070 $MC_AUXFU_PREDEF_VALUE[index] (value)
- MD22080 $MC_AUXFU_PREDEF_SPEC[index] (specification of output behavior)

Predefined auxiliary functions

System function														
	Index n (index for the machine data 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]									
						Output behavior: **Bits 0 - 8** MD22080 $MC_AUXFU_PREDEF_SPEC[n]								
						8	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Stop	0	M	0	0	1	0	1	0	0	0	0	0	0	1
Conditional stop	1	M	0	1	1	0	1	0	0	0	0	0	0	1
End of subroutine	2	M	0	2	1	0	1	0	0	0	0	0	0	1
	3	M	0	17	1	0	1	0	0	0	0	0	0	1
	4	M	0	30	1	0	1	0	0	0	0	0	0	1
Tool change	5	M	(1)	6 [1)]	(1)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	0	1
Spindle right	6	M	(1)	3	(2)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	0	1
Spindle left	7	M	(1)	4	(2)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	0	1
Spindle stop	8	M	(1)	5	(2)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	0	1
Position spindle	9	M	(1)	19	(2)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	0	1
Axis mode	10	M	(1)	70 [2)]	(2)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	0	1
Automatic gear stage	11	M	(1)	40	(4)	(0)	0	0	1	(0)	(0)	(0)	0	1
Gear stage 1	12	M	(1)	41	(4)	(0)	0	0	1	(0)	(0)	(0)	0	1
Gear stage 2	13	M	(1)	42	(4)	(0)	0	0	1	(0)	(0)	(0)	0	1
Gear stage 3	14	M	(1)	43	(4)	(0)	0	0	1	(0)	(0)	(0)	0	1
Gear stage 4	15	M	(1)	44	(4)	(0)	0	0	1	(0)	(0)	(0)	0	1
Gear stage 5	16	M	(1)	45	(4)	(0)	0	0	1	(0)	(0)	(0)	0	1
Spindle speed	17	S	(1)	-1	(3)	0	(0)	(1)	(0)	(0)	(0)	0	0	1
Feed	18	F	0	-1	(1)	0	(0)	(1)	(0)	0	(0)	0	0	1
Cutting edge selection	19	D	0	-1	(1)	0	(0)	(1)	(0)	0	(0)	0	0	1
DL	20	L	0	-1	(1)	0	(0)	(1)	(0)	0	(0)	0	0	1
Tool selection	21	T	(1)	-1	(1)	0	(0)	(1)	(0)	0	(0)	0	0	1
Stop (associated)	22	M	0	-1 [3)]	1	0	1	0	0	0	0	0	0	1
Conditional stop (associated)	23	M	0	-1 [4)]	1	0	1	0	0	0	0	0	0	1
End of subroutine	24	M	0	-1 [5)]	1	0	1	0	0	0	0	0	0	1
Nibbling	25	M	0	20 [6)]	(10)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1
Nibbling	26	M	0	23 [6)]	(10)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1

Nibbling	27	M	0	22 [6)]	(11)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1
Nibbling	28	M	0	25 [6)]	(11)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1
Nibbling	29	M	0	26 [6)]	(12)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1
Nibbling	30	M	0	122 [6)]	(11)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1
Nibbling	31	M	0	125 [6)]	(11)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1
Nibbling	32	M	0	27 [6)]	(12)	(0)	(0)	(1)	(0)	0	(0)	(0)	0	1

1)	The value depends on: MD22560 $MC_TOOL_CHANGE_M_MODE
2)	A different value can be preset via the following machine data: - MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR or - MD20094 $MC_SPIND_RIGID_TAPPING_M_NR. Value 70 is always output to the PLC.
3)	The value is set via MD22254 $MC_AUXFU_ASSOC_M0_VALUE.
4)	The value is set via MD22256 $MC_AUXFU_ASSOC_M1_VALUE.
5)	The value is set via MD10714 $MN_M_NO_FCT_EOP.
6)	The value is set via MD26008 $MC_NIBBLE_PUNCH_CODE.
( )	Only values in parentheses can be changed.

2.1.2 Parameter: Group assignment

Group assignment

A predefined auxiliary function can be assigned via the group assignment of a particular auxiliary function group:
MD22040 $MC_AUXFU_PREDEF_GROUP[index] (group assignment)

If the value is zero, the auxiliary function is not assigned to any auxiliary function group.

For the meanings of the auxiliary function groups, see Section: Auxiliary function groups.

2.1.3 Parameter: Type, address extension, and value

Function

A predefined auxiliary function is programmed via parameters:
MD22050 $MC_AUXFU_PREDEF_TYPE[index] (type)
MD22060 $MC_AUXFU_PREDEF_EXTENSION[index] (address extension)
MD22070 $MC_AUXFU_PREDEF_VALUE[index] (value)

with the following syntax:
< type > [< address extension > =] < value >

Parameter: Type

The name of an auxiliary function is defined via the "type", e.g.:

- M: Special functions
- S: Spindle functions
- F: Feed

The "type" cannot be changed for predefined auxiliary functions.

Parameter: 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.

If no address extension is programmed, the address extension is implicitly set = 0. Auxiliary functions with the address extension = 0 always implicitly refer to the master spindle of the channel.

Example

Programming	Function
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

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

For all spindles of the channel, the auxiliary function M3 (machine data index = 6, corresponding to the table in Section: Predefined auxiliary functions) is assigned to the 2nd auxiliary function group.

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

Parameter: Value

Parameters: "Value" and "Type" define the meaning of an auxiliary function. I.e. the system function that is activated as a result of this auxiliary function. The "value" cannot be changed for a predefined auxiliary function. The "value" can be reconfigured via additional machine data for some predefined auxiliary functions. See Section: Associated auxiliary functions.

2.1.4 Parameter: Output behavior

Function

The "output behavior" defines when the auxiliary function is output to the NC/PLC interface and when it is acknowledged by the PLC:
MD22080 $MC_AUXFU_PREDEF_SPEC[index] (specification of output behavior)

Bit	Value	Meaning
0	1	Output duration one OB1 cycle (normal acknowledgment)
1	1	Output duration one OB40 cycle (quick acknowledgment)
2	1	No predefined auxiliary function
3	1	No output (can only be set as a single bit)
4	1	Spindle response following acknowledgment
5	1	Output prior to motion
6	1	Output during motion
7	1	Output at block end
8	1	The collected auxiliary function is not output after a block search.

Bit0: Output duration one OB1 cycle (normal acknowledgment)

An auxiliary function with normal acknowledgment is output to the NC/PLC interface at the beginning of the OB1 cycle. Auxiliary-function-specific change signals indicate 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 part 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 traverse movement and before reaching the end of block.

Bit1: Output duration one OB40 cycle (quick acknowledgment)

An auxiliary function with quick acknowledgment is output to the NC/PLC interface before the next OB1 cycle. Auxiliary-function-specific change signals indicate 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.

Bit2: No predefined auxiliary function

A predefined auxiliary function is treated like a user-defined auxiliary function if "No predefined auxiliary function" is set. The auxiliary function then no longer triggers the corresponding system function but is only output to the PLC.

Example:
Reconfiguration of auxiliary function "Position spindle" with index 9 (see table in Chapter: Predefined auxiliary functions), to a "user-defined" auxiliary function with normal acknowledgment and output of the traversing movement.

MD22080 $MC_AUXFU_PREDEF_SPEC [9] = 'H25' (100101_B)

Bit3: No output

The auxiliary function is not output to the PLC.

Bit4: Spindle response following acknowledgment

The corresponding spindle function is only started after acknowledgment by the PLC.

Bit5: Output prior to motion

The auxiliary function is output before the traverse movements programmed in the part program block (path and/or block-related positioning axis movements).

Bit6: Output during motion

The auxiliary function is output during the traverse movements programmed in the part program block (path and/or block-related positioning axis movements).

Bit7: Output at block end

The auxiliary function is output after completion of the traverse movements programmed in the part program block (path and/or block-related positioning axis movements).

Bit8: No output after a block search

The collected auxiliary function is not output after a block search.

Note

In the case of auxiliary functions for which no output behavior has been defined, the following default output behavior is active:

- Output duration one OB1 cycle
- Output at block end

Context: Output duration and output relative to movement

Output prior to motion

- The traverse movements (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 part program block.
- Traverse movements of the current part program block (path and/or positioning axis movements) are 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 traverse movements (path and/or positioning axis movements).
- The path velocity of the current part 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 traverse movements (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 traverse movements.
- 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

2.1.5 Examples of output behavior

Inputs

The following figures illustrate the differing behavior regarding:

- Output and acknowledgment of the auxiliary function
- Spindle response (speed change)
- Traverse movement (velocity change)

Parameterized output behavior
The binary values stated in the examples under "Spec" refer to the parameterizable output behavior:
MD22080 $MC_AUXFU_PREDEF_SPEC[Index]

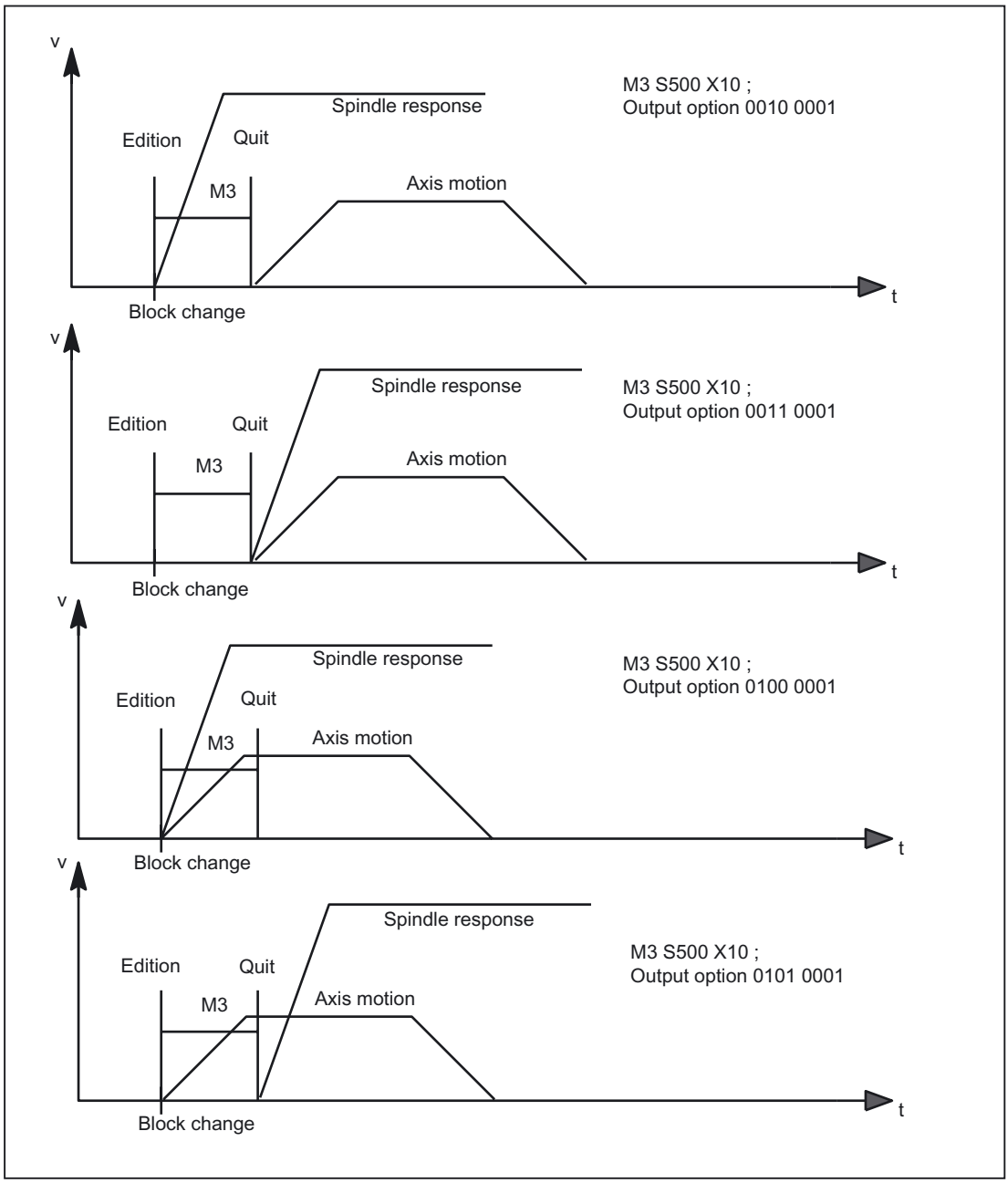


Fig. 2-1 Output behavior 1

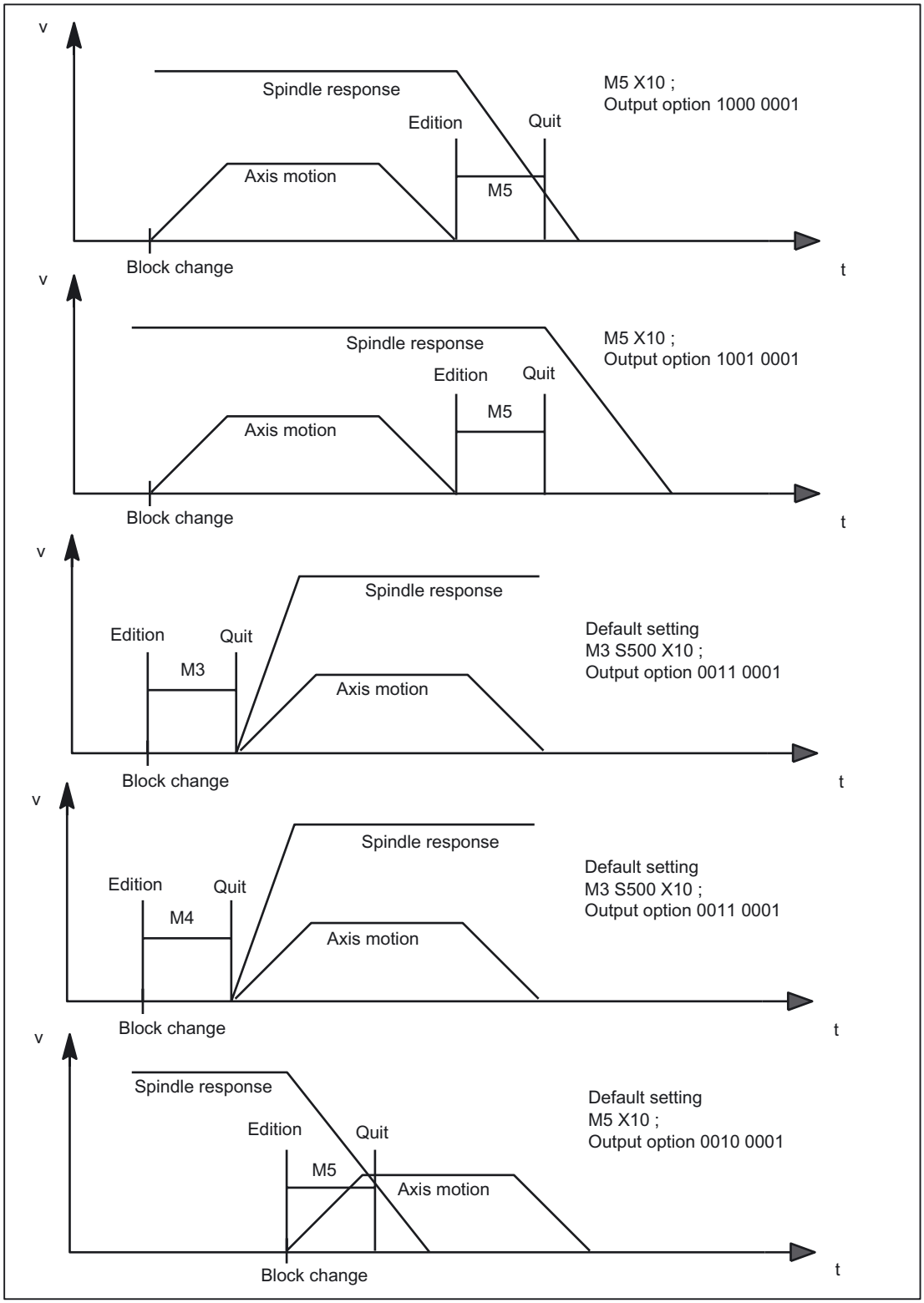


Fig. 2-2 Output behavior 2

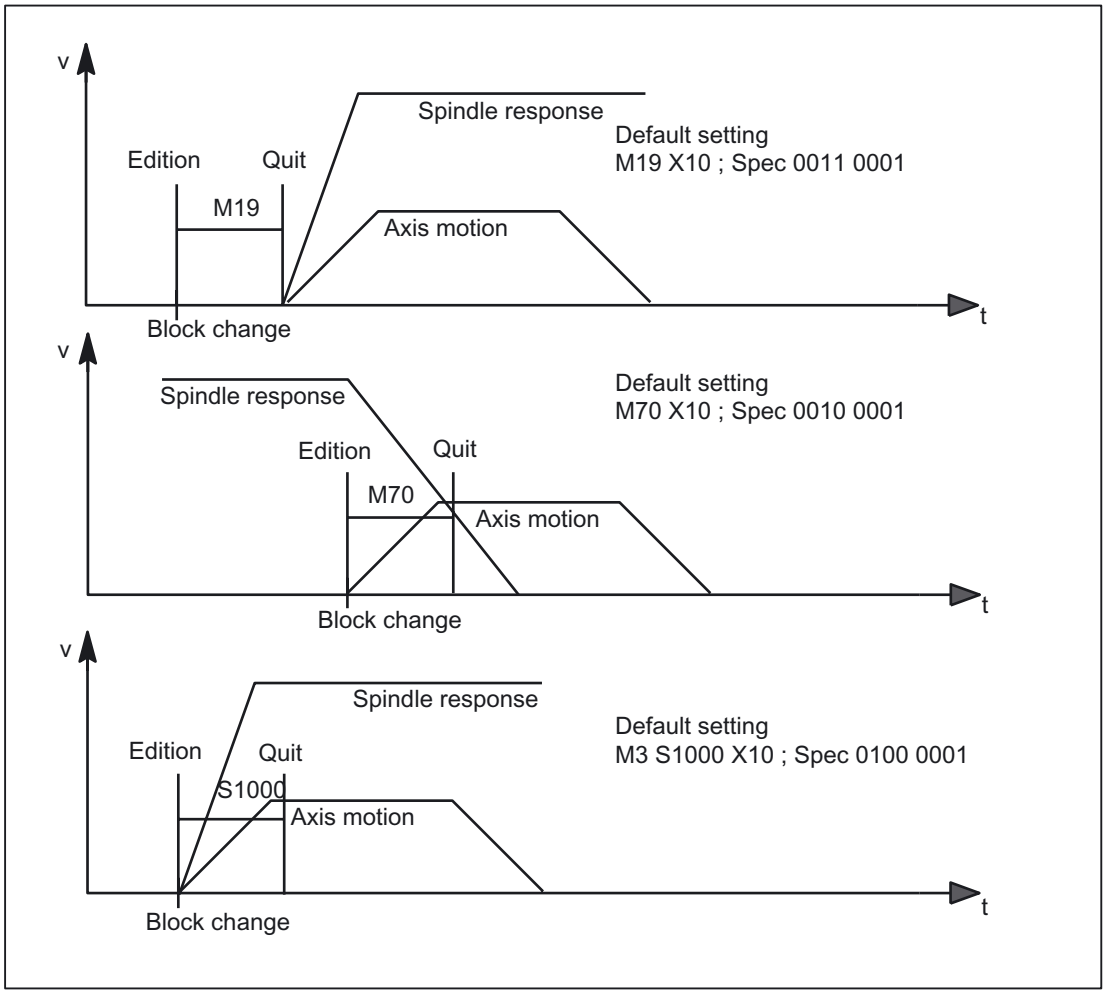


Fig. 2-3 Output behavior 3

2.2 User-defined auxiliary functions

2.2.1 User-specific and extended predefined auxiliary functions

Function

There are two uses for user-defined auxiliary functions:

- Extension of predefined auxiliary functions
- User-specific auxiliary functions

Extension of predefined auxiliary functions

Extension of predefined auxiliary functions refers to the parameter: "Address extension". The address extension defines the number of the spindle to which the auxiliary function applies.

User-specific auxiliary functions

One feature of user-specific auxiliary functions is that the parameter "Type" must always be H. They refer only to user functions. User-specific auxiliary functions do not activate system functions.

2.2.2 Maximum number of user-defined auxiliary functions

Maximum number

The maximum number of user-defined auxiliary functions per channel can be parameterized via:
MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN (maximum number of user-definable auxiliary functions)

2.2.3 Extension of predefined auxiliary functions

Function

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.

The extension of predefined auxiliary functions refers only to the parameter: "Address extension". 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 2nd and 3rd spindle of the channel.

Auxiliary function "Spindle right" for the 2nd 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 3rd 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

2.2.4 User-specific auxiliary functions

Function

User-specific auxiliary functions have the following properties:

- For parameter: "Type", only identifier "H" is permitted.
- User-specific auxiliary functions only activate user functions.
- User-specific auxiliary functions cannot activate system functions.
- A user-specific auxiliary function is output to the PLC in accordance with the parameterized output behavior.
- The functionality of a user-specific auxiliary functions is implemented by the machine manufacturer/user in the PLC user program.

2.2.5 Parameterization

2.2.5.1 Parameter: Group assignment

Group assignment

A user-defined auxiliary function can be assigned via the group assignment of a particular auxiliary function group:
MD22000 $MC_AUXFU_ASSIGN_GROUP[index] (group assignment)

If the value is zero, the auxiliary function is not assigned to any auxiliary function group.

For the meanings of the auxiliary function groups, see Section: Auxiliary function groups.

2.2.5.2 Parameter: Type, address extension, and value

Function

A user-defined auxiliary function is programmed with parameters:
MD22010 $MC_AUXFU_ASSIGN_TYPE[index] (type)
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[index] (address extension)
MD22030 $MC_AUXFU_ASSIGN_VALUE[index] (value)

with the following syntax:
< type > [< address extension > =] < value >

Parameter: Type

The name of an auxiliary function is defined via the "Type". Only the following names are permitted for user-defined auxiliary functions:

- User-specific auxiliary functions
  - H: Auxiliary function
- Extension of predefined auxiliary functions
  - M: Miscellaneous (i.e. special) function
  - S: Spindle function
  - T: Tool number

Parameter: Address extension

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
To assign all auxiliary functions of the same type and value to the same auxiliary function group, a value "–1" must be entered for parameter "Address extension".

Example
All user-specific auxiliary functions with the value = 8 are assigned to auxiliary function group 10.

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

Parameter: Value

The functionality of parameter: "Value" is not defined in user-specific. auxiliary functions. The value is generally used to activate the corresponding PLC user function.

Grouping together auxiliary functions
To assign all auxiliary functions of the same type and address extension to the same auxiliary function group, a value "–1" must be entered for parameter "Value".

Example
All user-specific auxiliary functions with the address extension = 2 are assigned to auxiliary function group 11.

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

2.2.5.3 Parameter: Output behavior

Function

The "Output behavior" of user-defined auxiliary functions can be parameterized via:
MD22035 $MC_AUXFU_ASSIGN_SPEC[index] (specification of output behavior)

For a description of the individual output parameters, see Section: "Parameter output behavior" of the predefined auxiliary functions. The information given there can be applied analogously to the output behavior of user-defined auxiliary functions.

2.2.5.4 Examples

Examples of the extension of predefined auxiliary functions

The following auxiliary functions are parameterized for the 2nd spindle of the channel:

Parameterization: M3

- Machine data index: 0 (1. 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

Machine data	Machine data index	Value
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

Parameterization: M4

- Machine data index: 1 (2. 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

Machine data	Machine data index	Value
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

Parameterization: M5

- Machine data index: 2 (3. 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

Machine data	Machine data index	Value
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

2.3 Type-specific output behavior

Function

The output behavior of the auxiliary function relative to a traverse motion programmed in the part program block can be defined type-specifically in the following machine data:

- MD22200 $MC_AUXFU_M_SYNC_TYPE (output time M functions)
- MD22210 $MC_AUXFU_S_SYNC_TYPE (output time S functions)
- MD22220 $MC_AUXFU_T_SYNC_TYPE (output time T functions)
- MD22230 $MC_AUXFU_H_SYNC_TYPE (output time H functions)
- MD22240 $MC_AUXFU_F_SYNC_TYPE (output time F functions)
- MD22250 $MC_AUXFU_D_SYNC_TYPE (output time D functions)
- MD22252 $MC_AUXFU_DL_SYNC_TYPE (output time DL functions)

Output behavior

The following output behaviors can be parameterized:
MD $MC_AUXFU_xx_SYNC_TYPE = <value>

Value	Meaning
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 predefined output specification

For a description of the various output behaviors, see Section: "Predefined auxiliary functions" > "Parameter: Output behavior".

Note

For the output behaviors that can be set for each type of auxiliary function, please refer to the relevant detailed machine data description.

References
/LIS/ Lists

Example

Output of auxiliary functions with different output behaviors in a part program block with traverse movement.

Parameterized output behavior

- T function: Output **before** the movement
  MD22220 $MC_AUXFU_T_SYNC_TYPE = 0
- M function: Output **during** the movement
  MD22200 $MC_AUXFU_M_SYNC_TYPE = 1
- H function: Output **at end of block**
  MD22230 $MC_AUXFU_H_SYNC_TYPE = 2

Part program block
N10 G01 X100 M07 H5 T5

Time sequence for auxiliary function output

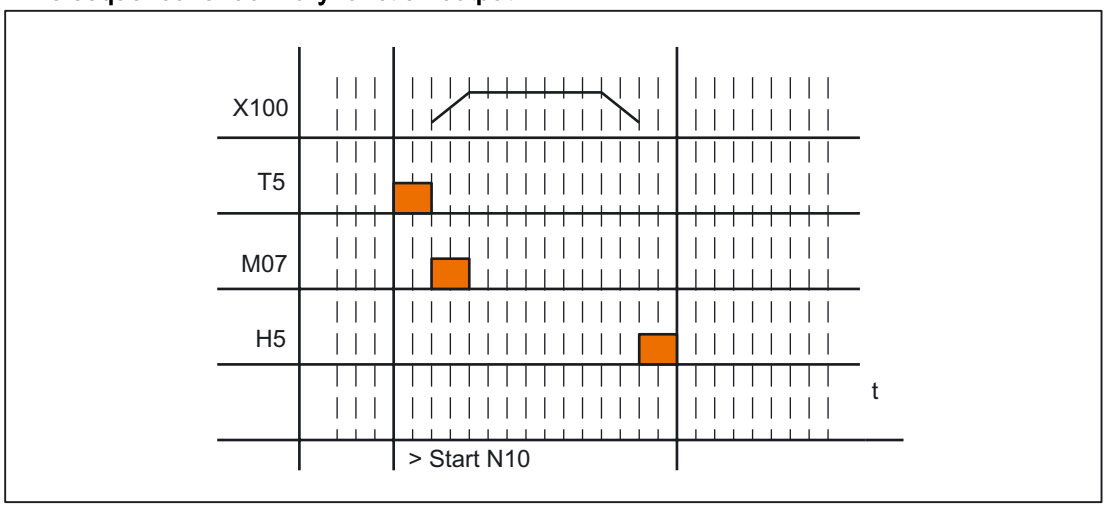


Fig. 2-4 Example of auxiliary function output

2.4 Programmable output duration

Function

The user-specific auxiliary functions for which the output behavior: "Output duration one OB1 cycle (slow acknowledgment)" was parameterized, can be defined for each output via the part program instruction QU (quick) for auxiliary functions with quick acknowledgment.

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 on the output of auxiliary functions M100 and M200 in a part 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

Programming	Comment
N10 G94 G01 X50 M100	Output of M100: during movement Acknowledgment: slow
N20 Y5 M100 M200	Output of M200: prior to movement Output of M100: during movement Acknowledgment: slow
N30 Y0 M=QU(100) M=QU(200)	Output of M200: prior to movement Output of M100: during movement Acknowledgment: quick
N40 X0	- - -
N50 M100 M200	Output of M200: immediately 1) Output of M100: immediately 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 image difference during execution of part program blocks N20 to N30.

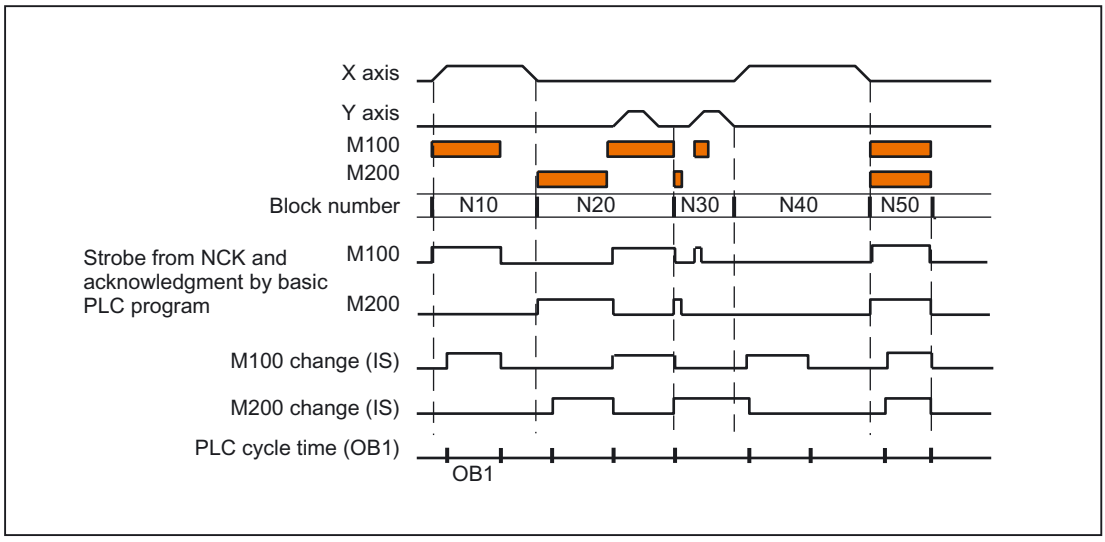


Fig. 2-5 Example of auxiliary function output

2.5 Priorities of the output behavior

Areas of the output behavior

The priority must be observed for the following areas in connection with the parameterized output behavior of an auxiliary function:

- Output duration (normal/quick acknowledgment)
- Output relative to motion (prior to/during/after the motion)

Priority sequence
The rule for the priority sequence is that the parameterized output behavior with lower priority becomes active if no output behavior with higher priority has been parameterized.

Area: Output duration

The following priorities apply to the output duration:

Priority	Output behavior	Defined via:
Highest	Specific to auxiliary functions	Part program instruction: QU(…)
	Specific to auxiliary functions	MD22035 $MC_AUXFU_ASSIGN_SYNC MD22080 $MC_AUXFU_PREDEF_SYNC
	Group-specific	MD11110 $MC_AUXFU_GROUP_SPEC
Lowest	Not defined	Default output behavior: Output duration one OB1 cycle

Area: Output relative to motion

The following rules apply to output relative to motion:

Priority	Output behavior	Defined via:
Highest	Specific to auxiliary functions	MD22035 $MC_AUXFU_ASSIGN_SYNC MD22080 $MC_AUXFU_PREDEF_SYNC
	Group-specific	MD11110 $MC_AUXFU_GROUP_SPEC
	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.

2.6 Auxiliary function output to the PLC

Function

On output of an auxiliary function to the PLC, the following signals and values are passed to the NC/PLC interface:

- Change signals
- Parameter: "Address extension"
- Parameter: "Value"

NC/PLC interface

The change signals and values of the auxiliary functions are within the following data range in the NC/PLC interface:

- Change signals on auxiliary function transfer from NC channel
  DB21, ... DBB58 – DBB67
- Transferred M and S functions
  DB21, ... DBB68 – DBB112
- Transferred T/D/DL functions
  DB21, ... DBB116 – DBB136
- Transferred T/D/DL functions
  DB21, ... DBB140 – DBB190
- Decoded M signals (M0 - M99)
  DB21, ... DBB194 – DBB206

References

A detailed description of the NC/PLC interface is to be found in:
/LIS/ Lists, Signals to/from channel (DB21 – DB30)

The access methods for the NC/PLC interface are described in:
/FB/ Description of Function Basic Machines, PLC Basic Program (P3)

2.7 Programming

Syntax

An auxiliary function is programmed in a part program block with the following syntax:

< type > [< address extension > =] < value >

Address expansion
If no address extension is programmed, address extension = 0 is implicitly set.

Symbolic addressing

The values for parameter: The "Address expansion" and "Value" can also be defined 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 1st spindle:

Programming syntax	Meaning
DEF SPINDEL_NR = 1	; 1. spindle in the channel
DEF DREHRICHTUNG = 3	; Clockwise rotation
N100 M[SPINDEL_NR] = DREHRICHTUNG	; In accordance with: 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 numeric value is transferred instead.

Programming examples

Programming examples of auxiliary functions with the corresponding values for output to the PLC.

Programming syntax	Output to PLC
DEF coolant = 12	- - -
DEF lubricant = 130	- - -
H[coolant]=lubricant	H12=130
H=coolant	H0=12
H5	H0=5
H=5.379	H0=5.379
H17=3.5	H17=3.5
H[coolant]=13.8	H12=13.8
H='HFF13'	H0=65299
H='B1110'	H0=14
H5.3=21	Error

2.8 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 prior to motion" or "Output during motion"

continuous-path mode can cause drops in velocity in continous-path mode (short traverse paths and high velocities). This is because it is necessary 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:
MD22100 $MC_AUXFU_QUICK_BLOCKCHANGE (block change delay with fast auxiliary functions)

Constraints

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, acknowledgment comes one OB40 cycle and execution of the auxiliary function comes one OB1 cycle after the change to the next part program block.

2.9 Associated auxiliary functions

Function

Associated auxiliary functions are user-defined auxiliary functions that have the same effect as the corresponding predefined auxiliary functions. For the following predefined auxiliary functions, user-defined auxiliary functions can be associated:

- 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 permitted as parameter "Type" of the user-defined auxiliary function.

Parameterization

Association of a user-defined auxiliary function with one of the predefined auxiliary functions mentioned is set in the following 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

General information

In the case of an associated user-defined auxiliary function, the same signals are output to the NCK/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 (Parameter: "Value") is output. This means it is possible to distinguish between predefined and user-defined auxiliary functions in the PLC user program.

Note

A change to the following machine data may require adjustment of the PLC user program:

- MD22254 $MC_AUXFU_ASSOC_M0_VALUE
- MD22256 $MC_AUXFU_ASSOC_M1_VALUE

Specific NC/PLC interface signals
The following specific NCK/PLC interface signals are available:

- DB 21, ... DBX318.5 (associated M00/M01 active) Check-back signal
- DB 21, ... DBX30.5 (activate associated M01) Activation signal

Example

Associating the user-defined auxiliary functions M123 with M0:

MD22254 $MC_AUXFU_ASSOC_M0_VALUE = 123

The user-defined auxiliary function M123 has the same functionality as M0 (stop).

Constraints

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.) must not be associated.

2.10 M function with implicit preprocessing stop

Function

Triggering a preprocessing stop in conjunction with an auxiliary function can be programmed via the STOPRE part program command. Triggering a preprocessing stop in conjunction with an M 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 (M function with preprocessing stop)

Preprocessing stop with M function M88

The user-defined M function M88 is intended to trigger a preprocessing stop.

Parameterization
MD10713 $MN_M_NO_FCT_STOPRE [0] = 88

Application
Part program (extract)

```
Programming          Comment
N100 G0 X10 M88      ; Traverse movement and implicit preprocessing stop via M88
N110 Y=R1            ; N110 is only interpreted after completion of the traverse
                     ; movement and acknowledgment of the M function.
```

Constraints

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)

2.11 Response to overstore

Overstore

On the SINUMERIK HMI operator interface, the following functions:

- NC START of a part program
- NC START to resume an interrupted part program

the auxiliary functions that are output with the start, can be changed with 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

Basically, 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) remains valid, until it is overwritten by another auxiliary function of the same auxiliary function group by overstoring again or by programming in a part program block.

2.12 Block search

2.12.1 Behavior on block search with calculation

Function

Block searches with calculation collect up auxiliary functions on a groupspecific basis. The last auxiliary function in each auxiliary function group is output after NC-START, before the actual reentry block, in a separate part program block that has the following output behavior:

- Output duration one OB1 cycle (normal acknowledgment)
- Output prior to motion

Output control

Bit 8 of machine data $MC_AUXFU_PREDEF_SPEC[index], $MC_AUXFU_ASSIGN_SPEC[index], and $MN_AUXFU_GROUP_SPEC[index] can be used to configure whether or not the corresponding auxiliary function is output to the PLC after a block search. This behavior does not affect the display and does not affect variables $AC_AUXFU_M_STATE[n], $AC_AUXFU_M_VALUE[n], and $AC_AUXFU_M_EXT[n]. The auxiliary function is always regarded as collected after a block search, even though it is not output to the PLC. An auxiliary function that is not output after a block search also overwrites an auxiliary function whose Bit 8 is not set during collection. 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 synchronous actions.

Auxiliary function without auxiliary function group

Auxiliary functions that are not assigned to an auxiliary function group are not collected up.

Auxiliary functions of auxiliary function group 1

Auxiliary functions of auxiliary function group 1 are never collected up.

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.

Behavior regarding: M19 (position spindle)

After block search, the last spindle positioning programmed with M19 is always performed, even if other spindle-specific auxiliary functions are programmed between the part program block 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 (traverse command minus/plus)

In this case, the spindle-specific auxiliary functions M3, M4, M5 are not suitable because they might not be output to the PLC until after spindle positioning.

For detailed information on the block search, please refer to:

References:

/FB/ Description of Functions Basic Functions: Mode Group, Channel, Program Operation Mode (K1)

2.12.2 Output suppression of spindle-specific auxiliary functions

Function

For example, with 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.

This requires suppression of automatic output of the spindle-specific auxiliary functions after block search. Output can then be performed manually later by overstoring or by an ASUB.

Parameterization

Suppression of automatic output of the spindle-specific auxiliary functions after block search is performed via:
MD11450 $MN_SEARCH_RUN_MODE, bit 2

Bit 2 = 0
Output of the spindle-specific auxiliary functions is performed in the action blocks

Bit 2 = 1
Output of the auxiliary functions is suppressed in the action blocks

System variables

The spindle-specific auxilary 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 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.

References

More detailed information on ASUB, block search, and action blocks is to be found in:

/FB/ Description of Functions Basic Machine; Mode Group, Channel, Program Operation (K1)
Chapter: Program test

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:
MD11450 $MN_SEARCH_RUN_MODE, Bit 2 = 1

The ASUB is started after block search for N55.

Part program

```
N05 M3 S200                          ; Spindle 1
N10 G4 F3
N15 SPOS = 111                       ; Spindle 1 is positioned at 111 degrees in the ASUB
N20 M2 = 4 S2 = 300                  ; Spindle 2
N25 G4 F3
N30 SPOS[2] = IC(77)                 ; Spindle 2 traverses incrementally through 77 degrees
N55 X10 G0                           ; Target block
N60 G4 F10
N99 M30
```

ASUB

```
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 (On M40 the suitable
                                             ; gear stage is ascertained)
M[SNR] = $P_SEARCH_SDIR[SNR]                 ; Output rotation direction, positioning
                                             ; or axis mode
SNR = SNR+1                                  ; Next spindle
REPEAT AUSG_SPI P=$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.

Control response with REPOS

If the started ASUB is concluded with REPOSA, spindle 1 remains at 111 degrees while spindle 2 is repositioned at 77 degrees.

If a different response is necessary, the program sequence for block search, e.g. "N05 M3 S..." und "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 a position calibration on crossing the zero mark or switch-on of position control. For this reason, the distance programmed from the position as the REPOS position (REPOSA in the ASUB) is assumed to be zero.

Constraints

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 when block search is applied across ranges in which spindles are not programmed with a direction of rotation, positioning, or axis mode, the last spindle mode is retained.
Programming M19, SPOS, and SPOSA is collected as "M-19" (internal M19) in system variable $P_SEARCH_SDIR as an alternative to M3, M4, M5, and M70.

On output of "M-19", the positioning data are read internally from the system variables $P_SEARCH_SPOS and $P_SEACH_SPOSMODE. Both system variables can also be written, 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.

2.13 Querying and displaying of output M help functions

2.13.1 Possibilities of information

Methods of information

Information on the status of M help functions is possible with:

- Display on the operator interface
- Query of system variables in part programs and synchronous actions

2.13.1.1 Status display on the operator interface

The output and acknowledgement status of help functions can be output on the operator interface.

Prerequisites

A prerequisite to execute a function-oriented acknowledgement and display of M help functions is that management of the help functions is to be executed in the PLC proper, i.e. in the user program. For that reason, the PLC programmer has to acknowledge the help functions himself. He must know which help functions are in which group and which he has to acknowledge.

Default

M help functions not managed via PLC are marked as output and transferred to the PLC by the NCK. Functional acknowledgement for these help functions does not exist. All M help functions collected after a block search are additionally displayed so that the operator knows which help functions are output after starting after block search.

PLC activities

With help function groups managed by the PLC itself, the PLC user program must acknowledge all help functions of this group after **transfer** and **at the end of the function**. The PLC programmer must know all help functions of these groups.

Miscellaneous

Only the M help functions are **group-specifically** displayed. Additionally, block-by-block display is still possible as before. Up to 15 groups can be displayed whereby per group **only the last M function of a group**—either collected or output to the PLC—is displayed. The M functions are displayed in various display types according to their status:

States and their display

Status	Display type
The **last** help function of the group is displayed	
Help function collected	Inverted in yellow typeface
Help function output from NCK to PLC	Inverted
Help function transferred from NCK to PLC and transfer acknowledgement executed	Black typeface on gray background
Help function managed by PLC and taken over directly by PLC	Black typeface on gray background
Help function managed by PLC and function acknowledgement executed	Black typeface on gray background

Display updating

The display is organized in such a way that first of all the collected help functions are displayed before the PLC-managed and the NCK-managed functions. A collected help function remains marked as collected until transferred from the NCK to the PLC. PLC-managed help functions remain until replaced by another help function. In the case of RESET, only the collected and NCK-managed functions are deleted.

2.13.1.2 Programming status query

System variables

System variables are available for status queries of group-specific M help functions.

The M help functions can be queried group-specifically in the part program and via synchronized actions through the following variables. The prerequisites and conditions described under “Status display on the operator interface” do apply.

- Value of the collected and output M help functions of the n+1th group:
  INT $AC_AUXFU_M_VALUE[n]
- Address extension of the collected or output M help function of the n+1th group:
  INT $AC_AUXFU_M_EXT[n]
- Display status of the M help function of the n+1th group:
  INT $AC_AUXFU_M_STATE[n]

  0: Help function not present

  1: M help function collected per search run

  2: M help function transferred to the PLC

  3: M help function transferred to the PLC and transfer acknowledgment executed.

  4: M help function managed by the PLC and taken over by the PLC.

  5: M help function managed by the PLC and function acknowledgement executed.

Example

All M help functions of the 1st group are to be saved in the sequence of their output.

id=1 every $AC_AUXFU_M_STATE[0] == 2 do $AC_FIFO[0,0] = $AC_AUXFU_M_VALUE[0]

References

For further information on system variables, please see /PGA1/ List of System Variables.

Supplementary Conditions

3

3.1 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.

3.2 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 one OB40 cycle (quick acknowledgment)
- 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 always ends with an exact stop, even if continuous-path mode is active.

Synchronized actions

With output auxiliary functions from synchronized actions, the parameterized output behavior is ignored except for the following parameters:

- Bit0: Output duration one OB1 cycle (normal acknowledgment)
- Bit1: Output duration one OB40 cycle (quick acknowledgment)

Auxiliary functions: M17 or M2/M30 (end of subroutine)

Only one auxiliary function in a 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.

Overlaying parameterized output behavior

The parmeterized output behavior of auxiliary functions: M17 or M2/M30 is overlaid by the output behavior defined in the following machine data:
MD20800 $MC_SPF_END_TO_VDI, Bit 0 (end of subroutine/stop to PLC)

Bit	Value	Meaning
0	0	The auxiliary functions M17 or M2/M30 (end of subroutine) are not output to the PLC. Continuous-path mode is not interrupted at the end of the subroutine
	1	The auxiliary functions M17 or M2/M30 (end of subroutine) are output to the PLC.

Auxiliary function: M1 (conditional stop)

Overlaying parameterized output behavior

The parmeterized output behavior of auxiliary function: M1 is overlaid by the output behavior defined in the following machine data:
MD20800 $MC_SPF_END_TO_VDI, Bit 1 (end of subroutine/stop to PLC)

Bit	Value	Meaning
1	0	The auxiliary function M01 (conditional stop) is always output to the PLC. Quick acknowledgment in not active because M01 is permanently assigned to auxiliary function group 1 and is therefore always output at the end of the block.
	1	Auxiliary function M01 (conditional stop) is always output to the PLC if function: "Programmed stop" is activated via the HMI user interface. In the case of quick acknowledgment, the M1 is output to the PLC during the movement. While the function is not active, this does not interrupt continuous-path mode.

Part program blocks without traverse movement

In a part program block without a traverse movement, all auxiliary functions are output in a block immediately, irrespective of their parameterized output behavior.

Examples

4

4.1 Defining auxiliary functions

Task

Parameterization 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 to auxiliary function group 2 by default.
- All S and S1 values of spindle 1 (master spindle) are assigned to auxiliary function group 3 by default.

- The gear stage last programmed is to be output after block search. For this purpose, the following auxiliary functions are assigned to auxiliary function group 9:
  - M40, M41, M42, M43, M44, M45
  - M1=40, M1=41, M1=42, M1=43, M1=44, M1=45
- Auxiliary functions M3, M4, M5, M70 and M1=3, M1=4, M1=5, M1=70 (auxiliary function group 2) and the S and S1 values (auxiliary function group 3) are to have the following output behavior:
  - Output duration one OB40 cycle (quick acknowledgment)
  - Output prior to motion
- The auxiliary functions for gear change M40 to M45 and M1=40 to M1=45 (auxiliary function group 9) are to 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. For this purpose, the following auxiliary functions are assigned to auxiliary function group 10:
  - M2=3, M2=4, M2=5, M2=70
- All S2 values are assigned to auxiliary function group 11.
- Auxiliary functions M2=3, M2=4, M2=5, M2=70 (auxiliary function group 10) and the S2 values (auxiliary function group 11) are to 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:
  - 12. auxiliary function group: M50, M51
  - 13. auxiliary function group: M52, M53
- Auxiliary functions M50, M51 (auxiliary function group 12) and M52, M53 (auxiliary function group 13) are to 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 part program.

Programming	Remarks
$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	

Programming	**Remarks**
$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	

Data Lists

5

5.1 Machine data

5.1.1 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
11100	AUXFU_MAXNUM_GROUP_ASSIGN	Maximum number of user-defined auxiliary functions per channel
11110	AUXFU_GROUP_SPEC[n],	Group-specific output behavior

5.1.2 Channel-specific 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	Subprogram end to PLC
22000	AUXFU_ASSIGN_GROUP	Auxiliary function group
22010	AUXFU_ASSIGN_TYPE	Type of auxiliary function
22020	AUXFU_ASSIGN_EXTENSION	Auxiliary function extension
22030	AUXFU_ASSIGN_VALUE	Auxiliary function value
22035	AUXFU_ASSIGN_SPEC	Output specification (user-def. AuxFu)
22040	AUXFU_PREDEF_GROUP	Predefined auxiliary function groups
22050	AUXFU_PREDEF_TYPE	Predefined auxiliary function type
22060	AUXFU_PREDEF_EXTENSION	Predefined auxiliary function extension
22070	AUXFU_PREDEF_VALUE	Predefined auxiliary function value
22080	AUXFU_PREDEF_SPEC	Output specification (predefined auxiliary function)
22100	AUXFU_QUICK_BLOCKCHANGE	Block change without delay
22110	AUXFU_H_TYPE_INT	Type of H auxiliary functions
22200	AUXFU_M_SYNC_TYPE	Output timing for M functions

Number	Identifier: $MC_	Description
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 for H functions
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions
22250	AUXFU_D_SYNC_TYPE	Output timing of D functions
22252	AUXFU_DL_SYNC_TYPE	Output timing of DL functions
22254	AUXFU_ASSOC_M0_VALUE	Additional M function for program stop
22256	AUXFU_ASSOC_M1_VALUE	Additional M function for conditional stop

5.2 Signals

5.2.1 Signals to channel

DB number	Byte.Bit	Description
21, ...	30.5	Activate associated M01

5.2.2 Signals from channel

DB number	Byte.Bit	Description
21, ...	58.0 – 58.4	M function 1 - 5 change
21, ...	59.0 – 59.4	M function 1 - 5 not included in list
21, ...	60.0 - 60.2	S function 1 - 3 change
21, ...	60.4 - 60.6	S function 1 - 3 quick
21, ...	61.0 - 61.2	T function 1 - 3 change
21, ...	61.4 - 61.6	T function 1 - 3 quick
21, ...	62.0 - 62.2	D function 1 - 3 change
21, ...	62.4 - 62.6	D function 1 - 3 quick
21, ...	63.0	E function 1 change
21, ...	63.4	E function 1 quick
21, ...	64.0 - 64.2	H function 1 - 3 change
21, ...	64.4 - 64.6	H function 1 - 3 quick
21, ...	65.0 - 65.5	F function 1 - 6 change

DB number	Byte.Bit	Description
21, ...	66.0 - 66.4	M function 1 - 5 quick
21, ...	67.0 - 67.5	F function 1 - 6 quick
21, ...	68 - 69	Extended address of M function 1 (binary)
21, ...	70 - 73	M function 1 (integer)
21, ...	74 - 75	Extended address of M function 2 (binary)
21, ...	76 - 79	M function 2 (integer)
21, ...	80 - 81	Extended address of M function 3 (binary)
21, ...	82 - 85	M function 3 (integer)
21, ...	86 - 87	Extended address of M function 4 (binary)
21, ...	88 - 91	M function 4 (integer)
21, ...	92 - 93	Extended address of M function 5 (binary)
21, ...	94 - 97	M function 5 (integer)
21, ...	98 - 99	Extended address of S function 1 (binary)
21, ...	100 - 103	S function 1 (real)
21, ...	104 - 105	Extended address of S function 2 (binary)
21, ...	106 - 109	S function 2 (real)
21, ...	110 - 111	Extended address of S function 3 (binary)
21, ...	112 - 115	S function 3 (real)
21, ...	116 - 117	Extended address of T function 1 (binary)
21, ...	118 - 119	T function 1 (integer)
21, ...	120 - 121	Extended address of T function 2 (binary)
21, ...	122 - 123	T function 2 (integer)
21, ...	124 - 125	Extended address of T function 3 (binary)
21, ...	126 - 127	T function 3 (integer)
21, ...	128	Extended address of D function 1 (binary)
21, ...	129	D function 1 (binary)
21, ...	130	Extended address of D function 2 (binary)
21, ...	131	D function 2 (binary)
21, ...	132	Extended address of D function 3 (binary)
21, ...	133	D function 3 (binary)
21, ...	134	Extended address of DL function (binary)
21, ...	136	DL function (real)
21, ...	140 - 141	Extended address of H function 1 (binary)
21, ...	142 - 145	H function 1 (real format)
21, ...	146 - 147	Extended address of H function 2 (binary)
21, ...	148 - 151	H function 2 (real or double integer)
21, ...	152 - 153	Extended address of H function 3 (binary)
21, ...	154 - 157	H function 3 (real format)
21, ...	158 - 159	Extended address of F function 1 (binary)
21, ...	160 - 163	F function 1 (real format)
21, ...	164 - 165	Extended address of F function 2 (binary)
21, ...	166 - 169	F function 2 (real format)

DB number	Byte.Bit	Description
21, ...	170 - 171	Extended address of F function 3 (binary)
21, ...	172 - 175	F function 3 (real format)
21, ...	176 - 177	Extended address of F function 4 (binary)
21, ...	178 - 181	F function 4 (real format)
21, ...	182 - 183	Extended address of F function 5 (binary)
21, ...	184 - 187	F function 5 (real format)
21, ...	188 - 189	Extended address of F function 6 (binary)
21, ...	190 - 193	F function 6 (real format)
21, ...	194	Dynamic M function: M00 - M07
21, ...	195	Dynamic M function: M08 - M15
21, ...	196	Dynamic M function: M16 - M23
21, ...	197	Dynamic M function: M24 - M31
21, ...	198	Dynamic M function: M32 - M39
21, ...	199	Dynamic M function: M40 - M47
21, ...	200	Dynamic M function: M48 - M55
21, ...	201	Dynamic M function: M56 - M63
21, ...	202	Dynamic M function: M64 - M71
21, ...	203	Dynamic M function: M72 - M79
21, ...	204	Dynamic M function: M80 - M87
21, ...	205	Dynamic M function: M88 - M95
21, ...	206.0 – 206.3	Dynamic M function: M96 - M99
21, ...	318.5	Associated M00/M01 active

5.2.3 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	78 - 81	F function for positioning axis (real)

5.2.4 Signals from axis/spindle

DB number	Byte.Bit	Description
31, ...	86 - 87	M function for spindle (binary)
31, ...	88 - 91	S function for spindle (real)

Index

$

$P_SEARCH_S, 2-31
$P_SEARCH_SDIR, 2-31
$P_SEARCH_SGEAR, 2-31
$P_SEARCH_SPOS, 2-31
$P_SEARCH_SPOSMODE, 2-31

A

Address extension, 2-23
Area
  Output duration, 2-22
  Output relative to motion, 2-22
Areas of the output behavior, 2-21
Auxiliary function output
  Block search, 2-29
Auxiliary function without auxiliary function group, 2-29
Auxiliary functions of auxiliary function group 1, 2-29

B

Behavior regarding
  M19, 2-30

C

Collected direction of rotation, 2-33
Collected S values, 2-33
Constraints
  Auxiliary function: M1, 3-3
  Auxiliary functions: M17 or M2 / M30, 3-2
  DL (additive tool offset), 3-1
  Maximum number of auxiliary functions per part program block, 3-1
  Part program blocks without traverse movement, 3-3
  Spindle replacement, 3-1
  Synchronized actions, 3-2
  Tool management, 3-1
Constraints, 2-25, 2-33
  Thread cutting, 3-2
Continuous-path mode, 2-25

D

D functions, 1-6
Definition of an auxiliary function, 1-2
DL functions, 1-7

E

Extension of predefined auxiliary functions, 2-12

F

F functions, 1-7
FA functions, 1-8
Fast acknowledgment, 2-6, 2-25

G

Group assignment, 2-14
Grouping together auxiliary functions, 2-4

M

M functions, 1-3
M98, 2-28
Maximum number, 2-12
MD10713, 2-27
MD10715, 2-28
MD11100, 2-12
MD11110, 2-22
MD11450, 2-30
MD20090, 3-1
MD20124, 3-1
MD20800, 3-2, 3-3
MD22000, 2-14, 2-16, 2-17, 2-18
MD22010, 2-14, 2-16, 2-17, 2-18

MD22020, 2-14, 2-16, 2-17, 2-18
MD22030, 2-14, 2-16, 2-17, 2-18
MD22035, 2-16, 2-17, 2-18, 2-22
MD22040, 2-1, 2-3
MD22050, 2-1, 2-3
MD22060, 2-1, 2-3
MD22070, 2-1, 2-3
MD22080, 2-1, 2-5, 2-6, 2-8, 2-22
MD22100, 2-25
MD22200, 2-18, 2-22
MD22210, 2-18, 2-22
MD22220, 2-18, 2-22
MD22230, 2-18, 2-22
MD22240, 2-18, 2-22
MD22250, 2-18, 2-22
MD22252, 2-18, 2-22
MD22254, 2-26, 2-27
MD22256, 2-26
MD22560, 3-1

N

Normal acknowledgment, 2-5

O

Output after motion, 2-8
Output behavior, 2-19
Output duration one OB1 cycle, 2-5
Output duration one OB40 cycle, 2-25
Output during motion, 2-7, 2-25
Output prior to motion, 2-7
Overstorage of auxiliary functions, 2-29

P

Parameterized output behavior, 2-8
Parameters
  Address extension, 2-4, 2-15
  Type, 2-4, 2-15
  Value, 2-4, 2-15
Part program blocks without path motion, 2-22
Predefined auxiliary functions, 1-1, 2-2
  Definition, 2-1
Priority sequence, 2-22

S

S functions, 1-4
Symbolic addressing, 2-24
Syntax, 2-23
System function, 2-13
System variables, 2-31

T

T functions, 1-5
Type of auxiliary function, 2-18

U

User-defined auxiliary functions, 1-1
User-specific auxiliary functions, 1-2, 2-12

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Mode Group, Channel, Program Operation, Reset Response (K1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 Mode group 2-1
2.1.1 Mode group Stop 2-4
2.1.2 Mode group RESET 2-4

2.2 Mode groups 2-5
2.2.1 Monitoring functions and interlocks of the individual modes 2-11
2.2.2 Mode change 2-12

2.3 Channel 2-13
2.3.1 Global start disable for channel 2-16

2.4 Program test 2-17
2.4.1 Program execution without setpoint outputs 2-18
2.4.2 Program execution in singleblock mode 2-20
2.4.3 Program execution with dry run feedrate 2-22
2.4.4 Skip part-program blocks 2-23

2.5 Block search 2-25
2.5.1 Sequence for block search of Type 1, 2 and 4 2-26
2.5.2 Automatic start of an ASUB after block search 2-31
2.5.3 Cascaded block search 2-32
2.5.4 Examples of block search with calculation 2-33

2.6 Block search Type 5 SERUPRO 2-38
2.6.1 REPOS 2-44
2.6.1.1 Continue machining after SERUPRO search target found 2-44
2.6.1.2 Repositioning on contour with controlled REPOS 2-54
2.6.2 Acceleration measures via MD 2-56
2.6.3 SERUPRO ASUP 2-57
2.6.4 Selfacting SERUPRO 2-60
2.6.5 Inhibit specific part of the program in the part program for SERUPRO 2-62
2.6.6 Special features in the part-program target block 2-66
2.6.6.1 STOPRE in the part-program target block 2-66
2.6.6.2 SPOS in target block 2-67
2.6.7 Behavior during POWER ON, mode change and RESET 2-67
2.6.8 Special features of functions supported during SERUPRO 2-68
2.6.8.1 Traversing to fixed stop 2-68
2.6.8.2 Force Control 2-69
2.6.8.3 Synchronous spindle 2-69
2.6.8.4 Couplings and master-slave 2-70
2.6.8.5 Axis functions 2-73
2.6.8.6 Gear stage change 2-75
2.6.8.7 Superimposed motion 2-75
2.6.8.8 REPOS offset in the interface 2-76
2.6.8.9 Making the initial settings more flexible 2-76

2.6.9 System variables and variables for SERUPRO sequence ... 2-77
2.6.10 Restrictions ... 2-79

2.7 Program operation mode ... 2-79
2.7.1 Initial settings ... 2-80
2.7.2 Selection and start of part program or part-program block ... 2-80
2.7.3 Part-program interruption ... 2-82
2.7.4 RESET command ... 2-83
2.7.5 Program status ... 2-84
2.7.6 Channel status ... 2-86
2.7.7 Responses to operator or program actions ... 2-87
2.7.8 Part-Program Start ... 2-88
2.7.9 Example of timing diagram for a program run ... 2-89
2.7.10 Program section repetitions ... 2-89
2.7.11 Eventdriven program calls ... 2-91
2.7.12 Control and effect on stop events ... 2-100
2.7.13 Asynchronous Subroutines (ASUBs), Interrupt Routines ... 2-102
2.7.14 Calling the ASUB outside program operation ... 2-107
2.7.15 Userdefined system ASUBs ... 2-111

2.8 Single block ... 2-115
2.8.1 Decoding single block SBL2 with implicit preprocessing stop ... 2-116
2.8.2 Single-block stop: Suppression using SBLOF ... 2-116
2.8.3 Single-block stop: inhibit according to situation ... 2-119
2.8.4 Single-block behavior in mode group with type A/B ... 2-120

2.9 Program control ... 2-121
2.9.1 Function selection (via operator interface or PLC) ... 2-121
2.9.2 Activation of skip levels ... 2-122
2.9.3 Adapting the size of the interpolation buffer ... 2-123
2.9.4 Program display modes ... 2-125
2.9.5 Execution from external source (buffer size and number) ... 2-131
2.9.6 Execution from external subroutines ... 2-132

2.10 System settings for power-up, RESET/part-program end and part-program start ... 2-135

2.11 Subroutine call with M, T, and D functions ... 2-144
2.11.1 Replacement of auxiliary functions with subroutines ... 2-144
2.11.2 M function replacement ... 2-147
2.11.3 Replacement of tool programming ... 2-148
2.11.3.1 T and D function replacement ... 2-148
2.11.3.2 M function replacement for tool change ... 2-150
2.11.3.3 Example of M/T function replacement for tool change ... 2-152
2.11.4 Parameter transfer to the replacement subroutine ... 2-153
2.11.5 Properties of replacement subroutines ... 2-157

2.12 Program runtime/workpiece counter ... 2-158
2.12.1 Function ... 2-158
2.12.2 Program runtime ... 2-158
2.12.3 Workpiece counter ... 2-160

3 Supplementary Conditions ... 3-1

3.1 Supplementary conditions ... 3-1

4 Examples ... 4-1

4.1 Examples ... 4-1

5 Data Lists ... 5-1

5.1 Machine data ... 5-1
5.1.1 General machine data ... 5-1

5.1.1.1 HMIspecific machine data ... 5-1
5.1.1.2 NC-specific machine data ... 5-1
5.1.2 Channel-specific machine data ... 5-2
5.1.2.1 Basic machine data ... 5-2
5.1.2.2 Block search ... 5-3
5.1.2.3 Reset response ... 5-4
5.1.2.4 Auxiliary function settings ... 5-4
5.1.2.5 Transformation definitions ... 5-5
5.1.2.6 Memory settings ... 5-6
5.1.2.7 Program runtime and workpiece counter ... 5-7
5.1.3 Axis/spindlespecific machine data ... 5-7
5.2 Setting data ... 5-7
5.2.1 Channelspecific setting data ... 5-7
5.3 Signals ... 5-8
5.3.1 Signals to NC ... 5-8
5.3.2 Signals to mode group ... 5-8
5.3.3 Signals from mode group ... 5-8
5.3.4 Signals to channel ... 5-9
5.3.5 Signals from channel ... 5-9
5.3.6 Signals to axis/spindle ... 5-10
5.3.7 Signals from axis/spindle ... 5-10
Index ... Index-1

Tables

Table 2-1 Mode change ... 2-12
Table 2-2 Effect on program status ... 2-85
Table 2-3 Responses to operator or program actions ... 2-87
Table 2-4 Typical program sequence ... 2-88
Table 2-5 Sequence when starting a part program ... 2-92
Table 2-6 Sequence at part program end ... 2-92
Table 2-7 Sequence with operator panel reset ... 2-93
Table 2-8 Table 1–9 Sequence during power-up ... 2-94
Table 2-9 Summary of NCK events, which stop at least for short time ... 2-101
Table 2-10 Program control: Interface signals ... 2-122
Table 2-11 Change system settings by means of MD ... 2-135
Table 2-12 Selection of RESET and powerup response ... 2-138
Table 2-13 Effect of MD20110 $MC_RESET_MODE_MASK Bits 0 to 6 ... 2-139
Table 2-14 Effect of MD20110 $MC_RESET_MODE_MASK Bits 7 to 12 ... 2-139
Table 2-15 Effect of MD20110 $MC_RESET_MODE_MASK Bits 13 to 17 (in SW 6.4 and higher, Bit 16 to Bit 17) ... 2-140
Table 2-16 Effect of MD20112 $MC_START_MODE_MASK Bits 1 to 7 ... 2-141
Table 2-17 Effect of MD20112 $MC_START_MODE_MASK Bits 8 to 12 ... 2-142
Table 2-18 Effect of MD20112 $MC_START_MODE_MASK Bits 13 to 17 ... 2-142

Brief Description

1

1.1 Brief description

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, MDA, 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, MDA, or JOG. This is ensured by the internal mode logic of the NC.

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

For purposes of testing or breaking in a new part program, the following options are available for reducing critical situations while still in the testing phase:

- 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 MDA modes is referred to as program operation. During execution, the program sequence can be controlled by PLC interface signals and commands.

Initial settings can be specified in channelspecific machine data for each channel. 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 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 := Decoding single block
- SLB3 := Stop in cycle

Basic block display

A second socalled 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).

Reset response

Machine data can be used to change the control-system response for particular system settings for functions such as G codes, tool length compensation, transformation, coupled axis groupings, tangential follow-up, and programmable synchronous spindle for certain system settings following:

- Power on (POWER ON)
- Reset/part program end
- Part program start

Subroutine call with M and T functions

For some applications, it can be advantageous to replace M and/or T functions with a subroutine call. This can be used, for example, to call the tool change routine.

Relevant machine data can be used to suitably define and control subroutines with M and/or T functions.

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 NC systems without tool management.

Detailed Description 2

2.1 Mode group

Mode group

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.

Mode group

Definition of a 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.

Note

This description assumes one mode group and one channel.

Functions requiring several channels (e.g., axis replacement) are described in:
References:
/FB2/ Description of Functions, Extended Functions; Mode Groups, Channels, Axis Replacement (K5)

The configuration of a mode group defines which channels are to be included in the group.

Mode group assignment

A mode group is a grouping of one or more channels. Axes and/or spindles are assigned to one channel.

A channel is assigned to a mode group using machine data:
MD10010 $MN_ASSIGN_CHAN_TO_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 channelspecific settings that pertain to the mode group.

Channelspecific 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 Geo axis replacement command can be programmed to declare a channel axis as a geometry axis and define its geometry axis number. Which spindle in the channel will be the master spindle is defined using SETMS.

With SW 4.4 and higher, it is possible to configure each axis existing in the channel as a spindle. The number of axes per channel depends on the control version. In order to optimize the performance utilization, the available channel and axis configurations are limited depending on the hardware.

With SINUMERIK 840D, 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 more information about other axis configurations such as axis containers, link axes, reciprocating axes, main run axes, rotary axes, linear axes, master axes, and slave axes and the various versions, please refer to:
References:
/FB1/ Description of Functions, Basic Machine; Axes, Coordinate Systems, Frames (K2)
/FB1/ Description of Functions, Basic Machine; Spindles (S1)
/BU/ Ordering Information, Catalog NC 60

Mode groupspecific interface signals

The exchange of mode groupspecific 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 lists all mode groupspecific interface signals:

Modegroup signals (PLC => NCK)	Modegroup signals (NCK → PLC)
Mode group Reset	Mode strobe: JOG, MDA, AUTOMATIC
Mode group Stop axes plus spindles	Machine function strobe: REF, REPOS, TEACH IN
Mode group Stop	All channels (1 to 10, max.) in Reset state
Mode change	Mode group Ready
Operating mode: JOG, MDA, AUTOM.	Active mode: JOG, MDA, AUTOMATIC
Single block: Type A, Type B	Digitizing
Machine function: REF, REPOS, TEACH IN,	Active machine function: REF, REPOS, TEACH IN var. INC, 10000 INC ..... 1 INC

Change in mode group

A change in the configuration 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 are no mode groupspecific machine data.

Channel gaps

Channels to which a mode group is assigned with machine data:
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP
are regarded as activated.

Instead of a mode group number, the number "0" can be assigned to channels.
This has the following results:

- The nonactivated channel does not take up memory space in the control.
- 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 control automatically sets
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP [0] = 1 (mode group 1).

Example configuration:

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

2.1.1 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)

2.1.2 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)

Sequence in the mode group channels

- 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., G function initial setting, are set to the parameterized values.
- All alarms with "Channel reset" criterion are canceled.

Completion of mode group Reset

Once all mode group channels are in Reset state, mode group Reset is completed as follows:

- All alarms with "Mode group reset" 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 = 1 (all channels in Reset state)

  DB11, ... DBX6.3 = 1 (mode group ready)

2.2 Mode groups

Unique mode

The channels of a mode group operate in one mode.

A mode group is either in **AUTOMATIC**, **JOG**, or **MDA** mode.

Several channels of the same mode group **cannot** be in different modes at the same time. If individual channels are assigned to different mode groups, a channel switchover activates the corresponding mode group. This allows mode changes to be initiated implicitly.

Modes in the mode group

The following operating modes are available:

- **AUTOMATIC**

  Automatic processing 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 supplementary conditions so permit.

- **JOG**

  Jogging (manual axis traversal)

  – The axes can be traversed manually with the handwheel or the traversing keys.

  – Channelspecific signals and interlocks are effective in ASUBs and when a movement is activated via IDS synchronized actions. Links are also taken into account.

  – Every channel in the mode group can be active.

- **MDA**

  Manual Data Automatic (the NC blocks are entered via the operator panel)

  – Restricted automatic execution of part programs and sections of part programs (can be only one block or a sequence of blocks).

  – Part program test

  – A maximum of 1 channel per mode group can be active (applies only to TEACH IN)

  – Axes can be manually traversed only 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 per IDS in all modes for the following functions in parallel to the channel:

- Command axis functions
- Spindle functions
- Technology cycles

Selection

The user can select the desired operating mode by means of soft keys on the operator interface.

This selection (AUTOMATIC, MDA, or JOG) is forwarded to interface signals:
DB11, ... DBX4.0 to DBX4.2 (mode strobe)
in the PLC, but is not activated at this point.

Activation and priorities

The required operating mode of the mode group is activated via interface signals:
DB11, ... DBX0.0 to DBX0.2 (operating mode).

If several modes are selected at the same time the priority is as follows:

- JOG (1st priority, high) (DB11, ... DBX0.2)
- MDA (2nd priority, medium) (DB11, ... DBX0.1)
- AUTOMATIC (3rd priority, low) (DB11, ... DBX0.0)

Display

The current operating mode of the mode group is displayed via interface signals:
DB11, ... DBX6.0 to DBX6.2 (active mode):

- JOG (DB11, ... DBX6.2)
- MDA (DB11, ... DBX6.1)
- AUTOMATIC (DB11, ... DBX6.0)

Global machine function for mode group

After mode selection, a machine function can be selected, which is then valid globally for the whole mode group.

Within JOG mode

Within JOG mode, one of the following machine functions can be selected:

- REF (reference point approach)

  References:
  /FB1/ Description of Functions Basic Machine; Reference Point Approach (R1)

- REPOS (repositioning)

  References:
  /BA/ Operator's Guide; Section "Repositioning on contour with controlled REPOS"

Within MDA mode

Within MDA mode, one of the following machine functions can be selected:

- REF (reference point approach)

  References:
  /FB1/ Description of Functions Basic Machine; Reference Point Approach (R1)

- REPOS (repositioning)

  References:
  /BA/, Operator's Guide

- TEACHIN (teach-in of axis positions)

  References:
  /BA/, Operator's Guide

TEACH IN, REPOS or REF

The selection of machine function TEACH IN, REPOS, or REF
from the user interface is stored in:
DB11, ... DBX5.0 to DBX5.2 (machine function strobe).

The chosen machine function TEACH IN, REPOS, or REF is activated in:
DB11, ... DBX1.0 to DBX1.2 (machine function).

The active machine function TEACH IN, REPOS, or REF is displayed in interface:
DB11, ... DBX7.0 to DBX7.2 (active machine function).

Operating statuses

The following three channel statuses can occur in each mode:

1. **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.

2. **Channel active**

   A program has been started, and the program is being executed or a reference point approach is in progress.

3. **Channel interrupted**

   The running program or reference point approach has been interrupted.

Functions in the modes

Operating modes are supplemented through userspecific functions. These functions are not related to any particular technology or machine. A subset of the available functions can be selected in each mode, depending on the operating state.

These functions are categorized as follows:

- NCKspecific functions
- Mode groupspecific functions
- Channel-specific functions

The individual functions can be started and/or executed from the three individual channel statuses: channel reset, channel active, or channel interrupted. The channel states and program states can be checked on the operator panel.

Boundary 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 details

JOG in AUTOMATIC mode is permitted if the mode group is in RESET state and the axis is jog-capable.

RESET for the mode group means:

- All channels in RESET state.
- All programs interrupted.
- No DRF channel active.

Jog-capable axis means:

- The axis is **not**:

  – A PLC axis (concurrent Pos axis), i.e., the PLC has fetched the axis via axis replacement (e.g., FC18).

  – A command axis, i.e., the axis was programmed by a synchronous action and the motion has not yet been completed.

  – A rotating spindle, e.g., a spindle rotating despite RESET.

  – An asynchronous reciprocating axis.

Note: The "jog-capable" property is independent of the "JOG in AUTOMATIC" function.

Activation:

The "JOG in AUTOMATIC" function can be activated using machine data MD10735 $MN_JOG_MODE_MASK.

1. Machine data:
   MD10735 $MN_JOG_MODE_MASK (Bit 0)
   must be set before POWER ON.
2. 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 status "RESET" and program status "Aborted" in all mode group channels. The axis in question must also be "Jog-capable". DRF must be deactivated (if not already deactivated).
3. RESET is initiated or the running program is finished with "M30/M2" in all mode group channels that do not have channel status "Reset" and program status "Aborted".
4. The relevant axis is made "Jog-capable" (e.g., axis replacement PLC→NCK).

Note: In most applications, the axes to be traversed are "JOG-capable" and switching to AUTO will also activate "JOG in AUTOMATIC"!

Features of JOG in AUTOMATIC

- 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.
  Remark: This phase can begin with the JOG movement of an axis and end with the end of the JOG movement of *another* axis.
- Axis replacement is not possible for an axis with active JOG movement. (The axis might change mode group). The NCK blocks any axis replacement attempt.
- The PLC user interface indicates "Automatic" mode:

  DB11.DBX6.0=1

  DB11.DBX6.1=0

  DB11.DBX6.2=0

  DB11.DBX7.0=0

  DB11.DBX7.1=0

  DB11.DBX7.2=0
- In "JOG in AUTOMATIC", the PLC user interface displays whether the mode group is in "Mode group RESET". This enables the NCK to now automatically switch internally to JOG, if required.

  DB11.DBX6.4

  DB11.DBX26.4 ; For mode group 2

  DB11.DBX46.4 ; For mode group 3
- In "JOG in AUTOMATIC", the PLC user interface displays whether the NCK has automatically switched to "Internal JOG".

  DB11.DBX6.5

  DB11.DBX26.5 ; For mode group

  DB11.DBX46.5 ; For mode group 3

Boundary 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 motion is 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.

2.2.1 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 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.

2.2.2 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. All channels must have entered a permissible operating mode before an errorfree mode change can be performed.

Possible mode changes

The following table shows possible mode changes for one channel.

Table 2-1 Mode change

		AUTOMATIC		JOG			MDA			
From					AUTO	MDA		JOG without handwheel		AUTO
To		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, ... D0.4 (Mode change disable).

  This suppresses the operating 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.

2.3 Channel

Assignment of programs

Part programs are assigned to channels. 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 channelspecific 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 more information on tool offset, refer to:
  References:
  /FB1/ Description of Functions, Basic Machine; Tool Offset (W1)

- Channelspecific 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 more information on frames, refer to:
  References:
  /FB1/ Description of Functions, Basic Machine; Axes, Coordinate Systems, Frames (K2), Section "External work offset"
- Display of channelspecific 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

The channels can be assigned their own channel name with 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 setpointissuing 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 channelspecific settings can also be made using machine data:

- Reset states or initial programming settings of G-groups via machine data:
  MD20150 $MC_GCODE_RESET_VALUES (reset state of G-groups)
- Auxiliary function groups for compilation and output timing
- 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. Machine data must be used to make changes to the configuration; these changes do not take effect until the next 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.

For more information on link axes and container axes, refer to
References:
/FB2/ Description of Functions, Extended Functions; Several Operator Panel Fronts and NCUs, Distributed Systems (B3)

Interface signals

NCK channel 1 signals lie in DB21 of the user interface, those of channel 2 in DB22. The channel or channels can be monitored and controlled from the PLC or NCK.

Technology in channel

The technology specification for the machine can be related to specific channels using machine data:
MD27800 $MC_TECHNOLOGY_MODE.

Among other functions, this information is used for evaluation in HMI, PLC, and standard cycles.

Siemens supplies standard machine data for milling. If the machine tool is not a milling machine, but some other type, a different data/program block can be loaded by the HMI or PLC depending on the technology mode set in the machine data.

Spindle functions using a PLC

It is possible to control special spindle motions via an axial PLC interface as an alternative to FC18 and to start and stop them using VDI interface signals without executing a part program.

This option is available only if the channel status is in "Interrupted" or "Reset" mode and the program status is in "Interrupted" or "Aborted" mode. This acceptance state will occur in case of Reset and in JOG mode.

A separate spindle start can be set for each spindle. The following spindle functions can be controlled by PLC via interface signals:

- Spindle Stop (corresponds to M5)
- Spindle Start clockwise (corresponds to M3)
- Spindle Start counterclockwise (corresponds to M4)
- Select gear stage
- Spindle positioning (corresponds to M19)

In the case of multichannel operation, the spindle started by the PLC becomes active in the channel that is handling the spindle when the start command is received.

For more information on the special spindle interface, refer to:
References:
/FB1/ Description of Functions, Basic Machine; Spindles (S1)

Autonomous singleaxis operations

It is possible to decouple a particular axis/spindle in the main run from the channel behavior triggered by the NC program run.

The PLC identifies the corresponding axis/spindle from the axial VDI signal:

DB31, ... DBB28.7 (PLC is controlling the axis) = 1 → assume control

DB31, ... DBB28.7 (PLC is controlling the axis) = 0 → relinquish control

The following functions can be controlled from the PLC:

- Cancel axis/spindle sequence (equivalent to delete distancetogo)
- Stop or interrupt axis/spindle
- Resume axis/spindle operation (continue the motion sequence)
- Reset axis/spindle to the initial state

For a detailed description of these stand-alone single axis operations, refer to:
References:
/FB2/ Description of Functions, Extended Functions; Positioning Axes (P2)

For more information on channel-specific signal exchange (PLC → NCK), refer to:
References:
/FB1/ Description of Functions, Basic Machine; PLC Basic Program (P3).

2.3.1 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.

2.4 Program test

Testing part programs

Purpose

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.

2.4.1 Program execution without setpoint outputs

Functionality

In the "Program test" status, a part program is executed without the output of axis or spindle setpoints.

The part program can be started and executed during the active program test function using interface signal:
DB21, ... DBX7.1 (NC Start)
i.e., using auxiliary function outputs, wait times, G-function outputs, etc.
Safety functions such as software limit switches and working area limits remain valid.

No axis motion

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.

Usage and Handling

Usage

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.

This selection sets 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

As feedback that the program test is active, the appropriate field on the operator interface is reversed and interface signal:
DB21, ... DBX33.7 (program test active)
is set.



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.

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.

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.

2.4.2 Program execution in singleblock mode

Functionality

The part program can be started via interface signal:
DB21, ... DBX7.1 (NC Start).

When the "Single block" function is activated, the part program stops executing 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 control.

The program status switches to "Program status stopped".
The channel status remains active.

The next part program block is processed on NC Start.

Usage

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. This is done by triggering NC Start.

Single-block type

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

  With this type of single block, all 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.

Action single block is the initial setting.

For information on selection of single block mode, refer to:
References:
/BA/ Operator's Guide



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 of single block type

An operator input in the Program control area on the HMI specifies the single block type.

Activation

On detection of the "Single Block" key on the machine control panel, the basic PLC program sets interface signal:
DB21, ... DBX0.4
to activate the single block function.

If there is no machine control panel available, HMI Embedded can be used to activate the "Single block" key in the machine area with the same result.

In the case of HMI Advanced without machine control panel, a user display must be configured with a suitable softkey that sets/resets interface signal:
DB21, ... DBX0.4.

Display

Active single block mode is indicated by a reversal in the relevant field in the status line on the operator interface.

As soon as the part program execution has processed a part program block in single-block mode, interface signal:
DB21, ...DBX35.3 (program status interrupted)
is set.

Control options

For certain sequences, continuous program execution can be desirable despite the selection of single-block mode.

Machine data:
MD10702 $MN_IGNORE_SINGLEBLOCK_MASK
specifies which operations are to be processed without interruption.

- Internal ASUPs
- User ASUBs
- Intermediate blocks
- Block search group blocks (action blocks)
- Init blocks
- Subroutines with DISPLOF
- Nonreorganizable blocks
- Nonrepositionable blocks
- Reposition block without travel information
- Tool approach block.

Assignments between operations and bits of the machine data can be found in the machine data description.

2.4.3 Program execution with dry run feedrate

Functionality

The part program can be started via interface signal:
DB21, ... DBX7.1 (NC Start).

If this function is activated, the traversing speeds programmed in conjunction with `G01`, `G02`, `G03`, `G33`, `G34`, and `G35` are replaced by the feedrate value stored in setting:
SD42100 $SC_DRY_RUN_FEED.

The dry run feedrate value also replaces the programmed revolutional feedrate in program blocks with `G95`.

Use



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

Dry run feedrate mode is selected via the operator interface in the "Program control" menu.

This selection sets interface signal:
DB21, ...DBX24.6 (dry run feedrate selected).

This does not activate the function. In addition, the required dry run feedrate value must be entered in the "Setting data/JOG data" menu.

Activation

The function is activated via interface signal:

DB21, ... DBX0.6 (activate dry run feed)

Changing the dry run feedrate

The effect of setting data:
SD42100 $SC_DRY_RUN_FEED
can be controlled using setting data:
SD42101 $SC_DRY_RUN_FEED_MODE.

The following options are available for changing the dry run feedrate:

1. Dry run feedrate is the maximum of the programmed feedrate and setting data SD42101.
2. Dry run feedrate is the minimum of the programmed feedrate and setting data SD42101.
3. Dry run feedrate is setting SD42101, regardless of the programmed feedrate.

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 more information on feedrate control, refer to:
References:
/FB1/ Description of Functions, Basic Machine; Feeds (V1)

Display

Active dry run feedrate mode is indicated by a reversal in the relevant field in the status line on the operator interface.

2.4.4 Skip part-program blocks

Identification

The operating function must be selected, the function must be activated, and the blocks in question must be identified with a slash character.

Functionality

When testing or breaking in new programs, it is useful to be able to disable or skip certain part program blocks during program execution.

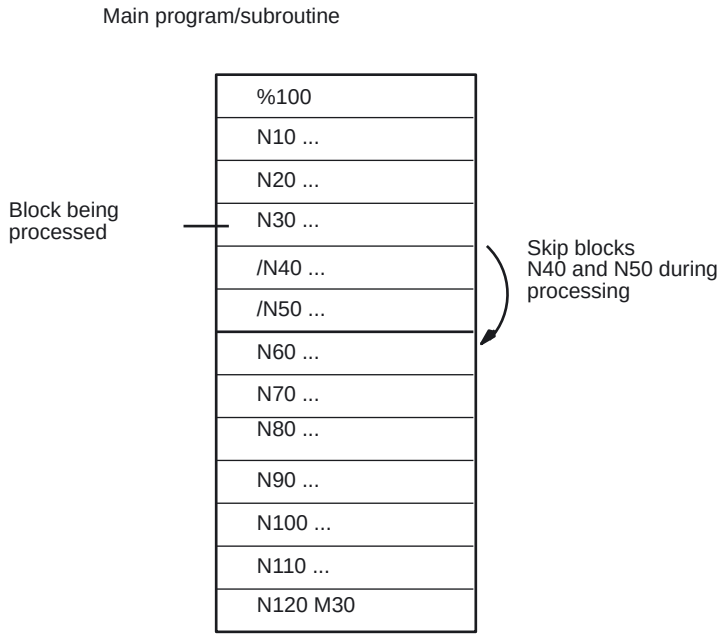


Fig. 2-1 Skipping part program blocks

Selection

The skip function is selected via the operator interface in the "Program control" menu.

This selection sets interface signal:
DB21, ...DBX26.0 (skip block selected).

This does not activate the function. In addition, a slash "/" must be placed before the blocks to be skipped.

Activation

This function is activated via interface signal:

DB21, ... DBX2.0 (activate skip block)

The "Skip part programs" function remains active during block searches.

Display

Activated "Skip block" function is indicated by a reversal in the relevant field on the operator interface.

2.5 Block search

Functionality

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.

Here, every effort is made to achieve to 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 more information on block searches, refer to:
References:
/FB1/ Description of Functions Basic Machine; Auxiliary Function Output to PLC (H2), Section: "Behavior on block search"

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".

2.5.1 Sequence for block search of Type 1, 2 and 4

Time sequence

The block search (Types 1, 2, and 4) proceeds as follows:

1. Activate search via input in HMI Advanced or HMI Embedded
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.

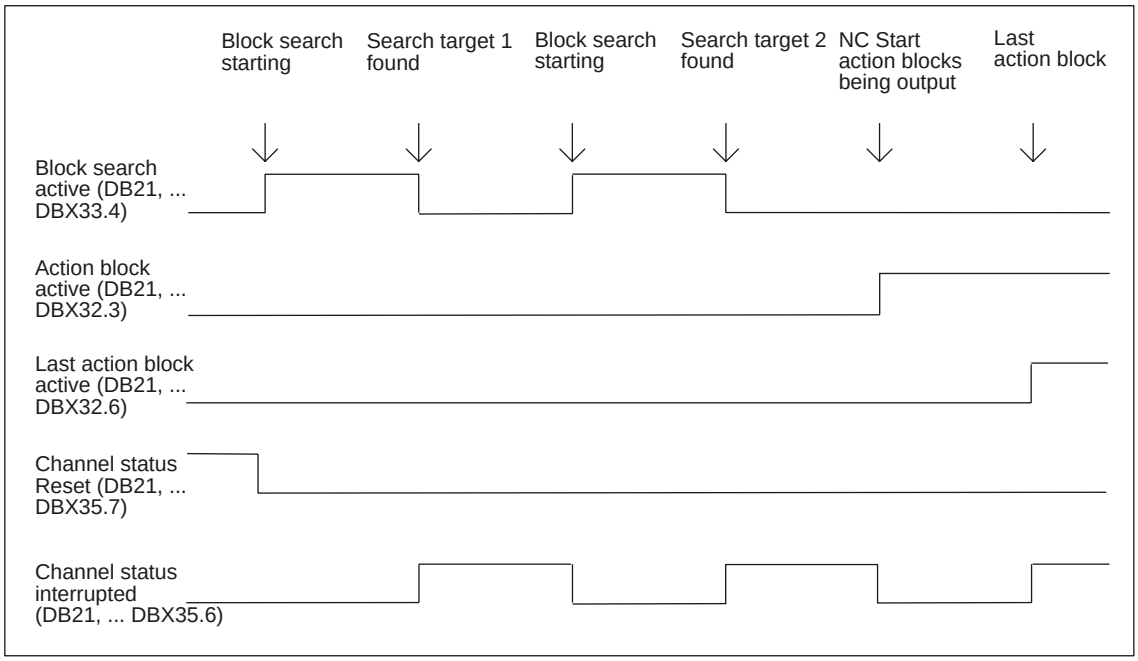


Fig. 2-2 Time sequence of interface signals

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 socalled 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 singleblock mode.

Boundary 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 movement "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 movement can be aborted with an alarm (e.g. circle end point error on G2/G3).

ASUB after and during block search

If an ASUB is started after "Block search with calculation", the positions in the interpreter are synchronized with this operation. $P_EP returns the position at which the axes are actually located to the ASUB.

The collective block search position can be scanned via system variable $AC_RETPOINT. This variable supplies the position as a non modulo-converted position, and an existing rotation count can be lost.

Block search type 2

In "Block search with calculation at contour", the ASUB must be exited with the REPOSA command. The axes are then automatically moved to the collected block search position. $P_EP then also returns to this value.

Block search type 4

After "Block search with calculation at block end point", automatic repositioning is not performed by the REPOS part program command between "Last action block active" and continuation of part program processing via an NC Start. The start point of the approach movement is the current axis position on NC Start; the end point results from the processing of the part program.

No approach motion is generated by the system with "Block search with calculation at block end point". $P_EP thus returns the position at which the axes were positioned by the ASUB or in JOG after the ASUB has been exited.

PLC actions after block search

To enable activation of PLC actions after a block search, e.g., starting of ASUBs or FCs at a defined position, interface signal:

DB21... DB32.6 (last action block active) is used.

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 movement.

By default, alarm 10208 is output at this moment to notify the operator that an NC Start is needed to continue program execution.

Alarm 10208 sets the following interface signals:

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

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

MD11450

Machine data:
MD11450 $MN_SEARCH_RUN_MODE
allows you to set a different response.

If MD11450 = 1, alarm 10208 is not output until the
PLC sets interface signal:
DB21, ... DBX1.6 (PLC action complete).

Auxiliary spindle functions after block search

Control system response and output

Machine data:
MD11450 MN_SEARCH_RUN_MODE
determines the control system response after the end of the block search via the following bits:

Bit 0 = 0:	NC Stop following output of the last action block (default).
Bit 0 = 1:	NC Stop following output of the last action block only if the PLC requests this with: DB21... DB32.6 (last action block active).
Bit 1 = 0:	No automatic ASUB start after block search.
Bit 1 = 1:	Automatic start of user program /_N_CMA_DIR/_N_PROG_EVENT_SPF as an ASUB.
Bit 2 = 0:	Output of spindle auxiliary functions (`M3`, `M4`, `M5`, `M19`, `M70`) in action blocks.
Bit 2 = 1:	Output of the spindle auxiliary functions and spindle setpoints after block search is suppressed (e.g., output in ASUB).
Bit 3 = 0:	Cascaded block search is enabled (i.e., several search targets can be specified).
Bit 3 = 1:	Cascaded block search is disabled.

The programmed spindle values are collected in the block search system variables during block searches irrespective of the configuration.

Spindle programming in ASUBs

The collective spindle programming can be output in an ASUB after

- Block search
- Output of action blocks:
  DB21, ... DB32.6 (last action block active) = 1

In this way it can be ensured that the maximum permissible speed of the active tool is not exceeded, regardless of the block search target.

For further information about the spindle auxiliary functions and a description of the new block search system variables, refer to:
References:
/FB1/ Description of Functions, Basic Machine; Spindles (S1),
Section Spindle auxiliary functions after block search

2.5.2 Automatic start of an ASUB after block search

Activation

Automatic ASUB Start after a block search is configured in existing machine data:
MD11450 $MN_SEARCH_RUN_MODE
with Bit 1 = 1 (TRUE):

Bit 1 = 1:	Automatic start of user program /_N_CMA_DIR/_N_PROG_EVENT_SPF as an ASUB.

When the last action block is activated, user program **/_N_CMA_DIR/_N_PROG_EVENT_SPF** is started as an ASUB.

Start event

The event that has started this user program can be determined by scanning system variable $P_PROG_EVENT. If it is an ASUB Start after a block search, the **value 5** can be scanned using system variable $P_PROG_EVENT.

Note

For further information about the parameter assignment for specific events, see "Event-driven program calls".

Example ASUB activation

Sequence for the automatic start of an ASUB after block search:

1. Start block search (with/without calculation, at contour, at endofblock 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 as an ASUB.
6. The NC stops after activation of the last ASUB block (REPOSA command) and outputs Alarm 10208 depending on Bit 0 of machine data:
   MD11450$MN_SEARCH_RUN_MODE.

2.5.3 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 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

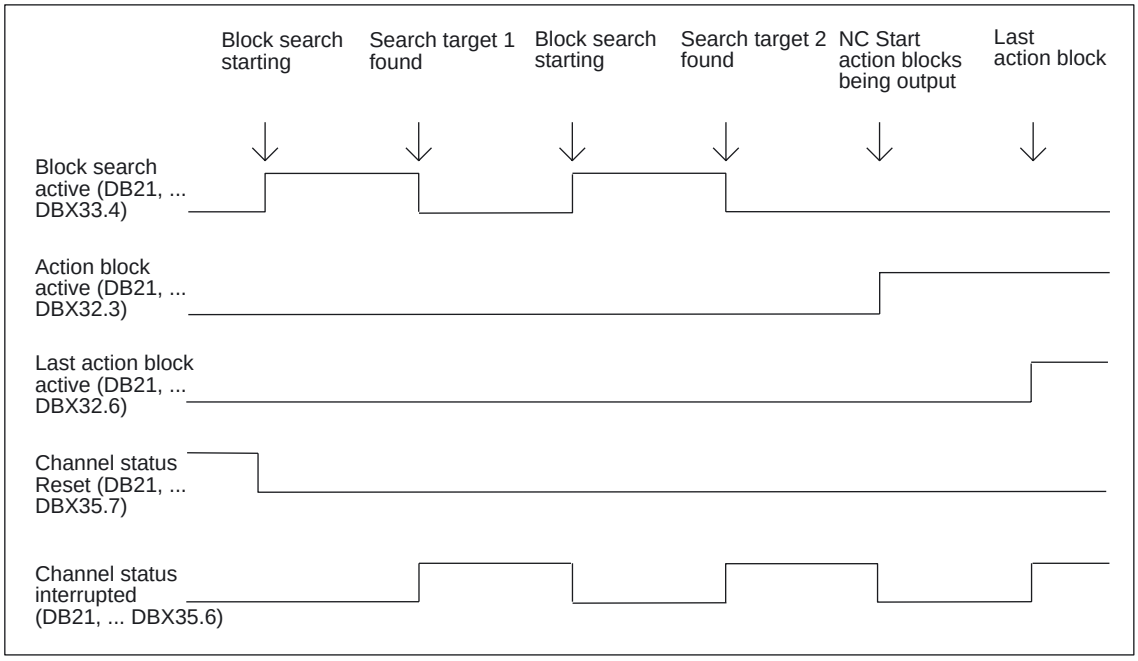


Fig. 2-3 Chronological order of interface signals

2.5.4 Examples of 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_ASUP_START_MASK Bit 0 = 1 (ASUB Start from stopped state)

2. Select ASUB "BLOCK_SEARCH_END" from PLC via FB4.

   References:
   /FB1/ Description of Functions, Basic Machines: Basic PLC Program (P3)

3. Load and select part program "WORKPIECE_1".
4. Search to block end point, block number `N220`.
5. MMC 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.

   References:
   /FB1/ Description of Functions, Basic Machines: Basic PLC Program (P3)

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, VDI interface signal:
   DB21-DB30 DBB318 Bit 0 (ASUP 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, JOGREPOS, overstoring)
10. Continue part program with NC Start.

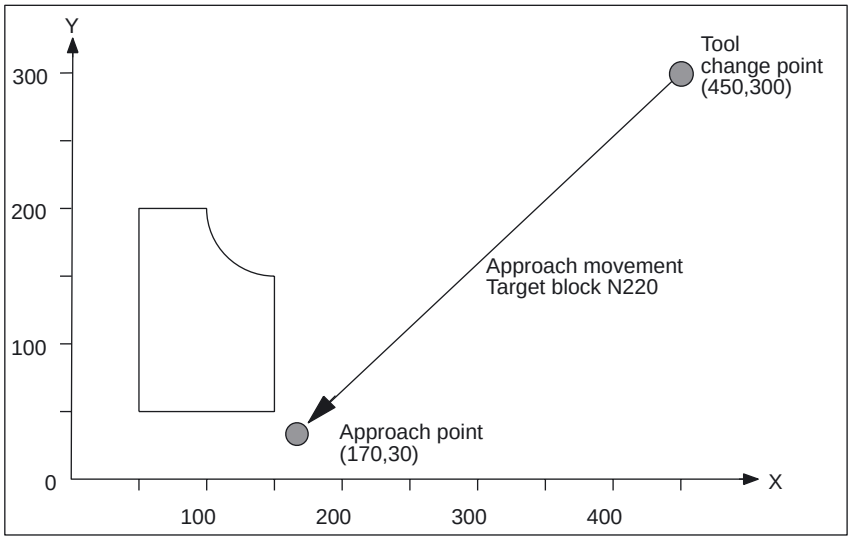


Fig. 2-4 Approach movement for search to block end point (target block N220)

Note

"Search to contour" with target block N220 would generate an approach movement 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

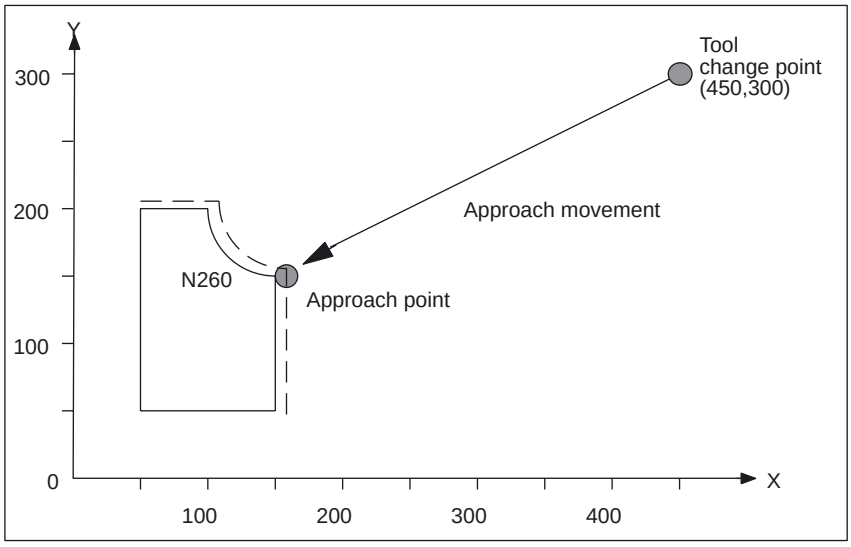


Fig. 2-5 Approach movement for search to contour (target block N260)

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

```
;Main program
...
;Machine contour section 1 with "CUTTER_1" tool
...
N100 G0 G40 X200 Y200                        ; Deselect radius compensation
N110 Z100 D0                                 ; Deselect length compensation
;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
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                                   ; Check-back signal to PLC
N1120 REPOSA                                ; ASUB end
```

2.6 Block search Type 5 SERUPRO

SERUPRO

The "Search via program test" is from now on referred to as **SERUPRO**. This acronym has been derived from **"SE**arch **RU**n by **PRO**gram test".

Function

SERUPRO can be used for a cross-channel block search.

This search permits a block search with calculation of **all** necessary data from the previous history, so as to acquire all previously valid status data for a particular overall NC status. The **PLC** is hereby updated to the **current status**.

The NC is operated in "Program test" mode during this block search so that interactions between a channel and synchronized actions or between several channels, can take place within an NCU.

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 "SelfActing SERUPRO" mode. Only the channel in which a target block has been selected can be started with a block search in SERUPRO mode.

Supported functions

Supported NC functions during **SERUPRO**:

- Gear stage change
- Setpoint and actual value linkages for drives such as "master-slave" as well as "electronic gear" and "axial master value coupling"
- Coupled motion in axis grouping
- Gantry axes

- Tangential follow-up of individual axes
- Superimposed motion interpolation
- Travel to fixed stop
- Synchronous spindle grouping

On reaching the beginning of the target block (see "Time sequence of SERUPRO" below), the user can activate a SERUPRO ASUB.

Special points should be noted during **SERUPRO ASUB** with regard to:

- Reference point approach
- Tool management
- Spindle ramp-up

Other functions after search target was found, such as:

- Continue machining after SERUPRO search target found (REPOS offset)
- Repositioning on contour with controlled REPOS
- Part program expansions for SERUPRO

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. The "Prog. Test Contour" softkey and the search target are operated via the HMI.
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. Numerous operator actions are permitted during this phase:
   - Start, Stop
   - Axis replacement
   - Deletion of distance-to-go
   - Mode change, ASUBs, etc.

     The program and channel statuses of interface signal:
     DB21, ... DBB35
     or system variable:
     $AC_PROG
     is supplied analogous to real operation.

4. The part program command WAITM/WAITE/WAITMC will wait for the partner channels involved.

   This waiting occurs if the partner channels are:

   – In SERUPRO mode

   – In Program test more or are actually running

5. Selection of program test and dry run feedrate is rejected with corresponding Alarm 16935.
6. The NC stops at the beginning of the target block, deselects "Program test" internally, and displays the Stop condition "Search target found" in its block display.
7. As required, the user can start an ASUB that is traversed really. This ASUB is referred to below as the **SERUPRO-ASUB**.
8. The user presses Start:

   The spindles are started. Then, the path axes start a **REPOS operation** that guides them to the block starting point of the target block.

   The REPOS operation is implemented by a system ASUB and can be expanded using the "Editable ASUB" function.

SERUPRO operation

The sequence of operations in items 2. to 6. corresponds to one **SERUPRO operation**.

SERUPRO ASUP

An ASUB that can be called optionally when the target block is reached. It is actually executed.

SERUPRO approach

Approach to the starting point of the target block during a block search in SERUPRO test mode.

Boundary conditions for block search SERUPRO

The SERUPRO function may only be activated in "AUTOMATIC" mode and may only be aborted in program status (channel status RESET).

If in normal mode **only** the PLC starts commonly several channels, then this can be simulated by SERUPRO in each channel.

If 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_SERUPRO_TEST_MASK
allows program testing to be deactivated in the stopped state without the SERUPRO operation being affected. 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

Machine data:
MD10708 $MN_SERUPRO_MASK
can be used to control the SERUPRO behavior as follows:

Stop at M0 during search phase.

Bit 0 = 0	NC is stopped at M0 during the search phase.
Bit 0 = 1	NC is not stopped at M0 during the search phase.

Allow part program command "START" if alarm 16942 is issued.

Bit 1 = 0	The alarm aborts the search phase during part program START.
Bit 1 = 1	The alarm is suppressed. A program uses part program command "Start(j)" in channel i according to the following sequence: The preselected program is started in channel j. Channel j performs a real start with moving axes. A program test can be selected beforehand in channel j by the user. Channel j will now **not** select a search target.
Bit 2	Reserved

SERUPRO not ended simultaneously.

Bit 3 = 0	Switches the function off. All channels that find the search target (also for self-acting SERUPRO) end SERUPRO at the same time.
Bit 3 = 1	SERUPRO ends as soon as the simulation has found the search target. No further synchronization operations take place in the channels started by SERUPRO.

Initial setting for SERUPRO

Machine data:
MD20112 $MC_START_MODE_MASK
is used to define the initial setting of the control for part program start with respect to 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 machine data:
MD22621 $MC_ENABLE_START_MODE_MASK_PRT
to select an initial setting that differs from the normal part program start.

The new setting must be stored in machine data:
MD22620 $MC_START_MODE_MASK_PRT.

The meaning of the bits of MD22620 is identical to those of MD20112.

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.
```

Identification of the active SERUPRO in the interface

DB21, ... DBX318.1

The VDI signal from NCK channel (NCK→PLC)
DB21, ... DBX318.1 (block search via program test is active)
has the following meaning and effect:

The NC runs in the internal "Program test" mode until the target block of the search is activated in the main run and the program status changes to "Stopped".

During this time, interface "Block search via program test is active" is set to 1.

For userdefined ASUB after the SERUPRO operation

Note

If the machine manufacturer decides to start an ASUB after the SERUPRO operation as described in item 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 Item 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 status 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, Bit1 = 1
according to the following sequence:

1. The SERUPRO operation has been performed completely.
2. The user presses "Start".
3. Automatic ASUB start.
4. The NCK stops automatically **before** the `REPOS` part program command and Alarm 10208 "Press NC Start to continue the program" appears.

5. The user presses "Start" again.
6. The NCK executes the REPOS movement 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.

2.6.1 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

2.6.1.1 Continue machining 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.

However, with a SERUPRO approach, a program section has to be "recovered".
This is the case when SERUPRO has ended the simulation and is to travel to the target block again. 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 movements 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 VDI 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 axes nor orientation axes, are then excluded from the 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 VDI signals to reposition on the contour. This mode is programmed in the part program and defines the approach behavior. See "Repositioning to contour with controlled REPOS".

The REPOS behavior of individual axes can also be controlled via VDI 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 prevent temporarily and also delayed. VDI signals can be used to subsequently reenable 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°.)

Note:
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 movement.
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. Note: REPOSDELAY is changed from edge to level evaluation.

Delayed approach of axis with REPOS offset

Using the axial, level-triggered axis/spindle VDI signal (PLC→NCK):
DB31, ... DBX10.0 (REPOSDELAY)
the REPOS offset for the axis is only applied with the edge of
DB21, ... DBX31.4 (REPOSMODEEDGE)
at its next programmed instance.

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 VDI signals

The 0/1 edge of the channel-specific VDI-Signal (PLC→NCK):
DB21, ... DBX31.4 (REPOSMODEEDGE)
is used to transfer the level signals from:
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0-2)
and
DB31, ... DBX10.0 (REPOSDELAY)
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 and restarted, and the REPOS offsets are controlled via signals:
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0-2)
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:
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0-2)
and
DB31, ... DBX10.0 (REPOSDELAY).

Note

In the current ASUB, level:
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 signals:
DB21, ... DBX31.4 (REPOSMODEEDGE) = 1
and
DB21, ... DBX319.0 (REPOSMODEEDGEACKN) = 1
and a RESET occurs in this situation, the NCK cancels interface signal:
DB21, ... DBX319.1–319.3 (Repos Path Mode Ackn0-2).

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 a RESET occurs in this situation, the NCK cancels interface signal:
DB21, ... DBX319.0 (REPOSMODEEDGEACKN) = 0
and
DB21, ... DBX319.1–319.3 (Repos Path Mode Ackn0–2).

Controlling SERUPRO approach with VDI 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 VDI signals

Control REPOS with VDI interface signals

REPOS offsets can be positively influenced with the following channelspecific VDI 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 VDI 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 Ackn0-2) 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

A handshake is made using the channel-specific VDI signal:
DB21, ... DBX319.0 (REPOSMODEEDGEACKN)
in which interface signal:
DB21, ... DBX31.4 (REPOSMODEEDGE)
is detected by the NC and is acknowledged to the PLC with DB21, ... DBX319.0.

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:
DB21, ... DBX319.1-319.3 (Repos Path Mode Ackn0 to 2)
and
DB31, ... DBX10.0 (Repos Delay)
with:
DB31, ... DBX70.2 (Repos Delay Ackn)
as follows:

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)):

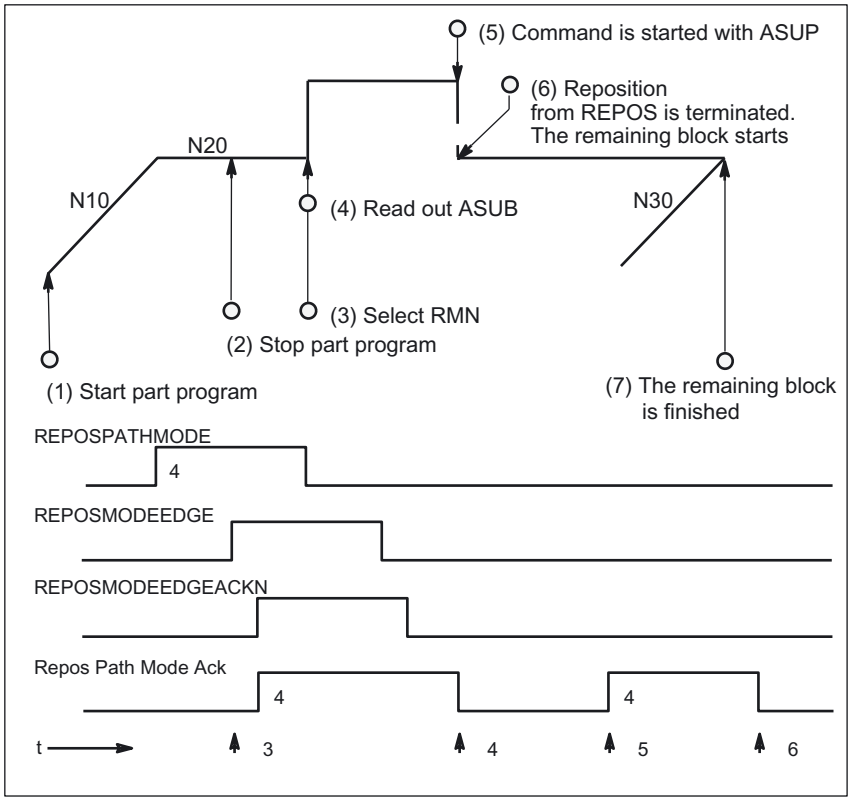


Fig. 2-6 REPOS sequence in part program with timed acknowledgement signals from NCK

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 a VDI 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 (RMN).

Valid REPOS offset

When the SERUPRO operation is complete, the user can read out the REPOS offset via the axis/spindle VDI 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 DB31, ... DBX70.0 is continuously updated.

Displaying the range of validity

The range of validity of the REPOS offset is indicated with interface signal:

DB31, ... DBX70 (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 replacement

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 VDI 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 refers to the target block.

2.6.1.2 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 individual path axes can be controlled by the PLC and, therefore, has priority over the actual RMI, RMB and RME commands in the part program.

RMI: Repositioning to interruption point

RMB: Repositioning to start of block

RME: Repositioning to end-of-block position

RMN: Repositioning to next point on path

Repositioning with RMN

Like RMI, RMB, and RME, the RMN function (REPOS Mode Next) is **redefined** for SERUPRO approach. After an interruption, the repositioning block is not started again completely with RMN, but is merely executed as follows 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.

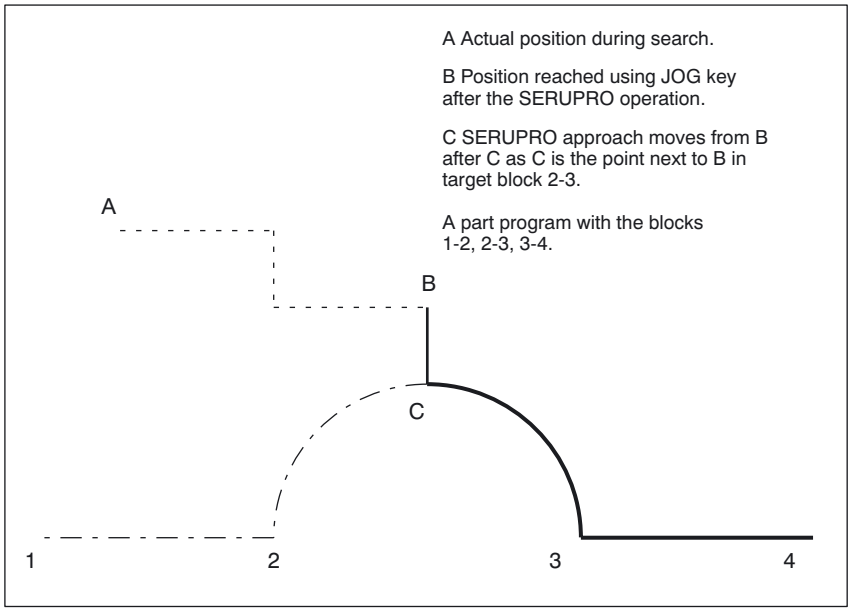


Fig. 2-7 SERUPRO approach with RMN

Application and procedure

SERUPRO approach with RMN offers the opportunity of using the application as 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,

- RMN 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 problem component of the target block.
- B-2, 2-3, 3-4 are always approached with RMI and RMB and the target block thus repeated once completely.

Selecting REPOS mode

Using the 3 bits of the channel-specific VDI signal (PLC→NCK)
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0-2) the respective function (RMB, RMI, RME or RMN) can be selected.

Repositioning point
RMNOTDEF REPOS mode not redefined
RMB Repositioning block start point or last end point
RMI Repositioning interrupt point
RME Repositioning end-of-block point
RMN Repositioning to next path point

DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=0 corresponds to **RMNOTDEF**
DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=1 corresponds to **RMB**
DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=2 corresponds to **RMI**
DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=3 corresponds to **RME**
DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=4 corresponds to **RMN**

With DB21, ... DBX31.0-31.2 (REPOSPATHMODE0-2) = 0, no settings are overwritten and the current program is valid. The interface signal responds to the level of the corresponding mode.

Note

RMN is a general REPOS expansion and is not restricted to SERUPRO. RMI and RMB are identical with respect to SERUPRO.

With DB21, ... DBX31.0–31.2 (REPOSPATHMODE0-2), 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 about programming the repositioning point, see
References: /PGA/, "Path Behavior" 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: RMB	Repositioning to beginning
2: RMI	Repositioning to point of interruption
3: RME	Repositioning to block end point
4: RMN	Repositioning to next geometric path point of interrupted block

2.6.2 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 additional 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.

2.6.3 SERUPRO ASUP

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 are changed.
- Tool data are not changed.
  Exception:
  The tool enabled during the test mode can assume 'active' status. As a result, the spindle may contain the incorrect tool 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 point 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 subroutine

```
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 preselected tool
                                              of the master spindle. Execute tool
                                              change only if tool 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

```
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 rampup 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 about operating mode switchover of spindles, see

References: /FB1/, S1, "Spindles" 2.1 Spindle operating modes

2.6.4 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 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.

2.6.5 Inhibit specific part of the program in the part program for SERUPRO

Programmed interrupt pointer

As a general rule, only the user of the machine knows the mechanical situation that is currently being executed in the program. The user can use a **"programmable interrupt pointer"** to suppress SERUPRO at a particular point in the program where complex mechanical conditions prevail to enable user intervention.

Through suitable manipulation of the interrupt pointer, it is possible, with "search for interrupt point", to resume processing to the search-suppressed location **prior** to the **search-suppressed location**. Search-suppressed areas can also be defined in areas of the part program in which the NCK is not allowed to resume processing.

The last block processed before the search-suppressed target area is used as a search pointer.

Input program section

The **IPTRLOCK** and **IPTRUNLOCK** language commands mark search-suppressed sections of the program. These language commands cannot be used in synchronized actions.

IPTRLOCK	**Start of search-suppressed program section**
IPTRUNLOCK	**End of search-suppressed program section**

IPTRLOCK
Freezes the interrupt pointer at the next "machine function block". This block is referred to below as the **Hold block**. If a program abort occurs after IPTRLOCK, the hold block is used as a search pointer.

Machine function block
Is a single block (SBL1) that can be executed in the main run with a stop after each function block.

IPTRUNLOCK
The interrupt pointer is repositioned at the current block at the time of the interrupt for the subsequent program section.

Note

When a search target is found, the interrupt pointer is

set to the hold block. The block search can be repeated for a new search target with the same hold block.

A search-suppressed area in the part program can be detected using the **$P_IPRTLOCK** variable.

Nesting rules

The following points govern the interaction between language commands IPTRLOCK and IPTRUNLOCK with nesting and the subroutine end.

1. IPTRUNLOCK is activated implicitly at the end of the subprogram in which IPTRLOCK was called.
2. IPTRLOCK has no effect in an existing search-suppressed area.
3. If subprogram1 calls subprogram2 in a search-suppressed area, subprogram2 remains search-suppressed. In particular, IPTRUNLOCK is ignored if programmed in subprogram2.

Examples of nesting with two program levels

Nesting of search-suppressed program sections in **2 program levels**.

```
                                        ;  Interpretation of the blocks in an
                                           illustrative sequence.
                                        ;  Subprogram1 is prepared for the
                                           block search:
N10010 IPTRLOCK()                       ;  Program level 1
N10020 R1 = R1 + 1                      ;
N10030 G4 F1                            ;  Hold block
                                           of the search-suppressed program
                                           section starts
...                                     ;
N10040 Subprogram2                      ;  Interpretation of subprogram2
...                                     ;  Program level 2
N20010 IPTRLOCK ()                      ;  Is ineffective
...                                     ;
N20020 IPTRUNLOCK ()                    ;  Is ineffective
...                                     ;
N20030 RET                              ;
...                                     ;
N10050 IPTRLOCK()                       ;
N10060 R2 = R2 + 2                      ;
N10070 G4 F1                            ;  End of search-suppressed program
                                           section
```

An interruption in the search-suppressed program section of the above program always returns block "N10030 G4 F1".

With implicit IPTRUNLOCK

Nesting of search-suppressed program sections in **two program levels with implicit IPTRUNLOCK**. The implicit IPTRUNLOCK in subprogram1 ends the search-suppressed area.

```
                                        ;  Interpretation of the blocks in an
                                           illustrative sequence.
                                        ;  Subprogram1 is prepared for the
                                           block search:
N10010 IPTRLOCK()                       ;  Program level 1
N10020 R1 = R1 + 1                      ;
N10030 G4 F1                            ;  Hold block
...                                     ;  of the search-suppressed program
                                           section starts
as in previous example                  ;
N20030 RET                              ;
...                                     ;
N10060 R2 = R2 + 2                      ;
N10070 RET                              ;  End of search-suppressed program
                                           section
N100 G4 F1                              ;  Main program is continued
```

An interruption in the search-suppressed program section of the above program always returns block "N10030 G4 F1".
An interruption at N100 then returns N100 in SPARPRI again.

On one program level

Nesting of **IPTRLOCK** and **IPTRUNLOCK** on one program level

```
                                        ;  Interpretation of the blocks in an
                                           illustrative sequence
                                        ;  Subprogram1 is prepared for the
                                           block search:
N10010 IPTRLOCK()                       ;  Program level 1
N10020 R1 = R1 + 1                      ;
N10030 G4 F1                            ;  Hold block
                                           of the search-suppressed program
                                           section starts
...                                     ;
...                                     ;
N10050 IPTRLOCK()                       ;  Is ineffective
...                                     ;
N10060 IPTRUNLOCK()                     ;  Is ineffective
N10070 R2 = R2 + 2                      ;  End of search-suppressed program
```

```
                                                    section
N100 IPTRLOCK()                                 ;   is ineffective
...                                             ;
```

An interruption in the search-suppressed program section of the above program always returns block "N10030 G4 F1" in SPARPI.
An interruption at N100 then returns N100 again.

Automatic interrupt pointer

In certain applications it can be useful to automatically define a prespecified type of coupling as a search-suppressed area. The automatic interrupt pointer function is activated with machine data
MD 22680 $MC_AUTO_IPTR_LOCK.

- Bit 0 = 1: Electronic gear with EGON
- Bit 1 = 1: Axial master value coupling with LEADON

This program section begins with the last executable block **before** activation and ends with deactivation.
The automatic interrupt pointer is not active for couplings, which are activated and deactivated in synchronized actions.

Example: To declare axial master value coupling as search-suppressed:

```
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 the search-suppressed program section (N200 - N500) always supplies N100 to the interrupt pointer.

Caution

Using an overlap of the "programmable interrupt pointer" and

"automatic interrupt pointer" 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.

2.6.6 Special features in the part-program target block

2.6.6.1 STOPRE in the part-program target block

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.

```
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 ...                                    ;
```

An implicit STOPRE before N50 allows the NCK to read and calculate the current work offset correctly.
In a SERUPRO operation to search target N50, repositioning to the implicit STOPRE is performed during the SERUPRO approach and the velocity is determined as F5000 from N40.

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. On 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. On the following commands executions:
   -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.

2.6.6.2 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.

2.6.7 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.

2.6.8 Special features of functions supported during SERUPRO

SERUPRO supports the following NC functions:

- Traversing to fixed stop: FXS and FOC automatically
- Force Control
- Synchronous spindle: Synchronous spindle grouping with COUPON
- Autonomous singleaxis operations: The PLC-controlled axes
- Linkages: Setpoint and actual value couplings can be simulated

  Master/slave for drives. These functions are closely simulated
  by electronic gears. The relevant axial master value coupling restrictions must be considered in each case!
- Axis couplings
  Coupled motion: Axis grouping with TRAILON and TRAILOF,
  Traverse gantry axis couplings
  Tangential control: Tangential followup of individual axes
- Axis functions
  Axis enable,
  Autonomous axis operations,
  Axis transfer
- Gear stage change: During program test (not fully automatically)
- Overlaid movements: Superimposed motion interpolation

For more information about some of these functions, see the following sections.

2.6.8.1 Traversing to fixed stop

FXS

The functionality of FOC is repeated automatically with REPOS and designated **FXS-REPOS** below. Every axis is taken into account and the torque last programmed before the search target is applied.

Furthermore, the meaning of system variable $AA_FXS is redefined for SERUPRO as follows:

- $AA_FXS displays the current status of program simulation.
- $VA_FXS always describes the real machine status.

The two system variables $AA_FXS and $VA_FXS have the same values continuously **outside** the SERUPRO function.

The user can treat FXS and FOC as special commands in a SERUPRO ASUB.

2.6.8.2 Force Control

System variables $AA_FOC, $VA_FOC

The meaning of system variable $AA_FOC is redefined for SERUPRO as follows:

- $AA_FOC represents the current status of program simulation.
- $VA_FOC always describes the real machine status.

The FOCREPOS function behaves analogously to the FXSREPOS function. The functionality of FOC is repeated automatically with REPOS and is designated FOCREPOS. Every axis is taken into account and the torque last programmed before the search target is applied.

Boundary condition

A continuously changing torque characteristic **cannot** be implemented with FOC-REPOS.

Example:

A program moves axis X from 0 to 100 and activates FOC every 20 increments for 10 increments at a time. This torque characteristic is usually generated with nonmodal FOC and cannot, therefore, be traced by FOCREPOS. FOCREPOS will traverse axis X from 0 to 100, with or without FOC, according to the last programming.

Note

For further information about the SERUPRO block search in relation to FXS or FOC, see

References: /FB/, F1, "Traversing to Fixed Stop" General Functions

---

2.6.8.3 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: /FB2/, S3, "Synchronous Spindles"

2.6.8.4 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 under SERUPRO there are only some, i.e., not all, leading axes, 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 point of view of the NCK and are operated with external data cannot be simulated correctly. These axes must not be integrated in couplings.



Caution

In order to be able to simulate couplings correctly, the couplings must have been switched off previously.

This can occur with machine data MD10708 $MA_SERUPRO_MASK.

Specifications for active couplings

The SERUPRO operation simulates axis couplings 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 movements. 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 about the repositioning of axis couplings, see

"Continue machining after SERUPRO search target found".

Master-slave

During the block search, only the link status should be updated without calculating the associated positions of the coupled axis.

A system ASUB can be started automatically after the block search is finished. In this subroutine, the user can control the link status and the associated axis positions subsequently. The information needed can be read from additional block search system variables.

System variable for master/slave

The following system variables are needed for the position offset between the axes to be coupled with the desired link status:

NCK variables	Description
$P_SEARCH_MASLD[X] Slave*	Position offset between slave and master axis when the link is closed.
$P_SEARCH_MASLC[X] Slave*	Current status of a master/slave link was changed during block search.
$AA_MASL_STAT[X] Slave*	Current status of a master/slave link active.

Slave* for slave axis identifier

Note

This block search for the master/slave link is effective only if the position offset between the axes can be determined.

In order to determine the programmed positions, the axes to be linked must be in the same channel at the time of the block search. If this is not the case, the block search is aborted with alarm 15395.

Variables $P_SEARCH_MASLD, $P_SEARCH_MASLC and $AA_MASL_STAT are cleared on MASLON.

For further information about the master/slave link, see:
References: /FB3/, TE3, "Speed/Torque Coupling, 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

```
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
```

The following machine data must be set so that the ASUB can start automatically:
$MN_ASUP_START_MASK = 'H03'
$MN_ASUP_START_PRIO_LEVEL = 100
$MN_SEARCH_RUN_MODE = 'H02'

Coupled axes

The following axis links are compatible with the SERUPRO operation:

- Coupled motion "TRAILON" and TRAILOF"
- Gantry axes
- Tangential control
- The activatable and deactivatable "master/slave" link as long as activation and deactivation is not selective

Note

If the following axes of the master axis are in another channel,

any attempt to accelerate the processing speed by setting

MD22601 $MC_SERUPRO_SPEED_FACTOR = positive

Coupled motion

The motionsynchronous action for coupled motion of an axis grouping with TRAILON and TRAILOF is supported by SERUPRO.

For further information about coupled motion with TRAILON, TRAILOF, see
References: /FB3/, M3, "Axis Couplings and ESR"
/PGA/, "Path Behavior" and "MotionSynchronous Actions"

Gantry axes

Mechanically linked machine axes can be moved without a mechanical offset using the gantry axis function. This operation is simulated correctly with SERUPRO.

For further information about the functionality of gantry axes, refer to
References: /FB3/, G1 "Gantry Axes"

Tangential control

Tangential follow-up of individual axes is supported by SERUPRO.

For further information about tangential control, see:
References: /FB3/, T3, "Tangential Control"

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.

2.6.8.5 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 movement, VDI signal DB31, ... DBX3.7 (program test axis/spindle enable) is then canceled, Alarm 21612: "Channel %1 axis %2 VDI signal ’servo enable’ reset during motion" is activated.

Autonomous axis operations

Autonomous singleaxis 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 singleaxis operations, see:
References: /FB2/, P2, "Positioning axes"

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 currently not taken into account in SERUPRO approach.

The following behavior can be activated for SERUPRO REPOS via machine data $MN_REPOS_MODE_MASK:
The neutral axes are moved as "command axes" in SERUPRO REPOS. The axis interpolates without a path context even it was last programmed as a path axis. In this case the velocity is derived from $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 VDI 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.

2.6.8.6 Gear stage change

Operational sequences

The gear stage change requires physical movements from the NCK in order to select a new gear.
A gear stage change is not required in the SERUPRO operation and is performed 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 about gear stage changes in DryRun, Program test and SERUPRO, see

References: /FB1/, S1, "Spindles" Programming

2.6.8.7 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.

2.6.8.8 REPOS offset in the interface

REPOS offset provided/valid

When the SERUPRO operation is finished, the user can use the OPI to read off the REPOS offset applied via the following REPOS process.

An axis-VDI-interface bit "REPOS offset" is provided. The axis-VDI-interface bit "REPOS offset" corresponds to DB31.DBX70.**BIT0**.

Value 0 No REPOS offset has to be applied to this axis

Value 1 A REPOS offset has to be applied to this axis.

Scope: The axis-VDI-interface bit "REPOS offset" is provided when the SERUPRO operation is finished. The "REPOS offset" is invalid when a SERUPRO ASUB starts or continues. The axis can be moved in JOG mode with a mode change between the SERUPRO end and start and the "REPOS offset" bit is continuously updated (i.e., the user traverses the REPOS offset by hand and the bit changes to 0). In the scope, the axis can also be traversed using FC18, with the "REPOS offset" continuously being updated.

The scope of "REPOS offset" is displayed with the axis-VDI-interface bit "REPOS offset valid" DB31.DBX70.**BIT1**.

DB31.DBX70.BIT1

Value 0 DB31.DBX70.BIT0 has not been validly calculated

Value 1 DB31.DBX70.BIT0 has been validly calculated

2.6.8.9 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 then be 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
```

2.6.9 System variables and variables for SERUPRO sequence

SERUPRO detection

The SERUPRO sequence can be detected using the following system variables:

$P_ISTEST is TRUE (valid also for program test)
$P_SEARCH is set to 5 (search in extended program test)
$AC_ASUP Bit 20 in system ASUB is set after the search target
is found (SERUPRO operation step 8.)
$P_ISTEST AND (5 == $P_SEARCHL) reliably detects SERUPRO.

$AC_SEARCH is **not** supplied by the SERUPRO operation.

Note

$P_SEARCHL is set at the beginning of the SERUPRO operation and

reset on RESET. As a result, $P_SEARCHL continues to be set in the SERUPRO ASUB and in the residual part program and can continue to be evaluated.

In contrast, the $P_ISTEST variable is set only in the SERUPRO operation and is thus suitable for searchspecific adaptation of programs.

Synchronized action

SERUPRO can be scanned in a synchronized action using system variable

$AC_SERUPRO = TRUE.

SERUPRO Updated REPOS acknowledgements can be scanned via:

"Programsensitive system variable"	Description
$AC_REPOS_PATH_MODE	Type of REPOS MODE
$AA_REPOS_DELAY	REPOS suppression is currently active for this axis

$AC_SERUPRO and $P_ISTEST, if SERUPRO is still active in the main run

Note

During interpretation of system variables $P_ISTEST and $AC_SERUPRO, a check is made to determine whether the SERUPRO target block has already been found.

If so, an implicit preprocessing stop is inserted before the two 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.

Additional variable for interpretation during simulation search

In JOG and MDA modes, NC variable "selectedWorkPProg" can be used to select whether the previously selected program or the program to be simulated is to be displayed in the HMI during the simulation, e.g., with SERUPRO. For more information, refer to

References: Lists (Book 2), Section "Variables".

2.6.10 Restrictions

Conditional use

SERUPRO supports the following NC functions subject to certain restrictions:

NCK functionality	Restrictions
Master/slave for drives and SERUPRO	Selective enabling and disabling of the master/slave link with MASLON active
Axis enables and SERUPRO	Real servo enable missing during program testing
Axis replacement and SERUPRO	Axes traversing as path axes before RELEASE are ignored on REPOS

2.7 Program operation mode

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 controlled 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.

2.7.1 Initial settings

Machine data

Conditions for program operation can be set using machine data.

MD settings

Initial settings

Initial settings can be programmed in channelspecific 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 further information, see:
References: /FB1/ H2, "Auxiliary Function Output to PLC"

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.
Machine data MD22510 $MC_GCODE_GROUPS_TO_PLC (G codes that are output to NCK-PLC interface on block change/RESET) can be used to activate output of the G codes to the PLC interface.

A list of G groups with the associated G functions and descriptions can be found in:

References: /PG/ "Programming Guide: Fundamentals"

2.7.2 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 (for further information, see Section on Channel synchronization).

The START command can only be executed in AUTOMATIC and MDA modes. The relevant channel must have status 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.

Relevant enable signal settings:

- 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" must not be present
- No axis or NCK alarm must be active

For descriptions of the individual signals see Chapter 5.

Execution of command

The part program or part program block is automatically processed and interface signals:
DB21, ... DBX35.5 ("Channel status active")
and
DB21, ... DBX35.0 ("Program status running") are set.

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" (see /DA/)
- 10203 "No NC Start permitted for nonreferenced axes"

2.7.3 Part-program interruption

"Interrupted" status

Channel status

The STOP command is executed only if the channel concerned has status DB21, ... D35.5 ("Channel active").

STOP commands

There are various commands that stop the program execution and set the channel status to "Interrupted". These are:

- 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")
- Programming command "M00" or "M01".

For explanations of the interface signals, see the data lists. For explanations of the part program instructions, see:
References: /PG/ "Programming Guide: Fundamentals"

Execution of command

After execution of the STOP command, DB21, ... DBX35.3 ("Program status interrupted") is set. Processing of the interrupted program can continue from the point of interruption with the command START.

The following actions are executed when the STOP command is triggered:

- Part program execution is stopped at the next block limit (with NC stop at block limit, M00/M01 or single block), processing is stopped immediately with the other STOP commands.
- Any auxiliary functions of the current block not yet output by this point are no longer output.
- The axes of each channel are brought to a standstill along a braking ramp and part program execution is then stopped.
- The block indicator stops at the point of interruption.

Possible actions in the interrupt state

The following actions can be executed when the part program has been interrupted (program status stopped, channel interrupted):

- Overstore
  References: /BA/, "Operator's Guide"
- Block search
  References: /BA/, "Operator's Guide"
- Repositioning to the contour (machine function REPOS)
  References: /BA/, "Operator's Guide"
- Oriented tool retraction
  References: /PG/, "Programming Guide: Fundamentals"
- Interrupt routine (see Subsection 2.5.12)
- DRF function, workpiece zero offset
  References: /FB2/ H1, "Manual and Handwheel Travel"
- Starting an interrupted program with IS
  "NC Start" (DB21, ... DBX7.1) or
  via NC instruction START from another channel.

2.7.4 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 commands

The following Reset commands are available:

- Interface signal DB11, ... DBX0.7 ("Mode group reset")
- Interface signal DB21, ... DBX7.7 ("Reset")

For a further explanation of the individual signals, refer to the data lists.

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, 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 that have not yet been output at this point are not 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.

2.7.5 Program status

Interface information

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 modes. In all other modes the program status is aborted or interrupted.

Program statuses

The following program states are available:

- Interface signal DB21, ... DBX35.4 ("Program status aborted")
- Interface signal DB21, ... DBX35.3 ("Program status interrupted")
- Interface signal DB21, ... DBX35.2 ("Program status stopped")
- Interface signal DB21, ... DBX35.1 ("Program status waiting")
- Interface signal DB21, ... DBX35.0 ("Program status running")

For a further explanation of the individual signals, refer to the data lists.

The effect of commands/signals

The program status can be controlled by activating different commands or interface signals. The following table shows the resulting program status when these signals are set (assumption: status before the signal is set -> Program status running).

Table 2-2 Effect on program status

Commands	Program execution statuses				
	Aborted	Interrupted	Stopped	Wait	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 distancetogo"					X
Auxiliary functions output to PLC but not yet acknowledged			X		
Wait instruction in program				X	

2.7.6 Channel status

Interface representation

The current channel status is displayed in the interface. The PLC can then trigger certain responses and interlocks configured by the manufacturer depending on the status at the interface.
The channel status is displayed in all operating modes.

Channel statuses

The following channel statuses are available:

- Interface signal DB21, ... DBX35.7 ("Channel status reset")
- Interface signal DB21, ... DBX35.6 ("Channel status interrupted")
- Interface signal DB21, ... DBX35.5 ("Channel status active")

For a further explanation of the individual signals see Chapter 5.

The effect of commands/signals

The channel status can be modified through the activation of various commands or interface signals. The following table shows the resulting channel status (assumption: status before the signal -> channel status active).
"Channel status active" is achieved when a part program or
part program block is executed or when the axes are traversed in JOG mode.

Commands	Resulting channel status		
	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 distancetogo"			X
Auxiliary functions output to PLC but not yet acknowledged			X
Wait instruction in program			X

2.7.7 Responses to operator or program actions

Status transitions

The following table shows the channel and program statuses that result after certain operator and program actions.

The lefthand side of the table lists the channel and program statuses as well as the operating modes from which the starting situation must be 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 2-3 Responses to operator or program actions

Situation	Channel status			Program status					Active mode			Operator or program action (situation after the action)
	R	V	A	N	V	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

Operating modes
A --> aborted
M --> aborted
J --> aborted

2.7.8 Part-program start

Start handling

Table 2-4 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 set Reference point approached in all axes	
5			Program execution
6	M02/M30/Reset	None	End of program

2.7.9 Example of timing diagram for a program run

Signal sequences

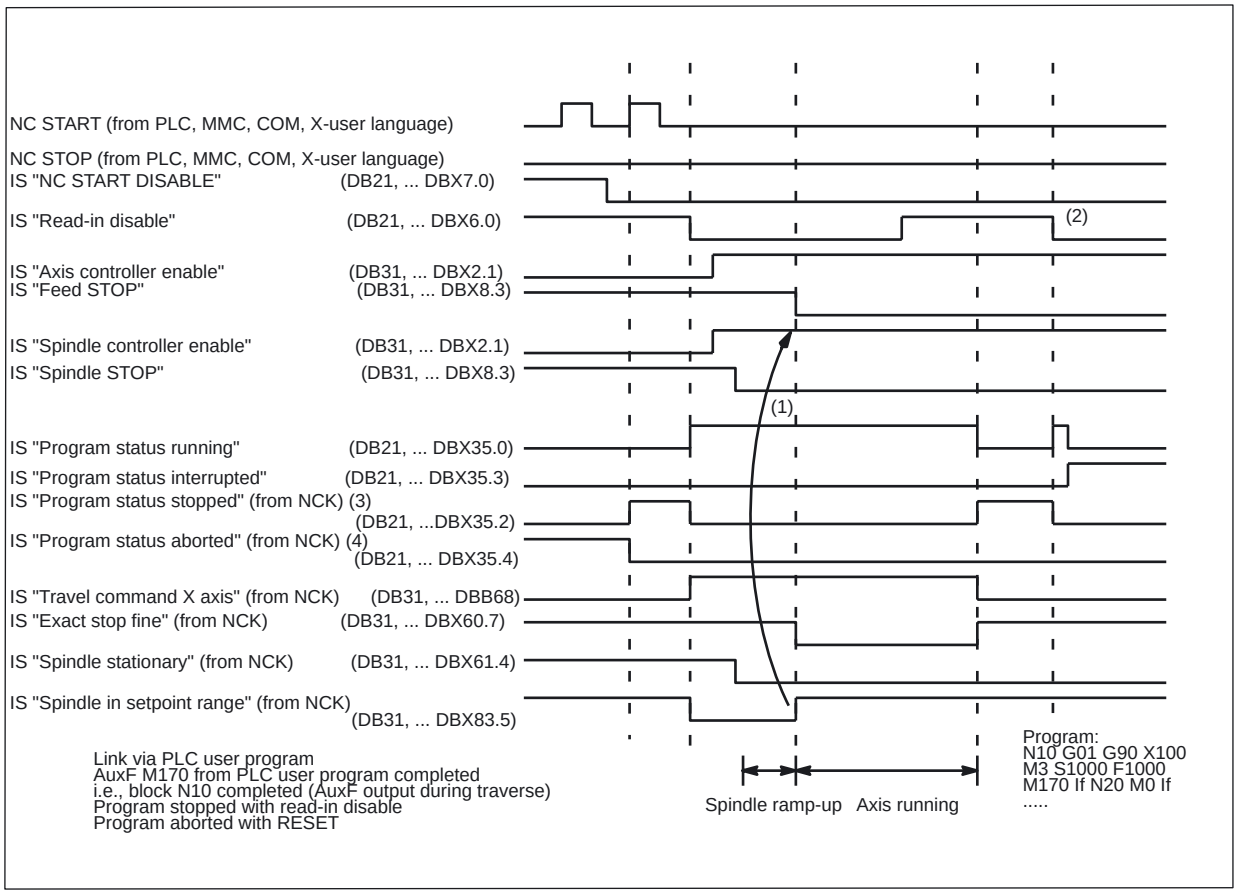


Fig. 2-8 Examples of signals during a program run

2.7.10 Program section repetitions

Use

Basic information

Program sections, which have already been programmed, are frequently needed again. Instead of using subprograms, you can repeat existing program sections in any order.

The advantage compared with copying blocks is that you save memory and can make modifications easily. Changes to positions only need to be made at one point.

Labels

Labels are used within a program to identify a block or a program section.

References: /PG/, Programming Guide: Fundamentals

The options available for repeating part of a program are:

- Repeat block
- Repeat area starting at label
- Repeat an area between two labels
- Repeat an area between a label and the end label

References: /PGA/, Programming Guide Advanced

Activation

Program section repetition is activated by programming. (Labels, Number of repetitions)

Example

Milling: Machine drilling position with different technology sequences

```
N10 CENTER DRILL()                ;  Load center drill
N20 POS_1:                        ;  Drill positions 1
N30 X1 Y1
N40 X2
N50 Y2
N60 X3 Y3
N70 ENDLABEL:                     ;  Res. Name, frequently permissible
N80 POS_2:                        ;  Drill positions 2
N90 X10 Y5
N100 X9 Y-5
N110 X3 Y3
N120 ENDLABEL:
N130 DRILL()                      ;  Change drill and drilling cycle
N140 THREAD(6)                    ;  Load tap M6 and threading cycle
N150 REPEAT POS_1                 ;  Repeat program section once from
                                     POS_1 up to ENDLABEL
N160 DRILL()                      ;  Change drill and drilling cycle
N170 THREAD(8)                    ;  Load tap M8 and threading cycle
N180 REPEAT POS_2                 ;  Repeat program section once from
                                     POS_2 up to ENDLABEL
N190 M30
```

2.7.11 Eventdriven program calls

Application

In the case of certain events, an implied user program is to start. This allows the user to activate the initial settings of functions or carry out initialization routines by part program command.

Event selection

MD 20108

Machine data MD20108 $MC_PROG_EVENT_MASK can be used to specify on a channel-specific basis which of the following events is to enable the user program:

- Bit0 = 1: Part program start
- Bit1 = 1: Part program end
- Bit2 = 1: Operator panel reset
- Bit3 = 1: Powerup (of the NC control)

The user program can be stored under the fixed path name
/_N_CMA_DIR/_N_PROG_EVENT_SPF or

Other program name

A name is specified in MD11620 $MN_PROG_EVENT_NAME.

The system then searches for the user program in the directories:

/_N_CUS_DIR/

/_N_CMA_DIR/

/_N_CST_DIR/

in the specified order. The first program found with the stored name is called when a configured event occurs.

The same protection mechanisms can be enabled as can generally be activated for cylinders (protection levels for writing, reading etc.).

Machine data MD20108 $MC_PROG_EVENT_MASK is ignored during the simulation.

Request which start event

In the user program, the system variable $P_PROG_EVENT can be used to request the event, which enabled the part program.

Event

Part program start

Table 2-5 Sequence when starting a part program

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Channel selection: Reset state Mode selection: AUTO or AUTO and Overstore or MDA or TEACHIN	Defined initial state: Channel in Reset state Mode: AUTO or AUTO and Overstore or MDA or TEACHIN	Select channel and mode Channel: in reset status and mode: as per selection
2	NC Start	None	NCK start
3	MD20112 $MC_START_MODE_MASK	Initialization sequence with evaluation	Initialization sequence with evaluation of MD20112
4	/_N_CMA_DIR/_N_PROG_ EVENT_SPF or name from MD11620	As a subroutine	Implied call of the path name as a subroutine
5		None	Processing of the data part of the main program
6		None	Processing of the program part of the main program

Event

Part program end

Table 2-6 Sequence at part program end

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Channel selection: Active state Mode selection: AUTO or AUTO and Overstore or MDA or TEACHIN	Defined initial state: Channel in Active state Mode: AUTO or AUTO and Overstore or MDA or TEACHIN	Select channel and mode, channel: in active status and mode: as per selection
2	NC Start	Block with end of part program	Block is changed
3	MD20110 $MC_ RESET_MODE_MASK, MD20150 $MC_ GCODE_RESET_VALUES, MD20152 $MC_ GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	Control enabled Reset sequence with evaluation of MD 20110 and MD 20150 and MD 20152

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
4	/_N_CMA_DIR/_N_PROG_ EVENT_SPF or name from MD11620	As an ASUB	Implied call of the path name as an ASUB
5	MD20110 $MC_ RESET_MODE_MASK, MD20150 $MC_ GCODE_RESET_VALUES, MD20152 $MC_ GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	Control activated Reset sequence with evaluation of MD20110 and MD20150 and MD20152. Meaning: The G-code indexed position continues to be specified with machine data

Event

Operator panel reset

Table 2-7 Sequence with operator panel reset

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Select channel and mode: Any	Initial state: Any mode, any channel status	Select mode/channel status from any state
2	Reset		
3	MD20110 $MC_ RESET_MODE_MASK, MD20150 $MC_ GCODE_RESET_VALUES, MD20152 $MC_ GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	Control activated Reset sequence with evaluation of MD20110 and MD20150 and MD20152
4	/_N_CMA_DIR/_N_PROG_ EVENT_SPF or name from MD11620	As an ASUB	Implied call of the path name as an ASUB
5	MD20110 $MC_ RESET_MODE_MASK, MD20150 $MC_ GCODE_RESET_VALUES, MD20152$MC_ GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	Control activated Reset sequence with evaluation of MD20110 and MD20150 and MD20152. Meaning: The G-code indexed position continues to be specified with machine data

Event

Power-up

Table 2-8 Table 1–9 Sequence during power-up

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Reset	after power up	
2	MD20110 $MC_ RESET_MODE_MASK, MD20150 $MC_ GCODE_RESET_VALUES, MD20152 $MC_ GCODE_RESET_MODE	Control activated: After power-up: Reset sequence with evaluation	After power-up, control enables the reset sequence with evaluation of MD 20110 and MD 20150 and MD 20152
3	/_N_CMA_DIR/_N_PROG_ EVENT_SPF or name from MD11620	As an ASUB	Implied call of the path name as an ASUB
4	MD20110 $MC_ RESET_MODE_MASK, MD20150 $MC_ GCODE_RESET_VALUES, MD20152 $MC_ GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	Control activated Reset sequence with evaluation of MD20110 and MD20150 and MD20152. Meaning: The G-code indexed position continues to be specified with machine data

Chronological sequences

For part program start and part-program end:

Time sequence of VDI signals DB21-DB30, ... DBB35 ("Program status" and "Channel status") when processing a part program with an event-driven program call for part program start and part program end:

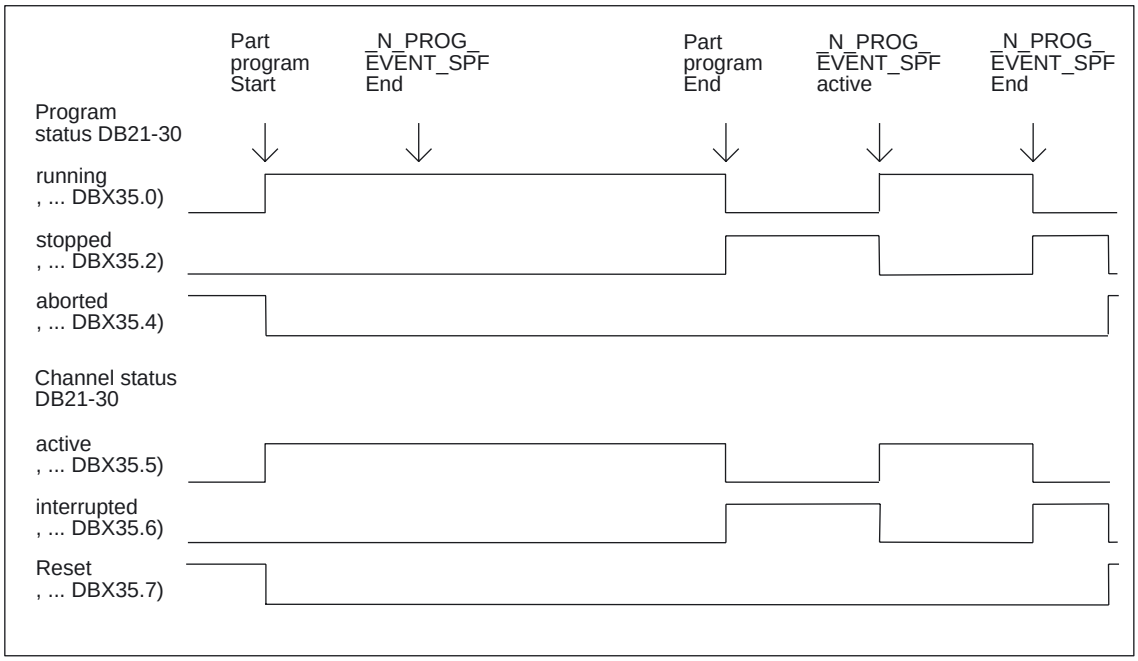


Fig. 2-9 Time sequence of the interface signals for program status and channel status (1)

For operator panel reset (SW 6.3 and higher):

Time sequence of VDI signals DB21-DB30, ... DBB35 ("Program status" and "Channel status") when processing with an event-driven program call:

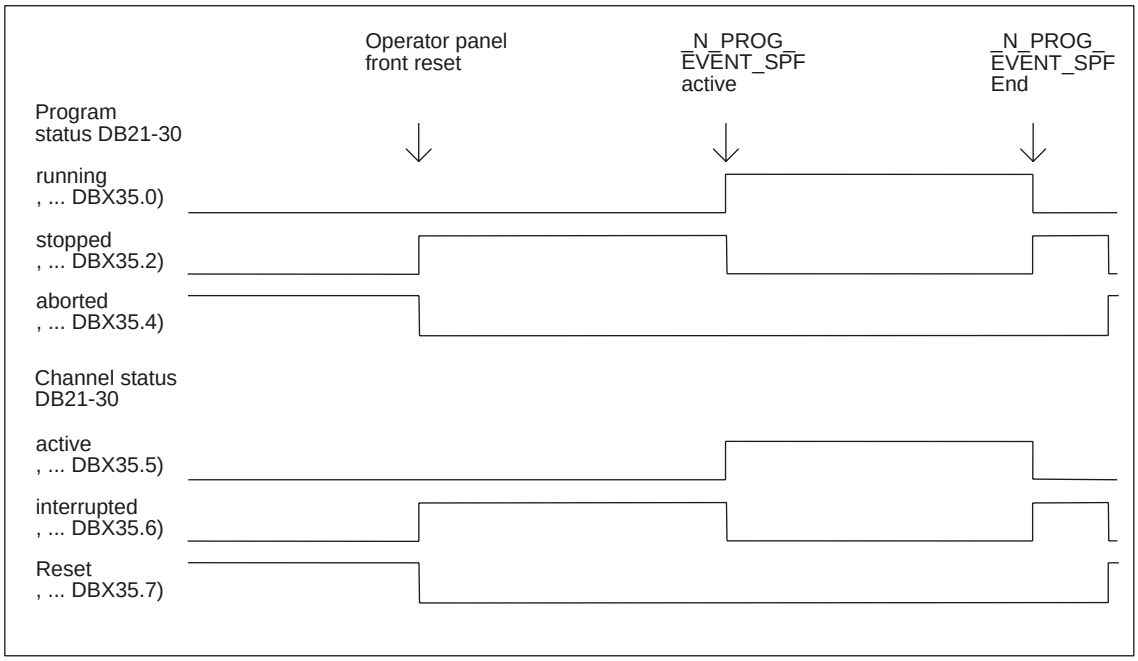


Fig. 2-10 Time sequence of the interface signals for program status and channel status (2)

The chronological sequences from SW 6.3 and higher are shown:

Note

DB21, ... DBX35.4 ("Program status aborted") and DB21, ... DBX35.7 ("Channel status reset") is are only received if _N_PROG_EVENT_SPF has been completed.

Neither DB21, ... DBX35.4 ("Program status aborted") nor DB21, ... DBX35.7 ("Channel status aborted") are received between the program end and the start of the program event.

This is also the case between an operator panel reset and the start of the program event.

Special points to be noted

The following must be noted for user program _N_PROG_EVENT_SPF:

- It is run with the lowest priority and can, therefore, be interrupted by the user ASUB.
- The PLC can be advised of the processing status of _N_PROG_EVENT_SPF via user M functions.
- The trigger event can be determined on the **interface** via the PLC program:
  DB2x.DBB376 provides the following information:
  0 No active event
  Bit 0 = 1 Part program start from channel status RESET
  Bit 1 = 1 Part program end
  Bit 2 = 1 Operator panel RESET
  Bit 3 = 1 Power-up
  Bit 4 = 1 First start after search
  Bit 5-7 reserved, currently always 0
  The general request to 0 makes it possible to determine whether any of the events is present. If a running event disappears with RESET, the associated display bit is also cleared in the interface.

  When very short events occur, the corresponding bit remains active at least for the duration of a complete PLC cycle.
- _N_PROG_EVENT_SPF is always processed in the channel in which the corresponding event has occurred. By scanning MD20000 $MC_CHAN_NAME, it is possible to determine in program _N_PROG_EVENT_SPF which channel is currently being processed.
  Note: Power-up is an event that takes place in all channels.
- Each time MD20108 $MC_PROG_EVENT_MASK is reconfigured, /_N_CMA_DIR/_N_PROG_EVENT_SPF must be loaded or enabled. Otherwise, the alarm 14011 "Program _N_PROG_EVENT_SPF does not exist or not enabled for execution" is output.
- The display can be suppressed in the current block display using the DISPLOF attribute in the PROC statement.
- A single block stop can be disabled with SBLOF attribute command or via MD10702 $MN_IGNORE_SINGLEBLOCK_MASK with Bit 0.

- The read-in disable and single-block processing behavior can be controlled using machine data MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK and MD20107 $MC_PROG_EVENT_IGN_INHIBIT as follows.

  MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK
  _N_PROG_EVENT_SPF makes a block change in spite of single-block mode without an additional Start if
  Bit 0 = 1 is set after the part program start event
  Bit 1 = 1 is set after the part program end event
  Bit 2 = 1 is set after the operator panel reset event
  Bit 3 = 1 is set after the power-up event
  Bit 4 = 1 is set after the first start after search event

  MD 20107: PROG_EVENT_IGN_INHIBIT
  _N_PROG_EVENT_SPF makes a block change in spite of read-in disable if
  Bit 0 = 1 is set after the part program start event
  Bit 1 = 1 is set after the part program end event
  Bit 2 = 1 is set after the operator panel reset event
  Bit 3 = 1 is set after the power-up event
  Bit 4 = 1 is set after the first start after search event
  The following restriction applies to Bit0 == 1(prog-event after part program start):
  If the prog-event is ended with part program command "RET", RET always leads to an executable block (analogous to M17).
  There is no new behavior for Bit0 == 0, i.e., RET is interpreted in the interpreter and does not lead to an "executable block".

No sequences for **start/end of part program** are passed:

- If a user ASUB is started from the reset status, the described sequences for the event for start/end of part program are not passed.
- **Settable Prog-Event properties**

  Machine data MD20109 $MC_PROG_EVENT_MASK_PROPERTIES can be used to define further properties of "event-driven program calls" for specific channels in SW 6.3 and higher:

  – Bit0 = 0: An ASUB started from the RESET channel state is followed by an "event-driven program call" as in earlier versions

  – Bit0 = 1: An ASUB started from the RESET channel state is not followed by an "event-driven program call"

When **starting a part program:**

- /_N_CMA_DIR/_N_PROG_EVENT_SPF is executed as a subroutine.
  _N_PROG_EVENT_SPF must be ended with M17 or RET.
  No return using the REPOS command is permitted; this will generate Alarm 16020 "Repositioning not possible".

Errors **in the case of operator panel reset** or **after powerup**:

- If an error is present when the operator panel is reset or after powerup EMERGENCY STOP or Mode group/NCKContinue, then _N_PROG_EVENT_SPF will only be processed after EMERGENCY STOP has been acknowledged or the error has been acknowledged in all channels.

The following always applies to **powerup**:

- The powerup event occurs in **all** channels at the same time.

Event programs

Example for call by all events

For MD20108 $MC_PROG_EVENT = 'H0F', i.e.
call of _N_PROG_EVENT_SPF for
part program start
part program end and operator panel reset
and power-up:

PROC PROG_EVENT DISPLOF

Sequence for **part program start**

```
IF ($P_PROG_EVENT == 1)
N 10 MY_GUD_VAR = 0                          ;  Initialize GUD variable
N 20 M17
ENDIF
```

Sequence for **part program end** and **operator panel reset**

```
IF ($P_PROG_EVENT == 2) OR ($P_PROG_EVENT
== 3)
N10 DRFOF                                    ;  Deactivate DRF offsets
N20 IF $MC_CHAN_NAME == "CHAN1"
N30 CANCEL(2)                                ;  Delete modal synchronized action 2
N40 ENDIF
N50 M17
ENDIF
```

Sequence for **powerup**

```
IF ($P_PROG_EVENT == 4)
N10 IF $MC_CHAN_NAME == "CHAN1"

N20 IDS=1 EVERY $A_INA[1] > 5.0 DO
$A_OUT[1] = 1
N30 ENDIF
N40 M17
ENDIF
M17
```

Example for call of operator panel reset

For MD20108 $MC_PROG_EVENT = 'H04'

```
PROC PROG_EVENT DISPLOF
N10 DRFOF                                   ;  Deactivate DRF offsets
N20 M17
```

Start with RESET key

Control with MD PROG_EVENT_IGN_INHIBIT

The part program whose name is in MD11620 $MN_PROG_EVENT_NAME and which was stored in one of the directories /_N_CUS_DIR/, /_N_CMA_DIR/, or /_N_CST_DIR/ or the _N_PROG_ENENT_SPF program (default) is automatically started with the RESET key and executed up to the end, regardless of any read-in disable, if the following MD settings are present:

CHANDAT(3)
$MC_PROG_EVENT_IGN_INHIBIT= 'H04F'
$MC_PROG_EVENT_MASK= 'H04F'

Note

For block search (SSL):

If MD11450 $MN_SEARCH_RUN_MODE Bit 1 = TRUE, activation of the last action block also implicitly activates "Automatic ASUB start" after block search of /_N_CMA_DIR/_N_PROG_EVENT_SPF.

For further explanation, refer to "Automatic start of an ASUB after block search" in the "Program Test" section.

For further explanation of power-up of the control, see

References: /IAD/, "Power-On and Power-Up"

2.7.12 Control and effect on stop events

Controlling stop events

Stop events can be controlled for a particular program area in a program section. This program area is referred to as the stop delay area; it is

- Enabled with language command DELAYFSTON

  Disabled with language command DELAYFSTOF

References: /PGA/, "Programming Guide Advanced"

NCK events that can be stopped on short notice and whose reaction can result in the stop delay area for a program section are evaluated according to the following criteria:

- **Immediate** Stops immediately even
  in the stop delay area. Is known as a "hard stop event".
- **Delayed** Stop (even a short term stop) takes place after the stop delay area. Is known as a "soft stop event".
- **Alarm 16954** Program is aborted because illegal program commands have been used in the stop delay area.
- **Alarm 16955** Program is resumed because 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 permits the user to lower the override to 0. The next block in the stop delay area then begins with an override of 0 and the alarm described above occurs.) MD11411 $MN_ENABLE_ALARM_MASK Bit 7 activates this alarm.

Stop event

- A stop event can be triggered by the following:

Event	Classification
VDI 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

In addition, there are NCK events that only stop for a short time to perform a switching operation and then 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".

Boundary conditions

The following exceptions 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, is 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 "Alarm 16954".
- If a STOP DELAY AREA with an override of 0% is entered, the STOP DELAY AREA will not be accepted!

Table 2-9 Summary of NCK events, which stop at least for short time

NCK events	Response	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 end of 1st block outside stop delay area Single-block already active before 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

NCK events	Response	Stop criteria
STOPPROGATASUPEND	System fault	SR end should always deselect the stop delay section.
WAITM	Alarm 16954	NC prog.: WAITM
WAITE	Alarm 16954	NC prog.: WAITE
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 with LFON (stop and fastlift 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"
SYSTEM_SHUTDOWN	immediate	System shutdown for SINUMERIK 840Di
ESR	delayed	Extended stop and retract
EXT_ZERO_POINT	delayed	External work offset
STOPRUN	Alarm 16955	OPI: PI "_N_FINDST" STOPRUN

2.7.13 Asynchronous Subroutines (ASUBs), Interrupt Routines

Overview

Interrupt inputs allow the NC to interrupt the current NC processing operation so that it can react to more urgent events in interrupt routines or ASUBs.

Interrupt routines/ASUBs

Term

Identical functionality is identified by the terms ASUB and Interrupt routines. For the purposes of simplification, only the term interrupt routine will be used from now on.

- 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 subroutine at the point of interruption at a later stage.
- Where several interrupt routines exist, they have to be given different levels of priority so that incoming activation signals arriving at the same time can be placed in a certain order.
- Interrupt routines can be activated from the PLC program by means of a "Function call".

Note

Interrupt routines can be called if the mode group is in program operation mode, i.e., in cases where part program blocks are being processed in either AUTOMATIC or in MDA mode. This restriction is reduced in "Calling the ASUB outside program operation".

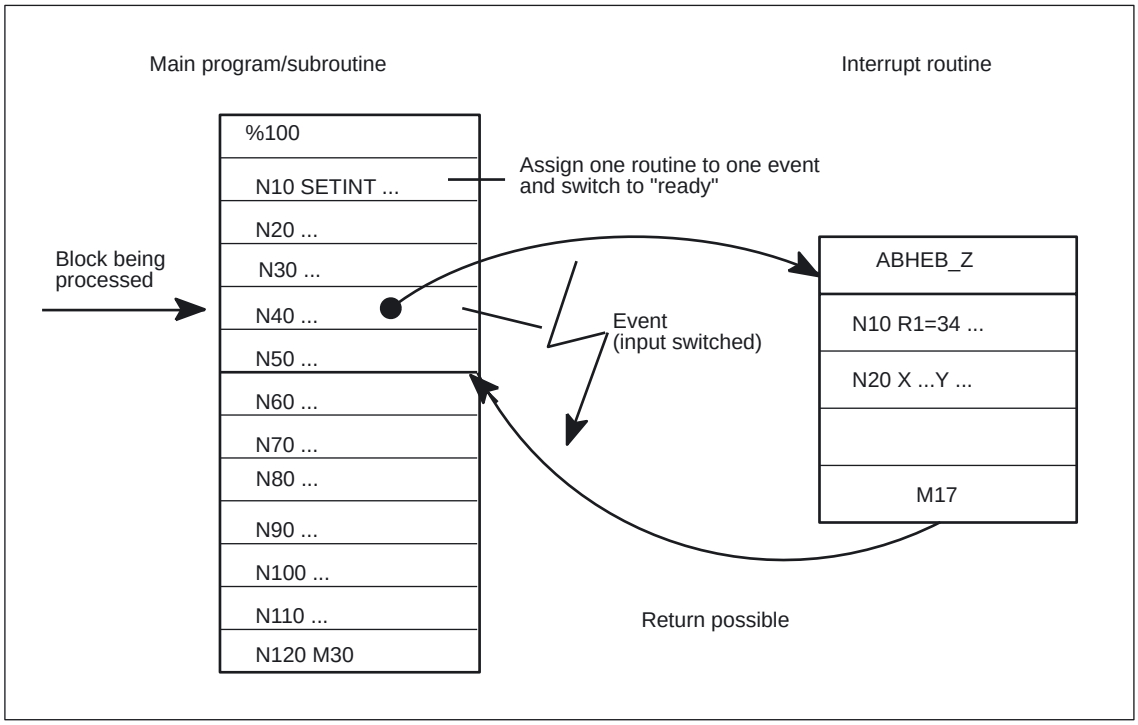


Fig. 2-11 Working with interrupt routines

Parameterization by SETINT

An interrupt signal must be assigned to the part programs via NC instruction SETINT. This turns the part program into an interrupt routine.
The following parameters can be used in the SETINT instruction:

- 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 NC address ALF.
  If mirroring is active for execution via frames, machine data MD21202 $MC_LIFTFAST_WITH_MIRROR can be used to control whether the direction of retraction is to be mirrored as well during fast retraction.
  The displacement for the fast retraction is stored for the 3 geometry axes in MD21200 $MC_LIFTFAST_DIST (displacement during fast retraction from contour).
  If LIFTFAST (fast retraction) is assigned, the maximum axis acceleration for positioning operations (MD 32300: MAX_AX_ACCEL) is reduced by the factor entered in MD20610 $MC_ADD_MOVE_ACCEL_RESERVE (acceleration reserve for overlaid motions).

- BLSYNC: If this parameter is set, the current program block is processed and only then is the interrupt routine started.

References: /PG/, "Programming Guide: Fundamentals"

Interrupt signals

A total of 8 interrupt signals are available.
All inputs can be controlled from the PLC.
In addition, the first four interrupt signals are controlled via the 4 rapid NC inputs on the NCU module (X121).
The signal state of the rapid NC inputs can be read out via the PLC interface (DB10).
Transmission of the rapid NC input signals to the interrupt signals can be disabled via the PLC interface (DB10).

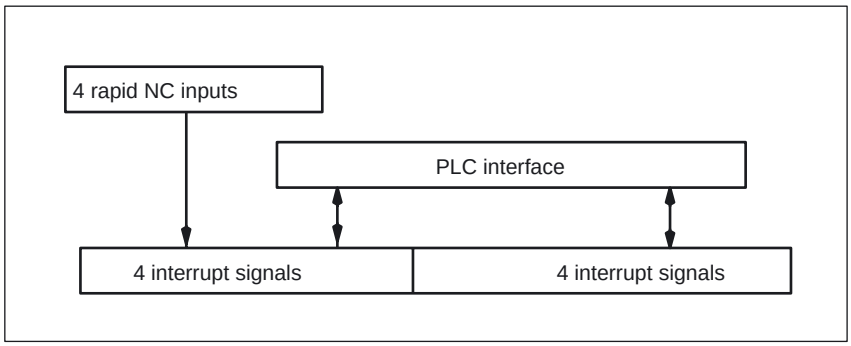


Fig. 2-12 Interrupt signals

For further information about PLC control of the rapid NC inputs (interrupt signals) see:

References: /FB2/, A4, "Digital and Analog NCK I/Os"

Activation of interrupt routine

Interrupt routines can be activated by two different methods:

- By a 0/1 transition of the interrupt signal, triggered by a 0/1 transition at the rapid NC input
- By the call of function call ASUPST (/B1/, /P3/)

Upon activation, all machine axes are decelerated to a standstill according to the acceleration ramp (MD32300 $MA_MAX_AX_ACCEL (axis acceleration)), 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 put in the buffer so that they can be called up again when the interrupt routine is completed.

Exceptions when reorganization is not possible:

- In thread cutting blocks
- With complex geometries (e.g., spline or radius compensation)

Processing of interrupt routine

The "Interrupt" program is automatically started on completion of reorganization. It is treated by the system like a normal subroutine (displayed on the operator interface, 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. REPOSL M17

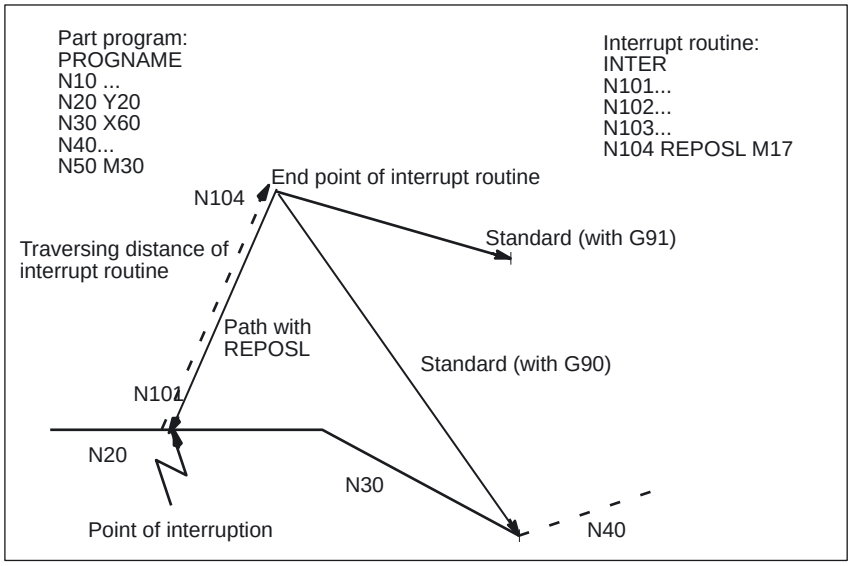


Fig. 2-13 End of interrupt routine

For more information about the REPOS instruction (e.g., syntax), see
References: /PG/, "Programming Guide: Fundamentals"

SAVE command

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.

Interrupt 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 does not respond to the 0/1 signal transition.
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.

Clear assignment

The assignment interrupt signal <-> part program is cleared when the following happens:

- Channel in Reset state
- CLRINT instruction in part program.

For more information on the DISABLE, ENABLE, and CLRINT instruction (e.g., syntax), refer to:

References: /PG/, Programming Guide: Fundamentals

2.7.14 Calling the ASUB outside program operation

ASUB activation states

ASUBs/interrupt routines can also be enabled in the program states or operating modes listed below, in addition to AUTOMATIC mode and MDA:

- JOG, JOG REF
- MDA Teach In, MDA Teach In REF, MDA Teach In JOG, MDA REF, MDA JOG
- AUTOMATIC, stopped, ready
- Not referenced

be activated.

If an interrupt routine is activated in JOG or REF mode, it will interrupt any jogging and referencing operations in progress.

Starting conditions

Machine data MD11602 $MN_ASUP_START_MASK can be used to enable the start of ASUBs for conditions under which ASUBs are otherwise locked by default.

- Stop by means of Stop key, M0, M01
- Not all axes are referenced yet
- Readin disable is active

Details

Explicit ASUB start

If MD11602 $MN_ASUP_START_MASK is set such that the ASUB may not be started automatically, the routine can still be activated by the Start key. Any rapid retraction that may be parameterized is always started.

Priorities

Machine data MD11604 $MN_ASUP_START_PRIO_LEVEL can be used to specify the minimum priority level that the settings in MD11602 $MN_ASUP_START_MASK are to affect (1-128, 1 corresponds to the highest priority level). The priority specified in this machine data covers all priority levels from 1 up to the specified level.

Effect of VDI signals on mode group channels

Machine data MD11600 $MN_BAG_MASK can be set to control the effect of mode group signals (mode group reset, mode group stop axes and spindles, mode change disable) on mode group channels in which interrupt routines are currently being processed.

The machine data also defines whether the internal program execution mode applies only to the one channel in which the interrupt routine has been activated or to all channels in the mode group.

If, via MD 11600 $MN_BAG_MASK, the channel in which the interrupt is running has left the mode group, the mode group signals Mode Group RESET, Mode Group STOP, and Mode Group STOPALL have no effect on this channel. In this way the ASUP can proceed unhindered by mode group signals.

NC response

The following table describes the NC response in the individual operational states:

Status of NC	Start of ASUB	Control system reaction
Program is active	Interrupt, (PLC)	Rapid retraction or stop axes Interruption of program for duration of ASUB Approach to interruption point if ASUB contains REPOS Continuation of 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: MD 20110 $MC_RESET_MODE_MASK MD20112 $MC__START_MODE_MASK **References:** /FB1/, K2, "WorkpieceRelated Actual Value System"
Program mode AUTOMATIC or MDA and **channel stopped**	Interrupt, (PLC)	ASUB is executed. At the end of the ASUB the STOP state is reapplied. REPOS in the ASUB: - ASUB processing is stopped by 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.

Status of NC	Start of ASUB	Control system reaction
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 machining, evaluation of: MD11602 $MN_ASUP_START_MASK MD 11604 $MN_ASUP_START_PRIO_LEVEL internal switchover to "internal program execution mode" if appropriate, activation of ASUB, restoration of 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, block search, alarm, which cannot be reset by an NC start, digitizing activated, channel in error state	**ASUB cannot be started**	The signal "Interrupt request not possible" is generated.

Activation

ASUBs can also be activated by using synchronized actions to set outputs that enable the input of the interrupt indirectly by short-circuit.
Example:

1. Define number of active digital I/Os
   FASTIO_DIG_NUM_INPUTS=3
   FASTIO_DIG_NUM_OUTPUTS=3
2. Generate a short-circuit with the following MDs
   FASTIO_DIG_SHORT_CIRCUIT[0]='H0102B102'
   FASTIO_DIG_SHORT_CIRCUIT[1]='H0202B202'
3. Hardware assignment of external NC input byte for NC program interrupt
   SETINT_ASSIGN_FASTIN=2 ; better to use 1 byte more than needed
4. SETINT (1) PRIO=1 SYNCASUP; Define input as ASUB trigger
5. IDS=1 EVERY $$AC_PATHN>=0.5 DO $A_OUT_[9]=1

References: /FBSY/, Synchronized Actions

REPOS

ASUB sequences may be generated for which there is no unambiguous return to an interruption point in the block processing sequence. System variable: $P_REPINF can be used to scan the ASUB to determine whether a REPOS is possible.

System variable: $P_REPINF	
Value	Meaning
0	Repositioning with REPOS not possible because: - Not called in ASUB - ASUB executed from RESET state - ASUP executed from JOG
1	Repositioning with REPOS possible in ASUB

Acknowledgment signal DB21, ... DBX318.0

Control systems can also be stopped/started from AUTOMATIC mode automatically prior to the end of the ASUB. In the stopped status, the VDI acknowledgment signal DB21, ... DBX318.0 ("ASUB is stopped") can be set on the PLC interface.

ASUB with REPOSA

An ASUB with REPOSA can be initiated in the AUTOMATIC status.

Example:

```
                                        ;  ASUP program
                                        ;
N10 G0 G91                              ;
N20 Y10                                 ;
N30 X20                                 ;
N40 REPOSA                              ;
```

If an ASUB is started after the accumulated part program blocks have been output, the NCK will stop before executing the REPOSA block and the following interface signal is set: DB21, ... DBX318.0 = 1 (ASUB is stopped)

Note

In the situation described above, the signal: "ASUB done" is not yet set by PLC function block FC9.

Where an ASUB is completed without REPOS, the "ASUB done" and

"ASUB is stopped" signals occur simultaneously.

Starting ASUBs in spite of active read-in disable and/or IPO single block (SBL1)

To start ASUBs in spite of an active read-in disable:
DB21, ... DBX6.1 == 1 (read-in disable)
and/or IPO single block (SBL1), the following machine data must be parameterized:

- MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (note prevention of single-block stop)
- MD11602 $MN_ASUP_START_MASK (ignore stop conditions for ASUBs)
- MD20116 $MC_IGNORE_INHIBIT_ASUP (execute interrupt program in spite of read-in disable)
- MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP (execute interrupt routine in single block completely)

Cross-mode Start of ASUBs

Requirements:

- Option: Cross-mode actions (Order no.: 6FC5 251-0AD04-0AA0)
- MD11602 $MN_ASUP_START_MASK, at least Bit 0 = 1

For error-free execution of the function, the following settings in particular must be noted:

- MD11600 $MN_BAG_MASK
- MD11604 $MN_ASUP_START_PRIO_LEVEL
- Interrupt assignment priority

Recommended settings:

- MD11600 $MN_BAG_MASK = H11
- MD11602 $MN_ASUP_START_MASK = H111
- MD11604 $MN_ASUP_START_PRIO_LEVEL = 7

2.7.15 Userdefined system ASUBs

Introduction

The SW supplied with the NCK contains preprogrammed processes (internal ASUBs) for implementation of the RET and REPOS functions. They can be replaced by ASUBs written by the machine tool manufacturer.

Handling

Replacement of system routine

MD 11610 MN_ASUB_EDITABLE controls which of the system routines are to be replaced by user-defined routines.

Value	Meaning
0	The user-defined routine stored in _N_ASUP_SPF in directory _N_CUS_DIR is not activated for either RET or REPOS.
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.

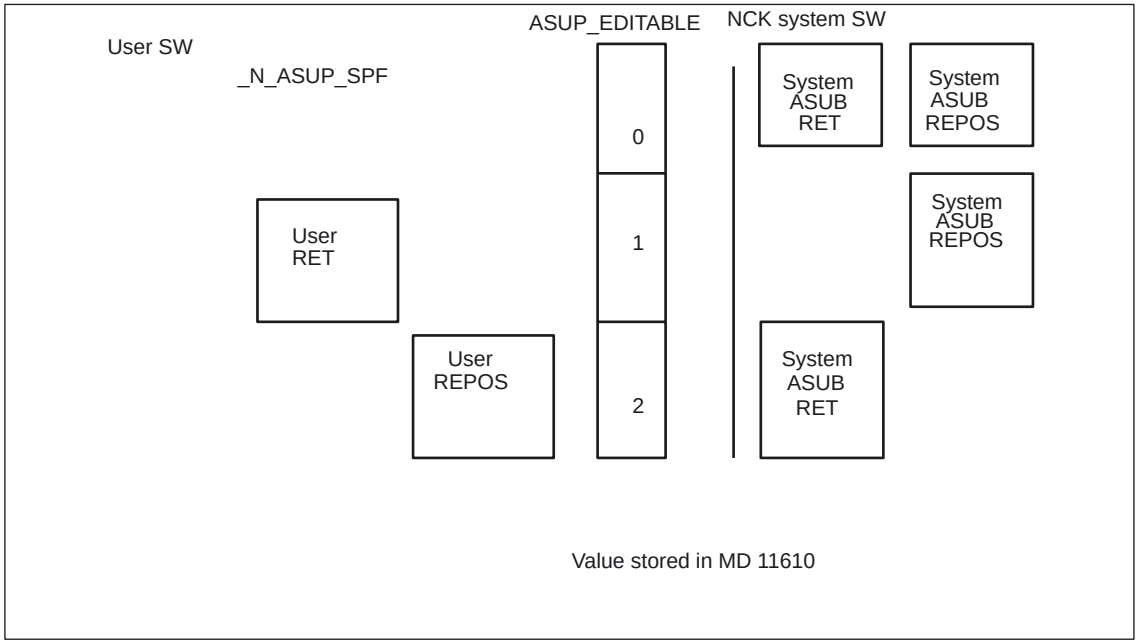


Fig. 2-14 Replacing system ASUBs with user routines

The following diagram illustrates which routines are used:

Installation of user system ASUBs

One routine named _N_ASUP_SPF can be loaded in directory _N_CUS_DIR. It must implement the actions desired by the user for:
RET on value 1 in MD11610
REPOS on value 2 in MD11610.

$AC_ASUP

Bit	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET							1			1	*		1							
REPOS	1	1	1	1	1	1		1	1		*	1		1	1					

*) If Bit 9 is set, then the branch is determined by MD20114: MODESWITCH_MASK starting with:
Bit 0 is set in MD 20114: →RET
Bit 0 in MD20114 not set: →REPOS

Meaning of bits

The meaning of the bits of system variable $AC_ASUP is as follows:

Bit	Meaning
0	User interrupt "ASUB with Blsync" Continuation: Freely selectable REORG or RET
1	User interrupt "ASUB"; The position at which the routine was **stopped** is stored for continuation with REPOS. Continuation: Freely selectable REORG or RET
2	User interrupt "ASUB from channel status Ready" Continuation: Freely selectable REORG or RET
3	User interrupt "ASUB in a manual mode and in channel status Not Ready". Continuation: Freely selectable REORG or RET
4	User interrupt "ASUB"; The **current** position as the interrupt occurred is stored for continuation with REPOS. Continuation: Freely selectable REORG or RET
5	Abortion of subroutine repeat Continuation: With system ASUB REPOS
6	Activation of decoder single block Continuation: With system ASUB REPOS
7	Activation of delete distance-to-go Continuation: With system ASUB RET
8	Activation of axis synchronization Continuation: With system ASUB REPOS
9	Operating mode switchover Continuation: On system ASUB REPOS or RET depending on MD11610

10	Program continuation during Teach In or after Teach In deactivation Continuation: With system ASUB RET
11	Overstore selection Continuation: With system ASUB REPOS
12	Alarm with reaction correction block with REPOS Continuation: With system ASUB REPOS
13	Retraction motion with G33 and stop Continuation: With system ASUB RET
14	Activation of dry run feedrate Continuation: With system ASUB REPOS
15	Deactivation of dry run feedrate Continuation: With system ASUB REPOS
16	Activation of skip block Continuation: With system ASUB REPOS
17	Deactivation of skip block Continuation: With system ASUB REPOS
18	Activate machine data Continuation: With system ASUB REPOS
19	Activate user data Continuation: With system ASUB REPOS

User routine protection level

If a userdefined routine is to be used, i.e., if a value other than 0 is in MD 11610: ASUP_EDITABLE, a protection level can be defined for userspecific routine _N_ASUP_SPF. The protection level can be set to values between 0 and 7 and must be entered in MD11612 $MN_ASUB_EDIT_PROTECTION_LEVEL. For further information about protection levels, refer to:

References: /IAD/, Installation and Start-Up Guide, Protection Level Concept

Single-block processing

Bit 0 in machine data MD10702 $MN_IGNORE_SINGLEBLOCK_MASK determines whether internal ASUBs or the described user ASUBs are to be executed without interruption after every block even when single-block processing is selected:

Bit 0 set:	Internal ASUBs are not stopped in any block.
Bit 0 not set	Internal ASUBs are stopped in every block.



Danger

The machine manufacturer is responsible for the contents of ASUB routines used to replace ASUP.SYF supplied by Siemens.

2.8 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.

2.8.1 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

With channel-specific setting data SD42200 $SC_SINGLEBLOCK2_STOPRE, a preprocessing stop is executed on every block when SBL2 is active. 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.

2.8.2 Single-block stop: Suppression using SBLOF

Single block off

Programs identified by the language command SBLOF are executed completely in one block as with every type of single block.

SBLOF is also valid in subroutines, which are called.

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 the end of the subprogram

In the program

SBLOF must be alone in a block. With effect from this block, single-block stop is deactivated until the next programmed SBLON or until the end of the active subroutine level.

If SBLOF is active, this also applies in subroutines, which are called.

Area

Example: 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 system-internal asynchronous subroutines ASUP1.SYF and ASUP2.SYF, started via Reorg/Repos, can process the system ASUB in one step via programming of SBLOF.

ASUB.SPF

Example: ASUB.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
```

Boundary conditions

- The current block display can be suppressed in cycles using DISPLOF.
- If DISPLOF is programmed together with SBLOF, 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 can be overwritten on a program-specific basis using SBLOF.

Cycle

Example 1: A cycle is to act like a command for the 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
```

2.8.3 Single-block stop: inhibit according to situation

Suppress stopping in single cases

Depending on MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, setting bits 0 to 12 = 1 can suppress stopping at the end of the block during the following processing operations.

Program execution must not stop after single blocks in the case of the following even if block-by-block processing is selected:

1. An internal Asub
2. A user Asub
3. Subroutines with the attribute DISPLOF
4. Intermediate blocks
5. Block search group blocks
6. Init blocks
7. Blocks, which cannot be reorganized
8. Blocks, which are not repositionable
9. A repositioning block, which contains no traversing information
10. A prerun/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 movement 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 continuouspath mode G64 active), further block changes are allowed.

For decoding single block, MD10702 is only effective with "internal ASUB", "user ASUB" and "subroutines 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 ASUB or user ASUB that is suppressed with MD10702 $MN_IGNORE_SINGLEBLOCK_MASK can be activated again by programming SBLON in the ASUB.

This functionality can be suppressed again with MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP. which disables the command SBLON.

Boundary conditions

The following restriction applies to decoding single block SBL2:

- Block search approach blocks
- Block not in ASUB; DISPLOF, SBLOF
- Non-reorganizable and non-repositionable blocks
- Blocks that are not generated in the interpreter, e.g., intermediate blocks

2.8.4 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

The user should select single-block (DB21.DBB0.BIT4) in **one** channel (**KS**) in a mode group. Single-block type A or B refers to **other** channels (KA) of a mode group.

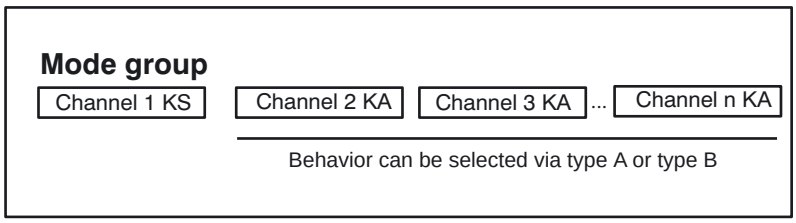


Fig. 2-15 Channel classification for single block in the mode group

Type A, DB11 DBX1.7=1

- 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, DB11 DBX1.6=1

- All channels are stopped.
- All channels receive a start.
- Channel KS stops at the end of the block.
- Channels KA receive a **STOPATEND** (analogous to "NC Stop at block limit" DB21.DBB7.BIT2).
- All channels are stopped at a block limit (at some point in time).

2.9 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
5. Execution from external source (buffer size and number)
6. Execution from external subroutines

2.9.1 Function selection (via operator interface or PLC)

Operator interface or PLC

The user can control part program execution via the operator interface 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 operator 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.

Table 2-10 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:
/LIS/ Lists "Interface Signals"
/BAD/ Operator's Guide "Machine Operating Area"

2.9.2 Activation of skip levels

/, /0, ... /9

It is possible to skip blocks, which are not to be executed every time the program runs. The blocks to be skipped are always identified by a slash character ("/") in front of the block number.
The skip levels are specified in the part program with "/0" to "/7".
Only one skip level can be specified per part program block.

Number of levels

MD9423 $MM_MAX_SKP_LEVEL is used to define the number of skip levels.

Signals

Example:
It is possible to skip blocks that are not to be executed every time the program runs (e.g., program test blocks):
/ ; block skipped, (DB21, ... DBX2.0) 1st skip level
/ N005 ; block skipped, (DB21, ... DBX2.0) 1st "
/0 N005 ; block skipped, (DB21, ... DBX2.0) 1st "
/1 N010 ; block skipped, (DB21, ... DBX2.1) 2nd "
/2 N020 ; block skipped, (DB21, ... DBX2.2) 3rd "
/3 N030 ; block skipped, (DB21, ... DBX2.3) 4th "
/4 N040 ; block skipped, (DB21, ... DBX2.4) 5th "
/5 N050 ; block skipped, (DB21, ... DBX2.5) 6th "
/6 N060 ; block skipped, (DB21, ... DBX2.6), 7th "
/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

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 interface MMC -> PLC (DB21 to DB30) in DBB26.
- "/8" to "/9" via interface MMC -> PLC (DB21 to DB30) in DBX27.0 to DBX27.1.

References: /BAD/ Operator's Guide "Machine Operating Area"

Note

The levels to be skipped can only be changed when the control is in the Stop/Reset state.

control possible.

2.9.3 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 memoryconfiguring MD 28060 $MM_IPO_BUFFER_SIZE. 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

Setting data SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER can be used to dynamically limit the number of blocks in the interpolation buffer to a **smaller** value than that in MD28060 $MC_MM_IPO_BUFFER_SIZE; the minimum number is 2 blocks.

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

2.9.4 Program display modes

Basic block display

A second socalled basic block display can be used with the existing block display to show all blocks that produce an **action on the machine**.

Look Ahead 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 Look Ahead 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 Look Ahead are deleted and preprocessed again afterwards.

Processed values

Values preprocessed in the basic block display coincide with the:

- Selected tools
- Feed and spindle speed
- Position values actually approached
  Exceptions:
  Discrepancies can occur when tool radius compensation is active.

With modulo axes, the programmed value, which may be outside the modulo range, is displayed in the basic block display.

Note

As a basic rule 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.

Configure basic block display

The basic block display can be configured via the following machine data:

NCK machine data for basic block display	Meaning:
MD28400 $MC_MM_ABSBLOCK	Activate basic block display
MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[2]	Size of display buffer
Display of 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.

Activation

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.

Optimizing the size of the 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].

Boundary conditions

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 supplementary 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 can be controlled by the following methods:

- G code DIAMCYCOF (expansion of G code group 26)
  Like DIAMOF, this G code deactivates the diameter programming function. The previously active G code setting remains valid for the position and basic block displays and enables the cycle to calculate with radius, while a diameter value remains displayed in the basic block display.
- 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. A display block containing the string "..." is generated as character for missing display blocks.
In single-block mode, all the display blocks are always generated.

Structure for a DIN block

Basic structure of display block for a DIN block

- Block number/label
- G function of first G group
  (only when altered as compared to the last machine function block).
- Axis positions
  (sequence according to MD20070 $MC_AXCONF_MACHAX_USED).
- Other modal G functions
  (only when altered 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 (WKS / ENS) specified in MD9424 $MM_MA_COORDINATE_SYSTEM.

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 represented as absolute references.
  Addresses AP/RP are represented with their programmed values.

Original block:	**Display block:**
N10 G90 X10.123	N10 X10.123
N20 G91 X1	N20 X11.123

- **Address assignments** (nonDIN 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	
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 displayed irrespective of the output type for the PLC (M22110 $MC_AUXFU_H_TYPE_INT).

- 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 tool holder" function (MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER) 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>=).

- The following rule applies to **Spindle programming** via S, M3, M4, M5, M41 - M45 and M70 (or MD 20094: SPIND_RIGID_TAPPING_M_NR) with reference to the address extension:
  If an address extension was programmed, it is also resolved.

  If several spindles were configured, the address extension is always output as well.
  If no address extension was programmed, the number of the master spindle is inserted (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 subroutine 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).

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

2.9.5 Execution from external source (buffer size and number)

Application

Individual machining steps for producing complex workpieces may involve program sequences that require so much memory they cannot be stored in the NC memory. This type of program can be selected and executed from an external device in "Execution from external source" mode. This "external device" takes the following form with HMI Advanced and with HMI Embedded:

HMI Advanced:	**HMI Embedded:**
Local hard disk or	ATA card or a
Network drive	Network drive

Functions

In principle, any program that is accessible via the directory structure of the data management system of the HMI operator interface can be selected and retroloaded.

- **Execution from external source via V24 interface:**
  In HMI Embedded, the "Execution from external source" softkey can be used to transfer external programs to the NC via the V24 interface.
- **External subroutine:**
  Executing a subroutine in "Execution from external source" mode.

  The "external" subroutine is called via part program command EXTCALL with specification of a call path (optional) and the subroutine identifier.
- **Modal execution from external source:**
  Retaining the selection "Execution from external source" after Reset or part program end.

  Deselection by next program in NC memory:
  If a program was selected on the HMI for "Execution from external source", this condition remains active after a Reset/end of part program. "Execution from external source" mode is not deselected until another program stored in the NC memory is selected for execution.

Buffer interpretation

- **Settable size of reload memory / FIFO buffer:**

  A FIFO buffer must be set up in the NCK in order to execute a program (main program or subroutine) in "Execution from external" mode. The default setting for this buffer size is 30KB.

  Machine data MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE can be used to define the size of the reload buffer. The number of reload buffers is set with MD18362 $MN_MM_EXT_PROG_BUFFER_NUM. A reload buffer must be made available for each of the programs (main run or subroutine) that are processed simultaneously in "Execution from external source" mode.

  Reload buffers are set up in the NCK DRAM. If there is insufficient DRAM available, Alarm 4077 "New value of machine data MD 18360/18362 not set" is generated.

2.9.6 Execution from external subroutines

Subprogram call

The "external" subroutine is called with part program command

- **EXTCALL** with specified <path> and optionally <program name>

Program command EXTCALL is capable of reloading a program in "Execution from external source" mode via the HMI. The external programs must be accessible via the directory structure on the HMI operator interface. They can be stored on the following data media:

HMI Advanced:	**Local hard disk or network drive**
HMI Embedded:	**ATA card or network drive**

Setting Data

SD42700 $SC_EXT_PROG_PATH

SD42700 $SC_EXT_PROG_PATH can be used to set the call path flexibly.
SD42700 $SC_EXT_PROG_PATH contains the path name, which, together with the programmed subroutine identifier, represents the absolute path name of the program to be called.

If an external subroutine is called without an absolute path name, HMI Advanced runs through the same search path as applied when a subroutine is called from the NC main memory.

Path compilation

An external subroutine is called by means of parts program command **EXTCALL**.
From

- - The subroutine name programmed in EXTCALL and

  - Setting data SD42700 $SC_EXT_PROG_PATH

results the program path for the external subroutine call through a character string comprising

- The contents of SD42700 $SC_EXT_PROG_PATH (e.g., /_N_WKS_DIR/_N_WKST1_WPD)
- The character "/" as a separator (if a path was specified with SD42700 $SC_EXT_PROG_PATH)
- The subroutine path or identifier programmed in EXTCALL.

SD42700 $SC_EXT_PROG_PATH is preassigned with a blank. If an external subroutine is called without an absolute path name, HMI Advanced runs through the same search path as applied when a subroutine is called from the NCK memory.

current directory / subroutine identifier
current directory / subroutine identifier_SPF
current directory / subroutine identifier_MPF
/_N_SPF_DIR / subroutine identifier_SPF
/_N_CUS_DIR / subroutine identifier_SPF
/_N_CMA_DIR / subroutine identifier_SPF
/_N_CST_DIR / subroutine identifier_SPF

"current directory":	stands for the directory in which the main program has been selected
"subroutine identifier"	stands for the subroutine identifier programmed in EXTCALL

Example

The program to be reloaded is stored on the local hard disk of HMI Advanced:

- **SD42700 $SC_EXT_PROG_PATH = "_N_WKS_DIR/_N_WST1"**

  Main program_N_MAIN_MPF
  (stored in NC memory and is selected for execution)

```
:N010 PROC MAIN
N020 ....
N030
EXTCALL "ROUGHING"
N040 .....
N050 M30
```

Subroutine_N_ROUGHING_SPF
(stored in HMI memory under workpieces->WP1)

```
N010 PROC ROUGHING
N020 G1 F1000
N030 X= ... Y= ... Z= ...
N040 .....
....
....
N999999 M17
```

Windows path name

Program to be reloaded is stored on network drive or ATA card

- **EXTCALL Windows path name**

  Call for network drive (HMI Embedded or HMI Advanced) e.g.,
  EXTCALL \\R4711\Workpieces\Contour.1.spf

  Call for ATA card (HMI Embedded) e.g.,
  EXTCALL C:\Workpieces\Contour.2.spf

Note

An absolute path must always be specified in HMI Embedded.

For further explanations about operation under HMI Embedded/Advanced, refer to:

References:

/BEM/, HMI Embedded, "Execution from network", "Execution from ATA card" and "EXTCALL"

/BAD/, HMI Advanced, access to external network drive/computer

Power On, RESET

External subroutine calls are aborted and the respective FIFO (reload) buffer erased by a RESET or POWER ON.

A program selected for "Process from external" remains selected for "Process from external" after a Reset. A POWER ON deletes the selection. Default program MPF0 is selected instead.

2.10 System settings for power-up, RESET/part-program end and part-program start

Concept

The control system response can be altered for functions such as G codes, tool length compensation, transformation, coupled axis groupings, tangential followup, and programmable synchronous spindle after

- Power up (power ON),
- Reset/part program end and
- Part program start

by machine data
MD20110 $MC_RESET_MODE_MASK
MD20150 $MC_GCODE_RESET_VALUES and its extension
MD20152 $MC_GCODE_RESET_MODE (specification of initial controller setting on reset) and
MD20112 $MC_START_MODE_MASK (specification of initial controller setting after part program start).

For certain applications (e.g., grinding) it is advisable not to capture the reset position of individual functions until part program start. This is possible through the appropriate setting of MD20110, MD20150, MD20152 and MD20112.

Table 2-11 Change system settings by means of MD

Status	Can be changed in MD
Power up (POWER ON)	MD20110 $MC_RESET_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	MD20110 $MC_START_MODE_MASK and MD20112 $MC_RESET_MODE_MASK

Procedure

Select the desired system response.

- After power-up (POWER ON)
  MD20110 $MC_RESET_MODE_MASK, Bit 0 = 0 or 1

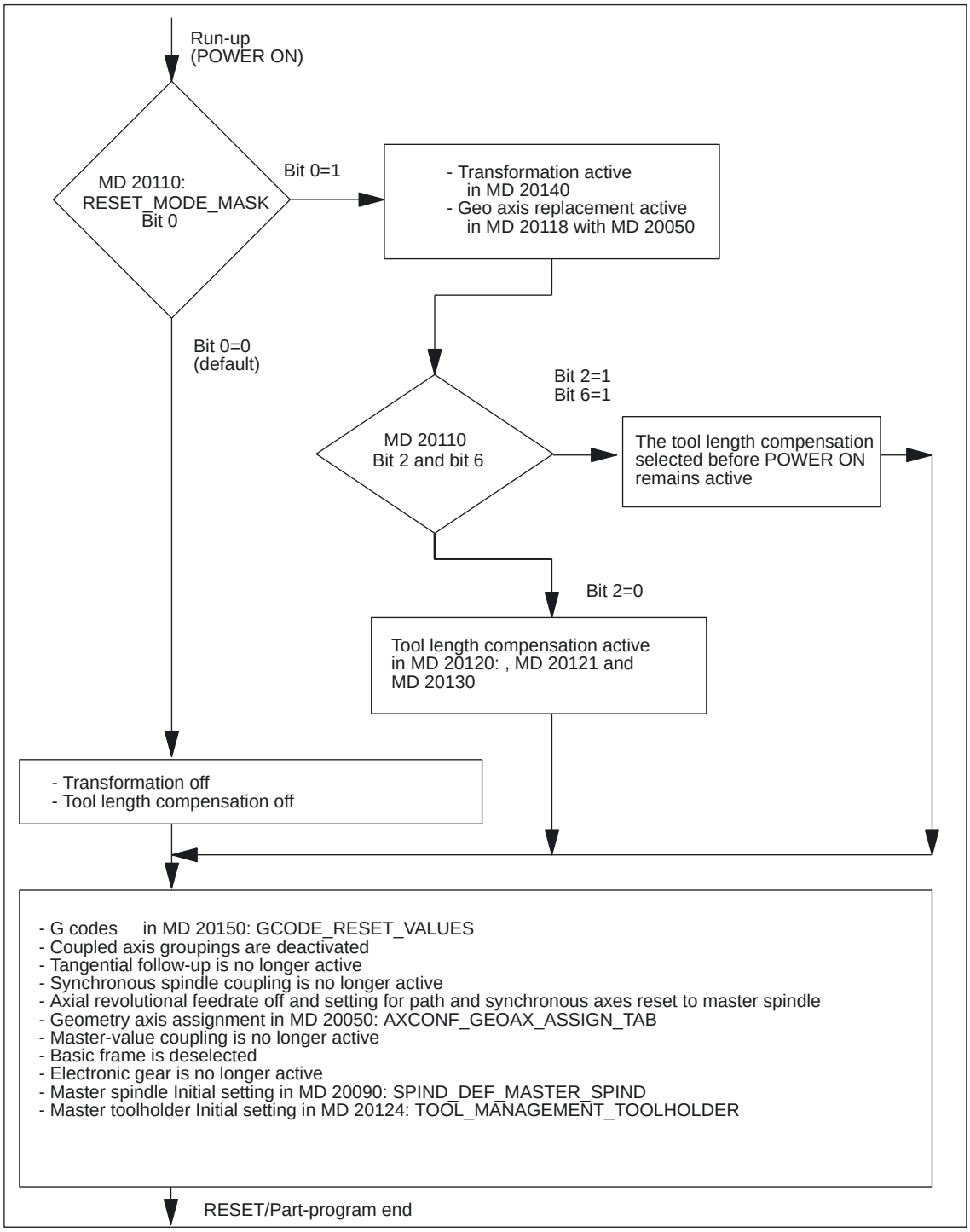


Fig. 2-16 System settings after power-up

- After power-up (POWER ON)
  MD20110 $MC_RESET_MODE_MASK, Bit 0 = 0 or 1

Bits 4 - 13 can be combined optionally.

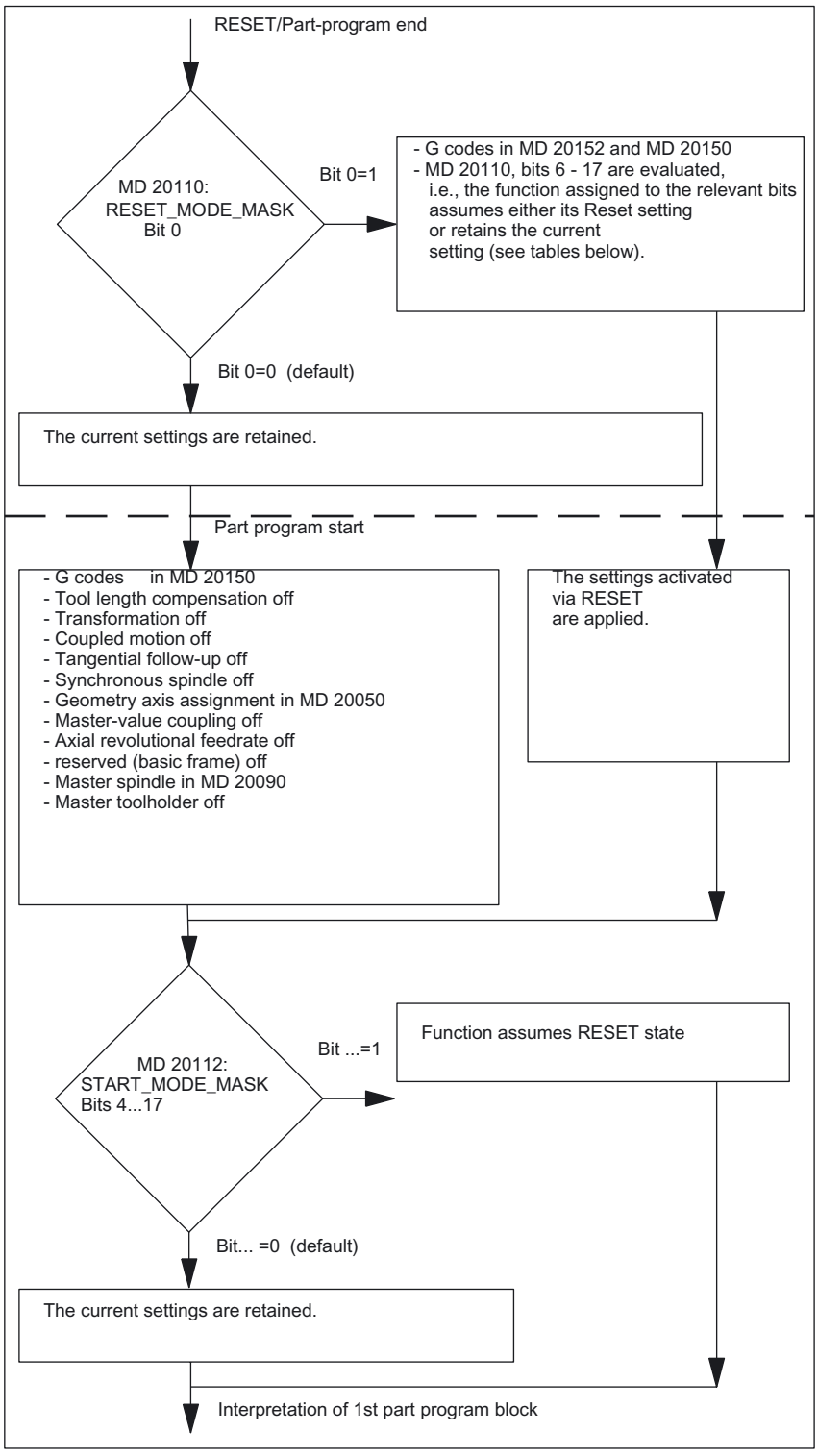


Fig. 2-17 System settings after RESET/part program end and part program start

Table 2-12 Selection of RESET and powerup response

MD number	**RESET_MODE_MASK Definition of control initial setting after power-up and reset/part program end**	
Response after	Bit 0 = 0	Bit 0 = 1
Power ON (booting)	- Transformation not active - Tool length compensation not active - G-Codes according to MD20150 $MC_ GCODE_RESET_VALUES - No coupled axis groupings active - No tangential follow-up active - Nonconfigured Synchronous spindle coupling is deactivated - Geometry axis assignment according to MD20050 - No master value coupling active - Axial revolutional feedrate OFF - Basic frame is deselected	- Transformation active according to MD20140 $MC_TRAFO_RESET_VALUE - Tool length compensation active according to MD20120 $MC_TOOL_RESET_VALUE, MD20121 $MC_TOOL_PRESEL_RESET_VALUE and MD20130 $MC_CUTTING_EDGE_RESET_VALUE, if Bits 2 and 6 in MD20110 = 1, the tool length compensations selected on Power On, otherwise WLK according to MD20120, MD20121, MD20130 - G-Codes according to MD20150 $MC_GCODE_RESET_VALUES - No coupled axis groups active - No tangential follow-on active - Nonconfigured Synchronous spindle coupling is switched off - Geometry axis assignment according to MD20118, MD20050 - No master value coupling active - Axial revolutional feedrate OFF - Basic frame is deselected
RESET/ end of prog.	The current settings are retained. With the next part program start, the following initial setting is activated:	In the case of configured synchronous spindle couple, the coupling is set in accordance with MD21330 $MC_ COUPLE_RESET_MODE_1.
	- G codes according to MD20150 $MC_GCODE_RESET_VALUES - Tool length compensation not active - Transformation not active - No coupled axis groupings active - No tangential follow-up active - Basic frame is deselected - Geometry axis assignment according to MD20050 - Axial revolutional feedrate OFF	GCODES according to MD20150, MD_20152

Table 2-13 Effect of MD20110 $MC_RESET_MODE_MASK Bits 0 to 6

Bit 0 = 1	Bit 1 = 1	Bit 2 = 1	Bit 3 = 1	Bit 4 = 1	Bit 5 = 1	Bit 6 = 1
Initial setting after power up and RESET/end of part program see above	No D, T, M output on tool selection; with active tool manage-ment irrelevant	If bit 0=1 and bit 6 =1, the tool offset of the last active tool is active after POWER ON (irrelevant with active tool management)	Use active tool or tool offset from last test program terminated in test mode	Current plane is retained, in SW 5 replaced by MD 20152 index 5 see below	Current settable frame is retained, as of SW 5 replaced by MD 20152 index 7 see below	Active tool offset is retained

Bit 0 = 0	Bit 1 = 0	Bit 2 = 0	Bit 3 = 0	Bit 4 = 0	Bit 5 = 0	Bit 6 = 0
Initial setting after power up and RESET/end of part program see above.	D, T, M output on tool selection; with active tool management irrelevant.	No tool offset active after POWER ON (no effect when tool management active).	Use active tool or tool offset from last program terminated before program testing activated.	Plane according to MD20150 $MC_ GCODE_ RESET_ VALUES; in SW 5 and higher, replaced by MD 20152 index 5 see below.	Frame according to MD20150 $MC_ GCODE_ RESET_ VALUES; in SW 5 and higher, replaced by MD 20152 index 7 see below.	WZK according to MD20120 $MC_TOOL_RESET_ VALUE, MD20121 $MC_TOOL_PRESEL_ RESET_VALUE, MD20130 $MC_CUTTING_ EDGE_RESET_VALUE; Output of D, T, M to PLC, depending on bit 1.

Table 2-14 Effect of MD20110 $MC_RESET_MODE_MASK Bits 7 to 12

Bit 7 = 1	Bit 8 = 1	Bit 9 = 1	Bit 10 = 1	Bit 11 = 1	Bit 12 = 1
Active trans-formation is retained.	Coupledaxis groupings are retained.	Tangential follow-up is retained.	Not configured synchronous spindle coupling remains active.	Current setting for revolutional feedrate is retained.	Modified geometry axis assignment is retained.

Bit 7 = 0	Bit 8 = 0	Bit 9 = 0	Bit 10 = 0	Bit11 = 0	Bit 12 = 0
Transformation according to MD20140 $MC_TRAFO_ RESET_ VALUE.	Coupledmotion groupings are deactivated.	Tangential follow-up is deactivated.	Not configured synchronous spindle coupling is switched off.	Revolutional feedrate is no longer valid.	Modified geometry axis assignment is deleted according to MD 20050, depending on MD20118 (because of compatibility).

Table 2-15 Effect of MD20110 $MC_RESET_MODE_MASK Bits 13 to 17 (in SW 6.4 and higher, Bit 16 to Bit 17)

Bit 13 = 1	Bit 14 = 1	Bit 15 = 1	Bit 16 = 1	Bit 17 = 1	
Active guide value coupling is retained.	The current setting of the basic frame is retained.	The active electronic gears are deactivated.	The current setting of the master spindle is retained.	The current setting of the master tool holder is retained.	

Bit 13 = 0	Bit 14 = 0	Bit 15 = 0	Bit 16 = 0	Bit 17 = 0	
Guide value coupling is separated.	Basic frame is deselected.	The active electronic gears are retained.	Initial setting for the master spindle according to MD20090.	Initial setting for the master tool holder according to MD20124.	

RESET response of master spindle

Up to SW 6.3 the master spindle setting was reset to the configured value on M30/RESET. Depending on the setting of Bit 0 of MD20110 $MC_RESET_MODE_MASK, there are two cases:

Bit 0 = 0:

No Init blocks are generated. (default: the current settings are retained)
Bit 0 = 1:

Init blocks are generated. The settings activated by RESET become valid.

MD20152 $MC_GCODE_RESET_MODE

In SW 5 and higher, MD20152 $MC_GCODE_RESET_MODE replaces Bits 4 and 5 from MD20110 $MC_RESET_MODE_MASK. In addition, the setting options are expanded:

Up to and including SW 4
the following applies to Bits 4 and 5 of MD20110 $MC_RESET_MODE_MASK:
Bit 4: Level control
Bit 5: Control of settable frames

Each G code group controlled in MD20150 $MC_GCODE_RESET_VALUES[i] can be selectively controlled with the additional MD20152 $MC_GCODE_RESET_MODE[i].

MD20152 $MC_GCODE_RESET_MODE[i] (i = G code group -1)	0	1
MD20150 $MC_GCODE_RESET_VALUES[i]	The value stored in MD20150 is active	The last active/ current value is active

Note

The previous setting option in machine data MD20110 $MC_RESET_MODE_MASK is omitted!

The corresponding bits of this MD are tagged as reserved. Write operations involving these bits are automatically redirected to the corresponding array elements of MD20150 $MC_GCODE_RESET_MODE and Alarm 4502 is output.

When MD20110 $MC_RESET_MODE_MASK is read, the information from the relevant field arrays of MD20150 $MC_GCODE_RESET_MASK are read again and written to MD20110 $MC_RESET_MODE_MASK.

The mode of functioning of MD20112 $MC_START_MODE_MASK remains the same as in SW5.

Part program start

The initial setting of the control system at part program start, e.g., G codes (especially active plane and active settable zero offset), active tool length compensation, transformation, and axis coupling, is determined according to the following table.

Application

If a bit is set in MD20112 $MC_START_MODE_MASK, the reset action of the relevant function can be delayed until the start of the part program.

Table 2-16 Effect of MD20112 $MC_START_MODE_MASK Bits 1 to 7

Bit 1 = 1	Bit 2 = 1	Bit 3 = 1	Bit 4 = 1	Bit 5 = 1	Bit 6 = 1	Bit 7 = 1
No D, T, M output on tool selection; with active tool management irrelevant	Reserved	Reserved	Plane according to MD20150 $MC_GCODE _RESET_ VALUES	Frame according to MD20150 $MC_GCODE _RESET_ VALUES	WZK according to MD20120 $MC_TOOL_RESET_VALUE, MD20121 $MC_TOOL_PRESEL_ RESET_ VALUE, and MD20130 $MC_CUTTING_ EDGE_RESET_VALUE; Output of D, T, M to PLC, depending on bit 1	Trans- formation according to MD20140 $MC_TRAFO_ RESET_ VALUE

Bit 1 = 0	Bit 2 = 0	Bit 3 = 0	Bit 4 = 0	Bit 5 = 0	Bit 6 = 0	Bit 7 = 0
D, T, M output on tool selection; with active tool management irrelevant	Reserved	Reserved	Current plane is retained	Current settable frame is retained	Active tool length offset is retained	Active transformation is retained

Table 2-17 Effect of MD20112 $MC_START_MODE_MASK Bits 8 to 12

Bit 8 = 1	Bit 9 = 1	Bit 10 = 1	Bit 11 = 1	Bit 12 = 1
Coupledaxis groupings are deactivated	Tangential follow-up is deactivated	Not configured synchronous spindle coupling is switched off	Reserved	Geometry axis assignment is deleted according to MD 20050, depending on MD 20118 (because of compatibility)

Bit 8 = 0	Bit 9 = 0	Bit 10 = 0	Bit 11 = 0	Bit 12 = 0
Coupledaxis groupings are retained	Tangential follow-up is retained	Not configured synchronous spindle coupling remains active	Reserved	Modified geometry axis assignment is retained

Table 2-18 Effect of MD20112 $MC_START_MODE_MASK Bits 13 to 17

Bit 13 = 1	Bit 14 = 1	Bit 15 = 1	Bit 16 = 1	Bit 17 = 1
Guide value coupling is separated	Reserved for basic frame	Reserved for electronic gears	Initial setting for master spindle according to MD20090 $MC_SPIND_DEF _MASTER_SPIND	Only if MD20124 $MC_TOOL_MANAGEMENT_ TOOLHOLDER > 0: Initial setting for the master tool holder according to MD 20124 $MC_Otherwise setting for master spindle

Bit 13 = 0	Bit 14 = 0	Bit 15 = 0	Bit 16 = 0	Bit 17 = 0
Active guide value coupling is retained	Reserved for basic frame	Reserved for electronic gears	Current setting of the master spindle (SETMS) is retained	The current setting of the master tool holder (SETMS) is retained

Note

In MD20110 $MC_RESET_MODE_MASK, bits set to 1 cause settings to be retained,

In MD20112 $MC_START_MODE_MASK, bits set to 0 cause settings to be retained.

Meaning of the machine data

The channelspecific machine data in the table has the following meanings. Details are specified in section NO TAG.

- **MD20120 $MC_TOOL_RESET_VALUE**
  Specification of tool (T number),
  whose tool length compensation values are to be taken into account on reset and power-up in accordance with **MD20110 $MC_RESET_MODE_MASK.**

  MD20121 $MC_TOOL_PRESEL_RESET_VALUE
  Specification of tool (T number) as preselected tool whose tool length compensation values are to be taken into account on reset and power-up in accordance with **MD20110 $MC_RESET_MODE_MASK.**

  MD20130 $MC_CUTTING_EDGE_RESET_VALUE
  Definition of edge number (D number) of tool in $MC_TOOL_RESET_VALUE

  MD20140 $MC_TRAFO_RESET_VALUE
  Definition of transformation data block (TRAORI, TRAANG, SW 4 and higher: TRANSMIT)

  MD20150 $MC_GCODE_RESET_VALUES
  RESET states of G groups

  MD20152 $MC_GCODE_RESET_MODE
  GCODE initial setting on RESET

  MD20152 specifies for each entry in MD20150 $MC_GCODE_RESET_VALUES whether the setting in accordance with MD20150 $MC_GCODE_RESET_VALUES is taken again (entry in MD20152=0) or the current setting is retained (entry in MD20152=1).

  MD21330 $MC_COUPLE_RESET_MODE_1
  Cancellation of an axis coupling

  MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB
  Assignment of geometry axis to channel axis
  acts on Bit 12 of MD 20110/20112
  MD20118 $MC_GEOAX_CHANGE_RESET

  acts on Bit 12 of MD 20110/20112

Example:

1. Activate RESET setting on RESET:
   MD20110 = 'H01' (Bit 0)
   MD20112 = '0'
2. Transformation remains active on RESET/part program start:
   MD20110 = 'H81' (Bit 0 + Bit 7)
   MD20112 = '0'
3. Tool length compensation remains active on RESET/part program start:
   MD20110 = 'H41' (Bit 0 + Bit 6)
   MD20112 = '0'
4. Active plane (Bit 4) and settable frame (Bit 5) remain active after RESET and are reset on part program start:
   MD20110 = 'H31' (Bit 4 + Bit 5)
   MD20112 = '30'

Note

for 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/MDA start, the first programming of the axes must use absolute measurements (because of the offset behavior).

Exception: With MD42442/MD42440 the offset behavior for G91 is suppressed.

2.11 Subroutine call with M, T, and D functions

2.11.1 Replacement of auxiliary functions with subroutines

Function

The M, T, and D auxiliary functions can be replaced with subroutines:

- M functions (switching operations)
- T functions (tool selection)
- D functions (tool offset selection)

The following machine data can be used to configure subroutine calls with M, T, and D auxiliary functions:

Machine data	Meaning
MD10715 $MN_M_NO_FCT_CYCLE	M function to be replaced by a subroutine
MD10716 $MN_M_NO_FCT_CYCLE_NAME	Name of subroutine for M function
MD10717 $MN_T_NO_FCT_CYCLE_NAME	Name of subroutine for T function
MD10718 $MN_M_NO_FCT_CYCLE_PAR	M function replacement with parameters
MD10719 $MN_T_NO_FCT_CYCLE_MODE	Parameter assignment of T function replacement
MD11717 $MN_D_NO_FCT_CYCLE_NAME	Name of subroutine for D function

Note

The functions described above are also effective in ISO dialect mode.

Block search response

During block search with calculation and SERUPRO, subroutine calls with M, T and D functions are executed in the same way as in normal program operation.

M functions that cannot be configured

M functions with a fixed meaning must not be used to call subroutines. The following M functions have standard predetermined meaning:

M function	Comments
M0 to M5, M17, M30, M19, M40 to M45	- - -
M98, M99	With activated external NC language: MD18800 $MN_MM_EXTERN_LANGUAGE == TRUE

M functions can be defined for specific tasks using the following machine data. Through this, they receive a fixed meaning and, therefore, must not be used to call subroutines:

Machine data	Specific tasks
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)

Machine data	Specific tasks
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

Exceptions

The configured M function for tool change:
MD22560 $MC_TOOL_CHANGE_M_CODE (M function for tool change)

Boundaries for macro function

Macros are used to define a character string replacement consisting of text only. This replacement is made prior to execution of the replacement mechanism described here.

Boundary conditions

For subroutine calls with M, T, and D functions, the following supplementary conditions apply:

- Only one function may be replaced for each part program line.
- A subroutine call must not be superimposed on M functions with predetermined meaning.
- In a part program block with an M, T or D function to be replaced, the following applies:
  - No modal subprogram call can be active
  - No subroutine return can be programmed
  - No part program end can be programmed

Note

The M, T, and D values passed to the cycle have not yet been executed and must, therefore, be programmed again in the cycle.

2.11.2 M function replacement

Subroutine call via M function

Note

Subroutine calls using an M function are referred to below as M function replacement.

The following machine data are used to configure M function replacement:

- MD10715 $MC_M_NO_FCT_CYCLE (M function to be replaced by a subroutine)
- MD10716 $MC_M_NO_FCT_CYCLE_NAME (Name of subroutine for M function)

The M function via which the subroutine specified in machine data MD10716 $MC_M_NO_FCT_CYCLE_NAME is called is defined in machine data MD10715.

Call rules

If the M function configured with MD10715 $MC_M_NO_FCT_CYCLE is programmed in a part program block, the appropriate subroutine is called at the end of the part program block. If the M function is programmed again within the called subroutine, the M function is not replaced again. In addition, other M function expansions configured with MD10715 $MC_M_NO_FCT_CYCLE or MD10716 $MC_M_NO_FCT_CYCLE_NAME are not executed.

Exceptions

The M function is also replaced in an ASUB if the ASUB was started in a subroutine that was called via an M function.

Address extension of M function

Using system variable $C_ME it is possible to read the address extension of the M function in the called subroutine.

Example configurations

- Call of subroutine SUB_M101 via M101
  MD10715 $MC_M_NO_FCT_CYCLE[0] = 101
  MD10716 $MC_M_NO_FCT_CYCLE_NAME[0] = "SUB_M101"
- Call of subroutine SUB_M102 through M102
  MD10715 $MC_M_NO_FCT_CYCLE[1] = 102
  MD10716 $MC_M_NO_FCT_CYCLE_NAME[1] = "SUB_M102"

2.11.3 Replacement of tool programming

2.11.3.1 T and D function replacement

Subroutine call via T function and D or DL function

Note

The subroutine call via T function is referred to hereinafter as T function replacement, and the subroutine call via D or DL function is referred to hereinafter as D function replacement.

Call rules

If a T function is programmed in a part program block, a machine data can be used to specify that the subroutine defined in MD10717 $MN_T_NO_FCT_CYCLE_NAME[n] is called at the start or end of the block. Similarly, a replacement subroutine for address D or DL can be defined with MD11717 $MN_D_NO_FCT_CYCLE_NAME[n].

With MD10719 $MN_D_NO_FCT_CYCLE_MODE Bit 1 and Bit 2, the call timing for the T function replacement and D or DL function replacement can be set.

Assigning subroutines

The machine data

- MD10717 $MN_T_NO_FCT_CYCLE_NAME is used to assign a subroutine to the T command
- MD11717 $MN_D_NO_FCT_CYCLE_NAME is used

to assign a subroutine to the D and DL command. The same subroutine should be configured for the T and D replacement. The corresponding values are not output, the T word and the D or DL word must be programmed again in the cycle.

Boundary condition

The subroutine configured with MD10716 $MN_M_NO_FCT_CYCLE_NAME[n] and with MD10717: T_NO_FCT_CYCLE_NAME or MD11717 $MN_D_NO_FCT_CYCLE_NAME cannot take effect simultaneously in a block (part program line), i.e. no more than one D, T, or M function replacement (or generally just one subroutine call) may be executed per block. Conflicts with other subroutine calls are signaled by alarm 14016.

Configurable time behavior for D/T function replacement

The timing of the call of the replacement subroutine can be parameterized as follows with (Bit 1 and Bit 2):

MD10719 Bit 1	MD10719 Bit 2	Timing of call of replacement subroutine
0	0	At block end (default)
1	0	At block start
0/1	1	At block start and block end

Timing of call at block end MD10719 Bit 1 = 0

After the replacement subroutine has been executed, the interpretation is resumed with the subprogram line following the line that triggered the replacement operation.

Timing of call at block start MD10719 Bit 1 = 1 and Bit 2 = 0

After the replacement subroutine has been executed, the part program line that controlled the call of the subroutine is interpreted. The T address and the D or DL address and the M function for the tool change are no longer processed.

Timing of the call at block start and at block end M10719 Bit 2 = 1

The replacement program is called twice.

Note

Which replacements are to be performed at the block start and which are to be performed at the block end is up to the user.

System variable $P_SUB_STAT can be used to determine the timing (block start/block end) of a replacement operation. The system variable $P_SUB_STAT returns for

- Value 0: Replacement subroutine is not active
- Value 1: Replacement subroutine is active, call at block start
- Value 2: Replacement subroutine is active, call at block end

Configuration of transfer of D or DL function to the replacement cycle

The T function replacement permits the use of machine data MD10719 $MN_T_NO_FCT_CYCLE_MODE for specifying which of the following are to occur if D or DL and T or programmed simultaneously in a block (Bit 0)

- D or DL must be passed as a parameter to the T replacement cycle
- A call is to be executed before the T replacement cycle

MD10719 Bit0	Transfer to the replacement subroutine
0	D or DL number is transferred to the cycle via a system variable (default value)
1	D or DL number is calculated directly in the block. This function is active only if the tool change has been configured with M function, otherwise the D or DL values are always transferred. *

*Tool change with M function is configured with
MD22550 $MC_TOOL_CHANGE_MODE = 1.

Example of T function replacement for tool change

Behavior same as before:

Tool change with M6 not active:

MD22550 $MC_TOOL_CHANGE_MODE = 0
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE" ; **T replacement cycle**

```
N110 D1                          ;
N120 G90 G0 X100 Y100 Z50        ; D1 is active
N130 D2 X110 Z0 T5               ; D1 remains active, programmed D2 is provided to
                                   the T replacement cycle as a variable
```

A comprehensive example for **M/T function replacement with tool change** including the associated replacement subroutines can be found at the end of the next section.

2.11.3.2 M function replacement for tool change

Calling a tool change program with M function

The following machine data are used to configure M function replacement for calling the tool change program:

- MD10715 $MN_M_NO_FCT_CYCLE (M function to be replaced by a subroutine)
- MD10716 $MN_M_NO_FCT_CYCLE_NAME (name of subroutine for M function)
- MD10718 $MN_M_NO_FCT_CYCLE_PAR (M function replacement with parameters)

The information required for the tool offset or tool offset selection is passed to the subroutine via a system variable.

Programming the M function replacement with parameter transfer

The address extension and function value of the M function must be explicitly, i.e., constantly, programmed for M function replacements with parameter transfer. An indirection definition via variables is not allowed.

Permissible programming:

- M< function value >
- M = < function value >
- M[<address extension>] = <function value>

Illegal programming:

- M = <variable1>
- M[<variable2>] = <variable1>

Boundary conditions

M and T functions for tool change in a block

If, in addition to the M function replacement with parameter transfer, a T function replacement was configured, the following behavior is applicable in case of a conflict, i.e., T and M function for tool change are in one block:

- The T function replacement does not take place.
  Instead, the T value is made available to the M function replacement via the appropriate system variable $C_T...
- Programming the address T in the M function subroutine to be replaced does not result in another replacement.

Configuration example of call of subroutine SUB_M6 through M6 with parameter transfer

MD10715 $MN_M_NO_FCT_CYCLE[2] = 6
MD10716 $MN_M_NO_FCT_CYCLE_NAME[2] = "SUB_M6"
MD10718 $MN_M_NO_FCT_CYCLE_PAR = 6

Program example of tool change with M function replacement

```
PROC MAIN
...
N10 T1 D1 M6
...
N90 M30

PROC SUB_M6
N110 IF $C_T_PROG == TRUE       ; Scan whether address T has been programmed
N120 T[$C_TE] = $C_T            ; Execute T selection
N130 ENDIF
N140 M[$C_ME] = 6               ; Execute tool change
N150 IF $C_D_PROG == TRUE       ; Scan whether address D has been programmed
N160 D = $C_D                   ; Execute D selection
```

```
N170 ENDIF
N190 M17
```

A comprehensive example for **M/T function replacement with tool change** including the associated replacement subroutines can be found in the next section.

2.11.3.3 Example of M/T function replacement for tool change

Replacement of T address and D or DL addresses at block start

- Tool change occurs with address T (thus without M code).
- Tool management not active.
- B axis as indexing axis with Hirth tooth system.

Configuration of machine data:

MD11717 $MN_D_NO_FCT_CYCLE_NAME = "D_T_SUB_PROG" ; **D replacement cycle**
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "D_T_SUB_PROG" ; **T replacement cycle**
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 'H2'
MD22550 $MC_TOOL_CHANGE_MODE = 0
MD10718 $MN_M_NO_FCT_CYCLE_PAR = -1

Programming with main and replacement subroutine

```
N410 G01 F1000 X10 T1 = 5 D1          ;  Main program

N1000 PROC D_T_SUB_PROG DISPLOF       ;  Replacement subroutine
 SBLOF
...
N4100 IF $C_T_PROG == TRUE               Scan whether address T has been
                                         programmed
N4110                                 ;  Replacement for address T with tool no.
N4120 POS[B] = CAC($C_T)              ;  Move revolver to indexing position
N4130 T[$C_TE] = $C_T                 ;  Select tool (T selection)
N4140 ENDIF

N4300 IF $C_D_PROG == TRUE            ;  Scan whether address D has been
                                         programmed
N4300                                 ;  Replacement for address D
N4310 D = $C_D                        ;  Select offset (D selection)
N4320 ENDIF

N4400 IF $C_DL_PROG == TRUE           ;  Scan whether address DL has been
                                         programmed
N4410                                 ;  Replacement for address DL
N4420 D = $C_DL                       ;  Select insert offset
N4430 ENDIF
...
N9999 RET
```

This causes part program line "N410 G01 F1000 X10 T1=5 D1" to execute the following program:

```
N1000 PROC D_T_SUB_PROG DISPLOF        ;  Replacement subroutine
 SBLOF
...
N4100 IF $C_T_PROG == TRUE                Scan whether address T has been
                                          programmed
N4110                                  ;  Replacement for address T with tool no.
N4120 POS[B] = CAC($C_T)               ;  Move revolver to indexing position
N4130 T[$C_TE] = $C_T                  ;  Select tool (T selection)
N4140 ENDIF

N4300 IF $C_D_PROG == TRUE             ;  Scan whether address D has been
                                          programmed
N4300                                  ;  Replacement for address D
N4310 D = $C_D                         ;  Select offset (D selection)
N4320 ENDIF

N4400 IF $C_DL_PROG == TRUE            ;  Scan whether address DL has been
                                          programmed
N4430 ENDIF
N9999 RET
N410 G01 F1000 X10                     ;  Rest of N410 without tool programming
```

2.11.4 Parameter transfer to the replacement subroutine

Rules for parameter transfer

The following basic procedure applies when transferring parameters to the replacement cycle:

- If on of the replacements indicated above is active, all information required for the tool or offset selection (T, D or DL, M function for tool change, address extensions) is forwarded to the replacement subroutine.
  Exception: MD10719 $MN_T_NO_FCT_CYCLE_MODE Bit 0 = 1. Here, it is possible to configure that the D/DL addresses not be transferred to the replacement subroutine.
- Only **one** of the replacement subroutines indicated above can be executed **in a part program line.**
  If there are multiple replacements of the T or D/DL address and the M function for the call of the tool change program, only one of the replacement subroutines is ever active even if their names are different. The, as described above, all necessary parameters for the tool offset or tool offset selection are made available. If configured accordingly, the subroutine is called before and/or after execution of the part program line.

- For replacement of the T or D/DL address and the M function for the call of the tool change program, **the same subroutine** can be configured in which the function to be replaced is determined by scanning system variables. The option also exists to configure different subroutines.

System variable for the transfer parameter

The programmed values for the transfer to the replacement subroutine can be read using the following system variable:

System variable	Remarks
$C_T_PROG	TRUE if address T has been programmed
$C_T	Value of address T (integer)
$C_TE	Address extension of address T
$C_TS_PROG	TRUE, if address T was programmed with the tool identifier of type STRING
$C_TS	Value of address T from tool identifier (string, with tool management only)
$C_D_PROG	TRUE if address D has been programmed
$C_D	Value of address D
$C_DL_PROG	TRUE if address DL has been programmed
$C_DL	Value of address DL
$C_M_PROG	TRUE if M function was programmed for tool change
$C_M	Value of replaced address M (integer) There are two different cases: 1. The replacement subroutine for the tool change configured with MD10718 $MN_M_NO_FCT_CYCLE_PAR was called. $C_M contains the value MD10715 $MN_M_NO_FCT_CYCLE[$MN_M_NO_FCT_CYCLE_PAR]. 2. In the case of a tool change with M code, only one replacement subroutine was configured for the T and/or D/DL addresses. If the M code for the tool change is programmed together with one of the addresses to be replaced, the value of MD22560 $MN_TOOL_CHANGE_M_MODE is transferred to the replacement subroutine.
$C_ME	Address extension of replaced M function



Caution

Parameter values

Values passed to the cycle have not yet been executed and must, therefore, be programmed again in the cycle.

Example of tool change with M6 active and MD10719: T_NO_FCT_CYCLE_MODE= 0

MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE" ; **T replacement cycle**

```
N210 D1                          ;
N220 G90 G0 X100 Y100 Z50        ; D1 is active
N230 D2 X110 Z0 T5               ; D1 remains active, programmed D2 is provided to
                                   the T replacement cycle as a variable
N240 M6                          ; New tool is selected
```

Expanded behavior:

Tool change with M6 active and MD10719: T_NO_FCT_CYCLE_MODE= 1

MD22550 $MC_TOOL_CHANGE_MODE = 1
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE" ; **T replacement cycle**

```
N310 D1                          ;
N320 G90 G0 X100 Y100 Z50        ; D1 is active
N330 D2 X110 Z0 T5               ; D2 is activated, D2 is not transferred to the T
                                   replacement cycle as a variable
N340 M6                          ; T5 is activated
```

Example also with parameter transfer on tool change with M6 active and MD10719 = 1

- A replacement cycle for T and M6 has been configured.
- In addition, the parameter transfer to the M6 replacement cycle was configured with MD10718 $MN_M_NO_FCT_CYCLE_PAR.

If M6 is now programmed with D or DL in the block, the D or DL number is also passed as a parameter to the M6 replacement cycle when machine data
MD 10719 $MN_T_NO_FCT_CYCLE_MODE = 1.

MD22550 $MC_TOOL_CHANGE_MODE = 1
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE" ; **T replacement cycle**

```
N410 D1                          ;
N420 G90 G0 X100 Y100 Z50        ; D1 is active
N430 D2 X110 T5 M6               ; D1 remains active, D2 and T5 are transferred to
                                   the M6 replacement cycle as a variable
```

Conflict resolution in case of multiple replacements with the same name

The following table provides information on how conflicts are resolved if all three replacement subroutines have been configured with different names.

Replacement	Configuration of replacement subroutines
For Address D and DL:	MD11717 $MN_FCT_CYCLE_NAME = "D_SUB_PROG"
For Address T:	MD10717 $MN_FCT_CYCLE_NAME = "T_SUB_PROG"
For M function	For the call of the tool change program
	MD10716 $MN_M_NO_FCT_CYCLE_NAME[0] = "M6_SUB_PROG"
	MD10715 $MN_M_NO_FCT_CYCLE[0] = 6
	MD10718 $MN_M_NO_FCT_CYCLE_PAR = 0
Parameterization	MD22550 $MC_TOOL_CHANGE_MODE = 1
	MD22560 $MC_TOOL_CHANGE_M_CODE = 206

The following are programmed in one part program line			Called replacement subroutine
D and/or DL	T	M6	
–	–	X	M6_SUB_PROG
–	X	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

Note

Configuration of MD10718 $MN_M_NO_FCT_CYCLE_PAR:

The configuration of a replacement subroutine for the tool change with M code is dependent on whether the tool change with M code was actually configured with MD22550 $MC_TOOL_CHANGE_MODE.

2.11.5 Properties of replacement subroutines

General rules for replacement subroutines

- Like any other subroutine, a replacement subroutine can contain a PROC statement.
- If the replacement subroutine is called from ISO mode, the PROC statement of the replacement subroutine causes an implicit switchover to the standard language mode.
- However, no transfer parameters can be defined. The transfer of data to the replacement subroutine always occurs via a system variable.
- The PROC statement enables program attributes such as SBLOF and DISPLOF to be programmed.
- The replacement subroutine behaves like any other subroutine with respect to SBLOF in active single block mode:

  – **Return jump with M17**: Stop at end of subroutine
  Note: Depending on MD20800 $MC_SPF_END_TO_VDI Bit0, the M function is output to the PLC.

  – **Return jump with RET**: No stop at end of subroutine
- If the replacement cycle has the DISPLOF attribute, the program line that has resulted in the replacement cycle call is displayed as the current block in the block display.
- With DELAYFSTON and DELAYFSTOF, areas or even the entire replacement cycle can be protected against interruptions such as NC Stop, read-in disable, etc.
  References: Programming Guide, PGA "Conditionally interruptible program sections".
- Replacements do not occur recursively, i.e., the function that has led to the replacement subroutine call is no longer replaced if it is programmed again in the replacement subroutine.

Note

The following applies in general to replacements

In the case of replacements that are called due to programmed auxiliary functions, the replacement subroutine call does not result in any output of the auxiliary function to the PLC. The auxiliary function is only output if it is programmed again in the replacement cycle.

Boundary condition for replacement subroutines

The following supplementary conditions apply to replacement subroutines:

- Replacements in synchronized actions are not permitted
- Replacements in technology cycles are not permitted
- A part program line that contains language constructs to be replaced at the block start may not be placed in front of any block-wise synchronized actions.
- Only the actions required for the respective replacements can be performed in the replacement cycle.
- The system does not monitor whether the user actually replaces the respective function in the replacement cycle. Therefore, if the user fails to implement this correctly, the actual intended function can be omitted.

2.12 Program runtime/workpiece counter

2.12.1 Function

The functions: "Program runtime" and "Workpiece counting" are not identical to the corresponding tool management functions, but are intended to support machines, which have no explicit tool management.

2.12.2 Program runtime

Function

The function "Program runtime" makes the NC internal timers available to monitor technological processes via system variables.

NC-specific system variables

The following NC-specific system variables are available:

Names	Meaning	Description
$AN_SETUP_TIME	Time since the last control powerup with default values ("Cold start" in min.)	Counts the time since the last control powerup with default values. It is automatically reset to the default values each time the control is powered up.
$AN_POWERON_TIME	Time since the last normal control powerup ("Warm start" in min.)	Counts the time since the last normal control powerup. It is automatically reset to the default values on each normal control powerup.
The timers of the system variables are always active. All system variables are reset to default values on each control power-up.		

Channelspecific system variables

The following channel-specific system variables are available

Several channel-specific system variables are available and can be activated via machine data. Each active runtime measurement is interrupted automatically by a program status "Program running" and an active override = ZERO.

Dry run feedrate and program testing
The behavior of the active time measurements during the functions: "Dry run feedrate" and "Program testing" can be specified via machine data.

The following channel-specific system variables are available:

Names	Meaning	Description
$AC_OPERATING_TIME	Total runtime of NC programs in Automatic mode (in s)	Totals the accumulated runtimes of all programs between NC Start and end of program/NC reset. It is automatically reset each time the control is powered up.
$AC_CYCLE_TIME	Runtime of the selected NC program (in s; always only one active per channel)	The runtime between the NC Start and the end of program/NC reset is measured in the selected NC program. It is reset automatically when a new NC program is started.
$AC_CUTTING_TIME	Tool operating time (in s)	The measurement produces the runtime of the path axes (at least one is active) without active rapid traverse in all NC programs between NC Start and program end/NC reset with active tool. The measurement is also interrupted when a dwell time is active. It is automatically reset to the default values each time the control is powered up.

Names	Meaning	Description
The timers of the system variables must be activated channel-specifically: MD27860 $MC_PROCESSTIMER_MODE The system variables are reset to default values on each control power-up.		
The timers are automatically activated with the standard machine data for SINUMERIK 802D.		

Note

The system variables can be accessed at any time by reading from the HMI operator interface.

Examples

- Activating the runtime measurement for the active NC program (no measurement with active dry run feedrate and program testing):
  $MC_PROCESSTIMER_MODE = 'H2'
- Activating the measurement for the tool action time (measurement also with dry run feedrate and program testing):
  $MC_PROCESSTIMER_MODE = 'H34'
- Activating the measurement for the total runtime and the tool action time (measurement also with program test):
  $MC_PROCESSTIMER_MODE = 'H25'

2.12.3 Workpiece counter

Functionality

The "Workpiece counter" function makes available various channel-specific system variables specifically for counting workpieces:

- $AC_REQUIRED_PARTS
  Number of workpieces to be completed (setpoint workpieces)
- $AC_TOTAL_PARTS
  Number of completed workpieces in total (actual total)
- $AC_ACTUAL_PARTS
  Number of completed workpieces (actual workpieces)
- $AC_SPECIAL_PARTS
  Number of workpieces specified by the user

Value range: 0 to 999 999 999

Access mode: Write/read

Note

The "Workpiece counter" function is independent of the tool management functions.

Activation

The workpiece counters are activated or the reset timing and counting algorithm are specified via the following two channel-specific machine data:

MD27880 $MC_PART_COUNTER (activation of workpiece counters)		
Bit	**Value**	**Meaning**
0	1	$AC_REQUIRED_PARTS is active
1	0	Alarm/VDI output with: $AC_REQUIRED_PARTS == $AC_ACTUAL_PARTS
	1	Alarm/VDI output with: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS
4	1	$AC_TOTAL_PARTS is active
5	1	With M2/M30: $AC_TOTAL_PARTS += 1
6	1	Equality under M function of $MC_PART_COUNTER_MCODE[**0**]: $AC_TOTAL_PARTS += 1
8	1	$AC_ACTUAL_PARTS is active
9	0	With M2/M30: $AC_ACTUAL_PARTS += 1
	1	Equality under M function of $MC_PART_COUNTER_MCODE[**1**]: $AC_ACTUAL_PARTS += 1
12	1	$AC_SPECIAL_PARTS is active
13	0	With M2/M30: $AC_SPECIAL_PARTS += 1
	1	Equality under M function of $MC_PART_COUNTER_MCODE[**2**]: $AC_SPECIAL_PARTS += 1

MD27882 $MC_PART_COUNTER_MCODE[<Index>] (workpiece counting via user-specific M function)	
Index	**Meaning**
0	Value of M function with which $AC_TOTAL_PARTS is counted up.
1	Value of M function with which $AC_ACTUAL_PARTS is counted up.
2	Value of M function with which $AC_SPECIAL_PARTS is counted up.

Boundary conditions

- All workpiece counters are set to the default values on control power-up and can be read and written independent of their activation.
- If $AC_REQUIRED_PARTS = 0, no ID check is performed with the counter pulse for $AC_ACTUAL_PARTS or $AC_SPECIAL_PARTS even if the MD bit is set.

Examples

Activation of workpiece counter $AC_REQUIRED_PARTS
MD27880 $MC_PART_COUNTER = 'H3'
Alarm displayed with: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

Activation of workpiece counter $AC_TOTAL_PARTS
MD27880 $MC_PART_COUNTER = 'H10'
MD27882 $MC_PART_COUNTER_MCODE[0] = 80
For each M02: $AC_TOTAL_PARTS += 1
Note: $MC_PART_COUNTER_MCODE[0] has no meaning.

Activation of workpiece counter $AC_ACTUAL_PARTS
MD27880 $MC_PART_COUNTER = 'H300'
MD27882 $MC_PART_COUNTER_MCODE[1] = 17
For each M17: $AC_ACTUAL_PARTS += 1

Activation of workpiece counter $AC_SPECIAL_PARTS
MD27880 $MC_PART_COUNTER = 'H3000'
MD27882 $MC_PART_COUNTER_MCODE[2] = 77
For each M77: $AC_SPECIAL_PARTS += 1

Deactivation of workpiece counter $AC_ACTUAL_PARTS
MD27880 $MC_PART_COUNTER = 'H200'
MD27882 $MC_PART_COUNTER_MCODE[1] = 50
Note: $AC_TOTAL_PARTS is not active

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

Mode change/NC RESET

The counters are not affected by a mode change or NC RESET.

Supplementary Conditions

3

3.1 Supplementary conditions

There are no supplementary conditions to note.

4 Examples

4.1 Examples

The examples appear with the descriptions in the individual sections of the function descriptions.

Data Lists

5

5.1 Machine data

5.1.1 General machine data

5.1.1.1 HMIspecific machine data

Number		Identifier: $MM_	Description
ADV	EMB		
9421	9421	MA_AXES_SHOW_GEO_FIRST	Display geo axes of channel first
9422	9422	MA_PRESET_MODE	PRESET/basic offset in JOG.
9423	9423	MA_MAX_SKP_LEVEL	Maximum number of skip levels

5.1.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10010	ASSIGN_CHAN_TO_MODE_GROUP	Channel valid in mode group
10280	PROG_FUNCTION_MASK	Compare commands ">" and "<" compatible to SW 6.3
10617	FRAME_SAVE_MASK	Response of frames to subroutine return jump with SAVE
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
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 tool change cycle for T function
10718	M_NO_FCT_CYCLE_PAR	M function replacement with parameters
10719	T_NO_FCT_CYCLE_MODE	Parameter assignment for T function replacement

Number	Identifier: $MN_	Description
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 for "ASUP_START_MASK effective"
11610	ASUP_EDITABLE	Activation of a user ASUB for RET/REPOS
11612	ASUP_EDIT_PROTECTION_LEVEL	Protection level of user-specific ASUB
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 one program level
18362	MM_EXT_PROG_NUM	Number of external program levels (DRAM)

5.1.2 Channel-specific machine data

5.1.2.1 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	Eventdriven 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 readin 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

Number	Identifier: $MC_	Description
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value with 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	Max. no. of dummy blocks with no traversing movements
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20400	LOOKAH_USE_VELO_NEXT_BLOCK	Look Ahead to programmed following block velocity
20430	LOOKAH_NUM_OVR_POINTS	Number of override switch points for Look Ahead
20440	LOOKAH_OVR_POINTS	Override switch points for LookAhead
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity
20600	MAX_PATH_JERK	Pathrelated maximum jerk
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements
20700	REFP_NC_START_LOCK	NC start disable without reference point
20750	ALLOW_GO_IN_G96	G0 logic in G96
20800	SPF_END_TO_VDI	Subprogram end to PLC
21000	CIRCLE_ERROR_CONST	Circle end point monitoring constant
21010	CIRCLE_ERROR_FACTOR	Circle end point monitoring factor
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system for automatic Frame definition
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour
21210	SETINT_ASSIGN_FASTIN	NCK input bytes for interrupts
21202	LIFTFAST_WITH_MIRROR	Lift fast with mirror
21250	START_INDEX_R_PARAM	Number of first channelspecific R parameter

5.1.2.2 Block search

Number	Identifier: $MC_	Description
20128	COLLECT_TOOL_CHANGE	Collect tool changes during 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	Enables MD 22620: 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

5.1.2.3 Reset response

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Initial setting at RESET
20112	START_MODE_MASK	Initial setting at special NC Start after power-up and at RESET
20118	GEOAX_CHANGE_RESET	Allow automatic geometry axis change
20120	TOOL_RESET_VALUE	Tool whose length compensation is selected during powerup (Reset/part program end)
20121	TOOL_PRESEL_RESET_VALUE	Preselected tool whose length compensation is selected in powerup (Reset/part program end)
20130	CUTTING_EDGE_RESET_VALUE	Tool cutting edge on power-up (Reset/part program end)
20140	TRAFO_RESET_VALUE	Active transformation on RESET
20150	GCODE_RESET_VALUES	Reset G groups
20152	GCODE_RESET_MODE	G code basic setting at RESET
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

5.1.2.4 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
22260	AUXFU_E_SYNC_TYPE (available soon)	Output timing of E functions.
22300	AUXFU_AT_BLOCK_SEARCH_END	Output of auxiliary functions after block search
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET
22410	F_VALUES_ACTIVE_AFTER_RESET	F function active after reset
22500	GCODE_OUTPUT_TO_PLC	G functions to PLC
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

5.1.2.5 Transformation definitions

Number	Identifier: $MC_	Description
24100	TRAFO_TYPE_1	Definition of transformation 1 in channel
24110	TRAFO_AXES_IN_1	Axis assignment for transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Assignment between GEO axis and channel axis for transformation 1
24200	TRAFO_TYPE_2	Definition of transformation 2 in channel
24210	TRAFO_AXES_IN_2	Axis assignment for transformation 2
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Assignment between GEO axis and channel axis for transformation 2
24300	TRAFO_TYPE_3	Definition of transformation 3 in channel
24310	TRAFO_AXES_IN_3	Axis assignment for transformation 3
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Assignment between GEO axis and channel axis for transformation 3
24400	TRAFO_TYPE_4	Definition of transformation 4 in channel
24410	TRAFO_AXES_IN_4	Axis assignment for transformation 4
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Assignment between GEO axis and channel axis for transformation 4
24430	TRAFO_TYPE_5	Definition of transformation 5 in channel
24432	TRAFO_AXES_IN_5	Axis assignment for transformation 5
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Assignment between GEO axis and channel axis for transformation 5
24440	TRAFO_TYPE_6	Definition of transformation 6 in channel
24442	TRAFO_AXES_IN_6	Axis assignment for transformation 6
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Assignment between GEO axis and channel axis for transformation 6
24450	TRAFO_TYPE_7	Definition of transformation 7 in channel
24452	TRAFO_AXES_IN_7	Axis assignment for transformation 7
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Assignment between GEO axis and channel axis for transformation 7
24460	TRAFO_TYPE_8	Definition of transformation 8 in channel
24462	TRAFO_AXES_IN_8	Axis assignment for transformation 8
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Assignment between GEO axis and channel axis for transformation 8
24500	TRAFO5_PART_OFFSET_1	Offset vector of 5-axis transformation 1
24510	TRAFO5_ROT_AX_OFFSET_1	Position offset of rotary axes 1/2 for 5axis transformation 1
24520	TRAFO5_ROT_SIGN_IS_PLUS_1	Sign of rotary axis 1/2 for 5axis transformation 1
24530	TRAFO5_NON_POLE_LIMIT_1	Definition of pole limit for 5-axis transformation 1
24540	TRAFO5_POLE_LIMIT_1	Pole end angle tolerance for interpolation for 5axis transformation 1
24550	TRAFO5_BASE_TOOL_1	Vector of base tool on activation of 5-axis transformation 1

Number	Identifier: $MC_	Description
24560	TRAFO5_JOINT_OFFSET_1	Vector of kinematic offset for 5-axis transformation 1
24600	TRAFO5_PART_OFFSET_2	Offset vector of 5-axis transformation 2
24610	TRAFO5_ROT_AX_OFFSET_2	Position offset of rotary axes 1/2 for 5axis transformation 2
24620	TRAFO5_ROT_SIGN_IS_PLUS_2	Sign of rotary axis 1/2 for 5axis transformation 2
24630	TRAFO5_NON_POLE_LIMIT_2	Definition of pole limit for 5-axis transformation 2
24640	TRAFO5_POLE_LIMIT_2	Pole end angle tolerance for interpolation for 5axis transformation 2
24650	TRAFO5_BASE_TOOL_2	Vector of base tool on activation of 5-axis transformation 2
24660	TRAFO5_JOINT_OFFSET_2	Vector of kinematic offset for 5-axis transformation 2

5.1.2.6 Memory settings

Number	Identifier: $MC_	Description
25000	REORG_LOG_LIMIT	Percentage of IPO buffer for log file enable
28000	MM_REORG_LOG_FILE_MEM	Memory size for REORG (DRAM)
28010	MM_NUM_REORG_LUD_MODULES	Number of blocks for local user variables for REORG (DRAM)
28020	MM_NUM_LUD_NAMES_TOTAL	Number of local user variables (DRAM)
28030	MM_NUM_LUD_NAMES_PER_PROG	Number of local user variables per program (DRAM)
28040	MM_LUD_VALUES_MEM (available soon)	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 (available soon)	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	Dimension basic block display
28402	MM_ABSBLOCK_BUFFER[2]	Dimension size of upload buffer
28500	MM_PREP_TASK_STACK_SIZE	Stack size of preparation task (DRAM)
28510	MM_IPO_TASK_STACK_SIZE	Stack size of IPO task (DRAM)

5.1.2.7 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

5.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
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

5.2 Setting data

5.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42010	THREAD_RAMP_DISP	Starting and deceleration distance of feed axis in 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	Name of an external program path for subroutine call EXTCALL
42750	ABSBLOCK_ENABLE	Enable basic block display
42990	MAX_BLOCKS_IN_IPOBUFFER	Control of max. number of blocks in interpolation buffer

5.3 Signals

5.3.1 Signals to NC

DB number	Byte.Bit	Description
10	56.1	EMERGENCY STOP

5.3.2 Signals to mode group

DB number	Byte.Bit	Description
11, ...	0.0	AUTOMATIC mode
11, ...	0.1	MDA mode
11, ...	0.2	JOG mode
11, ...	0.4	Mode change disable
11, ...	0.5	Mode group Stop
11, ...	0.6	Mode group Stop axes plus spindles
11, ...	0.7	Mode group RESET
11, ...	1.0	TEACHIN machine function
11, ...	1.1	Machine function REPOS
11, ...	1.2	Machine function REF

5.3.3 Signals from mode group

DB number	Byte.Bit	Description
11, ...	4.0	Selected mode AUTOMATIC
11, ...	4.1	Selected mode MDA
11, ...	4.2	Selected mode JOG
11, ...	5.0	Selected machine function TEACH-IN
11, ...	5.1	Selected machine function REPOS
11, ...	5.2	Selected machine function REF
11, ...	6.0	Active mode AUTOMATIC
11, ...	6.1	Active mode MDA
11, ...	6.2	Active mode JOG
11, ...	6.3	Mode group ready
11, ...	6.4	Mode group has been reset
11, ...	6.5	Internal Jog is active
11, ...	6.7	All channels in Reset state

DB number	Byte.Bit	Description
11, ...	7.0	Active machine function TEACH-IN
11, ...	7.1	Active machine function REPOS
11, ...	7.2	Active machine function REF

5.3.4 Signals to channel

DB number	Byte.Bit	Description
21, ...	0.3	Activate DRF
21, ...	0.4	Activate single block
21, ...	0.5	Activate M01
21, ...	0.6	Activate dry run feed
21, ...	1.0	Activate referencing
21, ...	1.6	PLC action ended
21, ...	1.7	Activate program test
21, ...	2.0 – 2.7	Skip-block levels: /0 to /7
21, ...	7.0	NC Start disable
21, ...	7.1	NC Start
21, ...	7.2	NC Stop at block limit
21, ...	7.3	NC stop
21, ...	7.4	NC Stop axes plus spindles
21, ...	7.7	Reset
21, ...	31.0 - 31.2	REPOSPATHMODE
21, ...	31.4	REPOSMODEEDGE

5.3.5 Signals from channel

DB number	Byte.Bit	Description
21, ...	24.3	DRF selected
21, ...	24.4	Select associated M01
21, ...	24.5	M01 selected
21, ...	24.6	Dry run feedrate selected
21, ...	25.3	Feedrate override selected for rapid traverse
21, ...	25.7	Program test selected
21, ...	26.0 – 26.7	Skip-block selected: /0 - /7
21, ...	27.0	Skip-block selected
21, ...	27.1	Skip-block selected
21, ...	31.6	Skip-block active /8
21, ...	31.7	Skip-block active /9

DB number	Byte.Bit	Description
21, ...	32.3	Action block active
21, ...	32.4	Approach block active
21, ...	32.5	M00/M01 active
21, ...	32.6	Last action block active
21, ...	33.0	Referencing active
21, ...	33.4	Block search active
21, ...	33.5	M02/M30 active
21, ...	33.6	Transformation active
21, ...	33.7	Program test active
21, ...	35.0	Program status: Running
21, ...	35.1	Program status: Wait
21, ...	35.2	Program status: Stopped
21, ...	35.3	Program status: Interrupted
21, ...	35.4	Program status: Aborted
21, ...	35.5	Channel status: Active
21, ...	35.6	Channel status: Interrupted
21, ...	35.7	Channel status: Reset
21, ...	208 - 271	Number of the active G function of G function group 1 – n (binary)
21, ...	318.0	ASUB is stopped
21, ...	318.1	Block search via program test is active
21, ...	319.0	REPOSMODEEDGEACKN
21, ...	319.1- 319.3	Repos Path Mode Ack: 0 - 2
21, ...	319.5	Repos DEFERRAL Chan

5.3.6 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	10.0	REPOSDELAY

5.3.7 Signals from axis/spindle

DB number	Byte.Bit	Description
31, ...	70.0	REPOS offset
31, ...	70.1	REPOS offset valid
31, ...	70.2	REPOS Delay Ack
31, ...	72.0	REPOSDELAY
31, ...	76.2	Path axis

Index

$

$AC_ACTUAL_PARTS, 2-160
$AC_REQUIRED_PARTS, 2-160
$AC_SPECIAL_PARTS, 2-160
$AC_TOTAL_PARTS, 2-160

A

Acceptance timing, 2-48
Action single block, 2-20
Asynchronous subroutines (ASUBs), 2-102
Automatic Operating Mode, 2-159
Autonomous singleaxis operations, 2-74

B

BLOCKREADINHIBIT_ON, 2-102
BLOCKSEARCHRUN_NEWCONF, 2-102

C

Calling the ASUB outside program operation, 2-107
Cascaded block search, 2-32
Channel
    Channel status, 2-86
Control
    Normal powerup, 2-159
Control powerup, 2-159
    With default values, 2-159
Control REPOS with VDI interface signals, 2-50
Counting
    Pulse, 2-161
Cross-mode synchronized actions, 2-6

D

DB11, ...
    D0.4, 2-12
    DBX0.0, 2-6
    DBX0.1, 2-6
    DBX0.2, 2-6
    DBX0.5, 2-4
    DBX0.6, 2-4
    DBX0.7, 2-4
    DBX1.0, 2-8
    DBX1.1, 2-8
    DBX1.2, 2-8
    DBX4.0, 2-6
    DBX4.1, 2-6
    DBX4.2, 2-6
    DBX5.0, 2-7
    DBX5.1, 2-7
    DBX5.2, 2-7
    DBX6.0, 2-7
    DBX6.1, 2-7
    DBX6.2, 2-7
    DBX6.3, 2-5
    DBX6.7, 2-5
    DBX7.0, 2-8
    DBX7.1, 2-8
    DBX7.2, 2-8
DB11, ...
    DBX0.7, 2-84
DB11.
    DBX0.0, 2-9
    DBX26.4, 2-10
    DBX26.5, 2-10
    DBX46.4, 2-10
    DBX46.5, 2-10
    DBX6.0, 2-10
    DBX6.1, 2-10
    DBX6.2, 2-10
    DBX6.4, 2-10
    DBX6.5, 2-10
    DBX7.0, 2-10
    DBX7.1, 2-10
    DBX7.2, 2-10
DB21, ...
    D35.5, 2-82
    DB32.6, 2-30
    DBB35, 2-39
    DBX0.4, 2-21
    DBX0.6, 2-22
    DBX1.6, 2-27, 2-29
    DBX1.7, 2-19
    DBX2.0, 2-24, 2-82

DBX24.6, 2-22
DBX25.7, 2-19
DBX26.0, 2-24
DBX31.0-31.2, 2-48, 2-50, 2-55
DBX31.4, 2-48, 2-49, 2-50
DBX310.1-319.3, 2-50
DBX318.1, 2-42
DBX319.0, 2-49, 2-50
DBX319.1-319.3, 2-49
DBX319.5, 2-50, 2-53
DBX32.3, 2-27
DBX32.4, 2-27, 2-28
DBX32.6, 2-27
DBX33.4, 2-27
DBX33.7, 2-19
DBX35.0, 2-81
DBX35.3, 2-21, 2-82
DBX35.5, 2-81
DBX36.6, 2-29
DBX36.7, 2-29
DBX7.1, 2-18, 2-20, 2-22, 2-27
DBX7.2, 2-82
DBX7.3, 2-82
DBX7.4, 2-82
DBX7.5, 2-17
DB21, ...
DBX319.1-319.3, 2-52
DB21, ...
DBX7.7, 2-84
DB21, ...
DBX35.7, 2-84
DB21, ...
DBX35.4, 2-96
DB21, ...
DBX35. 7, 2-96
DB21, ...
DBX318.0, 2-110
DB21, ...
DBX318.0, 2-110
DB21,-...-
DBX319.1-319.3, 2-51
DB21-DB30, ...
DBB35, 2-94, 2-95
DB31, ...
DBB28.7, 2-16
DBX10.0, 2-45, 2-47, 2-48, 2-50, 2-51
DBX2.2, 2-28, 2-46
DBX60.6, 2-19
DBX60.7, 2-19
DBX70, 2-53
DBX70.0, 2-50, 2-52
DBX70.1, 2-50
DBX70.2, 2-50, 2-52
DBX72.0, 2-50
DBX76.4, 2-50, 2-53
DB31, ...
DBX70.2, 2-51
DB31, ...
DBX70.1, 2-52
DB31, ...
DBX70.0, 2-52
DB31, ...
DBX10.0, 2-55
DB31, ...
DBX3.7, 2-73
Decoding single block, 2-20
Decoding single block SBL2 with implicit preprocessing stop, 2-116
DELDISTOGO_SYNC, 2-101
Dry run feedrate, 2-22

E

ESR, 2-102
Eventcontrolled program sequences, 2-91
Execute external subroutine, 2-131, 2-132
Execution from external source, 2-131
Execution from external source via V24 interface, 2-131
EXT_ZERO_POINT, 2-102
External subroutine, 2-132

F

Feed stop, 2-86
Function selection (via operator interface or PLC), 2-121

G

Gap, 2-4

H

Hold block, 2-62

I

ID check, 2-161
Implicit preprocessing stop, 2-67
INIT_SYNC, 2-102
Initial setting
Program operation, 2-80
Interpolator, 2-13
Interrupt, 2-101

Interrupt pointer
Automatically., 2-65
Interrupt routines, 2-102

L

LIFTFAST, 2-104

M

Main technological application, 2-15
MD10010, 2-1, 2-3, 2-4
MD10702, 2-21, 2-28
MD10707, 2-40
MD10708, 2-40, 2-41
MD10714, 2-145
MD10715, 2-145
MD10716, 2-145
MD10717, 2-145
MD10718, 2-145
MD10719, 2-145
MD10735, 2-9
MD10804, 2-145
MD10806, 2-145
MD10814, 2-145
MD11411, 2-17
MD11450, 2-29, 2-30, 2-31, 2-32
MD11470, 2-44, 2-45, 2-46, 2-47
MD11602, 2-109
MD11604, 2-109
MD11717, 2-145
MD18800, 2-145
MD20000, 2-14
MD20094, 2-145
MD20095, 2-145
MD20110, 2-108
MD20112, 2-41, 2-108
MD20150, 2-14
MD22254, 2-146
MD22256, 2-146
MD22560, 2-146
MD22620, 2-42, 2-76
MD22621, 2-41
MD26008, 2-146
MD26012, 2-146
MD27800, 2-15
MD27880, 2-161
MD27882, 2-161
Mirroring
Retraction direction (lift fast), 2-104
MMCCMD, 2-102
Modal execution from external source, 2-131
Mode group
Number, 2-3

N

NC
NC instruction, 2-81
NC Start, 2-81, 2-159
Program, 2-160
Reset, 2-159
Stop, 2-86
NEWCONF_PREP_STOP, 2-102

O

Operating modes, 2-5
Change, 2-12
Interlocks, 2-11
Monitoring functions, 2-11
Operator panel reset, 2-93, 2-95
Override, 2-159
OVERSTORE_BUFFER_END_REACHED, 2-101

P

Part program
End, 2-135
Interruption, 2-82
Skipping of specific part program blocks, 2-23
Start, 2-92, 2-135
Starting, 2-88
Part program end, 2-92
Part program start, 2-135
Part program start and part program end, 2-94
Power-up, 2-94
PREP_STOP, 2-101
PROG_END, 2-101
PROGCANCELSUB, 2-101
PROGMODESLASHOFF, 2-102
PROGMODESLASHON, 2-102
Program
Program action, 2-87
Program operation mode, 2-79
Program status, 2-84
Program test, 2-17
Program display modes, 2-125
Program execution without setpoint outputs, 2-18
Program section
Repetition, 2-89
PROGRESETREPEAT, 2-101

R

Rapid traverse, 2-159
Reaching simulated target point for LEAD with JOG, 2-71
Read-in disable, 2-86
REPOS acknowledgments, 2-50
REPOS operation, 2-44
Reposition positioning axes, 2-47
Repositioning neutral axes after SERUPRO, 2-47
Repositioning point, 2-55
Reset, 2-86, 2-101
RESET, 2-135
RESET command, 2-83
RETREAT_MOVE_THREAD, 2-102
Runtime
  Measurement, 2-160
  Of the NC program, 2-159
  Of the path axes, 2-159

S

SD42100, 2-22
SD42101, 2-22
SD42444, 2-27
Selection, 2-80
Self-acting SERUPRO, 2-60
SERUPRO approach, 2-40
SERUPRO ASUP, 2-40
SERUPRO operation, 2-40
SERUPRO supports the following NC functions, 2-68
SerurpoMasterChan, 2-60
SET_USER_DATA, 2-102
SETINT, 2-104
Single block, 2-86
  Program operation mode, 2-20
  Suppression, 2-116, 2-119
SINGLEBLOCK_DECODIER, 2-101
SINGLEBLOCK_IPO, 2-101
SINGLEBLOCKSTOP, 2-101
Spindle stop, 2-86
STOP_ALARM, 2-102
STOPALL, 2-101
STOPATEND_ALARM, 2-102
STOPATIPOBUF_EMPTY_ALARM_REORG, 2-102
STOPATIPOBUFFER_ISEMPTY_ALARM, 2-102
STOPPROG, 2-101
STOPPROGATASUPEND, 2-102
STOPPROGATBLOCKEND, 2-101
STOPRUN, 2-102
System variable
  Channel-specific, 2-159
  NCK-specific, 2-159
SYSTEM_SHUTDOWN, 2-102

T

Tangential control, 2-73
TEACH IN, 2-6
Timers
  Channel-specific, 2-159
Tool
  Management, 1-4
  Operating time, 2-159

U

Updating the REPOS offset within the scope, 2-52
Userdefined ASUB, 2-43
Userdefined system ASUBs, 2-111

W

WAITE, 2-102
WAITM, 2-102
WAITMC, 2-102
Workpiece
  Counter: Example, 2-162
Workpiecerelated actualvalue system
  RESET and power-up response, 2-138

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Axis Types, Coordinate Systems, Frames (K2)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Axes 1-1

1.2 Coordinate systems 1-3

1.3 Frames 1-4

2 Detailed Description 2-1

2.1 Axes 2-1
2.1.1 Overview 2-1
2.1.2 Machine axes 2-2
2.1.3 Channel axes 2-4
2.1.4 Geometry axes 2-4
2.1.5 Replaceable geometry axes 2-4
2.1.6 Special axes 2-9
2.1.7 Path axes 2-9
2.1.8 Positioning axes 2-10
2.1.9 Main axes 2-11
2.1.10 Synchronized axes 2-12
2.1.11 Axis configuration 2-13
2.1.12 Link axes 2-17

2.2 Zeros and reference points 2-20
2.2.1 Reference points in working space 2-20
2.2.2 Position of coordinate systems and reference points 2-21

2.3 Coordinate systems 2-23
2.3.1 Overview 2-23
2.3.2 Machine coordinate system (MCS) 2-24
2.3.3 Basic coordinate system (BCS) 2-26
2.3.4 Additive offsets 2-28
2.3.5 Basic zero system (BZS) 2-31
2.3.6 Settable zero system (SZS) 2-33
2.3.7 Configurable settable zero system (SZS) 2-34
2.3.8 Workpiece coordinate system (WCS) 2-37

2.4 Frames 2-38
2.4.1 Coordinate axes, zeros and reference points 2-38
2.4.2 Frames 2-39
2.4.3 Frame components 2-40
2.4.3.1 Translation 2-40
2.4.3.2 Fine offset 2-40
2.4.3.3 Rotations for geometry axes 2-41
2.4.3.4 Scaling 2-46
2.4.3.5 Mirroring 2-46
2.4.3.6 Chain operator 2-47
2.4.3.7 Programmable axis identifiers 2-47
2.4.3.8 Coordinate transformation 2-48

2.4.4 Frames in data management and active frames ... 2-49
2.4.4.1 Overview ... 2-49
2.4.4.2 Activating data management frames ... 2-51
2.4.4.3 NCU global frames ... 2-52
2.4.5 Frame chain and coordinate systems ... 2-52
2.4.5.1 Overview ... 2-52
2.4.5.2 Configurable SZS ... 2-54
2.4.5.3 Manual traverse in the SZS coordinate system ... 2-55
2.4.5.4 Suppression of frames ... 2-56
2.4.6 Frame chain frames ... 2-58
2.4.6.1 Overview ... 2-58
2.4.6.2 Settable frames $P_UIFR[n] ... 2-58
2.4.6.3 Channel basic frames $P_CHBFR[n] ... 2-59
2.4.6.4 NCU global basic frames $P_NCBFR[n] ... 2-60
2.4.6.5 Complete basic frame $P_ACTBFRAME ... 2-62
2.4.6.6 Programmable frame $P_PFRAME ... 2-63
2.4.6.7 Channel-specific system frames ... 2-65
2.4.6.8 $P_ACTFRAME ... 2-67
2.4.7 Implicit frame changes ... 2-68
2.4.7.1 Frames and switchover of geometry axes ... 2-68
2.4.7.2 Frame for selection and deselection of transformations ... 2-71
2.4.7.3 Adapting active frames ... 2-90
2.4.8 Predefined frame functions ... 2-91
2.4.8.1 Inverse frame ... 2-91
2.4.8.2 Additive frame in frame chain ... 2-94
2.4.9 Functions ... 2-95
2.4.9.1 Setting zeros, workpiece measuring and tool measuring ... 2-95
2.4.9.2 Zero offset external via system frames ... 2-95
2.4.9.3 Toolholder ... 2-96
2.4.10 Subroutine return with SAVE ... 2-108
2.4.11 Data backup ... 2-108
2.4.12 Positions in the coordinate system ... 2-109
2.4.13 Control system response ... 2-110
2.4.13.1 Power on ... 2-110
2.4.13.2 Mode change ... 2-110
2.4.13.3 Reset, program end ... 2-111
2.4.13.4 Response at part-program start ... 2-113
2.4.13.5 Block search ... 2-113
2.4.13.6 Repos ... 2-113

2.5 Workpiece-related actual-value system ... 2-114
2.5.1 Overview ... 2-114
2.5.2 Use of workpiece-related actual-value system ... 2-115
2.5.3 Special reactions ... 2-117

3 Supplementary Conditions ... 3-1

3.1 Supplementary conditions ... 3-1

4 Examples ... 4-1

4.1 Axes ... 4-1

4.2 Coordinate systems ... 4-4

4.3 Frames ... 4-5

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 Memory specific machine data 5-1
5.1.2 NC-specific machine data 5-2
5.1.3 Channel-specific machine data 5-2
5.1.4 Axis/spindle-specific machine data 5-3

5.2 Setting data 5-4
5.2.1 Channel-specific setting data 5-4

5.3 System variables 5-4

5.4 Signals 5-5
5.4.1 Signals from channel 5-5
5.4.2 Signals to axis/spindle 5-6

Index Index-1

Tables

Table 2-1 Zeros and reference points 2-20

Brief Description

1

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 AXCONF_GEOAX_ASIGN_TAB

MD20060 AXCONF_GEOAX_NAME_TAB

MD20070 AXCONF_MACHAX_USED

MD20080 AXCONF_CHANAX_NAME_TAB

MD10000 AXCONF_MACHAX_NAME_TAB

MD35000 SPIND_ASSIGN_TO_MACHAX

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:
/FB2/ Multiple 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:
/FB2/ Multiple Operator Panels on Multiple NCUs, Distributed Systems (B3)

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 coordinate system (BCS) 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.

1.3 Frames

FRAME

A FRAME is a closed calculation rule that translates one Cartesian coordinate system into another.

FRAME components

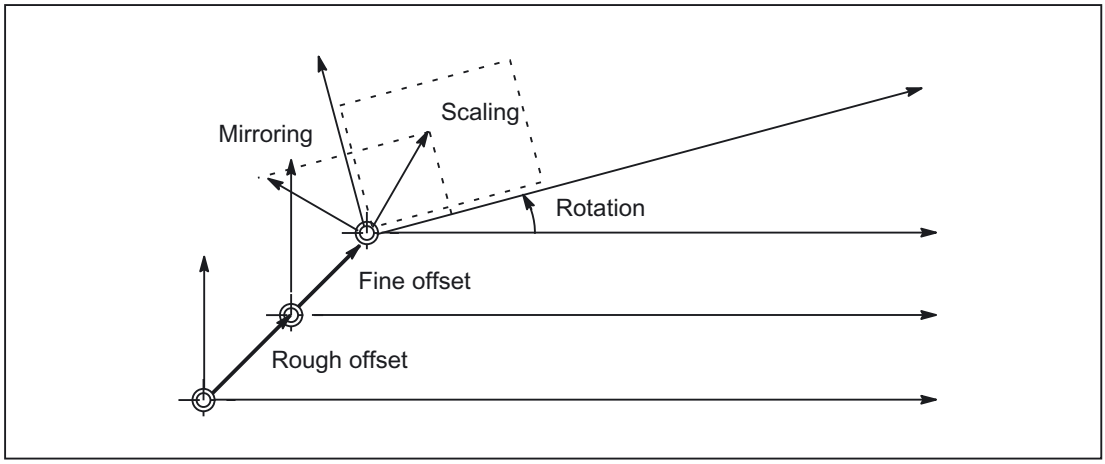


Fig. 1-1 FRAME components

A FRAME consists of the following components:

Offset	Rough offset	Programmable with: • TRANS • ATRANS (additive translation component) • CTRANS (zero offset for multiple axes) and • G58 (axial zero offset)
	Fine offset	Programmable with: • CFINE and • G59 (axial zero offset)
Rotation		Programmable with: • ROT/ROTS • AROT/AROTS and • CROTS
Scaling		Programmable with: • SCALE and • ASCALE
Mirroring		Programmable with: • MIRROR and • AMIRROR

Features in relation to axes

The rough and fine offsets, scaling and mirroring can be programmed for geometry and special axes. A rotation can also be programmed for geometry axes.

Rough and fine offsets

The translation component of FRAMES comprises:

- Rough offset with TRANS, ATRANS and CTRANS

  The rough 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

G58 and G59 can be programmed to replace the rough and fine offsets of the programmable frame on an axial basis. These functions can only be used when the fine offset is configured.

- Rough offset with G58

  G58 changes only the absolute translation component (rough 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, which 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.

ASCALE must be programmed if a new scaling is to be based on a previous scaling, rotation, translation or mirroring.

Mirroring

You can set the axis, about which mirroring is performed, via
MD10610 MIRROR_REF_AX:

MD10610 = 0:	Mirroring is performed around the programmed axis.
MD10610 = 1 or 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

The current FRAME is composed of the total basic frame, the settable FRAME, the 4 system frames and the programmable FRAME.

The current complete frame is calculated according to the formula below:

$P_ACTFRAME = $P_SETFRAME : $P_EXTFRAME : $P_PARTFRAME : $P_ACTBFRAME : $P_IFRAME : $P_TOOLFRAME : $P_PFRAME

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 following are set via the setting data:
SD42440 FRAME_OFFSET_INCR_PROG

Value = 1	Zero offset is applied on FRAME and incremental programming of an axis (= default setting).
Value = 0	Only the programmed path is traversed.

Suppression of frames

Current frames can be suppressed with the following instructions:

G53	Current zero offset (ZO)
G153	Current frame, incl. basic frame
SUPA	Current ZO, incl. programmed offsets

NCU global basic frames

For rotary indexing machine technology, for example, a channel must be used to define frames for other channels. These cross-channel frames are shown in the "NCU global basic frames" below.

Properties of the NCU global basic frames:

- Can be read and written from all channels.
- Can only be activated in the channels.
- Up to 16 NCU global basic frames are available.

Global frames can be used to apply offsets, scale factors and mirroring operations to channel and machine axes.

All basic frames (up to 16 global and 16 channel-specific) are chained to produce the total basic frame. The standard configuration is designed for at least one basic frame per channel.

Settable frames can be defined as either NCU global or channel-specific.

Channel coordination

With NCU global frames, the user must ensure channel coordination and activation of the frames (e.g., using the `WAITMC` command), to allow the frames to be calculated at the desired point in the program.

Cross-channel activation of frames is not supported.

Detailed Description

2

2.1 Axes

2.1.1 Overview

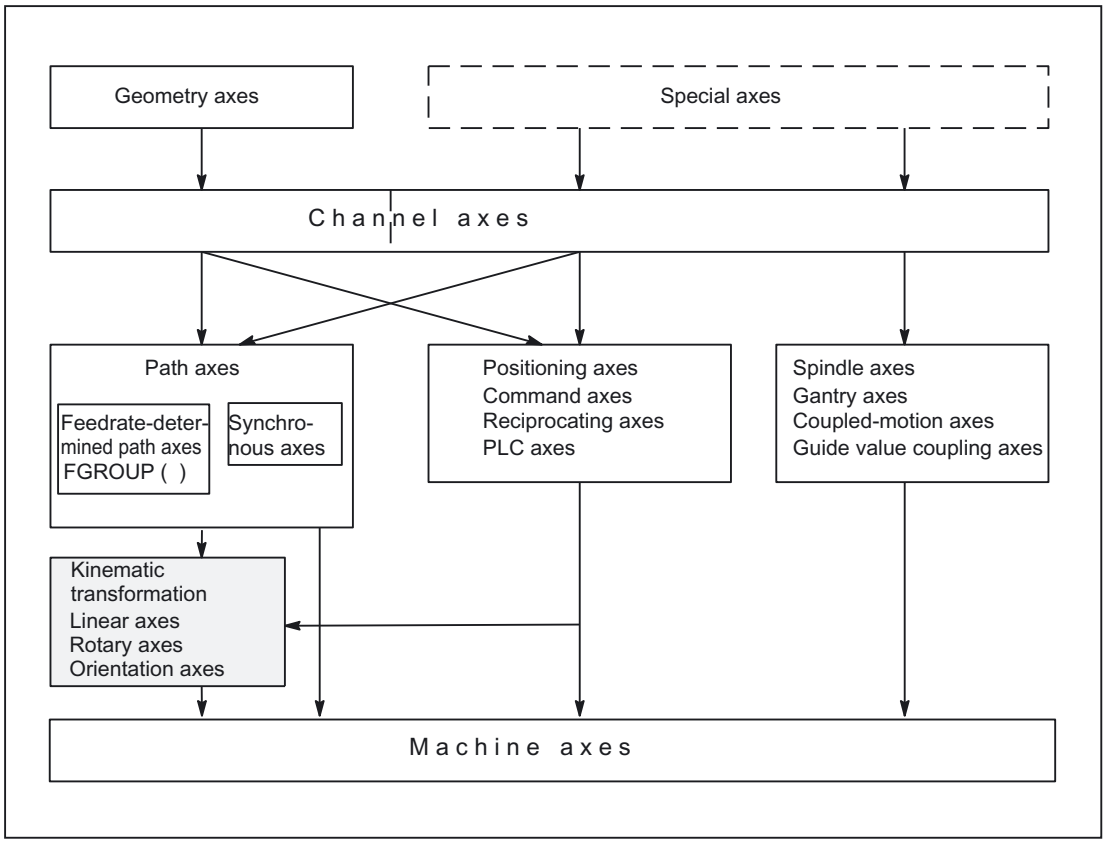


Fig. 2-1 Relationship between geometry axes, special axes and machine axes

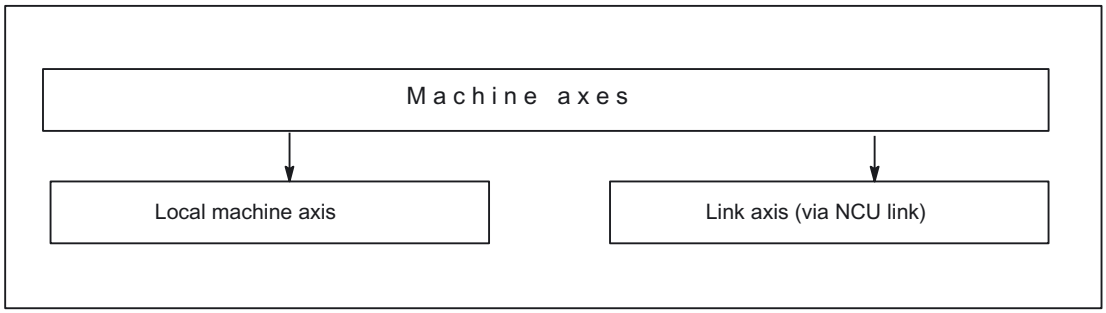


Fig. 2-2 Local and external machine axes (link axes)

2.1.2 Machine axes

Meaning

Machine axes are the axes that actually exist on a machine tool.

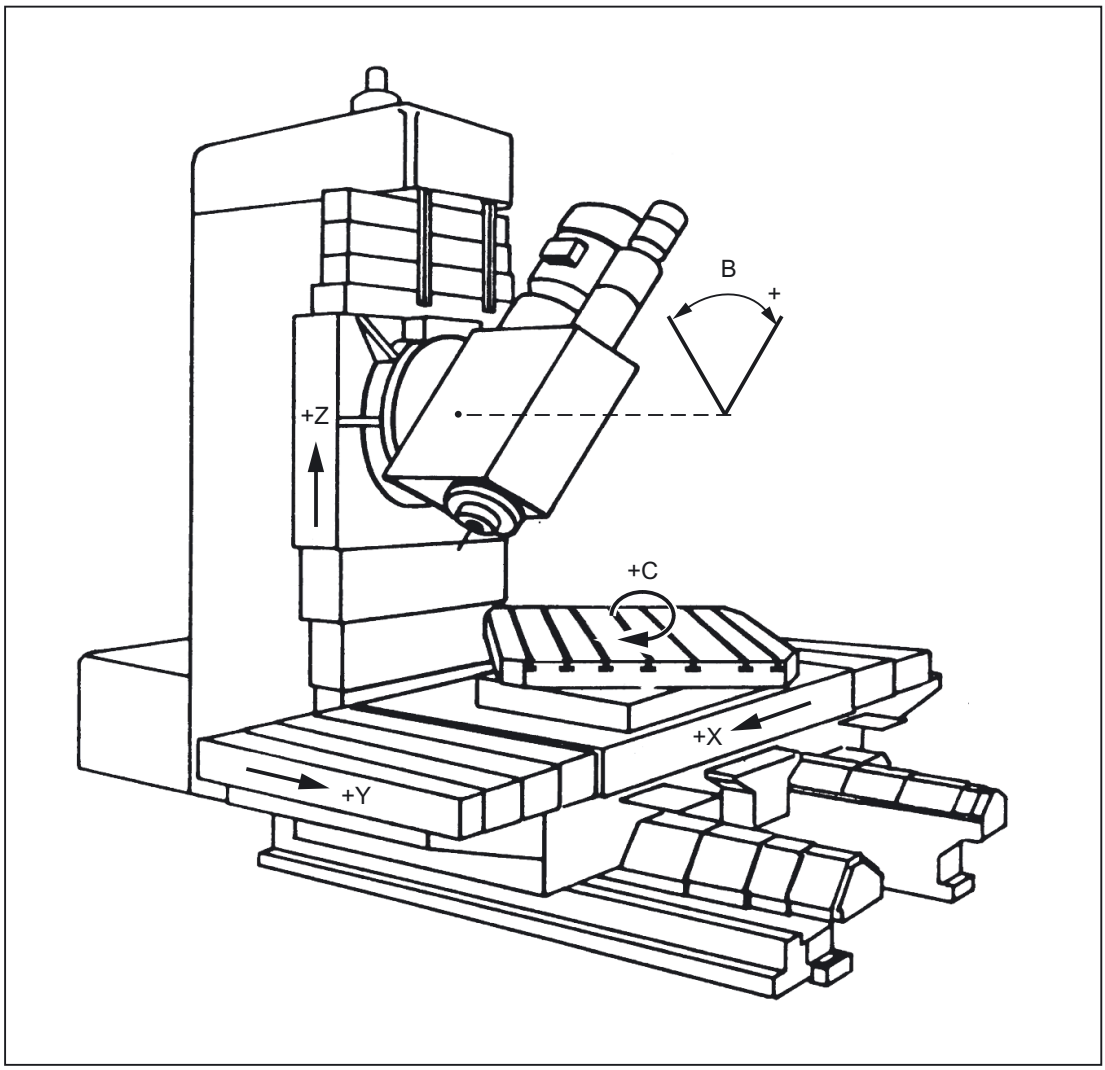


Fig. 2-3 Machine axes X, Y, Z, B, S on a Cartesian machine

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.

2.1.3 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.

2.1.4 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.

2.1.5 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

MD20110 RESET_MODE_MASK

Bit 12:	Reset response of the modified geometry axis assignment.
Bit 12 = 0	When machine data $MC_GEOAX_CHANGE_RESET is enabled, a modified geometry axis assignment is cleared on Reset or end of part program. The basic setting defined in the machine data for the geometry axis assignment is activated.
Bit 12 = 1	A modified geometry axis assignment remains active after a reset/part-program end.

MD20118 GEOAX_CHANGE_RESET

FALSE:	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.
TRUE:	The configuration of the geometry axes remains unchanged or is switched to the basic state defined in machine data AXCONF_GEOAX_ASSIGN_TAB on Reset and part-program end (according to machine data $MC_RESET_MODE_MASK) and on part-program start (according to machine data$MC_START_MODE_MASK).

Program start

Analogously to the Reset response, the response on Start is determined by:

MD20112 START_MODE_MASK

Bit 12:	Response of the modified geometry axis assignment.
Bit 12 = 0	A modified geometry axis assignment remains active on part-program start.
Bit 12 = 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

The PLC is informed when a geometry axis has been replaced using GEOAX( ) through the optional output of an M code that can be set in machine data MD22532 GEOAX_CHANGE_M_CODE.

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.
- When the transformation is deactivated, the MD-defined basic setting for the geometry axis configuration becomes valid again.

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:
/FB3/ Description of Functions, Special Functions; 3-Axis to 5-Axis Transformation (F2)
/FB2/ Description of Functions, Expansion 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 such that the geometry axes are imaged on the first three channel axes, i.e., on XX, YY and ZZ.

```
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.
```

2.1.6 Special axes

Meaning

In contrast to geometry axes, no geometrical relationship is defined between the special axes.

Note

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

2.1.7 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.

2.1.8 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 transporting workpieces
- Tool magazine/turret

Reference

References:
/FB2/ Description of Functions, Expansion Functions; Positioning Axes (P2)
/FB1/ Description of Functions, Basic Machine; Spindles (S1)
/FB3/ Description of Functions, Special Functions; Gantry Axes (G1)
/FB2/ Description of Functions, Expansion Functions; Positioning Axes (P2)
/FB3/ Description of Functions, Special Functions; Axis Couplings and ESR (M3)
/FB1/ Description of Functions, Basic Machine; PLC Basic Program (P3)
/FB2/ Description of Functions, Expansion Functions; Oscillation (P5)
/FBSY/ Description of Functions, Synchronized Actions

2.1.9 Main axes

Meaning

A main axis is an axis that is interpolated by the main run.

This interpolation can be started:

- 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 PLC basic 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.
The actions below are affected:

- The sequence for canceling the axis (equivalent to delete distance-to-go)
- Stopping or interrupting the axis
- Continuing the axis (continue sequence of motion)
- Resetting the axis to its basic status

2.1.10 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 instruction 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

```
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 = 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 AXCONF_CHANAX_NAME_TAB

Application

In the case of helical interpolation FGROUP can be programmed to determine whether:

- The programmed feedrate should be valid on the path
  (all 3 programmed axes are path axes)
- The programmed feedrate should be valid on the circuit
  (2 axes are path axes and the infeed axis is a synchronous axis)

2.1.11 Axis configuration

The figure below shows the assignment between the geometry axes, special axes, channel axes and machine axes as well as the names of the individual axis types. MD are used for assignment.

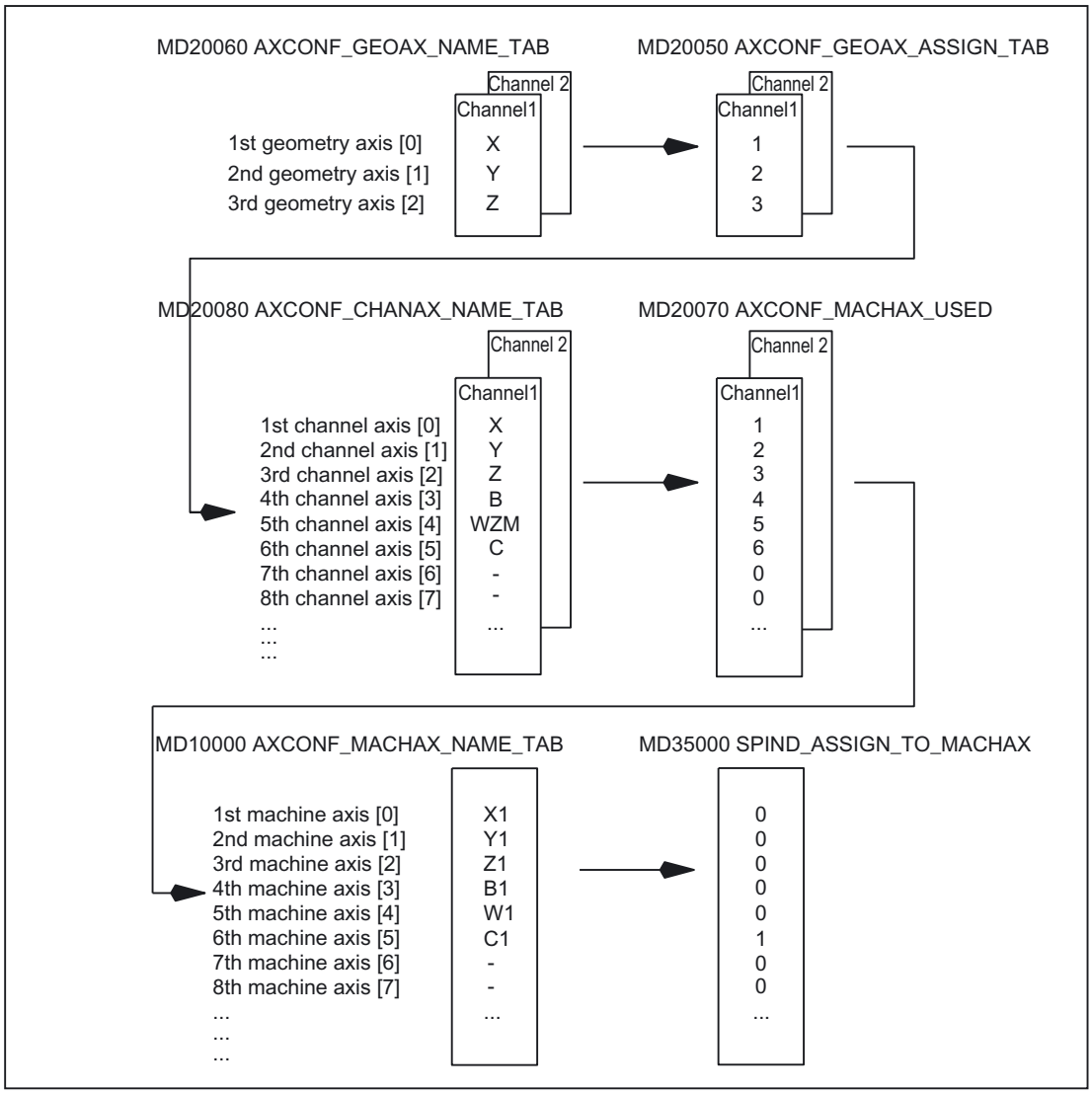


Fig. 2-4 Axis configuration

Note

Leading zeroes in user-defined axis identifiers are ignored.

Example:

MD10000 AXCONF_MACHAX_NAME_TAB[0] = X01 is equivalent to X1

The geometry axes must be assigned to the channel axes in ascending order leaving no gaps.

Special points to be noted

- Three geometry axes are assigned to the channel axes in the MD.
- All channel axes that are not assigned to the three geometry axes are special axes.
- The channel axes are assigned to machine axes.
- The spindles are also assigned to machine axes.

Channel axis gaps

A machine axis (local axis or link axis) does not have to be assigned to every channel axis according to
MD20080 AXCONF_CHANAX_NAME_TAB
via
MD20070 AXCONF_MACHAX_USED.

If the concrete machine has a machine axis, which has been specifically assigned to the channel axis, references to the logical machine axis image
MD10002 AXCONF_LOGIC_MACHAX_TAB
are entered in the machine data
MD20070 AXCONF_MACHAX_USED,
otherwise the value is 0.

Application:

Consistent, semidefined channel axis names for various machine versions of a manufacturer's machine series.

Advantages:

- Uniform basic configuration of various machines
- Simple reconfiguration on removal of a machine
- Portability of programs

Reliability of channel axis gaps

The machine data
MD11640 ENABLE_CHAN_AX_GAP
must be used to explicitly inform of channel axis gaps.

If this is not carried out, an entry of 0 in the machine data
MD20070 AXCONF_MACHAX_USED
prevents other machine axes being assigned to channel axes.

References:
/FB2/ Description of Functions, Expansion Functions; Multiple Control Panels on Multiple NCUs, Distributed Systems (B3)

Example

In the example below, a machine tool channel axis is specified without a real machine axis.

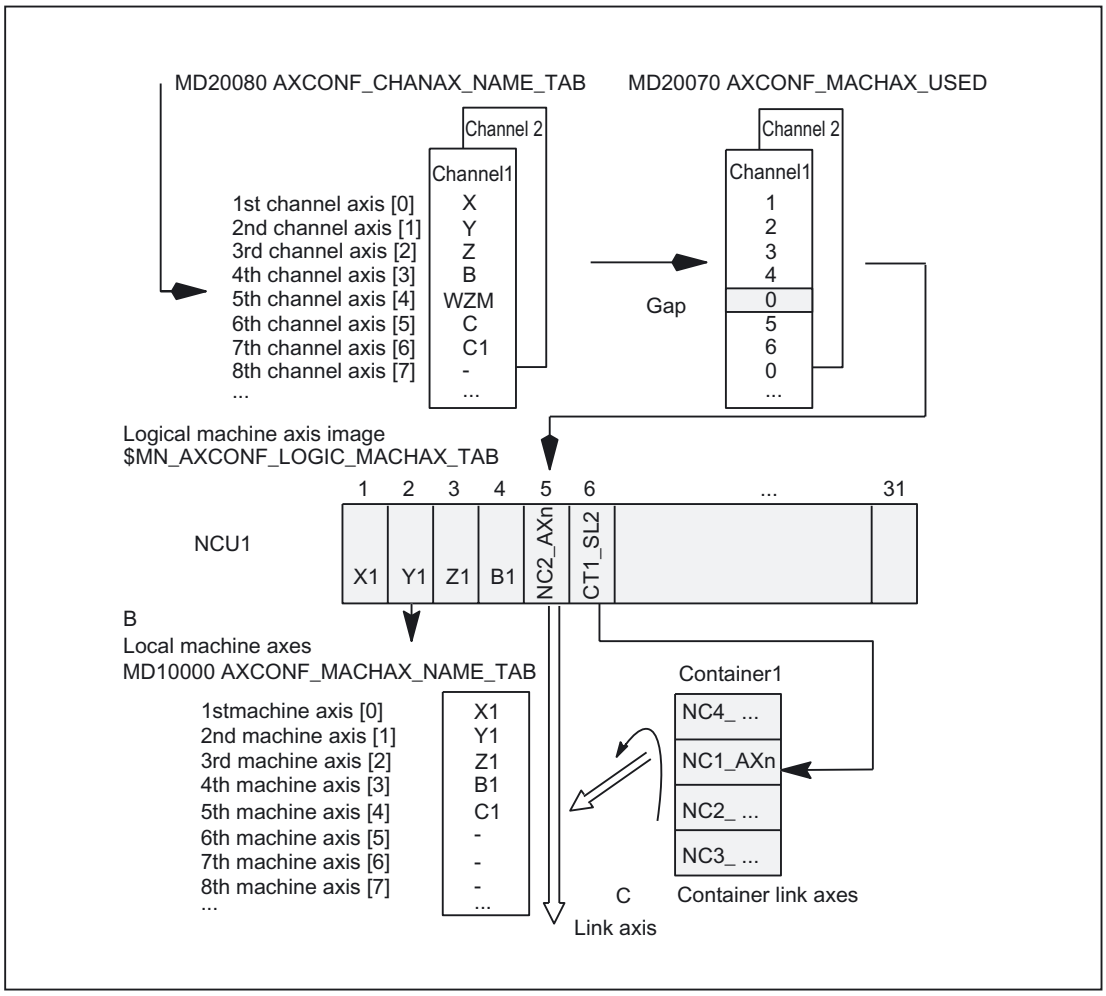


Fig. 2-5 Axis configuration with channel axis gap

Note

The gaps count as axes with reference to the number of channel axes and their indices.

If an attempt is made to define a channel axis gap on the geo axis via the machine data MD20050 AXCONF_GEOAX_ASSIGN_TAB,
the attempt is rejected without an alarm.

Using channel axes in
MD24120ff. TRAFO_GEOAX_ASSIGN_TAB1...8
and
MD24110ff. TRAFO_AXES_IN1...8,
to which no machine axes are defined using
MD20070 AXCONF_MACHAX_USED
(gap), triggers alarms 4346 or 4347.

2.1.12 Link axes

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.

Requirements:

- The participating NCUs, NCU1 and NCU2, must be connected by means of high-speed communication via the link module.

References:
/PHD/ Configuring Manual NCU 571-573.2; Link Module

- The axis must be configured appropriately by machine data.
- The link axis option must be installed.

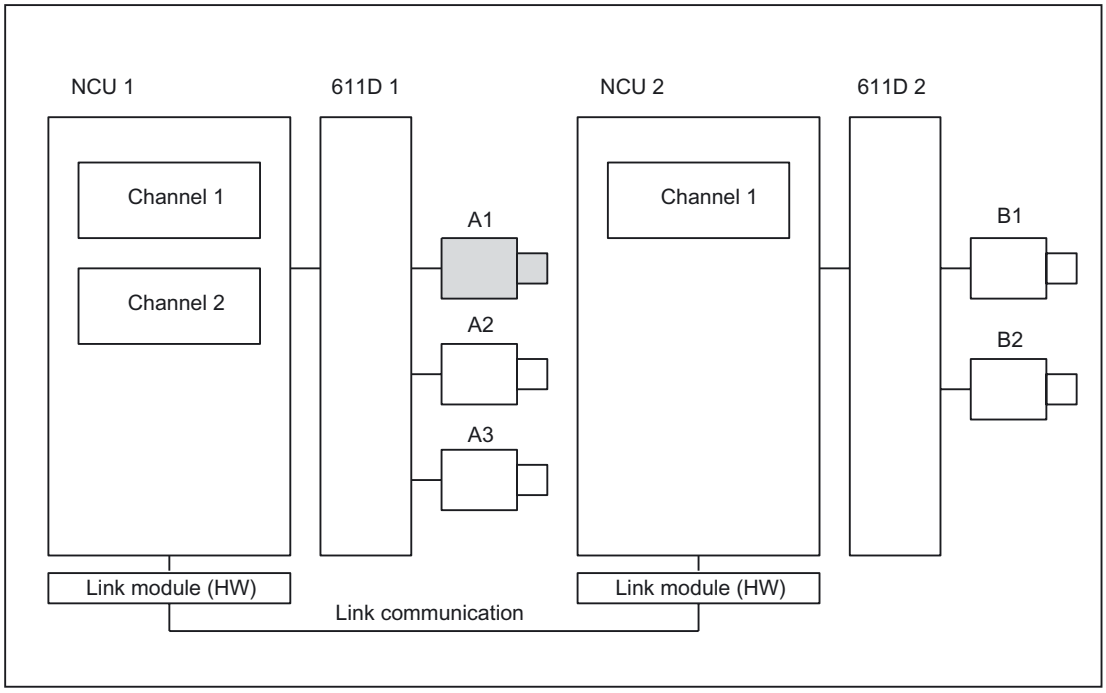


Fig. 2-6 Overview of link axes

The link axes are described in
References:
/FB2/ Description of Functions, Expansion Functions; Multiple Operator Panels on Multiple NCUs, Distributed Systems (B3)

Note

The link axis functionality is currently not available with the SINUMERIK 840Di.

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:

- Container number
- A slot (circular buffer location within the container)

The entry in a circular buffer location contains:

- A local axis

  or

- A link axis

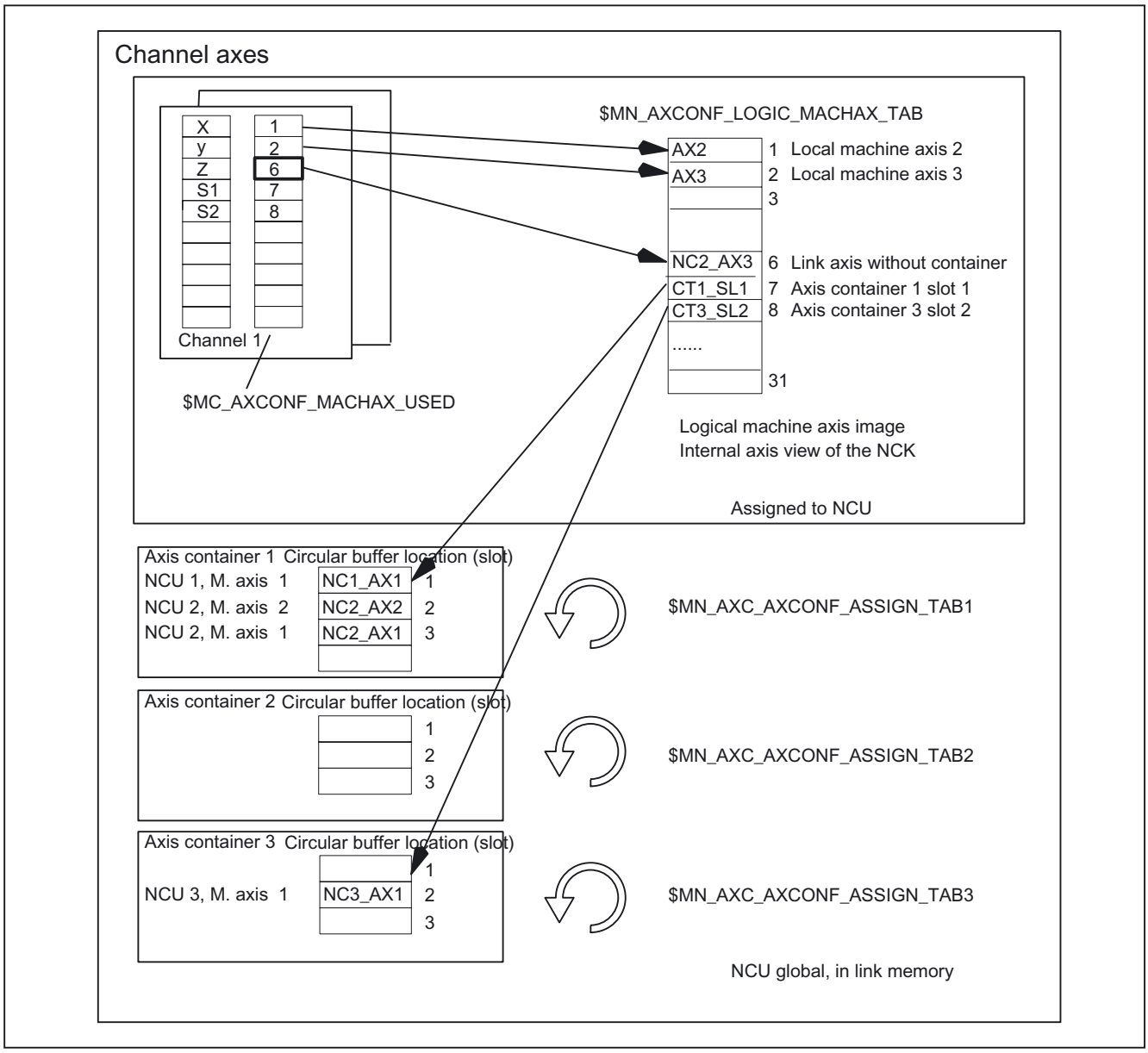


Fig. 2-7 Mapping of channel axes onto axis containers via logical machine axis image

Axis container entries contain local machine axes or link axes from the perspective of an individual NCU.

The entries in the logical machine axis image $MN_AXCONF_LOGIC_MACHAX_TAB of an individual NCU are fixed.

Note

The axis container functionality is currently not available with the SINUMERIK 840Di.

The axis container function is described in
References:
/FB2/ Description of Functions, Expansion Functions; Multiple Operator Panels on Multiple NCUs, Distributed Systems (B3)

2.2 Zeros and reference points

2.2.1 Reference points in working space

The following zeros and reference points are defined in the working space, in which tools move when machining workpieces:

Table 2-1 Zeros and reference points

Zero points		Reference points	
	**M** = Machine zero		**R** = Reference point
	**W** = Workpiece zero		**T** = Toolholder reference point

- **M**achine zero **M**

  The machine zero M defines the machine coordinate system MCS. All other reference points refer to the machine zero.

- **W**orkpiece 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.

- **R**eference 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.

- **T**oolholder 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 Point) and the toolholder reference point.

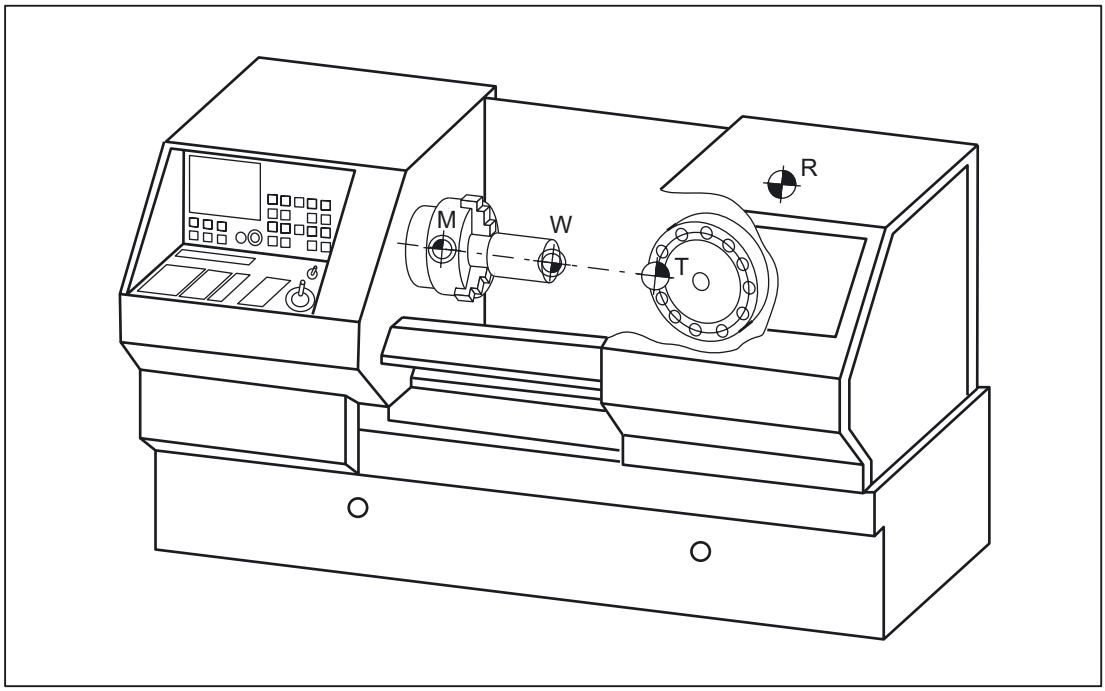


Fig. 2-8 Zeros and reference points on a turning machine

The zero of the coordinate system MCS corresponds to M and the zero of the WCS corresponds to W. In the working space of the machine the reference point is defined as R and the toolholder reference point with T.

2.2.2 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.

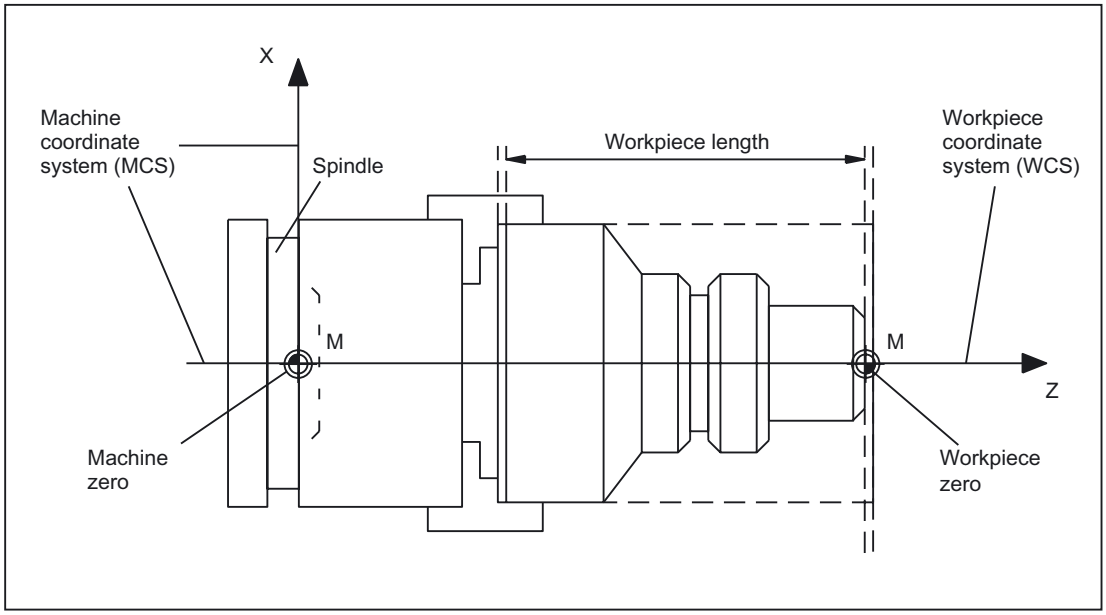


Fig. 2-9 Position of coordinate systems by machine zero M and workpiece zero W

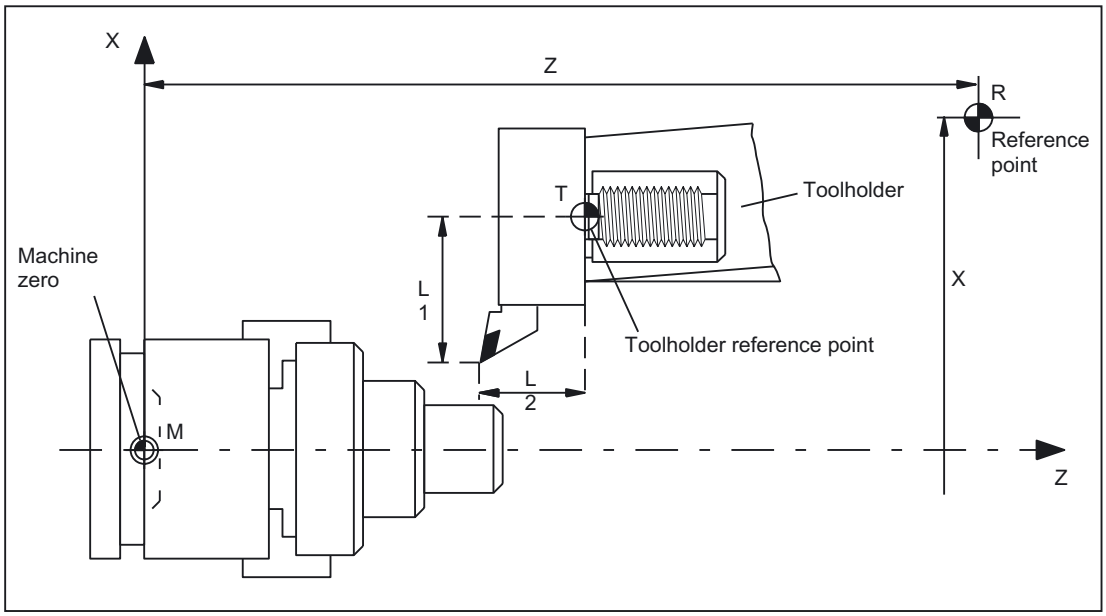


Fig. 2-10 Position of reference point in relation to machine zero

2.3 Coordinate systems

2.3.1 Overview

Meaning

DIN 66217 stipulates that machine tools must use right-handed, rectangular (Cartesian) coordinate systems.

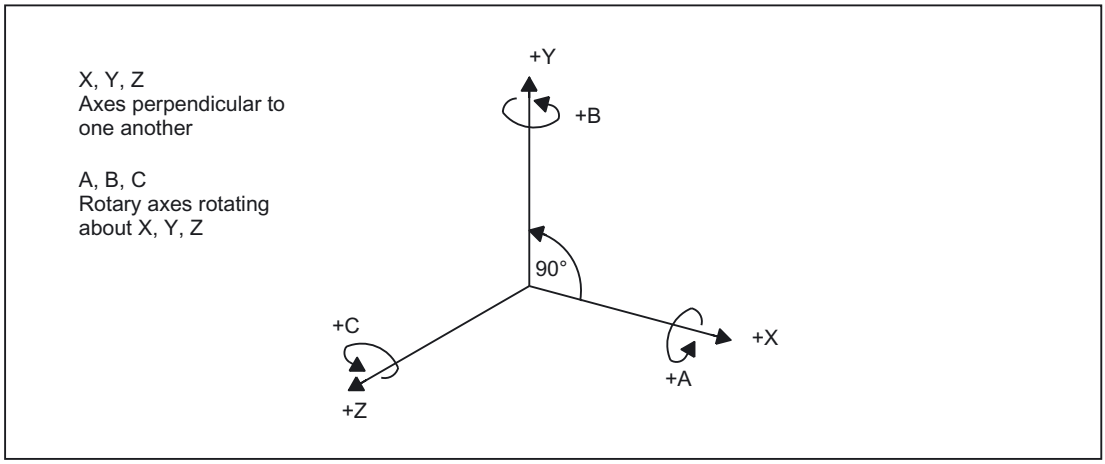


Fig. 2-11 Clockwise, rectangular Cartesian coordinate system

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

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 BZS.

An activated settable `FRAME G54` to `G599` converts the BZS to the SZS.

The WCS, which is the basis for programming, is defined by the programmable `FRAME`.

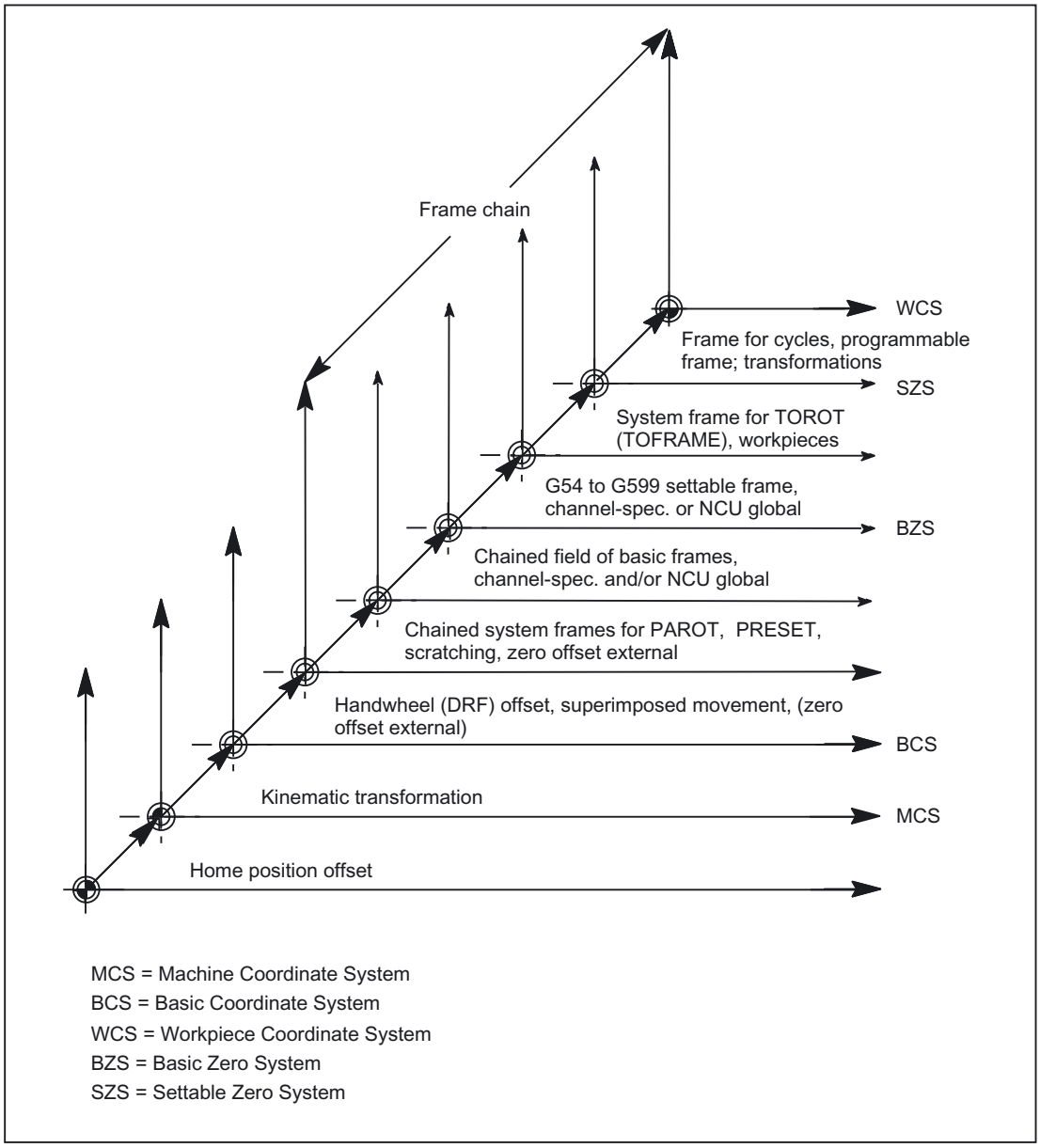


Fig. 2-12 Interrelationship between coordinate systems

2.3.2 Machine coordinate system (MCS)

The machine coordinate system (MCS) is made up of all physically available machine axes.

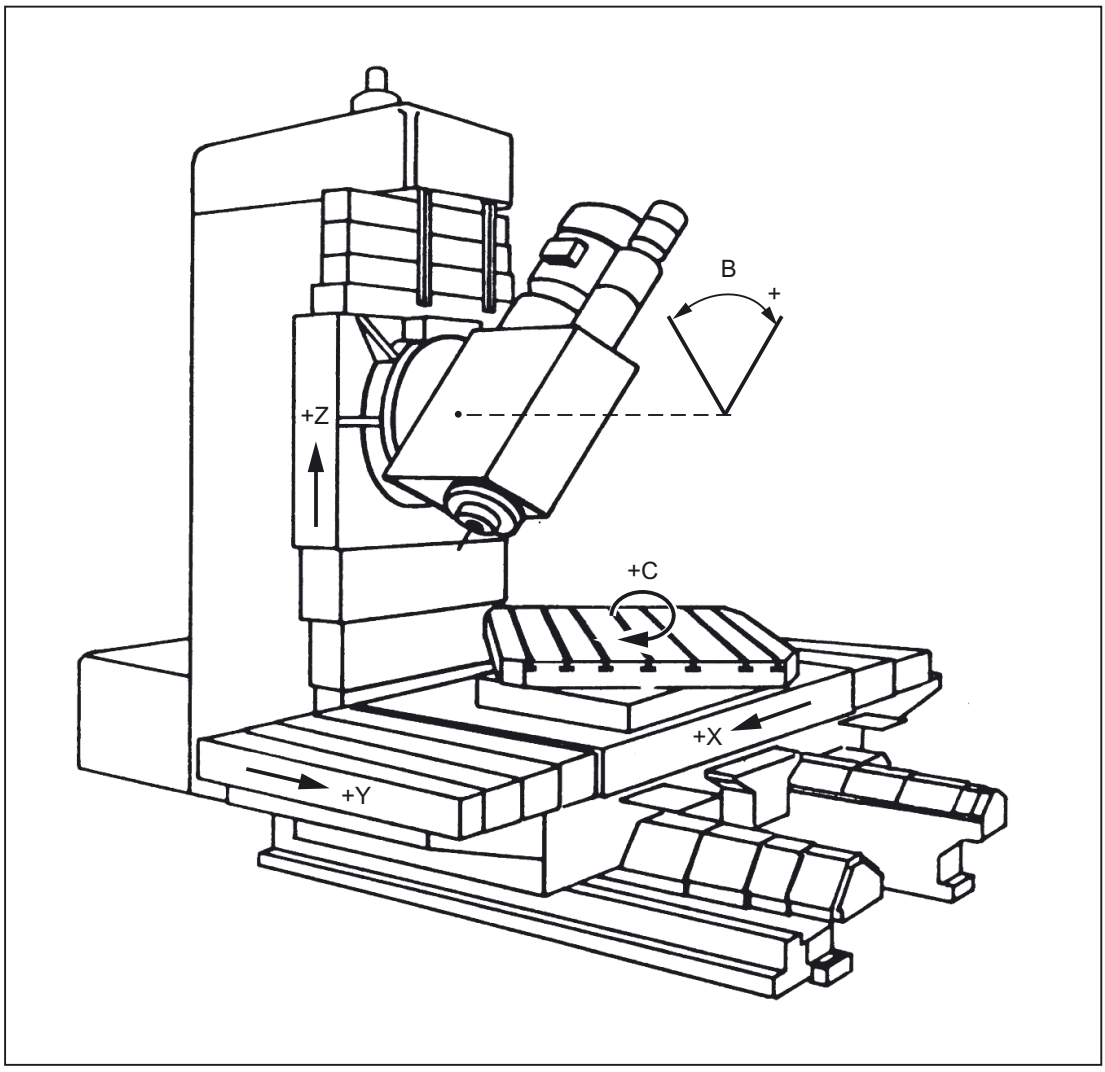


Fig. 2-13 MCS with machine axes X, Y, Z, B, C (5-axis milling machine)

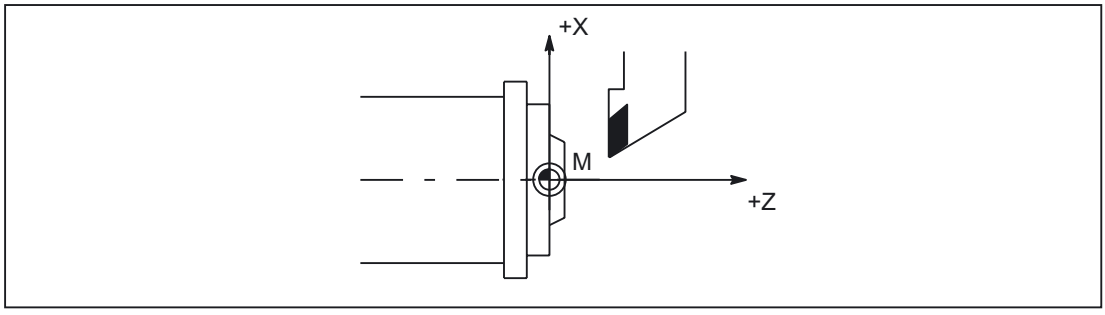


Fig. 2-14 MCS with machine axes X, Z (turning machine)

Axial preset offset

The "Preset" function can be used to redefine the control zero in the machine coordinate system. The preset values act on machine axes. Axes do not move when "Preset" is active.

Note

After Preset, the reference points are invalid!

If possible do not use this function.

2.3.3 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

The BCS and the MCS always coincide when the BCS can be mapped onto the MCS without kinematic transformation (e.g., TRANSMIT/face transformation, 5axis transformation and up to three machine axes).

On such machines, machine axes and geometry axes can have the same names.

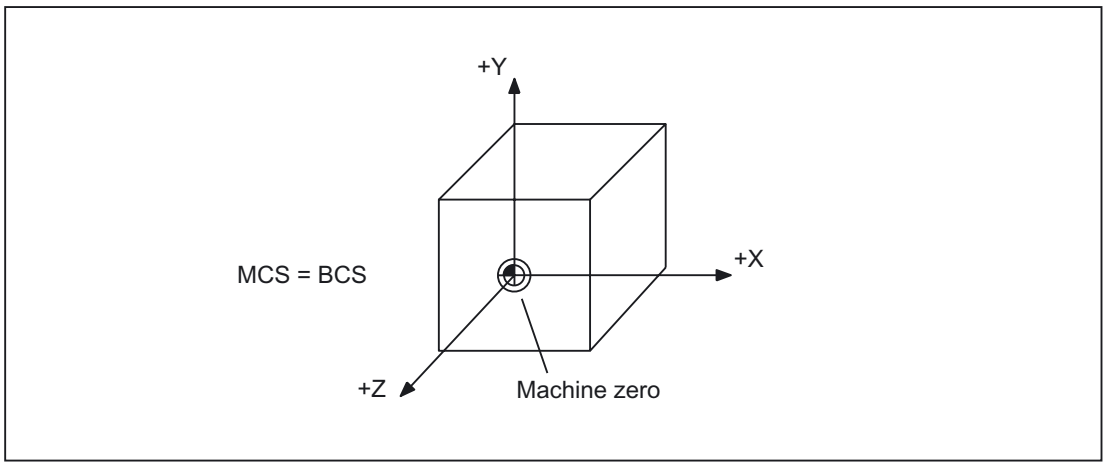


Fig. 2-15 MCS=BCS without kinematic transformation

Machine tools with kinematic transformation

The BCS and the 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.

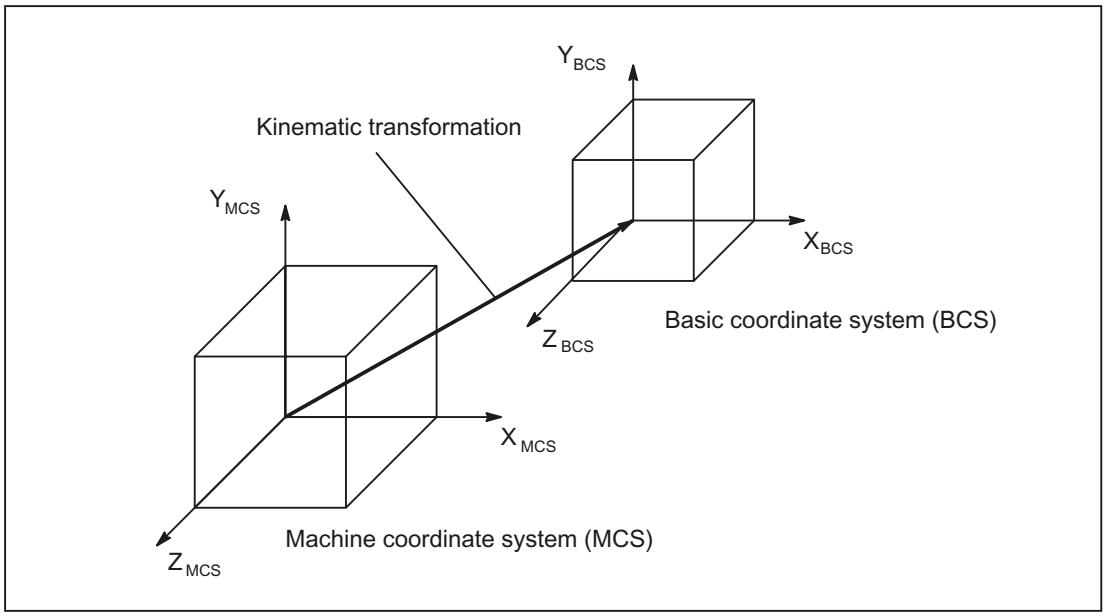


Fig. 2-16 Kinematic transformation between the MCS and BCS

Machine kinematics

The workpiece is always programmed in a two or threedimensional, rightangled 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:
/FB3/ Description of Functions, Special Functions; 3-Axis to 5-Axis Transformation (F2)
/FB2/ Description of Functions, Expansion Functions; Kinematic Transformation (M1)

2.3.4 Additive offsets

Zero offsets external

The "Zero offset external" is an axial offset. Unlike with frames, no components for rotation, scaling and mirroring are possible.

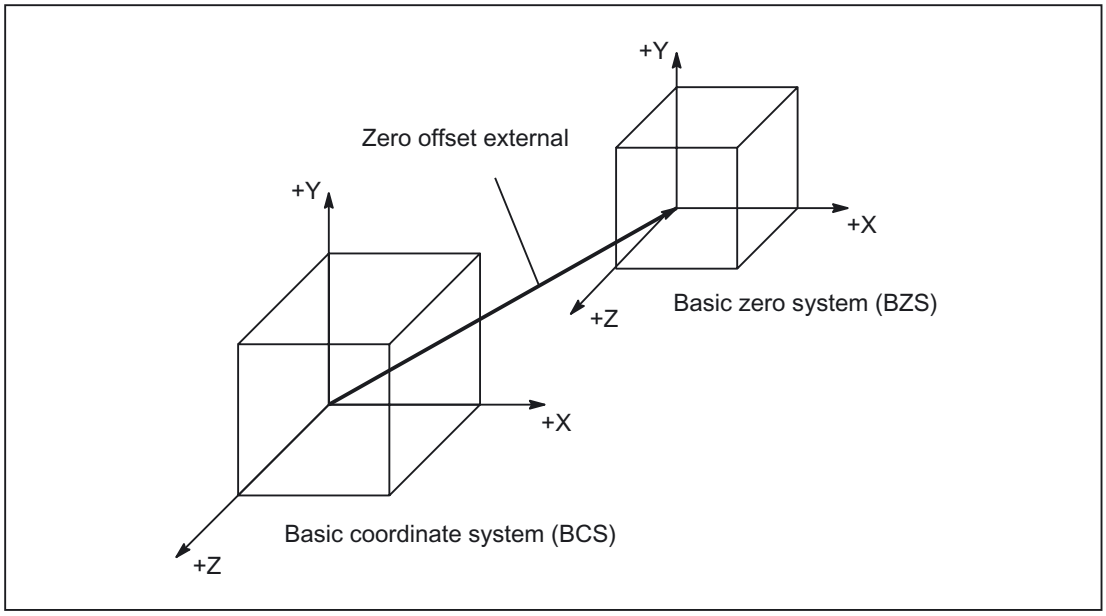


Fig. 2-17 Zero offset external between BCS and BZS

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 PLC signal "Accept external zero offset" (DB31 to DBX3.0) activates the previously defined offset values. 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:

```
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 using the machine data MD28082 MM_SYSTEM_FRAME_MASK.

Programming

Setting a new offset via the axis-specific system variables:

$AA_ETRANS[axis]=$R_i$

The instruction below reads the axisspecific 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 you to set an additional incremental work offset for geometry axes and special axes in the basic coordinate system using a manual pulse encoder.

The DRF offset can be read via axis-specific system variable $AC_DRF[<axis>].

Reference
/FB2/ Description of Functions, Extended Functions; Jog With/Without Handwheel (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].

Run-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.

Depending on the machine data
MD24008 CHSFRAME_POWERON_MASK,
system frames are retained on POWER ON.

RESET/end of program

The activated values remain active after RESET and program end.

Reset response of the channel-specific system frames is as follows:

With machine data setting
MD24006 CHSFRAME_RESET_MASK, Bit 1 = 1,
the system frame for the "Zero offset external" becomes active following RESET.

With machine data setting
MD24006 CHSFRAME_RESET_MASK, Bit 1 = 0,
the "Zero offset external" in the active system frame is deleted in the data management.

Following RESET, these frames are active:

System frame for
MD24006 CHSFRAME_RESET_MASK, Bit 4 = 1 (workpiece reference points)
MD24006 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 "Zero offset external" automatically becomes active again with the next traverse motion in the block.

After a mode change from "Reference point approach" mode, the VDI signal for the referenced axes must be set for reactivation.

2.3.5 Basic zero system (BZS)

BZS

The basic zero system (BZS) is the basic coordinate system with a basic offset.

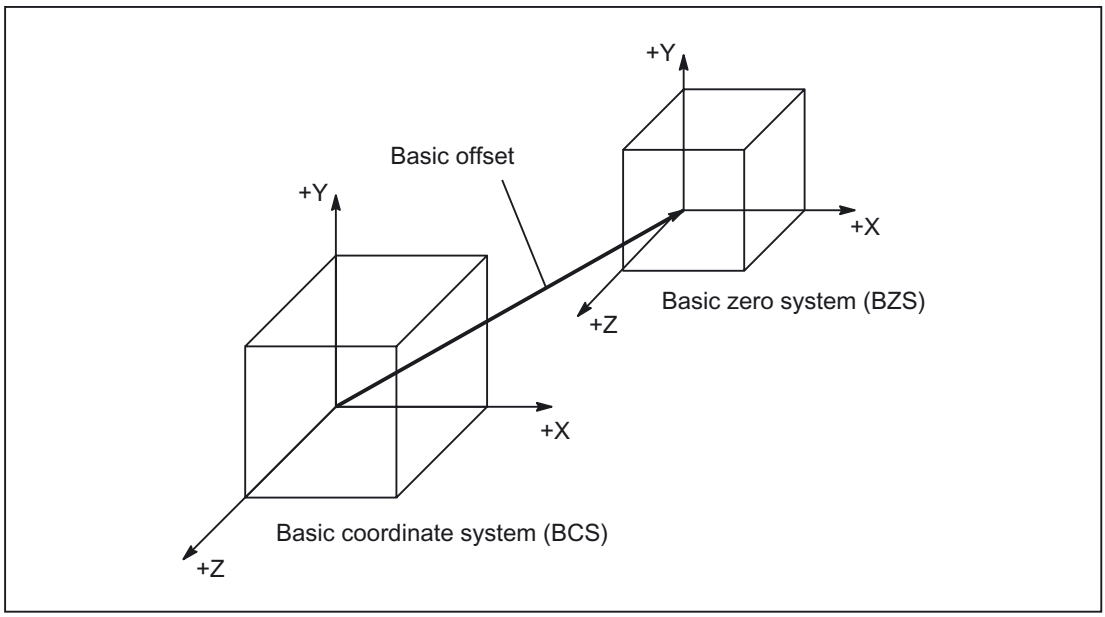


Fig. 2-18 Basic offset between BCS and BZS

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

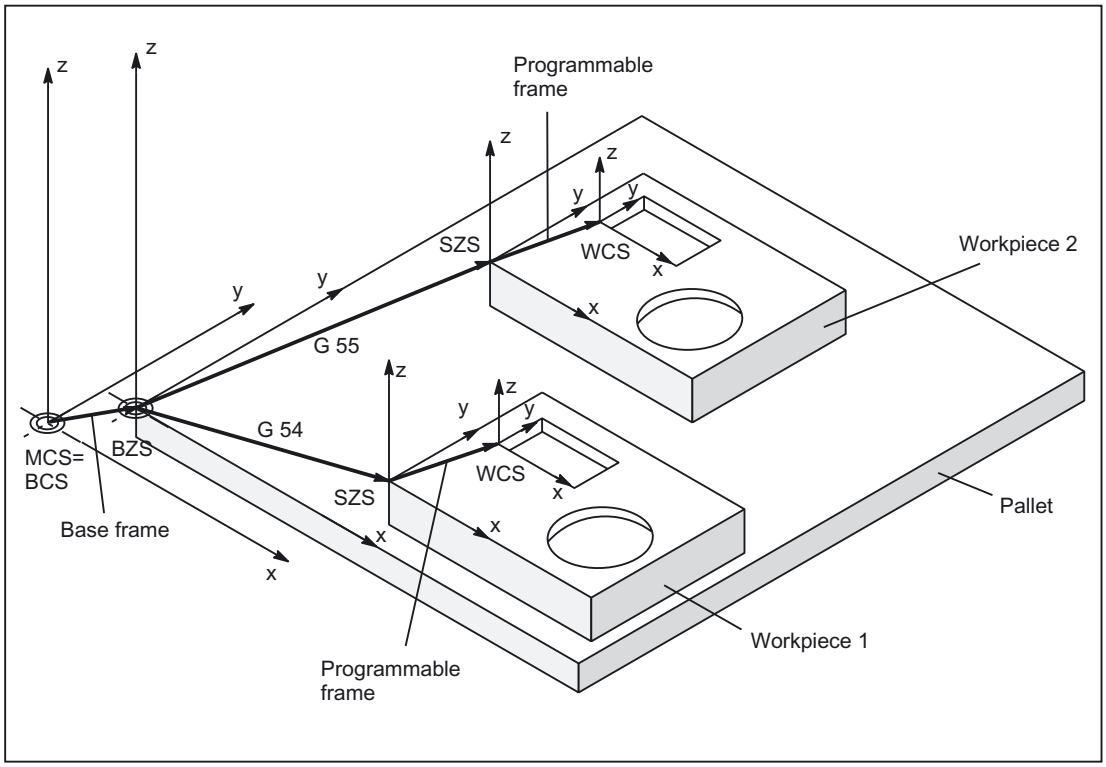


Fig. 2-19 Example of the use of the basic offset

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

Machine manufacturer

Recommendation:

Use the 3rd basic offset onwards for your own applications.
The 1st and 2nd basic offsets are reserved for PRESET and the "Zero offset external".

2.3.6 Settable zero system (SZS)

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.

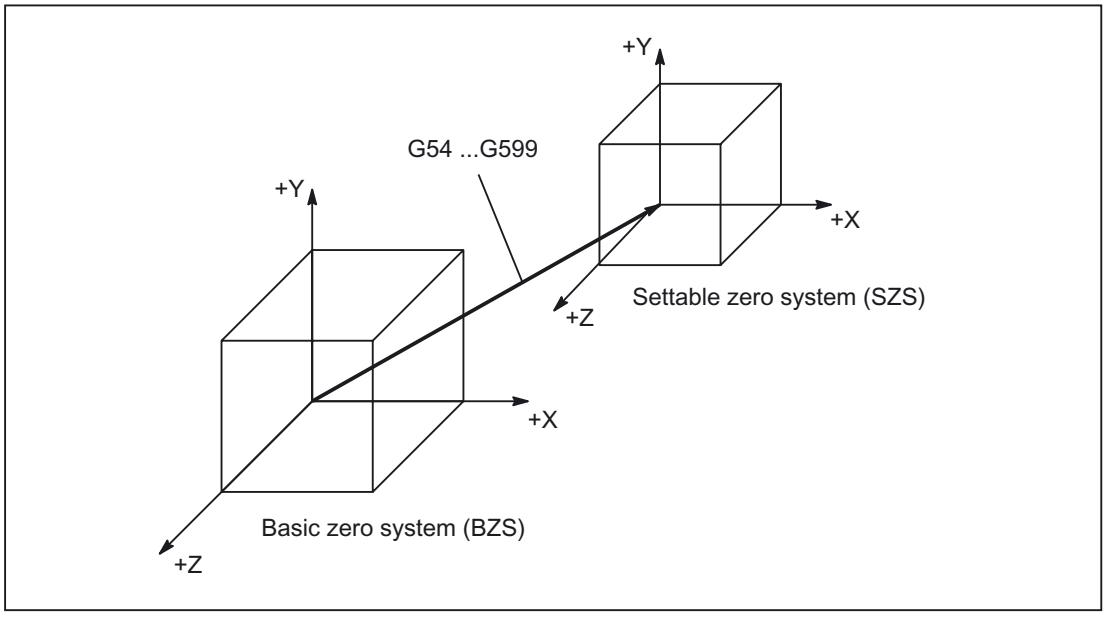


Fig. 2-20 Settable FRAME G54 to G599 between BZS and SZS

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 configuration is made using:
MD9424 $MM_MA_COORDINATE_SYSTEM = <Coordinate system>

Coordinate system:	
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

2.3.7 Configurable settable zero system (SZS)

Application

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 insert translations or rotations either in the:

- Programmed frame $P_PFRAME

  or

- Cycle system frame $P_CYCFRAME

The WCS is, therefore, modified by cycles. A cycle interrupted by the user should thus also be executed in the programmed WCS. The SZS can be used to display the last valid coordinates.

The machine data
MD24030 FRAME_ACS_SET
can then be set to define whether the SZS must be interpreted with or without the currently programmed frame $P_PFRAME and the transformation frame $P_TRAFRAME. The default value of 1 should be retained.

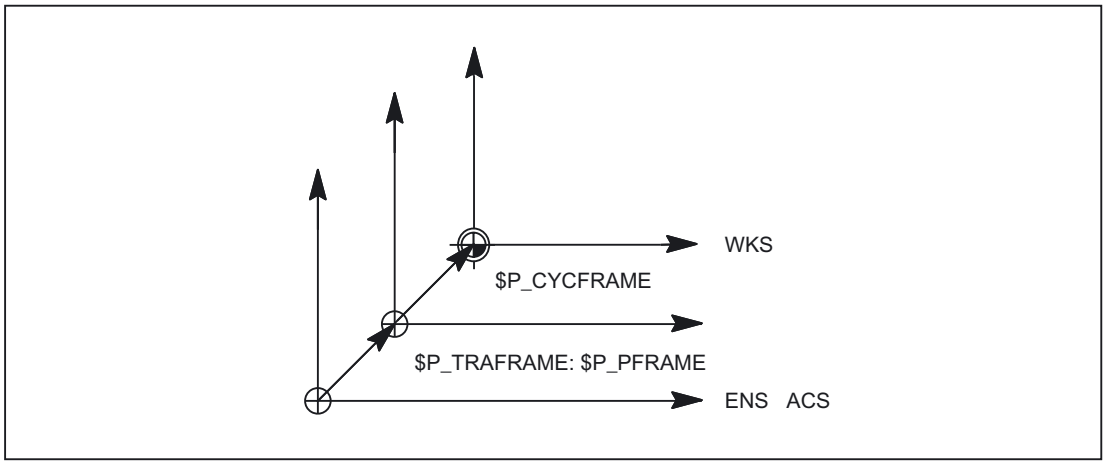


Fig. 2-21 SZS with MD24030 FRAME_ACS_SET = 0

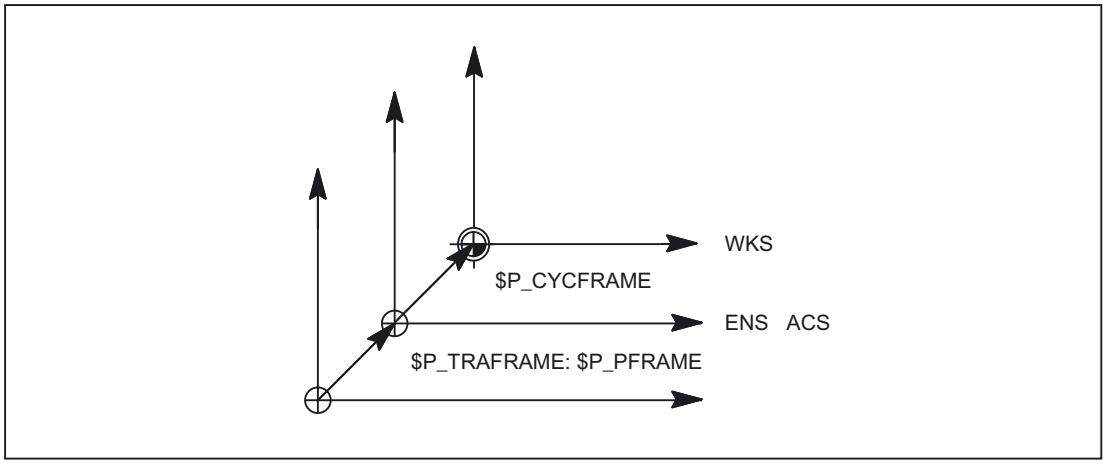


Fig. 2-22 SZS with MD24030 FRAME_ACS_SET = 1

Manual traverse in SZS

The geometry axes are traversed in the WCS in JOG mode. It is also possible to carry out manual traversing in the SZS coordinate system. Using variable $AC_JOG_COORD it is possible to switch between manual traverse in the WCS or SZS.

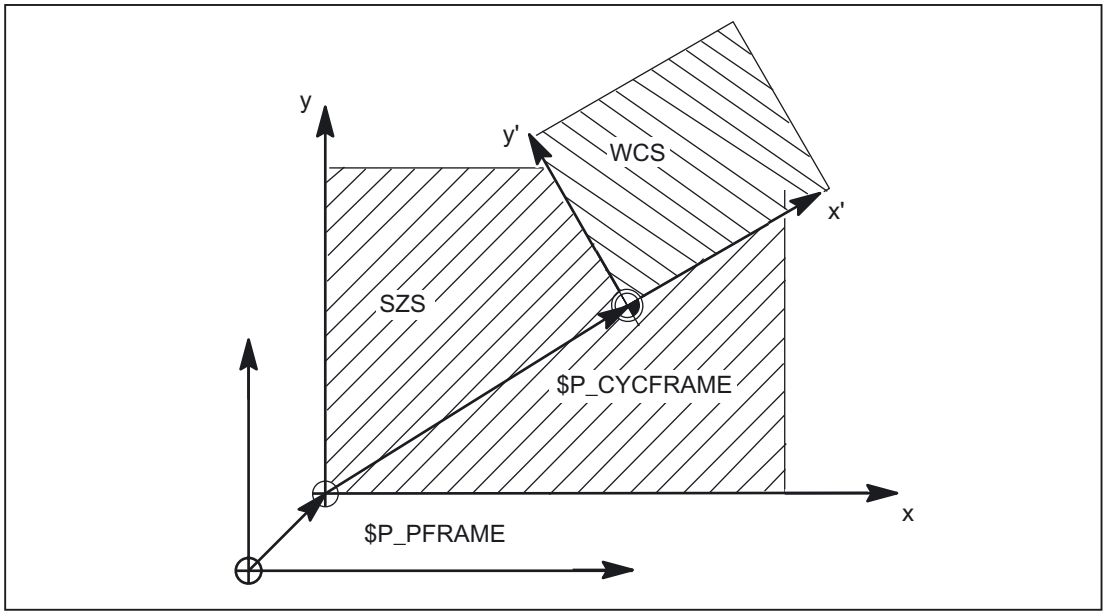


Fig. 2-23 Manual traverse in WCS or SZS

2.3.8 Workpiece coordinate system (WCS)

WCS

The workpiece coordinate system (WCS) is the programming basis.

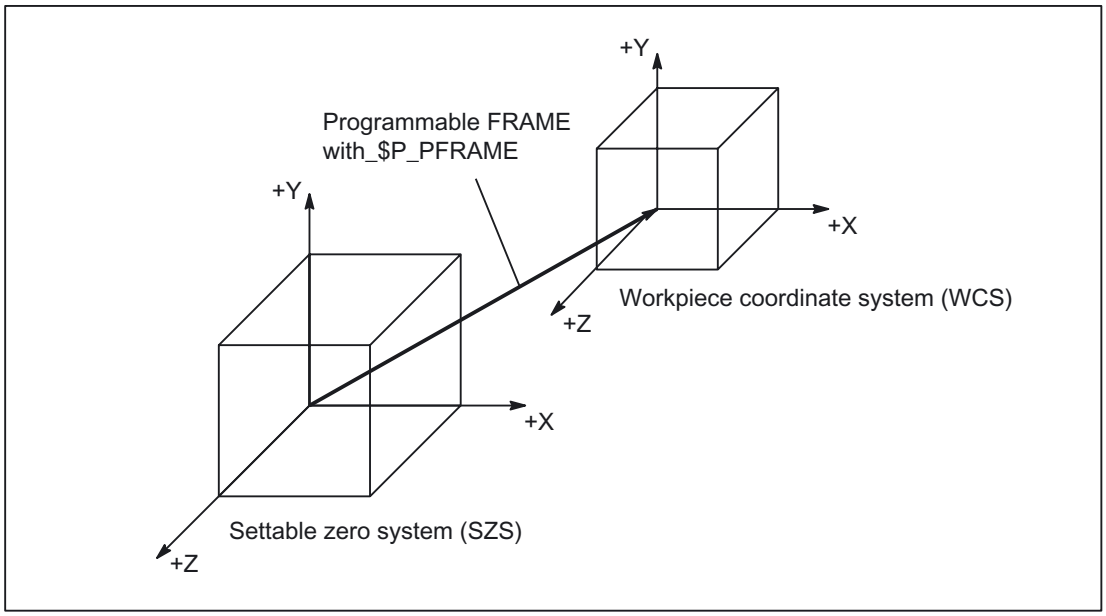
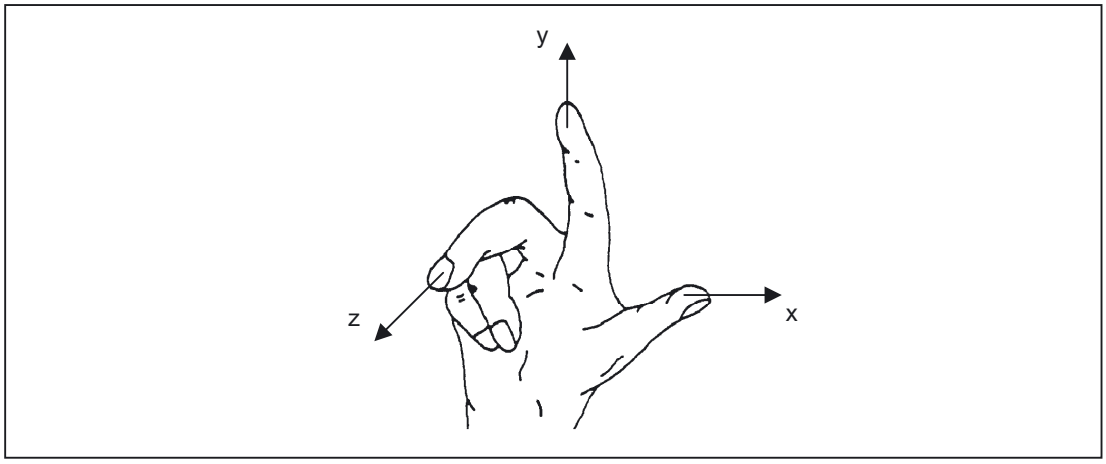


Fig. 2-24 Programmable FRAME between SZS and WCS

2.4 Frames

2.4.1 Coordinate axes, zeros and reference points

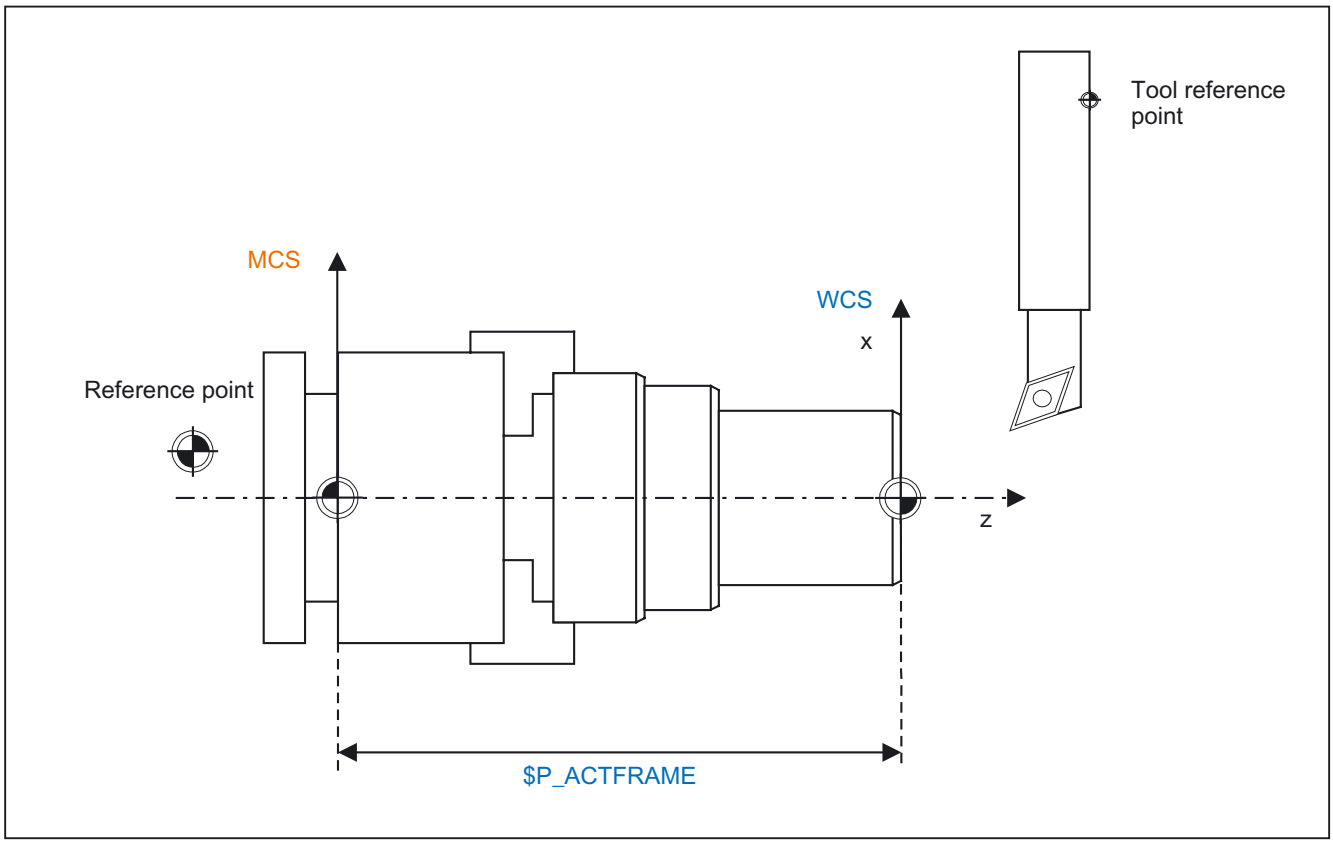
As a rule, a coordinate system is formed of three mutually perpendicular coordinate axes. The positive directions of the coordinate axes are determined using the right hand rule. The coordinate system is related to the workpiece and programming takes place independently of whether the tool or the workpiece 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.



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.

	M = Machine zero		R = Reference point
	W = Workpiece zero		T = Toolholder reference point



The position of the reference point R is defined by cam switches. The reference point must be approached each time the control is activated. The control can only then work with the measuring system and transfer all position values to the coordinate systems.

The machine zero M defines the machine coordinate system MCS. All other reference points refer to the machine zero.

The workpiece zero W defines the workpiece coordinate system in relation to the machine zero. The programmed part-program blocks are executed in the workpiece coordinate system.

The toolholder reference point is located on the toolholder locator. By entering the tool lengths, the control calculates the distance between the tool tip (TCP Tool Center Pos) and the toolholder reference point.

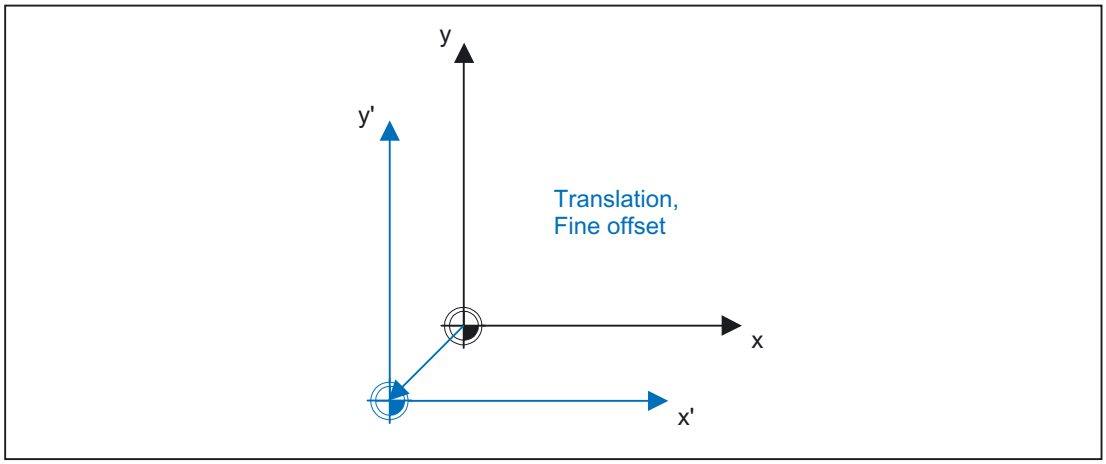
2.4.2 Frames

A frame is a structure, which contains a value for the translation, fine offset, rotation (only for geometry axes), scaling and mirroring for each axis. Activating a frame causes a **static coordinate transformation** to be carried out via a defined calculation rule.

2.4.3 Frame components

2.4.3.1 Translation

The program commands below are used to program the translation:




```
$P_UIFR[1] = CTRANS(x,10,y,10)
$P_UIFR[1,x,tr] = 10                ; Frame components
TRANS x = 10 y = 10                 ; Prog. frame only
```

2.4.3.2 Fine offset

The machine data
MD18600 $MN_MM_FRAME_FINE_TRANS
can be used to configure the fine offset as follows:

0:	The fine offset cannot be entered or programmed.
1:	Fine offset possible for settable frames, basic frames and the prog. frame via command or program.

The fine offset can be programmed in the program using command `CFINE` (x, .., y, ...). The coarse offset is defined with `CTRANS`(...). Coarse and fine offset add up to the total offset.

$P_UBFR = CTRANS(x, 10) : **CFINE(x, 0.1)** : CROT(x, 45)

$P_UIFR[1]=CFINE(x, 0.5, y, 1.0, z, 0.1)

Access to the individual components of the fine offset is achieved through component specification **FI**.

finex = $P_UIFR[$P_UIFRNUM, x, **FI**]

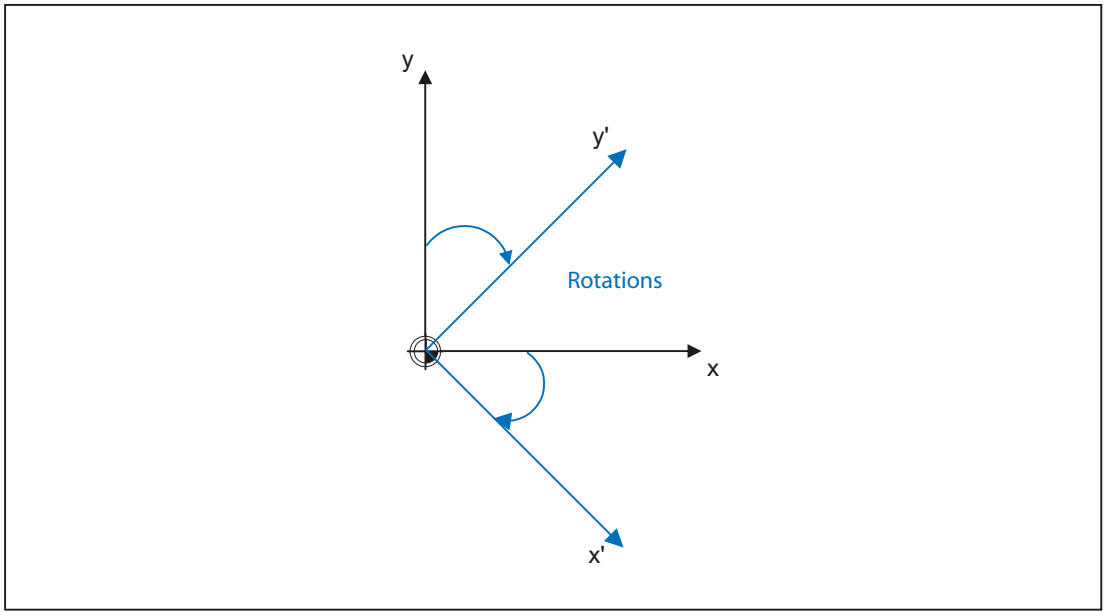
A fine offset can only be programmed if machine data
MD18600 $MN_FRAME_FINE_TRANS
is predefined with a value of 1.
If this is not the case, every assignment of a fine offset to settable frames and the basic frame is rejected with the alarm "FRAME: fine offset not possible".

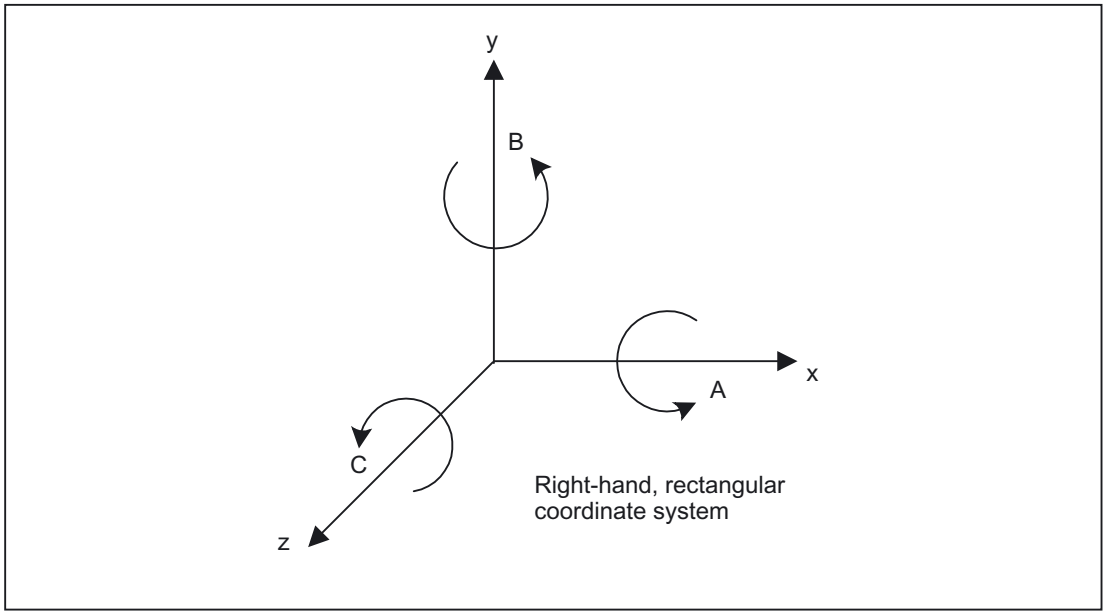
A fine offset changed by the operator does not apply until after activation of the corresponding frame, i.e., activation via `G500`, `G54` to `G599`.
Once activated, a fine offset of a frame remains active the whole time the frame is active.

When the offset of the current frame is displayed, the total offset (coarse offset and fine offset) is output.

2.4.3.3 Rotations for geometry axes

The direction of rotation about 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 identify rotations whose axes are parallel to X, Y and Z.

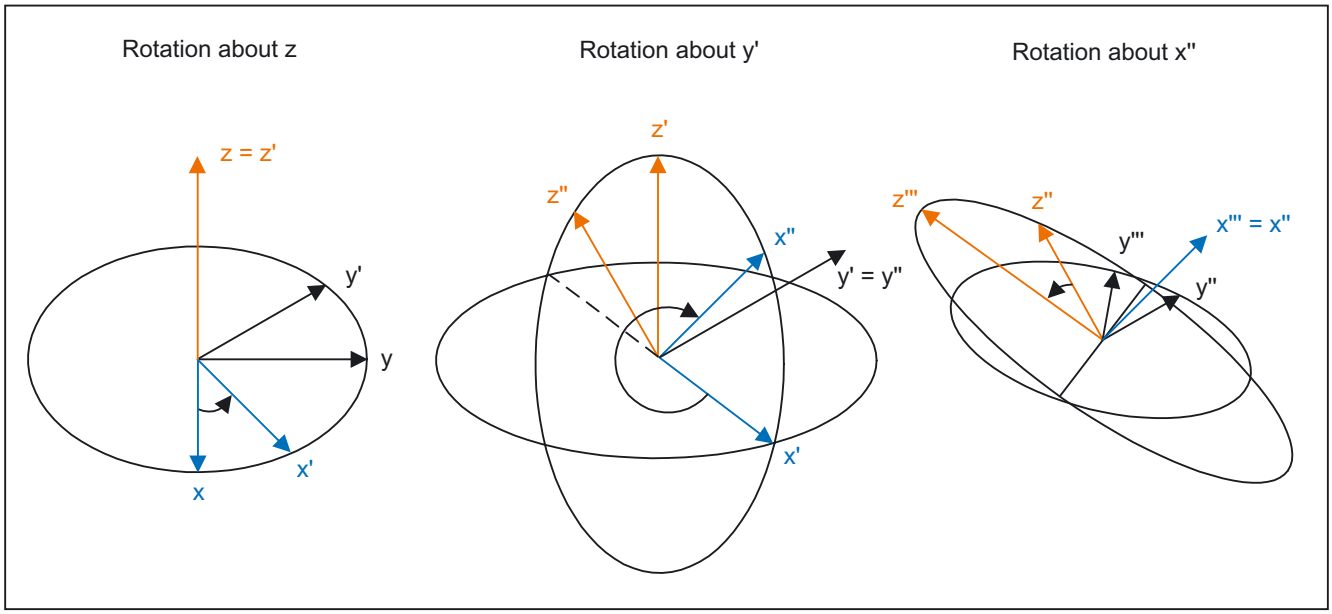
The machine data below is used to configure the rotation in the frame:

MD10600 $MN_FRAME_ANGLE_INPUT_MODE

1: RPY notation
2: Euler angle

RPY

Rotations with a **RPY angle** are carried out in the order Z, Y', X".

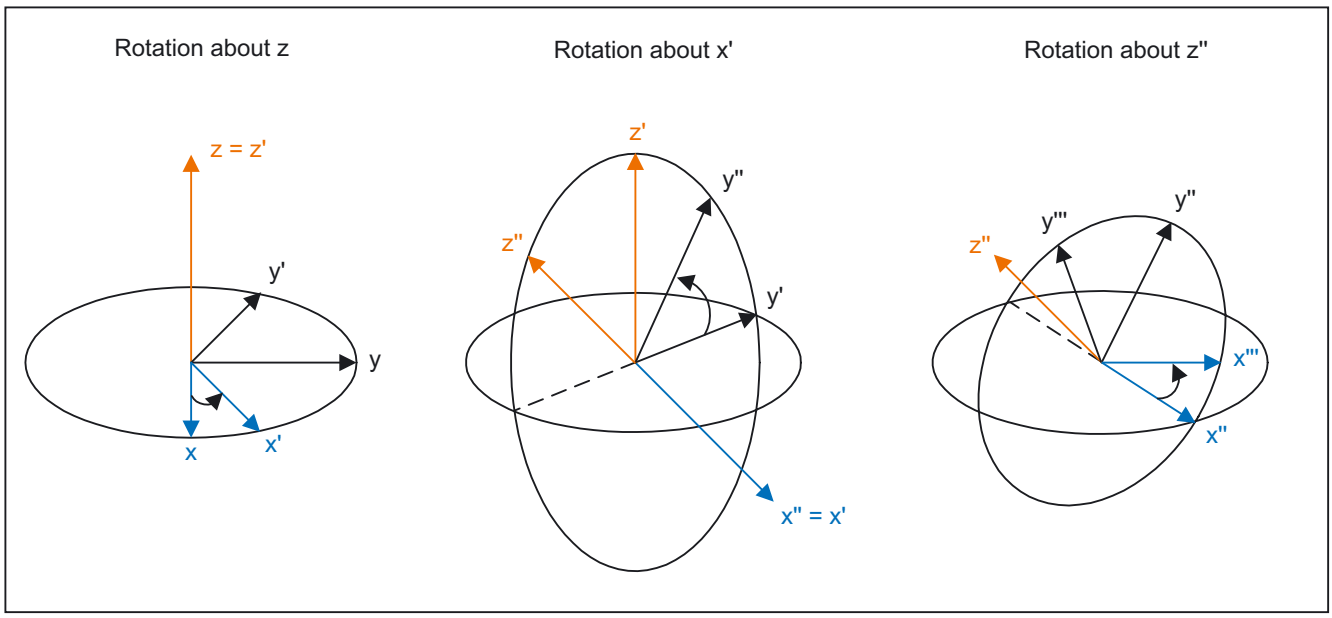


The angles are only defined ambiguously in the following ranges:

-180 <= x <= 180
 -90 < y < 90
-180 <= z <= 180

Euler angle

Rotations with a **Euler angle** are carried out in the order Z, X', Z".



The angles are only defined ambiguously in the following ranges:

0	<=	x	<	180
-180	<=	y	<=	180
-180	<=	z	<=	180

The written angles can be uniquely read back again in these areas. When rotations that are larger than the specified angles are entered, these are converted to a mode of representation that does not exceed the specified range limits.

RPY example:

$P_UIFR[1] = crot(x, 10, y, 90, z, 40), when read back, produces
$P_UIFR[1] = crot(x, 0, y, 90, z, 30).

$P_UIFR[1] = crot(x, 190, y, 0, z, -200), when read, produces
$P_UIFR[1] = crot(x, -170, y, 0, z, 160).

On writing and reading frame rotation components, these limits should be observed so the same results are achieved on writing and reading, or on repeated writing.

The program commands below are used to program the rotation:

```
$P_UIFR[1] = CROT(x, 10, y, 10)
ROT x = 10 y = 10
$P_UIFR[1,x, rt] = 10
```

CRPL - Constant Rotation Plane

The predefined function **C**onstant **R**otation **Pl**ane:
FRAME CRPL(INT, REAL)
allows a rotation to be programmed in any plane for each frame.

This method offers the advantage that no axis identifier, around which a rotation should be executed, has to be specified for a geometry coordinate axis.

As a rule, turning machines have only two geo axes, meaning that, up to now, a rotation in the plane could not be programmed.

Parameter:

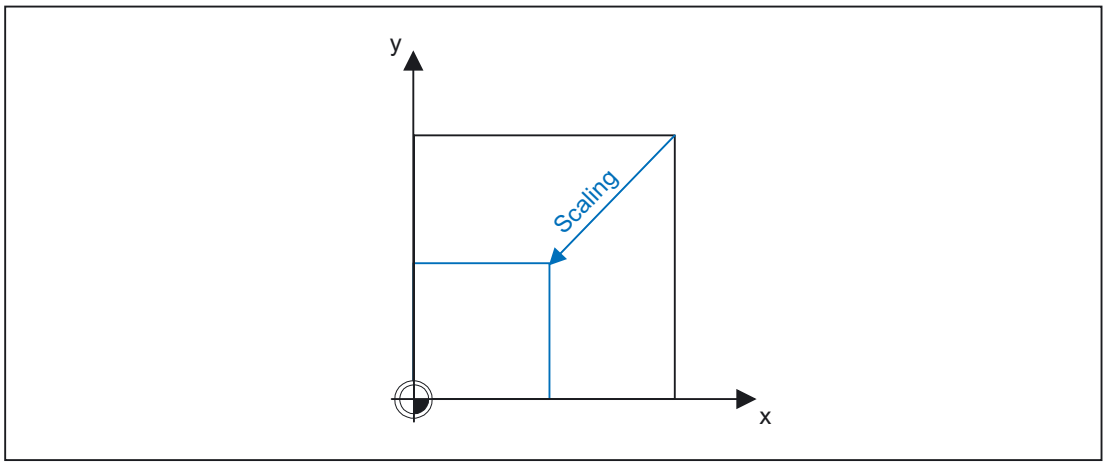
INT	0:	Rotation in the active plane
	1:	Rotation about z
	2:	Rotation about y
	3:	Rotation about x
REAL	Angle of rotation in degrees	
	RPY:	-180 <= x <= 180
		-90 <= y <= 90
		-180 <= z <= 180
	Euler:	-180 <= x <= 180
		0 <= y <= 180
		-180 <= z <= 180
	The user must keep to the named angles, in the interests of a unique backward calculation. If the limits are violated, a unique backward calculation is impossible. Entry is not aborted with an alarm.	

CRPL( ) can be chained with frames and known frame functions, such as CTRANS( ), CROT( ), CMIRROR( ), CSCALE( ), CFINE( ).

Example:

```
$P_CYCFRAME = $P_CYCFRAME : CRPL(0, 30)
$P_CYCFRAME = CTRANS(x,10) : CRPL(1, 30)
$P_CYCFRAME = CROT(x,10) : CRPL(2, 30)
$P_CYCFRAME = CRPL(3, 30) : CMIRROR(y)
```

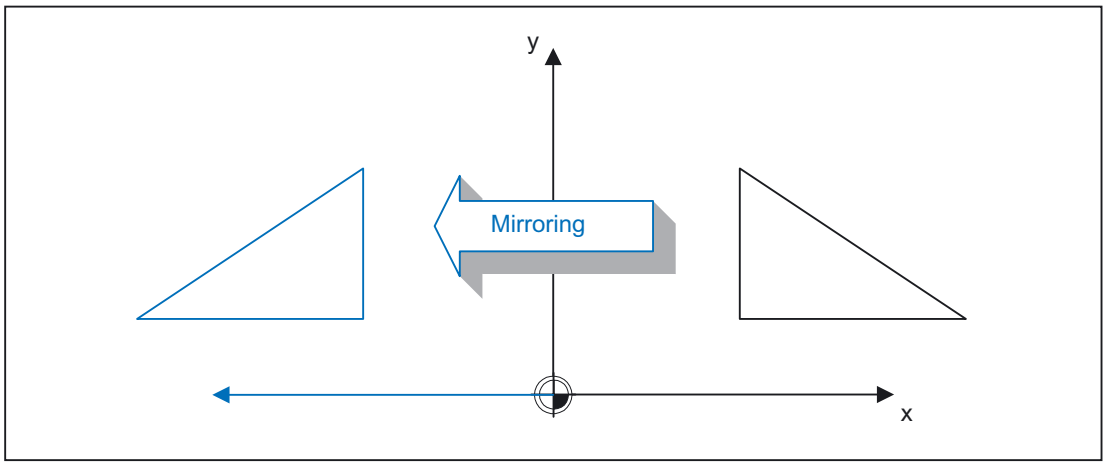
2.4.3.4 Scaling



The program commands below are used to program the scaling:

```
$P_UIFR[1] = CSCALE(x, 1, y, 1)
SCALE x = 1 y = 1
$P_UIFR[1,x, sc] = 1
```

2.4.3.5 Mirroring



The program commands below are used to program a mirroring:

```
$P_UIFR[1] = CMIRROR(x, 1, y, 1)
MIRROR x = 1 y = 1
$P_UIFR[1,x, mi] = 1
```

2.4.3.6 Chain operator

Frame components or complete frames can be combined into a complete frame using the chain operator (:).

2.4.3.7 Programmable axis identifiers

Geo axis, channel axis 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.

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 (see MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[ ]).

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 with `SPI()`, as only geometry axes are permitted for `CROT()`.

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(..)`.

For example, if the spindle is assigned to channel axis "A", programming of
N10 $P_UIFR[1] = CTRANS(SPI(1), 33.33, X,1) : CSCALE(SPI(1),33.33):CMIRROR(SPI(1))
becomes
$P_UIFR[1]=CTRANS(X,1,A,33.33):CSCALE(A,33.33):CMIRROR(A) when decompiled.

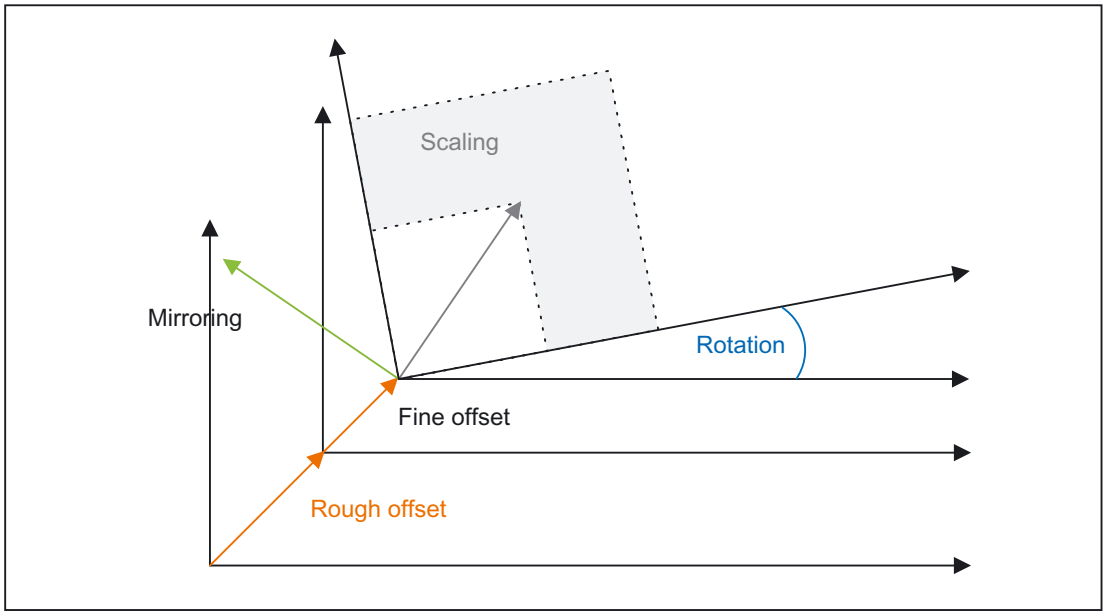
If a spindle and an assigned axis are programmed in a frame instruction, the alarm 16420 "Axis % multiply programmed" is output.

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)
```

2.4.3.8 Coordinate transformation



The formulae below are used to discover the coordinate transformation for geometry axes:

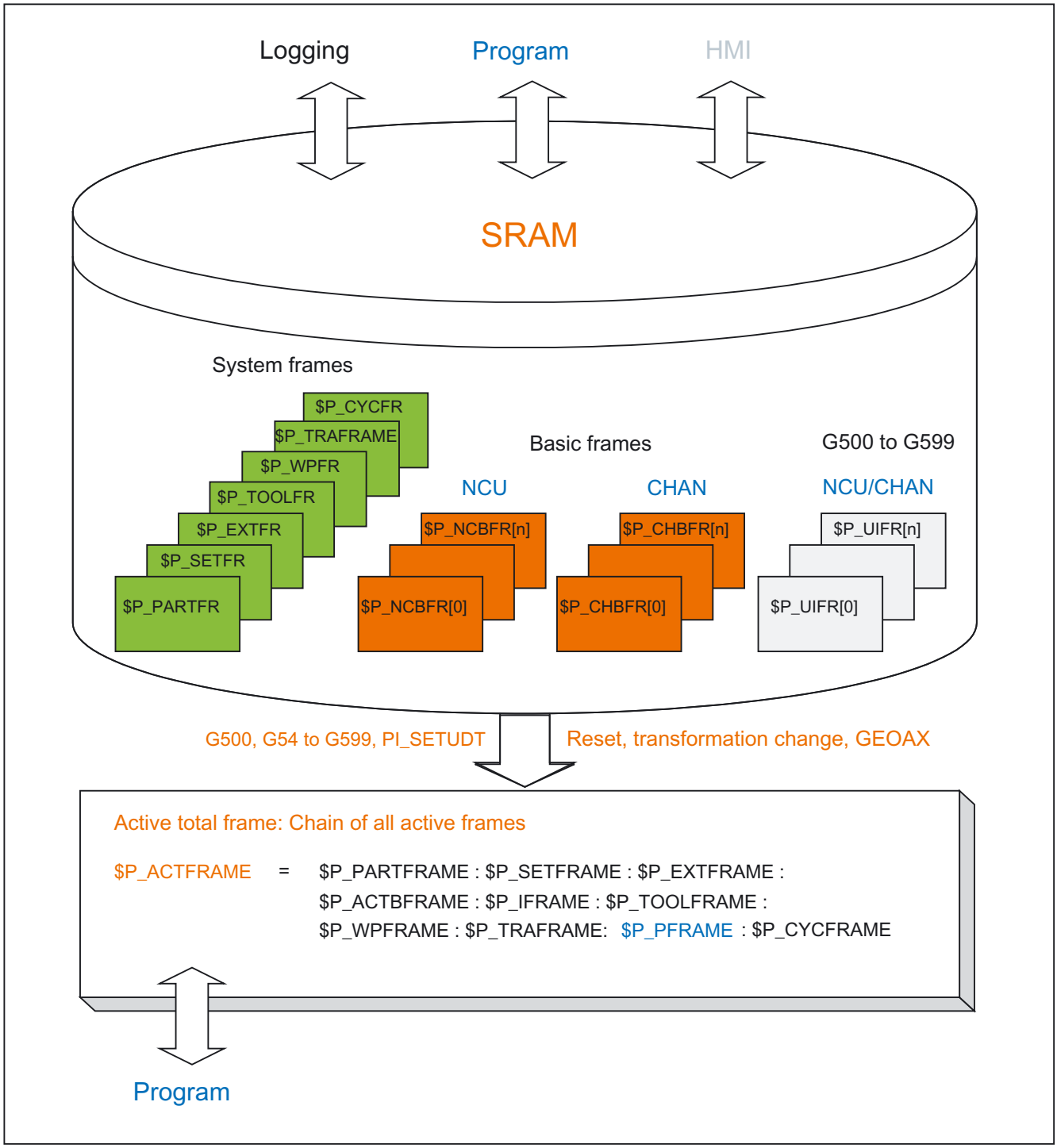
WKS → BKS	$\vec{v} = R * \underline{S} * \underline{M} * \vec{v}' + \vec{t}$
BKS → WKS	$\vec{v}' = inv(\underline{M}) * inv(\underline{S}) * inv(\underline{R}) + (\vec{v} - \vec{t})$

V: Position vector in BCS
V': Position vector in WCS

2.4.4 Frames in data management and active frames

2.4.4.1 Overview

There are various types of frame: system frames, basic frames, settable frames and the programmable frame. Apart from the programmable frame, all types have a frame in the data management and an active frame. The programmable frame is only active. Frames in the data managements are stored in the SRAM and can be archived. The program can describe both data management frames and active frames. HMI can only describe data management frames.


Logging
Program
HMI
SRAM
System frames
$P_CYCFR
$P_TRAFRAME
$P_WPFR
$P_TOOLFR
$P_EXTFR
$P_SETFR
$P_PARTFR
Basic frames
NCU
$P_NCBFR[n]
$P_NCBFR[0]
CHAN
$P_CHBFR[n]
$P_CHBFR[0]
G500 to G599
NCU/CHAN
$P_UIFR[n]
$P_UIFR[0]
G500, G54 to G599, PI_SETUDT
Reset, transformation change, GEOAX
Active total frame: Chain of all active frames
$P_ACTFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ACTBFRAME : $P_IFRAME : $P_TOOLFRAME :
$P_WPFRAME : $P_TRAFRAME: $P_PFRAME : $P_CYCFRAME
Program

2.4.4.2 Activating data management frames

Data management frames become active frames by executing G500, G54 to G599, or RESET with the appropriate machine data setting, transformation change, GEOAX. The HMI writes to data management frames and, on RESET, activates the frames by means of a PI service.

If a change is made to a frame via HMI with the aid of the PI _N_SETUDT, it has no effect on the program until it is restarted or is in a reset state, provided that HMI machine data MD9440 ACTIVATE_SEL_USER_DATA
is set.

WCS position specifications do not change immediately when a change is made to a frame via HMI, but after restart.

System frames are activated during the preprocessing stage by executing the appropriate system functions, either from the part program or using the HMI.

System frames, which are saved in the data management, are also activated by means of a G500, G54 to G599 instruction. The cycle programmer has the option to modify the system frames and activate them by means of a G500, G54 to G599 instruction. However, this option should only be used with reservation.

Activation of all data management frames can be changed with the machine data MD24050 $MC_FRAME_SAA_MODE.

Bitmask for Save and Activate data management frames	
Bit 0:	Data management frames are only activated by programming bitmasks $P_CHBFRMASK, $P_NCBFRMASK and $P_CHSFRMASK. G500 to G599 only activates the corresponding settable frame. Reset response is independent of this.
Bit 1:	Data management frames are not implicitly described by system functions, such as TOROT, PAROT, zero offset external and transformations.

The **$P_CHSFRMASK** variable is used to activate system frames from the data management.

The variable value is specified as a bit code, corresponding to the machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK

If the corresponding bit is set to one, the data management frame is activated. If the bit is equal to zero, the data management frame is not active and the currently active system frame remains active.

After RESET, the system frames for PRESET and zero offset external are activated, whose bits are set in the machine data
MD24006 $MC_CHSFRAME_RESET_MASK.

The system frames for TCARR, PAROT and TOROT, TOFRAME are activated according to the setting in the machine data
MD20150 $MC_GCODE_RESET_VALUES[ ].

As when selecting and deselecting transformations and GEOAX( ), when switching geometry axes the current complete frame is either deleted or recalculated on the basis of the new geometry axis configuration, and activated. The system frames and all other frames are conditioned again in relation to the geometry axes.

2.4.4.3 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 as global frames via machine data MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES.

If the value of this machine data is greater than zero, there are no channel-specific settable frames.

Machine data
MD28080 $MC_MM_NUM_USER_FRAMES
is then irrelevant and is not evaluated.

The number of global basic frames is configured via machine data
MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES.

The machine data
MD28081 $MC_MM_NUM_BASE_FRAMES
can also make channel-specific basic frames available simultaneously.

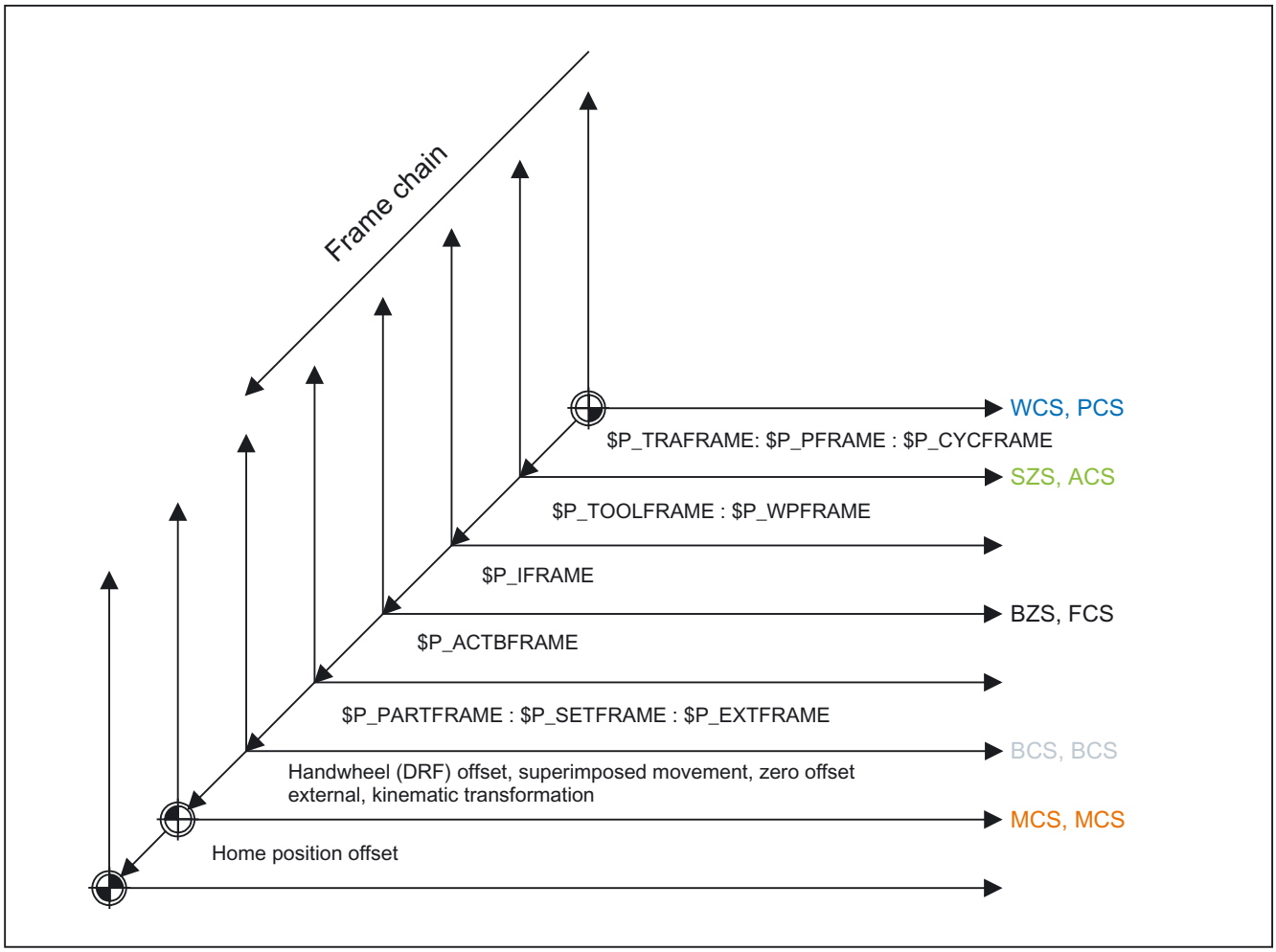
Global frames can be read and written from all channels of an NCU. When writing global frames, the user must ensure channel coordination. This can be achieved using wait markers, for example.

2.4.5 Frame chain and coordinate systems

2.4.5.1 Overview

The figure below shows the frame chain for the current complete frame. The frame chain is stored between the BCS and WCS. The SZS (**S**ettable **Z**ero **S**ystem) corresponds to the WCS, transformed by the programmable frame. The BZS (**B**asic **Z**ero **S**ystem) still includes the current settable frame. The system frame for the zero offset external is only available if it has been configured, otherwise the zero offset external is interpolated as a superimposed motion of the axis, as it has been up to this point.

WCS:	**W**orkpiece **C**oordinate **S**ystem	PCS:	**P**art **C**oordinate **S**ystem
SZS:	**S**ettable **Z**ero **S**ystem	ACS:	**A**djustable **C**oordinate **S**ystem
BZS:	**B**asic **Z**ero **S**ystem	FCS:	**F**oot **C**oordinate **S**ystem
BCS:	**B**asic **C**oordinate **S**ystem	BCS:	**B**asic **C**oordinate **S**ystem
MCS:	**M**achine **C**oordinate **S**ystem	MCS:	**M**achine **C**oordinate **S**ystem

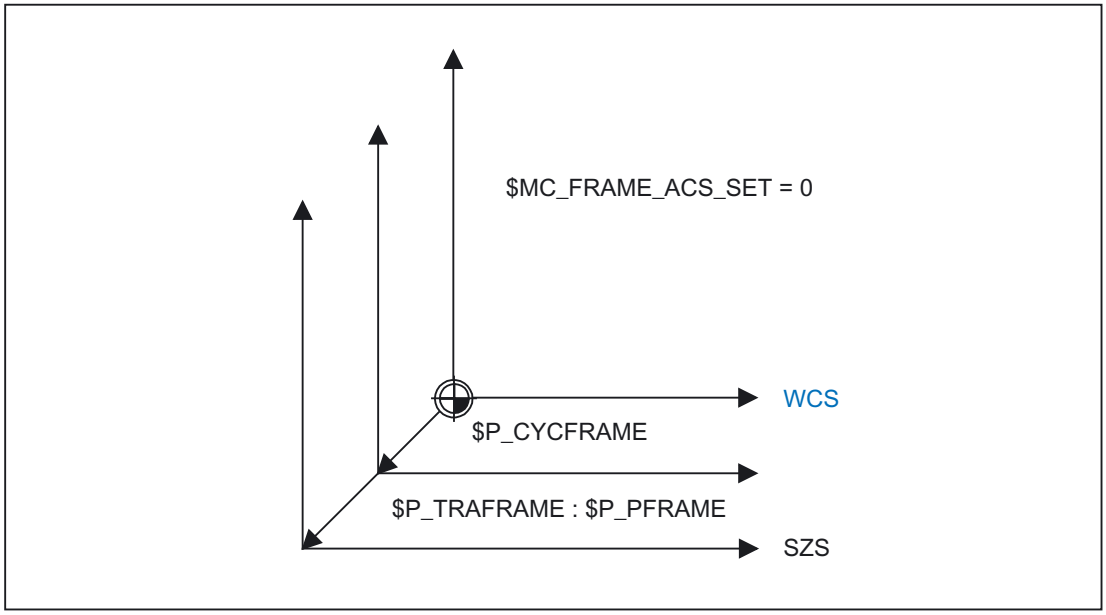


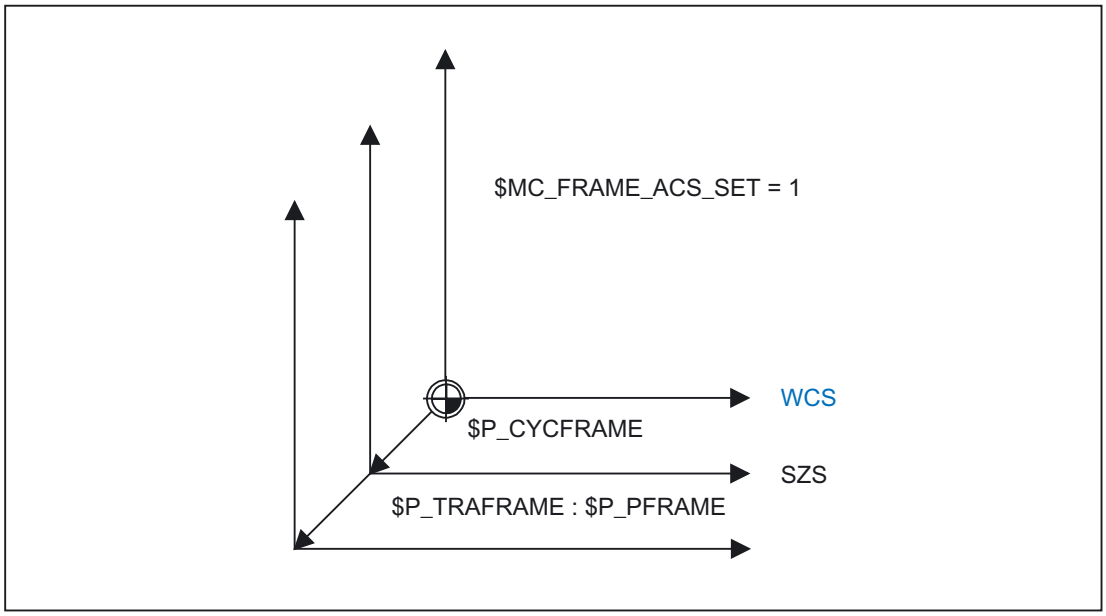
The current complete frame is calculated according to the formula below:

$P_ACTFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ACTBFRAME : $P_IFRAME :
$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME :
$P_PFRAME : $P_CYCFRAME

2.4.5.2 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.





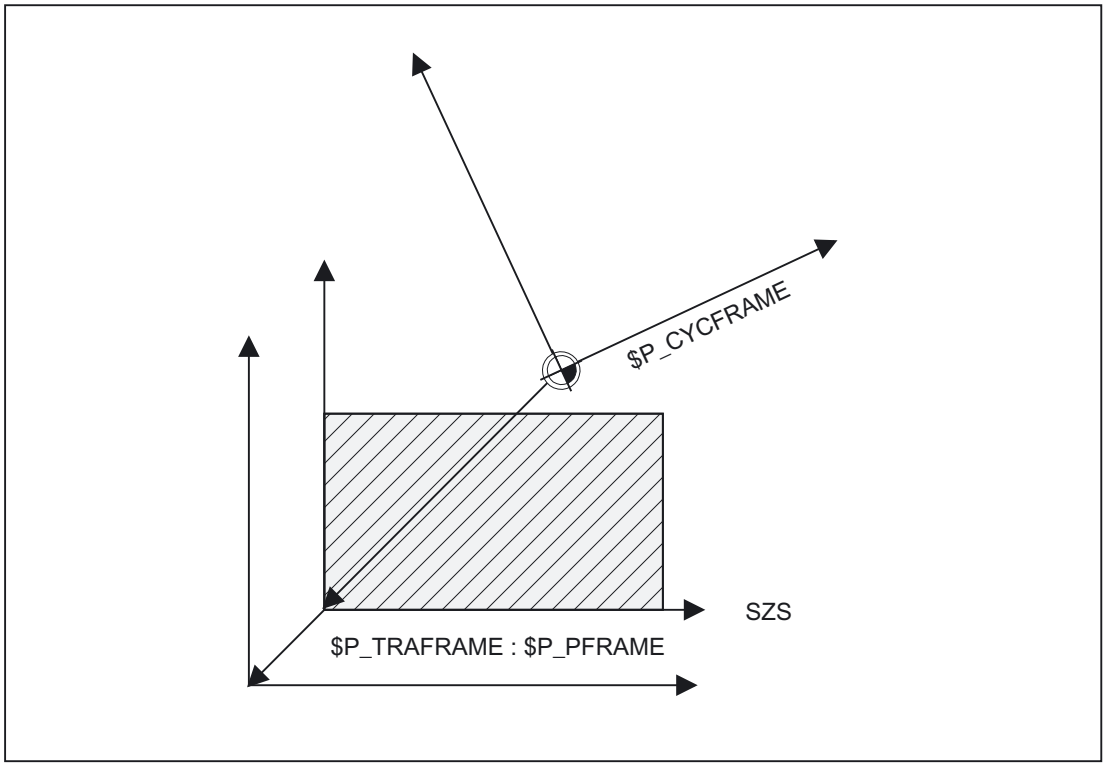
The machine data
MD24030 $MC_FRAME_ACS_SET
can be used to set whether the SZS is with or without the prog. frame and the transformation frame.

The value 0 is only made available for reasons of compatibility. The value 1 is set as a default, and should be retained. All cycles should be reset so that they only use the cycle frame.

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.

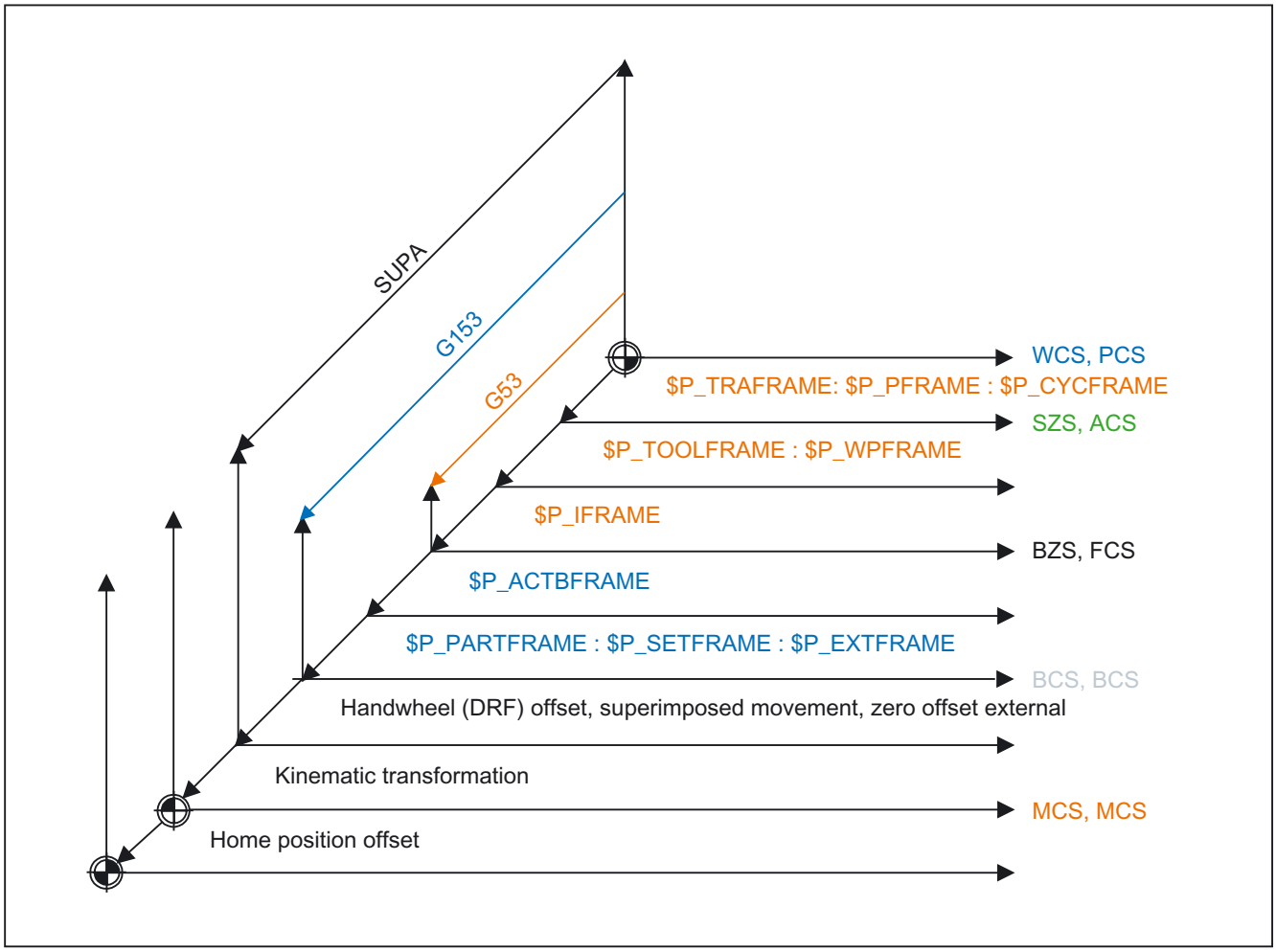
2.4.5.3 Manual traverse in the SZS coordinate system

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.



2.4.5.4 Suppression of frames

G53	Non-modal suppression of the following frames: System frame for cycles Programmable frame System frame for transformations, workpieces, TOROT and TOFRAME Active settable frame
G153	Non-modal 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], superimposed motion
G500	Modal activation of the G500 frame. The G500 frame should be a zero frame.
DRFOF	Deactivate (clear) the handwheel offsets (DRF)



Frame suppressions SUPA, G153 and G53 lead to the WCS, SZS and possibly the BZS jumping when frame suppression is active.

The machine data
MD24020 $MC_FRAME_SUPPRESS_MODE
enables this characteristic for the position display and the predefined position variables to be changed.

The settings below can be made with MD24020 $MC_FRAME_SUPPRESS_MODE:

Bit 0:	Positions for display (OPI) are without frame suppression.
Bit 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.

2.4.6 Frame chain frames

2.4.6.1 Overview

There are up to four frame variants:

- Settable frames (G500, G54 to G599)
- Basic frames
- Programmable frame
- System frames

2.4.6.2 Settable frames $P_UIFR[n]

The number of NCU global settable frames is set via machine data
MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES.
The number can be between 0 and 100.
If the machine data has a value greater than zero, there are only NCU global settable frames, otherwise the machine data
MD28080 $MC_MM_NUM_USER_FRAMES
specifies the number of channel-specific settable frames.

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.

2.4.6.3 Channel basic frames $P_CHBFR[n]

The number of basic frames can be configured in the channel via machine data MD28081 $MC_MM_NUM_BASE_FRAMES.
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. The chained complete basic frame is not activated simultaneously when writing a basic frame field element, but rather activation only takes place on execution of a `G500,G54,` to `G599` instruction. The variable is used primarily for storing write operations to the basic frame from 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,` 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 to the basic frame from 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.

2.4.6.4 NCU global basic frames $P_NCBFR[n]

The number of global basic frames can be configured via machine data MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES.
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. The chained complete basic frame is not activated simultaneously when writing a basic frame field element, but rather activation only takes place on execution of a `G500,G54,` to `G599` instruction. If the modified frame is to be active in every channel of the NCU, every channel must execute a `G500,G54` to `G599` instruction. The variable is used primarily for storing write operations to the basic frame from 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:

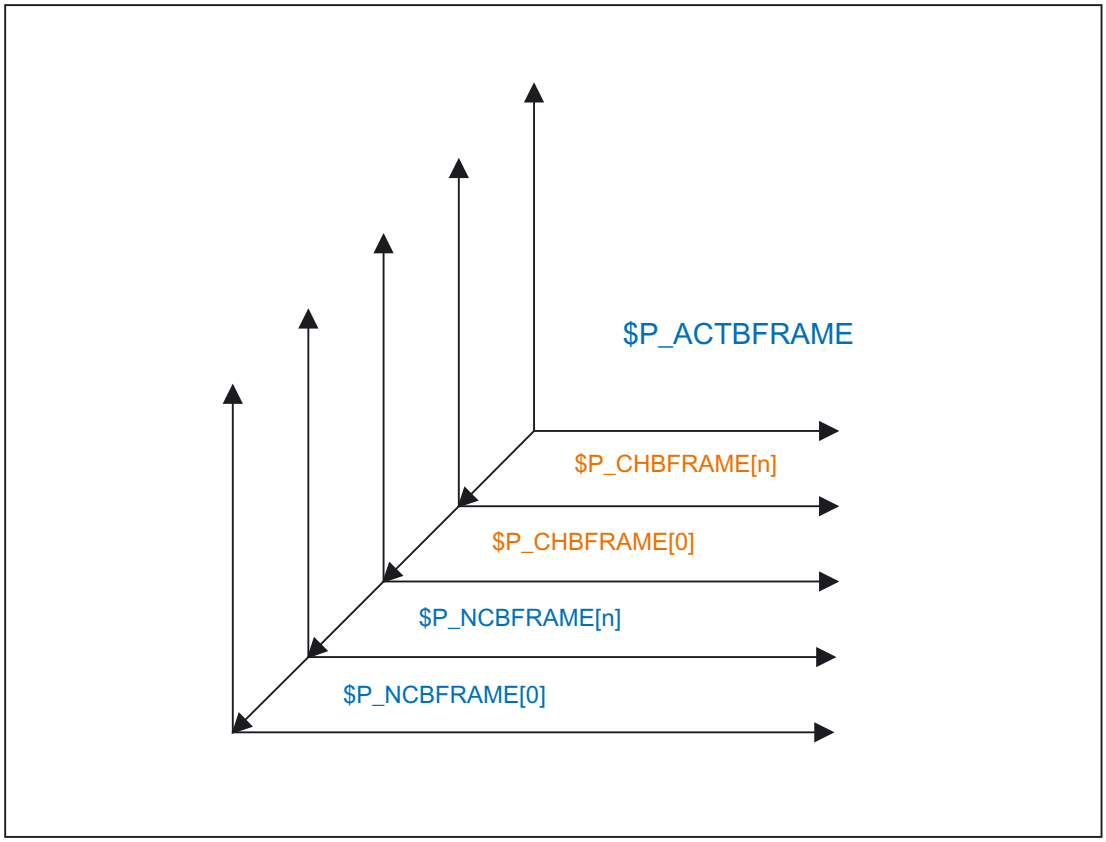
```
$P_NCBFR[0] = CTRANS( x, 10 ):CROT( y, 45   ; Faulty assignment on the global basic
)                                             frame
```

The frames
$P_UBFR, $P_BFRAME, $P_CHBFR[n],
$P_CHBFRAME[n], $P_NCBFRAME[n],
and
$P_ACTBFRAME und $P_ACTFRAME
are channel-specific.
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.

2.4.6.5 Complete basic frame $P_ACTBFRAME

The chained complete basic frame is determined by the variable. The variable is readonly.



$P_ACTBFRAME
corresponds to
$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 operator panel interface. 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]
```

2.4.6.6 Programmable frame $P_PFRAME

The programmable frame is only available as an active frame.
This frame is reserved for the programmer.

The programmable frame can be retained on RESET using the machine data setting MD24010 $MC_PFRAME_RESET_MODE = 1.
This functionality is particularly important if one wishes to retract from an oblique hole after RESET.

MIRROR

Previously (up to SW P4), mirroring of a geometry axis had been related to a defined reference axis using machine data
MD10610 $MN_MIRROR_REF_AX.

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 no longer mapped onto the mirroring of a reference axis and rotations of other axes.
This setting can be configured using the machine data setting
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 the mirror and a further AMIRROR X0 deactivates it. MIRROR always has an absolute effect and AMIRROR an additive effect.

Machine data setting
MD10612 $MN_MIRROR_TOGGLE = 0
can be used to define that the programmed values should be evaluated.
A value of 0, i.e., 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.

It is possible to read or write mirrors component by component independent of 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          ; X axis is always mirrored.
$P_NCBFR[0,x,mi] = 0          ; X axis mirror is OFF.
```

Axial replacement G58, G59

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 for axial overwriting of the additively programmed translations for the specified axes, which were programmed with ATRANS.

G59 X... Y... Z... A... ...

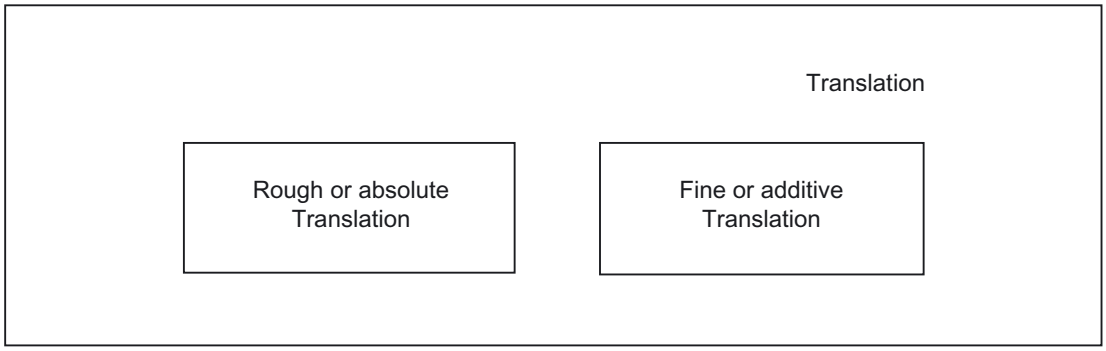
Example:

```
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 machine data
MD24000 $MC_FRAME_ADD_COMPONENTS
is set to TRUE, otherwise the 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

2.4.6.7 Channel-specific system frames

Functionality

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 using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK.

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

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 SRAM and can, therefore, be archived and reloaded. System frames in data management can be read and written in the program using the following variables:

$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)

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 aG500, 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 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.
  If the system frame has not been configured using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK,
  the variable returns a zero frame.

- $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.

  If the system frame has not been configured using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK,
  the variable returns a zero frame.

- $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.
  If the system frame has not been configured using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK,
  the variable returns a zero frame.

- $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.
  If the system frame has not been configured using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK,
  the variable returns a zero frame.

- $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.
  If the system frame has not been configured using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK,
  the variable returns a zero frame.

- $P_CYCFRAME

  In the part program, the variable $P_CYCFRAME can be used to read and write the current system frame for cycles.
  If the system frame has not been configured using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK,
  the variable returns a zero frame.

- $P_TRAFRAME

  In the part program, the variable $P_TRAFRAME can be used to read and write the current system frame for transformations.
  If the system frame has not been configured using machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK,
  the variable returns a zero frame.

2.4.6.8 $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_ACTBFRAME : $P_IFRAME :
$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME :
$P_PFRAME : $P_CYCFRAME

2.4.7 Implicit frame changes

2.4.7.1 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, scales and mirrors 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.
  The aspects described in the Chapter "Frames for selection and deselection of transformations" are relevant to TRANSMIT, TRACYL and TRAANG.

  References:
  /FB1/ Description of Functions, Basic Machine; Axes, Coordinate System, Frames (K2); Chapter: Frame for selection and deselection of transformations

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

  The current complete frame is calculated again when the geometry axes are switched over, and the translations, scales and mirrors 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".
  The aspects described in the Chapter "Frames for selection and deselection of transformations" are relevant to TRANSMIT, TRACYL and TRAANG.

  References:
  /FB1/ Description of Functions, Basic Machine; Axes, Coordinate System, Frames (K2); Chapter: Frame for selection and deselection of transformations

- 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, scales and mirrors 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), scales and mirrors 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

```
$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:

```
$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)
```

2.4.7.2 Frame for selection and deselection of transformations

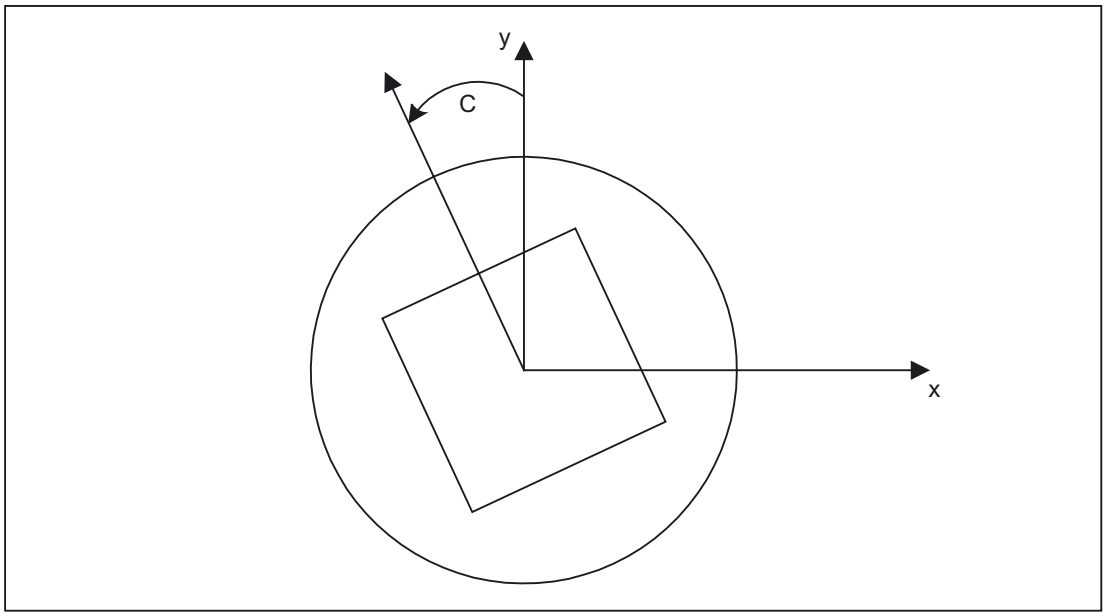
This function is available with NCK 51.00.00 and higher. Transformations `TRANSMIT`, `TRACYL` and `TRAANG` are supported.

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 machine data
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.

TRANSMIT



Transmit expansions:

The machine data
MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 1
MD24905 $MC_TRANSMIT_ROT_AX_FRAME_2 = 1
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.

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

MD24905 $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 rotations. This setting is only effective if the transformation frame has been configured.

Frame expansions:

The expansions described below are only valid for the machine data

$MN_FRAME_GEOAX_CHANGE_MODE = 1

$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:

On selecting transmit, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.

Rotations:

Rotations before the transformation are taken over.

Mirrorings:

Mirrorings of the virtual axis are deleted. Mirrorings of the rotary axis can be taken into account in the transformation.

Scalings:

Scalings of the virtual axis are deleted. Scalings 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 on 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
```

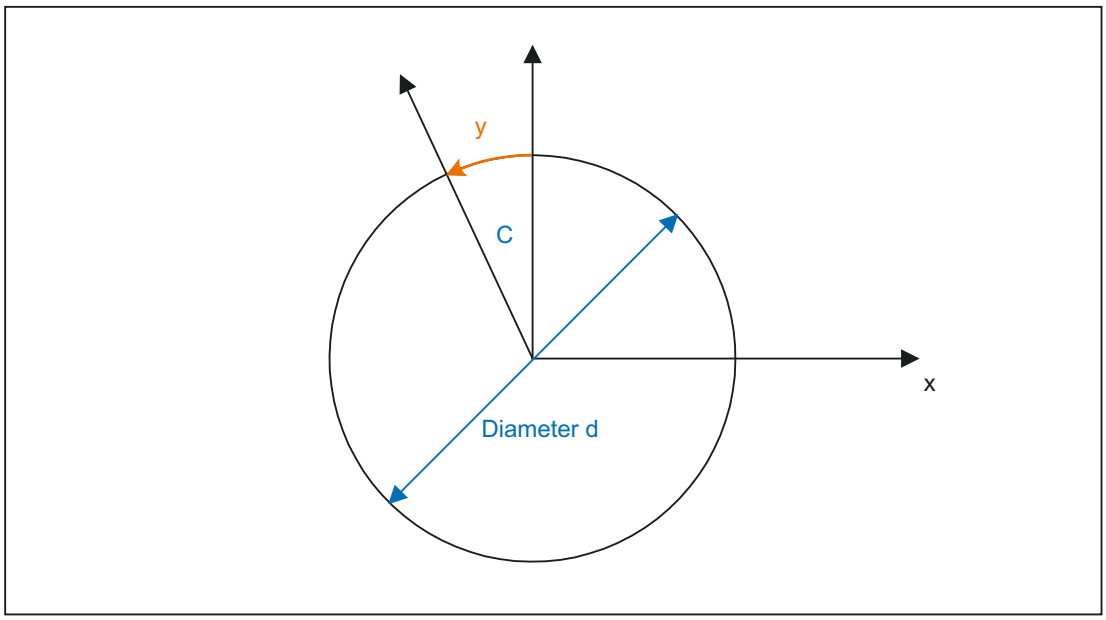
```
; Change tool
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490
```

```
; Square 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 sheath surface. This setting is only effective if the transformation frame has been configured.

Frame expansions:

The expansions described below are only valid for the machine data

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

The selection of transformation `TRANSMIT` produces a virtual geometry axis on the sheath surface, 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. All components of the virtual geometry axis are deleted. All other geometry axes accept their axial components when the transformation is selected.

Translations:

On selecting tracyl, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.

Rotations:

Rotations before the transformation are taken over.

Mirrorings:

Mirrorings of the virtual axis are deleted. Mirrorings of the rotary axis can be taken into account in the transformation.

Scalings:

Scalings of the virtual axis are deleted. Scalings 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 on 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, resurface by grinding selected
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
```

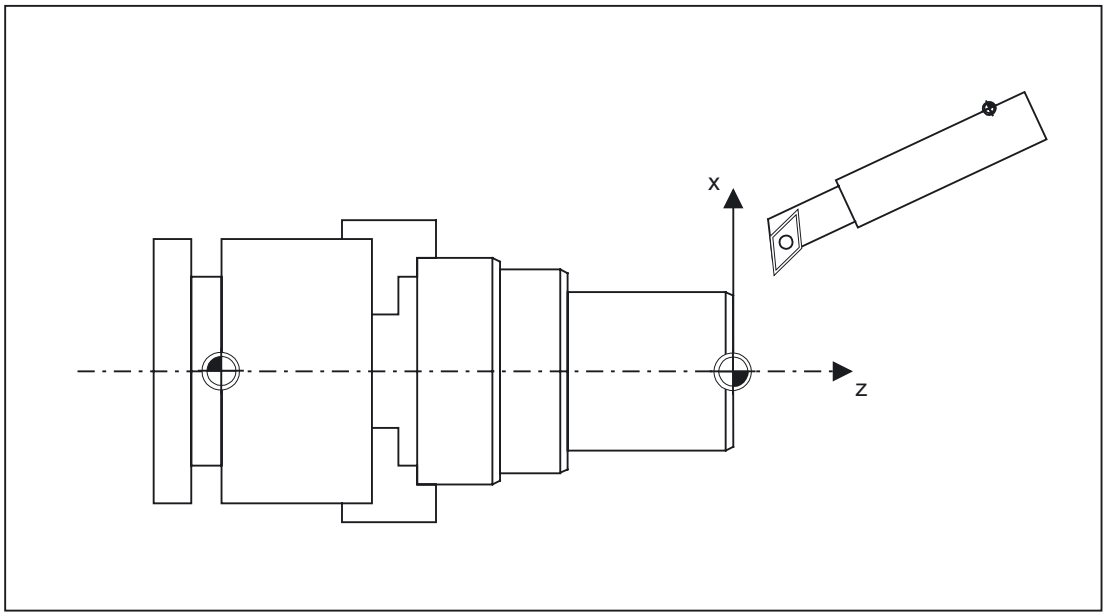
```
N700 if $P_IFRAME <> $P_UIFR[1]
N710 setal(61000)
N720 endif
N730 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N740 setal(61000)
N750 endif
N760
```

```
; 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 machine data

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

Translations:

On selecting traang, translations of the virtual axis are retained.

Rotations:

Rotations before the transformation are taken over.

Mirrorings:

Mirrorings of the virtual axis are taken over.

Scalings:

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 on 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                     ; Oblique 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
```

```
; Change tool
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490
```

```
; Square 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
```

2.4.7.3 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
```

2.4.8 Predefined frame functions

2.4.8.1 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
```

2.4.8.2 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 )
Parameter 1:        Type: FRAME     ; Additive measured or calculated frame
Parameter 2:        Type: STRING    ; Strings for current frames:
                                    ; "$P_CYCFRAME", "$P_PFRAME", "$P_WPFRAME",
                                    ; "$P_TOOLFRAME", "$P_IFRAME",
                                    ; "$P_CHBFRAME[0..16]", "$P_NCBFRAME[0..16]",
                                    ; "$P_EXTFRAME", "$P_SETFRAME"
                                    ; "$P_PARTFRAME"
                                    ; Strings for data management frames
                                    ; "$P_CYCFR", "$P_TRAFR", "$P_WPFR",
                                    ; "$P_TOOLFR", "$P_UIFR[0..99]",
                                    ; "$P_CHBFR[0..16]", "$P_NCBFR[0..16]",
                                    ; "$P_EXTFR", "$P_SETFR", "$P_PARTFR"

Function value      Type: INT       ; 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 such that the new complete frame comprises 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.

2.4.9 Functions

2.4.9.1 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.

2.4.9.2 Zero offset external via system frames

Previous function:

The zero offset external is either defined by the PLC via the OPI or programmed in the part program by the axis variable $AA_ETRANS[axis] = value. This zero offset is activated by the PLC via a VDI signal.

After the PLC has activated the offset, every axis compensation value is applied with the next block. With G64, in certain circumstances this could also be multiple blocks, if there is no Stop at the block end.

Functional expansion:

With SW package 6 and higher, the zero offset external can be managed and enabled via a system frame.
The system frame is configured with bit 1 in machine data
MD28082 $MC_MM_SYSTEM_FRAME_MASK = 'B0010'.

If this bit is set, the system function will only use the system frame, disabling the existing functionality. Values are written by the PLC (OPI) or, in the part program, with axial variables $AA_ETRANS[axis].
As previously, activation is by the PLC, via an axial VDI signal.

If a level change of the axis signal from 0 to 1 is detected, the movement is stopped immediately, the preprocessing is reorganized and the current system frame is written and activated with the axis value of $AA_ETRANS[axis]. The system frame in the data management is also described. The offset is then first applied and the interrupted motion is continued.
With G91 and machine data setting
MD42440 $MC_FRAME_OFFSET_INCR_PROG = 0,
the zero offset external is also applied with the approach block.
Special treatment when programming incrementally is not relevant for the zero offset external. Activation always leads to the offset being applied immediately.

The zero offset external acts on the absolute translation absolutely (coarse offset) of the current system frame. Therefore, a multiple activation of a zero offset external does not act additively, and only the coarse component of the translation is overwritten with the value $AA_ETRANS[axis]. The current system frame is the system frame currently active in the main run.

2.4.9.3 Toolholder

Translations

With kinematics of type P and M the selection of a toolholder activates an additive frame (table offset of the orientational toolholder), which takes into account the zero point offset as a result of the rotation of the table.
The zero offset is entered into a system frame ($P_PARTFR).
The translational part of this frame is then overwritten.
Other contents of the frame are left unchanged.
In order to be able to use this system frame, bit 2 must be set in the machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK.

Alternatively, there is the option to enter this offset into the basic frame identified by machine data
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER.
This option is available in the interests of compatibility with older software versions.
It is not recommended for use with new systems.

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 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 movement by the machine itself.

The ratios are shown in the figure below:

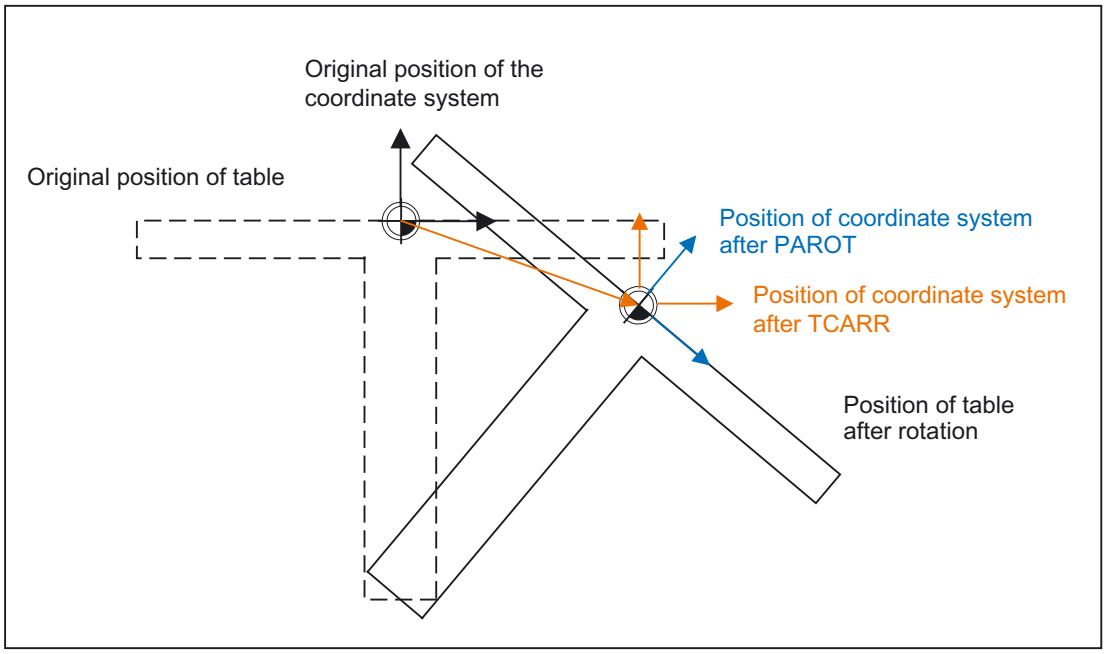


Fig. 2-25 Frame on activation of a rotary table with TCARR

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 an orientational toolholder 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.

Up to and including SW P6.1, the rotation activated by PAROT is calculated in the programmable frame ($P_PFRAME), thus changing its rotation component.

With SW P6.2 and higher, the entire programmable frame remains unchanged, including its rotation component.
The rotation component, which describes the rotation of the tool table, is then entered into system frame $PARTFR, if bit 2 of machine data
MD28082 $MC_MM_SYSTEM_FRAME_MASK
is set.
Alternatively, the basic frame described in machine data
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER
can also be used.

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, independently of whether this frame is found 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 orientational 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 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.

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 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.

Language command PAROT is not rejected if no orientational toolholder is active. However, such a call then produces no frame changes.

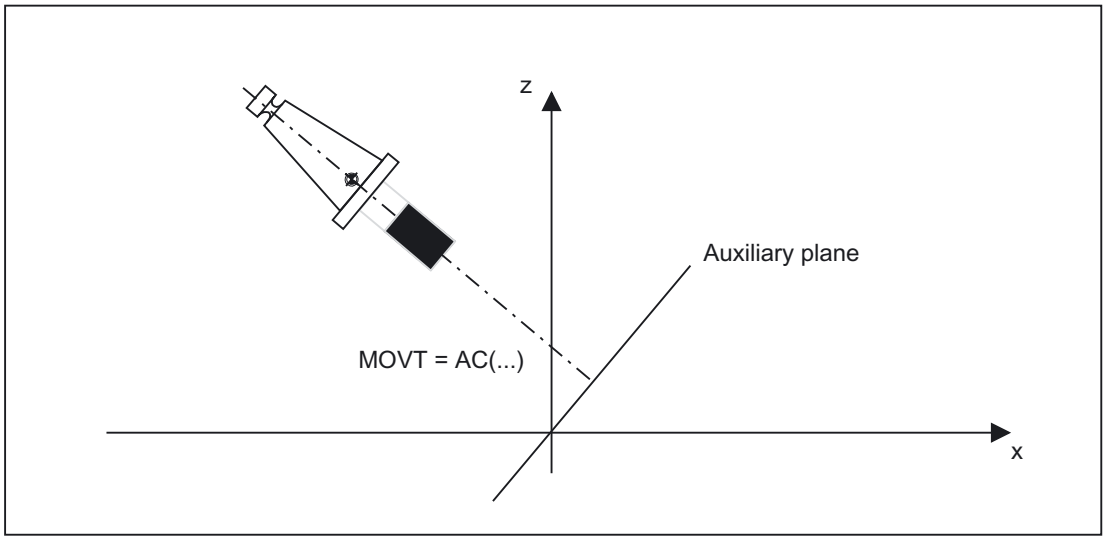
Machining in direction of tool orientation

Particularly on machines with orientational tools, 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 oblique machining operations, a frame defining the oblique 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 actually requested motion perpendicular to the plane - to drill in 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.

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 an orientational toolholder. 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 both where no orientational toolholder is active and for the case of an orientational 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 movement of MOVT.
With an active 5-axis transformation, the path of the tool center point (TCP) is not affected by the change or orientation, i.e., the path remains a straight line and its direction 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.

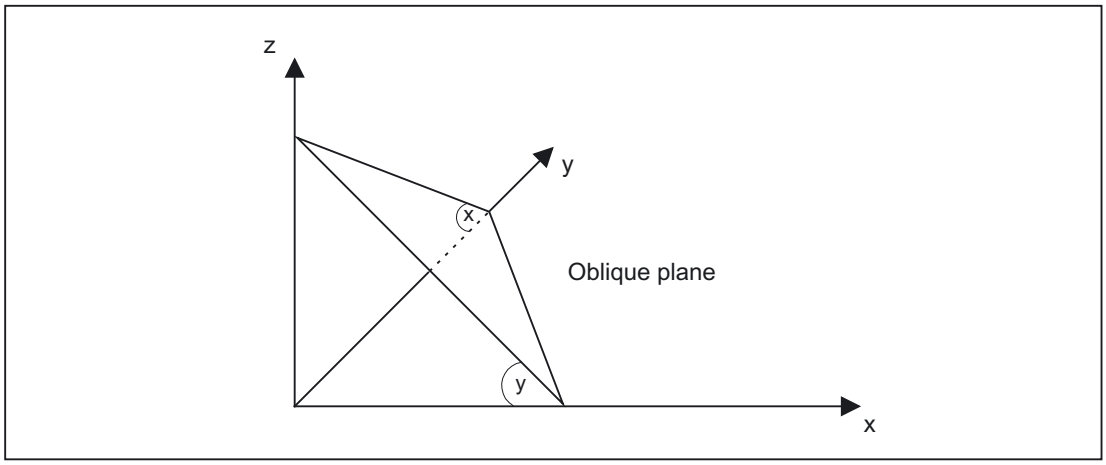
A block with MOVT must not contain any programming of geometry axes (alarm 14157).

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 is also the case when programming in a block, e.g., with ROT X... Y... Z...
(the sequence of rotations is defined by machine data
MD10600 $MN_FRAME_ANGLE_INPUT_MODE
and is independent of the sequence of the axis letters in the block),
and when constructing a frame in multiple blocks, e.g., in the format:

```
N10 ROT Y...
N20 AROT X...
N30 AROT Z...
```

In workpiece drawings, oblique surfaces are frequently described by way of solid angles, i.e., the angles, which the intersection lines of the oblique 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.



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 2 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 here gives the angle, by which the X axis must rotate around the Y axis to bring the X axis to the line of intersection formed by the oblique plane and the XZ plane. The same principle applies for the programmed value of X.

Note

In the shown position of the oblique 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 surrounded 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. . . .

On programming the solid angles, they are converted into the equivalent RPY or Euler angles, depending on machine data
MD10600 $MN_FRAME_ANGLE_INPUT_MODE.
These then appear also 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. As a result of this, the G codes below are newly introduced into G code group 53:

```
TOFRAMEX
TOFRMAEY
TOFRAMEZ
TOROTX
TOROTY
TOROTZ
```

These G codes enable corresponding frames to be defined. The functions of TOFRAME and TOFRAMEZ or of TOROT and TOROTZ are identical to one another.

With software version P6 and higher, there is the option to write frames produced by TOROT or TOFRAME into their own system frame ($P_TOOLFR).
For this, bit 3 must be set in machine data
MD28082 $MC_MM_SYSTEM_FRAME_MASK.
The programmable frame is then retained unchanged.

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.

In new systems, it is recommended that only the intended system frame is used for frames produced by G codes in 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 Hirthtoothed rotary axes. The language commands uniquely define the Z direction of the new frame.

In previous software versions, machine data
MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE
can be used to choose between 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.

Setting data
SD42980 $SC_TOFRAME_MODE
is, therefore, introduced, which can be used to control the response of TOFRAME and TOROT. It can accept values of 0 (inactive) to 3. If the value of the setting data is not zero, the effect of machine data
MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE is overwritten.

The values 1 to 3 have the following meanings:

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 Y 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.

Characteristics and expansions:

Settings 1 and 2 are reached by rotating the coordinate system around the Z axis, starting from any position on the X and Y axis, until the desired setting is reached. Setting 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 variant 1, both the old and new X axes form an angle of under 90 degrees, with variant 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 G codes 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:

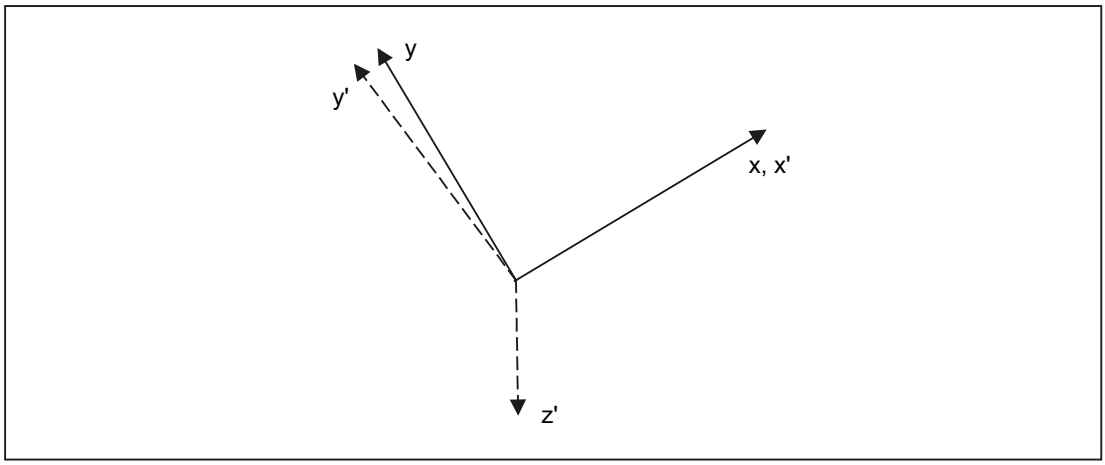
	TOFRAME, TOFRAMEZ TOROT, TOROTZ	TOFRAMEY TOROTY	TOFRAMEX TOROTX
Tool direction (vertical axis)	Z	Y	X
Secondary axis (horizontal axis)	X	Z	Y
Secondary axis (ordinate)	Y	X	Z

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).

In the figure below, the situation for setting data setting
SD42980 $SC_TOFRAME_MODE=1
is shown:



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).

For setting data setting
SD42980 $SC_TOFRAME_MODE=2,
Y and Y' would coincide accordingly and X and X' would form an angle of 8.13 degrees.

For setting data setting
SD42980 $SC_TOFRAME_MODE=3,
X and X', as well as Y and Y', would 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 and PAROT

Previously, TCARR has used the basic frame identified by 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 frame. In this case, it is not possible to shut down PAROT or TOROT separately. On RESET, the programmable frame is deleted, which means the 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 bit 2 in machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK.

Machine data
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER
is then no longer evaluated.

If the system frame for TCARR is configured, TCARR and PAROT describe that system frame; otherwise the basic frame identified by machine data
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER
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 such, that a partoriented WCS results.

The system frames are stored in the SRAM and, therefore, remain stored after Reset. The system frames also remain active in the case of a mode change.

For the display, the G codes PAROT and TOROTOF, 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 code group as PAROT and appears, therefore, in the G code display.

TOROT and TOFRAME

The system frame for TOROT and TOFRAME is configured with bit 3 in machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK.
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 system frame for TOROT and TOFRAME. The current $P_TOOLFRAME and the data management frame $P_TOOLFR are also deleted. TOROTOF is in the same G code 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 components of 1st offset vector
N40          $TC_CARR2[1] = 0                ; Y components of 1st offset vector
N50          $TC_CARR3[1] = 0                ; Z components of 1st offset vector
N60          $TC_CARR4[1] = 0                ; X components of 2nd offset vector
N70          $TC_CARR5[1] = 0.               ; Y components of 2nd offset vector
N80          $TC_CARR6[1] = -15.              ; Z components of 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 components of 3rd offset vector
N180         $TC_CARR16[1] = 0               ; Y components of 3rd offset vector
N190         $TC_CARR17[1] = 0               ; Z components of 3rd offset vector
N200         $TC_CARR18[1] = 0               ; X components of 4th offset vector
N210         $TC_CARR19[1] = 0               ; Y components of 4th offset vector
N220         $TC_CARR20[1] = 15.             ; Z components 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] = "M"             ; Toolholder type
N260         X0 Y0 Z0 A0 B45 F2000
N270         TCARR=1 X0 Y10 Z0 T1 TCOABS     ; Selection of orient. toolholder
N280         PAROT                           ; Rotation of table
N290         TOROT                           ; Rotation of z axis in tool orient.
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 with tool
N400M30
```

2.4.10 Subroutine return with SAVE

Settable frames G54 to G599

If the same G code is active on the subroutine return as in the subroutine call, then the active settable frame is retained.

If this is not the case, the settable frame at the instant the subroutine was called is reactivated (response as now).

The new response can be deactivated by setting bit 0 in machine data MD10617 $MN_FRAME_SAVE_MASK.

Basic frames $P_CHBFR[ ] and $P_NCBFR[ ]

The active total basic frame is retained on return from a subroutine.

The new response can be deactivated by setting bit 1 in machine data MD10617 $MN_FRAME_SAVE_MASK.

Programmable frame

The programmable frame is recreated when the subroutine return is carried out.

System frames $P_TRAFR, $P_SETFR, $P_EXTFR, $P_PARTFR, $P_TOOLFR, $P_WPFR, $P_CYCFR

These system frames are not affected by the SAVE attribute. A compatibility machine data is not provided.

2.4.11 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.

2.4.12 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].

2.4.13 Control system response

2.4.13.1 Power on

The table below describes the frame states after POWER ON:

Programmable frame	Deleted
Settable frames	Retained, depending on MD20110 $MC_RESET_MODE_MASK
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. System frames are carried out in the first-order in the data management.
Zero offset external	Permanent, but has to be activated again. The system frame is retained.
DRF offset	Deleted

2.4.13.2 Mode change

System frames are retained and remain active when the operating mode is changed. 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.

The response for PLC axes and command axes can be set via machine data MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED.

2.4.13.3 Reset, program end

The Reset response of the basic frame is set via
MD20110 $MC_RESET_MODE_MASK.

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

Bit 0:	System frame for PRESET and scratching is active after RESET.
Bit 1:	System frame for zero offset external is active after RESET.
Bit 2 and bit 3:	Not evaluated.
Bit 4:	System frame for workpiece reference points is active after RESET.
Bit 5:	System frame for cycles is active after RESET.

The reset response of the system frames for TCARR, PAROT, TOROT, and TOFRAME depends on the G code reset setting.
The machine data below are used for this setting:

MD20110 $MC_RESET_MODE_MASK

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 $MC_RESET_MODE_MASK		
Bit 0 = 1 and bit 14 = 0	Chained complete basic frame is deleted.	
Bit 0 = 1 and bit 14 = 1	The complete basic frame results from $MC_BASEFRAMES_RESET_MASK and $MN_BASEFRAMES_RESET_MASK.	
	$MC_BASEFRAMES_RESET_MASK:	
	Bit 0 = 1:	1st channel basic frame is calculated into the chained complete basic frame.
	Bit 7 = 1:	8th channel basic frame is calculated into the chained complete basic frame.
	$MN_BASEFRAMES_RESET_MASK:	
	Bit 0 = 1:	1st NCU global basic frame is calculated into the chained complete basic frame.
	Bit 7 = 1:	8th NCU global basic frame is calculated into the chained complete basic frame.

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 and on 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[ ].**
Zero offset external	Retained
DRF offset	Retained

On RESET, system frames in the data management can be deleted using machine data

MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK

Bitmask for clearing channel-specific system frames in the data management on Reset.

Bit 0: System frame for PRESET and scratching is cleared on RESET.
Bit 1: System frame for zero offset external is cleared on RESET.
Bit 2: Reserved, for TCARR and PAROT see MD20150 $MC_GCODE_RESET_VALUES[ ].
Bit 3: Reserved, for TOROT and TOFRAME see MD20150 $MC_GCODE_RESET_VALUES[ ].
Bit 4: System frame for workpiece reference points is cleared on RESET.
Bit 5: System frame for cycles is cleared on RESET.
Bit 6: Reserved, reset response depends on MD20110 $MC_RESET_MODE_MASK.

2.4.13.4 Response at part-program start

Programmable frame	Deleted
Settable frames	Retained, depending on MD20112 $MC_START_MODE_MASK

Complete basic frame	**Retained**
System frames	**Retained**
Zero offset external	Retained
DRF offset	Retained

2.4.13.5 Block search

Data management frames are also modified when carrying out a block search with calculation.

If the block search is aborted with RESET, machine data
MD28560 $MC_MM_SEARCH_RUN_RESTORE_MODE
can be used to configure all data management frames to be reset to the values they had before the block search.
With cascaded block searches, frames are set to the status of the previous block search.
The "SERUPRO" function is not supported.

MD28560 $MC_MM_SEARCH_RUN_RESTORE_MODE

Bitmask for restoring data when a simulated program run is aborted.

The following settings apply:

Bit 0: All frames in the data management are restored.

2.4.13.6 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.

2.5 Workpiece-related actual-value system

2.5.1 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).

2.5.2 Use of 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 MMC 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).

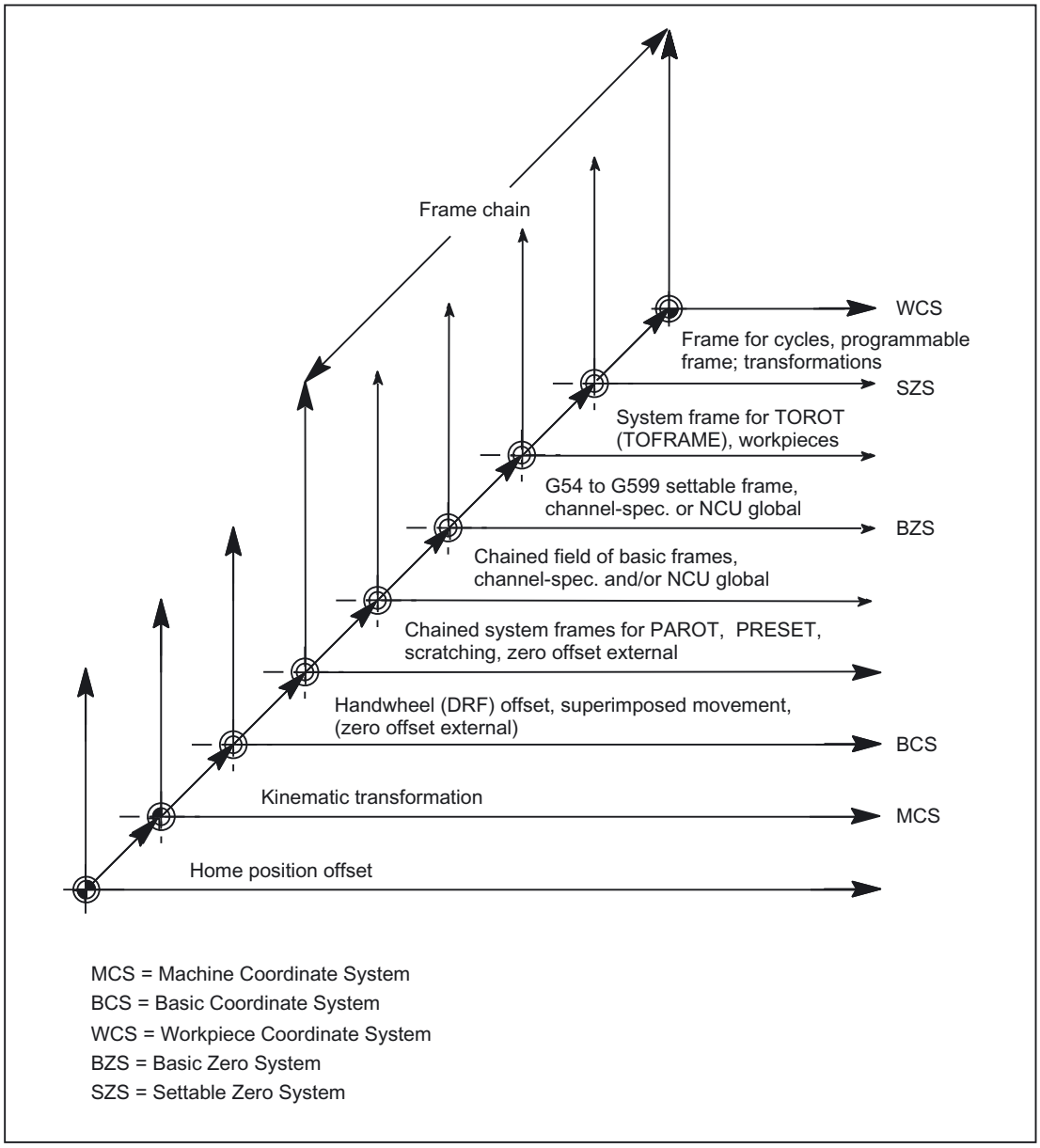


Fig. 2-26 Interrelationship between coordinate systems

References:
/PG/ Programming Guide, Fundamentals
/FB1/ Description of Functions, Basic Machine; Tool Offset (W1)
/FB1/ Description of Functions, Basic Machine; Auxiliary Function Outputs to PLC (H2)
/FB2/ Description of Functions, Expansion Functions; Kinematic Transformation (M1)
/FB3/ Description of Functions, Special Functions; Axis Couplings and ESR (M3);
Chapter: Coupled Motion, Chapter: Master Value Coupling
/FB3/ Description of Functions, Special Functions; Tangential Control (T3)

2.5.3 Special reactions

Overstore

Overstoring in RESET state of:

- Frames (zero offsets)
- Active plane
- Activated transformation
- Tool offset

immediately affects the actualvalue display of all axes in the channel.

MMC inputs

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 MMC machine data
MD9440 ACTIVATE_SEL_USER_DATA
for the operator panel 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 poweron 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.

References:
/FB1/ Description of Functions, Basic Machine; Mode Group, Channel, Program Operation, Reset Response (K1)

Supplementary Conditions

3

3.1 Supplementary conditions

There are no supplementary conditions to note.

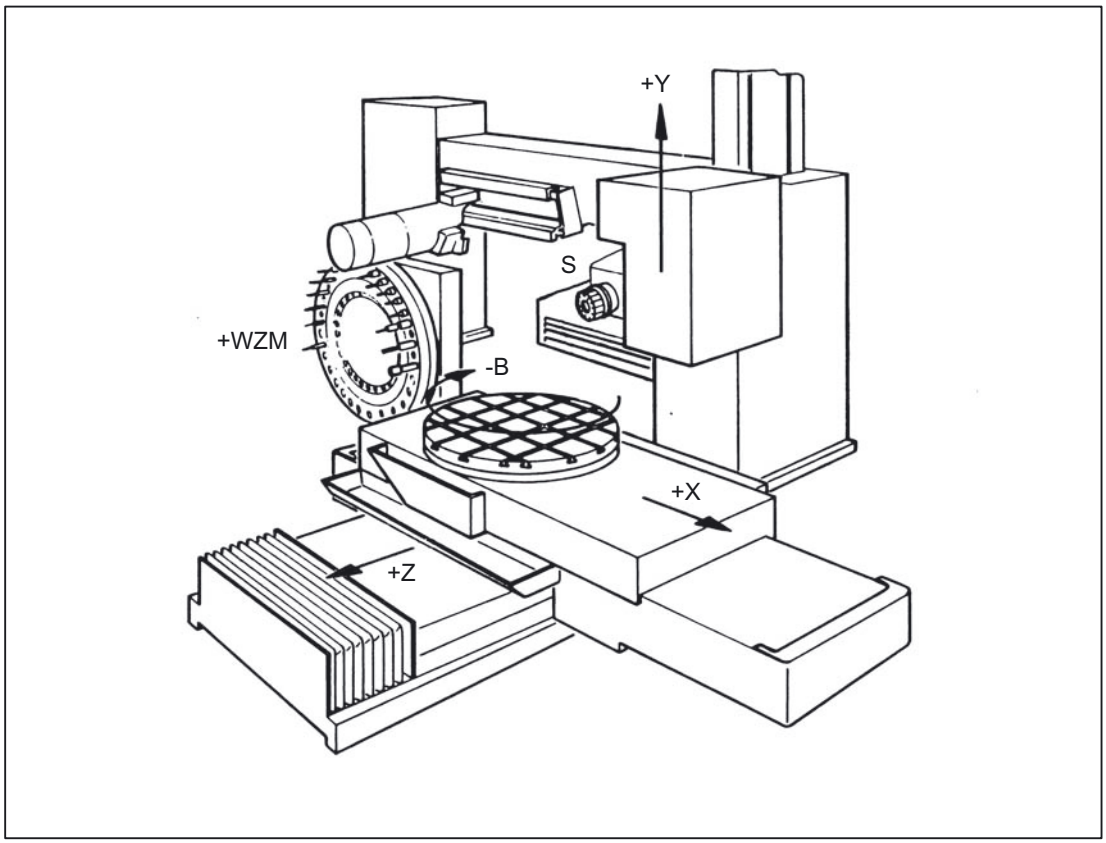
Examples

4

4.1 Axes

Axis configuration for a 3-axis 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

Machine data	Value
MD20070 AXCONF_MACHAX_USED[1]	= 2
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

4.2 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
. . .
```

4.3 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 5-axis 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)
```

Data Lists

5

5.1 Machine data

5.1.1 Memory specific machine data

Number		Identifier: $MM_	Description
HMI Advanced	HMI Embedded		
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	9247	USER_CLASS_BASE_ZERO_OFF_PA	Availability of basic offset in "Parameters" operating area
9248	9248	USER_CLASS_BASE_ZERO_OFF_MA	Availability of basic offset in Machine operating area
	9249	USER_CLASS_VERT_MODE_SK	Provide protection for vertical area soft keys
	9400	TOOL_REF_GEO_AXIS1	Absolute dimension tool length compensation geometry axis 1
	9401	TOOL_REF_GEO_AXIS2	Absolute dimension tool length compensation geometry axis 2
	9402	TOOL_REF_GEO_AXIS3	Absolute dimension tool length compensation geometry axis 3
9424	9424	MA_COORDINATE_SYSTEM	Coordinate system for actualvalue display
	9425	MA_SCRATCH_DEFAULT_MODE	Tool offset computation for geometry axes with scratching
9440	9440	ACTIVE_SEL_USER_DATA	Active data (frames) are immediately operative after editing

Number		Identifier: $MM_	Description
9449		WRITE_TOA_LIMIT_MASK	Applicability of MD9203 to edge data and locationdependent offsets
9450	9450	MM_WRITE_TOA_FINE_LIMIT	Limit value for wear fine
9451	9451	MM_WRITE_ZOA_FINE_LIMIT	Limit value for offset fine
	9459	PA_ZOA_MODE	Display mode of zero offset

5.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB	Machine axis name
10600	FRAME_ANGLE_INPUT_MODE	Input type for rotation with frame
10602	FRAME_GEOAX_CHANGE_MODE	Frames and switchover of geometry axes
10610	MIRROR_REF_AX	Reference axis for FRAME element mirroring
10612	MIRROR_TOGGLE	Toggle mirroring
10613	NCBFRAME_RESET_MASK	RESET response of global basic frame
10615	NCBFRAME_POWERON_MASK	POWER ON response of global basic frames
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 allowed
18600	MM_FRAME_FINE_TRANS	Fine offset for all settable FRAMES and the basic frame
18601	MM_NUM_GLOBAL_USER_FRAMES	Number of globally predefined user frames
18602	MM_NUM_GLOBAL_BASE_FRAMES	Number of global basic frames

5.1.3 Channel-specific machine data

Number	Identifier: $MC_	Description
21015	INVOLUTE_RADIUS_DELTA	NC start disable without reference point
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/special axis name in channel
22532	GEOAX_CHANGE_M_CODE	M code for replacement of geometry axes
22534	TRAFO_CHANGE_M_CODE	M code for transformation changeover

Number	Identifier: $MC_	Description
24000	FRAME_ADD_COMPONENTS	Separate programming/modification of additively programmable frame components
24002	CHBFRAME_RESET_MASK	RESET response of channelspecific basic frames
24004	CHBFRAME_POWERON_MASK	POWER ON response of channelspecific basic frames
24006	CHSFRAME_RESET_MASK	RESET response of channelspecific system frames
24007	CHSFRAME_RESET_CLEAR_MASK	Clear system frames on RESET
24008	CHSFRAME_POWERON_MASK	Delete system frames on 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	Save and activate 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
28081	MM_NUM_BASE_FRAMES	Number of channel/specific Basic frames
28082	MM_SYSTEM_FRAME_FRAMES	Configuration screen form for channel-specific system frames
28560	MM_SEARCH_RUN_RESTORE_MODE	Restore data after a simulation (SW 7.1 and higher)

5.1.4 Axis/spindle-specific 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

5.2 Setting data

5.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42440	FRAME_OFFSET_INCR_PROG	Zero offsets are traversed on incremental programming
42980	TOFRAME_MODE	Determination of the direction of X and Y axes for frame definition

5.3 System variables

Names	Description
$AC_DRF[axis]	DRF offset (differential resolver function)
$AA_ETRANS[axis]	Offset value zero offset external
$AA_OFF[axis]	Overlaid motion for programmed axis
$P_UIFR[n]	Settable frames, activated via G500, G54 to G599
$P_CHBFR[n]	Channel basic frames, activated via G500, G54 to G599
$P_SETFR	System frame for PRESET and scratching in data management
$P_EXTFR	System frame for zero offset external in data management
$P_PARTFR	System frame for TCARR and PAROT in data management
$P_TOOLFR	System frame for TOROT and TOFRAME in data management
$P_WPFR	System frame for workpiece reference points
$P_CYCFR	System frame for cycles
$P_TRAFRAME	System frame for transformations
$P_NCBFR[n]	NCU basic frames, activated via G500, G54 to G599
$P_UBFR	Basic frame in channel activated after G500, G54 to G599 Corresponds to $P_CHBFR[0].
$P_SETFRAME	Current system frame for PRESET and scratching
$P_EXTFRAME	Current system frame for zero offset external
$P_PARTFRAME	Current system frame for TCARR and PAROT with an orientational toolholder
$P_TOOLFRAME	Current system frame for TOROT and TOFRAME
$P_WPFRAME	Current system frame for workpiece reference points
$P_CYCFRAME	Current system frame for cycles
$P_TRAFRAME	Current system frame for transformations

Names	Description
$P_CHBFRAME[n]	Current basic frame in channel, 0 to 15 NCU basic frames can be configured via machine data MD28081 MM_NUM_BASE_FRAMES.
$P_NCBFRAME[n]	Current NCU basic frame, 0 to 15 NCU basic frames can be configured via machine data MD18602 MM_NUM_GLOBAL_BASE_FRAMES.
$P_ACTBFRAME	Current chained total basic frame
$P_BFRAME	Current first basic frame in the channel. Corresponds to $P_CHBFRAME
$P_IFRAME	Current settable frame
$P_PFRAME	Current programmable frame
$P_ACTFRAME	Current total frame
$P_UIFRNUM	Number of active frame $P_UIFR
$P_NCBFRMASK	Bit mask for definition of NCU global basic frames
$P_CHBFRMASK	Bit mask for definition of channelspecific basic frames
$P_CHSFRMASK	Bit mask for activating system frames in data management

$AA_ETRANS[X]

$AA_ETRANS[X] is an axisspecific system variable of the DOUBLE type. The default setting in the system for this variable is zero. Values set by the user are activated by IS "Zero offset external" (DB31-61, ... DBX3.0).

5.4 Signals

5.4.1 Signals from channel

DB number	Byte.Bit	Description
21, ...	61.0	T function modification
21, ...	62.0	D function modification
21, ...	118-119	T function
21, ...	129	D function
21, ...	208	Number of active function G group 1
21, ...	209	Number of active function G group 2
...	...	...
21, ...	230	Number of active function G group 29

5.4.2 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	3.0	Zero offset external
31, ...	60.0	Spindle/no axis

Index

A

Actual-value system
    Workpiece-related, 2-114
ATRANS, 1-5
Axis configuration, 2-13

B

Basic coordinate system (BCS), 1-3, 2-26

C

CFINE, 1-5
Channel axes, 2-4
Configurable SZS, 2-34
CTRANS, 1-5

F

FGROUP, 2-9, 2-12
Fine offset, 1-5
FRAME, 2-4

G

G58, 1-5
G59, 1-5
Geometry axes, 2-4, 2-9, 2-26

H

Helical interpolation, 2-13
Helix interpolation, 2-13

K

Kinematic transformation, 2-26

L

Loader axes, 2-9

M

Machine axes, 2-2
Machine coordinate system (MCS), 1-3, 2-24
Machine tool axes, 2-9
Machine zero M, 2-20
Main axes, 2-11
Manual traverse in SZS, 2-36
MD10000, 1-2, 2-14
MD10002, 2-15
MD10600, 2-42, 2-100, 2-101
MD10602, 2-68, 2-69, 2-71, 2-79, 2-84
MD10610, 1-7, 2-63
MD10612, 2-63
MD10613, 2-112
MD10615, 2-110
MD10617, 2-108
MD11640, 2-15
MD18600, 2-40, 2-41
MD18601, 2-52, 2-58, 2-109
MD18602, 2-52, 2-60, 2-109
MD20050, 1-2, 2-17, 2-90
MD20060, 1-2
MD20070, 1-2, 2-15, 2-17
MD20080, 1-2, 2-12, 2-15
MD20110, 2-6, 2-110, 2-111, 2-112, 2-113, 2-118
MD20112, 2-6, 2-113, 2-118
MD20118, 2-6
MD20150, 2-51, 2-112, 2-113
MD20152, 2-112
MD20184, 2-96, 2-98, 2-105, 2-106
MD21110, 2-103
MD22532, 2-7
MD24002, 2-112
MD24004, 2-110
MD24006, 2-30, 2-51, 2-111, 2-112
MD24007, 2-112
MD24008, 2-30, 2-110
MD24020, 2-57

MD24030, 2-35, 2-55
MD24040, 2-90
MD24050, 2-51
MD24110, 2-17
MD24120, 2-17
MD24805, 2-79
MD24855, 2-79
MD24905, 2-71
MD28080, 2-52, 2-58
MD28081, 2-52
MD28082, 2-29, 2-51, 2-65, 2-66, 2-67, 2-95, 2-96, 2-98, 2-102, 2-106, 2-108
MD28560, 2-113
MD32074, 2-110
MD35000, 1-2, 2-47
MD42440, 2-96
MD9440, 2-51, 2-117

P

Path axes, 2-9
POS, 2-9, 2-10
POSA, 2-9, 2-10
Position of coordinate systems and reference points, 2-21
Positioning axes, 2-10

R

Reference point R, 2-20
Reference points in working space, 2-20
Replaceable geometry axes, 2-4
Rotary axes, 2-9
Rough offset, 1-5

S

SD42980, 2-103, 2-104, 2-105
Special axes, 2-9
Synchronized axes, 2-12

T

TCP Tool Center Position, 2-20
Tool revolver axes, 2-9
Toolholder reference point T, 2-20
TRANS, 1-5
Transformation, 2-26
TRANSMIT, 2-26

W

Workpiece coordinate system (WCS), 1-3, 1-4, 2-37
Workpiece zero W, 2-20

Z

Zeros and reference points, 2-20

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Emergency Stop (N2)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 Relevant standards 2-1

2.2 Emergency stop control elements 2-2

2.3 Emergency stop sequence 2-3

2.4 Emergency stop acknowledgement 2-5

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Examples 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 Drive-specific machine data 5-1
5.1.2 Axis/spindlespecific machine data 5-1

5.2 Signals 5-1
5.2.1 Signals to NC 5-1
5.2.2 Signals from NC 5-2
5.2.3 Signals to BAG 5-2

Index Index-1

Brief Description

1

1.1 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.

Detailed Description 2

2.1 Relevant standards

Relevant standards

Compliance with the following standards is essential for the emergency stop function:

- EN 292 Part 1
- EN 292 Part 2
- EN 418
- EN 60204 Part 1:1992 Section 10.7

VDE 0113 Part 1 only applies for a transitional period and will be replaced by 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 human action in cases where the normal stop function is not suitable.

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.

Standard EN 2922

In accordance with the basic safety requirements of the EC Machinery Directive regarding emergency stop, as stipulated in Subsection 6.1.1 of EN 2922, machines must be equipped with an emergency 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 acknowledgment) rests exclusively with the machine manufacturer.

2.2 Emergency stop control elements

EN standard

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:

- Mushroomhead pushbutton switches
- Wires/cables, cords, rods
- Puller grips
- In special cases: Footoperated 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 to 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 to interface signal DB10, DBX56.2 (acknowledge emergency stop).

Connection conditions

See the hardware configuration guide (Operator Components Manual) for information on connecting the emergency stop button.
References:
/BH/ Operator Components Manual

2.3 Emergency stop sequence

EN 418 standard

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 is selected and the correct stop category (defined in EN 60204) can be determined according to a risk assessment.

Emergency stop sequence

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 within their respectively configured, axis-specific times: MD36610 $MA_AX_EMERGENCY_STOP_TIME (braking ramp time for errors).

   The maximum brake ramp that can be achieved is defined by the maximum brake current of the respective drive. The maximum brake current is achieved by setting a setpoint = 0 (fast braking).
2. The interface signal DB11, ... DBX6.3 (BAG ready for operation) is cleared.
3. The interface signal DB10, DBX106.1 (emergency stop active) is set.
4. Alarm: "3000 Emergency Stop" is displayed.

5. After elapse of a time configurable in machine data:
   MD36620 $MA_SERVO_DISABLE_DELAY_TIME (cutout delay servo enable),
   the servo enables of the machine axes are cancelled.

   Please note the configuration rule below:

   MD36620 $MA_SERVO_DISABLE_DELAY_TIME ≥
   MD36610 $MA_AX_EMERGENCY_STOP_TIME
6. All machine axes are switched to follow-up mode within the control.
   The machine axes are therefore no longer in position control.

Sequence on 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 sequence of operations in the NC is started with interface signal
  DB10 DBX 56.1 (emergency stop).

  After the machine axes are at standstill, the energy supply must be interrupted in accordance 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 the 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 to DBB7.

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 channelspecific emergency stop states) can therefore be assumed.

2.4 Emergency stop acknowledgement

EN 418 standard

The emergency stop control element may only be reset as a result of manual manipulation of the emergency stop control element. Resetting of the emergency stop control element alone must not trigger a restart command.
A machine restart must be impossible until each of the actuated emergency stop control elements has been deliberately reset by hand.

Emergency stop acknowledgement

1. Interface signal DB10, DBX56.2 (acknowledge emergency stop) = 1
2. Interface signal DB11, ... DBX0.7 (Mode Group reset) = 1

Note

Both interface signals:
DB10, DBX56.2 (acknowledge emergency stop)
and
DB21, ... DBX7.7 (Reset)
must remain set together until the interface signal:
DB10, DBX106.1(emergency stop active)
has been reset.

The emergency stop state cannot be reset with the interface signal DB21, ... DBX7.7 (Reset) alone.

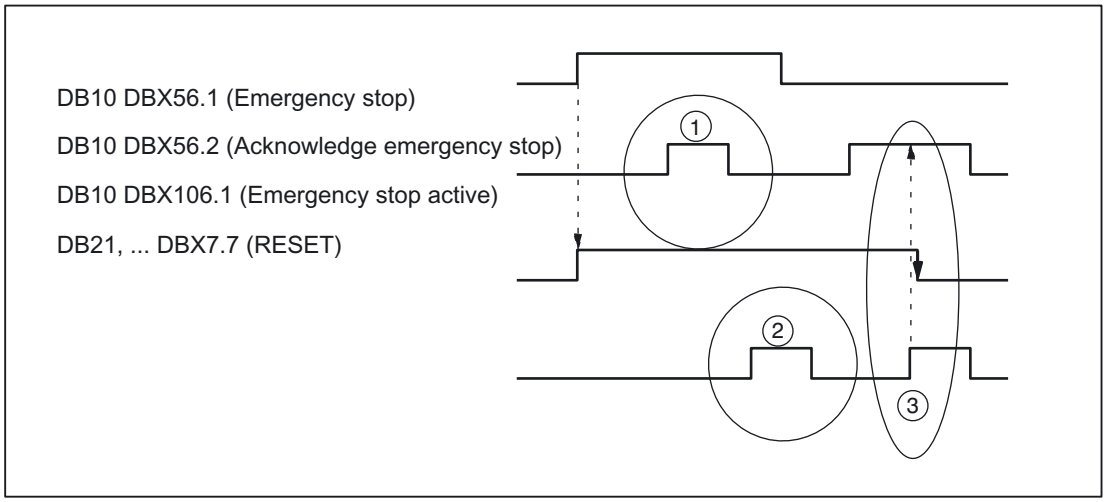


Fig. 2-1 Resetting the emergency stop state

(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)

Effects

Resetting the emergency stop state has the following effects:

- Within the control, the servo enable is set, the follow-up mode deactivated and the position control activated for all machine axes.
- Interface signals DB31, ... DBB60.5 (position control active) are set.
- Interface signals DB11, ... DBX6.3 (Mode Group ready for operation) are set.
- Interface signal DB10, DBX106.1 (emergency stop active) is reset.
- Alarm: "3000 Emergency Stop" is cleared.
- 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).

This will work only if interface signal:
DB10, DBX56.1 (emergency stop)
is not set while the controller is powering up.

Supplementary Conditions

3

3.1 Supplementary conditions

No supplementary conditions apply.

4 Examples

4.1 Examples

No examples are available.

Data Lists

5

5.1 Machine data

5.1.1 Drive-specific machine data

Number	Identifier: $MD_	Description
1404	PULSE_SUPPRESSION_DELAY	Time for pulse suppression

5.1.2 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

5.2 Signals

5.2.1 Signals to NC

DB number	Byte.Bit	Description
10	56.1	EMERGENCY STOP
10	56.2	Acknowledge EMERGENCY STOP

5.2.2 Signals from NC

DB number	Byte.Bit	Description
10	106.1	EMERGENCY STOP active

5.2.3 Signals to BAG

DB number	Byte.Bit	Description
11, ...	0.7	Mode group reset

Index

E

EMERGENCY STOP
- Acknowledgment, 2-5
- Interface, 2-2
- Sequence, 2-3

Emergency stop key, 2-2

M

MD36620, 2-4

R

Reset, 2-5

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Transverse Axes (P1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 Defining a geometry axis as transverse axis 2-1

2.2 Dimensional information for transverse axes 2-2

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Examples 4-1

5 Data Lists 5-1

5.1 Machine data 5-1

5.1.1 Channelspecific machine data 5-1

Index Index-1

Brief Description

1

1.1 Brief description

Within the framework of "milling" 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.

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.

Every geometry axis of a channel can be defined as a transverse axis. However, only exactly one transverse axis can be defined per channel.

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.

Detailed Description

2

2.1 Defining a geometry axis as transverse axis

Transverse axis

Within the framework of "milling" 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.

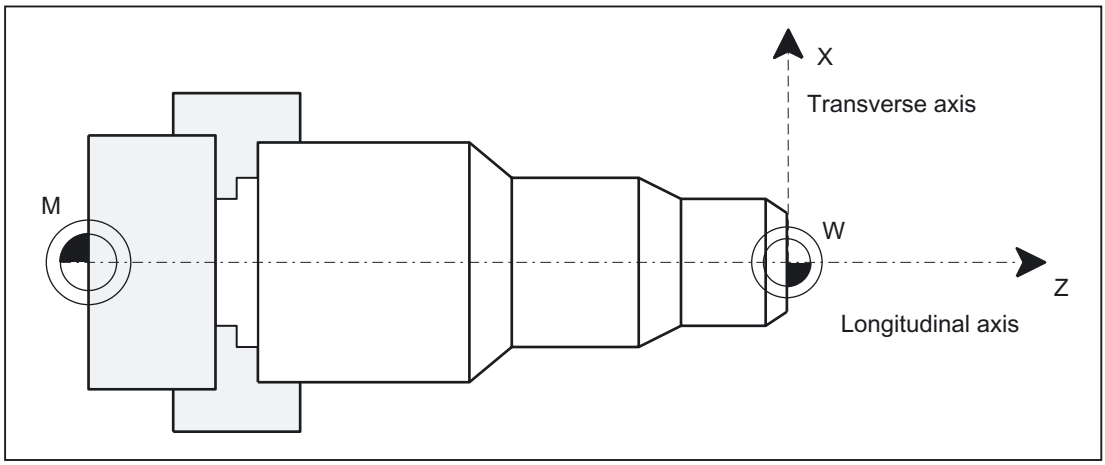


Fig. 2-1 Position of the transverse axis in the machine coordinate system

Definition

A geometry axis is defined as a transverse axis with:
MD20100 $MC_DIAMETER_AX_DEF (geometry axis as transverse axis function)

Restrictions

However, only exactly one geometry axis can be defined as transverse axis per channel.

2.2 Dimensional information for transverse axes

Transverse axes can be programed 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.

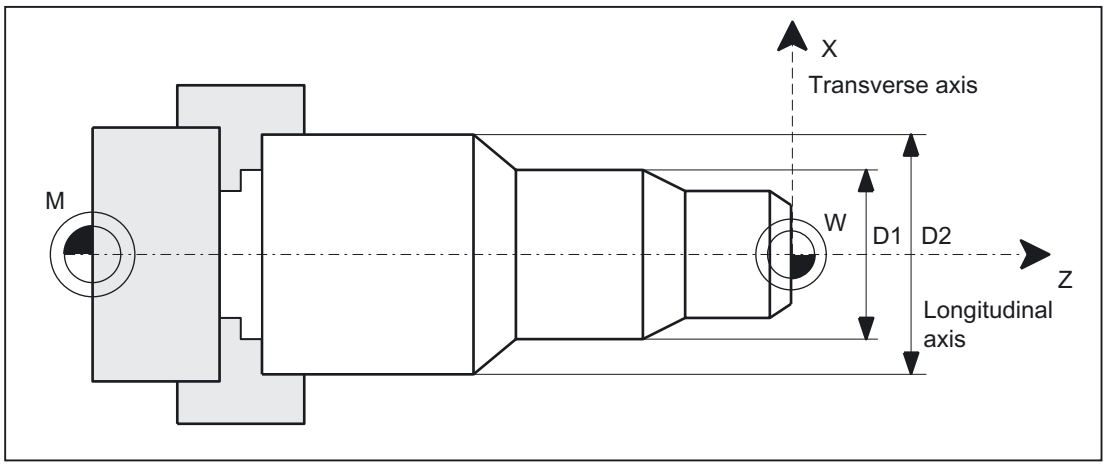


Fig. 2-2 Transverse axis with diameter information (D1, D2)

Switching the diameter programming on/off

The diameter programming is switched on/off with the part program instructions:

- `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`

Reference is made exclusively to the transverse axis of the channel.

`DIAMON`, `DIAMOF` and `DIAM90` belong to G group 29 and are modally operative.

Initial setting

The initial setting is configured with:
MD20150 $MN_GCODE_RESET_VALUES [28] (initial setting of the G groups)

Diameter-related data

After activation of the diameter programming, the following data refer to diameter dimensions:

DIAMON

- 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 travel with handwheel
- 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:
  - $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

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:
  - $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*]

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

Supplementary Conditions

3

3.1 Supplementary conditions

There are no supplementary conditions to note.

4 Examples

4.1 Examples

No examples are available.

Data Lists

5

5.1 Machine data

5.1.1 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
20150	GCODE_RESET_VALUES[n]	Reset G groups

Index

D

DIAM90, 2-2
DIAMOF, 2-2
DIAMON, 2-2

M

MD20100, 2-1
MD20150, 2-2

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D/840Di/810D

Power Line Basic PLC Program (P3 PI)

Function Manual

Valid for

Control
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840D powerline/840DE powerline

Software	*Version*
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety information

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified personnel

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

Prescribed usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 Key PLC CPU data for 810D, 840D and 840Di 2-1

2.2 Reserve resources (timers, FC, FB, DB, I/O) 2-8

2.3 Starting up hardware configuration of PLC CPUs 2-9

2.4 Starting up the PLC program 2-14
2.4.1 Installing the basic program for 810D, 840D 2-14
2.4.2 Application of basic program 2-16
2.4.3 Version codes 2-17
2.4.4 Machine program 2-18
2.4.5 Data backup 2-18
2.4.6 PLC series startup, PLC archives: 2-19
2.4.7 Software upgrades 2-21
2.4.8 I/O modules (FM, CP modules) 2-22
2.4.9 Troubleshooting 2-22

2.5 Linking PLC CPUs to 810D, 840D 2-23
2.5.1 Properties of PLC CPUs 2-24
2.5.2 Interface on 810D and 840D with integrated PLC 2-24
2.5.3 Diagnostic buffer on PLC 2-27

2.6 Interface structure 2-29
2.6.1 PLC/NCK interface 2-29
2.6.2 PLC/MMC interface 2-36
2.6.3 PLC/MCP/HHU interface 2-39

2.7 Structure and functions of the basic program 2-43
2.7.1 Startup and synchronization of NCK PLC 2-45
2.7.2 Cyclic operation (OB1) 2-45
2.7.3 Time-alarm processing (OB 35) 2-48
2.7.4 Process-alarm processing (OB 40) 2-48
2.7.5 Response to NC failure 2-48
2.7.6 Functions of the basic program called from the user program 2-50
2.7.7 Symbolic programming of user program with interface DB 2-52
2.7.8 M decoding acc. to list 2-54
2.7.9 PLC machine data 2-58
2.7.10 Configuration of machine control panel, handheld unit 2-62

2.8 SPL for Safety Integrated 2-71

2.9 Assignment overview 2-71
2.9.1 Assignment: NC/PLC interface 2-71
2.9.2 Assignment: FB/FC 2-71
2.9.3 Assignment: DB 2-72
2.9.4 Assignment: Timers 2-74

2.10 Memory requirements of basic PLC program for 810D, 840D ... 2-74

2.11 Supplementary conditions and NC VAR selector ... 2-77
2.11.1 Supplementary conditions ... 2-77
2.11.1.1 Programming and parameterizing tools ... 2-77
2.11.1.2 SIMATIC documentation required ... 2-79
2.11.1.3 Relevant SINUMERIK documents ... 2-80
2.11.2 NC VAR selector ... 2-80
2.11.2.1 Overview ... 2-80
2.11.2.2 Description of Functions ... 2-83
2.11.2.3 Startup, installation ... 2-91

2.12 Block descriptions ... 2-92
2.12.1 FB 1: RUN_UP Basic program, startup section ... 2-92
2.12.2 FB 2: Read GET NC variable ... 2-106
2.12.3 FB 3: PUT write NC variables ... 2-113
2.12.4 FB 4: PI_SERV General PI services ... 2-120
2.12.5 FB 5: GETGUD read GUD variable ... 2-143
2.12.6 FB 7: PI_SERV2 General PI services ... 2-148
2.12.7 FB 9: M : N operating-unit switchover ... 2-151
2.12.8 FB 10: Safety relay (SI relay) ... 2-157
2.12.9 FB 11: Brake test ... 2-160
2.12.10 FB 29: Signal recorder and data trigger diagnostics ... 2-165
2.12.11 FC 2: GP_HP Basic program, cyclical section ... 2-169
2.12.12 FC 3: GP_PRAL Basic program, interruptcontrolled section ... 2-170
2.12.13 FC 5: GP_DIAG Basic program, diagnostic alarm, and module failure ... 2-173
2.12.14 FC 7: TM_REV Transfer block for tool change with revolver ... 2-175
2.12.15 FC 8: TM_TRANS transfer block for tool management ... 2-179
2.12.16 FC 9: ASUB startup of asynchronous subprograms ... 2-186
2.12.17 FC 10: AL_MSG error and operating messages ... 2-189
2.12.18 FC 12: AUXFU call interface for user with auxiliary functions ... 2-191
2.12.19 FC 13: BHGDisp display control for handheld unit ... 2-192
2.12.20 FC 15: POS_AX positioning of linear and rotary axes ... 2-195
2.12.21 FC 16: PART_AX positioning of indexing axes ... 2-201
2.12.22 FC 17: YDelta star/delta changeover ... 2-205
2.12.23 FC 18: SpinCtrl spindle control ... 2-208
2.12.24 FC 19: MCP_IFM transmission of MCP signals to interface ... 2-218
2.12.25 FC 21: transfer PLC NCK data exchange ... 2-225
2.12.26 FC 22: TM_DIR Direction selection for tool management ... 2-232
2.12.27 FC 24: MCP_IFM2 Transmission of MCP signals to interface ... 2-235
2.12.28 FC 25: MCP_IFT transfer of MCP/OP signals to interface ... 2-239
2.12.29 FC 26: HPU_MCP Transfer of HPU/HT6 signals to the interface ... 2-243
2.12.29.1 General ... 2-243
2.12.29.2 MCP selection signals to the user interface ... 2-245
2.12.29.3 Checkback signals from user interface for controlling displays ... 2-247
2.12.30 FC 19, FC 24, FC 25, FC 26 source code description ... 2-248

2.13 Signal/data descriptions ... 2-250
2.13.1 Interface signals NCK/PLC, MMC/PLC, MCP/PLC ... 2-250
2.13.2 Decoded M signals ... 2-251
2.13.3 G Functions ... 2-252
2.13.4 Message signals in DB 2 ... 2-254

2.14 Useful Tips on Programming with STEP 7 ... 2-257
2.14.1 General ... 2-257
2.14.2 Copying data ... 2-257
2.14.3 ANY and POINTER ... 2-258
2.14.3.1 General ... 2-258
2.14.3.2 Use of POINTER and ANY in FC if POINTER or ANY is available as parameter ... 2-258

2.14.3.3 Use of POINTER and ANY in FB if POINTER or ANY is available as parameter 2-259
2.14.3.4 POINTER or ANY variable for transfer to FC or FB 2-260
2.14.4 Multiinstance DB 2-262
2.14.5 Strings 2-264
2.14.6 Determining offset addresses for data block structures 2-265

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Examples 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Channelspecific machine data 5-1

Index Index-1

Brief Description

1

1.1 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), MMC (Man Machine Communication) and MCP (Machine Control Panel) areas. A distinction is made between the following groups for signals and data:

- Cyclic signal exchange
- Eventdriven signal exchange
- Messages

Cyclic signal exchange

The cyclically-exchanged signals consist primarily of bit arrays.

- They contain **commands** transmitted 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 groups
  - 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 that the signals from the NCK remain constant throughout the 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 the auxiliary functions are used to start execution of a block, the type of auxiliary function determines whether the NCK has to wait before executing the function (e.g. during a tool change) or whether the function is executed in parallel to machining of the workpiece (e.g. for tool preparation on milling machines with chaintype magazines).

Data transfer must be as fast and yet as reliable as possible, in order to minimize the effect on the NC machining process. Data transfer is therefore controlled by alarms and acknowledgments. 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 do not require user acknowledgment, this does not affect NC processing.

Event-driven signal exchange PLC → NCK

An "eventdriven signal exchange PLC → NCK" takes place whenever the PLC passes a request to the NCK (e.g., traversal of an auxiliary axis). In this case, the data transfer is also controlled by acknowledgment. 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. A defined bit field is used to transfer the message signals to the basic program. The signals are evaluated there and entered in the PLC diagnostics buffer on the occurrence of the message events. If an MMC is present, the messages are displayed on the operator interface.

Note

The function of the machine is largely determined by the PLC program. Every PLC program in the working memory can be edited with the programming device.

Detailed Description

2

2.1 Key PLC CPU data for 810D, 840D and 840Di

The tables below show the performance range of the PLC CPUs and the scope of the basic PLC program relative to the various controller types.

Type of control: 810D and 840D

Key CPU data

	810D / 840D	810D / 840D	810D / 840D
PLC CPU	Integrated PLC CPU314	Integrated PLC CPU315-2DP	Integrated PLC CPU315-2DP master/slave
MLFB		6ES7 315-2AF00-0AB0	6ES7 315-2AF01-0AB0
Memory for user and basic program	64, 96, 128 KB	64, 96 128, 160, 192, 224, 256, 288 KB (dependent on option)	96, 160, 224, 288, 352, 416, 480 KB (dependent on option)
Data block memory	Like user memory	Like user memory	Up to 96 KB
Memory submodule	no	no	no
Bit memories	2048	2048	2048/ 4096 with PLC operating system 03.10.13 or later
Timers	128	128	128
Counters	64	64	64
Clock memories	8	8	8
Program/data blocks OB FB FC DB	 1, 10, 20, 35, 40, 80-82, 85, 87, 100, 121-122 1-127 1-127 1-127	 1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122 1-127 1-127 1-127	 1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122 0-255 0-255 1-399
Max. data block length	16 KB	16 KB	16 KB
Max. block length FC, FB	24 KB	24 KB	24 KB
Inputs/outputs (address capacity) - digital - analog	 768 64	 1024/1024 64	 1024/1024 64

	810D / 840D	810D / 840D	810D / 840D
Inputs/outputs 1) (addressing)	Subrack 0 is not available for I/O devices:	Through optional configuring of I/O devices:	Through optional configuring of I/O devices:
- digital - analog	from I/O byte 32 onwards from PI/PO byte 384 onwards	from I/O byte 0 onwards from PI/PO byte 272 onwards	from I/O byte 0 onwards from PI/PO byte 272 onwards
Processing time - Bit commands (I/O) - Word commands	 0.3 ms/kA 1-4 ms/kA	 0.3 ms/kA 1-4 ms/kA	 0.3 ms/kA 1-4 ms/kA
PDIAG (Alarm S,SQ)	no	no	yes
PROFIBUS	N/A	Master	Master/Slave
Number of PROFIBUS slaves		Min. 16, max. 64 SDB 2000 ≤ 8 KB	Min. 16, max. 64 SDB 2000 ≤ 32 KB
PBC (programmable block communication)	no	no	yes
Consistent data to standard slave via SFC 14, 15	N/A	26	26
1) Subrack 0 is integrated in the NC. Subracks 1 to 3 are available for I/O devices.			

I/O expansion

	810D / 840D	810D / 840D	810D / 840D
PLC CPU	Integrated PLC CPU314	Integrated PLC CPU315-2DP	Integrated PLC CPU315-2DP master/slave
MLFB		6ES7 315-2AF00-0AB0	6ES7 315-2AF01-0AB0
I/O modules	24	24	24
PROFIBUS DP modules	N/A	yes	yes
Interfaces (MPI)	1	1	1

Types of control: 840Di, 810D and 840D

Key CPU data

	840Di	810D	840D
PLC CPU	Integrated PLC 315-2DP master/slave	Integrated PLC 315-2DP master/slave	Integrated PLC 314C-2DP master/slave
MLFB	6ES7 315-2AF03-0AB0	6ES7 315-2AF03-0AB0	6FC5 314-6CF00-0AB0
Memory for user and basic program	64, 96, 128, 160, 192, 224, 256 KB	64, 96, 128, 160, 192, 224, 288 KB	96, 160, 224, 352, 416, 480 KB (dependent on option)
Data block memory	Like user memory	Like user memory	Up to 96 KB
Memory submodule	no	no	no
Bit memories	4096	4096	4096
Timers	128	128	256
Counters	64	64	256
Clock memories	8	8	8

	840Di	810D	840D
Program/data blocks			
OB	1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122	1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122	1, 10, 20, 35, 40, 80-82, 85-87,100, 121-122
FB	0-255	0-255	0-255
FC	0-255	0-255	0-255
DB	1-399	1-399	1-399
Max. length of data block	16 KB	16 KB	16 KB
Max. block length FC, FB	24 KB	24 KB	24 KB
Inputs/outputs (address capacity)			
- digital	1024/1024	1024/1024	1024/1024
- analog	64	64	64
Inputs/outputs 1) (addressing)	Through optional configuring of I/O devices:	Through optional configuring of I/O devices:	Through optional configuring of I/O devices:
- digital	from I/O byte 0 onwards	from I/O byte 0 onwards	from I/O byte 0 onwards
- analog	from PI/PO byte 272 onwards (PROFIBUS only)	from PI/PO byte 272 onwards	from PI/PO byte 272 onwards
Processing time			
- Bit commands (I/O)	0.3 ms/kA	0.3 ms/kA	0.1 ms/kA
- Word commands	1-4 ms/kA	1-4 ms/kA	0.25-1.2 ms/kA
PDIAG (Alarm S,SQ)	Yes	Yes	Yes
PROFIBUS	Master	Master/Slave	Master/Slave
Number of PROFIBUS slaves	Max. 64 SDB 2000 ≤ 32 KB	Max. 64 SDB 2000 ≤ 32 KB	Max. 32 SDB 2000 ≤ 32 KB
Max. number of PROFIBUS slots	256	256	256
PBC (programmable block communication)	Yes	Yes	Yes
Consistent data to standard slave via SFC 14, 15	26	26	32

1) Subrack 0 is integrated in the NC. Subracks 1 to 3 are available for I/O devices.

I/O expansion

	840Di	810D	840D
PLC CPU	Integrated PLC 315-2DP master/slave	Integrated PLC 315-2DP master/slave	Integrated PLC 314C-2DP master/slave
MLFB	6ES7 315-2AF03-0AB0	6ES7 315-2AF03-0AB0	6FC5 314-6CF00-0AB0
I/O modules	PROFIBUS only	24	24
PROFIBUS DP modules	Yes	Yes	Yes
Interfaces (MPI)	1	1	1

Types of control: 840Di and 840D

Key CPU data

	840Di	840D
PLC CPU	Integrated PLC 317-2DP master/slave	Integrated PLC 317-2DP master/slave
MLFB	6FC5 317-2AJ10-0AB0	6FC5 317-2AJ10-1AB0
Memory for user and basic program	128 to 768 KB	128 to 768 KB
Data block memory	Max. 256 KB	Max. 256 KB
Memory submodule	no	no
Bit memories	32768	32768
Timers	512	512
Counters	512	512
Clock memories	8	8
Program/data blocks: OB FB FC DB	 10, 20-21, 32-35, 40, 55-57, 80, 82, 85-87, 100, 121-122 0-2048 0-2048 1-2048	 10, 20-21, 32-35, 40, 55-57, 80, 82, 85-87,100, 121-122 0-2048 0-2048 1-2048
Max. length of data block	32 KB	32 KB
Max. block length FC, FB	64 KB	64 KB
Inputs/outputs 1) (address capacity in bytes): - digital / - analog - incl. reserved area - process image	 4096/4096 8192/8192 256/256	 4096/4096 8192/8192 256/256
Inputs/outputs 2) (addressing): - digital - analog	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 272 onwards (PROFIBUS only)	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 272 onwards
Processing time: - Bit commands (I/O) - Word commands	 ≤ 0.031 ms/kA 0.1 ms/kA	 ≤ 0.103 ms/kA 0.1 ms/kA
PDIAG (Alarm S,SQ)	Yes	Yes
PROFIBUS	Master/Slave	Master/Slave
No. of PROFIBUS slaves	max. 125	max. 125
Max. number of PROFIBUS slots	512	512
DP master system: DP	1	1
DP master system: MPI/DP	2	N/A

	840Di	840D
PBC (programmable block communication)	Yes	Yes
Consistent data to standard slave via SFC 14, 15	128	128
1) Notice: The inputs/outputs above 4096 are reserved for integrated drives.		
2) Subrack 0 is integrated in the NC. Subracks 1 to 3 are available for I/O devices.		

I/O expansion

	840Di	840D
PLC CPU	Integrated PLC 317-2DP master/slave	Integrated PLC 317-2DP master/slave
MLFB	6FC5 317-2AJ10-0AB0	6FC5 317-2AJ10-1AB0
I/O modules	PROFIBUS only	24
PROFIBUS DP modules	1 (2)	1
Interfaces (MPI)	1 (0)	1

Note

Number of PROFIBUS slaves

Since the SDB 2000 and other SDBs must be stored by the PLC operating system in the static RAM, which can also be accessed by the PROFIBUS ASIC, SDBs smaller than 32 Kbytes may be rejected when the configuration is loaded. It cannot be determined whether the configuration is legal until the SDB container has been loaded to the CPU. As a result, the values specified in the above tables should only be seen as guide values.

If the configuration is illegal, a general reset request is issued when the SDBs are loaded. The cause of the configuring error can be found in the diagnostic buffer on completion of the general reset.

PLC versions

In SW 3.5 and higher on the 840D, version 6 (version code 35.06.03) is installed with PLC 314 and version 3 (version code 35.03.03) with PLC 315-2DP or higher.

These versions are compatible with the corresponding SIMATIC CPU300.
All modules and software packages approved by SIMATIC for these versions and CPUs are therefore suitable. Modules that can only be installed in subrack 0 are the exception (modules FM NC and FM 357 are also exceptions).

Version code: XX.YY.ZZ

- XX: SIMATIC CPU PLC version
- YY: Firmware transfer increment
- ZZ: Internal increment

Example

PLC 315-2DP with MLFB 6ES7 315-2AF00-0AB0:	04.02.14
PLC 315-2DP with MLFB 6ES7 315-2AF01-0AB0:	03.10.23
PLC 314:	07.02.12

HMI version display

The PLC module, the PLC operating system version and the module code appear in the last line of the HMI version display.

Example

PLC module	PLC operating system version	Module code
S7 PLC_315-2DP system	03.10.23	1200

Module codes

The table below shows the relationship between the module code and the corresponding PLC module, the suitable PLC operating system and its current software version:

Module code	PLC module	Suitable PLC operating systems (corresponding SIMATIC MLFB)	PLC operating system SW version
0208	PLC 314	6ES7 314-1AE0-0AB0	07.02.12
1008	PLC 3152DP with ASPC 2 Step C	6ES7 315-2AF00-0AB0	04.02.14
1100	PLC 3152DP with ASPC 2 Step D	6ES7 315-2AF01-0AB0	03.10.23
1200	PLC 3152DP with ASPC 2 Step E	6ES7 315-2AF01-0AB0	03.10.23
		6ES7 315–2AF03-0AB0 FW1.2	12.30.10
1400	PLC 314C2DP with IBC 16	6ES7 314-6CF00-0AB0 FW1.0.2	10.60.20
2200	PLC 317-2DP with IBC 32	6ES7 317-2AJ10-0AB0 FW2.1	20.71.15
MCI 1 (840Di)	PLC 3152DP with ASPC 2 Step E	6ES7 315-2AF03-0AB0 FW1.0	04.20.36
MCI 2 (840Di) 2100	PLC 317-2DP with IBC32	6ES7 317-2AJ10-0AB0 FW2.1	20.70.17

810 D, 840D

The tables below show the key data of the OPI interface and the PLC basic program functionality with reference to SINUMERIK 810D, 840D and 840Di:

OPI interface

	840Di	810D	840D
Number	N/A	N/A	1

PLC basic program functions

	840Di	810D	840D
Axes/spindles	1)	5	31
Channels	1)	2	10
Mode groups	1)	1	10
Status/control signals	+	+	+
M decoders (M00-99)	+	+	+
G group decoders	+	+	+
Aux. function distributors	+	+	+
Interrupt-driven output of auxiliary functions	+	+	+
Move axes/spindles from PLC	+	+	+
Async. subprogram interface	-	-	-
Error/operating messages	+	+	+
MCP and handheld unit signals via NCK	+	+	+
Reading/writing of NC variables	+	+	+
PI services	+	+	+
Tool management	+	+	+
Star/delta switchover	+	+	+
Display control handheld unit	+	+	+

1) Depends on chosen system software package

2.2 Reserve resources (timers, FC, FB, DB, I/O)

Reserved components

The components below are reserved for the basic program:

Component	Reserved range
Timers	T0 - T9
Functions (general)	FC 0 - FC 29
Functions (in ShopMill/ShopTurn)	FC 0 - FC 35
Function blocks	FB 0 - FB 29
Data blocks (general)[1)]	DB 1 - DB 62; DB 71 - DB 80
Data blocks (in ShopMill/ShopTurn) 1)	DB 1 - DB 62; DB 71 - DB 89
1) The data blocks for channels, axes/spindles and tool management functions that are not activated may be assigned as required by the user.	

PLC 317-2DP

PLC CPU: PLC 317-2DP are reserved for further number bands for SIEMENS applications referring to FC, FB, DB and I/O areas.

FC, FB and DB

Component	Reserved range
Functions	FC 1000 - FC 1023
Function blocks	FB 1000 - FB 1023
Data blocks	DB 1000 - DB 1099

I/O range

Component	Reserved range
Address area	256 - 271 [1)]
Inputs/outputs	4096 upwards[2)]
1) Reserved for the NC module and future expansions	
2) Reserved for integrated drives However, diagnostics addresses for modules can only be placed in the uppermost address range, as suggested by STEP7.	

2.3 Starting up hardware configuration of PLC CPUs

General procedure

STEP 7 is used to define the hardware configuration for a PLC CPU, including the associated I/O.

The procedure to be followed is shown below:

1. Load tool box to PG/PC
2. Create a new project (File, New, Project)
3. Insert, Hardware, SIMATIC 300 station
4. Select SIMATIC 300 station1 with mouse
5. Open object by right-clicking with the mouse to start the HWConfig
6. Destination system, load to PG, the hardware equipment complement is read back from the central system
7. Configure distributed I/Os
8. Insert PLC basic program

The addresses for the I/O modules can be changed if necessary (permissible only on certain PLC CPUs, e.g. PLC 3152DP).

As an alternative, the entire hardware configuration can be entered manually (see also appropriate STEP7 documentation). The notices below must be observed.

STEP7, Version 3
In STEP7 Version 3 and higher, the hardware configuration of the SINUMERIK components must be defined with the entries in SIMATIC\RACK 300. The install or setup program of the basic program on the tool box diskettes is required for this purpose.

STEP7 Version 5.1 SP2 and Toolbox 6.03.02
In STEP7 V5.1 SP2 and Toolbox 6.03.02 and higher, the SINUMERIK components are stored under SIMATIC 300\SINUMERIK. The current hardware expansion for STEP 7 can also be found under eSupport.
Current path (02/13/2004): sinumerik_software > 840d/810d/fm-nc > patches & fixes > plc > Hardware_fuer_STEP7 > Hardware up to NCU*.5/CCU3/840Di_MCI2 > V6.5.2.0

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
CCU1 810D CPU	6FC5 410-0AA00-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
CCU2 810D CPU	6FC5 410-0AA01-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
CCU1 810DE CPU	6FC5 410-0AY01-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
CCU2 810D CPU	6FC5 410-0AX02-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 810DE Light CCU1 module with system software (export)	6FC5 410-0AY00-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 810D CCU2 module with system software (standard)	6FC5 410-0AX02-1AA0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
SINUMERIK 840DE NCU 561.2 without system sw	6FC5 356-0BB11-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 571 (export version)	6FC5 357-0BA10-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 571 (export version) with PROFIBUS DP	6FC5 357-0BA11-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 571.2 (export version) with PROFIBUS DP	6FC5 357-0BA11-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840DE NCU 571.2 without system software	6FC5 357-0BB11-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572	6FC5 357-0BA20-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572	6FC5 357-0BA20-1AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572	6FC5 357-0BA21-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572	6FC5 357-0BA21-1AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572.2 (export version) with PROFIBUS DP	6FC5 357-0BA21-1AE1	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 572.2 without system sw	6FC5 357-0BB21-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 572.3 without system sw	6FC5 357-0BB22-0AE0	6ES7 315-2AF01-0AB0	10D/840D with PLC315-2AF01
SINUMERIK 840D NCU 572 (export version)	6FC5 357-0BY20-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572 (export version)	6FC5 357-0BY20-1AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572 (export version) with PROFIBUS DP	6FC5 357-0BY21-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572 (export version) with PROFIBUS DP	6FC5 357-0BY21-1AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572.2 (export version) with PROFIBUS DP	6FC5 357-0BY21-1AE1	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572 with digitizing	6FC5 357-0BA24-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572.2 with digitizing and PROFIBUS DP	6FC5 357-0BA24-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 572.2 without system sw	6FC5 357-0BB24-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572 (export version) with digitizing	6FC5 357-0BY24-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
SINUMERIK 840D NCU 572.2 (export version) with digitizing and PROFIBUS DP	6FC5 357-0BY24-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573	6FC5 357-0BA30-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840DE NCU 573 (export version)	6FC5 357-0BY30-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 573 with digitizing	6FC5 357-0BA31-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 573 (export version) with digitizing	6FC5 357-0BY31-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 573 with PROFIBUS DP	6FC5 357-0BA32-0AE1	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573 (export version) with PROFIBUS DP	6FC5 357-0BY32-0AE1	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573 with PROFIBUS DP	6FC5 357-0BA33-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573 (export version) with PROFIBUS DP	6FC5 357-0BY33-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573.2 (Pentium Pro) up to 12 axes with PROFIBUS DP	6FC5 357-0BA32-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) up to 31 axes with PROFIBUS DP	6FC5 357-0BA33-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 573.2 without system sw	6FC5 357-0BB33-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 573.2 Pentium II without system software	6FC5 357-0BB33-0AE1	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) for digitizing with PROFIBUS DP	6FC5 357-0BA31-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 573.2 without system sw	6FC5 357-0BB31-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) (export version) for 12 axes with PROFIBUS DP	6FC5 357-0BY32-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) (export version) for 31 axes with PROFIBUS DP	6FC5 357-0BY33-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
SINUMERIK 840D NCU 573.2 (Pentium Pro) (export version) for digitizing with PROFIBUS DP	6FC5 357-0BY31-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840Di	6FC5 220-0AA00-1AA0	6ES7 315-2AF03-0AB0	810D/810Di with PLC315-2AF03
SINUMERIK 840Di with PK bus		6ES7 315-2AF03-0AB0	810D/810Di with PLC315-2AF03, PK bus
SINUMERIK 840D NCU 572.3	6FC5 357-0BB22-0AE0	with operating system 03.10.23: 6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572.3	6FC5 357-0BB22-0AE0	with operating system 12.30.07: 6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03
SINUMERIK 840D NCU 573.3	6FC5 357-0BB33-0AE2	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.3	6FC5 357-0BB33-0AE2	with operating system 12.30.07: 6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03 (as of STEP7 V5.0 and Toolbox 05.03.05)
SINUMERIK 840D NCU 572.4	6FC5 357-0BB23-0AE0	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2DP
SINUMERIK 840D NCU 573.4	6FC5 357-0BB34-0AE1	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2DP
SINUMERIK 810D CCU3	6FC5 410-0AY03-0AA0	6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03
SINUMERIK 840D NCU 571.3	6FC5 357-0BB11-0AE1	6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03
SINUMERIK 840D NCU 561.3	6FC5356-0BB11-0AE1	6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03 (as of STEP7 V5.0 and Toolbox 05.03.05)
SINUMERIK 840D NCU 571.4	6FC5357-0BB12-0AE0	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2 DP (STEP7 V5.1 SP3 and higher and Toolbox 06.03.02)
SINUMERIK 840D NCU 561.4	6FC5356-0BB12-0AE0	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2 DP (STEP7 V5.1 SP3 and higher and Toolbox 06.03.02)
SINUMERIK 840D NCU 573.5	6FC5357-0BB35-0A E0	6ES7 317-2AJ00-0AB0	810D/840D with PLC317-2 DP (STEP7 V5.2 SP1 and higher and Toolbox 06.05.02)
SINUMERIK 840Di with MCI2	6FC5 222-0AA02-1AA0	6ES7 317-2AJ00-0AB0	810Di with PLC317-2 DP (STEP7 V5.2 SP1 and higher and Toolbox 06.05.02)

Note

On the SINUMERIK 810D or 840D, SIMATIC subrack 0 is integrated in the NC. The following components are plugged into this subrack:

- Slot 2: The integrated PLC (PLC 314 or PLC 315-2DP)

- Slot 3: An IM 360

- Slot 4: The FM NCU
With PLC 314, NC software version 3.5 and higher, the FM NCU must also be defined if further MPI (C bus) devices are included in subrack 1 to subrack 3 (e.g., FM modules with C bus connection). The properties of the FM NCU must not be changed, as process interrupts (e.g. auxiliary functions) of the NCU may, in this case, no longer function.

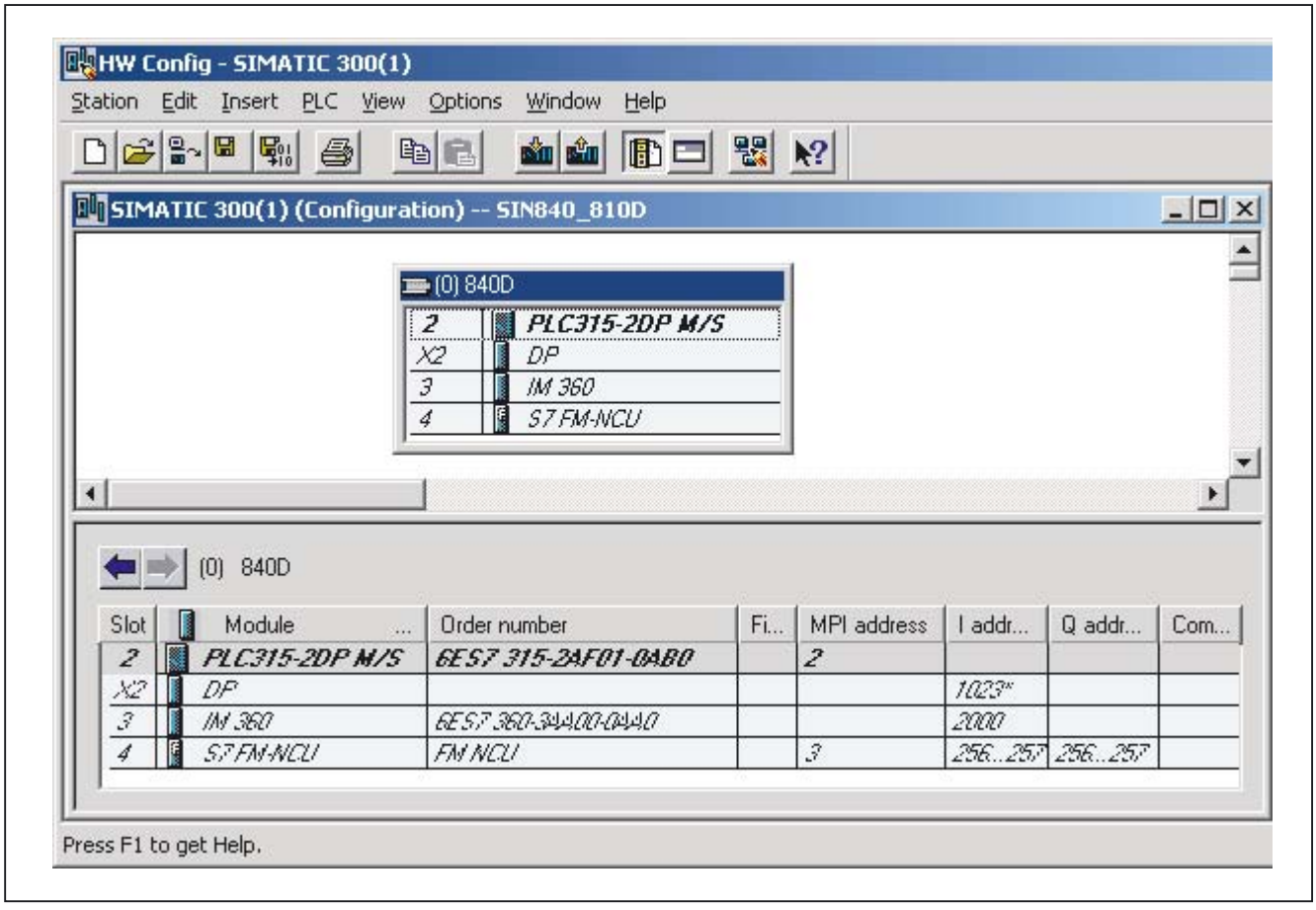


Fig. 2-1 Hardware configuration of row 0 on 810D, 840D

MCP (Machine Control Panel) and HHU (HandHeld Unit)

(only for SINUMERIK 810D to SW 3.x)

If the MCP or HHU is configured (deviation from the norm), an additional SIMATIC 300 station must be inserted into the machine project for each operator component. Any type of CPU must be inserted in location 2 on row 0 in this station by means of the hardware configuration (HW config.). The MPI address of the operator component must be set as the MPI address. MPI network (1) can then be marked in the SIMATIC manager. The global data can then be activated via the "Tools" menu item.
The rest of the procedure is described in detail in the Installation and Startup Guide.

2.4 Starting up the PLC program

2.4.1 Installing the basic program for 810D, 840D

Before initial startup of the NC, an NC and PLC general reset must be carried out to initialize the relevant memory areas. To do this, set switch S3 to setting 1 and switch S4 to setting 3 and switch the control off and on again (POWER ON reset).

Installation

With Toolbox SW 6.1 and higher, the installation is performed by a WINDOWS-compliant setup program for the basic program, hardware selection in STEP7 (SINUMERIK 810D/840D option package) and the NC-Var Selector. To start the installation, run "setup.exe" in the main CD directory. You can then choose which components to install. After the installation, you can select the basic program library directly from STEP 7 (gp8x0d61, 61 is the main basic program version).

The concrete version of the basic program can be scanned for the object properties of the library or the program folder in the comment field.

From software version 3.7 to version 4.2, installation is carried out using INSTALL.BAT (INSTALL1.BAT, INSTALL4.BAT).
This program installs the basic program and additional files for the relevant STEP7 version. With automatic installation in STEP 7 V3 and higher, the TYP, GSD and meta files in the hardware catalog are also augmented and updated.
The hardware components of the SINUMERIK system are then also available for hardware configuration under STEP 7. This eliminates the need for unzipping, mentioned below.
The basic program is supplied in zipped format as a **project** for STEP 7 Version 1.x or as a **library** for STEP 7 Version 2.x and subsequent versions.

General

The OB source programs, including standard parameterization, interface symbols and DB templates for handheld unit and M decoding functions are enclosed in the SIMATIC project or SIMATIC library of the basic program.

STEP 7 must be installed before the basic program.

STEP7 V1.x

The basic program is stored as a compressed file with the name GP840D.EXE (or GP810D.EXE and GPFMNC.EXE) in the main directory on the diskette. The basic program (GP840D.exe) must be copied to the main directory (root) of a drive (e.g., c:\) and then called. The project structure required for the basic program is generated automatically. The catalog name of the basic program is GP840Dxy.S7A. In this case, xy stands for the basic program version.

Note

When STEP 7 V1 is used, the GP840Dxy.S7A catalog must be copied to the root directory. Any existing catalog with the same name GP840Dxy.S7A must be deleted beforehand.

STEP7 V2.x, 3.x

The basic program is stored as a compressed file with the name "GP840D.EXE" in directory S7V2.840 or S7V2.810 or S7V2 on the basic program diskette. The basic program (GP840D.exe) must be copied to subcatalog "S7LIBS" of STEP7 V2 (step7_v2) or subsequent versions thereof and then called. The library structure required for the basic program is generated automatically. The catalog name of the basic program is GP840Dxy. In this case, xy stands for the basic program version. The file "MET.EXE" must be copied to the basic catalog of STEP 7 and called from there via the DOS window with "MET.EXE –O".

Note

The name GP840D specified above refers to the basic program of the SINUMERIK 840D. The basic program is named GP810D on the SINUMERIK 810D and GPFMNC on the FM-NC. With effect from SW 4.2, the basic program for 810D and 840D is combined. It is now called GP8x0D.

2.4.2 Application of basic program

A new CPU program (e.g., "Turnma1") must be set up in a project by means of the STEP7 software for each installation (machine).

Comment

The catalog structures of a project and the procedure for creating projects and user programs are described in the relevant SIMATIC documentation.

STEP7 V1

A network link to the PLC must be activated for the machine CPU program under menu items "Edit", "Configuration". This is done in the "Services", "Parameterize" menu followed by selection of the MPI parameters.

Default:

- **"Networked"**
- "MPI subnetwork number = 0"
- "CPU MPI Addr = 2"

The following must be copied into the CPU program for the machinespecific program files:

1. The basic-program blocks (FCs, FBs, DBs, OBs, SFCs, SFBs, and UDTs) ("File", "Manage Project" menu in the Step7 program editor).
2. File GPOB840D.AWL (or GPOB810D.AWL or GPOBFMNC.AWL) and other STL (AWL) files if appropriate must also be copied from the basic program catalog into the CPU program. The OBs contained in this file are the basis for the user program with the associated basic program calls. Existing user blocks must be copied as STL files to the newly created CPU program (catalog name CPU1.S7D) and compiled.
3. We also recommend that the symbolic names are transferred with the files from the basic program package using the symbol editor.

STEP7 V2

The basic-program blocks are copied using the SIMATIC Manager and "File"/"Open"/"Library".

The following sections must be copied from the library:
AP off: FCs, FBs, DBs, OBs, SFC, SFB, UDT and the SDB container.

For the 810D, the SDB container includes:

- The source_files (SO):
  - GPOB810D or GPOB840D
  - Possibly MDECLIST, HHU_DB and others
- Symbol table (SY)

Note

The SDB container is only included for these control variants.

Compatibility with STEP 7

No interdependencies exist between the basic program (including older program versions) and currently valid versions of STEP 7.

2.4.3 Version codes

Basic program

The basic-program version, including the control type, is output in the version display of the MMC in SW 4 (NCK, PLC) and higher.

In earlier versions of the basic program, the PLC version is stored in data block DB 17 on the data double word DBD 0. A HEX number must be set as the data format.

For example, the following display appears in V 3.2:

DB17.DBD0: 03**32**_0100

The version is shown in decades 3 and 4 of the display (bold type). The two leftaligned decades contain the control type of this basic program. The remaining decades contain a development code.

The type of control is encoded as follows:

Leftaligned decade of DB17.DBD0 (byte 0)	Type of control
01	FM-NC
02	SINUMERIK 810D
03	SINUMERIK 840D (571, 572, 573)
04	SINUMERIK 840Di

User program

In SW 4 and higher, users can also display the version codes of their own user programs on the MMC.

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. Parameter assignments for this string are made via a pointer in FB 1. Parameterization requires symbolic definition of the data block.

References:
/FB1/ Description of Functions, Basic Machine; PLC Basic Program (P3);
Chapter: Block Description

2.4.4 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 NC (e.g., read/write NC data, tool-management acknowledgments, etc., are activated and executed via basic-program FCs and FBs). The machine program can be created in different creation languages (e.g., STL, LAD, CSF, 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 the blocks, which call them. 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 recompiled. This general procedure applies analogously to instance data blocks for FBs. If this sequence of operations is not observed, timestamp conflicts occur when the data are retranslated into STEP7. In some cases, therefore, it may not be possible to recompile blocks, creating problems, for example, with the "Block status" function. It is moreover advisable to generate blocks in ASCIISTL by means of the STEP7 editor when they have been created in the ladder diagram or in single statements (incremental mode).

2.4.5 Data backup

The PLC CPU does not store symbolic names, only the data-type descriptions of the block parameters (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) and the data types of the global data blocks. Without the associated project for this machine, therefore, blocks cannot be recompiled meaningfully (e.g., for the Block status function or in the event of modifications subsequently required to PLC CPU programs). It is therefore necessary to keep a backup copy of the STEP7 project located in the PLCCPU on the machine. This is extremely useful for servicing purposes and saves time and problems. If the STEP7 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 STL 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.

2.4.6 PLC series startup, PLC archives:

After the blocks have been loaded to the PLC CPU, a series archive can be generated via the MMC operator 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, and does not back up, e.g., symbolic information. In addition, no CPU DBs (SFC 22 DBs) or SDBs generated in the CPU are saved.

In Toolbox 06.03.03 and STEP 7 V5.1 and higher, the PLC series archive can be generated directly from the corresponding SIMATIC project.
To do this, select the "Options" → "Settings" menu item and the "Archive" tab in STEP 7. This contains an entry "SINUMERIK (*.arc)", which must be selected to create a series startup file. After selection of the archive, select the "File" → "Archive" menu item. The relevant series archive will then be generated. If the project contains several programs, the program path can be selected. A series archive is set up for the selected program path. All blocks contained in the program path are incorporated into the archive, except for CPU-DBs (SFC 22 DBs).

The process of generating a series archive can be automated (comparable to the command interface in STEP 7, V5.1 and higher). 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")
rem Instantiate command interface of STEP7
Set S7Ext = CreateObject("SimaticExt.S7ContainerExt")
Call S7Ext.Magic("")
```

Functions:

Function **Magic**(bstrVal As String) As Long

Function **MakeSeriesStartUp**(FileName As String, Option As Long, Container As S7Container) As Long

Function **Magic**(bstrVal As String) As Long

Call gives 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 Step7 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 **MakeSeriesStartUp**(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 (supported in MMC SW 6.2 and higher)

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)

```
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 i = 1 to S7Prog.Next.Count
        Set Cont = S7Prog.Next.Item(i)
    Check block container
        If (Cont.ConcreteType = S7BlockContainer) Then
            Exit For
        End if
Next
Error = S7Ext.MakeSerienIB("f:\dh\arc.dir\PLC.arc", 0, Cont)'Error analysis now'
```

2.4.7 Software upgrades

Software upgrade

Whenever you update the PLC or NCK software, always reset the PLC to its initial state first. This initial clear state can be achieved by means of a general PLC reset. All existing blocks are cleared when the PLC is reset.
It is usually necessary to include the new basic program when a new NC software version is installed. The basic programs blocks must be loaded into the user project for this purpose. OB 1, OB 40, OB 100, FC 12 and DB 4 should not be loaded if these blocks are already included in the user project. These blocks may have been modified by the user. The new basic program must be linked with the user program.

To do this, proceed as follows:

1. Generate the text or source file of all user blocks before copying the basic program.
2. Then copy the new basic program blocks to this machine project (for a description, see Subsection "Application of basic program")
3. All user programs *.awl must then be recompiled in the correct order! (See also the "Machine program" section.)
   This newly compiled machine program must then be loaded to the PLC CPU using STEP 7.

However, it is normally sufficient to recompile the organization blocks (OB) and the instance data blocks of the machine program. This means you only need to generate sources for the organization blocks and the instance data blocks (before upgrading).

Overall reset

A description of how to perform a general PLC reset appears in the Installation and Startup Guide. However, a general reset does not delete the contents of the diagnostic buffer nor the node address on the MPI bus. Another possible general reset method is described below. This method must be used when the normal general reset process does not work.

Proceed as follows:

No.	Action	Effect
1	Control system is switched off	
2	PLC switch setting 3 (MRES) and switch control on again or perform hardware reset.	LED labeled PS flashes slowly.
3	Set PLC startup switch to position 2 (STOP) and back to position 3 (MRES).	The LED labeled PS starts to flash faster.
4	Set PLC startup switch to setting 2 or 0.	

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 version, i.e., selected variables in an older software version must not be reselected when the software is upgraded.

2.4.8 I/O modules (FM, CP modules)

Special packages for STEP7 are generally required for more complex I/O modules. Some of these special packages include support blocks (FC, FB) stored in a STEP7 library. The blocks contain functions for operating the relevant module which are parameterized and called by the user program. In many cases, the FC numbers of the CP and FM module handling blocks are also included in the number range of the basic program for the 810D and 840D systems.
What action can be taken if such a conflict occurs?
The block numbers of the basic program must remain unchanged. The block numbers of handling blocks can be assigned new, free numbers using STEP7. These new blocks (with new FC numbers) are then called in the user program with the parameter assignments required by the function.

2.4.9 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	**Error**	**Cause/description**	**Remedy**
1	No connection via MPI to PLC.	The MPI cable is not connected or is defective. The STEP 7 software for the MPI card may not be configured correctly.	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 are missing, the MPI cable is defective or not connected. In Windows 95 versions of STEP 7, the hardware must be checked in the Control Panel; the PC/PG interface configuration should also be checked as well.

Errors, cause/description and remedy			
Serial no. error information	**Error**	**Cause/description**	**Remedy**
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. **Actions:** 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 switch S4 to position 2). A hardware reset must then be performed. The diagnostic buffer can then be read out with STEP7. 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.

2.5 Linking PLC CPUs to 810D, 840D

General

The AS 300 family is used as the PLC for all systems. The essential difference between the NCU variants lies in the method by which they are linked. On the 840D and 810D, the PLC 314 CPU (user memory capacity up to 128 KB) or PLC 315-2 DP (user memory capacity up to 288 KB) is integrated as a submodule into the NC unit.
The PLC/CPU 315-2 DP also supports distributed I/Os on the PROFIBUS (L2DP). The relevant performance data for individual PLC CPUs can be found in the above table or in Catalog FT70.

2.5.1 Properties of PLC CPUs

SINUMERIK 810D/840D/840Di PLC CPUs are based on standard SIMATIC CPUs in the S7-300 family. As a result, they generally possess the same functions. Functional deviations are shown in the table above. Owing to differences in their memory system as compared to the S7 CPU, certain functions are not available (e.g., blocks on memory card, project on memory card).

Note

With the current SIMATIC CPUs, the PLC is not automatically started after voltage failure and recovery when a PLC Stop is initiated via software operation. In this instance, the PLC remains in the Stop state with an appropriate diagnostic entry for safety reasons. You can start the PLC only via software operation "Execute a restart" or by setting the switch to "Stop" and then "RUN". This behavior is also integrated in the current versions of the SINUMERIK PLC.

2.5.2 Interface on 810D and 840D with integrated PLC

Physical interfaces

As the 810D and 840D systems have an integrated PLC, signals can be exchanged between the NCK and PLC directly via a dualport RAM.

Exchange with operator panel and MCP

Data are generally exchanged with the operator panel (OP), machine control panel (MCP) and handheld unit (HHU) on the 840D via the operator panel interface (OPI), the COM module being responsible for data transport.

All devices specified above can also be operated on the multipoint interface (MPI) in the case of the 840 D. With the 810D, data communication with the operator panel (OP), machine control panel (MCP) and handheld unit (HHU) takes place only via the MPI.

The programming device is connected directly to the PLC via the MPI (multipoint interface).

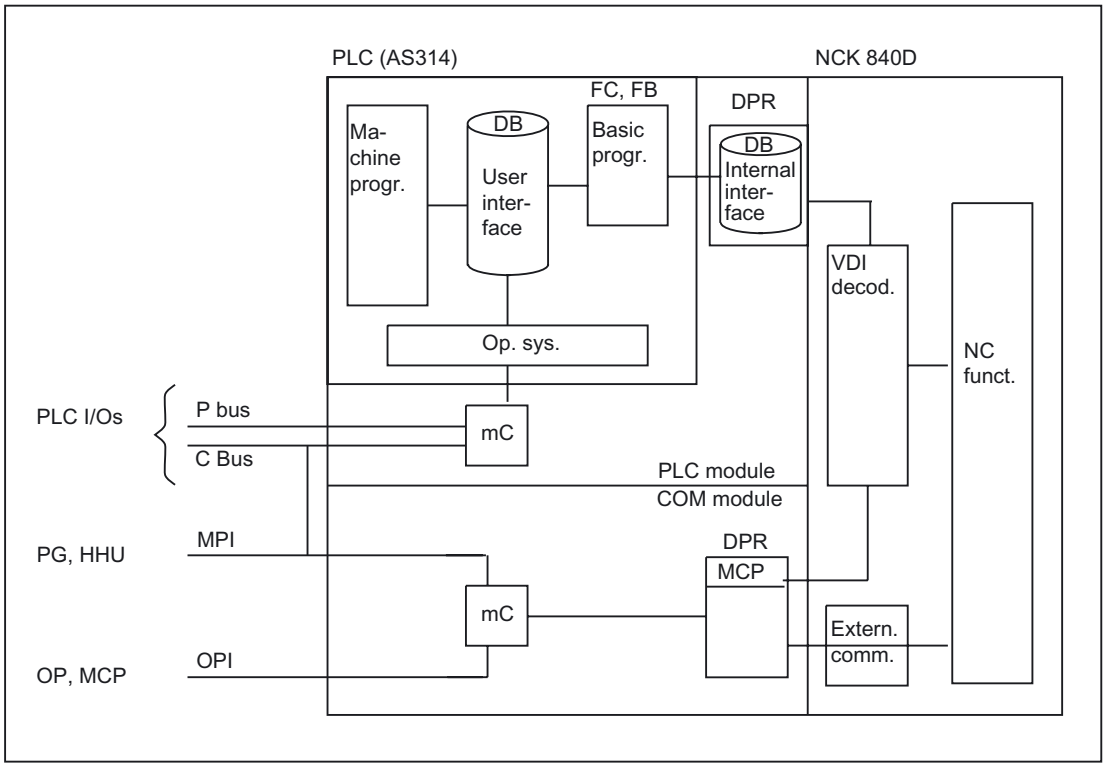


Fig. 2-2 NCK/PLC connection on 810D, 840D (integrated PLC)

NCK/PLC interface

NCK/PLC data exchange is organized by the basic program in the PLC.
The **status information** (e.g., "Program running") stored in the internal DPR is copied to data blocks by the basic program at the beginning of the cycle (OB1), which the user can then access (user interface). The user also enters NC **control signals** (e.g., NC start) in the interface data blocks, and these are also transferred to the NC at the start of the cycle.

Auxiliary functions transferred to the PLC dependent on the workpiece program are first evaluated by the basic program (alarmdriven) and then transferred to the user interface at the start of OB1. If the relevant NC block contains auxiliary functions that require the interruption of the NC machining process (e.g., `M06` for tool change), the basic program halts the execution of the block on the NC 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 does not contain auxiliary functions requiring the interruption of the NC machining process (e.g., `M08` for "Cooling on"), the transfer of these "rapid" auxiliary functions is enabled directly in OB 40, so that block execution is only marginally 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 must be loaded by the user and called in the PLC program of the machine manufacturer (machine program). You can find an overview of the FC, FB and DB blocks, divided according to their basic and expanded function, in:
References:
/FB1/ Description of Functions, Basic Machine; PLC Basic Program (P3);
Chapter: Starting Up the PLC Program

OP/PLC interface

Data are exchanged between the OP and PLC via the OP/NC serial bus, COM module and C bus. The COM module transfers the data intact from one bus segment to another. It merely converts the baud rate. The OP is always the active partner (client) and the PLC is always the passive partner (server). Data transmitted or requested by the OP are read from and written to the OP/PLC interface area by the PLC operating system (timing: Cycle control point). From the viewpoint of the PLC application, the data are identical to I/O signals.

MCP/PLC interface, HHU/PLC interface (840D only)

Data are exchanged between MCP/PLC and HHU/PLC via the serial bus MCP, HHU/NC, COM module, and NCK. The NCK transfers the MCP/HHU signals to and fetches them from the internal NC/NCK 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) and the start addresses.

MCP/PLC interface, HHU/PLC interface (810D only)

Data exchange between MCP/PLC and HHU/PLC takes place via the MPI interface on the PLC. The Communication with global data (GD)[1)] service is used for this purpose (see also STEP7 User's Guide). The PLC operating system handles the transfer of signals from and to the user interface. The STEP7 **Communication configuration** configuring tool is used to define both GD parameters as well as operand areas (e.g., I/O) and their initial addresses. In SW 2.2 and higher, data exchange is possible as on the 840D.

[1)] IC (GD) = Implicit Communication (Global Data)

2.5.3 Diagnostic buffer on PLC

General

The diagnostic buffer on the PLC, which can be read out using STEP 7, displays diagnostic information about the PLC operating system. In addition, 10 entries are made to the diagnostic buffer via the FC by means of the basic program and the "Alarms/Messages" function. These alarms and messages are indicated in the diagnostic buffer as an event with "Event ID: <ID>" without any explanatory text.

Meaning of displayed data

An example is presented below to illustrate what information is displayed for an event and how to interpret this information.

Diagnostic buffer (extract, relevant data selected):

Event details: 1 of 10	Event ID: 16# B046
No entry in text database. Hex values are displayed.	
Event ID:	16# **B046**
OB:	16# 01
PK:	16# 01
DatID 1/ 2:	16# 59 C9
Supplementary info1 / 2 / 3:	16# 02**00** 0000 00**20**
Outgoing event:	36:02:459 08.04.03

Type of event

The two most significant digits of Event ID 16# **B0**46 contain the code for the event type:

A1:	Alarm set
A0:	Alarm cancelled
B1:	Message set
B0:	Message cancelled

Alarm/Message Number

The two least significant digits of Event ID 16# B0**46** contain the code for the most significant decimal places of the alarm or message number. The two least significant digits of supplementary information 1 / 2 / 3 must also be taken into account when determining the complete alarm or message number:

- Supplementary information 1 16# 02**00** 0000 0020
- Supplementary information 2 16# 0200 0000 00**20**

All values must be converted to decimal values and lined up in order of alarm or message number:

Event ID:	46 hex	=	70 dec
Supplementary information 1	00 hex	=	00 dec
Supplementary information 2	20 hex	=	32 dec
Alarm number (OM) 1)		=	**70 00 32**

1) controlled by DB2.DBX184.0

OM = operational message

This signifies the event with Event ID 16# B046 and the indicated supplementary information "Message 700032 deleted".

Notes

- The meaning of the message number is specified by the machine manufacturer.
- Events with Event ID 16# xx**28** or 16# xx**29** are generated from the basic program.
- The messages stored in the diagnostic buffer can be read out on the MMC with the associated message texts.

2.6 Interface structure

Interface data blocks

The PLC user interfaces on the 840D and 810D are identical except for the data volume. Mapping in interface data blocks is required on account of the large number of signals. These are global data blocks from the viewpoint of the PLC program. During system startup, the basic program creates these data blocks from current NC 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.

2.6.1 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:

- NCKspecific signals
- Mode groupspecific signals
- Channelspecific signals
- Axis/spindle/drivespecific 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.

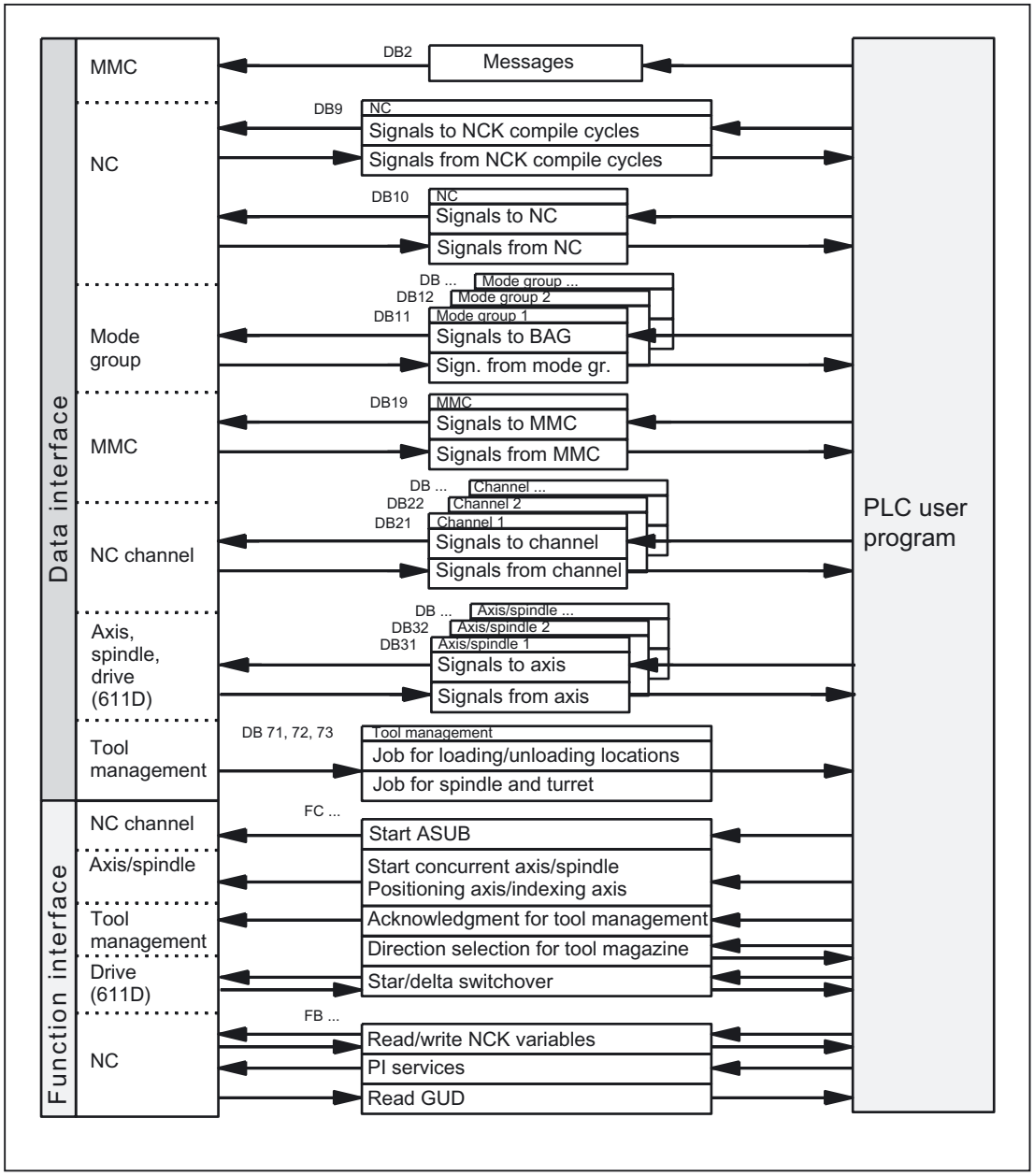


Fig. 2-3 PLC/NCK interface

Compile-cycle signals

In addition to the standard signals exchanged between the PLC and NCK, an interface DB for compile cycles is also generated if required (DB 9). The signals, which are dependent on the compile cycles, are transmitted cyclically at the start of OB1.

Signals PLC/NC

The group of signals from the PLC to NC includes:

- Signals for modifying the highspeed digital I/O signals of the NC
- Keyswitch and emergency stop signals

Signals NC/PLC

The group of signals from the NC to PLC includes:

- Actual values of the digital and analog I/O signals of the NC
- Ready and status signals of the NC

Also output in this group are the handwheel selection signals from the MMC and the status signals of the MMC.

The signals for handwheel selection are decoded by the basic program and entered in the machine/axisspecific interface.

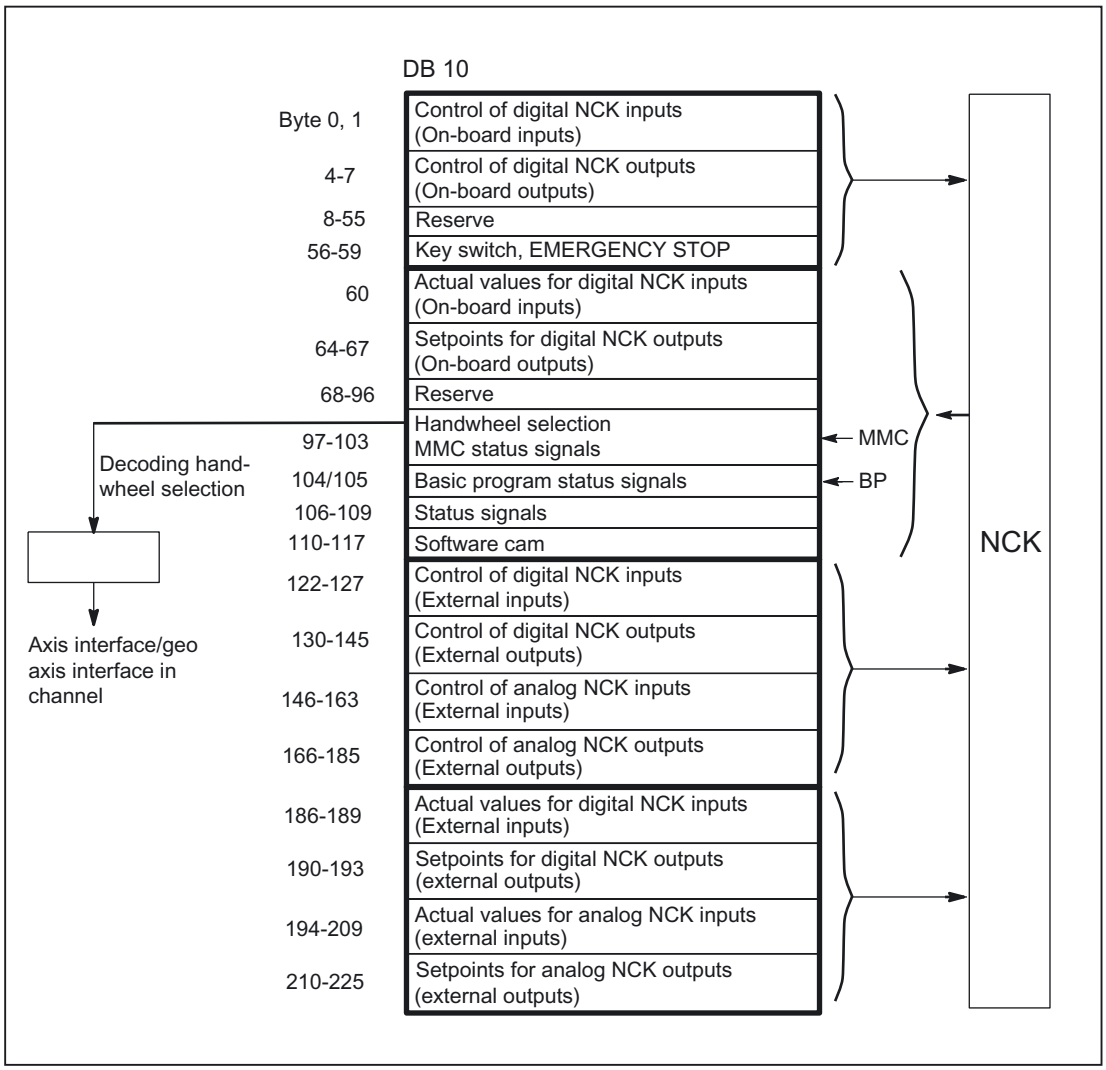


Fig. 2-4 PLC/NC interface

Digital/analog inputs/outputs of the NCK

The following must be noted with respect to the digital and analog inputs and outputs 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 parts 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

When implementing digital and analog NCK I/Os, you must observe the information in the following references:

References:
/FB2/ Description of Functions, 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 MMC are transferred to the operating mode group (mode group). The mode signals are valid for all NC channels of the mode group on the 810D and 840D. On 840D systems, several mode groups can optionally be defined in the NCK.

The mode group reports its current status to the PLC.

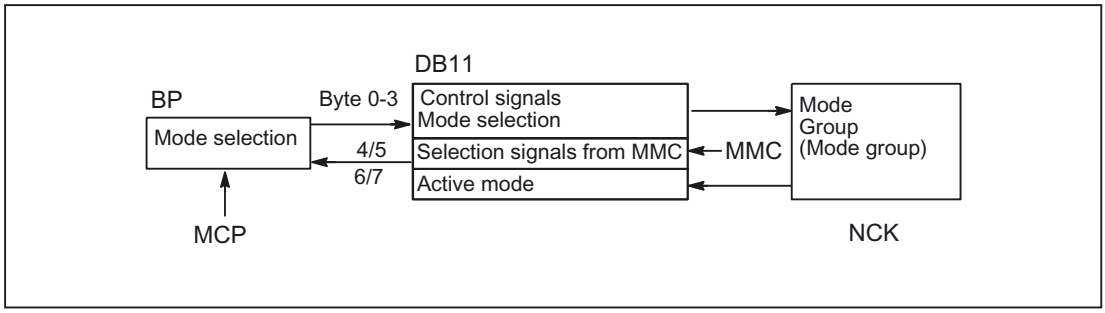


Fig. 2-5 PLC/mode group signals (e.g., mode group 1)

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 channelspecific interface by the MMC (MMC signals are entered by the PLC operating system) are also transferred at this time if the signals have been defined on the NC OP, not on the MCP.

* MMC signals are entered by the PLC operating system

Auxiliary functions and G functions are entered in the interface data blocks in two ways. First, they are entered with the change signals.

- **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.

  * M signals `M0` - `M99` are transferred from the NCK with extended address 0

- In the case of the **G functions**, the group is additionally decoded and the G functions which are active in the relevant group are entered in the interface data block.
- **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.

When the **tool management** function is activated in the NCK, the assignment of spindle or revolver and the loading/unloading points are entered (DB71-73) in separate interface DBs.

The triggering and parameter assignment of **NCK functions** is performed by means of PLC function calls.
The following function calls are available, for example:

- Position a linear axis or rotary axis
- Position an indexing axis
- Start a prepared asynchronous subprogram (ASUB)
- Read/write NC variables
- Update magazine assignment

Some of the above functions are described in their own function documentation.

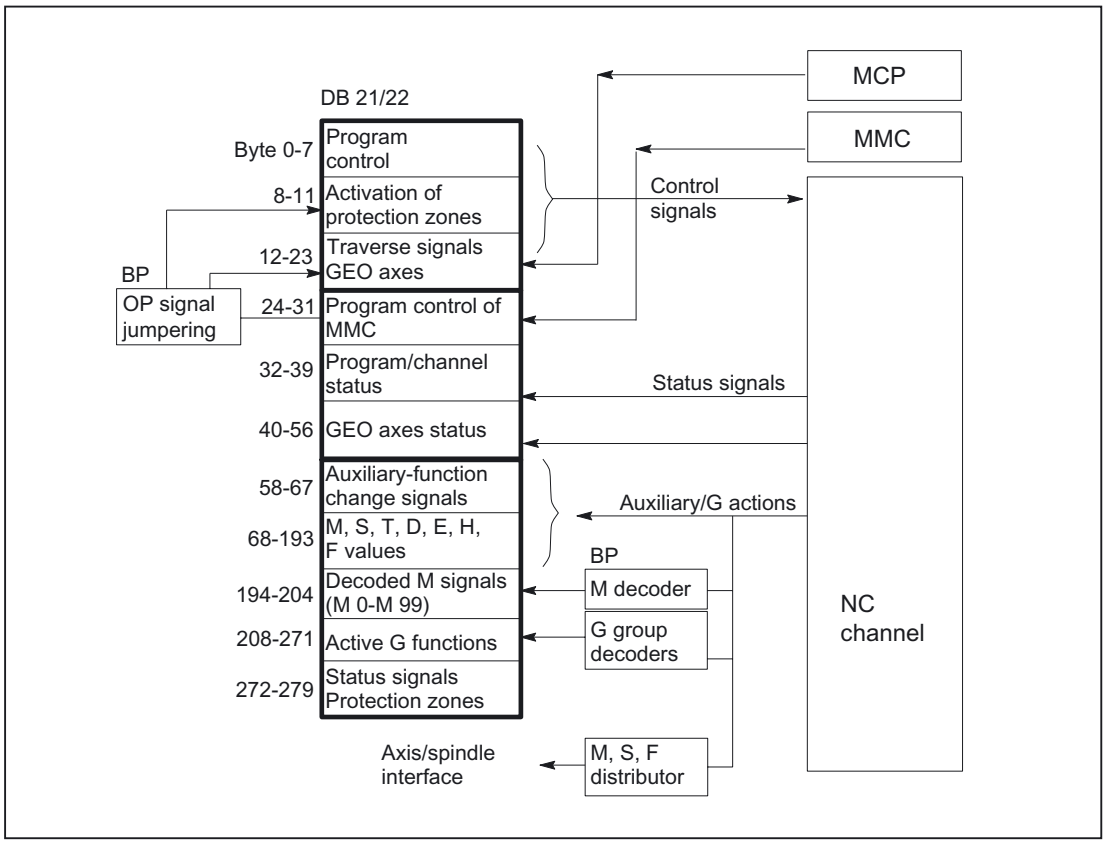


Fig. 2-6 PLC/NC channel interface

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 transferred cyclically at the start of OB 1, with the following exceptions:

Exceptions include:

MMC INC mode, axial F value, M/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 an S value requires processing together with the corresponding M value (`M03`, `M04`, `M05`).

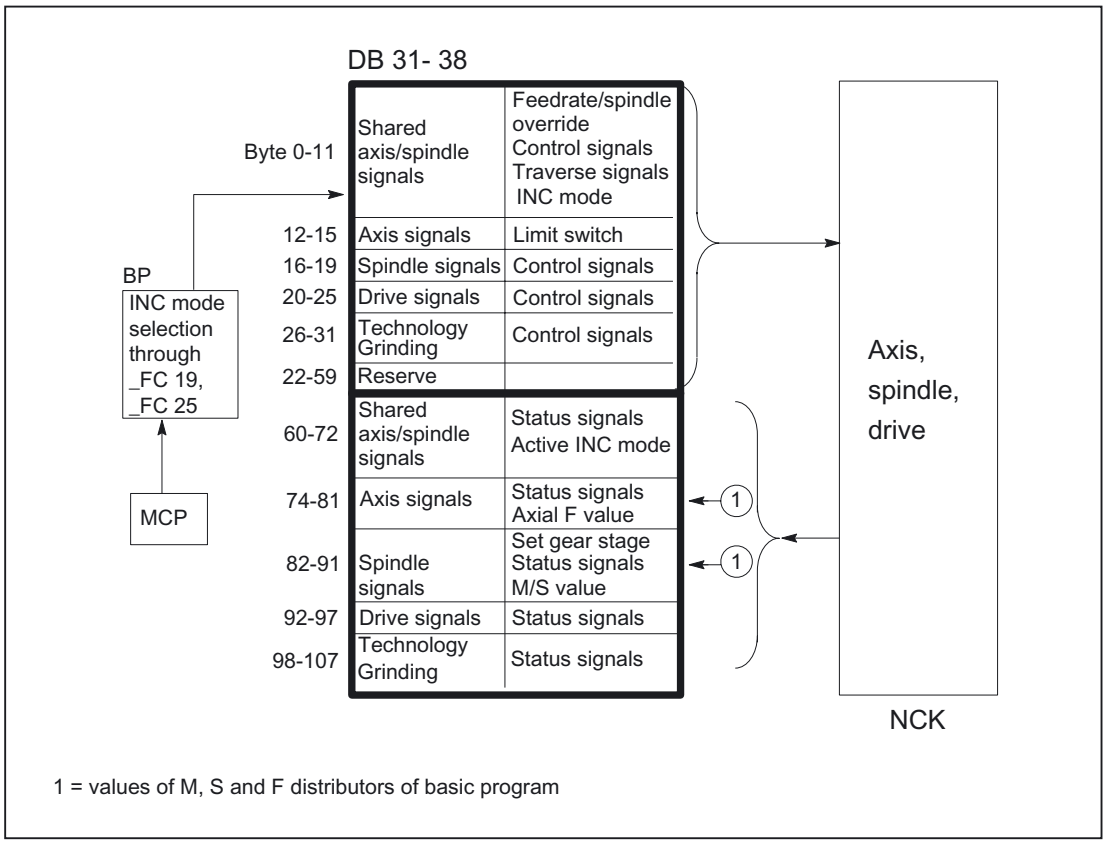


Fig. 2-7 Interface between PLC and axes/spindles/drives

2.6.2 PLC/MMC interface

General

The following groups of functions are required for the PLC/MMC interface:

- Control signals
- Machine operation
- PLC messages
- PLC status display

Control signals

In some cases, signals are input via the machine control panel and must be taken into account by the MMC. This group of signals includes, for example, display actual values in MCS or WCS, key disable, etc. These are exchanged with the MMC via a separate interface DB (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. However, it is also possible to perform some operator actions on the MMC (e.g., mode selection, INC mode selection).

The PLC operating system enters the operating signals sent by the MMC directly into the interface data blocks. In standard cases, the basic program, which decodes the operating signals, allows the operator actions to be performed on the machine control panel (provided the operator actions are available) or on the MMC. If required, the user can switch off the operation via MMC 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 (OS or MMC) via the MPI or via the OPI through the COM module, 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 is executed by the MMC software.
- An SFC (system function call) can also be used by the user program to enter messages in the diagnostic buffer.
- The events are entered in the diagnostic buffer in coded format.
  The associated message texts must be stored on the OP or MMC.

An FC (FC10) for message acquisition is prepared in conjunction with the basic program. This FB records events, subdivides them into signal groups and reports them to the MMC via the diagnostic 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 VDI 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 when at least one bit signal is set to "1". This signal is generally linked to the disable signal of the VDI interface (on modules with diagnostic functions). The group signals are acquired completely in cycles.
  - **Evaluation 2;**Acquisition of
  - **Error messages**

    A fixed specification exists to define which signals in a group generate an error message when they change from "0" to "1".
  - **Evaluation 3;**Acquisition of
  - **Operational messages**

    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 defined as standard as 10 areas with 8 bytes each, but can also be adjusted to suit the requirements of the machine manufacturer via basic program parameters in FB 1.

Acknowledgment procedures

The following acknowledgment procedures are implemented for error and operating messages:

Operating messages are intended for the display of normal operating states as information for the user. Acknowledgment signals are therefore not required for this type of message. An entry is made in the diagnostic buffer for incoming and outgoing messages. The MMC maintains an up to date log of existing operating messages using the identifiers "operating message arrived" and "operating message departed".

Error messages are used to display error states on the machine which generally lead to a machine stoppage. 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 error message is only deleted if the user has acknowledged the message (e.g., by pressing a key on the machine control panel). 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 "error departed". This enables the MMC to also maintain an up to date log of existing error messages. The time of day indicating the time of error occurrence remains available for messages which are still pending (in contrast to a received interrogation).

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 MMC.

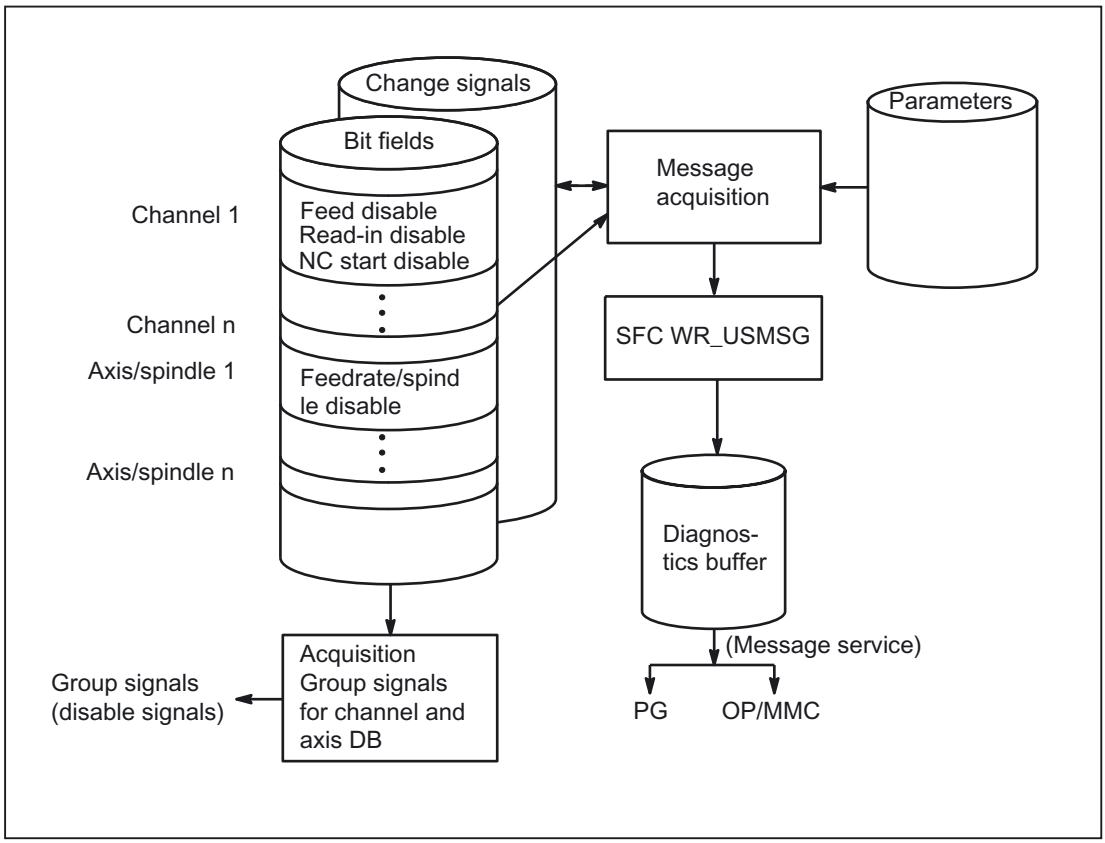


Fig. 2-8 Acquisition and signaling of PLC events

2.6.3 PLC/MCP/HHU interface

General

On the SINUMERIK 840D/810D, the machine control panel (MCP) is connected on the same bus that connects the OP with the NC. The advantage of this is that only one bus cable is required to connect the operator unit. The handheld unit (HHU) can be connected to the MPI of the PLC or to the operator panel interface (OPI) (840D only). However, since the OP bus on the 840D supports higher baud rates, two different types of bus topology are provided.

840D topology

On the 840D, the machine control panel is connected to the OPI bus segment (transmission rate 1.5 Mbps) as an active global data node. Where the connection of further keys and displays is required for customized operator panels, an additional keyboard (machine control panel without operating unit) can be used. 64 pushbuttons, switches, etc. and 64 display elements can be connected via ribbon cable.

The signals arriving from the machine control panel are copied by the COM module into the DPR (dualport RAM) for transfer to the NC. The NC in turn transmits them to the PLC (VDI task). The basic program of the PLC enters the incoming signals in the input image. The NC-related signals are generally distributed by the basic program to the VDI interface. This can be modified by the user if required.

The signals (displays) from the PLC to the MCP (displays) are transferred in the opposite direction.

The signals of the handheld unit (HHU) are transferred either via the operator panel interface in the same way as via the machine control panel or by means of the GD service (GD = Global Data) of the MPI interface. The PLC operating system enters the HHU data, for example, in the input image and transfers the display values, e.g., from the output image, back to the HHU. The corresponding parameters are set via system data block SDB210, which is generated with the STEP7 communication configuration tool.

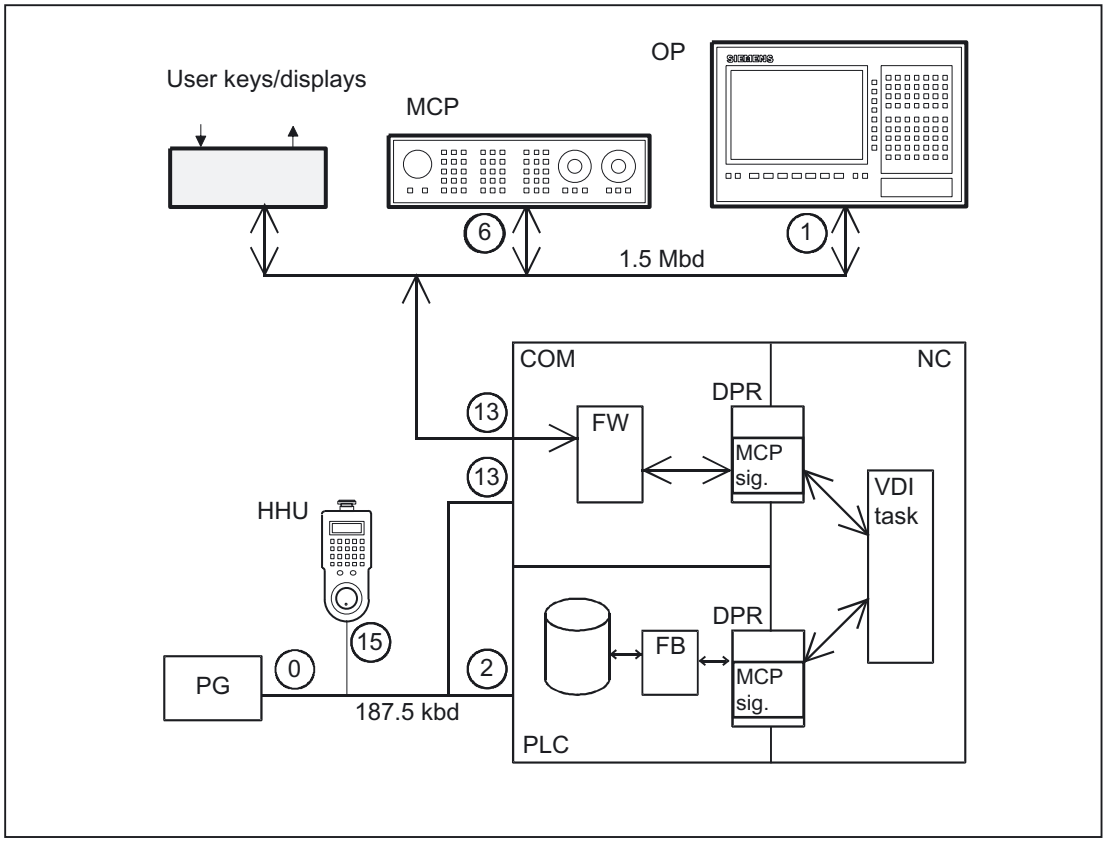


Fig. 2-9 Connection of the machine control panel for 810D

810D topology

On the 810D, the machine control panel, handheld unit and OP are connected to the MPI (multipoint Interface) of the AS 300. The transmission rate in this configuration is 187.5 Kbps. The PLC operating system copies the incoming signals straight to the user interface (e.g., input image) at the cycle control point.
Transfer to the VDI interface is performed, as on the 840D, by the user program or by a standard routine of the basic program.

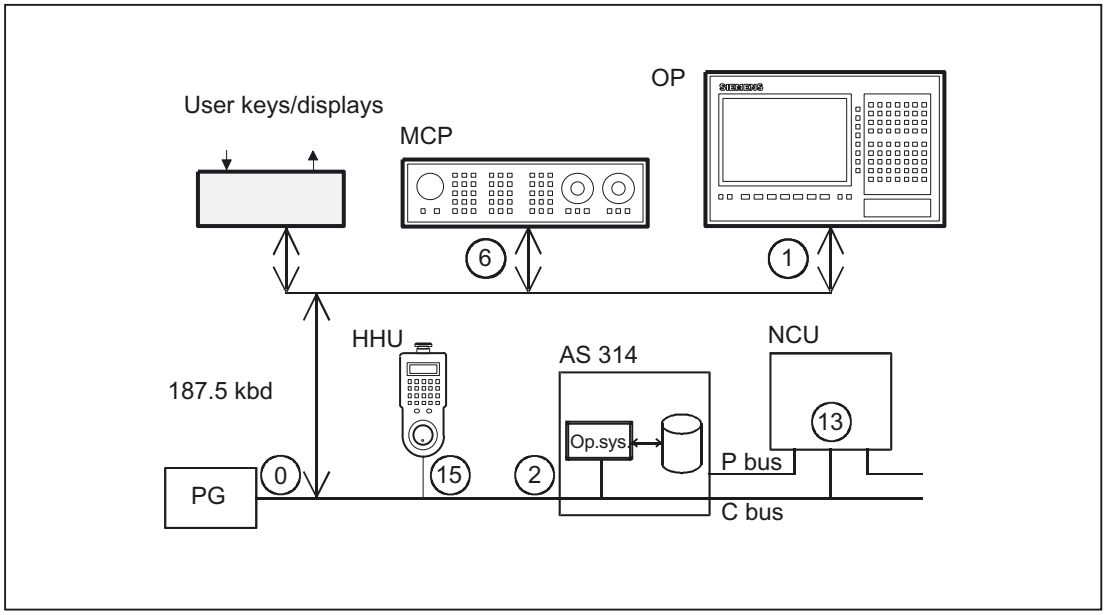


Fig. 2-10 Connection of the machine control panel for 810D

Bus addresses

The default bus addresses for the standard configurations are entered in the "Connecting the MCP on the 810D" figures. In addition to the bus addresses, the implicit communication service (global data) also requires the definition of a GD circle number.

The following should be taken into account when allocating bus addresses (node no.):

Bus addresses 840D

The two bus segments on the 840D must be examined separately:

Operator panel bus segment:		
Bus station	**Perm. setting range**	**Standard setting**
Operator panel (OP)	1 - 31	1
Machine control panel/keyboard interface	15	(setting via DIP fix)
COM module	31	13
Programming device/PC (e.g., for startup)	fixed	0

PLC bus segment:			
Bus station	**Setting range**	**Default setting**	**Comment**
PLC	31	2	
COM module	Fixed depending on PLC address	3	
Programming device/PC (e.g., for startup)	fixed	0	

MCP interface in the PLC

The signals from the machine control panel are routed via the I/O area by default. A distinction is made between NC and machinespecific signals. NCspecific key signals are normally distributed by FC 19 to the various mode group, NCK, axis and spindlespecific interfaces. The reverse applies to the status signals, which are routed to the machine control panel interface.

Note

FC 19 must be called in the PLC 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.

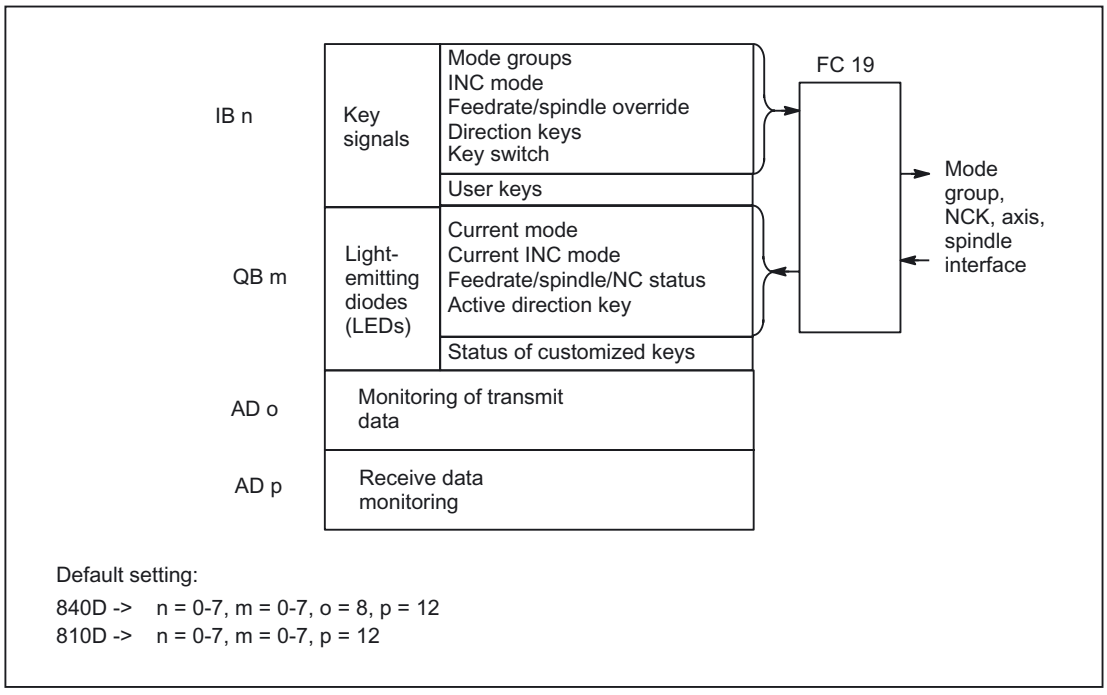


Fig. 2-11 Interface to and from machine control panel

2.7 Structure and functions of the basic program

General

The program is modular in design, i.e., it is structured according to NCK functions.

In the operating system, a distinction is made between the following levels of execution:

- Startup and synchronization (OB 100)
- Cyclic mode (OB 1)
- Process interrupt handling (OB 40)

Each section of the basic program - as illustrated in the figure below - must be called by the user in OBs 1, 40 and 100.

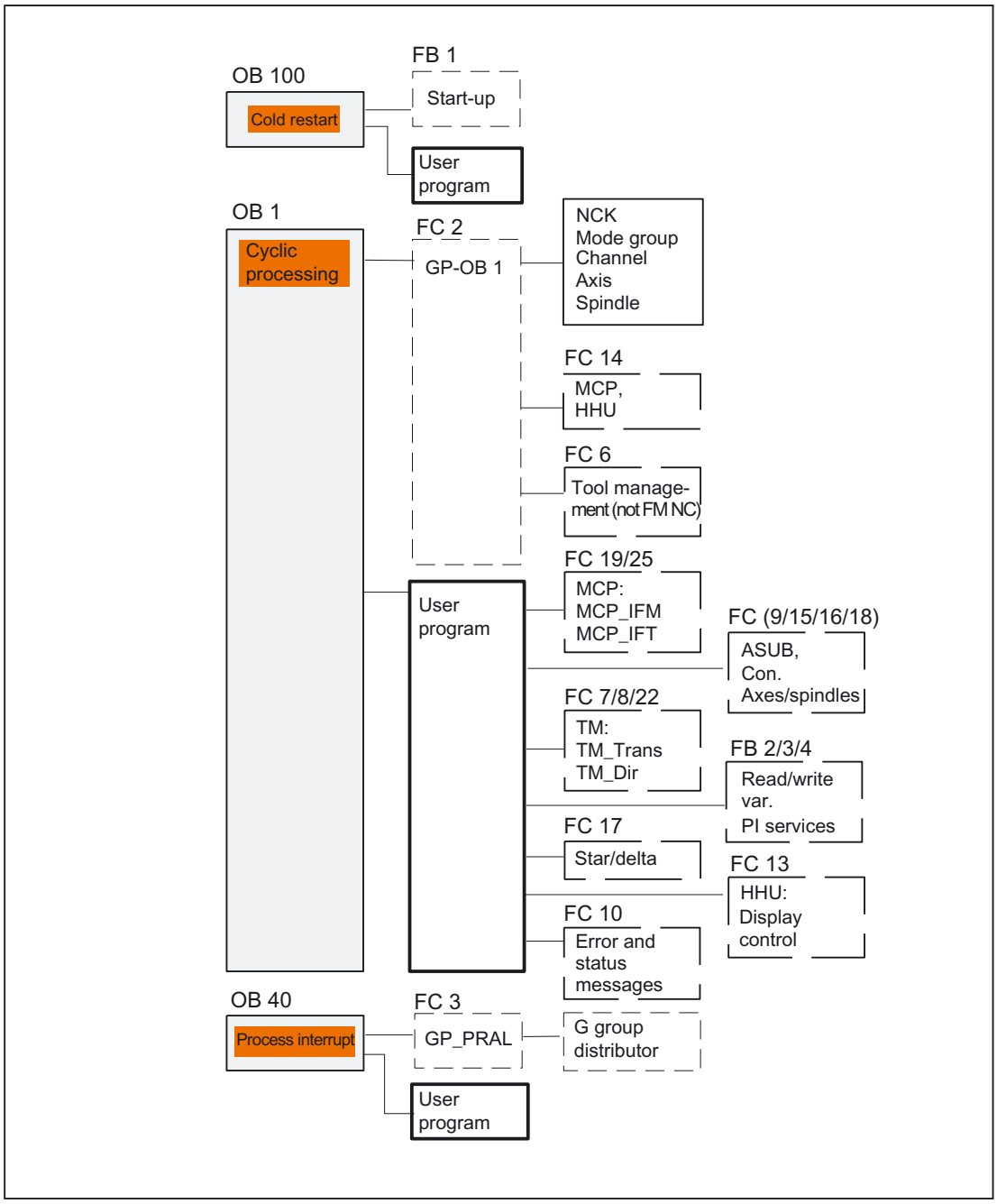


Fig. 2-12 Structure of the PLC program

2.7.1 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. An undefined state may otherwise develop on the PLC (e.g., all PLC LEDs flashing).

Startup

NCK and PLC synchronization 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 the event of an error, the basic program outputs an error identifier to the diagnostic buffer and switches the PLC to STOP.

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 MMC and NCK during powerup.

Sign of life

After a proper 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 signal from the NCK fails to arrive, the PLC/NCK interface is neutralized and the signal "NCK CPU ready" in DB 10 is set to zero.

2.7.2 Cyclic operation (OB1)

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 and G functions
- M decoding (`M00` - `M99`),
- M, S, F distribution
- Transmission of the machine control panel signals via the NCK (on the 840D only)
- 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-groupspecific signals such as operating modes
- Channelspecific signals such as program and feed modifications

Axis- and spindlespecific signals such as feed disable

Auxiliary and G functions

The auxiliary and G functions have the following characteristics:

- Transfer to the PLC is blocksynchronous (referred to a parts program block)
- Transfer is acknowledgecontrolled.
- The acknowledgment times have an immediate effect on the execution time of NC blocks containing auxiliary functions requiring acknowledgment.

The value range is presented in the table below:

Function	Structure		Value range		Data type	
	1. Value	2. Value	1. Value	2. Value	1. Value	2. 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 documentation "Lists of SINUMERIK 840D, 810D"). 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 acknowledgment is passed to the NCK. The acknowledgment of highspeed 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 both 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 spindlespecific 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

On the 840D, the MCP signals are transferred to the NCK via the serial bus (MPI) and from there to the PLC. A function call from the basic program transfers the signals to the interface of inputs and outputs specified by basic program parameters. The status signals for controlling the LEDs on the machine control panel are passed in the opposite direction.

User messages

The acquisition and processing of the user error and operational messages is performed by an FC in the basic program.

2.7.3 Time-alarm processing (OB 35)

General

The user must program **OB 35** for time-alarm processing. The default time base setting of **OB 35** is 100 ms. Another time base can be selected using the STEP7 application "S7 Configuration". However, the OB 35 with a time base setting not less than approx. 15 ms must not be used without additional measures, otherwise this would cause the PLC CPU to stop. The stop is caused by the reading of the MMC system state list during runup of the MMC. This reading process blocks priority class control for approx. 8 to 12 ms. The OB 35 with a time base set to a rather lower value is then no longer processed correctly. If, however, small time base settings are required for OB 35, the stop can be prevented by programming OB 80 with at least the program command "BE".

2.7.4 Process-alarm processing (OB 40)

General

A process alarm **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.

2.7.5 Response to NC failure

General

During cyclic operation, the PLC continuously monitors NC availability by querying the sign of life character. If the NCK is no longer reacting, then the NCK PLC interface is neutralized and IS **NCK CPU ready** in **signals from NC group (DB 10.DBX 104.7)** is reset. The signals sent by the NCK to the PLC are also set to an initial state.

The PLC itself remains active so that it can continue to control machine functions.

Signals NCK to PLC

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).

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 runup 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.

2.7.6 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 can be called at a suitable point in the user program and supplied with parameters.

These functions can be used, for example, to pass the following jobs from the PLC to the NCK:

- Traverse concurrent axes (FC 15, FC 16),
- Start asynchronous subprograms (ASUBs) (FC 9),
- Select NC programs and NC blocks (FB 4),
- Control of spindle (FC 18),
- Read/write variables (FB 2, FB 3).

Note

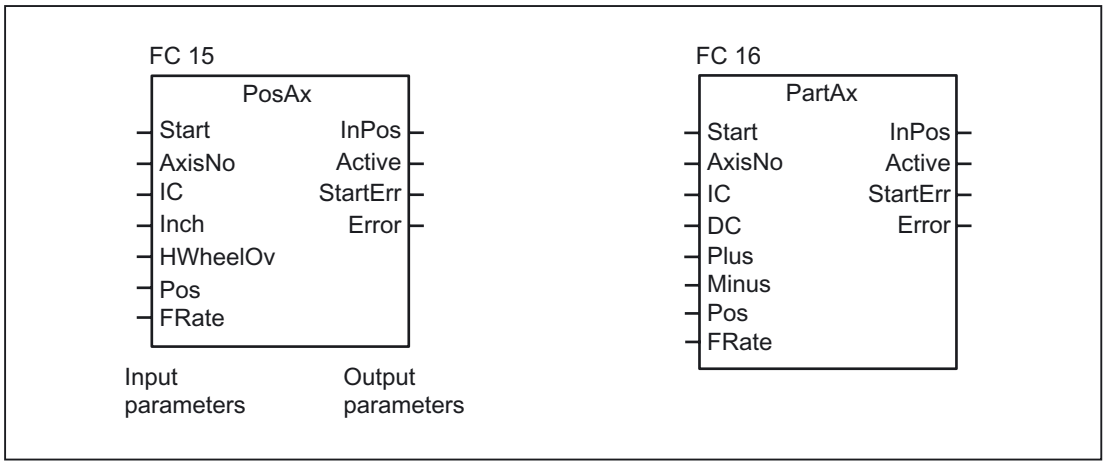
The following note will later help you to check and diagnose a function call (FCs, FBs of basic program). These are FCs and FBs, which are controlled by a trigger signal (e.g., via parameter Req, Start, etc.), and which supply an execution acknowledgment as an output parameter (e.g., via parameter Done, NDR, Error, etc.). 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.
This control mechanism may be positioned 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, flags, data variables and timerelated advantages over data variables).
The trigger parameter must be a global variable (e.g., flag, data variable).
In OB 100, jobs still activated by the user program (Parameter Req,
Start, etc.:= TRUE) must be set to zero at the named parameters. 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 can be traversed either from the PLC or from the NC.
- They can be started from a function call on the PLC in all operating modes.
- The start is independent of NC block boundaries.

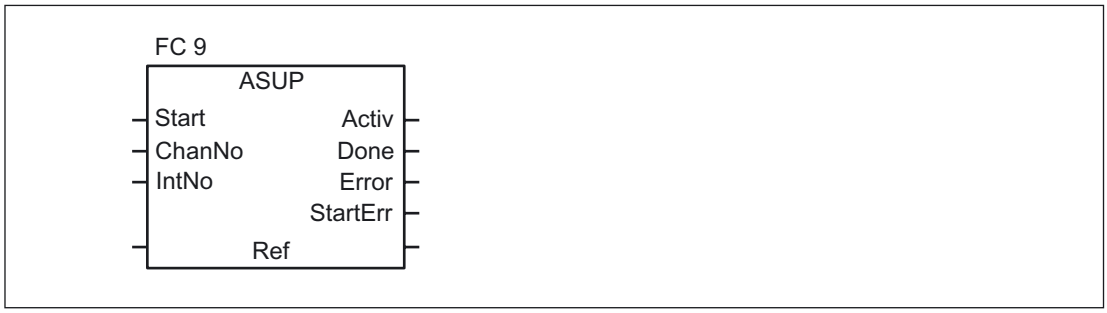
Function calls are available for positioning (FC 15) and indexing axes (FC 16).



ASUBs

Asynchronous subprograms (ASUBs) can be used to activate any selected function in the NC. 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). ASUBs can only be started in MDA or Automatic mode with **running** parts program.

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. The asynchronous subprogram is started by FC 9.



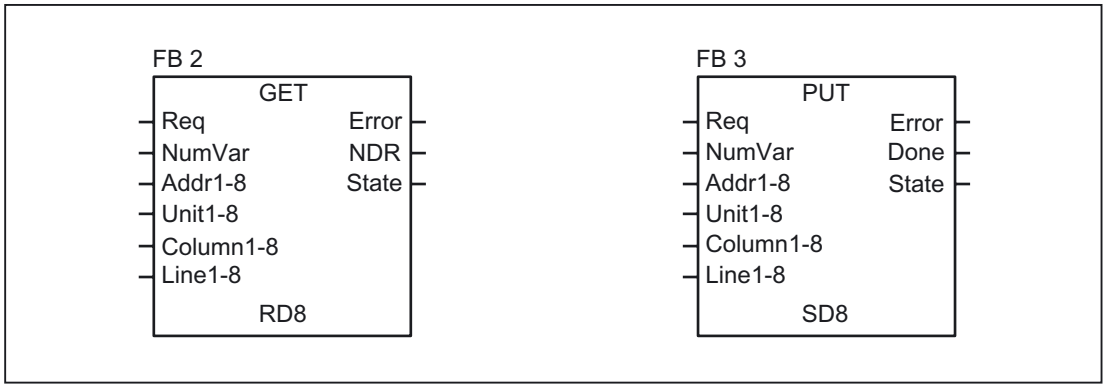
Note

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.

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, 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, projectrelated 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 bits) are converted to PLC format (32 bits with 24-bit mantissa and 8bit exponent) and vice versa). A loss of accuracy results from the conversion from 64 bits to 32bit REAL. The maximum precision of 32bit REAL numbers is approximately 10 to the power of 7.



2.7.7 Symbolic programming of user program with interface DB

General

Note

With basic program SW 3.2 and higher, files NST_UDT.STL and TM_UDT.AWL are stored on the basic program diskette supplied with the system.

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**	**Meaning**
UDT 2	DB 2	Alarms/Messages
UDT 10	DB 10	NCK signals
UDT 11	DB 11	Mode group signals
UDT 19	DB 19	MMC signals
UDT 21	DB 21 to DB 30	Channel signal
UDT 31	DB 31 to DB 61	Axis/spindle signals
UDT 71	DB 71	Tool management: Load/unload locations
UDT 72	DB 72	Tool management: Change in spindle
UDT 73	DB 73	Tool management: Change in revolver

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. To do so, however, a fully qualified instruction is needed for data access in the earlier program (e.g., "U DB31.DBX 60.0" - this command is converted to "AxisX.E_SpKA" when the symbols function is activated in the editor).

Description

Abbreviated symbolic names of the interface signals are defined in the two STL files NST_UDT.AWL and TM_UDT.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 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

English versions of the UDT interfaces are available in SW 4.4 and higher under NST_UDTB.AWL and TM_UDTB.AWL. Only the English versions will continue to be developed in future versions.

Proceed as follows to convert the German language symbols used thus far into English language symbols:

- If sources are not being used, only the NST_UDTB.AWL and TM_UDTB.AWL blocks must be translated. The new symbols can then be seen immediately.
- If you are using sources, the sources must first be compiled with the previous UDTs (NST_UDT.AWL and TM_UDT.AWL). It is then necessary to compile NST_UDTB.AWL and TM_UDTB.AWL. After this step, you need to initiate a reverse compilation into the sources in accordance with the previous source setup. The interface symbols are then permanently converted to English.

2.7.8 M decoding acc. to list

Description of Functions

When the **M decoding according to list** function is activated via the BP parameter of FB1 "ListMDecGrp", up to 256 M functions with extended address can be decoded by the basic program.

The assignment between 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 crosschannel 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 NC 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**:

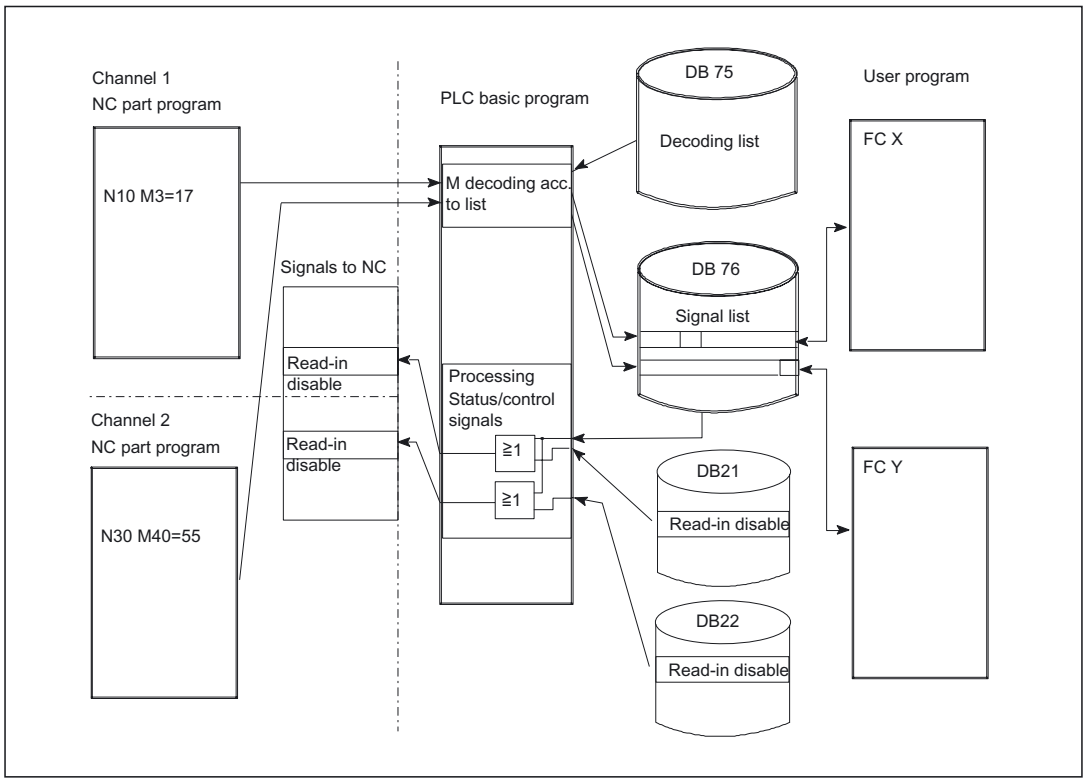


Fig. 2-13 M decoding acc. to list

Structure of decoding list

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.

The source file for the decoding list (MDECLIST:AWL) is supplied with the basic program.

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**	**Remarks**
MExtAdr	Int	Up to 99	Extended M address
MFirstAdr	DInt	Up to 99,999,999	First M address in group
MLastAdr	Dint	Up to 99,999,999	Last M address in group

Example

```
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 := 0;
      MSigGrp[1].MFirstAdr := L#0;
      MSigGrp[1].MLastAdr := L#0;
      MSigGrp[2].MExtAdr := 0;
      MSigGrp[2].MFirstAdr := L#0;
      MSigGrp[2].MLastAdr := L#0;
      MSigGrp[15].MExtAdr := 0;
      MSigGrp[15].MFirstAdr := L#0;
      MSigGrp[15].MLastAdr := L#0;
      MSigGrp[16].MExtAdr := 0;
      MSigGrp[16].MFirstAdr := L#0;
      MSigGrp[16].MLastAdr := L#0;
END_DATA_BLOCK
```

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.

Activation of the function

The number of groups to be evaluated is specified in basic-program parameter ListMGDecGrp.

The decoding function is activated if this value is between 1 and 16.

Before activation of decoding, the decoding list DB75 must be transferred to the PLC followed by a restart.

Example

The table below contains the parameters for the following programming example:

Example parameters				
Group	**Decoding list (DB 75)**			**Signal list**
	Extended M address	First M address	Last M address in group	DB 76
1	2	1	5	DBX0.0 to DBX0.4
2	3	12	23	DBX2.0 to DBX3.3
3	40	55	55	DBX4.0

```
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
```

2.7.9 PLC machine data

General

The user has the option of storing PLCspecific machine data in the NC. These data can then be processed while the PLC is running up (OB 100). This enables, for example, user options, machine expansion levels, machine configurations, etc., to be implemented.

The interface to read these data is stored in the DB 20. However, DB20 is set up by the basic program during powerup only when user machine data are used, i.e., sum of BP parameters UDInt, UDHex and UDReal is greater than zero.

The size of the individual ranges and, therefore, the total length of DB 20,
is set using PLC machine data
MD14504 MAXNUM_USER_DATA_INT,
MD14506 MAXNUM_USER_DATA_HEX,
MD14508 MAXNUM_USER_DATA_FLOAT
and specified for the user in BP parameters
UDInt, UDHex and UDReal.

The basic program stores the data in DB 20 in the following order:
Integer MD, Hex field MD, Real MD.

The integer and real values are stored in DB 20 in S7 format.
Hexadecimal data are stored in DB20 in the order in which they are input (use as bit fields).

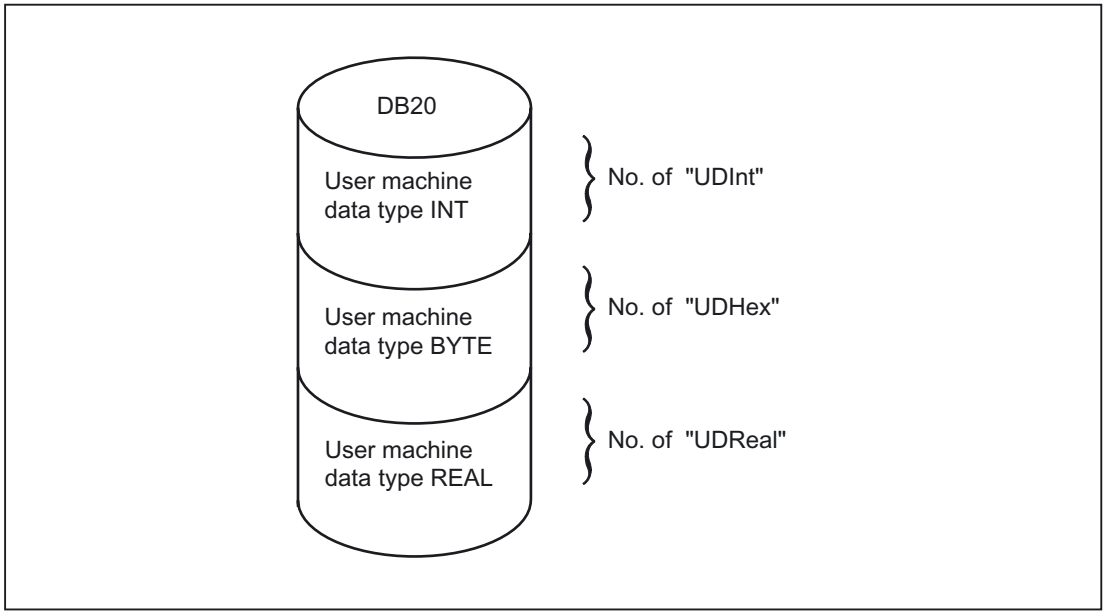


Fig. 2-14 DB 20

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.

Interrupts	
400120	**Delete DB 20 in PLC and restart**
Explanation	DB length is not the same as the required DB length
Response	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 USER_DATA_INT[0]                    123
MD14510 USER_DATA_INT[1]                    456
MD14510 USER_DATA_INT[2]                    789
MD14510 USER_DATA_INT[3]                    1011
...
MD14512 USER_DATA_HEX[0]                    12
MD14512 USER_DATA_HEX[1]                    AC
...
MD14514 USER_DATA_FLOAT[0]                  123.456
BP parameters (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 run-up, DB20 was generated with a length of 28 bytes:

DB 20

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	e+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.

An entry is made in the symbol table to allow data access in symbolic form:

Interrupts

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];
          V    "UData".UDHex0[1];
          V    "UData".UDHex0[2];
          V    "UData".UDHex0[3];
          V    "UData".UDHex0[4];
          V    "UData".UDHex0[5];
          V    "UData".UDHex0[6];
          V    "UData".UDHex0[7];
          ..   ...
          .
          U    "UData".UDHex0[15];

          L    "UData".UDReal[0];
...
```

2.7.10 Configuration of machine control panel, handheld unit

General

The communications system integrated in the NC permits a maximum of 2 machine control panels and one handheld unit to exchange data with the 810D and the 840D. An SDB 210 is not required to transfer the signals of these components. The information given below is based on the assumption that no SDB 210 has been installed for the components concerned.

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 and output addresses are also used in FC 19, FC 24, FC 25, FC 26 and FC 13. Addresses for status information, MPI or OPI (a GD parameter set must be set for the handheld unit rather than an MPI address) must also be defined. The default time settings for timeout and cyclic forced retriggering do not have to be changed.

Activation

Each component is activated either via the number of machine control panels (parameter MCPNum) or, in the case of the handheld unit, parameter BHG := 2 (BHG := 1 corresponds to a link via the MPI interface in conjunction with an SDB 210). Whether a component is to be linked to the OPI or the MPI is determined by parameters MCPMPI and BHGMPI.

Handheld unit

The handheld unit addresses the MPI or OPI by means of a GD parameter set. These parameter values must be assigned according to the handheld unit settings. However, the parameter names on the handheld unit are the reverse of the parameter names in the basic program. All Send type parameters on the handheld unit must be defined as Rec type (and Rec in Send type) in the basic program.

Control signals

Parameters MCP1Stop, MCP2Stop and BHGStop can be used to stop communication with individual components (parameter setting = 1). 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 first machine control panel:

```
SET;
S            gp_par.MCP1Stop;
```

Setting parameters MCP1Stop, MCP2Stop, BHGStop also results in a suppression of alarms 40260 to 40262.

MPI switchover, OPI address

An existing connection with an MCP (Machine Control Panel) or HHU (HandHeld Unit) can be aborted. Another MCP or HHU component already connected to the bus (with another MPI or OPI 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. After checkback in DB10 byte 104 (relevant bits 0, 1, 2 set to 0). Change in the bus address or GD parameter set of this unit to that of 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 in DB10 byte 104 (relevant bits 0, 1, 2 is set to 1).

As described in Section Control signals, all parameters must be programmed according to data type.

Switching off flashing MCP:

With MCP firmware V5.01.01 and higher, flashing can be suppressed in offline mode. No communication takes place in offline mode (e.g., if the MCP connection fails). During active communication, a value can be specified in the output data using status bits 24 or 25 in MCP1StatSend and MCP2StatSend.
The checkback signal transferred by the value is signaled back in the same bits in MCP1StatRec, MCP2StatRec. Following a successful checkback signal, the status bits in send status must be reset.
Implementation example is stored in the Toolbox.

Temperature monitoring of the MCP:

With MCP firmware V5.01.02 and higher, an increased temperature is signaled back via bit 28 = 1 in MCP1StatRec, MCP2StatRec.

Configuring

There are basically two communication mechanisms for transferring data between the MCP/HHU and PLC. With the first mechanism, data are transported via the COM module (840D/810D). The mechanism is parameterized completely via the MCP/HHU parameters in FB1.

An overview of the various interfacing options as a function of the NC type used is given below. In each case, the parameter set of FB1 and the valid status information relevant for the respective data transmission method are specified.

If an error is detected due to a timeout monitor, a corresponding entry is made in the diagnostic buffer of the PLC CPU (errors 400260 to 400262). In this case, the input signals from the MCP or from the handheld unit (MCP1In/MCP2In or BHGIn) are initialized with 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message reset by the BP.

840D: OPI/MPI connection

Communication starts from the PLC BP via the NCK and COM mode, i.e., even a link via the MPI does not require an SDB210. Parameter settings are made via the relevant parameters in FB1.

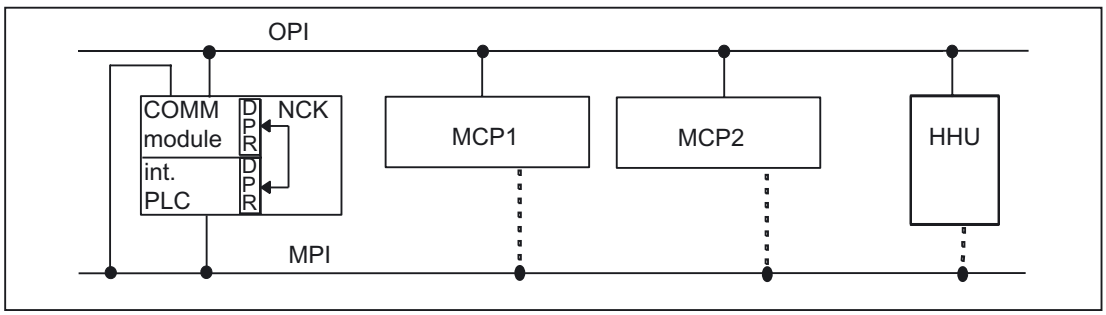


Fig. 2-15 840D: OPI/MPI connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum=1 or 2 (number of MCPs)		BHG=2 (transfer via COM module)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend	MCP2StatSend	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl	MCP2Cycl	BHGTimeout
MCPMPI = FALSE (OPI), TRUE (MPI)		BHGCycl
MCP1Stop	MCP2Stop	BHGRecGDNo
		BHGRecGBZNo
		BHGRecObjNo
		BHGSendGDNo
		BHGSendGBZNo
		BHGSendObjNo
		BHGMPI = FALSE (OPI), TRUE (MPI)
		BHGStop

Status information		
Available in	**Bit No.**	**Description**
MCP1StatSend MCP2StatSend BHGStatSend	4	Syntax error in GD package: Error in parameter set (FB1)
MCP1StatSend MCP2StatSend BHGStatSend	27	Transmitter: Timeout
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Timeout

An error entry is also made in the PLC diagnostic buffer for timeouts (bits 10 and 27), resulting in the following error messages on the MMC:

- 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 have yet been exchanged between the MCP/HHU and PLC.
The monitoring function is activated as soon as all components have signaled "Ready" after powerup.

840D: MPI connection for HHU (not for new development)

Communications for HHU by PLC operating system and parameterization via SDB210.

Communication for the MCP is controlled from the PLC BP via the NCK and COM module as described above.

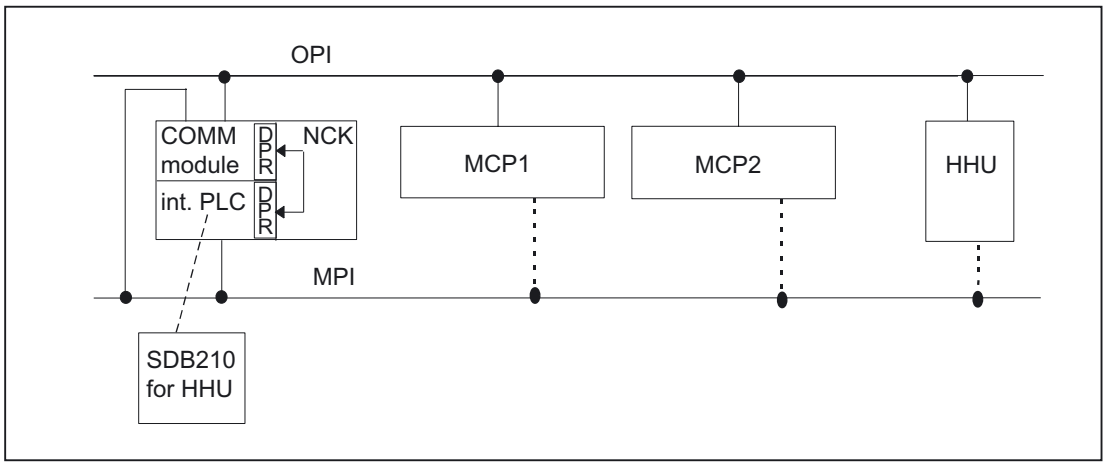


Fig. 2-16 840D: MPI connection for HHU

Relevant parameters (FB1):

Communication between the PLC and HHU is implemented through configuring and subsequent loading of SDB210 (Global Data). To allow the basic program to access the HHU data and implement HHU failure monitoring, the addresses set via global data must be declared in FB1 parameters.

Relevant parameters (FB1)		
MCP		**HHU (parameterization via SDB210)**
Parameter settings are made via the relevant parameters in FB1.		BHG = 1 (transfer via SDB210)
		BHGIn (as parameterized in SDB210)
		BHGOut (as parameterized in SDB210)
		BHGStatRec (as parameterized in SDB210)
		BHGTimeout (as parameterized in SDB210)

Status information (MCP1 and MCP2 see table Status information)		
Available in	**Bit No.**	**Description**
BHGStatRec	10	Receiver: Timeout

An error entry is also made in the PLC diagnostic buffer for timeouts,

resulting in the following error message on the MMC:

- 400262: HHU failure

An HHU failure is only recognized if data exchange has taken place previously with the HHU. The first exchange of data with the HHU activates the monitoring function.

MPI connection

Communication starts from the PLC BP via the NCK and COM mode, i.e., even a link via the MPI does not require an SDB210. Parameter settings are made via the relevant parameters in FB1.

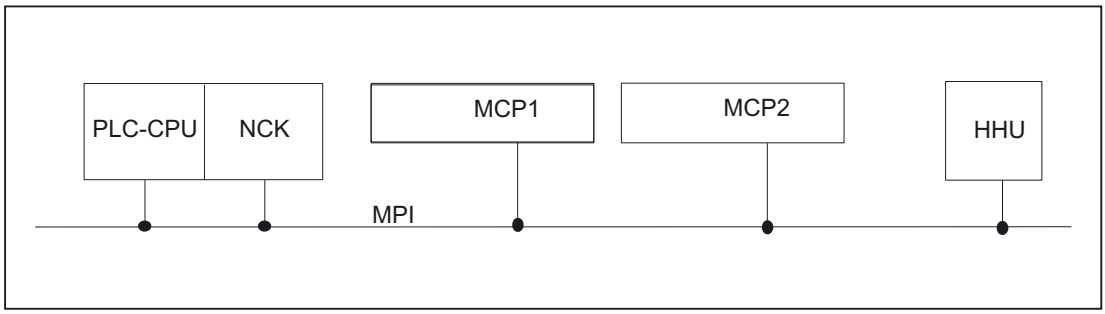


Fig. 2-17 MPI connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum=1 or 2 (number of MCPs)		BHG=2 (transfer via COM module)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend	MCP2StatSend	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl	MCP2Cycl	BHGTimeout
MCPMPI = TRUE (MPI)		BHGCycl
MCP1Stop	MCP2Stop	BHGRecGDNo
		BHGRecGBZNo
		BHGRecObjNo
		BHGSendGDNo
		BHGSendGBZNo
		BHGSendObjNo
		BHGMPI = TRUE (MPI)
		BHGStop

Status information		
Available in	**Bit No.**	**Description**
MCP1StatSendMCP2Stat Send BHGStatSend	4	Syntax error in GD package: Error in parameter set (FB1)
MCP1StatSendMCP2Stat SendB HGStatSend	27	Transmitter: Timeout
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Timeout

An error entry is also made in the PLC diagnostic buffer for timeouts (bits 10 and 27),

resulting in the following error messages on the MMC:

- 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 have yet been exchanged between the MCP/HHU and PLC.
The monitoring function is activated as soon as all components have signaled "Ready" after powerup.

MPI connection and 810D (SW 4 and lower)

Communications for MCP and HHU by PLC operating system and parameterization via SDB210.

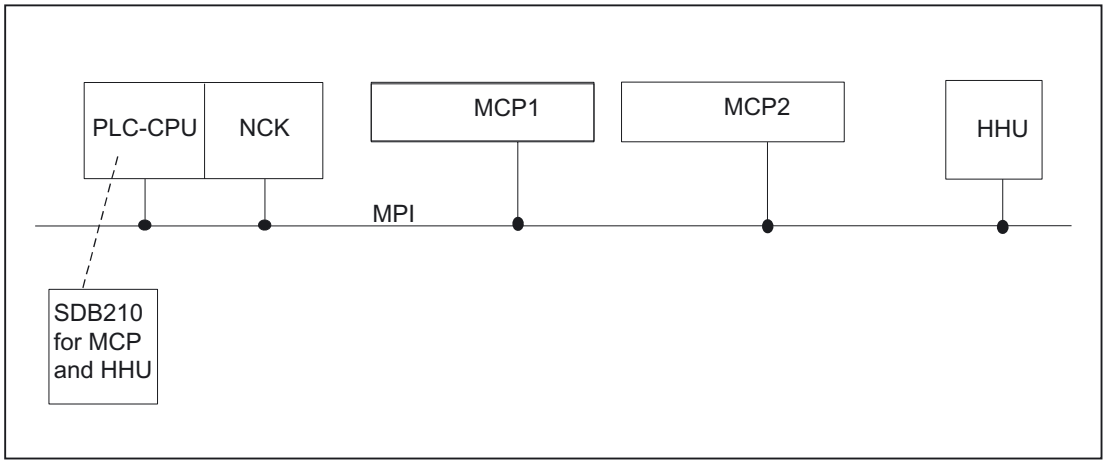


Fig. 2-18 MPI connection and 810D

Relevant parameters (FB1):

Communication between the PLC and HHU is implemented through configuring and subsequent loading of SDB210 (Global Data). To allow the basic program to access the HHU data and implement HHU failure monitoring, the addresses set via global data must be declared in FB1 parameters.

Relevant parameters (FB1) (all entries as parameterized in SDB210 global data)		
MCP		**HHU**
MCPNum=1 or 2 (number of MCPs)		BHG=1 (MPI)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1Timeout	MCP2Timeout	BHGTimeout

Status information		
Available in	**Bit No.**	**Description**
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Time out

An error entry is also made in the PLC diagnostic buffer for timeouts,

resulting in the following error messages on the MMC:

- 400260: MCP 1 failure

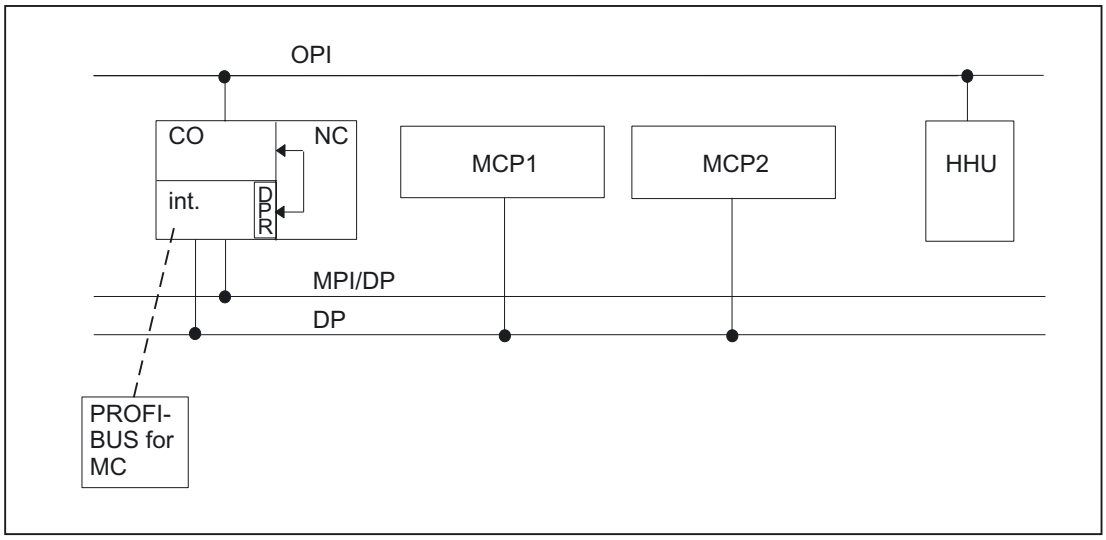
  Or

- 400261: MCP 2 failure
- 400262: HHU failure

An MCP/HHU failure is only detected following a cold restart if the unit has already been involved in data exchange. The first exchange of data with the MCP/HHU activates the monitoring function.

840D: PROFIBUS connection

With an MCP PROFIBUS connection, these components must be considered in the STEP 7 hardware configuration. 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. PROFIBUS slave address can be stored in MCP1BusAdr and MCPBusAdr.



Relevant parameters (FB1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		HHU = 2 (via COM module)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend (n.r.)	MCP2StatSend (n.r.)	BHGStatSend
MCP1StatRec (n.r.)	MCP2StatRec (n.r.)	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout (n.r.)	MCP2Timeout (n.r.)	BHGOutLen
MCP1Cycl (n.r.)	MCP2Cycl (n.r.)	BHGTimeout
MCPMPI = FALSE		BHGCycl
MCP1Stop (n.r.)	MCP2Stop (n.r.)	BHGRecGDNo
MCPBusType = 3		BHGRecGBZNo
		BHGRecObjNo
		BHGSendGDNo
		BHGSendGBZNo
		BHGSendObjNo
		BHGMPI = FALSE
		BHGStop

MCP failure switches the PLC to the STOP state. If this is undesirable, OB 82, OB 86 can be used to avoid a stop. The MCP (as PROFIBUS slave) can also be switched on and off via SFC12.
In the event of a failure, alarm messages are not generated by the basic program.
M to N is not possible with the MCP variant.

2.8 SPL for Safety Integrated

See:
References:
/FBSI/ Description of Functions, Safety Integrated

2.9 Assignment overview

2.9.1 Assignment: NC/PLC interface

The assignment of the NC/PLC interface is comprehensively described in:

References:
/LIS/ Lists

2.9.2 Assignment: FB/FC

FB number	FC number	Meaning
1		Basic program
2 - 29		Reserved for Siemens
	1	Initialization of basic program
	2 - 29	Reserved for Siemens
	30 - 35	See below: ShopMill, ManualTurn
	30 – 127 1)	**User area**
30 – 127 1)		**User area**

1)
The actual upper limit of the block number (FB/FC) depends on the current PLC CPU. See section "Key PLC CPU data for 810D, 840D". For FC and FB assignment, see section "Memory requirements of basic PLC program for 810D, 840D".

ManualTurn

ManualTurn uses FC 30 to 35 and DB 81 to 89.

ManualTurn is an operating system for conventional, cyclecontrolled turning machines. The above FCs and DBs can be used provided that the machine to be configured is not a turning machine with a maximum of two axes and one spindle. If your machine is of this type and if you might require conventional operating methods in addition to CNC functions, then you should not use the FCs and DBs above.

ShopMill

ShopMill uses FC30 to 35 and DB 81 to 89.

ShopMill is an operating system for 2 1/2D milling machines in the workshop environment. The above FCs and DBs can be used provided that the machine to be configured is not a milling machine for 2 1/2D machining operations. However, if you intend to use your machine for applications of this type, then the FCs and DBs should not be used.

2.9.3 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.	**Designation**	**Name**	**Package**
1		Reserved for Siemens	BP
4	PLC-MELD	PLC messages	BP
8		Reserved for Siemens	
9	NC-COMPILE	Interface for NC compile cycles	BP
10	NC INTERFACE	Central NC interface	BP
11	Mode group 1	Interface mode group	BP
12		Computer link and transport system interface	
14		Reserved for Siemens	
15		Basic program	
16		PI Service definition	
17		Version ID	
18		Reserved for basic program	

Overview of data blocks			
DB no.	**Designation**	**Name**	**Package**
19		MMC 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 - 80		Reserved for Siemens	
81 - 89		See below: ShopMill, ManualTurn	
81 –127 [1)]		**user area**	
1) The actual upper limit of the block number (DB) depends on the current PLC CPU. See section "Key PLC CPU data for 810D, 840D".			

Note

The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as required by the user.

ManualTurn

ManualTurn uses FC 30 to 35 and DB 81 to 89.

ManualTurn is an operating system for conventional, cyclecontrolled turning machines. The above FCs and DBs can be used provided that the machine to be configured is not a turning machine with a maximum of two axes and one spindle. If your machine is of this type and if you might require conventional operating methods in addition to CNC functions, then you should not use the FCs and DBs above.

ShopMill

ShopMill uses FC30 to 35 and DB 81 to 89.

ShopMill is an operating system for 2 1/2D milling machines in the workshop environment. The above FCs and DBs can be used provided that the machine to be configured is not a milling machine for 2 1/2D machining operations. However, if you intend to use your machine for applications of this type, then the FCs and DBs should not be used.

2.9.4 Assignment: Timers

Timer No.	Meaning
0 - 9	Reserved for Siemens
10 – 127 1)	User area
1) The actual upper limit of the block number (timer) depends on the current PLC CPU. See section "Key PLC CPU data for 810D, 840D".	

2.10 Memory requirements of basic PLC program for 810D, 840D

General

The basic program consists of basic and optional functions. The **basic functions** include cyclic signal exchange between the NC and PLC.
The **options** include, for example, the FCs, which can be used if required.

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 with SINUMERIK 840D				
Block type no.	Function	Remark	Block size (bytes)	
			Load memory	Working memory
Basic functions in basic program				
FB 1, 15, 16, 17, 18		Must be loaded	3616	3052
FC 1, 2, 3, 4, 11, 20		Must be loaded	7208	6608
DB 4, 5, 7, 8, 17, 19		Must be loaded	2490	966
DB 2, 3, 6		Are generated by the BP	992	812
OB 1, 40, 100		Must be loaded	490	282
		Total	14796	11720

PLC/NCK, PLC/MMC interface				
DB 10	PLC/NCK signals	Must be loaded	318	262
DB 11	Signals PLC/Mode group	Is generated by BP	80	44
DB 21, 30	PLC/channel signals	Are generated by BP as a function of NCMD	352 each	316 each
DB 31, ...61	PLC/axis or spindle signals	Are generated by BP as a function of NCMD	180 each	144 each

Basic program options				
	Machine control panel			
FC 19	Transfer of MCP signals, M variant	Must be loaded when M variant of MCP is installed	1498	1258
FC 25	Transfer of MCP signals, T variant	Must be loaded when T variant of MCP is installed	1358	1160
FC 24	Transfer of MCP signals, slim variant	Must be loaded when slim variant of MCP is installed	1358	1160
FC 26	Transfer of MCP signals, HPU variant	Must be loaded for HPUs	1358	1160
FC 14	MPI/OPI transfer	Must be loaded if MCPNum > 0	942	802
	Handheld unit			
FC 13	Display control HHU	Can be loaded for handheld units	1264	1044
	Error/operating messages			
FC 10	Acquisition FM/BM	Load when FM/BM is used	1572	1350
	ASUB			
FC 9	ASUB start	Load when PLC ASUBs are used	656	538

Basic program options				
	Concurrent axes			
FC 15	Positioning of linear/rotary axes	Load for axis positioning by PLC	656	546
FC 16	Positioning of indexing axes	Load for axis positioning by PLC	674	560
	Star/delta changeover			
FC 17	Star/delta switchover of MSD	Load for star/delta switchover	612	494
	Spindle control			
FC 18	Spindle control	Load for spindle control from PLC	826	676
	PLC/NC communication			
FB 2	Read NC variable	Load for Read NC variable	396	224
DB n1)	Read NC variable	One instance DB per FB 2 call	426 each	270 each
FB 3	Write NC variable	Load for Write NC variable	396	224
DB m1)	Write NC variable	One instance DB per FB 3 call	426 each	270 each
FB 4	PI services	Load for PI services	334	214

Basic program options				
DB o1)	PI services	One instance DB per FB 4 call	234 each	130 each
DB 16	PI services description	Load for PI services	1190	408
FB 5	Read GUD variables	Load for PI services	532	365
DB p	Read GUD variables	One instance DB per FB 5 call	308 each	166 each
FB 6	General communication	Load for Read/write NC variables and PI services	5986	5228
DB 15	General communication	Instance DB for FB 6	440	172
Tool management				
FC 6	Basic function	Load for tool management option	1382	1182
FC 7	Transfer function turret	Load for tool management option	530	430
FC 8	Transfer function	Load for tool management option	1002	834
FC 22	Direction selection	Load if direction selection is required	404	300
DB 71	Loading locations	Generated by BP as a function of NC MD	30*B	30*B
DB 72	Spindles	Generated by BP as a function of NC MD	48*Sp	48*Sp
DB 73	Turret	Generated by BP as a function of NC MD	44*R	44*R
DB 74	Basic function	Generated by BP as a function of NC MD	(B+ Sp+R)*20	(B+ Sp+R)*20
Compile cycles				
DB 9	Interface PLC compile cycles	Is generated by BP as a function of NC option	472	436
1): DB number must be specified by PLC user				

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)	
			Load memory	Working memory
Minimum configuration (1 spindle, 2 axes and TMCP)				
See above	Basic program, base		14796	11720
	Interface DBs		1290	1054
	MCP		2300	1962
		Total	18386	14736

Block type no.	Function	Remark	Block size (bytes)	
			Load memory	Working memory
Maximum configuration (2 channels, 4 spindles, 4 axes, TMCP)				
See above	Basic program, base		14796	11720
See above	Interface DBs		2542	2090
See above	MCP		2300	1962
See above	Error/status messages		1572	1350
See above	ASUBs	1 ASUB initiation	656	538
See above	Concurrent axis	For 2 turrets	674	560
See above	PLC/NC communication	1 x read variable and 1 x write variable	8070	6388
See above	Tool management	2 turrets with one loading point each	3430	2854
See above	Compile cycles		472	436
		Total	34512	27898

2.11 Supplementary conditions and NC VAR selector

2.11.1 Supplementary conditions

2.11.1.1 Programming and parameterizing tools

Hardware

Programming devices or PCs with the following equipment are required for the PLCs installed on the 810D and 840D:

	Minimum	Recommendation
Processor	80486	Pentium
RAM (MB)	32	Or more
Hard disk, free capacity (MB)	200	> 400
Interfaces	MPI incl. cable Memory card	
Graphics	VGA or TIGA	SVGA
Mouse	yes	
Operating system	Windows 95/98/NT STEP 7 and higher Version 4	Windows 95/98/NT or higher STEP 7 and higher Version 5.1

The **STEP7 package for the S7300** can be obtained and 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
  - **Communication Configuration** configuration 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
  - PLC STOP/complete restart/overall reset triggering from the programming device
  - 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/C/B/P/D
- Archiving of utility routines
  - Allocation of the output statuses 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 configuring modules
    (e.g., CP3425 -> NCM package)

Note

More information about possible functions can be found in SIMATIC catalogs and STEP 7 documentation.

2.11.1.2 SIMATIC documentation required

References:
SIMATIC S 7 System Overview
S7-300, CPU 314, CPU 315-2DP Operation List
Programming with STEP 7
STEP 7 User's Guide
STEP 7 Programming Manual; Designing User Programs
STEP 7 Reference Manual; STL Instruction List
STEP 7 Reference Manual; LAD Ladder Diagram
STEP 7 Reference Manual; Standard and System Functions
STEP 7 Manual: Conversion of STEP 5 Programs
STEP 7 General Index
CPU 314, CPU 315-2DP Manual

2.11.1.3 Relevant SINUMERIK documents

References:
/IAD/ 840D, 611D Installation and Startup Guide; PLC Interface
/IAG/ 810D, 611D Installation and Startup Guide; PLC Interface
/BH/ Operator Components Manual (HW)/840D/FM NC/810D
/FB/ 840D, 810D Description of Functions
/LIS/ 840D/810 D Lists
/FBP/ PLC C Programming

2.11.2 NC VAR selector

2.11.2.1 Overview

General

A catalog with an optional catalog name must be set up via the Windows Explorer. The selected data of the VAR selector (data.VAR and data.AWL (STL)) must be stored in this catalog. An "Insert", "External source" into the STEP 7 machine project for the "Data.AWL" file must then be carried out via the STEP 7 manager (V3 and higher). 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.

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".

Note

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 version, i.e., selected variables in an older software version must not be reselected when the software is upgraded.

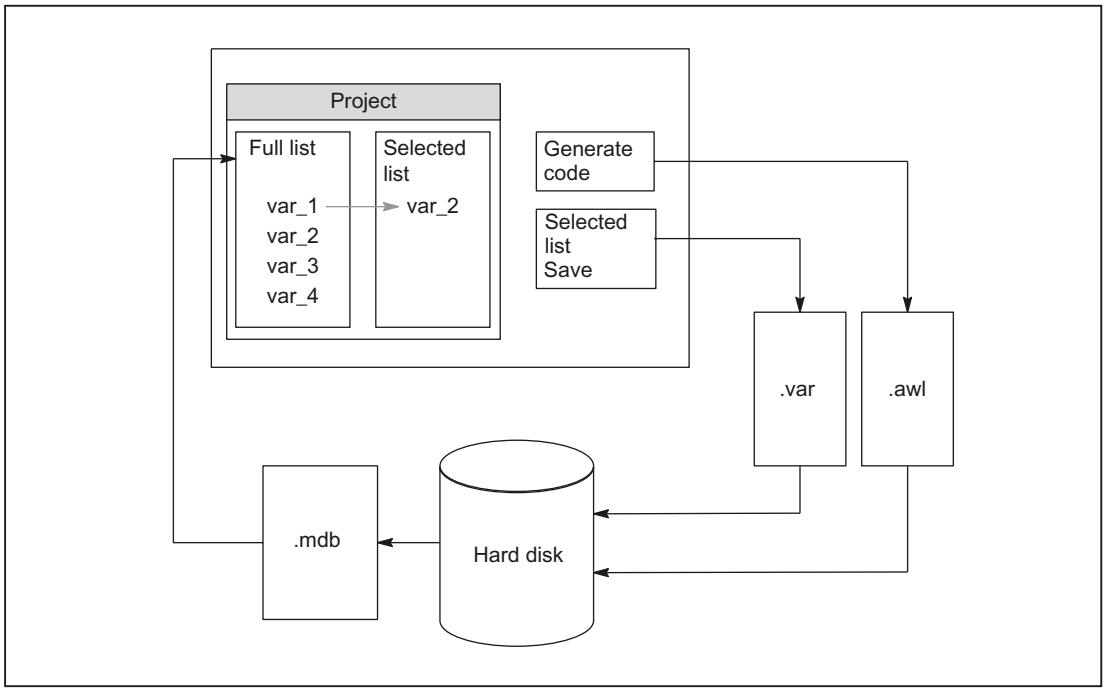


Fig. 2-19 NC VAR selector

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.

In SW V6.3 and higher, the variable lists ncv*.mdb are separated according to:	
NC variables including machine and setting data:	ncv_NcData.mdb
Machine data for 611D drive:	ncv_611d.mdb
Machine data for 611D linear drive:	ncv_611dLinear.mdb
Machine data of the 611D drive, Performance 2:	ncv_611d_P2.mdb
Machine data of the 611D linear drive, Performance 2:	ncv_611d_P2Linear.mdb
Machine data of the hydraulic drive:	ncv_Hydraulics.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.

Once he has generated 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 not therefore necessary to install different versions of the "NC VAR selector" in parallel.

System features, supplementary conditions

The PC application "NC VAR selector" requires Windows 95 (or later operating system).

The assignment of names to variables is described in:
References:
/LIS/ Lists; Chapter: Variables,
or in the Variables Help file (integrated in NC VAR selector).

2.11.2.2 Description of functions

Overview

The figure below illustrates how the NC VAR selector is used within the STEP 7 environment.

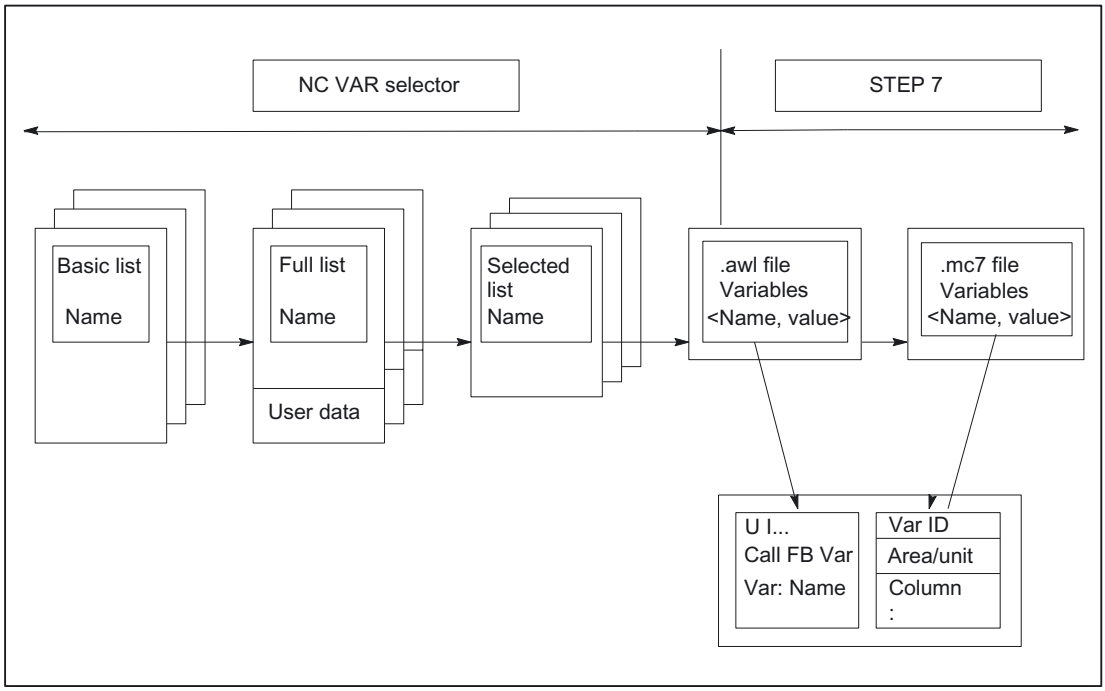


Fig. 2-20 Application of NC VAR selector in the STEP 7 environment

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.

Note

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 machinespecific 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.

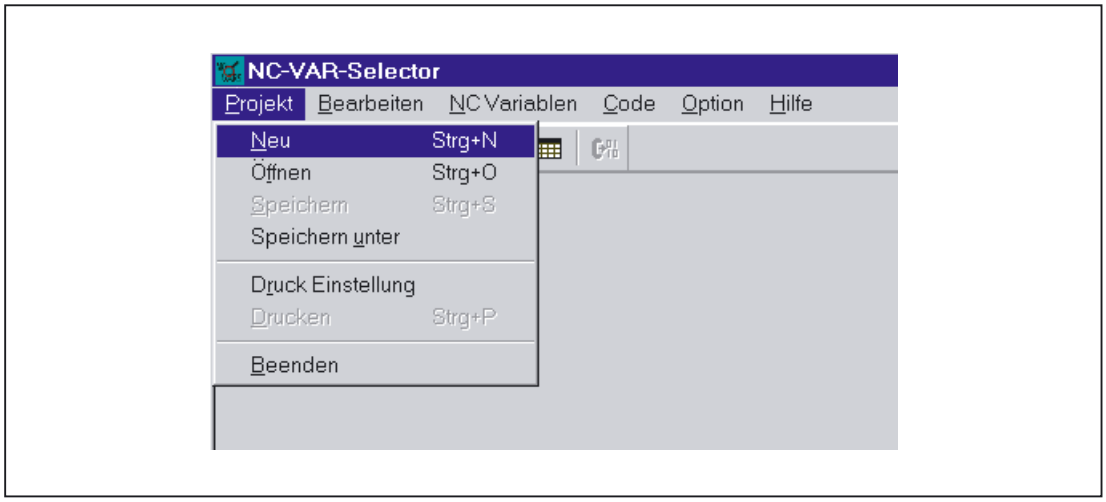


Fig. 2-21 Basic display with basic menu

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).

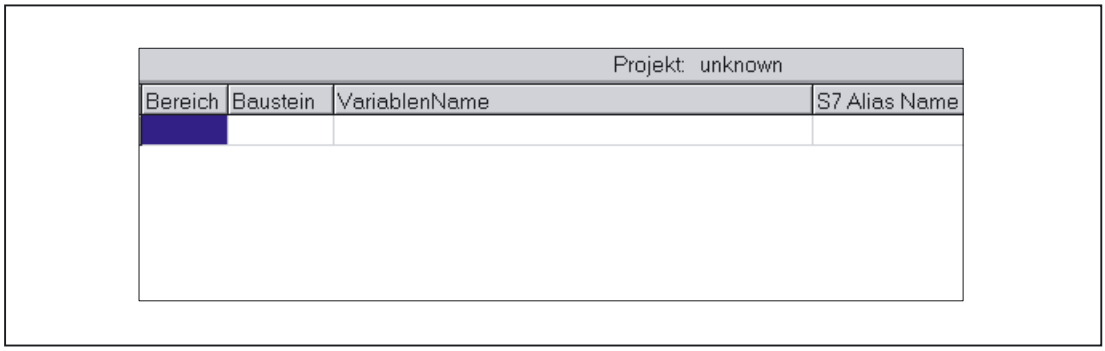


Fig. 2-22 Window with selected variables for new project

The selected variables are displayed in a window.

Opening an 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.

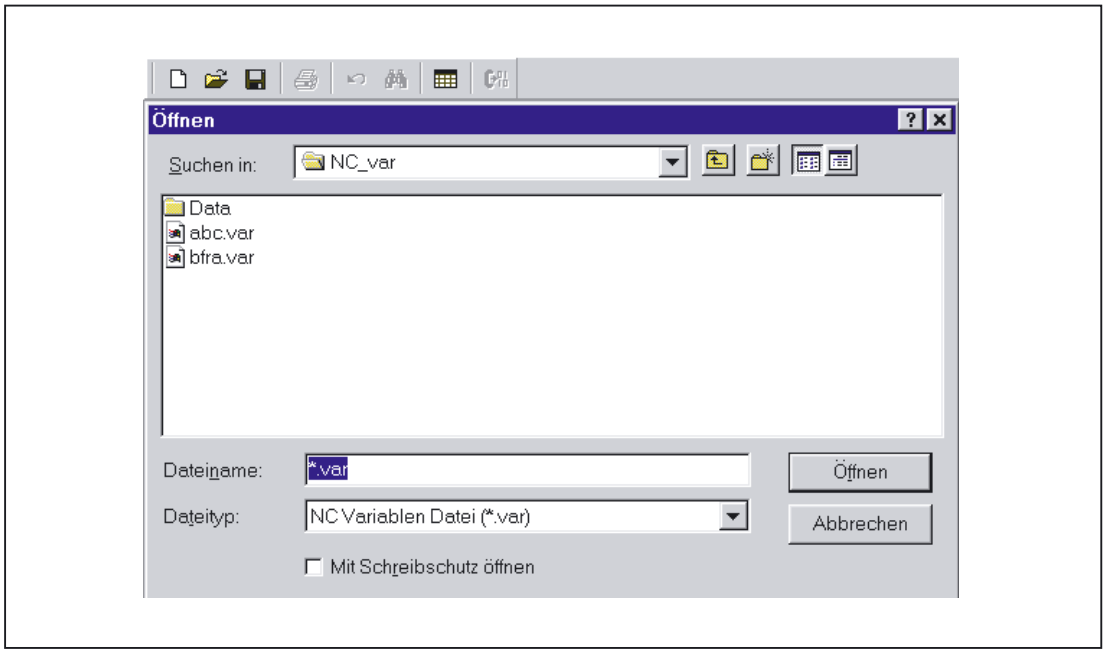


Fig. 2-23 Selection window for existing projects

If, after selection of the project, new variables are to be added, a complete list of NCK variables must be selected (see: Selecting complete list). 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 the NC Var selector path Data\Swxy (xy stands for SW no., e.g., SW 5.3:=xy=53). This list can be selected as an NC variables list. The available variable lists are provided thematically.

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.

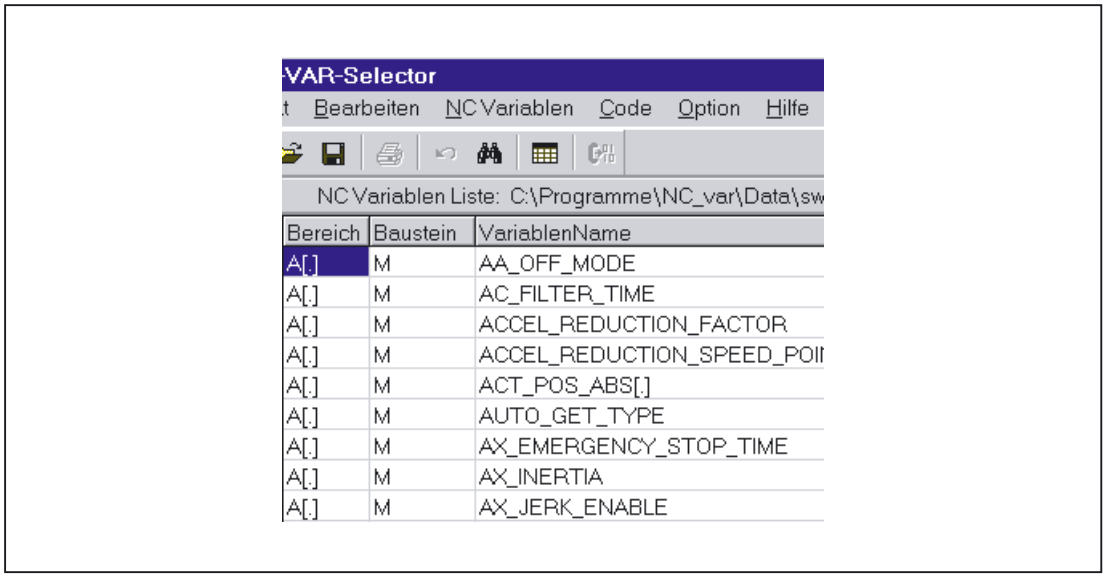


Fig. 2-24 Window with selected Complete List

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.

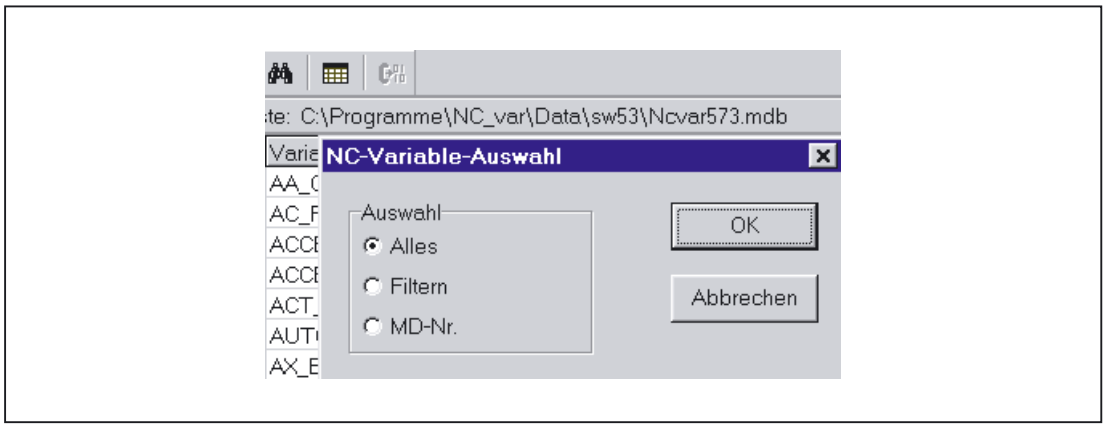


Fig. 2-25 Window with filter criteria for displaying list of variables

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

Selecting variables

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 doubleclicking on the S7 variable name field. This action can also be undone under the "Edit" menu item.

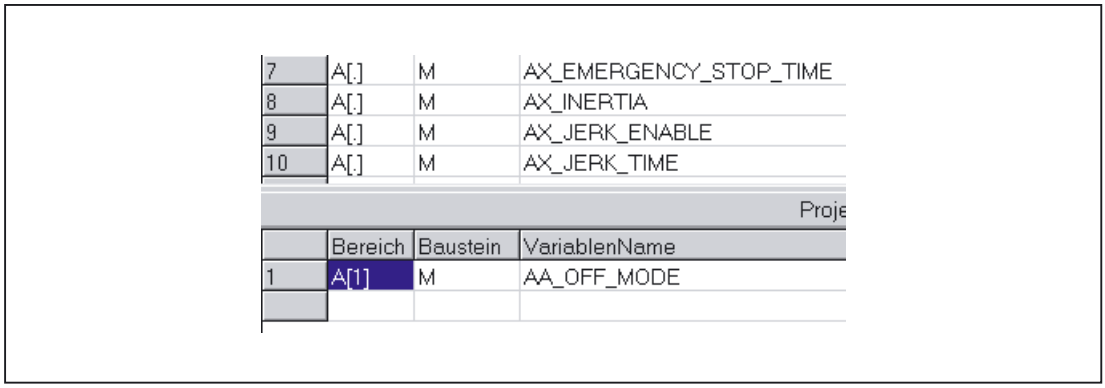


Fig. 2-26 Screen with complete list and selected variables

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 multidimensional 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:
/LIS/ **Lists;** 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.

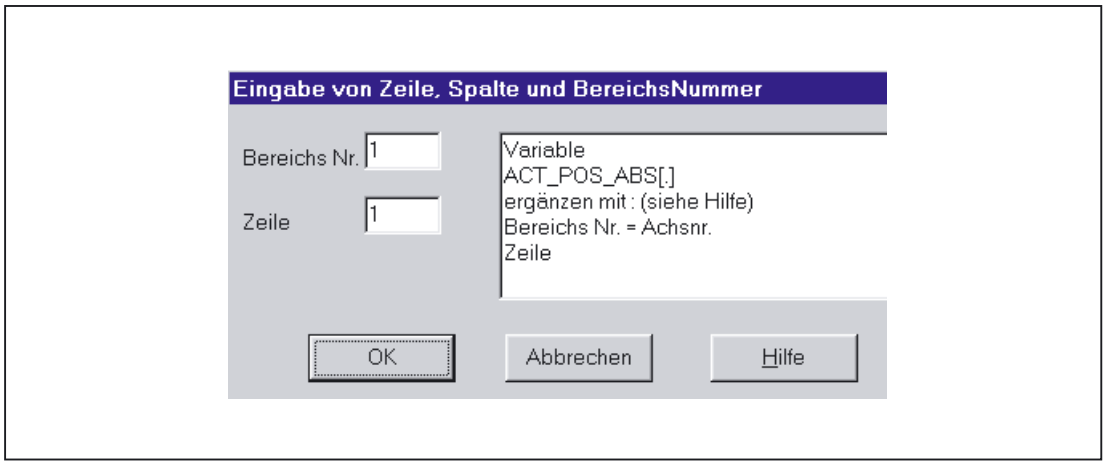


Fig. 2-27 Entry field for line, column and block no.

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 doubleclick function. It is possible to select several variables for deletion (see "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 projectspecific 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.

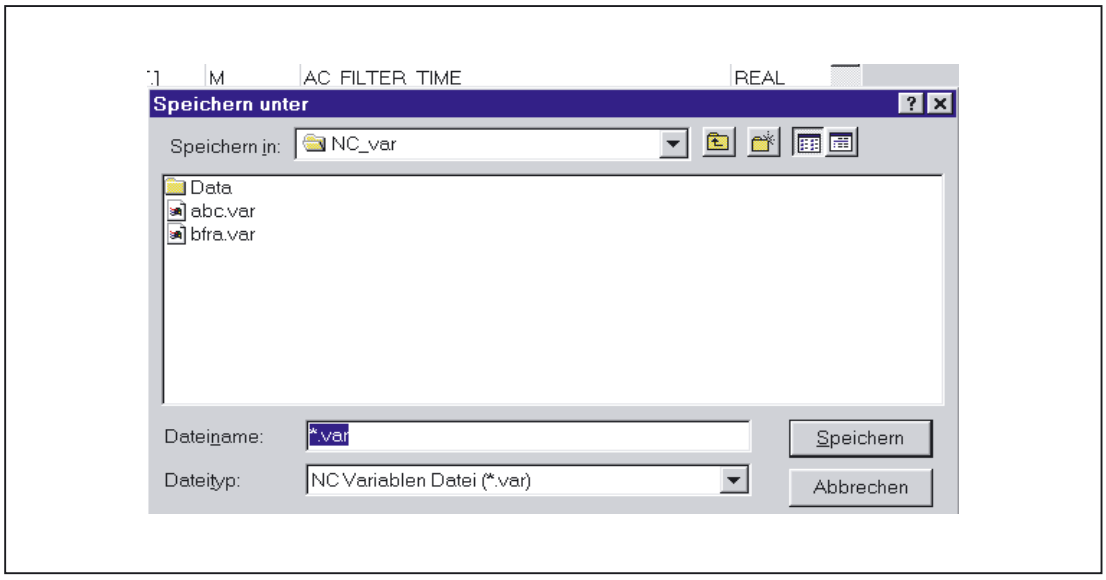


Fig. 2-28 Window for project path and name of file to be stored

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 "Unit 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 a STEP7 project

The generated AWL file is transferred to a selectable SIMATIC project (program path) and compiled. Furthermore, the symbol can also be transferred. This function will be available as of STEP7 Version 5.1 and NCVar Selector 6.04.05. This process take some time due to the time require by STEP7. Before transferring a new AWL file, the file window of the AWL file must be closed in the LAD/FUP/AWL 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 Operator's Guide
- The Description of Variables

The copyright and the version number can also be displayed.

2.11.2.3 Startup, installation

The Windows application "NC Var selector" is installed using the SETUP program supplied with the package.

2.12 Block descriptions

2.12.1 FB 1: RUN_UP Basic program, startup section

Description of Functions

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 startup of the control, it is vital to synchronize the NCK and PLC, as these systems have their own types of powerup 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 PLC 314 and PLC 315-2DP only know the start-up type cold restart. 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

An additional SDB 210 must be generated via the STEP7 Communication Configuration application for the **operator components** connected on the **MPI interface**.
The corresponding procedure is explained in the Installation and Startup Guide.



Caution

The text below is only relevant if an MCP1In, MCP1Out, MCP1StatSend, or MCP1StatRec pointer parameter (or the parameters for the second MCP and the HHU) is to be assigned to a data block (these pointers are usually stored on inputs or outputs).

To assign the MCP1In, MCP1Out, MCP1StatSend, or MCP1StatRec pointer parameter (or the parameters for the second MCP and the HHU) to data block elements:

The In parameter (e.g., MCP1In) for this operator component **must** point to a data block.

The parameterized DB number must be the same for the other parameters if the other pointers point to data blocks (the other parameters can point to inputs, outputs or flags).

All pointers of an operator component do not have to point to data blocks.

Declaration 810D, 840D

```
Code                                          Comment
FUNCTION_BLOCK FB 1
VAR_INPUT
    MCPNum:             INT:= 1;              //0: No MCP
                                              //1: 1 MCP (default)
                                              //2: 2 MCPs
    MCP1In:             POINTER;              //Start addr. MCP 1 input signals
    MCP1Out:            POINTER;              //Start addr. MCP 1 output signals
    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. MCP 2 input signals
    MCP2Out:            POINTER;              //Start addr. MCP 2 output signals
    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;
```

```
Code                                                    Comment
    MCP2StopI:          BOOL:= FALSE;
    MCP1NotSend:        BOOL:= FALSE;
    MCP2NotSend         BOOL:= FALSE;
    MCPSDB210:          BOOL:= FALSE;
    MCPCopyDB77:        BOOL:= FALSE;
    MCPBusType:         BYTE = 0;
    HHU:                INT ;                   //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;            //Selection of handwheel using MMC
    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 display
END_VAR
```

```
Code                                                    Comment
VAR_OUTPUT
    MaxBAG:                 INT ;
    MaxChan:                INT ;
    MaxAxis:                INT ;
    ActivChan:              ARRAY[1..10] OF BOOL;
    ActivAxis:              ARRAY[1.0.31] OF BOOL;
    UDInt : INT ;
    UDHex: INT ;
    UDReal : INT ;
END_VAR
```

Description of formal parameters 810D, 840D

The table below lists all formal parameters of the RUN_UP function for 810D, 840D:

Signal	I/O	Type	Value range	Remarks
MCPNum	I	Int	Up to 2	Number of active MCPs 0: No MCP installed
MCP1In MCP2In	I	Pointer	I0.0 to I120.0 or F0.0 to F248.0 or DBn DBX0.0 to DBXm.0	Start address for input signals of relevant machine control panel
MCP1Out MCP2Out	I	Pointer	Q0.0 to Q120.0 or F0.0 to F248.0 or DBn DBX0.0 to DBXm.0	Start address for output signals of relevant machine control panel
MCP1StatSend MCP2StatSend	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for data transmission to the machine control panel: DW#16#08000000: Timeout, otherwise 0
MCP1StatRec MCP2StatRec	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for data reception by machine control panel: DW#16#00000400: Timeout, otherwise 0
MCP1BusAdr MCP2BusAdr	I	Int	1...15	Bus address of machine control panel
MCP1Timeout MCP2Timeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for machine control panel
MCP1Cycl MCP2Cycl	I	S5time	Recommendation: 200 ms	Time reference for cyclic updating of signals to machine control panel

Signal	I/O	Type	Value range	Remarks
MCPMPI	I	Bool		1: All machine control panels connected to the MPI bus (without GD parameterization)
MCP1Stop MCP2Stop	I	Bool		0: Start transfer of machine control panel signals 1: Stop transfer of machine control panel signals
MCP1NotSend MCP2NotSend	I	Bool		0: Send and receive operation activated 1: Receive MCP signals only (under development)
MCPSDB210	I	Bool		0: No SDB 210 for MCP 1: Activate timeout monitors on SDB 210 for MCP
MCPCopyDB77	I	Bool		1: Copy between DB 77 and MCP pointers on DB 7 Can only be used with standard SDB 210 configuration on DB 77.
MCPBusType	I	Byte		0: MPI or OPI b#16#33: PROFIBUS for MCP1 and MCP2
HHU	I	Int		HHU interface 0: No HHU 1: HHU to MPI with SDB 210 configuration (for SW V3.x): 2: HHU to OPI or MPI if FB 1 parameter BHGMPI is also set to TRUE (SW V4.x and higher)
BHGIn	I	Pointer	I0.0 to I124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address PLC receive data from HHU
BHGOut	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address PLC transmit data to HHU
BHGStatSend	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for transmitting data to HHU: DW#16#08000000: Timeout, otherwise 0
BHGStatRec	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for receiving data from HHU: DW#16#00000400: Timeout, otherwise 0

Signal	I/O	Type	Value range	Remarks
BHGInLen	I	Byte	HHU default: B#16#6 (6 bytes)	Quantity of data received from handheld unit
BHGOutLen	I	Byte	HHU default: B#16#14 (20 bytes)	Quantity of data transmitted to handheld unit
BHGTimeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for handheld unit
BHGCycl	I	S5time	Recommendation: 100 ms	Time reference for cyclic updating of signals to handheld unit
BHGRecGDNo	I	Int	HHU default: 2	Receive GD circle no.
BHGRecGBZNo	I	Int	HHU default: 2	Receive GI no.
BHGRecObjNo	I	Int	HHU default: 1	Object number for receive GI
BHGSendGDNo	I	Int	HHU default: 2	Transmit GD circle no.
BHGSendGBZNo	I	Int	HHU default: 1	Transmit GI no.
BHGSendObjNo	I	Int	HHU default: 1	Object number for transmit GI
BHGMPI	I	Bool		1: Handheld unit coupled to MPI (without SDB 210 config.) Parameter HHU must be set to 2.
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 HHU signals only (under development)
NCCyclTimeout	I	S5time	Recommendation: 200 ms	Cyclic sign-of-life monitoring NCK
NCRunupTimeout	I	S5time	Recommendation: 50 s	Powerup 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
MMCToIF	I	Bool		Transmission of MMC signals to interface (modes, program control, etc.) true: active
HWheelMMC	I	Bool		True: Handwheel selection via MMC False: Handwheel selection via user program

Signal	I/O	Type	Value range	Remarks
MsgUser	I	Int	0...25	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	Default, false	Evaluate T function in OB40
IRAuxfuH	I	Bool	Default, false	Evaluate H function in OB40
IRAuxfuE	I	Bool	Default, false	Evaluate DL function in OB40
UserVersion	I	Pointer		Pointer to string variable. The associated string variable is indicated in the version display (max. 41 characters).
MaxBAG	Q	INT	1..10	Number of mode groups
MaxChan	Q	INT	1..10	Number of channels
MaxAxis	Q	INT	1..31	Number of axes
ActivChan	Q	ARRAY[1..10] OF BOOL		Bit string for active channels
ActivAxis	Q	ARRAY[1..31] OF BOOL		Bit string for active axes
UDInt	Q	Int		Quantity of integer machine data in DB 20
UDHex	Q	Int		Quantity of hexadecimal machine data in DB20
UDReal	Q	Int		Quantity of real machine data in DB 20

Declaration 810D

```
Code                                                    Comment
FUNCTION_BLOCK FB 1
VAR_INPUT
    MCPNum:               INT:= 1;                      //0: No MCP
                                                        //1: 1 MCP (default)
                                                        //2: 2 MCPs
    MCP1In:               POINTER;
    MCP1Out:              POINTER;
    MCP1StatRec:          POINTER;
    MCP1Timeout:          S5TIME:= S5T#700MS;
    MCP2In:               POINTER;
    MCP2Out:              POINTER;
    MCP2StatRec:          POINTER;
    MCP2Timeout:          S5TIME:= S5T#700MS;
    HHU:                  INT:= 0;                      //Handheld unit interface
                                                        //0 - no HHU
```

```
Code                                                    Comment
                                                        //1 - HHU on MPI
    BHGIn:                  POINTER;                    //Transmit data of the HHU
    BHGOut:                 POINTER;                    //Receive data of the HHU
    BHGStatRec:             POINTER;                    //Status DW for receiving HHU
    BHGTimeout:             S5TIME:= S5T#700MS;
    NCCyclTimeout:          S5TIME:= S5T#200MS;
    NCRunupTimeout:         S5TIME:= S5T#50S;
    ListMDecGrp:            INT:= 0;
    NCKomm:                 BOOL:= FALSE;
    MMCToIF:                BOOL:= TRUE;
    HWheelMMC:              BOOL:= TRUE;                //Selection of handwheel using MMC
    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 E function in OB40
    UserVersion:            Pointer;                    //Pointer to string variable indicated
                                                        //in version display
END_VAR

VAR OUTPUT
    MaxBAG:                 INT ;
    MaxChan:                INT ;
    MaxAxis:                INT ;
    ActivChan:              ARRAY[1..10] OF BOOL;
    ActivAxis:              ARRAY[1.0.31] OF BOOL;
    UDInt : INT ;
    UDHex: INT ;
    UDReal : INT ;
END_VAR
```

Description of formal parameters 810D and 840D

The table below lists all formal parameters of the RUN_UP function for the 810D and 840D:

Signal	I/O	Type	Value range	Remarks
MCPNum	I	Int	Up to 2	Number of active MCPs 0: No MCP installed
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[1)]

Signal	I/O	Type	Value range	Remarks
MCP1Out MCP2Out	I	Pointer	Q0.0 to Q120.0 or M0.0 to M248.0 or DBn.DBX0.0 to DBXm.0	Start address for output signals of relevant machine control panel[1)]
MCP1StatSend MCP2StatSend	I	Pointer	Q0.0 to Q124.0, M0.0 to M252.0 or DBn.DBX0.0 to DBXm.0	Start address for status double word for data transmission from machine control panel: DW#16#00080000: Timeout, otherwise 0 [1)]
MCP1StatRec MCP2StatRec	I	Pointer	Q0.0 to Q124.0, M0.0 to M252.0 or DBn.DBX0.0 to DBXm.0	Start address for status double word for data reception by machine control panel: DW#16#00040000: Timeout, otherwise 0 [1)]
MCP1BusAdr MCP2BusAdr	I	Int	1... 15	Bus address of machine control panel
MCP1Timeout MCP2Timeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for machine control panel
MCP1Cycl MCP2Cycl	I	S5time	Recommendation: 200 ms	Time reference for cyclic updating of signals to machine control panel
MCPMPI	I	Bool		1: All machine control panels connected to the MPI bus (without GD parameterization)
MCP1Stop MCP2Stop	I	Bool		0: Start transfer of machine control panel signals 1: Stop transfer of machine control panel signals
MCP1NotSend MCP2NotSend	I	Bool		0: Send and receive operation activated 1: Receive machine control panel signals only
HHU	I	Int		Handheld unit interface 0: No HHU 1: HHU on MPI with SDB 210 configuration
BHGIn	I	Pointer	I0.0 to I124.0, M0.0 to M252.0 or DBn.DBX0.0 to DBXm.0	Start address PLC receive data from HHU [2)]
BHGOut	I	Pointer	Q0.0 to Q124.0, M0.0 to M252.0 or DBn.DBX0.0 to DBXm.0	Start address PLC transmit data to HHU [2)]

Signal	I/O	Type	Value range	Remarks
BHGStatSend	I	Pointer	Q0.0 to Q124.0, M0.0 to M252.0 or DBn.DBX0.0 to DBXm.0	Start address for the status double word for sending data to the HHU: DW#16#08000000: Timeout, otherwise 0 2)
BHGStatRec	I	Pointer	Q0.0 to Q124.0, M0.0 to M252.0 or DBn.DBX0.0 to DBXm.0	Start address for the status double word for receiving data from HHU: DW#16#00040000: Timeout, otherwise 0 2)
BHGInLen	I	BYTE	HHU default: B#16#6 (6 bytes)	Quantity of data received from handheld unit
BHGOutLen	I	Byte	HHU default: B#16#14 (10 bytes)	Quantity of data transmitted to handheld unit
BHGTimeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for handheld unit
BHGCycl	I	S5time	Recommendation: 100 ms	Time reference for cyclic updating of signals to handheld unit
BHGRecGDNo	I	Int	HHU default: 1	Receive GD circle no.
BHGRecGBZNo	I	Int	HHU default: 2	Receive GI no.
BHGRecObjNo	I	Int	HHU default: 1	Object number for receive GI
BHGSendGDNo	I	Int	HHU default: 2	Transmit GD circle no.
BHGSendGBZNo	I	Int	HHU default: 1	Transmit GI no.
BHGSendObjNo	I	Int	HHU default: 1	Object number for transmit GI
BHGMPI	I	Bool		1: Handheld unit coupled to MPI (without SDB 210 config.) Parameter HHU must be set to 2.
BHGStop	I	Bool		0: Start transfer of HHU signals 1: Stop transfer of HHU signals
BHGNotSend	I	Bool		0: Send and receive operation activated 1: Receive HHU signals only
NCCyclTimeout	I	S5time	Recommendation: 200 ms	Cyclic sign-of-life monitoring NCK
NCRunupTimeout	I	S5time	Recommendation: 50 s	Powerup monitoring NCK
ListMDecGrp	I	INT	0...16	
NCKomm	I	Bool		PLC NC communications services (FB 2/3/4/5/7: Put/Get/PI_SERV/GETGUD) 1: active

Signal	I/O	Type	Value range	Remarks
MMCToIF	I	Bool		Transmission of MMC signals to interface (modes, program control, etc.) true: active
HWheelMMC	I	Bool		True: Handwheel selection via MMC False: Handwheel selection using user prog.
MsgUser	I	Int	0...25	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 OB40
IRAuxfuH	I	Bool		Evaluate H function in OB40
IRAuxfuE	I	Bool		Evaluate DL function in OB40
UserVersion	I	Pointer		Pointer to string variable. The associated string variable is indicated in the version display (max. 54 characters).
MaxBAG	Q	INT	1..10	Number of mode groups
MaxChan	Q	INT	1..10	Number of channels
MaxAxis	Q	INT	1..31	Number of axes
ActivChan	Q	ARRAY [1..10] OF BOOL		Bit string for active channels
ActivAxis	Q	ARRAY [1.0.31] OF BOOL		Bit string for active axes
UDInt	Q	Int		Quantity of integer machine data in DB20
UDHex	Q	Int		Quantity of hexadecimal machine data in DB20
UDReal	Q	Int		Quantity of real machine data in DB20

1) To ensure BP monitoring of the MCP(s), the addresses must be specified as set in SDB 210 on the 810D. The start address is set via SDB 210 on the 810D. In the supplied SDB 210, the start address is specified as IB 0 for input signals and QB 0 for output signals. If a different start address is required, this must be specified using the STEP 7 Communication Configuration package.

2) To ensure BP monitoring of the HHU(s), the addresses must be specified as set in SDB 210 on the 810D.

MCP/HHU monitoring (for 810D, 840D)

The following status information regarding communication with the machine control panels is output in the event of an error:

Available in:	Bit No.	Description
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Time out
SINUMERIK 840D only: MCP1StatSend MCP2StatSend BHGStatSend	27	Transmitter: Timeout

In addition, an error entry is generated in the diagnostics buffer of the PLC,

resulting in the following error messages on the MMC:

- 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 initialized with 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message reset by the BP.

810D example call

An example call for the FB 1 in OB 100 appears below. This example is part of the diskette with the basic program for 810D.

```
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 :=           14,
                       MCP1Timeout :=          S5T#700MS,
                       MCPMPI:=                TRUE,
                       NCCyclTimeout :=        S5T#200MS,
                       NCRunupTimeout :=       S5T#50S);
//INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

840D example call

An example call for the FB 1 in OB 100 appears below. This example is part of the diskette with basic program for 840D.

```
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
```

2.12.2 FB 2: Read GET NC variable

Description of functions

The PLC user program can read variables from the NCK area using FB GET.

FB 2 also includes an Instance DB from the user area (capable of multi-instances with SW V3.7 and higher).

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 ADDR1ADDR8 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 process lasts for several PLC cycles (normally 1-2). The block can be called up in cyclic mode only.

Any errors are indicated by 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.
"DB name.S7 name" is transferred as the actual parameter of the
NCK variable address (Addr1 to Addr8) when FB 2 is called.

Variable addressing

For some NC variables, it is necessary to select area no. and/or line or column from the NC-VAR selector. A basic type can be selected for these variables, i.e., area/column/line are preset to "0".
The contents of the area number, line and column specified by the NC VAR selector are checked for a "0" in the FB. If a "0" is present, the value is transferred to the input parameter. The user must supply the required parameters (UnitX/ColumnX/LineX) before calling FB GET.
Here unit corresponds to the area number.

Note

After communication between the PLC and NC (read/write NC variables, FB2, 3, 5, or PI general services, FB4) has been aborted by POWER OFF, the start jobs must be deleted in the first OB1 run after cold restart or reset (signal: Req = 0).

FB 2 can read NC variables only if basic program parameter NCKomm ="1" has been set (in OB 100: FB 1, DB7).
When **channel-specific** variables are read, only the variables of **one** channel may be addressed in a job (FB 2 call) via Addr1 to Addr8.

In areas V and H, different logic axis numbers must not be assigned in one job.
(Failure to observe this rule results in Error:= TRUE, State:= W#16#02).

NCK variables within **one** group can be combined in a job:

	Area				
Group 1	C 1	N	B	Q	T
Group 2	C[2]	N	B	Q	T
Group 3	V .	H[.]			

The same rules apply to channels 3 to 10 as illustrated as examples in the above table in groups 1 and 2.

Note

Especially when reading several long strings, the number of usable variables can be less than 8.

Declaration

```
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 ;
    FMNCNo :                int;1)
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 table below list all formal parameters of the GET function.

Signal	I/O	Type	Value range	Remarks
Req	I	Bool		Job start with positive signal edge
NumVar	I	Int	1 to 8 (corresponds to use of Addr1 to Addr8)	Number of variables to be read
Addr1 to Addr8	I	Any	[DBName].[VarName]	Variable identifiers from **NC Var selector**
Unit1 to Unit8	I	Byte		Area address, optional for variable addressing

Signal	I/O	Type	Value range	Remarks
Column1 to Column8	I	Word		Column address, optional for variable addressing
Line1 to Line8	I	Word		Line address, optional for variable addressing
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
NDR	Q	Bool		Job successfully executed Data are available
State	Q	Word		See error identifiers
RD1 to 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:

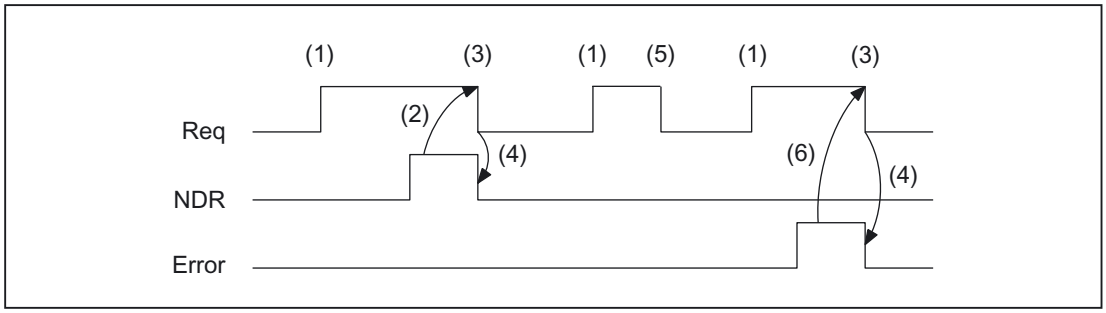
State		Meaning	Note
WORD H	WORD L		
1 to 8	1	Access error	In high byte number of Var in which error occurred
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgment, job not executable	Internal error, try: NC reset
1 to 8	4	Insufficient local user memory available	Read var. is longer than specified in RD1 (to RD8); in high byte number of var in which error occurred
0	5	Format conversion error	Error on conversion of var. type double: Var. is not within 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 to 8	8	Incorrect target area (RD)	RD1 to RD8 may not be local data
0	9	Transmission occupied	Job must be repeated
1 to 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

Configuration steps

Proceed as follows to read NC variables:

- Select variables with the NC VAR selector.
- Store the selected variables in a *.VAR file in the required project catalog (*.S7D).
- 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.
- Set FB 2 parameters

Pulse diagram



(1) Activation of function

(2) Positive acknowledgment: Receive new data

(3) Reset function activation after receipt of acknowledgment

(4) Signal change by means of FB

(5) If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.

(6) Negative acknowledgment: Error has occurred, error code in the output parameter state.

Call example

Reading of three channelspecific 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: AXCONF_MACHAX_USED[1]	char	20070	1	C1AxConfMachAxUsed1
C[1]	M	MD20070: AXCONF_MACHAX_USED[2]	char	20070	1	C1AxConfMachAxUsed2

Area	Block	Name	Type	No.	Byte	S7 Name
C[1]	M	MD20090: SPIND_DEF_MASTER_SPIND	int	20090	1	C1SpindDefMasterSpind

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 DB 120):

Symbol	Operand	Data type
NCVAR	DB120	DB120

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
    E 7.7;                                          //Unassigned machine control panel
                                                    pushbutton
    S       M 100.0;                                //Activate req.
    V       M 100.1;                                //NDR completed message
    R       M 100.0;                                //Terminate job
    V       E 7.6;                                  //Manual error acknowledgment
    V       M 102.0;                                //Error pending
    R       M 100.0;                                //Terminate job
    Call fb 2, db 110(
            Req :=                                  M 100.0,
            NumVar :=                               3,          //Read 3 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 DB120 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 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 NCVAR 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
float	REAL
long	DINT
integer	DINT

Classification of data types	
NCK data type	**S7 data type**
uint_32	DWORD
int_16	INT
uint_16	WORD
unsigned	WORD
char	CHAR or BYTE
string	STRING
bool	BOOL

2.12.3 FB 3: PUT write NC variables

Description of functions

The PLC user program can write variables in the NCK area using FB PUT.

Every FB 3 call must be assigned a separate instance DB from the user area. (multiinstance capability in SW 3.7 and higher).

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 process lasts for several PLC cycles (normally 1-2). The block can be called up in cyclic mode only.

Any errors are indicated by 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. "DB name.S7 name" is transferred as the actual parameter of the NCK variable address (Addr1 to Addr8) when FB3 is called.

Variable addressing

For some NC variables, it is necessary to select area no. and/or line or column in the NC VAR selector. A basic type can be selected for these variables,
i.e., area/column/line are preset to "0".
The contents of the area number, line and column specified by the NC VAR selector are checked for a "0" in the FB. If a "0" is present, the value is transferred to the input parameter. The user must supply the required parameters (UnitX/ColumnX/LineX) before calling FB PUT.
Here unit corresponds to the area number.

Machine data, GUD

In order to define machine data and GUDs without a password, the protection levels of the data you want to access must be redefined to the lowest level. The procedure is described in the Installation Guide (in the section describing the protection level concept) and in the Programming Guide Advanced (protection levels for user data).

Note

After communication between the PLC and NC (read/write NC variables, FB2, 3, 5, or PI general services, FB4) has been aborted by POWER OFF, the start jobs must be deleted in the first OB1 run after cold restart or reset (signal: Req = 0).

FB 3 can only write NC variables if basic program parameter NCKomm has been set to "1" (in OB100: FB 1, DB7). When **channelspecific** variables are written, only variables from **one** and the same channel may be addressed via Addr1 to Addr8 in a job (FB 3 call).

In areas V and H, different logic axis numbers must not be assigned in one job. (Failure to observe this rule results in Error:= TRUE, State:= W#16#02).

NCK variables within **one** group can be combined in a job:

	Area				
Group 1	C[1]	N	B	Q	T
Group 2	C[2]	N	B	Q	T
Group 3	V[.]	H[.]			

The same rules apply to channels 3 to 10 as illustrated as examples in the above table in groups 1 and 2.

Note

Especially when reading several long strings, the number of usable variables can be less than 8.

Declaration

```
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	I/O	Type	Value range	Remarks
Req	I	Bool		Job start with positive signal edge
NumVar	I	Int	1 to 8 (corresponds to use of Addr1 to Addr8)	Number of variables to be written
Addr1 to Addr8	I	Any	[DBName].[VarName]	Variable identifiers from **NC Var selector**
Unit 1 to Unit 8	I	Byte		Area address, optional for variable addressing
Column 1 to Column 8	I	Word		Column address, optional for variable addressing
Line 1 to Line 8	I	Word		Line address, optional for variable addressing
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
Done	Q	Bool		Job successfully executed
State	Q	Word		See error identifiers
SD1 to 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:

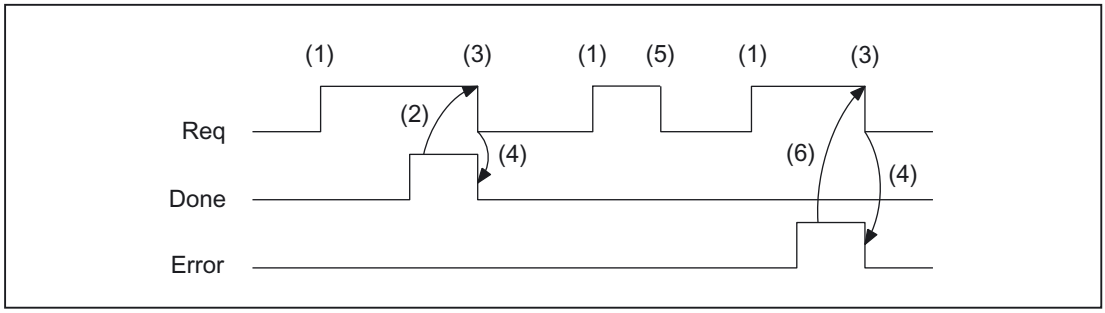
State		Meaning	Note
WORD H	WORD L		
1 to 8	1	Access error	In high byte number of Var in which error occurred
0	2	Error in job	Incorrect compilation of Var in a job
0	3	Negative acknowledgment, job not executable	Internal error, try: Check job, NC Reset
1 to 8	4	Data areas or data types do not match or string is empty	Check data to be written in SD1 to SD8; in high byte number of the Var in which 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 to 8	8	Incorrect target area (SD)	SD1 to SD8 may not be local data

State		Meaning	Note
WORD H	WORD L		
0	9	Transmission occupied	Job must be repeated
1 to 8	10	Error in variable addressing	Unit or column/line contains value 0
0	11	Variable addr. invalid or var. is read-only	Check Addr (or variable name), area, unit
0	12	NumVar = 0	Check parameter NumVar

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 acknowledgment: variables have been written
(3) Reset function activation after receipt of acknowledgment
(4) Signal change by means of FB
(5) If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.
(6) Negative acknowledgment: Error has occurred, error code in the output parameter state.

Call example

Writing of three channelspecific 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	DB120	DB120

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
E 7.7;                               //Unassigned machine control panel pushbutton
    S               M 100.0;         //Activate req.
    V               M 100.1;         //Done completed message
    R               M 100.0;         //Terminate job
    V               E 7.6;           //Manual error acknowledgment
    V               M 102.0;         //Error pending
    R               M 100.0;         //Terminate job
    Call fb 3, db 111(
              Req :=                 M 100.0,
              NumVar :=               3,                      //Write 3 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);
```

2.12.4 FB 4: PI_SERV general PI services

Description of functions

FB PI_SERV can be used to start program-instance services in the NCK area. The possible services are described in this section. A program section, which carries out a particular function (e.g., with tool management, search for empty location in a magazine), is executed in the NCK by making a request via the PI service.
Every FB 4 call must be assigned a separate instance DB from the user area. The multi-instance capability can also be applied in SW 3.7 and higher. Please refer to the STEP 7 Descriptions for more information.
The specified service is referenced via the PIService parameter. The selected PI service is supplied via the freely assignable additional input variables with varying data types (Addr1 to Addr4 for strings, WVar1 to WVar 10 for integer or word variables). A 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 logic "1" in status parameter Done. Any errors are indicated by Error and State.
The "PI" data block (DB16) contains internal descriptions of the possible PI services. A name must then be assigned to this DB 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 extends over several PLC cycles (generally 1 to 2). The block can be called up in cyclic mode only.

Note

After communication between the PLC and NC (read/write NC variables, FB2, 3, 5, or PI general services, FB4) has been aborted by POWER OFF, the start jobs must be deleted in the first OB1 run after cold restart or reset (signal: Req = 0).

FB 4 can start PI services only if the basic program parameter NCKomm has been set to "1" (in OB100: FB 1, DB7).

Declaration

```
FUNCTION_BLOCK FB 4
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 ;
END_VAR
VAR_OUTPUT
    Error :                 BOOL ;
    Done :                  BOOL ;
    State :                 WORD ;
END_VAR
```

Description of formal parameters

The following table shows all formal parameters of the function PI_SERV.

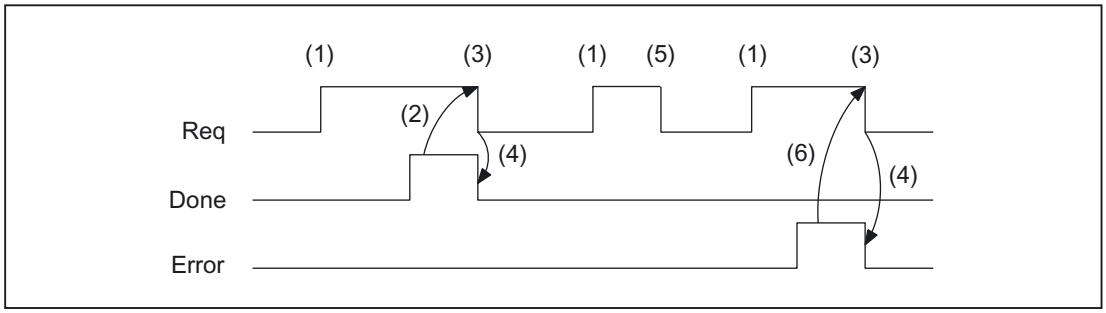
Signal	I/O	Type	Value range	Remarks
Req	I	Bool		Job request
PIService	I	Any	[DBName].[VarName] default is: "PI".[VarName]	PI service description[1)]
Unit	I	Int	1...	Area number
Addr1 to Addr4	I	Any	[DBName].[VarName]	Reference to strings specification according to selected PI service
WVar1 to WVar10	I	Word	1...	Integer or word variables. Specification according to selected PI service (WVar10 SW4 and higher)
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
Done	Q	Bool		Job successfully executed
State	Q	Word		See error identifiers
[1)] See README file on basic program diskette supplied				

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 block output State. The error identifiers, which may be encountered, are as follows:

State	Meaning	Note
3	Negative acknowledgment, job not executable	Internal error, try: NC reset
6	FIFO full	Job must be repeated since queue is full
7	Option not set	BP parameter "NCKomm" is not set
9	Transmission occupied	Job must be repeated

Pulse diagram



(1) Activation of function

(2) Positive acknowledgment: PI service has been executed

(3) Reset function activation after receipt of acknowledgment

(4) Signal change by means of FB

(5) If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.

(6) Negative acknowledgment: Error has occurred, error code in the output parameter State

Overview of PI services

The following section provides an overview of the PI services that can be started from the PLC. The meaning and application of the general FB 4 input variables (Unit, Addr ..., WVar ...) depend on the individual PI service concerned.

PI service	Function	Available in 840D/810D	
SELECT	Select program for processing for one channel	x	
ASUB	Assign interrupt	x	
FINDBL	Activate block search	x	
SETUFR	Activate user frames	x	
CONFIG	Reconfiguration of tagged machine data	x	
CANCEL	Execute cancel	x	
DELETO	Delete tool	x	
CREATO	Generate tool	x	
CREACE	Create cutting edge	x	
TMCRTO	Create tool	x	
TMFDPL	Empty location search for loading	x	
TMMVTL	Prepare magazine location for loading, unload tool		x
TMPOSM	Position magazine location or tool		x
LOGIN	Activate password	x	x
LOGOUT	Reset password	x	x
MMCSEM	Semaphores for various PI services		x
CRCEDN	Create new cutting edge		x
DELECE	Delete a cutting edge		x
TMFPBP	Empty location search		x
TSEARC	Complex search using search screen forms		x
TMPCIT	Set increment value for workpiece counter		x
DIGION	Digitizing on		x
DIGIOF	Digitizing off		x
NCRES	Initiate NC Reset		x
TMRASS	Reset active status		x
TRESMO	Reset monitoring values		x
ACTDEF	Activate a definition (GUD or macro)		x
x: PI service is available			

PI service: SELECT

Select a program for execution for a channel.

Function:

A program stored on the NCK is selected for processing for one channel. This is possible only if the file may be executed. The path name and program name must be entered as described in the Programming Guide (Chapter "File and Program Management", Section "Program Memory"). Please also refer to example of FB 4 for notation of path and program names.

Possible block types	
Block types	
Workpiece directory	WPD
Main program	MPF
Subroutine	SPF
Cycles	CYC
Asynchronous subprograms	ASP
Binary files	BIN

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.SELECT	Program selection
Unit	INT	1 to 10	Channel
Addr1	STRING		Path name
Addr2	STRING		Program name

PI service: ASUB

Assign interrupt

Function:

A program stored on the NCK is assigned an interrupt signal for a channel. This is possible only if the file may be executed. The path name and program name must be entered as described in the Programming Guide (Chapter "File and Program Management", Section "Program Memory"). Please also refer to example of FB 4 for notation of path and program names.

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.ASUP	Assign interrupt
Unit	INT	1 to 10	Channel
WVar1	WORD	1 to 8	Interrupt number

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
WVar2	WORD	1 to 8	Priority
WVar3	WORD	0/1	LIFTFAST
WVar4	WORD	0/1	BLSYNC
Addr1	STRING		Path name
Addr2	STRING		Program name

Note

The SETINT instruction is also used to make the assignment.

The ASUB PI service may only be executed when the channel to be activated is in Reset.

Only PRIO 1 is possible when selecting the ASUB via FB 4 and then starting it with FC 9.

References:
/PA/ Programming Guide

PI service: FINDBL

Activate block search

Function:

A channel is switched to block search mode and the appropriate acknowledgment then transmitted. The block search is then executed immediately by the NCK. The search pointer must already be in the NCK 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. The operator is responsible for providing a collisionfree approach path.

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.FINDBL	Block search
Unit	INT	1 to 10	Channel
WVar1	WORD	x	Preprocessing mode
x: Describes the preprocessing mode x = 1 without calculation x = 2 with calculation x = 3 with main block observation			

PI service: SETUFR

Activate user frames

Function:

User frames are loaded to the NCK. All necessary frame values must be transferred to the NCK beforehand by writing variables with FB 3.

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.SETUFR	Activate user frames
Unit	INT	1 to 10	Channel

PI service: CONFIG

Reconfiguration

Function:

The reconfiguration command activates machine data, which have been entered sequentially by the operator or the PLC, almost in parallel.
The command can only be activated when the control is in Reset 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			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CONFIG	Reconfiguration
Unit	INT	1	
WVar1	INT	1	Classification

PI service: CANCEL

Execute cancel

Function:

The CANCEL command activates the Cancel function (in accordance with key on MMC).

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CANCEL	Cancel

PI service: DELETO

Delete tool

Function:

Deletes the tool assigned to the transferred T number with all cutting edges (in TO, in some cases TU, TUE and TG (type 400), TD and TS blocks).

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.DELETO	Delete tool
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: CREATO

Create tool

Function:

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**	**Value range**	**Meaning**
PIService	ANY	PI.CREATO	Create tool
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: CREACE

Create cutting edge

Function:

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**	**Value range**	**Meaning**
PIService	ANY	PI.CREACE	Create cutting edge
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: TMCRTO

Create tool

Function:

Creation of a tool with specification of an identifier, a duplo number and a T number (optional). 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 NCK. If a TU block exists, it will contain the data set for the tool.
After execution of the PI, 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 PI service MMCSEM for more information.

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMCRTO	Create tool
Unit	INT	1, 2	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			

PI service: TMFDPL

Empty location search for loading

Function:
(dependent on 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, 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 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 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 PI service MMCSEM for more information.

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMFDPL	Empty location for loading
Unit	INT	1, 2	TOA
WVar1	INT		T number
WVar2	INT		Location_number _to
WVar3	INT		Magazine_number _to

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
WVar4	INT		Location_number_ID
WVar5	INT		Magazine_number _ID

PI service: TMMVTL

Prepare magazine location for loading, unload tool

Function:

This PI service is used both to load and unload tools. Whether the PI 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 movements.

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 <- "acknowledgment"

The following variables from the TMC block are used to monitor case 2), 6):

magCBCmd (area no. = TO unit)

magCBCmdState <- "acknowledgment"

Load function

Prepares the specified real magazine for the specified channel for loading, i.e., traverses the magazine 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, 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, the number of the location for tool loading is listed in the TM area in component **magCMCmdPar2** for the real magazine of the channel.

Unload function

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 via a T number or by means of the location and magazine numbers. The value -1 is entered at unused specification points.

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 PI service MMCSEM for more information.

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMMVTL	Make magazine location ready for loading, unload tool
Unit	INT	1, 2	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

PI service: TMPOSM

Position magazine location or tool

Function:
(dependent on 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.

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").

Or

- 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).

Or

- 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**	**Value range**	**Meaning**
PIService	ANY	PI.TMPOSM	Position magazine location or tool
Unit	INT	1, 2	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

PI service: LOGIN

Create password

Function:

Transfers the parameterized password to the NCK. The passwords generally consist of 8 characters. If required, blanks must be added to the string of the password.

Example:

Password: STRING[8] := 'SUNRISE?';

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.LOGIN	Create password
Unit	INT	1	NCK
Addr1	STRING	8 characters	Password

PI service: LOGOUT

Reset password

Function:

The password last transferred to the NCK is reset.

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.LOGOUT	Reset password
Unit	INT	1	NCK

Semaphores for PI services

MMCSEM

For use by MMC and PLC

10 semaphores are provided for each channel. These protect critical functions for the MMC/PLC. By setting the semaphore for the corresponding function number, several MMC/PLC units can be synchronized with it in cases where a function contains a critical section with respect to data to be fetched by the NCK. Semaphores are managed by the MMC/PLC. A semaphore value of 1stipulates a Test & Set operation for the semaphore of the specified function number. The return value of the PI service represents the result of this operation:

- Return value OK:

  Semaphore has been set, critical function can be called.

- Return value REJECTED:

  Semaphore was already set, critical function cannot be called at the present time. The operation must be repeated later.

Note

On completion of the operation (reading data of this PI service) it is **essential** that the **semaphore** is **enabled again**.

Parameter:

WVar1=FunctionNumber

This function number represents a 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 ...    Reserved
10:

WVar2=SemaphorValue

0:       Reset semaphore
1:       Test and set semaphore
```

Parameterization:

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.MMCSEM	Create new cutting edge
Unit	INT	1, 2 to 10	Channel
WVar1	INT	1 to 10	FunctionNumber
WVar2	Word	0, 1	SemaphoreValue

Create new cutting edge

CRCEDN

Function:

Create tool edge by specifying the cutting edge number.

If the T number of an existing tool is specified in parameter “T number” in the PI service, then a cutting edge is set up 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 for the T number (model of absolute D numbers), then the D number value range might extend from 00001 to 31999. The new cutting edge is set up with the specified D number. If the specified cutting edge already exists, then the PI service is aborted in both cases.

Parameterization:

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CRCEDN	Create new cutting edge
Unit	INT		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

Delete a cutting edge

DELECE

Function:

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 for the T number (model of absolute D numbers), then the D number value range might extend from 00001 to 31999. If the specified cutting edge does not exist, then the PI service is aborted in both cases.

Parameterization:

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.DELETE	Delete cutting edge
Unit	INT		TOA
WVar1	INT		T number of tool for which the cutting edge must be deleted. 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

Empty location search

TMFPBP

Function:
(dependent on parameter assignment)

See description in FB 7.

Complex search using search screen forms

TSEARC

Function:
(dependent on 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 operations. 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. This is achieved by selecting the relevant comparison criteria in the operand masks (parMaskT..) in the TF block (i.e., which tool data are to be compared?), entering the types of comparison logic (==, <, >, <=, >=, &&) in the comparison operator data (parDataT..), and entering the comparison values 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. The function number provided for the semaphore feature (PI service MMCSEM) is the function number for TSEARC.

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:

MagNrFrom	PlaceNrFrom	MagNrTo	PlaceNrTo	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, location #P2 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 symmetrical search (see parameter "SearchDirection"), the search domain must only include one magazine (cases 2 and 5 in the table above). If another search domain is specified, the service will malfunction. A reference location must be specified in the parameters MagNrRef and PlaceNrRef for a symmetrical search (see parameter "SearchDirection"). The reference location is specified in the parameters MagNrRef and PlaceNrRef. 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 associated WVar1 parameters for this PI service. See PI service MMCSEM for more information.

The PI service is valid from NCK SW 4.2 and higher.

Parameterization:

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TSEARC	Complex search using search screen forms
Unit	INT	1, 2	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: Symmetrical with real magazine location positioned in front of the location specified by MagNrRef and PlaceNrRef

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
WVar8	INT	0, 1, 2, 3	Kind of Search 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

Set increment value for workpiece counter

TMPCIT

Function:

Incrementing the workpiece counter of the spindle tool

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI.TMPCIT	Set increment value for workpiece counter
Unit	INT	1 to 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

Reset active status

TMRASS

Function:

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 VDI 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	Value range	Meaning
PIService	ANY	PI. TMRASS	Reset active status
Unit	INT	1 to 10	TO area

Reset monitoring values (TRESMO)

This PI service resets the monitoring values of the designated edges of the designated tools to their setpoint (initial) values.
This only relates to tools with active monitoring.

Compare this with the RESETMON NC command.

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI. TRESMO	Reset monitoring values
Unit	INT	1 to 10	TO area
WVar1	WORD	-max ..max	Tool Number 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: Total-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.

Digitizing on (DIGION)

Function:

Selecting digitizing in the specified channel

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI.DIGION	Digitizing on
Unit	INT	1 to 10	Channel

Digitizing OFF (DIGIOF)

Function:

Deactivating digitizing in the specified channel

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI.DIGIOF	Digitizing off
Unit	INT	1 to 10	Channel

Initiate NC Reset

NCRES

Function:

Initiates an NCK Reset. The Unit and WVar1 parameters must be assigned 0.

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI.NCRES	Initiate NC Reset
Unit	INT	0	0
WVar1	WORD	0	0

Call example

Program selection in channel 1 (main program and workpiece program)

Entry of PI for DB 16 and STR for DB 124 with the S7 SYMBOL editor:

Parameterization:

Symbol	Operand	Data type
PI	DB16	DB16
STR	DB124	DB124

```
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]:=               //Main program
                                 '/_N_MPF_DIR/';
           PName_WST:            string[32]:=
                                 '_N_ABC_MPF';
            Path_WST:            string[32]:=               //Workpiece program
                                 '/_N_WKS_DIR/_N_ZYL_WPD';
   end_struct
BEGIN
END_DATA_BLOCK
Function FC "PICall" : VOID
   call fb4,db126(
   I 7.7;                        //Unassigned machine control panel key
   S       M 0.0;                //Activate req.
   V       M 1.1;                //Done completed message
   R       M 0.0;                //Terminate job
   V       I 7.6;                //Manual error acknowledgment
   V       M 1.0;                //Error pending
   R       M 0.0;                //Terminate job
                    Req :=      M0.0,
                    PIService   PI.SELECT,
                    :=
                    Unit :=     1,                            // CHAN 1
                    Addr1 :=    STR.Path,
                    Addr2 :=    STR.PNam  //Main-program
                                e,        selection
```

```
                                        //Addr1:=STR.Path
                                        _WST,
                                        //Addr2:=STR.PNam
                                        e_WST,
                                        //Workpiece-
                                        program selection
                    Error :=    M1.0,
                    Done :=     M1.1,
                    State :=    MW2);
```

2.12.5 FB 5: GETGUD read GUD variable

Description of functions

The PLC user program can read a GUD variable (GUD = Global User Data) from the NCK or channel area using the FB GETGUD. 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. (multiinstance capability in SW 3.7 and higher). 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 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 the 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 the variable pointer. GUD variable fields are an exception. In these, Line1 .. Line8 must also be parameterized with the field index of this variable.

Successful completion of the read process is indicated by a logic "1" in status parameter Done.

The read process extends over several PLC cycles generally 1 to 2).
The block can be called up in cyclic mode only.

Any errors are indicated by Error and State.

Note

After communication between the PLC and NC (read/write NC variables, FB2, 3, 5, or PI general services, FB4) has been aborted by POWER OFF, the start jobs must be deleted in the first OB1 run after cold restart or reset (signal: Req = 0).

FB 5 can read GUD variables only if basic program parameter NCKomm has been set to "1" FB 1, DB 7).

Declaration

```
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 =
                                                       1
    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	I/O	Type	Value range	Remarks
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 variable 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 variable token
VarToken	I	Any		Address to a 10byte token (see example)
Error	Q	Bool		Job negatively acknowledged or not executable
Done	Q	Bool		Job successfully executed.
State	Q	Word		See error identifiers
RD	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:

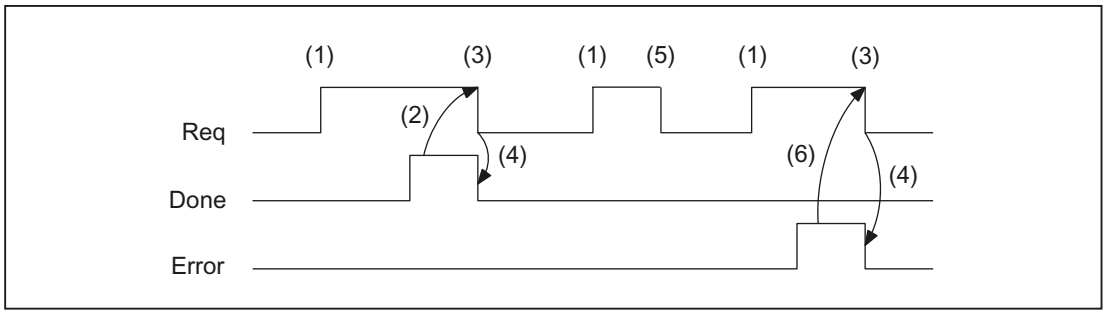
State		Meaning	Note
WORD H	WORD L		
0	1	Access error	
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgment, job not executable	Internal error, try: NC reset

State		Meaning	Note
WORD H	WORD L		
0	4	Data areas or data types do not tally	Check data to be read 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	Check Addr (or variable name), area, unit

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) If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.

(6) Negative acknowledgment: Error has occurred, error code in the output parameter state.

Call example

Reading of a GUD variable with the name "GUDVAR1" as an integer variable (see also table in FB 2: Assignment of NC data type in SIMATIC data type).

Call and parameterization of FB 5 with instance DB 111:

```
DATA_BLOCK DB GUDVAR                          //Assignment to symbol table
    STRUC
    GUDVar1 : STRING[32] := 'GUDVAR1';        //Name is defined by user
    GUDVar1T :
    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
DATA_BLOCK DB 111                             //Unassigned user DB, as instance for
                                              FB 5
FB 5
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 a variable.
Function FC "VariablenCall" :                 VOID
    I 7.7;                                    //Unassigned machine control panel key
    S       M 100.0;                          //Activate req.
    V       M 100.1;                          //Done completed message
    R       M 100.0;                          //Terminate job
    V       I 7.6;                            //Manual error acknowledgment
    V       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 :=                           GUDVAR.GUDVar1,
            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 :=                     GUDVAR.GUDVar1T,
            Error :=                        M102.0,
            Done :=                         M100.1,
            State :=                        MW104,
            RD :=                           P#DB99.DBX0.0
                                            DINT 1
            );
```

2.12.6 FB 7: PI_SERV2 general PI services

Description of functions

A detailed description of the FB 7 is contained in the description of FB 4. The only difference to FB 4 is the number of WVar1 and subsequent parameters. In FB 7, WVar1 to WVar16 are defined in VAR_INPUT (FB4 has WVar1 to WVar10). All other parameters are identical to FB 4. This PI server can be used for all PI services previously implemented with FB 4. In addition, the PI services listed below can only be handled with FB 7.

Declaration

```
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

The following table shows all formal parameters of the function PI_SERV.

Signal	I/O	Type	Value range	Remarks
Req	I	Bool		Job request
PIService	I	Any	[DBName].[VarName] default 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...	Integers or word variables. Specification according to selected PI service
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
Done	Q	Bool		Job has been executed successfully
State	Q	Word		See error identifiers

Overview of additional PI services

The following section provides an overview of the PI services that can be started from the PLC. The meaning and application of the general FB 7 input variables (Unit, Addr..., WVar...) depends on the individual PI service concerned.

PI service	Function	Available in
		D/810D
TMFPBP	Empty location search	*(SW 4 and higher)

Empty location search

TMFPBP

Function:
(dependent on parameter assignment)

This service searches the magazine(s) named in the relevant parameters for an empty location, which meets the specified criteria (tool size and location type). The result of the empty location search can be fetched from variables magCMCmdPar1 (magazine number) and magCMCmdPar2 (location number) in block TMC when the service has functioned correctly. As the PI service stores a result in variables magCMCmdPar1 and magCMCmdPar2, the service must be protected by the semaphore mechanism (PI service MMCSEM) with the function number for _N_TMFDPL in cases where several control units or PLCs are operating on one NC. The search area can be predefined in the following way by setting parameters MagazineNumber_From, LocationNumber_From, MagazineNumber_To, LocationNumber_To:

MagazineNumber_From	LocationNumber_From	MagazineNumber_To	LocationNumber_To	Search area
WVar1	WVar2	WVar3	WVar4	
#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, location #P2 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

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 PI service MMCSEM for more information.

Parameterization:

Parameterization			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMFPBP	Empty location search
Unit	INT	... max. TOA	TOA
WVar1	INT		MagazineNumber_From: Magazine number of magazine from which search must begin (start of search area)
WVar2	INT		LocationNumber_From: Location number of location in magazine MagazineNumber_From at which search must begin
WVar3	INT		MagazineNumber_To: Magazine number of magazine at which search must end
WVar4	INT		LocationNumber_To: Location number of location in magazine MagazineNumber_To at which search must end
WVar5	INT		MagazineNumber_Ref:
WVar6	INT		LocationNumber_Ref:
WVar7	INT	0, 1 .. 7	Number of required half locations to left
WVar8	INT	0, 1 .. 7	Number of required half locations to right
WVar9	INT	0, 1 .. 7	Number of required half locations in upward direction
WVar10	INT	0, 1 .. 7	Number of required half locations in downward direction
WVar11	INT	0, 1 .. 7	Number of required location type
WVar12	INT	0: default 1: forwards 2: backwards 3: symmetrical	Specifies the required search direction 0: Empty location search strategy is set in $TC_MAMP2

2.12.7 FB 9: M : N operating-unit switchover

Description of functions

This block allows switchover between several **operating units** (MMC operator panel fronts and/or MCP machine control panels), which are connected to one or more NCU control modules via a bus system.

References:
/FB2/ Description of Functions, Expansion 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 data block DB19 (see FB 2 above, Chapter 5 Signal Description). FB 9 uses 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 operating-unit switchover:

Tabulated overview of functions:	
Basic function	**Meaning**
MMC call waiting	MMC wants to go online to an NCU
MMC coming	MMC is connecting to an NCU
MMC going	MMC is disconnecting from an NCU
Forced break	MMC must break connection to an NCU
Operating focus changeover to server mode	Change operating focus from one NCU to the other
Active/passive operating mode:	Operator control and monitoring/monitoring only
MCP switchover	As an option, MCP can be switched over with the MMC

Brief description of a few important functions

Active/passive operating mode

An online MMC can operate in two different modes:

Active mode:	Operator can control and monitor
Passive mode:	Operator can monitor (MMC header only)

After switchover to an NCU, this initially requests active operating mode in the PLC of the online NCU. If two MMCs are simultaneously connected online to one NCU, one of the two is always in active and the other in passive operating mode. The operator can request active mode on the passive MMC at the press of a button.

MCP switchover

As an option, an MCP assigned to the MMC can be switched over at the same time. To achieve this, the MCP address must be entered in the **mstt_adress** parameter of the NETNAMES.INI configuration file on the MMCs and **MCPEnable** must be set to true. The MCP of the passive MMC 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 OB 100.

Releases

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

Note

Keys actuated at the moment of switchover remain operative until the new MCP is activated (by the MMC, which is subsequently activated). The override settings for feedrate and spindle also remain valid. To deactivate actuated keys, the input image of the machine control signals must be switched to non actuated signal level on a falling edge of DB10.DBX104.0. The override settings should remain unchanged.
Measures for deactivating keys must be implemented in the PLC user program (see below: Example of override switchover).

Declaration of function

```
FUNKTION_BLOCK FB9
VAR_INPUT
    Ack               :BOOL;            //Acknowledge alarms
    OPMixedMode:      BOOL:= FALSE;     // Hybrid 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;            // Alarm: Error in MMC bus address, bus
                                        type!
    Alarm2            :BOOL;            // Alarm: No confirmation MMC1 offline!
    Alarm3            :BOOL;            // Alarm: MMC1 is not going offline!
    Alarm4            :BOOL;            // Alarm: No confirmation MMC2 offline!
    Alarm5            :BOOL;            // Alarm: MMC2 is not going offline!
    Alarm6           : BOOL ;           // Alarm: Queuing MMC is not going online!
    Report           : BOOL ;           // Message: Sign-of-life monitoring
    ErrorMMC         : BOOL ;           // Error detection MMC
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	**I/O**	**Type**	**Remarks**
Ack	I	BOOL	Acknowledge alarms
OPMixedMode	I	BOOL	Mixed mode deactivated for OP without M to 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 front. **FALSE: =** MCP is not switched over with operator panel front. This can be used to permanently link an MCP. See also MMCx_MCP_SHIFT_LOCK in DB 19
Alarm1	Q	BOOL	Alarm: Error in MMC bus address, bus type!
Alarm2	Q	BOOL	Alarm: No confirmation MMC1 offline!
Alarm3	Q	BOOL	Alarm: MMC1 is not going offline!
Alarm4	Q	BOOL	Alarm: No confirmation MMC2 offline!
Alarm5	Q	BOOL	Alarm: MMC2 is not going offline!
Alarm6	Q	BOOL	Alarm: Queuing MMC is not going online!
Report	Q	BOOL	Message: Sign of life monitoring MMC
ErrorMMC	Q	BOOL	Error detection MMC

Note

The block must be called by the user program. The user must provide an instance DB with any number for this purpose. The call is not multi instance capable.

FB9 call

```
CALL    FB     9 , DB  109 (
    Ack               := Error_ack,          // e.g., MCP reset
    OPMixedMode       := FALSE,
    ActivEnable       = TRUE,               //
    MCPEnable         := TRUE,              // Enable MCP switchover
    Alarm1            := DB2.dbx188.0,      // Error message 700.100
    Alarm2            := DB2.dbx188.1,      // Error message 700.101
    Alarm3            := DB2.dbx188.2,      // Error message 700.102
    Alarm4            := DB2.dbx188.3,      // Error message 700.103
    Alarm5            := DB2.dbx188.4,      // Error message 700.104
    Alarm6            := DB2.dbx188.5,      // Error message 700.105
    Report            :=DB2.dbx192.0)       // Operational message 700.132
```

Note

Input parameter “MCPEnable” must also be set to true to enable MCP switchover. The default value of these parameters is set in this way and need not be specially assigned when the function is called.

Alarm, error

The output parameters "Alarm1" to "Alarm6" and "Report" can be transferred to the DB2 areas for MMC alarm and error messages.

If execution of an MMC function has failed (for which an appropriate error message cannot be displayed), status parameter ErrorMMC is set to 'logic 1' (e.g., booting error when no connection is made).

Example of FB1 call (call in OB100)

```
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 first MCP
    MCP1Cycl                      := S5T#200MS,
    MCP1Stop                      := TRUE,         // MCP disabled
    NCCyclTimeout                 := S5T#200MS,
    NCRunupTimeout                 := S5T#50S);
```

Example of override switchover

```
// Auxiliary flags used M100.0, M100.1, M100.2, M100.3
// Positive edge of MCP1Ready must check override and actions for activation
// Initiate MCP block
// This example covers feed override; for spindle override, interfaces and
// input bytes must be exchanged.
V     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 assignable input or
                                 memory byte)
wei1:
V     M          100.2;          //Switchover takes place
O     DB10.DBX   104.0;          //MCP1Ready
JCN smth2;
V     DB10.DBX   104.0;          //MCP1Ready
FP    M          100.1;          //Edge trigger flag 2
JC smth2;
V     M          100.2;          //Switchover takes place
R     M          100.2;          //Reset auxiliary flag 1
JC smth2;
V     M          100.3;          //Comparison has taken place
JC MCP;                          //Call MCP program
// Guide the stored override to the interface of the switched MCP
// until the override values match
L     EB28;                      //Buffer storage open
T DB21.DBB 4;                    //Guide override interface
L EB 3;                          //Override input byte for feed
<>i;                             //Match?
JC smth2;                        //No, jump
S     M100.3;                    //Yes, set auxiliary flag 2
// When override values match, call the MCP program again
MCP: CALL "MCP_IFM"(             //FC 19
      BAGNo       := B#16#1,
      ChanNo      := B#16#1,
      SpindleIFN  := B#16#0,
      o
```

```
      FeedHold    := M 101.0,
      SpindleHol  := M 101.1);
      d
wei2: NOP         0;
```

2.12.8 FB 10: Safety relay (SI relay)

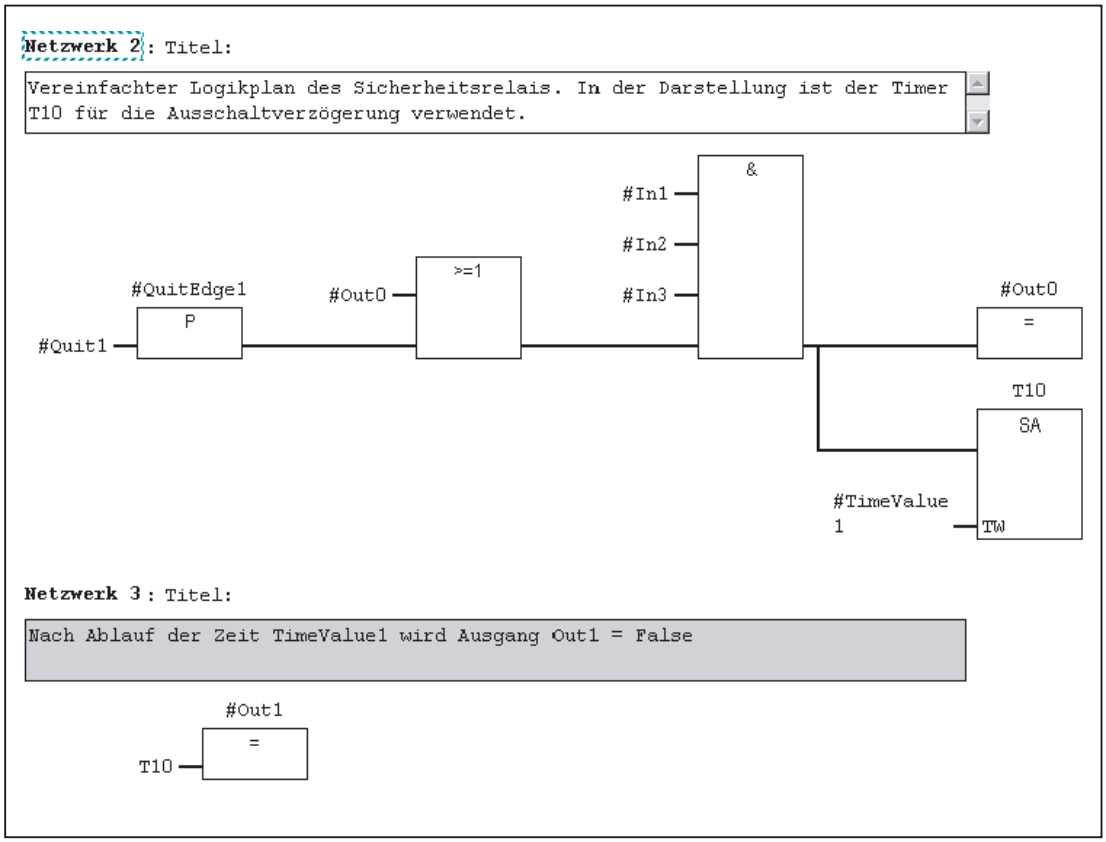
Description of functions

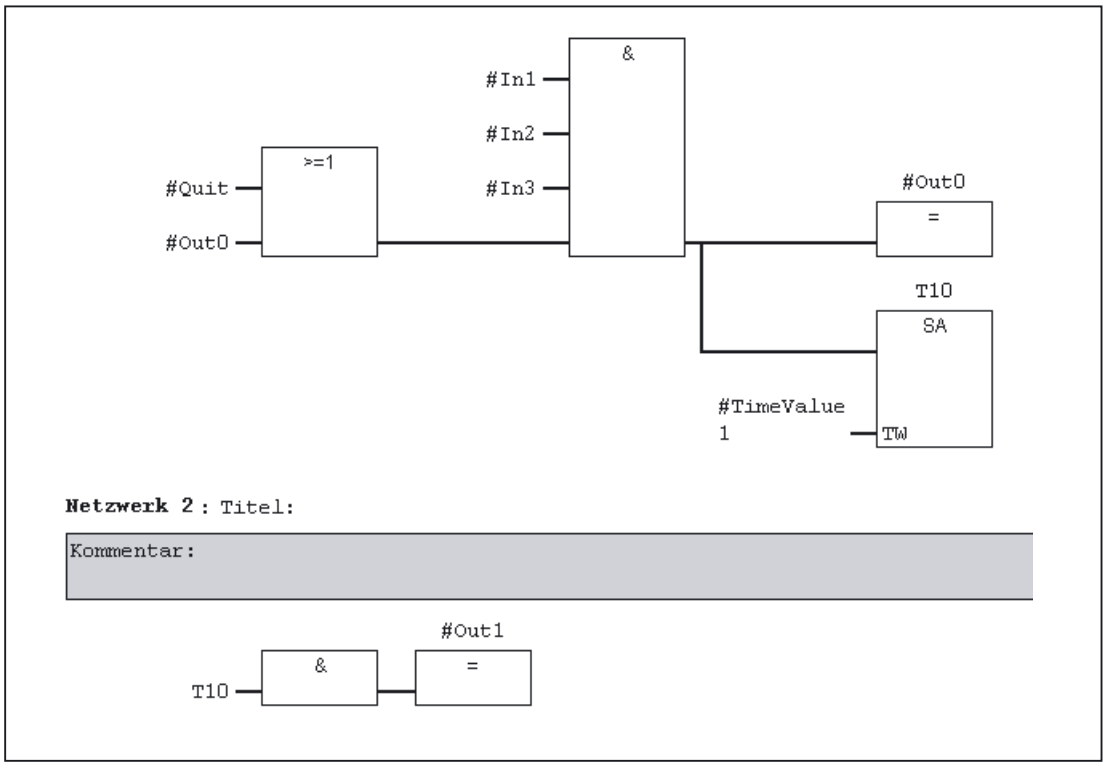
The SPL block "Safety relay" for "Safety Integrated" is the PLC equivalent to the NC function of the same name. The standard SPL "Safety relay" block is designed to support the implementation of an emergency stop function with safe programmable logic. However, it can also be used to implement other similar 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. Parameter FirstRun must be switched to the value TRUE via a retentive data (memory bit, bit in data block) on the first run after control booting. The 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 separate instance.

The corresponding NCK SPL block is described in:
References:
/FBSI/ Description of Functions, Safety Integrated.

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.


Netzwerk 2: Titel:
Vereinfachter Logikplan des Sicherheitsrelais. In der Darstellung ist der Timer T10 für die Ausschaltverzögerung verwendet.
#In1
#In2
#In3
&
#QuitEdge1
P
#Quit1
#Out0
>=1
#Out0
=
T10
SA
#TimeValue
1
TW
Netzwerk 3: Titel:
Nach Ablauf der Zeit TimeValue1 wird Ausgang Out1 = False
#Out1
=
T10


#In1
#In2
#In3
&
#Quit
#Out0
>=1
#Out0
=
T10
SA
#TimeValue
1
TW
Netzwerk 2: Titel:
Kommentar:
#Out1
T10
&
=

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
    Ack1;            : BOOL               //Ack 1 signal
    Ack2 :           . 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 initial 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	**I/O**	**Type**	**Remarks**
In1	I	BOOL	Input 1
In2	I	BOOL	Input 2
In3	I	BOOL	Input 3
Ack1	I	BOOL	Acknowledge input 1
Ack2	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	Q	BOOL	Output undelayed
Out1	Q	BOOL	Output delayed by TimeValue1
Out2	Q	BOOL	Output delayed by TimeValue2
Out3	Q	BOOL	Output delayed by TimeValue3
FirstRun	I/O	BOOL	Activation of initial setting

Note

The block must be called cyclically by the user program once following SPL program startup. The user must provide an instance DB with any number for this purpose. The call is multi-instance-capable.

2.12.9 FB 11: Brake test

Description of Functions

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 so called "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 errorfree 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 required parameterization of NC and drive is described in:
References:
/FBSI/ Description of Functions, 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 servo disable, feedrate enable signals). The "PLC monitoring axis" (DB "Axis".DBX 28.7) must also be set to status 1 by the user program for the entire duration of the test. Before the "PLC monitoring axis" signal is activated, the axis must be switched to "neutral axis" status (e.g., byte 8 must be set to channel 0 in the axis DB, activation signal must be set in the same byte, current-status checkback can be polled in byte 68). The block must not be started until the NC checkback via the appropriate bit (DB "Axis".DBX 63.1) has arrived. 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

Note on FB 18

If FC 15, FC 16 or FC 18 are 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 FC 18 repeatedly, whereby the inactive FC 18 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 checkback**	**Monitoring time value**
Start brake test	DBX 71.0 = 1	TV_BTactiv
Close brake	Bclosed = 1	TV_Bclose
Issue travel command	DBX 64.6 Or DBX 64.7	TV_FeedCommand
Issue test travel command	DBX62.5 = 1	TV_FXSreached
Wait for hold time	DBX62.5 = 1	TV_FXShold
Deselect brake test/ open brake	DBX71.0 = 0	TV_BTactiv
Issue Test O.K.		

Declaration of the function

FUNCTION_BLOCK FB 11

```
VAR_INPUT
    Start          : BOOL ;            //Start of brake test
    Ack            : 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      : S5TIME ;          //TimeValue -> close brake
    TV_FeedComman  : S5TIME            //TimeValue -> force FeedCommand
    d
    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	**I/O**	**Type**	**Remarks**
Start	I	BOOL	Start brake test
Ack	I	BOOL	Acknowledge error
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, check of axis signal DBX71.0
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 -> issue travel command 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	Q	BOOL	Request for close brake
MoveAxis	Q	BOOL	Request to initiate travel movement
Done	Q	BOOL	Test successfully completed
Error	Q	BOOL	Error occurred.
State	Q	BYTE	ErrorStatus

Error identifiers

Error identifiers	
State	**Meaning**
0	No error
1	Start conditions not fulfilled, e.g., axis not under closedloop control/brake closed/axis disabled
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)

Error identifiers	
State	**Meaning**
5	Fixed end stop will not be reached -> axis reset was initiated
6	Traversing inhibit/approach too slow -> fixed end stop cannot be reached. TV FXSreached monitoring timeout
7	Brake is not holding at all (end position is reached)/approach velocity too high
8	Brake opens during the holding period
9	Error in brake test deselection
10	Internal error
11	"PLC monitoring axis" signal not activated by 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 FB11 call:

```
UN      M        111.1;   //Request to close brake, Z axis of FB
=       Q         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
S       DBX      8.4;     //Request neutral axis
V       DBX      68.6;    //Checkback signal, axis is neutral
V       M        110.6;
FP      M        110.1
R       M        110.6
S       M        110.5;   //Next step
R       DBX      8.4;
S       DBX      28.7;    //Request PLC-monitored axis
V       DBX      63.1;    //Checkback signal, axis monitored by PLC
V       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
         Ack               :=E     3.7,     //Acknowledge error with Reset key
         Bclosed          := I     54.0,    //Checkback signal, brake close
                                             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: Travel command
                                             issued
         TV_FXSreache     := S5T#1S,        //Monitoring time value: Fixed stop
                                             reached
         TV_FXShold        := S5T#2S,       //Monitoring time value: Test time Brake
         CloseBrake       :=M      111.1,   //Request to close brake
         MoveAxis         :=M      111.2,   //Request to initiate travel movement
         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
                  ";
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
V        M        111.0;   //Start brake test for FB
V        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
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, Z
                                            axis
         Pos              := -              //Traversing distance: Minus 5 mm
                          5.000000e+000,
         FRate            :=                 //Feedrate: 1000 mm/min
                          1.000000e+003,
```

```
InPos           :=M     113.0,   //Position reached
Error           :=M     113.1,   //Error has occurred
State           := MB   114);    //Error status
```

2.12.10 FB 29: Signal recorder and data trigger diagnostics

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 Boolean signals (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 (byte value), and Var2 and Var3 (integer values) is 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 is set up by the user. The DB of the ring buffer must be passed to the diagnostics FB in the BufDB parameter. The ring buffer must use an array structure, as specified 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 associated instance DB for the FB is a DB from the user area.

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.

FUNCTION_BLOCK FB 29

```
VAR_INPUT
Func : INT ;                                //Function number
//0 = No function, 1 = Signal recorder, 2 = Data trigger
Signal_1 : BOOL ;
    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
    Box: ARRAY [0 to 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:

Signal	I/O	Type	Value range	Remarks
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
VAR	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	Q	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 is complete (function 1), read out the ring buffer DB in STEP7 by opening the data block in the data view. The contents of the ring buffer DB can only 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
```

2.12.11 FC 2: GP_HP Basic program, cyclical section

Description of functions

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 machine control panel 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
                        Ack :=         I6.1);       //Acknowledgment of error
                                                    messages
                                                    //via I 6.1
END_ORGANIZATION_BLOCK
```

2.12.12 FC 3: GP_PRAL Basic program, interrupt controlled section

Description of functions

Block synchronized 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. Normal auxiliary functions are only acknowledged when they have existed for the duration of one cycle. This allows the application to issue a read disable to the NC.

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.

The G functions are evaluated immediately and passed to the application interface.

NC process alarms

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. (only if 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, job-controlled 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 with SW 3.2 and higher:

- Selected **auxiliary functions**
- **Tool-change function** for tool-management option
- **Position reached** for positioning axes, indexing axes and spindles with activation via PLC
- Block transfer to FM (function available soon)

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).
The change signals and function value are available to the user in the associated channel DB. The change signal for this interruptcontrolled function is reset to zero in the cyclical 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 15 (positioning axis), FC 16 (indexing axis) or FC 18 (spindle control) for an axis or spindle, the associated "GP_InPosition" bit makes it possible to implement instantaneous evaluation of the "InPos" signal of the FCs listed above. This capability 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 and
                                       //indexing axes, spindles
```

```
GP_AuxFunction : ARRAY[1..10] OF BOOL;   //Channel-oriented for //auxiliary
                                         functions
GP_FMBlock : ARRAY[1..10] OF BOOL;       //Channel-oriented for block
                                         //transfer to FM (available soon)

//Further local user data may be defined from this point onwards
END_VAR
BEGIN
    CALL FC 3;
    //INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

2.12.13 FC 5: GP_DIAG basic program, diagnostic alarm, and module failure

Description of functions

Description of functions

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. The PROFIBUS MCPs assigned on FB1 are excluded from stopping the PLC.

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 OB82 and OB86 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
```

2.12.14 FC 7: TM_REV transfer block for tool change with revolver

Description of functions

After a revolver has been changed, the user will call this block. 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.

Output parameter "Ready" is set to the value TRUE when the job has been executed correctly.

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

It is not permissible to abort the transfer (e.g., by an external signal reset). 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. Machine data 20310 bit 12 must not be set to 1 for circular magazines.

Manual revolver switching

If a manual action is used to rotate the revolver, this information must be forwarded to the tool management. The asynchronous transfer function of FC 8 must be used to transfer the modified positions of the revolver. This must only occur once on the first manual rotation in the sequence. In this case, the following parameterization of the asynchronous transfer is needed via FC 8:

TaskIdent = 4
TaskIdentNo = Channel
NewToolMag = Magazine number of the revolver
NewToolLoc = Original location of the tool
OldToolMag = Magazine no. buffer storage (spindle) = 9998
OldToolLoc = Buffer storage number of the spindle
Status = 1

This action also causes the same T command to be resent to the tool-management interface if the previous T is programmed again.

Declaration of the function

STL representation

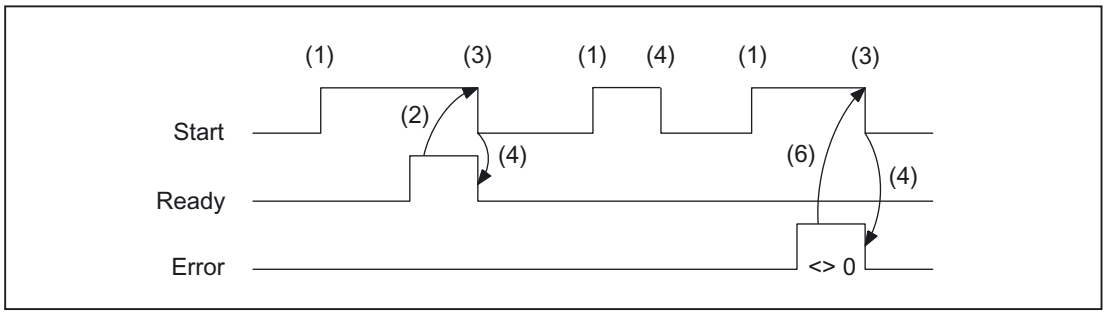
```
FUNCTION FC 7:          void
//NAME :TM_REV
VAR_INPUT
    Start:            BOOL ;
    ChgdRevNo:        BYTE ;
END_VAR
VAR_OUTPUT
    Ready:            BOOL ;
    Error :           INT ;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the TM_REV function.

Signal	I/O	Type	Value range	Remarks
Start	I	Bool		= transfer is started
ChgdRevNo	I	Byte	1..	Number of revolver interface
Ready	Q	Bool		= transfer completed
Error	Q	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")

Timing diagram



(1) Activation of function
(2) Positive acknowledgment: Tool management has been transferred
(3) Reset function activation after receipt of acknowledgment
(4) Signal change by means of FB
(5) This signal chart is not permissible. The job must generally be terminated since the new tool positions must be forwarded to the tool management in the NCK.
(6) Negative acknowledgment: Error has occurred, error code in the output parameter "Error".

Call example

```
CALL FC 7 (                 //Tool management transfer of 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
```

2.12.15 FC 8: TM_TRANS transfer block for tool management

Description of functions

In the case of changed tool positions or status changes, the user will call FC TM_TRANS. This FC is parameterized with parameter "TaskIdent":

- 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 transmitted 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 so called "old tool" (e.g., on loading), the value 0 is assigned to parameters "OldToolMag", "OldToolLoc".

Block FC TM_TRANS may be started (with "Start" parameter = "TRUE") only 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.

Output parameter "Ready" is set to the value TRUE when the job has been executed correctly.

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 complete). 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 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.



Warning

It is not permissible to abort the transfer (e.g., by an external signal reset). 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 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
```

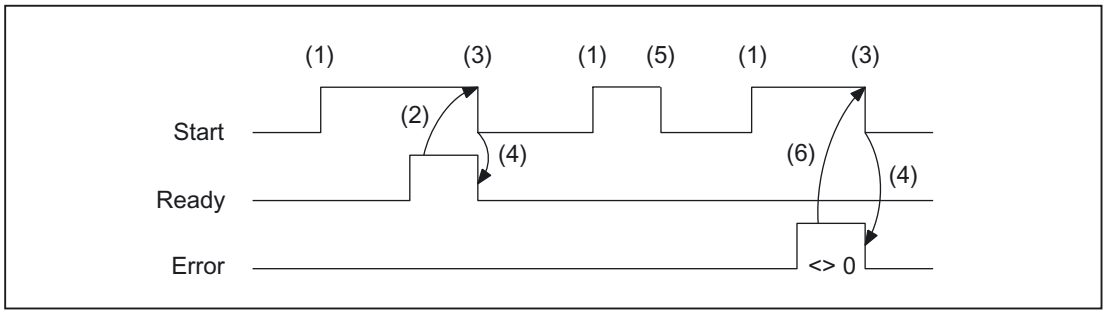
Description of formal parameters

The table below lists all formal parameters of the TM_TRANS function:

Signal	I/O	Type	Value range	Remarks
Start	I	Bool		1 = Start of transfer
TaskIdent	I	Byte	1..5	Interface or tank 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 new tool -1: Tool remains at its location NewToolLoc = any value. Only permissible if TaskIdent = 2
NewToolLoc	I	Int	0 to max. location number	Current location number of new tool

Signal	I/O	Type	Value range	Remarks
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	Q	Bool		1= transfer completed
Error	Q	Int	0..65535	Error checkback 0: No error has occurred 1: Unknown "TaskIdent" 2: Unknown "TaskIdentNo" 3: Illegal job ("interface active" signal for selected revolver = "FALSE") Other values: The number corresponds to the error message of the tool management function in the NCK caused by this transfer.

Timing diagram



(1) Activation of function

(2) Positive acknowledgment: Tool management has been transferred

(3) Reset function activation after receipt of acknowledgment

(4) Signal change by means of FB

(5) This signal chart is not permissible. The job must generally be terminated since the new tool positions must be forwarded to the tool management in the NCK.

(6) Negative acknowledgment: Error has occurred, error code in the output parameter "Error".

Status list

Status = 1:
The operation is complete (loading/unloading/reloading, prepare change, change).

The parameters of FC 8 (FC TM_TRANS) "NewToolMag", "NewToolLoc", "OldToolMag" and "OldToolLoc" must be parameterized to the tool target positions specified in the interface (except for Prepare change). For further information, please refer to description of parameters of FC TM_TRANS or general Section of tool management in the PLC.

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 acknowledgment, 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 complete.

Status = 2:
The "new" tool cannot be made available.

This status is only admissible in conjunction with the "Change tool" 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 has 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 TM_TRANS. 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 take the specified magazine position into account in the subsequent command. However, it can only do so if the 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 "positioning to load point" is enabled, status 5 (not status 1) cannot be used for the function termination.

Status = 6:
The tool management job is 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:
Repetition of the "Prepare Tool" command initiated.

This status is only admissible in conjunction with the Prepare Tool Change 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", "OldToolLoc" must correspond to the original positions of the tools.

Status = 103:
The "new" tool can be inserted.

This status is only permissible in conjunction with the preparations for a tool change if the PLC may reject the new tool (MD: MC_TOOL_MANAGEMENT_MASK, bit 4). The tool positions have remained unchanged. This status is required to continue the forward motion in the NCK (otherwise processing is stopped).

References:
/FBW/ escription of Functions, 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 104 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 acknowledgment status is that the state information 1 to 7 leads to the termination of the command. If FC 8 receives one of the statuses, 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
```

2.12.16 FC 9: ASUB startup of asynchronous subprograms

Description of functions

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).
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 required 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. 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 function is initiated only with the LowHigh edge.

Declaration

```
FUNCTION FC 9: VOID                                    //ASUB
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
```

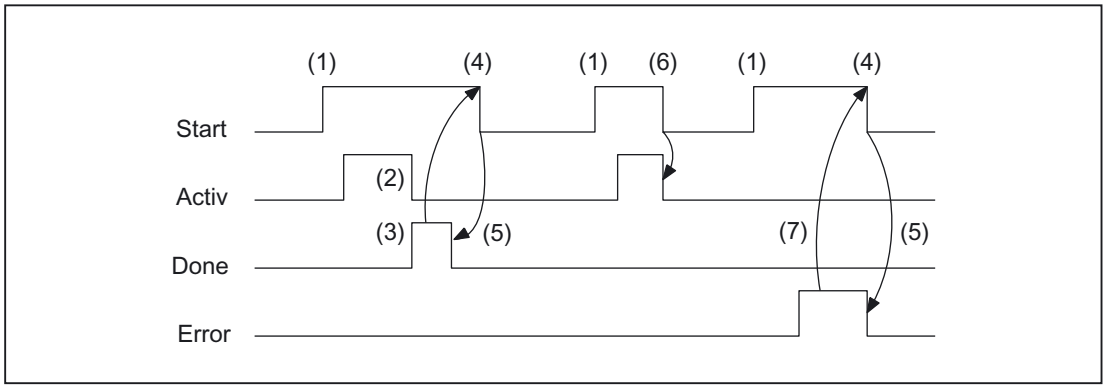
Description of formal parameters

The table below lists all formal parameters of the ASUB function.

Signal	I/O	Type	Value range	Remarks
Start	I	Bool		
ChanNo	I	Int	1 - 10	No. of NC channel
IntNo	I	Int	1 - 8	Interrupt no.
Active	Q	Bool		1 = active
Done	Q	Bool		1 = ASUB terminated
Error	Q	Bool		

Signal	I/O	Type	Value range	Remarks
StartErr	Q	Bool		1 = Interrupt number not allocated
Ref	I/O	Word	Global variable (MW, DBW,..)	1 word per FC 9 (for internal use)

Timing diagram



(1) Activation of function
(2) ASUP active
(3) Positive acknowledgment: ASUB ended
(4) Reset function activation after receipt of acknowledgment
(5) Signal change by means of FB
(6) If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.
(7) Negative acknowledgment: Error has occurred

Call example

```
CALL FC 9 (          //Start an asynchronous subroutine
                     //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);
```

2.12.17 FC 10: AL_MSG error and operating messages

Description of functions

FC AL_MSG evaluates the signals entered in DB 2 and displays them as incoming and outgoing error and operational messages on the MMC. The incoming signals (positive edge) are displayed immediately in the case of both error and operating messages. Outgoing signals (negative edge) are only canceled immediately in the case of operating messages. Error messages remain stored on the MMC - even if the signals no longer exist - until the "acknowledge" parameter is issued, i.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 alarm 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 alarm 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;                 //Set disable condition, reset via
                           //FC AL_MSG, if M 50.0 outputs the signal "0".
```

Error and operational messages are provided by the user in data block DB 2.

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 MMC.

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	I/O	Type	Value range	Remarks
ToUserIF	I	Bool		1 = Transfer 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);         //Acknowledgment of
                                          //error message carried out via
                                          //input I6.1
```

2.12.18 FC 12: AUXFU call interface for user with auxiliary functions

Description of functions

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 jobcontrolled 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

```
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	I/O	Type	Value range	Remarks
Chan	I	Byte	0 to 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 no. (0, 1, 2, etc.)
+ 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
```

2.12.19 FC 13: BHGDisp display control for handheld unit

Description of functions

This module carries out the display control of the handheld unit (HHU). The information to be output on the display is stored as 32 characters in string data ChrArray. A fixed text assignment of 32 characters is therefore required for this string when the data block is created. Variable components within this string can be inserted by means of the optional numerical converter, for which 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. 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).

Signals

Byte 1 is supplied by the output signals of the HHU and the character specifications are supplied by the module. These may not be written by the PLC user program.

Additional parameters

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.

Note

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. In this case, it is advisable to use the input signal from the HHU to the PLC "Acknowledgment digital display" (DB m+5.7) for parameter "Convert". In this way it can be ensured that the most recent numerical information is displayed.

Declaration of the function

STL representation

```
DATA_BLOCK "strdat"                                  //The data block number
                                                     //is defined in the symbol file
    STRUCT
    disp:             STRING [32]:= 'line 1 line     //32 characters are defined
                      2 ';
    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]
    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 to 32)
    Digits :          BYTE ;                         //Number of decimal places (1 to
                                                     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	I/O	Type	Value range	Remarks
Row	I	Byte	0-3	Display line 0: No display output 1: Line 1 2: Line 2 3: Line 1 and line 2
ChrArray	I	String	>= string[32]	This string contains the entire display content
Convert	I	Bool		Activation of numerical conversion
Addr	I	Pointer		Points to the variable to be converted
DataType	I	Byte	1-8	Data type of variable 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 (see parameter Digits)
StringAddr	I	Int	1-32	Address within variable ChrArray
Digits	I	Byte	1-4	Relevant only for data type Real with sign 1: 6.1 digits unsigned 2: 5.2 digits unsigned 3: 4.3 digits unsigned 4: 3.4 digits unsigned
Error	Q	Bool		Conversion error, numerical overflow or error in StringAddr

Ranges of values

Value ranges of data types	
Data type	**Representable numerical range**
BOOL	0, 1
BYTE	0 to 255
WORD	0 to 65535
INT	- 32768 to + 32767
DWORD	0 to 9999999
DINT	- 9999999 to + 9999999

Value ranges of data types	
Data type	**Representable numerical range**
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

Call example

```
CALL FC 13 (
           Row :=               MB 26,
           ChrArray :=          "strdat".disp,     //DB with name strdat in
                                                   //data element symbol table
                                                   //disp is declared as string[32]
                                                   //and assigned entirely
                                                   //with character
           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);
```

2.12.20 FC 15: POS_AX positioning of linear and rotary axes

Description of functions

(Do not use for new applications, function is integrated in FC 18 in SW 3.6 and higher.)

FC POS_AX can be used to traverse axes in any operating mode, also from the PLC. (See also Description of Functions **Positioning Axes SW 2**).

In order to traverse the NC axes via the PLC, the traversing check must be activated for the PLC.

This can be achieved, for example, by calling FC "POS_AX" with activation of the "Start" parameter.

FC "POS_AX" then requests an axis check by the NC.

The NC feeds back the status of this axis in byte 68 in the associated axis interface (DB 31, ...) (see interface lists).

Once the axis check has been completed ("InPos" is True, "Start" changes to zero), it is switched to a neutral state by FC POS AX.

Alternatively, the PLC user program can also request the check for the PLC prior to calling FC "POS_AX".

By calling this function several times in succession, a better response reaction by the axes can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding axis interface in byte 8.

After return of the check, the axis can again be programmed by the NC program.

Note

Rotary axes can be positioned by the shortest possible route through the programming of a negative feed value in absolute programming mode. In incremental mode (parameter "IC" := TRUE), the traversing direction can be defined by means of the parameter "Pos" sign:
Positive sign indicates travel in a positive direction
Negative sign indicates travel in a negative direction

After the FC has been called, ACCU1 contains an error statement by the NCK (but not if the output parameters are assigned to a data block). This is generally the value 0 (signifies: No error has occurred). The interpretation of other numerical values is shown in the following table.

FC 15 must be called cyclically until the "Active" signal produces an edge transition from 1 to 0. Only when the "Active" signal has a 0 state can the axis concerned be restarted (the next start must be delayed by at least one PLC cycle). This also applies when the assignment in data byte 8 has changed.

The function cannot be aborted by means of parameter "Start", but only by means of the axial interface signals (e.g., delete distancetogo). The axial interface also returns status signals of the axis that may need to be evaluated (e.g., exact stop, traverse command).



Warning

If several block calls (FC 15, FC 16, FC 18) are 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.

Error identifiers

Error identifiers	
State	**Meaning**
Errors caused by PLC handling	
1	Several axis/spindle functions have been activated simultaneously
20	A function has been started without the position being reached
30	The axis/spindle has been transferred to the NC while still in motion
40	The axis is programmed in the NC program
Errors which occur during NCK routines. The alarm numbers are described in the 840D Diagnostics Guide	
100	Corresponds to alarm no.: 16830
105	Corresponds to alarm no.: 16770
106	Corresponds to alarm no.: 22052
107	Corresponds to alarm no.: 22051
108	Corresponds to alarm no.: 22050
109	Corresponds to alarm no.: 22055
110	Velocity/speed is negative
111	Setpoint speed is zero
112	Invalid gear stage
115	Programmed position has not been reached
117	G96/G961 is not active in the NC
118	G96/G961 is still active in the NC
120	Not an indexing axis
121	Indexing position error
125	DC (shortest path) not possible
126	Minus absolute value not possible
127	Plus absolute value not possible
130	Software limit switch plus
131	Software limit switch minus
132	Working area limitation plus
133	Working area limitation minus
System or other serious alarms	
200	Corresponds to system alarm no.: 450007

Declaration

```
FUNCTION FC 15: VOID                    //POS_AX
VAR_INPUT
    Start:                  BOOL ;
    AxisNo:                 INT ;
    IC:                     BOOL ;
    Inch:                   BOOL ;
    HWheelOv:               BOOL ;
    Pos:                    REAL;
    FRate:                  REAL;
END_VAR
VAR_OUTPUT
    InPos:                  BOOL ;
    Active:                 BOOL ;
    StartErr:               BOOL ;
    Error :                 BOOL ;
END_VAR
```

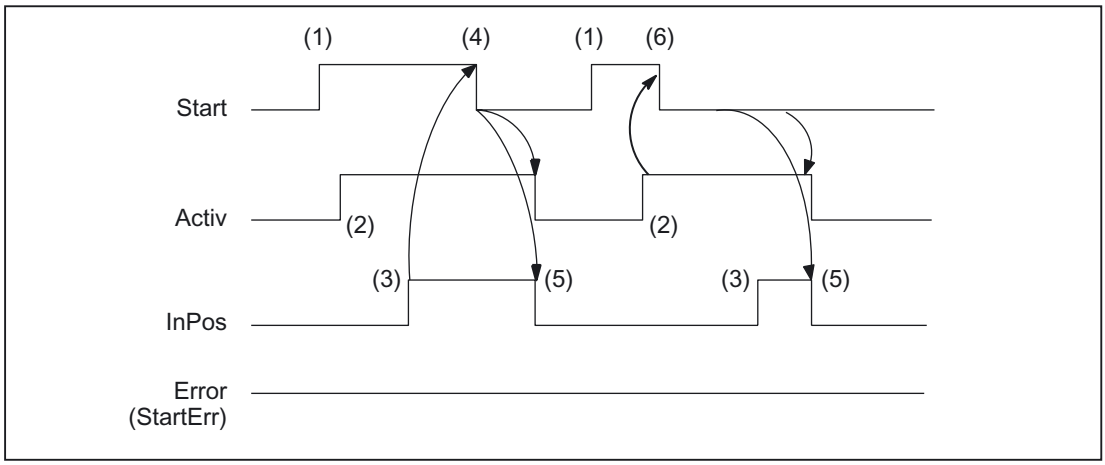
Description of formal parameters

The table below lists all formal parameters of the POS_AX function:

Signal	I/O	Type	Value range	Remarks
Start	I	Bool		
AxisNo	I	Byte	1 - 31	No. of axis to be traversed
IC	I	Bool		0 = Absolute 1 = Incremental
Inch	I	Bool		0 = mm 1 = Inch
HWheelOv	I	Bool		1 = Handwheel override
Pos	I	Real	∓ 0,1469368 l -38 to ∓ 0,1701412 l +39	Position of linear axis: mm Rotary axis: Degr.
FRate	I	Real	∓ 0,1469368 l -38 to ∓ 0,1701412 l +39	Feedrate of linear axis: mm/min Rotary axis: rev/min
InPos	Q	Bool		1 = In position
Active	Q	Bool		1 = Active

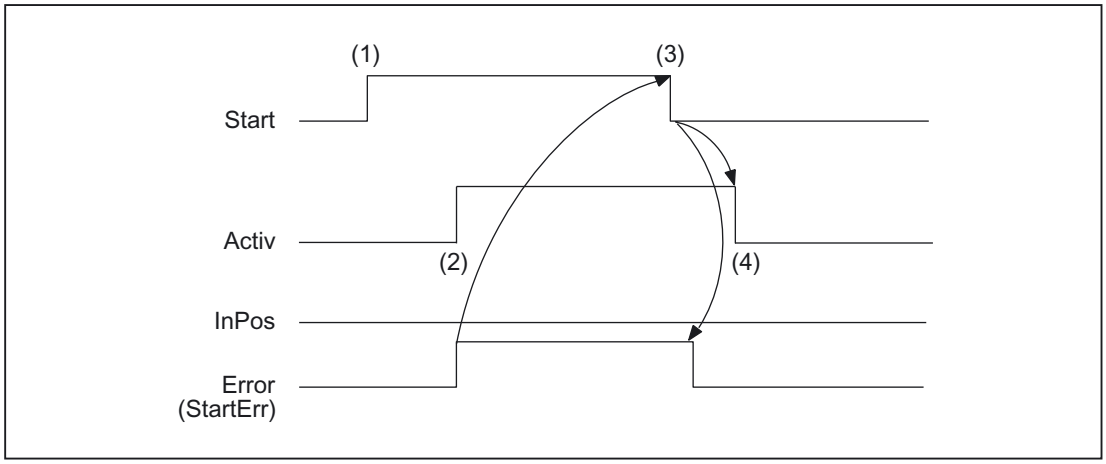
Signal	I/O	Type	Value range	Remarks
StartErr	Q	Bool		Axis cannot be started, see Table 8-48 Error identifiers
Error	Q	Bool		Traversing error [1] see table 8–48 Error codes
[1] Error evaluation by user in the PLC				

Timing diagram



(1) Activation of function
(2) Positioning axes active
(3) Positive acknowledgment: Position reached
(4) Reset function activation after receipt of acknowledgment
(5) Signal change using FC
(6) Reset function activation after receipt of active signal

Timing diagram (fault scenario)



(1) Activation of function by means of a positive edge
(2) Negative acknowledgment: Error has occurred
(3) Reset function activation after receipt of acknowledgment
(4) Signal change using FC

Call example

```
CALL FC 15 (
             Start :=           M 100.0,
             AxisNo :=          5,
             IC :=              #incr,          //e.g., local variable
             Inch :=            FALSE,
             HWheelOv :=        FALSE,
             Pos :=             MD160,
             FRate :=           MD164,
             InPos :=           Q 36.0,
             Active :=          Q 36.1,
             StartErr :=        Q 36.2,
             Error :=           Q 36.3);
```

2.12.21 FC 16: PART_AX positioning of indexing axes

Description of functions

(Do not use for new applications, function is integrated in FC 18 in SW 3.6 and higher.)

FC PART_AX can be used to traverse NC axes defined via machine data as "indexing axes", also from the PLC.

References:
/FB2/ Description of Functions, Expansion Functions; Indexing Axes (T1)

In order to traverse the indexing axes via the PLC, the traversing check must be activated for the PLC.

This can be achieved, for example, by calling FC "PART_AX" with activation of the "Start" parameter.

The FC "PART_AX" then requests checking of the axes from the NC.

The NC feeds back the status of this axis in byte 68 in the associated axis interface (DB 31, ...) (see interface lists).

On completion ("InPos" is True, "Start" changes to zero), the axis check function is switched to a neutral status by FC PART_AX.

Alternatively, the PLC user program can also request the check for the PLC prior to calling FC "PART_AX".

By calling this function several times in succession, a better response reaction by the axes can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding axis interface in byte 8.

After return of the check, the axis can again be programmed by the NC program.

Note

After the FC has been called, ACCU1 contains an error statement by the NCK (but not if the output parameters are assigned to a data block). This is generally the value 0 (signifies: No error has occurred). See table 8-48 for the interpretation of other numerical values for error codes in FC 15.

FC 16 must be called cyclically until the "Active" signal produces an edge transition from 1 to 0. Only when the "Active" signal has a 0 state can the axis concerned be restarted (the next start must be delayed by at least one PLC cycle). This also applies when the assignment in data byte 8 has changed.

The function cannot be aborted by means of parameter "Start", but only by means of the axial interface signals (e.g., delete distancetogo). The axial interface also returns status signals of the axis that may need to be evaluated (e.g., exact stop, traverse command).



Warning

If several block calls (FC 15, FC 16, FC 18) are 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.

Declaration

```
FUNCTION FC 16: VOID                        //PART_AX
VAR_INPUT
    Start:                  BOOL ;
    AxisNo:                 INT ;
    IC:                     BOOL ;
    DC:                     BOOL
    Minus:                  BOOL ;
    Plus:                   BOOL ;
    Pos:                    INT ;
    FRate:                  REAL;
END_VAR
VAR_OUTPUT
    InPos:                  BOOL ;
    Active:                 BOOL ;
    StartErr:               BOOL ;
    Error :                 BOOL ;
END_VAR
```

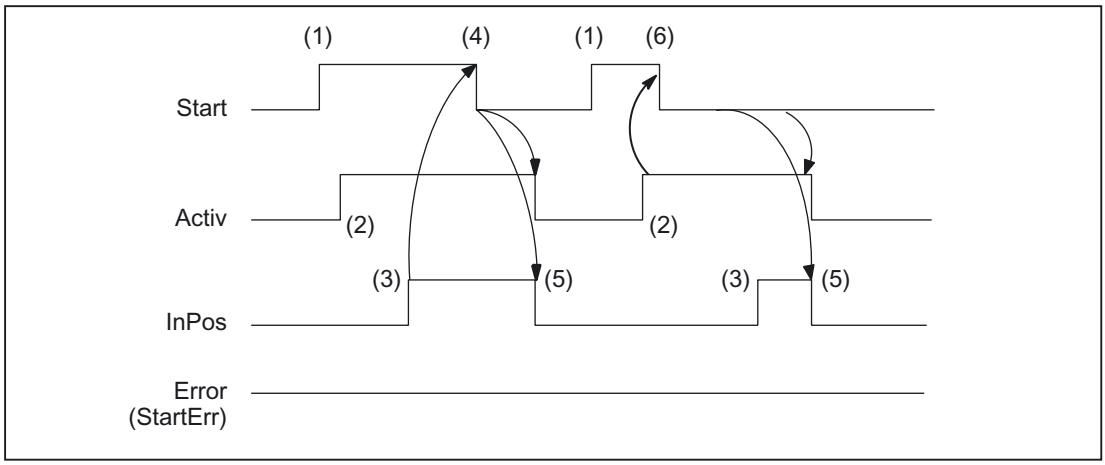
Description of formal parameters

The table below lists all formal parameters of the PART_AX function:

Signal	I/O	Type	Value range	Remarks
Start	I	Bool		
AxisNo	I	Int	1 - 31	No. of axis to be traversed
IC	I	Bool		= Absolute direction input 1 = Incremental direction input
DC	I	Bool		= Predefined direction = Shortest path

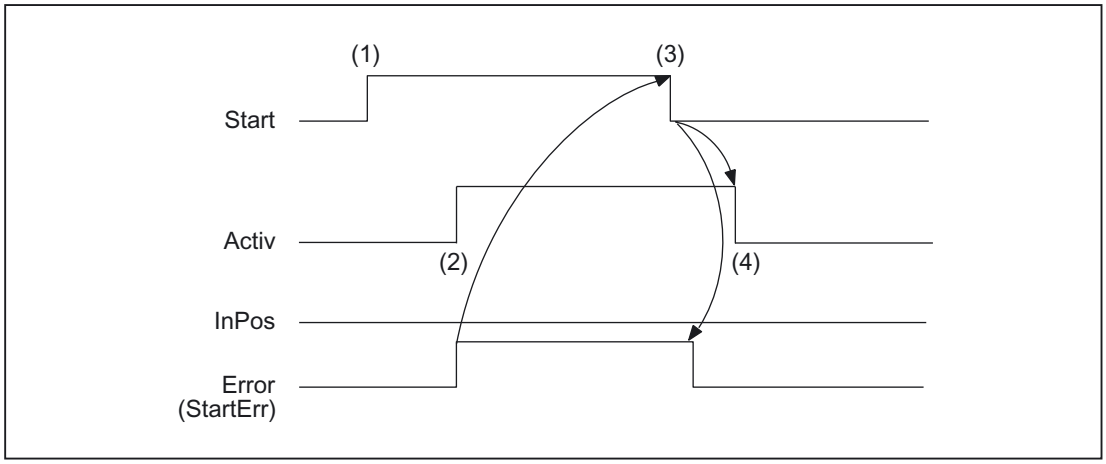
Signal	I/O	Type	Value range	Remarks
Minus	I	Bool		: Rotary axis motion as for linear axis 1: Motion in negative direction with rotary axes
Plus	I	Bool		: Rotary axis motion as for linear axis 1: Motion in positive direction with rotary axes
Pos	I	Int	1 to 60	No. of indexing position
FRate	I	Real	∓ 0,1469368 I -38 to ∓ 0,1701412 I +39	Feedrate of linear axis: mm/min Rotary axis: rev/min
InPos	Q	Bool		= Position reached
Active	Q	Bool		= Active
StartErr	Q	Bool		Axis cannot be started, see Table 8-48 Error identifiers
Error	Q	Bool		Traversing error [1] see table 8–48 Error codes
[1] Error evaluation by user in the PLC				

Timing diagram



(1) Activation of function by means of a positive edge
(2) Positioning axes active
(3) Positive acknowledgment: Position reached
(4) Reset function activation after receipt of acknowledgment
(5) Signal change using FC
(6) Reset function activation after receipt of active signal

Timing diagram (fault scenario)



(1) Activation of function by means of a positive edge
(2) Negative acknowledgment: Error occurred
(3) Reset function activation after receipt of acknowledgment
(4) Signal change using FC

Call example

```
CALL FC 16 (                    //Position an indexing axis
            Start :=            I72.4,
            AxisNo :=           6,
            IC :=               FALSE,
            DC :=               #short,              //e.g., local variable
            Minus :=            FALSE,
            Plus :=             FALSE,
            Pos :=              MW 168,
            FRate :=            MD164,
            InPos :=            Q 36.4,
            Active :=           Q 36.5,
            StartErr :=         Q 36.6,
            Error :=            Q 36.7);
```

2.12.22 FC 17: YDelta star/delta changeover

Description of functions

The block for 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:	Delete the "Motor selection in progress" interface signal in the relevant axis DB (DB 31, ... DBX21.5) and connect the changeover process using "Motor selection" A (DB 31, ... DBX21.5).
Step 2:	As soon as the "Pulses enabled" = 0 (DB 31, ... DBX93.7) checkback signal 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 "Motor selection in progress" (DB31, ... DBX21.5).

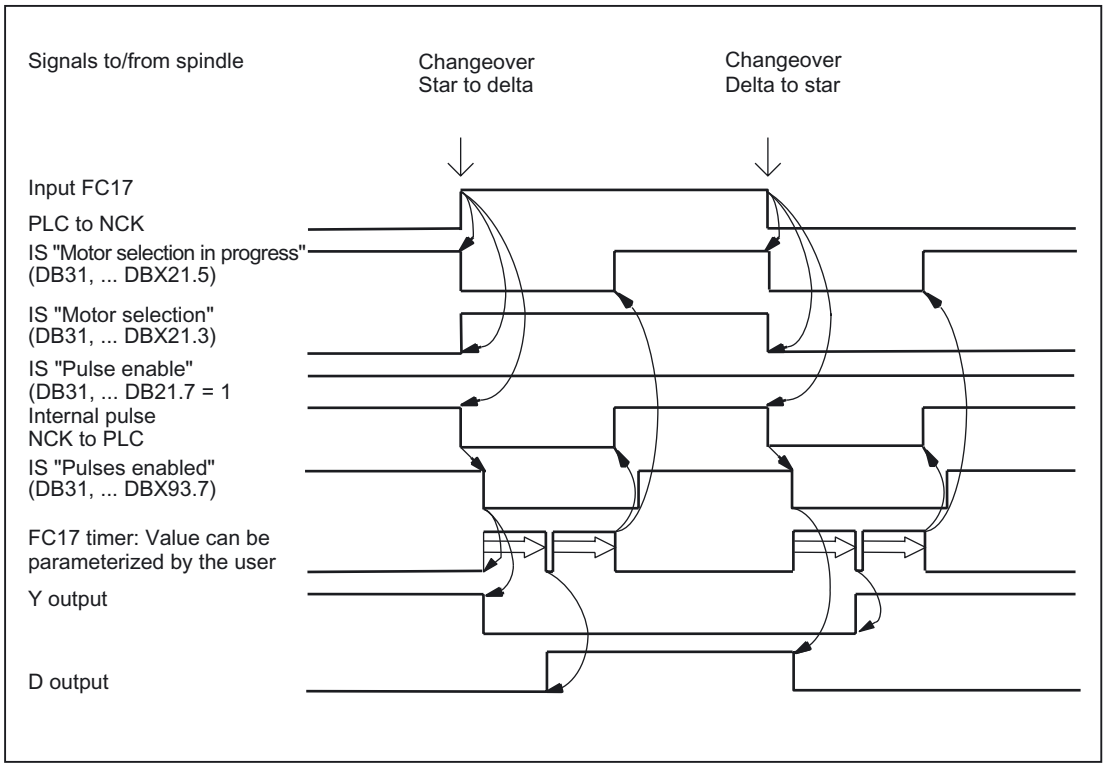


Fig. 2-29 Timing with interface signals with a delay of 500 ms set in FC 17

More information about motor speed adjustment can be found in:
References:
/FB/ Description of Functions, Basic Machine; Spindles (S1);
Section: Configurable Gear Adjustments
/FB/ Description of Functions, Basic Machine; Velocities, Setpoint/Actual Value Systems, Closed-Loop Control (G2)

Error message

If the parameter "SpindleIFNo" is not in the permissible range, the PLC is stopped with output of alarm 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

Changeover does not occur if the spindle is in an axis mode, such as M70, SPOS.

Supplementary 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 pulses on the drive, IS "Current controller active" (DB31, ... DBX61.7) and "Speed controller active" (DB31, ... DBX61.6) are deactivated simultaneously to IS "Pulses enabled" (DB31, ... DBX93.7).
- 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 FC17, the changeover of the star/delta contactor cannot be delayed by the user. 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	I/O	Type	Value range	Remarks
YDelta	I	Bool		= star = delta. 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..	Changeover time
TimerNo	I	Int	10..	Timer for programming the wait time
Y	Q	Bool		Energizing of star contactor
Delta	Q	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
```

2.12.23 FC 18: SpinCtrl spindle control

Description of functions

FC SpinCtrl can be used to control spindles and axes from the PLC.

References:
/FB1/ Description of Functions, Basic Machine; Spindles (S1)
/FB2/ Description of Functions, Expansion Functions; Positioning Axes (P2)
/FB/ Description of Functions, Expansion 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" state 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 (DB 31, ...) (see interface lists). Once the axis/spindle is operating under PLC control, the travel command for the active state 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 reaction 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 distancetogo). The axial interface also returns axis status signals that may need to be evaluated
(e.g., exact stop, traverse command).

InPos on spindle - rotation/oscillation:

For the "Rotate spindle" and "Oscillate spindle" functions, the meaning of the "InPos" parameter is defined as follows:
Setpoint speed is output --> function was started without error.
The reaching of the required spindle speed must be evaluated using the spindle interface.

Simultaneity:

Several axes can be traversed simultaneously or subject to a delay by the blocks FC 15, 16 and 18. The upper limit is defined by the maximum number of axes. The NCK handles the PLC function request (FC 15, 16, 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 15, FC 16, FC 18) are 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, value from MD35300: SPIND_POSCTRL_VELO (position control activation speed) is used
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 stage	
InPos:	Error :	With positioning error = "1"
State:	Error code	

The oscillation speed is taken from machine data:
MD35400 SPIND_OSCILL_DES_VELO

MD35010 GEAR_STEP_CHANGE_ ENABLE = 0	Function	MD35010: GEAR_STEP_CHA NGE_ENABLE = 1	Function
Pos = 0	Oscillation	Pos = 0	
Pos = 1	Oscillation	Pos = 1	Oscillation with gear stage change M41
Pos = 2	Oscillation	Pos = 2	Oscillation with gear stage change M42
Pos = 3	Oscillation	Pos = 3	Oscillation with gear stage change M43
Pos = 4	Oscillation	Pos = 4	Oscillation with gear stage change M44
Pos = 5	Oscillation	Pos = 5	Oscillation with gear stage change M45

4. Traverse indexing axis:

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

```
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	I/O	Type	Value range	Remarks
Start	I	Bool		Start spindle control from PLC
Stop	I	Bool		Stop spindle control from PLC
Funct	I	Byte	1 to B#16#0B	1: Position spindle 2: Rotate spindle 3: Oscillate spindle 4: Indexing axis 5: PosAxis metric 6: PosAxis inch 7: PosAxis metric with handwheel override 8: PosAxis inch with handwheel override 9: Rotate spindle with gear stage selection A: Rotate spindle with constant cutting rate (m/min) B: Rotate spindle with constant cutting rate (feet/min)

Signal	I/O	Type	Value range	Remarks
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: Rotational direction 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	Q	Bool		1 = Position reached, or function executed
Error	Q	Bool		1 = Error
State	Q	Byte	0 to 255	Error code

FRate

The feed rate in FC 18 can also be specified as:

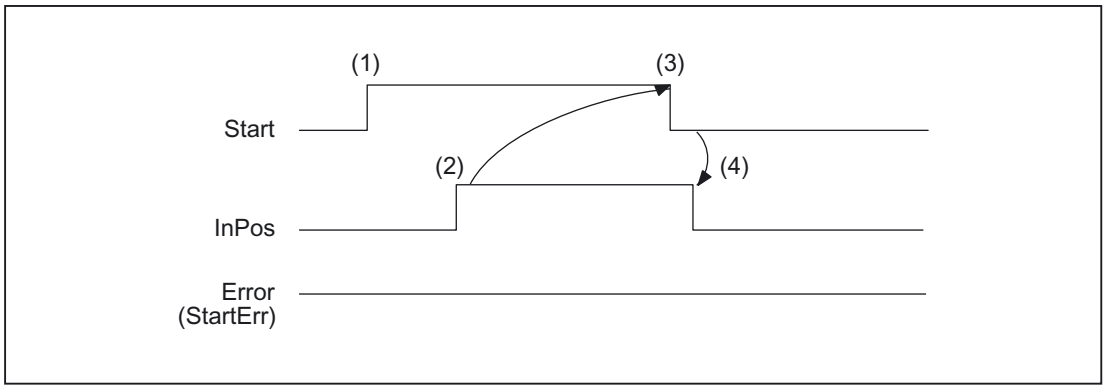
1. Cutting rate with unit m/min or feet/min
2. Constant grinding wheel surface speed in m/s or feet/s

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

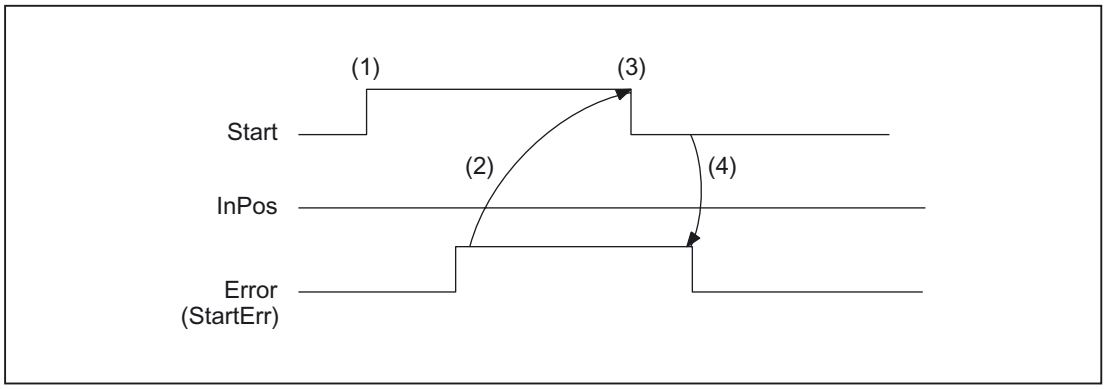
If a function could not be executed, this is indicated by the "Error" status parameter being set to 'logic 1'. The error cause is coded at block output "State". For a list of the error codes, see the table in the description of FC 15.

Timing diagram



(1) Activation of function by means of a positive signal edge with start or stop
(2) Positive acknowledgment: Function executed/Position reached
(3) Reset function activation after receipt of acknowledgment
(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 acknowledgment: Error has occurred
(3) Reset function activation after receipt of acknowledgment
(4) Signal change using FC

Call examples

1. Position spindle:

```
//Positive acknowledgment resets Start:
U M112.0;            //InPos
R M 100.0;           //Start
//Negative acknowledgment, 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 :=          Bq16q1,           //Position spindle
           Mode :=           Bq16q2,           //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 :=          Bq16q2,           //Rotate spindle
           Mode :=           Bq16q5,           //Rotational direction 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);
```

2.12.24 FC 19: MCP_IFM transmission of MCP signals to interface

Description of functions

With FC MCP_IFM (M variants), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- Axis selections
- WCS/MCS switchover commands
- Traversing keys
- Overrides
- Key switch

In the basic program (FC 2), FC 27 transmits handwheel selections, modes and other operation signals from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel.

Transfer of MMC 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 in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC 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 geoaxis 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 MMC** are decoded and activated in the (machine) axis or the Geo 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 acknowledgment.

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 MCP fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

With **SW V4 and higher**, multiple calls of FC 19 or FC 25 are permitted within the same 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.

With SW 5 and higher, the INC selections are only transferred to the mode group interface. This results in runtime improvements. The activation for this command is performed by this block once after powerup via DB10.DBX 57.0. Machine control panels can still be handled in parallel by this module. The module 2 call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st MCP. A support for two MCPs is still provided within certain restrictions in the control panel blocks (support is not provided for standard axis numbers 10 to 31, mutual interlocking of axis selections with two MCPs).

Flexible axis configuration

With SW V6 and higher, it is possible to be flexible in the assignment of axis selections or direction keys of machine axis numbers.
Better support is now provided by the MCP blocks for use of two MCPs, which are operated simultaneously, 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. The table starts from byte 8 (symbolic name: MCP1AxisTbl[1..22]) for the first machine control panel (MCP) and from 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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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 the restriction for the configured number of axes.

Declaration of the function

```
FUNCTION FC 19: void
//NAME                                  : MCP_IFM
VAR_INPUT
                     BAGNo :             BYTE ;
                     ChanNo:             BYTE ;
                     SpindleIFNo:        BYTE ;
END_VAR
VAR_OUTPUT
                     FeedHold :          BOOL ;
                     SpindleHold :       BOOL ;
END_VAR
BEGIN
END_FUNCTION
```

Explanation of the formal parameters

The table below shows all formal parameters of the "MCP_IFM" function:

Signal	I/O	Type	Value range	Remarks
BAGNo	I	Byte	0 - b#16# 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	Q	Bool		Feed stop from machine control panel, modal
SpindleHold	Q	Bool		Spindle stop from machine control panel, modal

MCP selection signals to the user interface

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

Operating modes and machine functions

Source: MCP button	Destination: Interface DB (Parameter ChanNo)
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. **(SW 5 and higher)**	DB11.DBB2, Bits 0 to 5

SW 4 and lower

Source: MCP button	Destination: Interface DB (Parameter ChanNo)
INC 1 ... 10 000, INC Var.	DB21, ... DBB13, Bit 0 to 5
INC 1 ... 10 000, INC Var.	DB21, ... DBB17, Bit 0 to 5
INC 1 ... 10 000, INC Var.	DB21, ... DBB21, Bit 0 to 5

Source: MCP button	Destination: Interface DB (all axis DBs)
INC 1 ... 10 000, INC Var. **(up to SW 4)**	DB31, ... DBB5, Bit 0 to 5

Direction keys rapid traverse override

The transfer is dependent upon the selected axis. The associated interface bits are deleted for axes, which are not selected.

Source: MCP button	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 button	Destination: Interface DB (all axis DBs)
Direction key +	DB31, ... DBX4.7
Direction key -	DB31, ... DBX4.6
Rapid traverse override	DB31, ... DBX4.5

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 applies to all axes.
Spindle override	DB31, ... DBB19 (parameter SpindleIFNo)

Channel signals

Source: MCP button	Destination: Interface DB (Parameter ChanNo)
NC Start	DB21, ... DBX7.1
NC stop	DB21, ... DBX7.3
Reset	DB21, ... DBX7.7
Single block	DB21, ... DBX0.4

Feedrate, spindle signals

Source: MCP button	Destination: FC output parameters
Feed stop Feed enable	Parameter: "FeedHold" latched, LEDs are driven
Spindle stop Spindle enable	Parameter: "SpindleHold" latched, LEDs are driven

Checkback signals from user interface for controlling displays

Operating modes and machine functions

Destination: MCP LED	Source: Interface DB (parameter ModeGroupNo)
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

All signals are ORed to obtain the LED checkback signals of the INC selections.

Destination: MCP LED	Source: Interface DB (parameter ChanNo)
INC 1 ... 10 000, INC Var.	DB21, ... DBB41, Bit 0 to 5
INC 1 ... 10 000, INC Var.	DB21, ... DBB47, Bit 0 to 5
INC 1 ... 10 000, INC Var.	DB21, ... DBB53, Bit 0 to 5

Destination: MCP LED	Source: Interface DB (all axis DBs)
INC 1 ... 10 000, INC Var.	DB31, ... DBB65, Bit 0 to 5

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 to 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 required flexibility can be obtained by applying the solution described below.

1. Before FC 19 is called, the information (RLO) relating to the newly defined axis selection key is transferred to the key selection identified by an axis number.
2. After FC 19 has been called, the information (RLO) relating to the LED identified by an axis number is transferred to the LED of the new axis selection key and the RLO of the previous axis LED then deleted.

Example:

The spindle is defined as the 4th axis and must be selected via axis key 9.

STL extract:

```
u i 5.2;                            //Selection of ninth axis
= e 4.2;                            //Selection of fourth axis
    call fc 19(
                    BAGNo :=            b#16#1,
                    ChanNo :=           b#16#1,
                    SpindleIFNo :=      b#16#4,
                    FeedHold :=         m 30.0,
                    SpindleHold :=      m 30.1);
u a 2.5;                            //LED fourth axis
= a 3.3;                            //LED ninth axis
clr;
= a 2.5;                            //Switch off LED on fourth axis
```

2.12.25 FC 21: transfer PLC NCK data exchange

Description of functions

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".

Declaration of 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	I/O	Type	Value range	Remarks
Enable	I	Bool		1 = FC 21 active
Funct	I	Byte	1.. 7	1: Synchronized actions to 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	Q	Bool		
ErrCode	Q	Int		Depends on "Funct"

Functions

1: Signals for synchronized actions to channel:

2: Signals for synchronized actions from channel:

Synchronized actions can be disabled or enabled by the PLC.
The data area is stored on the user interface in DB21 to DB30.DBB 300..307 (to channel) and DBB 308..315 (from channel). 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	I/O	Type	Value range	Remarks
Enable	I	Bool		1 = FC 21 active
Funct	I	Byte	1, 2	1: Synchronized actions to channel 2: Synchronized actions from channel
S7Var	I	Any	S7 data storage area	Not used
IVAR1	I	Int	1..MaxChannel	Channel number
Error	Q	Bool		
ErrCode	Q	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 DB 21
    SET;
    S DBX300.2;     //ID3
    S DBX301.1;     //ID10
    S DBX303.6;     //ID31
    L B#16#1;
    T MB11;
    SPA TRAN;
// Synchronized actions from NCK channel:
SYVK:               L B#16#2;
                    T MB11;
TRAN: CALL FC 21 (
          Enable    M 10.0,                          // if True, FC 21 is active
```

```
          :=
          Funct :=  MB 11,
          S7Var :=  #myAny,                          //Not used
          IVAR1 :=  1,                               //Channel no.
          IVAR2 :=  0,
          Error :=  M 10.1,
          ErrCode   MW 12);
          :=
...
END_FUNCTION
```

Functions

3, 4: Rapid PLC NCK data exchange

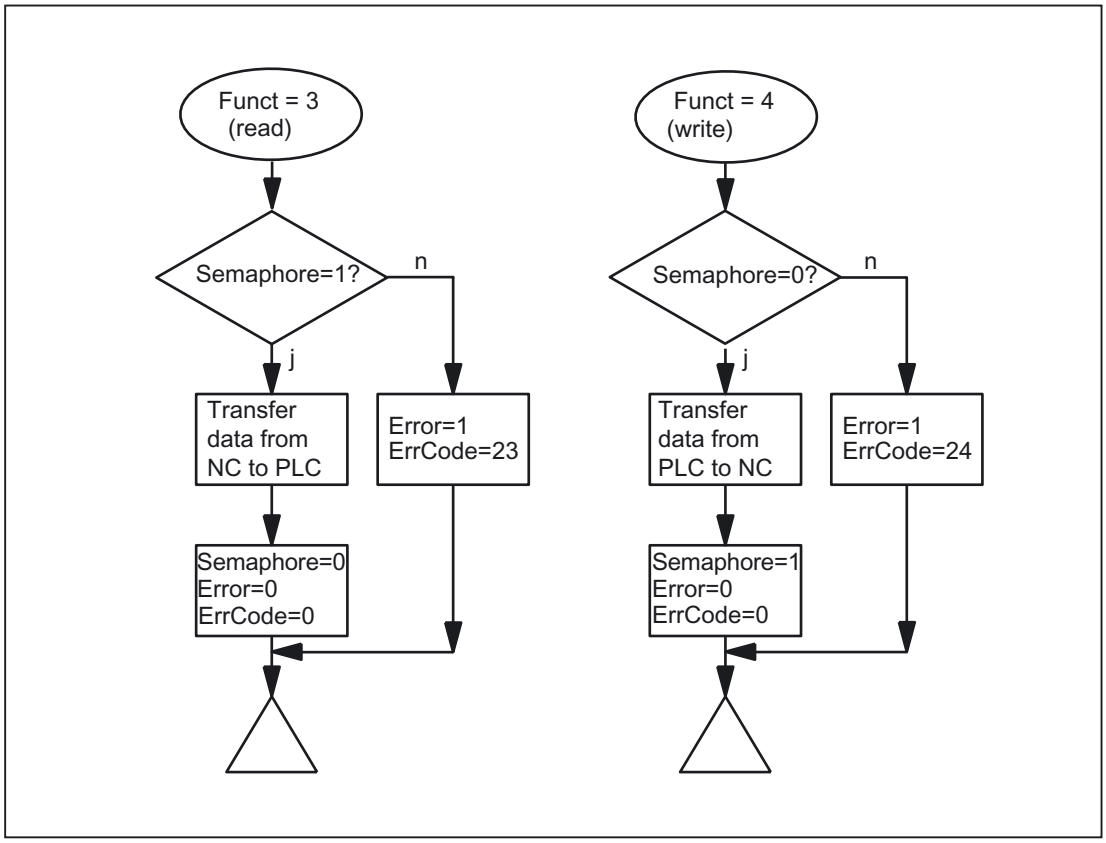
General

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 1024 bytes. The NCK is accessed (read/written) from the PLC via FC21. The assignment of this area (structure) must be declared identically in both the NC parts 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 Programming Guide). The concrete address is the data field is specified by a byte offset (0 to 1023) 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.

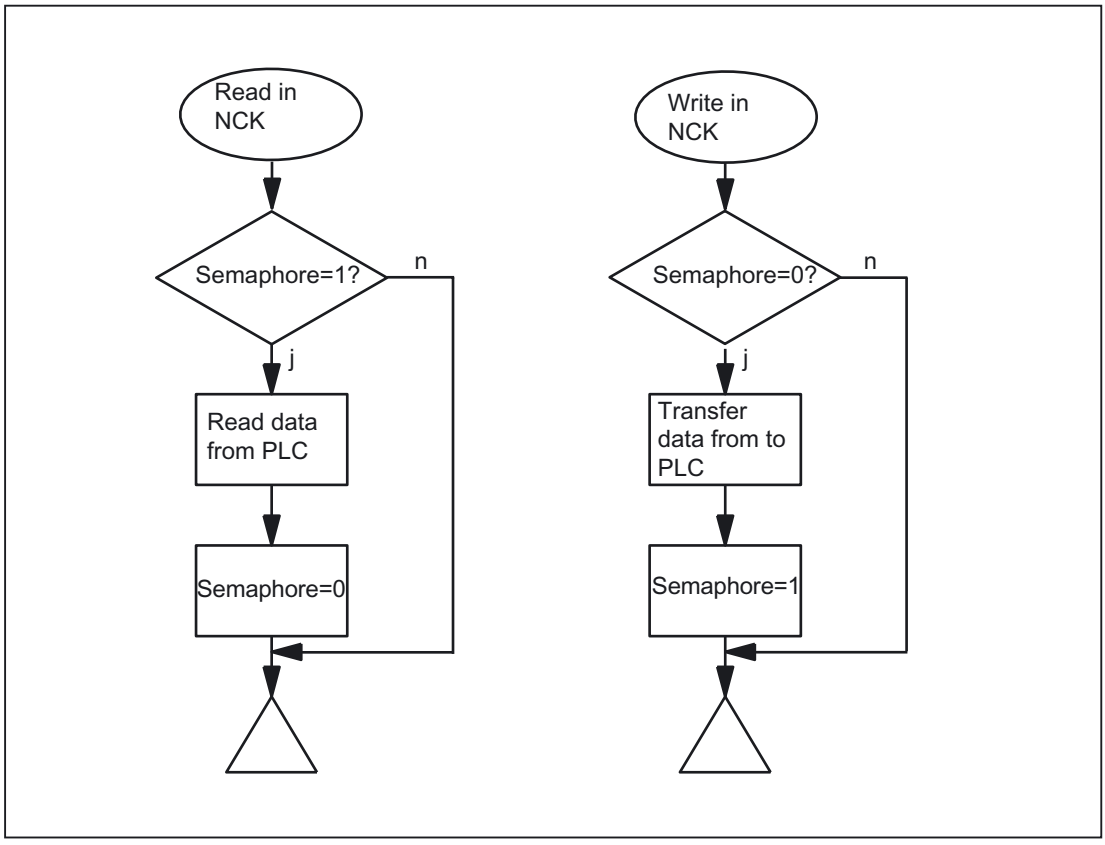
Without additional programming actions, data consistency is only ensured for 1 and 2 byte access in the NCK and in the PLC. However, 2 byte consistency is only ensured for the data types Word or Int, not for the data type Byte. In the case of longer data types or transfer of fields, which should be transferred 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 1023 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 = -1, data are transferred without a semaphore.

Data exchange with semaphore in PLC (schematic of FC 21)



Basic structure in NCK:



Variable value ranges

The following signals are relevant:

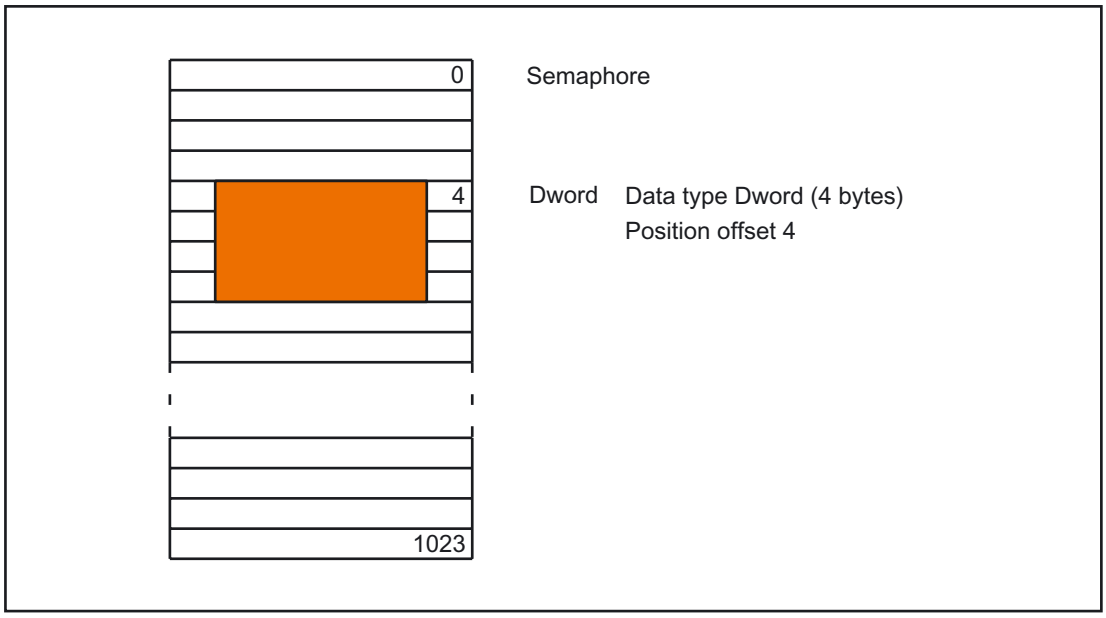
Signal	I/O	Type	Value range	Remarks
Enable	I	Bool		= FC 21 active
Funct	I	Byte	3 ,4	3: Read data 4: Write data
S7Var	I	Any	S7 data area, except local data	Source/destination data storage area
IVAR1	I	Int	0..1023	Position offset
IVAR2	I	Int	-1 .. 1023	Semaphore byte Transfer without semaphore: -1
Error	Q	Bool		
ErrCode	Q	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 MD 100:

Data type Dword (4 bytes)

Position offset 4




```
CALL FC 21 (
           Enable :=                        M 10.0,             // if True, FC 21
                                                                is 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 F10.0;
```

Examples of NCK programming:

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
                                                           is 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 function 5 is to transmit important control signals at high speed in between cyclic data transfers. Data bytes 6 and 7 of user interfaces DB21 to DB30 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	I/O	Type	Value range	Remarks
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	Q	Bool		
ErrCode	Q	Int		1: "Funct" invalid 10: Channel no. invalid

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. Data byte 2 of application interface DB31 to DB61 is transferred to the NC. The transfer is performed for all activated axes. This allows the servo enable to be transferred outside the PLC cycle, for example.

The following signals are relevant:

Signal	I/O	Type	Value range	Remarks
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	Q	Bool		
ErrCode	Q	Int		1: "Funct" invalid

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. Data byte 4 of application interface DB31 to DB61 is transferred to the NC. The transfer is performed for all activated axes. This function can be used, for example, to transfer a feed stop outside the PLC cycle.

The following signals are relevant:

Signal	I/O	Type	Value range	Remarks
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	Q	Bool		
ErrCode	Q	Int		1: "Funct" invalid

2.12.26 FC 22: TM_DIR direction selection for tool management

Description of functions

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 be called only in conjunction with tool management or after a DB 74 data block has been set up as described below in the example. In this example there are two magazines. The first magazine has 10 locations and the second has 12 locations. When adapting to the real machine, you only need to change the data AnzMag, MagNo[.].

```
DATA_BLOCK DB 74
STRUCT
    P: ARRAY[1 to 9] of DINT
    w1: WORD ;
    AnzMag: BYTE ;
    res:BYTE;
         MagNo: array [1 to 16] of struct //Byte 40
         AnzPlatz:INT;
         res1:BYTE;
         res2:BYTE;
    end_struct;
end_struct
BEGIN
    P[4]:=L#320;                          //Absolutely essential
    AnzMag:=b#16#2;                       //Total number of magazines = 2
    MagNo[1].AnzPlatz:=10;                //Number of locations for magazine 1 = 10
    MagNo[2].AnzPlatz:=12;                //Number of locations for magazine 2 = 12
END_DATA_BLOCK
```

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
```

Explanation of the formal parameters

The table below shows all formal parameters of the "TM_DIR" function.

Signal	I/O	Type	Value range	Remarks
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	Q	Bool		1 = Move magazine clockwise
Ccw	Q	Bool		1 = Move magazine counterclockwise
InPos	Q	Bool		1 = In position
Diff	Q	Int	0..	Differential path (shortest path)
Error	Q	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 clockwise 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
             );
```

2.12.27 FC 24: MCP_IFM2 transmission of MCP signals to interface

Description of functions

With FC MCP_IFM2 (M variant slim operator panel), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- Axis selections
- WCS/MCS switchover
- Traversing keys
- Overrides or override simulation signals

In the basic program (FC 2), FC 27 transmits handwheel selections, modes and other operation signals from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel.

Transfer of MMC 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 cyclical 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 in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC 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 geoaxis interface of the parameterized channel.

  – When the system is switched between MCS and WCS, the active axes are generally deselected.

The **handwheel selection signals of the MMC** are decoded and activated in the associated (machine) axis interface or geo axis interface of the relevant handwheel (only if FB 1 parameter "HWheelMMC := TRUE").

The associated LEDs of the machine control panel are derived from the acknowledgments 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 are permitted within the PLC cycle from **SW 4 and higher**. In this case, the first call in the cycle drives the LED displays. Moreover, all actions of the parameterized block are performed 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.

With SW 5 and higher, the INC selections of the override switch are only transferred to the mode group interface. This results in runtime improvements. The activation for this command is performed by this block once after powerup via DB10.DBX 57.0. Machine control panels can still be handled in parallel by this module. The module 2 call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st MCP. Support for of two MCPs is still provided within certain restrictions in the control panel blocks (the standard software does not support axis numbers 10 to 31, mutual interlocking of axis with two MCPs).

Flexible axis configuration

With SW V6 and higher, it is possible to be flexible in the assignment of axis selections or direction keys of machine axis numbers.
Better support is now provided by the MCP blocks for use of two MCPs, which are operated simultaneously, 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. The table starts from byte 8 (symbolic name: MCP1AxisTbl[1..22]) for the first machine control panel (MCP) and from 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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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 the restriction for 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
```

Explanation of the formal parameters

The table below shows all formal parameters of the "MCP_IFM2" function:

Signal	I/O	Type	Value range	Remarks
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. ModeGroupNo >= 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	Q	Bool		Feed stop from machine control panel, modal
SpindleHold	Q	Bool		Spindle stop from machine control panel, modal
SpindleDir	Q	Bool		Spindle direction 0 equals + (counterclockwise) 1 equals - (clockwise)

Call example

```
CALL FC 24 (          //Slim 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
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 to 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.

2.12.28 FC 25: MCP_IFT transfer of MCP/OP signals to interface

Description of functions

With FC MCP_IFT (T variants), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- Direction keys of four axes
- WCS/MCS switchover commands
- Overrides
- Key switch

In the basic program (FC 2), FC 27 transmits handwheel selections, modes and other operation signals from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel. Transmission of MMC signals to the interface can be deactivated by setting the value "FALSE" in the parameter in FB 1 (DB 7) "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 in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC 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 geoaxis interface of the parameterized channel.

The **handwheel selection signals from the MMC** are decoded and activated in the (machine) axis or the Geo axis interface of the handwheel selected (only if parameter "HWheelMMC := TRUE" in FB 1).

The associated LEDs of the machine control panel are derived from the acknowledgments 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.

In the case of machine control panel failure, the signals it supplies are set to zero.

With **SW V4 and higher**, multiple calls of FC 25 or FC 19 are permitted within the same 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.

With **SW 5 and higher**, the **INC selections** of the override switch are only transferred to the BAG group. This results in runtime improvements. The activation for this command is performed by this block once after powerup via DB10.DBX 57.0. Machine control panels can still be handled in parallel by this module. Support for of two MCPs is still provided within certain restrictions in the control panel blocks (the standard software does not support axis numbers 10 to 31, mutual interlocking of axis with two MCPs).

Flexible axis configuration

With SW V6 and higher, it is possible to be flexible in the assignment of axis selections or direction keys of machine axis numbers.
Better support is now provided by the MCP blocks for use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups.
The module 2 call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st 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 afford this flexibility, tables for axis numbers are stored in DB 10. The table starts from byte 8 (symbolic name: MCP1AxisTbl[1..22]) for the first machine control panel (MCP) and from 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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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 the restriction for the configured number of axes.

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
```

Explanation of the formal parameters

The table below shows all formal parameters of the "MCP_IFT" function:

Signal	I/O	Type	Value range	Remarks
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. ModeGroupNo >= 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	Q	Bool		Feed stop from machine control panel, modal
SpindleHold	Q	Bool		Spindle stop from machine control panel, 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 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 to 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.

2.12.29 FC 26: HPU_MCP Transfer of HPU/HT6 signals to the interface

2.12.29.1 General

Description of functions

With FC HPU_MCP (machine control panel signals of the hand-held programming device), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- WCS/MCS switchover
- Traversing keys
- Override

In the basic program (FC 2), FC 27 transmits handwheel selections, modes and other operation signals from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel.

Transfer of MMC 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 in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC 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 (for 6 axes).

  – When the WCS is selected, the signals of the first three axes are transferred to the Geo axis interface of the parameterized channel. The remaining three axes are transferred to the interface of the selected machine axis.

The **handwheel selection signals of the MMC** are decoded and activated in the associated (machine) axis interface or geo axis interface of the relevant handwheel (only if FB 1 parameter "HWheelMMC := TRUE").

The LEDs on the machine control panel derived from the selections in the acknowledgment.

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.

In the case of a failure of the MCP, the signals it supplies are set to zero.

With **SW V4 and higher**, multiple calls of FC 19 or FC 25 are permitted within the same PLC cycle. In this case, the first call in the cycle drives the LED displays. Moreover, all actions of the parameterized block are performed 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.

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.

With **SW 5 and higher**, the **INC selections** of the override switch are only transferred to the BAG group. This results in runtime improvements. The activation for this command is performed by this block once after powerup via DB10.DBX 57.0. 2 MCPs can still be handled in parallel by this module. The module 2 call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st MCP. Support for of two MCPs is still provided within certain restrictions in the control panel blocks (the standard software does not support axis numbers 10 to 31, mutual interlocking of axis with two MCPs).

Flexible axis configuration

With SW V6 and higher, it is possible to be flexible in the assignment of axis selections or direction keys of machine axis numbers.
Better support is now provided by the MCP blocks for use of two MCPs, which are operated simultaneously, 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. The table starts from byte 8 (symbolic name: MCP1AxisTbl[1..22]) for the first machine control panel (MCP) and from 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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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 the restriction for the configured number of axes.

Declaration of the function

```
FUNCTION FC 26: void
// NAME:                                    HPU_MCP
VAR_INPUT
    BAGNo :                                 BYTE ;
    ChanNo:                                 BYTE ;
END_VAR
BEGIN
END_FUNCTION
```

Explanation of the formal parameters

The table below shows all formal parameters of the "HPU_MCP" function.

Signal	I/O	Type	Value range	Remarks
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	B#16#0A	Channel no. for the channel signals

2.12.29.2 MCP selection signals to the user interface

Operating modes and machine functions

Source: MCP key	Destination: Interface DB (parameter ModeGroupNo) representation for mode group 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. **(SW 5 and higher)**	DB11.DBB2, Bits 0 to 5

SW 4 and lower

Source: MCP button	Destination: Interface DB (Parameter ChanNo)
INC 1 ... 10 000, INC Var.	DB21, ... DBB13, Bit 0 to 5
INC 1 ... 10 000, INC Var.	DB21, ... DBB17, Bit 0 to 5
INC 1 ... 10 000, INC Var.	DB21, ... DBB21, Bit 0 to 5

Source: MCP button	Destination: Interface DB (6 axis DBs)
INC 1 ... 10 000, INC Var. **(up to SW 4)**	DB31, ... DBB5, Bit 0 to 5

Direction keys rapid traverse override

The transfer is dependent upon the selected axis. The corresponding interface bits are deleted for axes that are not selected.

Source: MCP button	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 button	Destination: Interface DB (6 assigned axis DBs)
Direction key +	DB31, ... DBX4.7
Direction key -	DB31, ... DBX4.6
Rapid traverse override	DB31, ... DBX4.5

Override

Source: MCP setting	Destination: Interface DB (Parameter ChanNo)
Feedrate override	DB21, ... DBB4

Source: MCP setting	Destination: Interface DB (6 assigned axis DBs)
Feedrate override	DB31, ... DBB0
Spindle override	DB31, ... DBB19 (parameter SpindleIFNo)

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

2.12.29.3 Checkback signals from user interface for controlling displays

Operating modes and machine functions

Operating modes and machine functions	
Destination: MCP output	Source: Interface DB (parameter ModeGroupNo) representation for mode group 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

WCS/MCS output is operated through key actuation.

Call example

```
CALL FC 26 (                    //Machine control panel of the HPU/HT6
                                //signals to interface
    BAGNo :=    B#16#1,         //Mode group no. 1
    ChanNo :=   B#16#1);        //Channel no. 1
```

With these parameter settings, the signals from the first parameterized machine control panel are sent to the 1st mode group, the 1st channel and all axes.

2.12.30 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 HPU/HT6 variant)

Associated blocks

DB 7 (this was DB 5, up to SW 3), mode group number, channels, axes

DB 7, pointer of machine control panel

DB 8, storage for the next cycle

FC 20, output of error messages

Resources used

None.

General

Blocks FC 19 (M version), FC 24 (slim-line version), FC 25 (T version) and FC 26 (HPU/HT6 version) transfer 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. "ChanNo" selects the channel to be processed. The "SpindleIFNo" parameter defines the axis interface of the spindle. The spindle override is transferred to this spindle interface. The parameters are checked for errors. 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 descriptions

All four blocks have a similar structure. The blocks are organized into separate sections for individual subtasks.
In the input network, various parameters are copied into 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.

The MCS/WCS switchover with edge evaluation, axis selections, direction keys and rapid traverse overlay are determined in Network Global_in for further processing in the block. Userspecific modifications should be made to this Section of the program. The userspecific modifications will usually mainly involve axis selections.

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. If the mode group number is 0, this network is not processed. If the number is too large, message 401901 or 402501 is output and the control switches to Stop mode.

In the channel network, INCREMENT selections of the geo axes are transferred to the interface of all geo axes dynamically by the pushbutton switches (up to SW 4). The NC Start, Stop, Reset and SingleBlock 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. If the number is too large, message 401902 or 402502 is output and the control switches to Stop mode.

The spindle network transfers the spindle override into the interface configured via SpindleIFNo.

The axis network transfers the feedrate override to the selected axis interface. The direction keys are assigned to the selected axis/spindle. If an axis was selected previously, the direction information is set to 0. The INC checkback of the selected axis is displayed.

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

Network Global_in must be modified for more than 9 axes. If other keys and LEDs are to be used on the machine control panel here, proceed as follows:

1. The UD DW#16#Wert command deletes all LEDs defined for axis selections.
   The bit mask is currently processing the 9 axis selection LED.
2. The UW W#16# command, with the comment "Mask all axis selection keys" checks for a new selection of direction. 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.

2.13 Signal/data descriptions

2.13.1 Interface signals NCK/PLC, MMC/PLC, MCP/PLC

General

The NCK/PLC, MMC/PLC, and MSTT/PLC interface signals are listed in the list manual for SINUMERIK 840D with references to the respective function description where the signals are described.

References:
/LIS/ Lists

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.

2.13.2 Decoded M signals

General

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: DB21
- M functions from channel 2: DB22
- etc.

The signal length is one PLC cycle.

Note

The spindle-specific M functions below are not decoded: M3, M4, M5, and M70.

Addresses in DB21, ...	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. In order to securely identify the end of a part program in the channel, DB21, ... DBX33.5 (M02/M30 active) must be evaluated. 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.

2.13.3 G functions

General

The G 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: DB21
- G functions from channel 2: DB22
- 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 in the 8 G groups specified in the machine data element:
MD22510 $NC_ GCODE_GROUPS_TO_PLC
are overwritten according to the initial setting defined via the machine data, as are the values programmed in the part program.

Addresses in DB21, ...	Variable	Type	Initial setting	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

Addresses in DB21, ...	Variable	Type	Initial setting	Comment
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
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 247	G_FKT_GR_40	Byte	0	Active G function of group 40
DBB 248	G_FKT_GR_41	Byte	0	Active G function of group 41
DBB 249	G_FKT_GR_42	Byte	0	Active G function of group 42
DBB 250	G_FKT_GR_43	Byte	0	Active G function of group 43
DBB 251	G_FKT_GR_44	Byte	0	Active G function of group 44
DBB 252	G_FKT_GR_45	Byte	0	Active G function of group 45
DBB 253	G_FKT_GR_46	Byte	0	Active G function of group 46
DBB 254	G_FKT_GR_47	Byte	0	Active G function of group 47
DBB 255	G_FKT_GR_48	Byte	0	Active G function of group 48
DBB 256	G_FKT_GR_49	Byte	0	Active G function of group 49
DBB 257	G_FKT_GR_50	Byte	0	Active G function of group 50
DBB 258	G_FKT_GR_51	Byte	0	Active G function of group 51
DBB 259	G_FKT_GR_52	Byte	0	Active G function of group 52
DBB 260	G_FKT_GR_53	Byte	0	Active G function of group 53

Addresses in DB21, ...	Variable	Type	Initial setting	Comment
DBB 261	G_FKT_GR_54	Byte	0	Active G function of group 54
DBB 262	G_FKT_GR_55	Byte	0	Active G function of group 55
DBB 263	G_FKT_GR_56	Byte	0	Active G function of group 56
DBB 264	G_FKT_GR_57	Byte	0	Active G function of group 57

G functions (values)

A full list of all G functions can be found in:
References:
/PG/ Programming Guide, Basics

2.13.4 Message signals in DB 2

General

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 MMC. Text can be stored in the MMC for each message number.
References:
/IAD/ Installation and Start-Up Guide; Message Numbers

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.

Channel areas in DB 2

Area	Address	Message number
Channel 1	DBX0.0 - DBX11.7	510.000 -- 510.231
Channel 1, geo axes	DBX12.0 - DBX17.7	511.100 - 511.315
Channel 2	DBX18.0 - DBX29.7	520.000 - 520.231
Channel 2, geo axes	DBX30.0 - DBX35.7	521.100 - 521.315
Channel 3	DBX36.0 - DBX47.7	530.000 - 530.231
Channel 3, geo axes	DBX48.0 - DBX53.7	531.000 - 531.315
Channel 4	DBX54.0 - DBX65.7	540.000 - 540.231
Channel 4, geo axes	DBX66.0 - DBX71.7	541.100 - 541.315

Area	Address	Message number
Channel 5	DBX72.0 - DBX83.7	550.000 - 550.231
Channel 5, geo axes	DBX84.0 - DBX89.7	551.100 - 551.315
Channel 6	DBX90.0 - DBX101.7	560.000 - 560.231
Channel 6, geo axes	DBX102.0 - DBX107.7	561.100 - 561.315
Channel 7	DBX108.0 - DBX119.7	570.000 - 570.231
Channel 7, geo axes	DBX120.0 - DBX125.7	570.100 - 571.315
Channel 8	DBX126.0 - DBX137.7	580.000 - 580.231
Channel 8, geo axes	DBX138.0 - DBX143.7	581.100 - 581.315
Channels 9 and 10 are not currently implemented		

User areas in DB 2

Area	Address	Message number
Axis/spindle 1	DBX144.0 - DBX145.7	600.100 - 600.115
Axis/spindle 2	DBX146.0 - DBX147.7	600.200 - 600.215
Axis/spindle 3	DBX148.0 - DBX149.7	600.300 - 600.315
Axis/spindle 4	DBX150.0 - DBX151.7	600.400 - 600.415
Axis/spindle 5	DBX152.0 - DBX153.7	600.500 - 600.515
Axis/spindle 6	DBX154.0 - DBX155.7	600.600 - 600.615
Axis/spindle 7	DBX156.0 - DBX157.7	600.700 - 600.715
Axis/spindle 8	DBX158.0 - DBX159.7	600.800 - 600.815
Axis/spindle 9	DBX160.0 - DBX161.7	600.900 - 600.915
Axis/spindle 10	DBX162.0 - DBX163.7	601.000 - 601.015
Axis/spindle 11	DBX164.0 - DBX165.7	601.100 - 601.115
Axis/spindle 12	DBX166.0 - DBX167.7	601.200 - 601.215
Axis/spindle 13	DBX168.0 - DBX169.7	601.300 - 601.315
Axis/spindle 14	DBX170.0 - DBX171.7	601.400 - 601.415
Axis/spindle 15	DBX172.0 - DBX173.7	601.500 - 601.515
Axis/spindle 16	DBX174.0-DBX175.7	601.600 - 601.615
Axis/spindle 17	DBX176.0 - DBX177.7	601.700 - 601.715
Axis/spindle 18	DBX178.0 - DBX179.7	601.800 - 601.815
Axes 19 to 31 are not currently implemented		

User areas in DB 2

Area	Address	Message number
User area 0	DBX180.0 - DBX187.7	700.000 - 700.063
User area 1	DBX188.0 - DBX195.7	700.100 - 700.163
User area 2	DBX196.0 - DBX203.7	700.200 - 700.263
User area 3	DBX204.0 - DBX211.7	700.300 - 700.363
User area 4	DBX212.0 - DBX219.7	700.400 - 700.463
User area 5	DBX220.0 - DBX227.7	700.500 - 700.563
User area 6	DBX228.0 - DBX235.7	700.600 - 700.663
User area 7	DBX236.0 - DBX243.7	700.700 - 700.763
User area 8	DBX244.0 - DBX251.7	700.800 - 700.863
User area 9	DBX252.0 - DBX259.7	700.900 - 700.963
User area 10	DBX260.0 - DBX267.7	710.000 - 701.063
User area 11	DBX268.0 - DBX275.7	710.100 - 701.163
User area 12	DBX276.0 - DBX283.7	710.200 - 701.263
User area 13	DBX284.0 - DBX291.7	710.300 - 701.363
User area 14	DBX292.0 - DBX299.7	710.400 - 701.463
User area 15	DBX300.0 - DBX307.7	710.500 - 701.563
User area 16	DBX308.0 - DBX315.7	710.600 - 701.663
User area 17	DBX316.0 - DBX323.7	710.700 - 701.763
User area 18	DBX324.0 - DBX331.7	710.800 - 701.863
User area 19	DBX332.0 - DBX339.7	710.900 - 701.963
User area 20	DBX340.0 - DBX347.7	702.000 - 702.063
User area 21	DBX348.0 - DBX355.7	702.100 - 702.163
User area 22	DBX356.0 - DBX363.7	702.200 - 702.263
User area 23	DBX364.0 - DBX371.7	702.300 - 702.363
User area 24	DBX372.0 - DBX379.7	702.400 - 702.463

2.14 Useful tips on programming with STEP 7

2.14.1 General

General

Some useful tips on programming complex machining sequences in STEP7 are given in the following. This information concentrates mainly on the handling of data type POINTER and ANY. Detailed information about the structure of data types POINTER and ANY can be found in Chapter "CPU register and storage of data" in STEP7 manual "Designing user programs".

2.14.2 Copying data

High-speed copying

The following is an example of how to copy data at high speed from one DB into another.

```
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               42;                 //Transfer 84 bytes
    M001:
    L               DBW [AR1,P#0.0];    //Copy word-oriented
    L               DBW [AR1,P#0.0];
    T               DIW [AR2,P#0.0];
    +AR1            P#2.0;
    +AR2            P#2.0;
    TAK;
    LOOP            M001;
```

2.14.3 ANY and POINTER

2.14.3.1 General

General

The following programming examples illustrate different programming mechanisms. 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.

2.14.3.2 Use of POINTER and ANY in FC if POINTER or ANY is available as parameter

Description of functions

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 cross area 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.

```
FUNCTION FC 99: VOID                                    Comment
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
    L               W [AR1,P#0.0];                      //Retrieve DB number
```

```
FUNCTION FC 99: VOID                                    Comment
    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
```

2.14.3.3 Use of POINTER and ANY in FB if POINTER or ANY is available as parameter

Description of functions

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 cross area 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.

```
FUNCTIONBLOCK FB 99                                     Comment
VAR_INPUT
    Row : BYTE ;
    Convert : BOOL ;     //Activate numerical conversion
    Addr: POINTER;       //Points to variable
    Addr1 : ANY ;
END_VAR
VAR_TEMP
```

```
FUNCTIONBLOCK FB 99                                          Comment
    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
```

2.14.3.4 POINTER or ANY variable for transfer to FC or FB

With version 1 or later of STEP7 it is possible to define a POINTER or an ANY in VAR_TEMP. The following two examples show how an ANY can be supplied.

1. 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 register is also used for multi-instance DBs. For this reason, the relevant FB must not be declared as a multiinstance DB.

```
FUNCTIONBLOCK FB 100                          Comment
CODE_VERSION1                                 //To deactivate multi-instance DB with STEP 7 Version
                                              2 and higher
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
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
```

1. 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

```
FUNCTIONBLOCK FB 100                          Comment
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
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
```

2.14.4 Multi-instance DB

Multi-instance DB

With version 2 and higher of STEP 7, FBs might have a multi-instance capability, i.e., they might incorporate multi-instance DBs. The primary characteristic of multi-instance DBs is that they can be used for various instances of FBs (see STEP 7 documentation), thus allowing the DB data quantity to be optimized.

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

For more complex programs using pointers and address registers in FBs, which are to be multi-instance capable, certain rules specified by the programmer must be complied with. With multi-instance 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 multi-instance 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. It is best to save the AR2 register of the instance in a local variable (VAR_TEMP).
The 'load pointer to an instance variable' command returns a pointer value at 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

```
FUNCTION_BLOCK FB 99                    Comment
VAR_INPUT
    varin: INT ;
END_VAR
VAR
    variable1: ARRAY[0 to 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. If cross-area
                                        processing is to take place, then an area should be skipped
                                        when these two pointers are added.
AD              DW#16#00FF_FFFF,        //Skipping of an area
LAR1                                    //Load into AR1
TAR2;
+AR1 AR2;                               //AR2 instance offset to be added
                                        //The ARRAY of variable1 can now be accessed indirectly via
                                        AR1.
L               DIW [AR1, P#0.0];       //E.g., access to first element
END_FUNCTION_BLOCK
```

2.14.5 Strings

General

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 STRING

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 STEP7 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 string is stored in byte 2 by the 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.

2.14.6 Determining offset addresses for data block structures

General

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.

```
FUNCTION FC 99: VOID                    Comment
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
```

Supplementary Conditions

3

3.1 Supplementary conditions

There are no supplementary conditions to note.

Examples

4

4.1 Examples

No examples are available.

Data Lists

5

5.1 Machine data

5.1.1 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)
Machine data in integer/hex format is operated in the NC as DWORD. Machine data in floating comma format are operated in the NC as FLOAT (IEEE 8 byte). They are stored in the NC/PLC interface and can be read by the PLC user program during PLC power-up from the DB 20.		

5.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
28150	MM_NUM_VDIVAR_ELEMENTS	Number of elements for writing PLC variables

Index

A

Assignment of DBs, 2-72
ASUBs, 2-51

B

Basic PLC program (P3)|Connecting the PLC CPUs, 2-23
Basic PLC Program (P3)|Physical interfaces on 840D, 2-24
Basic PLC program (P3)|PLC interface on SINUMERIK 840D, 2-24
Bus addresses on SINUMERIK 840D, 2-41

C

Concurrent axes, 2-50
Configurability of machine control panel, handheld unit, 2-62
Cyclic operation, 2-45
Cyclic signal exchange, 1-1

D

Diagnostic buffer on PLC, 2-27

E

Eventdriven signal exchange, 1-1, 1-2

F

FB 10
    Safety relay, 2-157
FB 11
    Brake test, 2-160
FB 2
    GET, 2-106
FB 29
    Signal recorder and data trigger diagnostics, 2-165
FB 3
    PUT, 2-113
FB 4
    PI_SERV General PI services, 2-120
FB 5
    GETGUD read GUD variable, 2-143
FB 7
    PI_SERV2 General PI services, 2-148
FB 9
    Operating unit switchover, 2-92
FB 9 M:N operating-unit switchover, 2-151
FC 10
    AL_MSG, 2-189
FC 13
    BHGDisp, 2-192
FC 15
    POS_AX, 2-195
FC 16
    PART_AX, 2-201
FC 17
    YDelta, 2-205
FC 18
    SpinCtrl, 2-208
FC 19
    MCP_IFM, 2-218
FC 22
    TM_DIR, 2-232
FC 24
    MCP_IFM2, 2-235
FC 25
    MCP_IFT, 2-239
FC 7
    TM_REV, 2-175
FC 8
    TM_TRANS, 2-179
FC 9
    ASUB, 2-186
FC2
    GP_HP, 2-169
FC3
    GP_PRAL, 2-170, 2-173

I

Interface
840D, 2-24
PLC/MCP, 2-39
PLC/MMC, 2-36

M

M decoding acc. to list, 2-54
MAXNUM_USER_DATA_FLOAT, 5-1
MAXNUM_USER_DATA_HEX, 5-1
MAXNUM_USER_DATA_INT, 5-1
MD14504, 2-59
MD14506, 2-59
MD14508, 2-59
MD35400, 2-211
Memory requirements of basic PLC program
Maximum, 2-77
Minimum, 2-76
Message signals in DB2, 2-254
MM_NUM_VDIVAR_ELEMENTS, 5-1
Mode group, 2-46

N

NC failure, 2-48
NC VAR selector, 2-80
Startup, installation, 2-91
NC variables, 2-86

P

PI services
Overview, 2-123
PLC CPUs, properties, 2-24
PLC messages, 2-37
PLC/NCK interface, 2-29
PLC_CYCLIC_TIMEOUT, 5-1
Process alarm processing, 2-48
Programming and parameterizing tools, 2-77
Programming devices or PCs, 2-77

R

Read/write NC variables, 2-52

S

Signals
PLC/axes, spindles, 2-35
PLC/mode group, 2-32
PLC/Mode group, 2-33
PLC/NC, 2-31
PLC/NCK channels, 2-33
Startup and synchronization of NCK PLC, 2-45
Symbolic programming of user program with interface DB, 2-52

U

Useful Tips on Programming with STEP 7, 2-257
Useful tips on programming with Step7
ANY and POINTER, 2-258
Multiinstance DB, 2-262
POINTER or ANY variable, 2-260
Strings, 2-264
Use of ANY and POINTER in FB, 2-259
Use of ANY and POINTER in FC, 2-258
USER_DATA_FLOAT[n], 5-1
USER_DATA_HEX[n], 5-1
USER_DATA_INT[n], 5-1

SIEMENS

SINUMERIK 840D sl

PLC Basic Program Solution Line (P3 sl)

Function Manual

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

Valid for

Control
SINUMERIK 840D sl/840DE sl

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

2 Detailed Description 2-1

2.1 Key PLC CPU data for 810D, 840D and 840Di 2-1

2.2 Reserve resources (timers, counters, FC, FB, DB, I/O) 2-6

2.3 Starting up hardware configuration of PLC CPUs 2-7

2.4 Starting up the PLC program 2-11
2.4.1 Installing the basic program for 840D 2-11
2.4.2 Application of basic program 2-11
2.4.3 Version codes 2-12
2.4.4 Machine program 2-13
2.4.5 Data backup 2-13
2.4.6 PLC series start-up, PLC archives 2-14
2.4.7 Software upgrades 2-16
2.4.8 I/O modules (FM, CP modules) 2-17
2.4.9 Troubleshooting 2-18

2.5 Linking PLC CPUs to 840D 2-19
2.5.1 General 2-19
2.5.2 Properties of PLC CPUs 2-19
2.5.3 Interface on 840D with integrated PLC 2-19
2.5.4 Diagnostic buffer on PLC 2-22

2.6 Interface structure 2-23
2.6.1 PLC/NCK interface 2-23
2.6.2 Interface PLC/HMI 2-30
2.6.3 PLC/MCP/HHU interface 2-33

2.7 Structure and functions of the basic program 2-35
2.7.1 Start-up and synchronization of NCK PLC 2-37
2.7.2 Cyclic operation (OB1) 2-37
2.7.3 Time-interrupt processing (OB 35) 2-40
2.7.4 Process-interrupt processing (OB 40) 2-40
2.7.5 Diagnostic interrupt, module failure processing (OB 82, OB 86) 2-40
2.7.6 Response to NCK failure 2-41
2.7.7 Functions of the basic program called from the user program 2-42
2.7.8 Symbolic programming of user program with interface DB 2-45
2.7.9 M decoding acc. to list 2-46
2.7.10 PLC machine data 2-50
2.7.11 Configuring machine control panel, handheld unit 2-54
2.7.12 Switchover of machine control panel, handheld unit 2-61
2.7.13 Special functions of the machine control panel 2-63

2.8 SPL for Safety Integrated 2-63

2.9 Assignment overview 2-64
2.9.1 Assignment: NCK/PLC interface 2-64
2.9.2 Assignment: FB/FC 2-64

2.9.3 Assignment: DB ... 2-64
2.9.4 Assignment: Timers ... 2-66

2.10 Memory requirements of basic PLC program for 810D, 840D ... 2-66

2.11 General conditions and NC-VAR_Selector ... 2-69
2.11.1 Supplementary conditions ... 2-69
2.11.1.1 Programming and parameterizing tools ... 2-69
2.11.1.2 SIMATIC documentation required ... 2-71
2.11.1.3 Relevant SINUMERIK documents ... 2-71
2.11.2 NC VAR selector ... 2-72
2.11.2.1 Overview ... 2-72
2.11.2.2 Description of functions ... 2-74
2.11.2.3 Start-up, installation ... 2-84

2.12 Block descriptions ... 2-84
2.12.1 FB 1: RUN_UP Basic program, start-up section ... 2-84
2.12.2 FB 2: Read GET NC variable ... 2-92
2.12.3 FB 3: PUT write NC variables ... 2-99
2.12.4 FB 4: PI_SERV General PI services ... 2-106
2.12.5 FB 5: GETGUD read GUD variable ... 2-129
2.12.6 FB 7: PI_SERV2 General PI services ... 2-134
2.12.7 FB 9: M : N operating-unit switchover ... 2-138
2.12.8 FB 10: Safety relay (SI relay) ... 2-143
2.12.9 FB 11: Brake test ... 2-146
2.12.10 FB 29: Signal recorder and data trigger diagnostics ... 2-151
2.12.11 FC 2: GP_HP Basic program, cyclic section ... 2-154
2.12.12 FC 3: GP_PRAL Basic program, interrupt-driven section ... 2-156
2.12.13 FC 5: GP_DIAG Basic program, diagnostic alarm, and module failure ... 2-158
2.12.14 FC 7: TM_REV Transfer block for tool change with revolver ... 2-160
2.12.15 FC 8: TM_TRANS transfer block for tool management ... 2-164
2.12.16 FC 9: ASUB start-up of asynchronous subprograms ... 2-171
2.12.17 FC 10: AL_MSG error and operating messages ... 2-174
2.12.18 FC 12: AUXFU call interface for user with auxiliary functions ... 2-176
2.12.19 FC 13: BHGDisp Display control for handheld unit ... 2-177
2.12.20 FC 17: YDelta star/delta changeover ... 2-180
2.12.21 FC 18: SpinCtrl spindle control ... 2-183
2.12.22 FC 19: MCP_IFM transmission of MCP signals to interface ... 2-194
2.12.23 FC 21: transfer PLC NCK data exchange ... 2-201
2.12.24 FC 22: TM_DIR Direction selection for tool management ... 2-208
2.12.25 FC 24: MCP_IFM2 transmission of MCP signals to interface ... 2-210
2.12.26 FC 25: MCP_IFT transfer of MCP/OP signals to interface ... 2-214
2.12.27 FC 26: HPU_MCP Transfer of HPU/HT6 signals to the interface ... 2-217
2.12.27.1 MCP selection signals to the user interface ... 2-219
2.12.27.2 Checkback signals from user interface for controlling displays ... 2-221
2.12.28 FC 19, FC 24, FC 25, FC 26 source code description ... 2-221
2.12.29 Signal/data descriptions ... 2-223
2.12.29.1 Interface signals NCK/PLC, MMC/PLC, MCP/PLC ... 2-223
2.12.29.2 Decoded M signals ... 2-223
2.12.29.3 G Functions ... 2-225
2.12.29.4 Message signals in DB 2 ... 2-226
2.12.30 Useful Tips on Programming with STEP 7 ... 2-229
2.12.30.1 General ... 2-229
2.12.30.2 Copying data ... 2-229
2.12.30.3 ANY and POINTER ... 2-229

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Examples 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Channel-specific machine data 5-1

Index Index-1

Tables

Table 2-1 Hardware expansions 2-7

Brief Description

1

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). A distinction is made between the following groups for signals and data:

- Cyclic signal exchange
- Eventdriven signal exchange
- Messages

Cyclic signal exchange

Signals exchanged cyclically consist primarily of bit fields.

- They contain **commands** transmitted 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 groups
  - 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 the auxiliary functions are used to start execution of a block, the type of auxiliary function determines whether the NCK has to wait before executing the function (e.g., during a tool change) or whether the function is executed in parallel to machining of the workpiece (e.g., for tool preparation on milling machines with chain-type magazines).

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 acknowledgment-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. Where the data do not require user acknowledgment, this does not affect NCK machining.

Event-driven signal exchange PLC → NCK

An "event-driven signal exchange PLC → NCK" takes place whenever the PLC passes a request to the NCK (e.g., traversal of an auxiliary axis). In this case, data transfer is also acknowledgment-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. A defined bit field is used to transfer the message signals to the basic program, where they are evaluated and, if message events occur, entered in the PLC's interrupt buffer by means of the ALARM S/SQ function. Where an HMI (e.g. HMI embedded) is provided, the messages are transferred to and displayed on the HMI.

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 HIM 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.

Detailed Description

2

2.1 Key PLC CPU data for 810D, 840D and 840Di

The tables below show the performance range of the PLC CPUs and the range of the PLC basic program for various control types.

Type of control: Key PLC CPU data for 840D

Type of control	840Di sl	840D sl
PLC CPU	Integrated PLC CPU317-2DP master/slave	Integrated PLC CPU317-2DP master/slave
MLFB	6FC5 317-2AJ10-0AB0	6FC5 317-2AJ10-0AB0
Memory for user and basic program	128 to 768KB	128 to 768KB
Data block memory	Max. of 256KB	Max. of 256KB
Memory submodule	No	No
Bit memories	32768	32768
Timers	512	512
Counters	512	512
Clock memories	8	8
Program/data blocks OB		
	1, 10, 20-21, 32-35, 40, 55-57, 80-82, 85-87, 90, 100, 121-122	1, 10, 20-21, 32-35, 40, 55-57, 80-82, 85-87, 90, 100, 121-122
FB	0-2048	0-2048
FC	0-2048	0-2048
DB	1-2048	1-2048
Max. data block length	64 KB	64 KB
Max. block length FC, FB	64 KB	64 KB

Type of control		840Di sl	840D sl
Inputs/outputs (address capacity in bytes): - digital + analog incl. reserved area process image Notice! The inputs/outputs above 4096 are reserved for integrated drives.		4096/4096 (8192/8192) 256/256	4096/4096 (8192/8192) 256/256
Inputs/outputs 1) (addressing) - digital - analog	Row 0 is integrated in the NCU. Rows 1 to 3 are available for I/O devices	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 288 onwards Profibus only	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 288 onwards
Processing time - Bit commands (I/O) - Word commands		0.03 ms/kA 0.1 ms/kA	0.03 ms/kA 0.1 ms/kA
PDIAG (Alarm S,SQ)		Yes	Yes
PROFIBUS		Master/Slave	Master/Slave
Number of PROFIBUS slaves		max. 125	max. 125
Max. number of PROFIBUS slots		512	512
DP master system no. DP		1	1
DP master system no. MPI/DP		2	2
DP master system no. internal Profibus (PCI)		-	3
PBC programmable block communication		Yes	Yes
Consistent data to standard slave via SFC 14, 15		128	128
1) Subrack 0 is integrated in the NC. Rows 1 to 3 are available for I/O devices			

I/O expansion

I/O modules, central	PROFIBUS only	24 (option)
PROFIBUS DP interfaces	1 (2)	1 (2)
Interfaces (MPI)	1 (0)	1 (0)

Note

Number of PROFIBUS slaves

The content of SDB 2000 and other associated SDBs is stored by the PLC operating system in internal data structures which can also be accessed by the PROFIBUS ASIC. Furthermore, information from SDB 2000 is also conditioned (CPI interface) and transmitted to the NCK and the PLC basic program. This is necessary in order to control drives and PROFIsafe modules on the PROFIBUS. A memory area defined by the PLC is available for these data structures. Its size is limited by the maximum number of slots. This means that during loading, SDBs with fewer slaves than listed above may be refused. A slot is usually a slave module or the slave itself. Only on a module with both I and Q areas does one module count as 2 slots. It is, therefore, not possible to specify the size of SDB 2000 exactly. It cannot be determined whether the configuration is legal until the SDB container has been loaded to the CPU. The values shown above must be taken as guide values only. If the configuration is illegal, a general reset request is issued when the SDBs are loaded. The cause of the configuring error can be found in the diagnostic buffer on completion of the general reset.

PLC versions

PLC 317-2DP with SIMATIC version 2.1.6 (version ID 20.70.24) or higher

is integrated. These versions are compatible with the corresponding SIMATIC CPU 300, which means that all modules and software packages that are approved for these versions and CPUs on SIMATIC are, therefore, suitable. Exceptions include modules that can usually only be inserted in row 0 (other exceptions are the FMNC and FM 357 modules).

The version ID on the version screen comprises the following components:

- SIMATIC CPU PLC primary version identifier
- Firmware transfer increment
- Internal increment

Example ID

317-2DP PLC with MLFB 6FC5 317-2AJ10-0AB0: 20.70.24

Version screen on HMI

The PLC currently being used and the associated version of the PLC operating system are displayed under PLC on the version screen,
for example
PLC_317-2DP 20.70.24

The module code of the PLC module used appears in the preceding column. The following PLC module codes are currently in use:

Module code	PLC module	Suitable PLC operating systems (corresponding SIMATIC MLFB)	Operating system SW version ID
2000	PLC 317-2DP with IBC32	6ES7 317-2AJ10-0AB0, FW 2.1	20.70.24
MCI 2 (840Di) 2100	PLC 317-2DP with IBC32	6ES7 317-2AJ10-0AB0, FW 2.1	20.70.24

PLC basic program functions

	840Di	810D sl
Interfaces		
MPI	1	1
Ethernet		2+1
Functions of basic program		
Scope:		
Axes/spindles	See Catalog	31
Channels	6	10
Mode groups	6	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	+	+
Async. subprogram 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	+	+

Mode selector on PLC CPUs for 840D

On the NCU component, the right-hand twist button labeled "PLC" is used to set the PLC operating mode.
The switch settings are listed in the following table.

Switch setting	Meaning	Remarks
0	RUN-P	
1	RUN	Cannot load program
2	STOP	
3	MRES	

2.2 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.

Counters

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 between 1000 and 1099 is also reserved for data blocks.
Data blocks from non-activated channels, axes/spindles and tool management are free for the user.

I/O range:

The PLC 317 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 STEP7. Furthermore, the address range between 256 and 287 is assigned for the NCK, CP and HMI in rack 0 on the SIMATIC 300 station.

2.3 Starting up hardware configuration of PLC CPUs

General procedure

The hardware configuration for the PLC CPUs used in the NCU7x0, including other components of the NCU (NCK, CP, HMI, drive), must be defined via STEP 7.
Proceed as follows in STEP7:

1. Create a new project (File, New, Project)
2. Insert, Hardware, SIMATIC 300 station
3. Select the SIMATIC 300 station with the mouse
4. Right-click with the mouse and select Open Object
   and start HWConfig
5. A suitable SINUMERIK component is selected from the "SIMATIC 300\SINUMERIK\840D sl" hardware catalog.
6. Add I/O from the STEP7 hardware catalog
   The addresses for the I/O modules can be modified if required.

Requirements

In order to be able to select SINUMERIK components from the hardware catalog, you must run the Toolbox setup program first ("Hardware expansions for STEP7" and "Starter"). The current version of the hardware expansion for STEP7 can also be found under eSupport.

Table 2-1 Hardware expansions

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
SINUMERIK 840D NCU 710.1	6FC5 371-0AA00-?AA0	6ES7 317-2AJ00-0AB0	NCU710.1 (as of STEP7 V5.3 SP2 and Toolbox 840D sl 01.01.00)
SINUMERIK 840D NCU 720.1	6FC5 372-0AA00-?AA0	6ES7 317-2AJ00-0AB0	NCU720.1 (as of STEP7 V5.3 SP2 and Toolbox 840D sl 01.01.00)
SINUMERIK 840D NCU 730.1	6FC5 373-0AA00-?AA0	6ES7 317-2AJ00-0AB0	NCU730.1 (as of STEP7 V5.3 SP2 and Toolbox 840D sl 01.01.00)
SINUMERIK 840Di sl with MCI2	6FC5 222-0AA02-1AA0	6ES7 317-2AJ00-0AB0	840Di with PLC317-2 DP (as of STEP7 V5.3 SP2 and Toolbox 840D sl 01.02.00)

Note

On SINUMERIK 840D, SIMATIC line 0 is allocated for the SINUMERIK components. The following components are inserted in this line:
The integrated PLC with the various bus systems (slot 2)
An IM 360 (slot 3)
The NCK 840D sl (slot 4). The properties (I/O address 256, process interrupt) of the NCK must not be changed, otherwise process interrupts (e.g., auxiliary functions) from the NCK to the PLC will no longer function.
The integrated Ethernet CP 840D sl (slot 5)
The integrated HMI 840D sl (slot 6)

The integrated drive is linked on the internal PCI bus (PROFIBUS protocol).

Currently, this bus does not support I/O access, and SFCs are not supported for this bus.

If you are using a different NCU, you can change the NCU using drag & drop. The settings remain unaffected.

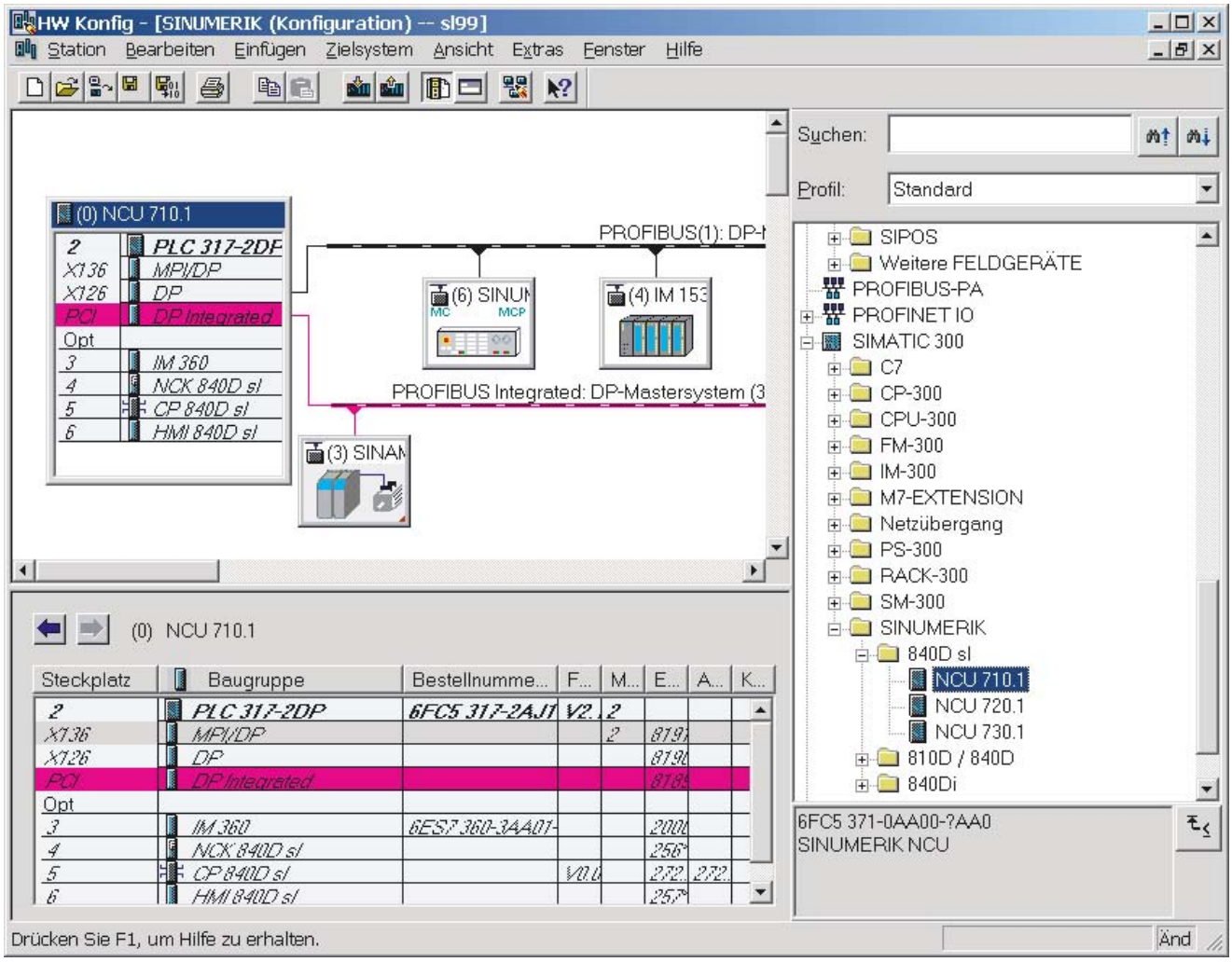


Fig. 2-1 Hardware configuration on the 840D sl and SINAMICS Properties dialog box

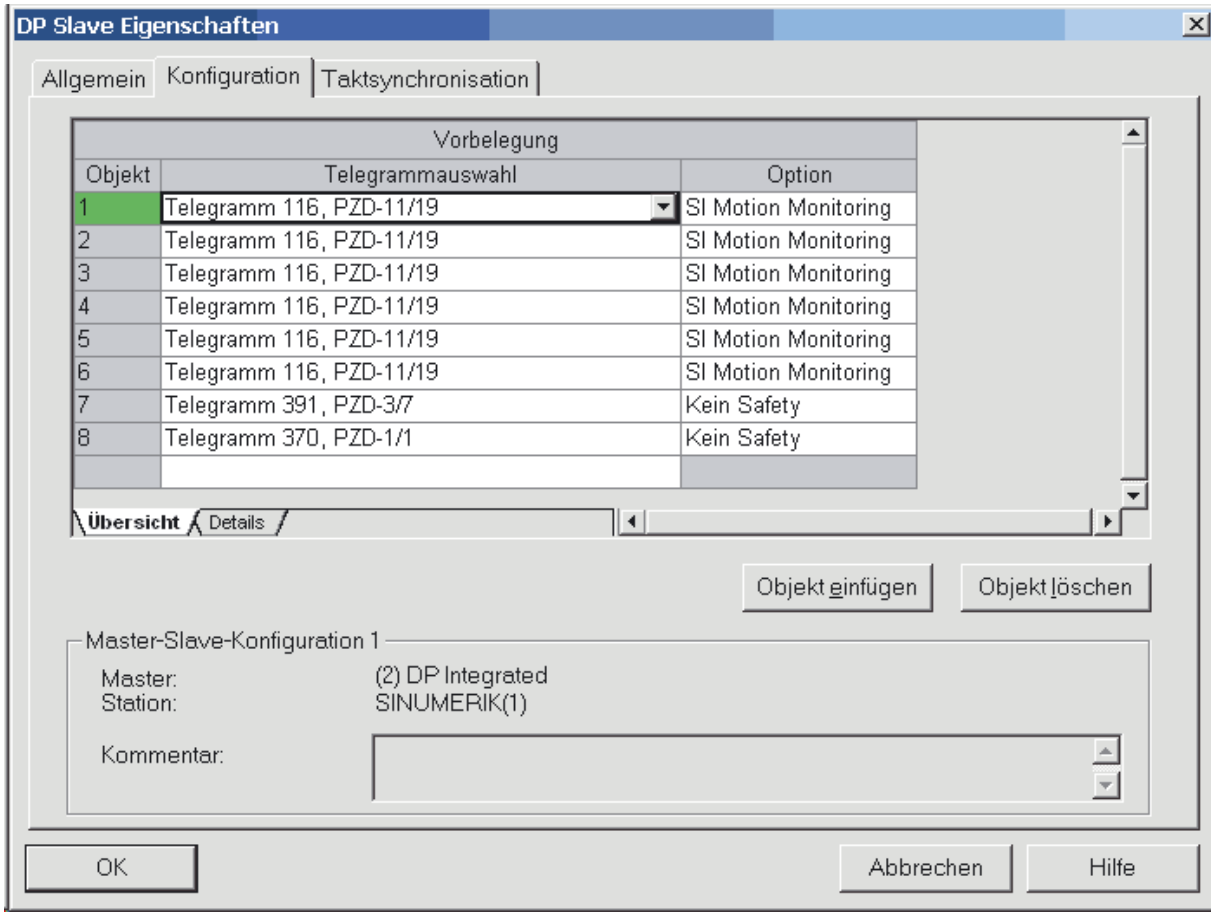


Fig. 2-2 DP slave properties

In the Properties dialog box for the integrated SINAMICS drive, object codes 1 to 6 are used to identify axes 1 to 6. The axes are set for message frame type 106 and Safety Motion Monitoring as standard. This is the default setting. Message frame type 106 features the longest possible user data length of an NC axis. The message frame type in the NC machine data may not tally at all with this setting. In the context of an NC axis, message frame type means: 2 encoders + DSC.

Object 7 contains the ALM via which, for example, enables have to be activated by the PLC user program. Object 8 contains the Dev0 (Device 0).

Axis expansions with the NX10 and NX15 modules are possible for NCU720 and NCU730. You will find NX10 and NX15 in the PROFIBUS DP\SINAMICS module catalog.

You will find the clock settings for the drives in the dialog box under the tab.

2.4 Starting up the PLC program

2.4.1 Installing the basic program for 840D

A complete general reset of the NCK and the PLC is necessary before starting up the NCU component for the first time. Turn the "SIM/NCK" switch on the left-hand side to position 1 and the "PLC" switch on the right-hand side to position 3. Then restart the controller. This action generates a "hard reset request" on the PLC. The memories of the PLC and NC are then initialized.

Installation

The basic program is installed by the Toolbox by means of the execution of a setup program for the basic program, hardware expansion for STEP7 (option package for SINUMERIK 840D sl) and NC-Var selector components, as well as for other tools. To start the installation, run setup.exe in the main CD directory. You can then choose which components to install. After the installation you can select the basic program library directly from STEP7 (bp7x0_12, 12 is the main basic program version 1.2). The actual version of the basic program can be scanned for the object properties of the library or the program folder in the comment field.

General

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 SIMATIC project or SIMATIC library of the basic program.

STEP7 must be installed before installing the basic program Toolbox. It is generally advisable to reinstall the hardware expansions for STEP7 from the Toolbox following an update of STEP7.

2.4.2 Application of basic program

A new CPU program (e.g., "Turnma1") must be set up in a project by means of the STEP7 software for each installation (machine).

Comment

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
- 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 STEP7 versions.

2.4.3 Version codes

Basic program

The version of the basic program is displayed on the HMI version screen along with the controller type.

The controller type is encoded as follows:

Left-aligned decade of DB17.DBD0 (byte 0)	Controller type
02	SINUMERIK 810D
03	SINUMERIK 840D (561,571, 572, 573)
04	SINUMERIK 840Di
03	SINUMERIK 840D sl (710, 720, 730)

User program

Users can also display their own PLC version codes on the HMI 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. Parameter assignments for this string are made via a pointer in FB1. Parameterization requires symbolic definition of the data block. See the FB1 block description for more information.

2.4.4 Machine program

The machine manufacturer creates the machine program using the library routines supplied with the basic program. The machine program contains the machine's logic links and machine sequences. The program also handles the interface signals to the NCK. More complex communication functions with the NCK (e.g., read/write NC data, tool-management acknowledgments, etc., are activated and executed via basic-program FCs and FBs).

The machine program can be created in various STEP7 creation languages (e.g. STL, LAD, CSF, FBD, 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 the blocks which call them. 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 procedure is valid in the same way for instance data blocks for FBs. If you do not follow this sequence, time-stamp conflicts will occur during recompilation in STEP7. In some cases, therefore, it may not be possible to recompile blocks, creating problems, for example, with the "Block status" function. It is, moreover, advisable to generate blocks in ASCIISTL by means of the STEP7 editor when they have been created in Ladder Diagram or in single statements (incremental mode).

2.4.5 Data backup

The PLC CPU does not store symbolic names, only the data-type descriptions of the block parameters (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) and the data types of the global data blocks. Without the associated project for this machine, therefore, blocks cannot be recompiled meaningfully (e.g., for the Block status function or in the event of modifications subsequently required to PLC CPU programs). It is, therefore, necessary to keep a backup copy of the STEP7 project located in the PLC CPU on the machine. This is extremely useful for servicing purposes and saves time and problems. If the STEP7 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 STL 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.

2.4.6 PLC series start-up, PLC archives

Once the blocks have been loaded to the PLC CPU, a series archive can be generated via the HMI operator 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, and does not back up, e.g., symbolic information. In addition, no CPU DBs (SFC 22 DBs) or SDBs generated in the CPU are saved.

Alternatively, the PLC series archive can be generated directly from the associated SIMATIC project.
To do this, select the "Options" → "Settings" menu item and the "Archive" tab in STEP7. This contains an entry "SINUMERIK (*.arc)", which must be selected to create a series start-up file. After selection of the archive, select the "File" → "Archive" menu item. The relevant series archive will then be generated. If the project contains several programs, the program path can be selected. A series archive is set up for the selected program path. All blocks contained in the program path are incorporated into the archive, except for CPU-DBs (SFC 22 DBs).

Automation:

The process of generating a series archive can be automated (comparable to the command interface in STEP7). 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 STEP7 command interface
Set S7Ext = CreateObject("SimaticExt.S7ContainerExt")
Call S7Ext.Magic("")
```

Functions:

Function **Magic**(bstrVal As String) As Long

Function **MakeSeriesstart-up**(FileName As String, Option As Long, Container As S7Container) As Long

Description:

Function **Magic**(bstrVal As String) As Long

Call gives 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 Step7 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 start-up file with general reset
Bit 0 = 1:	Series start-up file without general reset. When project contains SDBs, this option is inoperative. A general reset is then always executed.
Bit 1 = 1:	Series start-up 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 start-up files (e.g. disk full)

Use in script:

```
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).

```
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;
```

2.4.7 Software upgrades

Software upgrade

Whenever you update the PLC or NCK software, always reset the PLC to its initial state first. This initial clear state can be achieved by means of a general PLC reset. All existing blocks are cleared when the PLC is reset.
It is usually necessary to include the new basic program when a new NCU software version is installed. The basic programs blocks must be loaded into the user project for this purpose. OB 1, OB 40, OB 100, FC 12 and DB 4 should not be loaded if these blocks are already included in the user project. These blocks may have been modified by the user. The new basic program must be linked with the user program.

To do this, proceed as follows:

1. Generate the text or source file of all user blocks before copying the basic program.
2. Then copy the new basic program blocks to this machine project (for a description, see Subsection "Application of basic program")
3. All user programs *.awl must then be recompiled in the correct order! (See also the "Machine program" Section.)
   This newly compiled machine program must then be loaded to the PLC CPU using STEP 7.

However, it is normally sufficient to recompile the organization blocks (OB) and the instance data blocks of the machine program. This means you only need to generate sources for the organization blocks and the instance data blocks (before upgrading).

General reset

A description of how to perform a general PLC reset appears in the Installation and Start-Up Guide. However, a general reset does not delete the contents of the diagnostic buffer nor the node address on the MPI bus. Another possible general reset method is described below. This method must be used when the normal general reset process does not work.

Proceed as follows:

No.	Action	Activation
1	Control system is switched off	
2	PLC switch setting 3 (MRES) and switch control on again or perform hardware reset.	LED labeled PS flashes slowly.
3	Set PLC start-up switch to position 2 (STOP) and back to position 3 (MRES).	The LED labeled PS starts to flash faster.
4	Set PLC start-up switch to setting 2 or 0.	

NC tags

The latest NC VAR selector can be used for each NCU 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 version, i.e., selected variables in an older software version must not be reselected when the software is upgraded.

2.4.8 I/O modules (FM, CP modules)

Special packages for STEP7 are generally required for more complex I/O modules. Some of these special packages include support blocks (FC, FB) stored in a STEP7 library. The blocks contain functions for operating the relevant module which are parameterized and called by the user program. In many cases, the FC numbers for the CP and FM module handling blocks are also included in the number range of the basic program for the 840D. What action can be taken if such a conflict occurs?
The block numbers of the basic program must remain unchanged. The block numbers of handling blocks can be assigned new, free numbers using STEP7. These new blocks (with new FC numbers) are then called in the user program with the parameter assignments required by the function.

2.4.9 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 infor-mation	**Error**	**Cause/description**	**Remedy**
1	No connection via MPI to PLC.	The MPI cable is not connected or is defective. The STEP 7 software for the MPI card may not be configured correctly.	Test: Create a link with the programmer in the STEP7 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 faulty/has not been connected.
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. **Actions:** 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 STEP7. 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.

2.5 Linking PLC CPUs to 840D

2.5.1 General

General

The AS 300 family is used as the PLC for all systems. On the 840 D, the PLC CPU is integrated into the NCU component as a sub-module. The relevant performance data for PLC CPUs can be found in the above table or in the SIMATIC catalog.

2.5.2 Properties of PLC CPUs

SINUMERIK 810D/840Di PLC CPUs are based on standard SIMATIC CPUs in the S7300 family. As a result, they generally possess the same functions. Functional deviations are shown in the table above. Owing to differences in their memory system as compared to the SIMATIC CPU, certain functions are not available (e.g. save blocks on memory card, save project on memory card).

Note

With the current SIMATIC CPUs, the PLC is not automatically started after voltage failure and recovery when a PLC Stop is initiated via software operation. In this instance, the PLC remains in the Stop state with an appropriate diagnostic entry for safety reasons. You can start the PLC only via software operation "Execute a restart" or by setting the switch to "Stop" and then "RUN". This behavior is also integrated in the current versions of the SINUMERIK PLC.

2.5.3 Interface on 840D with integrated PLC

Physical interfaces

As the 840D system has an integrated PLC, signals can be exchanged between the NCK and PLC directly via a dual-port RAM.

Exchange with the operator panel front

Data exchange with the operator panel (HMI, OP) usually takes place via the internal software-based C bus. Alternatively, external HMI or OP systems can also be connected on Ethernet or the MPI bus. In the event of connection to the Ethernet bus, the integrated CP provides the link to the PLC. On the 840D, data exchange with the machine control panel (MCP) and handheld unit (HHU) is determined by the interface setup of the MCP/HHU. Possible connection paths include: MPI, PROFIBUS, Ethernet.
Programming devices should be connected to the Ethernet bus via the internal CP or via the MPI (Multi-Point-Interface) directly to the PLC.

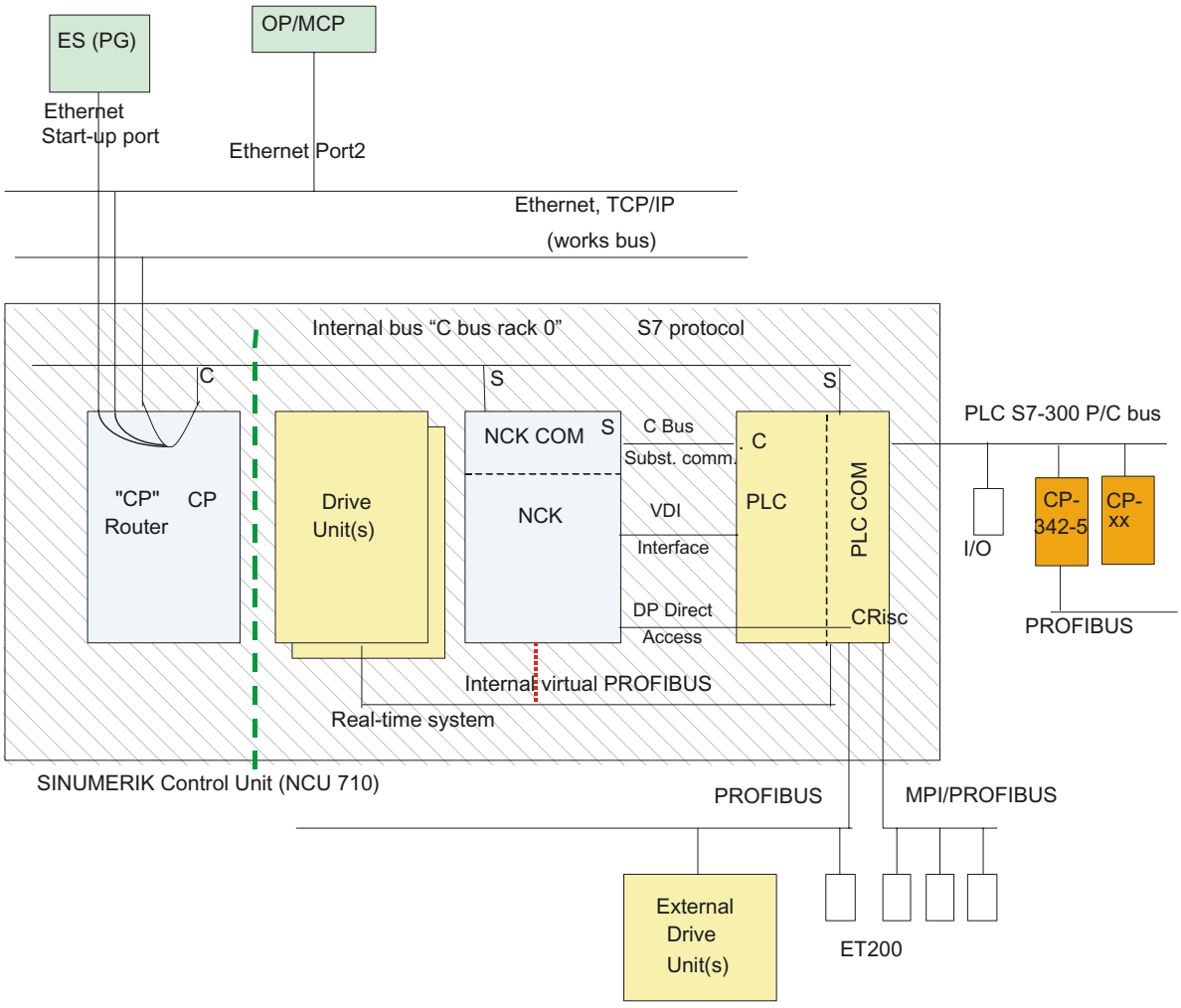


Fig. 2-3 NCK/PLC coupling on 840D (integrated PLC)

NCK/PLC interface

As illustrated in the figure, NCK/PLC data exchange is organized by the basic program in the PLC. At the beginning of the cycle (OB1), the **status information** (e.g. "Program running") written to the internal DPR by the NCK is copied to data blocks by the basic program, which the user can then access (user interface). The **control signals** for the NCK (e.g. NC start) entered in the interface data blocks by the user are also written to the internal DPR and transmitted 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 the interruption of the NCK machining process (e.g. M 06 for tool change), the basic program halts 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 does not contain auxiliary functions requiring the interruption of the decoding process (e.g. M 08 for cooling on), the transfer of these "rapid" auxiliary functions is enabled directly in OB 40, so that decoding is only marginally affected 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:
Chapter: Starting Up the PLC Program

Interface HMI/PLC

HMI/PLC data exchange takes place via the software C bus or via the Ethernet, internal CP, control-internal C bus link. The CP transfers the data intact from one bus segment to another. The HMI is always the active partner (client) and the PLC is always the passive partner (server). Data transmitted or requested by the HMI are 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 are identical to I/O signals.

Interface MCP/PLC, interface HHU/PLC (Ethernet link)

MCP/PLC and HHU/PLC data exchange takes place via the integrated CP's Ethernet bus. 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 (FB1, DB7) define the operand areas (e.g. I/O) and the start addresses.

Interface MCP/PLC (PROFIBUS link)

MCP/PLC data exchange takes place via the PLC's PROFIBUS. The MCP's I/O addresses must be set in the PLC's process image area and via HW configuration in STEP7. 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 MCP/PLC, interface HHU/PLC (MPI link)

MCP/PLC and HHU/PLC data exchange takes place via the PLC's MPI interface. The Communication with global data (GD) service (= implicit global data communication) is used for this purpose (see also STEP7 User Manual). The PLC operating system handles the transfer of signals from and to the configured operand areas. The STEP7 **NetPro** configuring tool is used to define both GD parameters and operand areas (e.g., I/O) and their start addresses. The addresses assigned in NetPro must also be set in the MCP*In, MCP*Out pointer variables on FB1.

Other parameters for MCP configuration must be set on FB1 in accordance with the FB1 description.

Please refer to chapter "Configuring the machine control panel, handheld unit" for more detailed information about configuration.

2.5.4 Diagnostic buffer on PLC

General

The diagnostic buffer on the PLC, which can be read out using STEP7, contains diagnostic information about the PLC operating system.

2.6 Interface structure

Interface data blocks

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.

2.6.1 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.

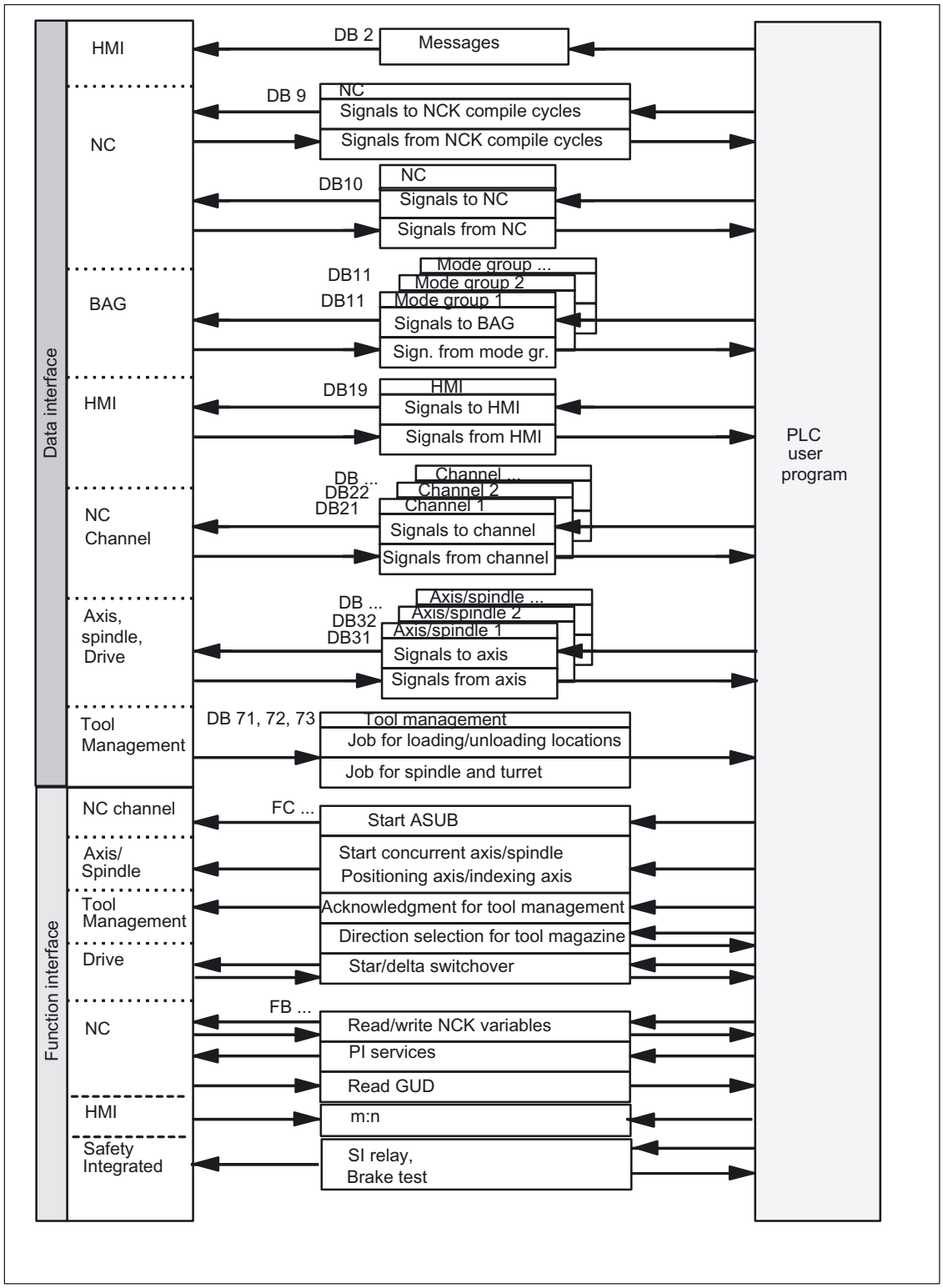


Fig. 2-4 PLC/NCK user interface

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

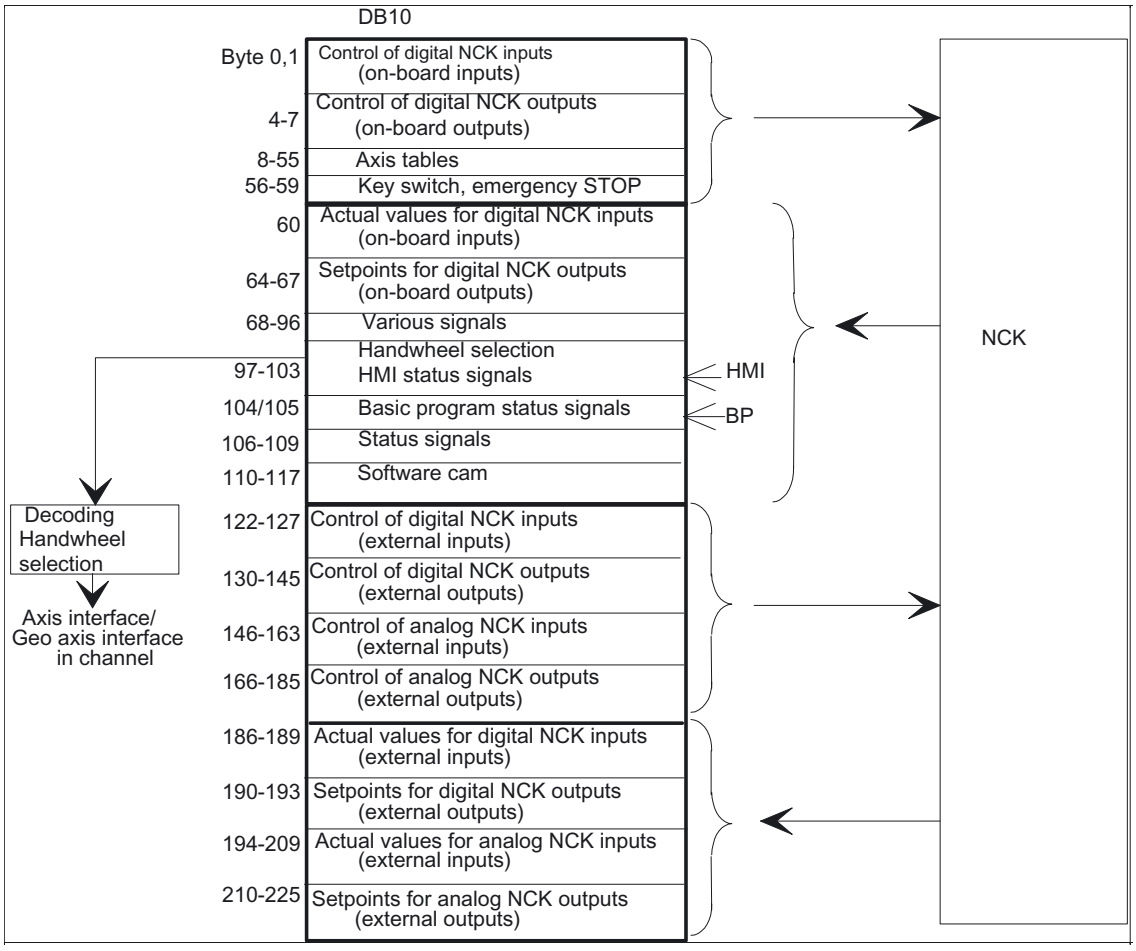


Fig. 2-5 PLC/NCK interface

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 inputs/outputs of the NCK

The following must be noted with respect to the digital and analog inputs and outputs 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 parts 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

When implementing digital and analog NCK I/Os, you must observe the information in the following references:

References:
/FB2/ Description of Functions, 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. On the 840D, these are valid for all NCK channels. On 840D systems, several mode groups can optionally be defined in the NCK.

The mode group reports its current status to the PLC.

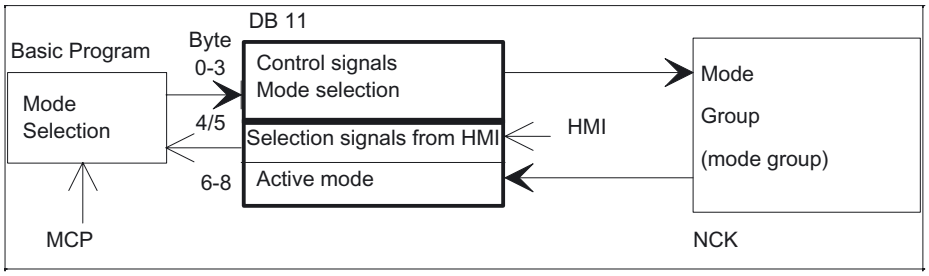


Fig. 2-6 PLC/Mode group interface

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 front, 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.

- **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.
- **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 axis-specific 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, for example:

- Position a linear axis or rotary axis
- Position an indexing axis
- Start a prepared asynchronous subprogram (ASUB)
- Read/write NC variables
- Update magazine and tool movement

Some of the above functions are described in their own function documentation.

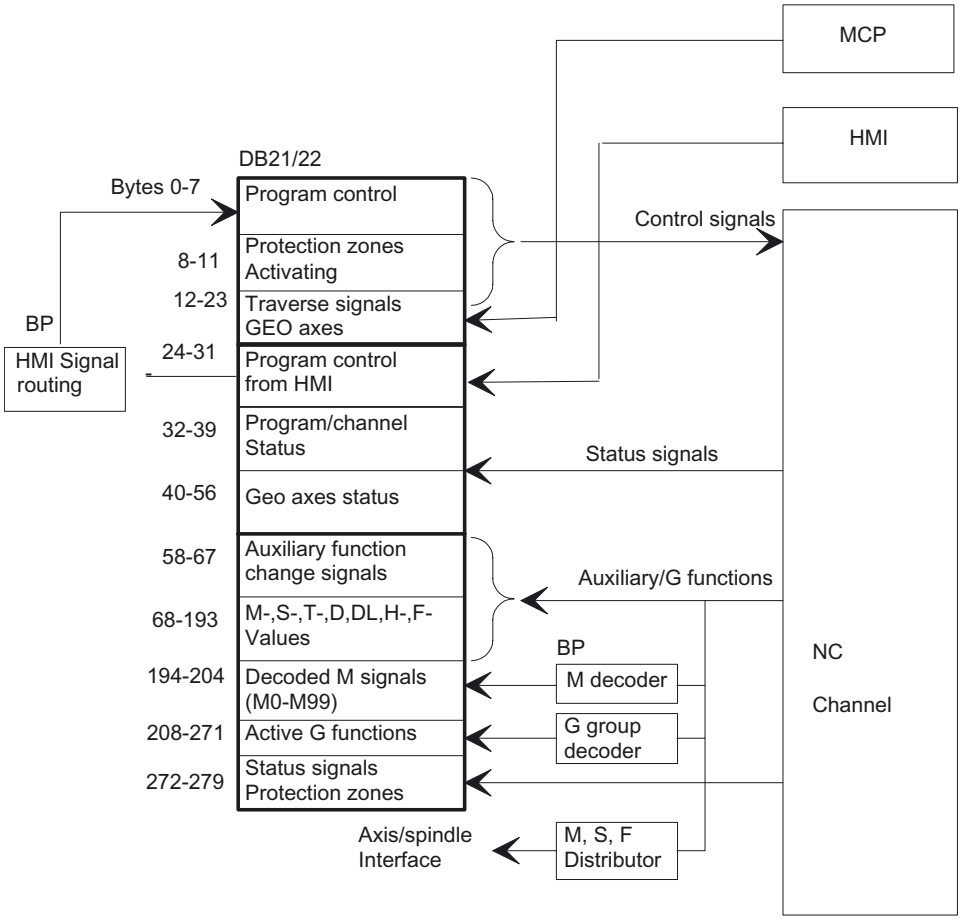


Fig. 2-7 PLC/NCK channel interface

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 and
- 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 requires processing.

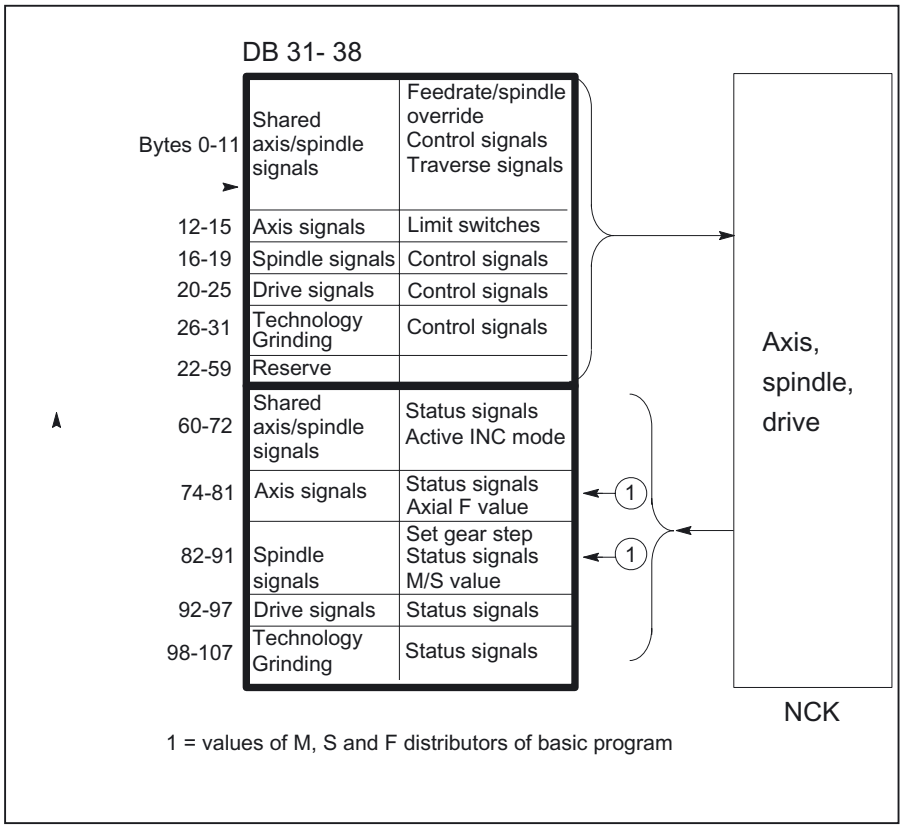


Fig. 2-8 Interface between PLC and axes/spindles/drives

2.6.2 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 operation via HMI via an FB1 parameter "MMCToIF".

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 MMC) 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 is 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 VDI 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 VDI 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.

Acknowledgment concept

The following acknowledgment procedures are implemented for error and operational messages:

Operating messages are intended for the display of normal operating states as information for the user. Acknowledgment signals are, therefore, not required for this type of message. An entry is made in the diagnostic buffer 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).

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.

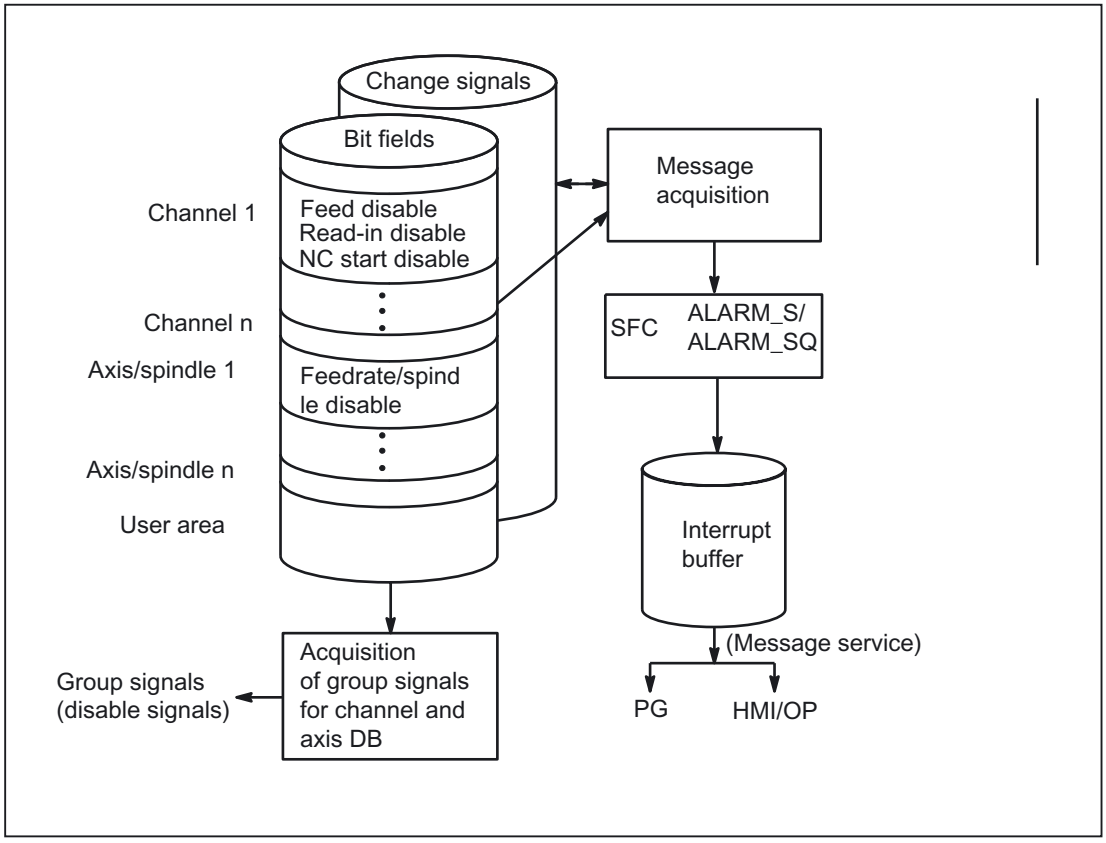


Fig. 2-9 Acquisition and signaling of PLC events

2.6.3 PLC/MCP/HHU interface

General

There are three 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 SINUMERIK 840 D, 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 operator unit.

840D topology

On the 840 D, the machine control panel and the handheld unit are connected to the CP 840D sl Ethernet bus (Fig. 9). 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. NCK-relevant signals are usually distributed by the basic program to the VDI 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.

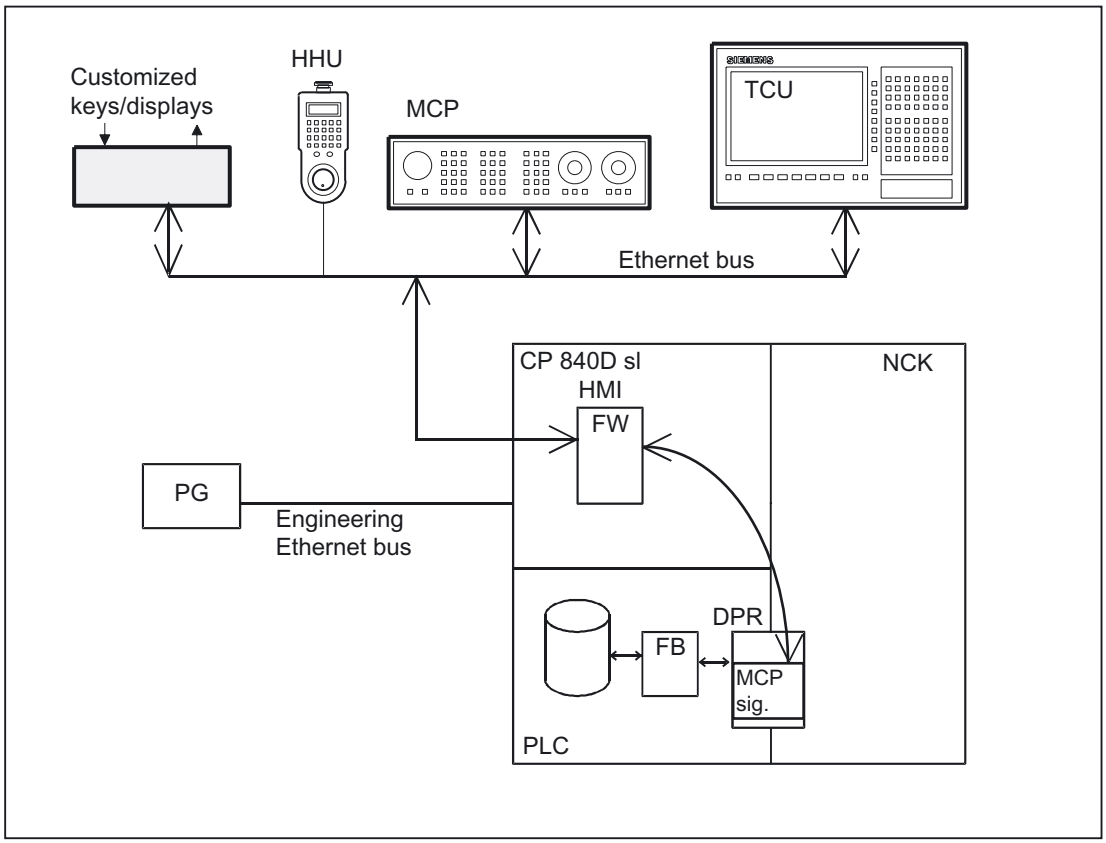


Fig. 2-10 Connection of the machine control panel on 840D

840Di topology

On the 840 Di, the machine control panel and handheld unit are connected to the PLC's MPI (MultiPoint Interface) or Profibus. The PLC operating system copies the incoming signals straight to the user interface (e.g., input image) at the cycle control point. As on the 840 D, transfer to the VDI interface is performed by the user program or by standard blocks of the basic program (e.g. FC19).

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 assigned to FB1 parameter MCPxBusAdr by the user.

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.

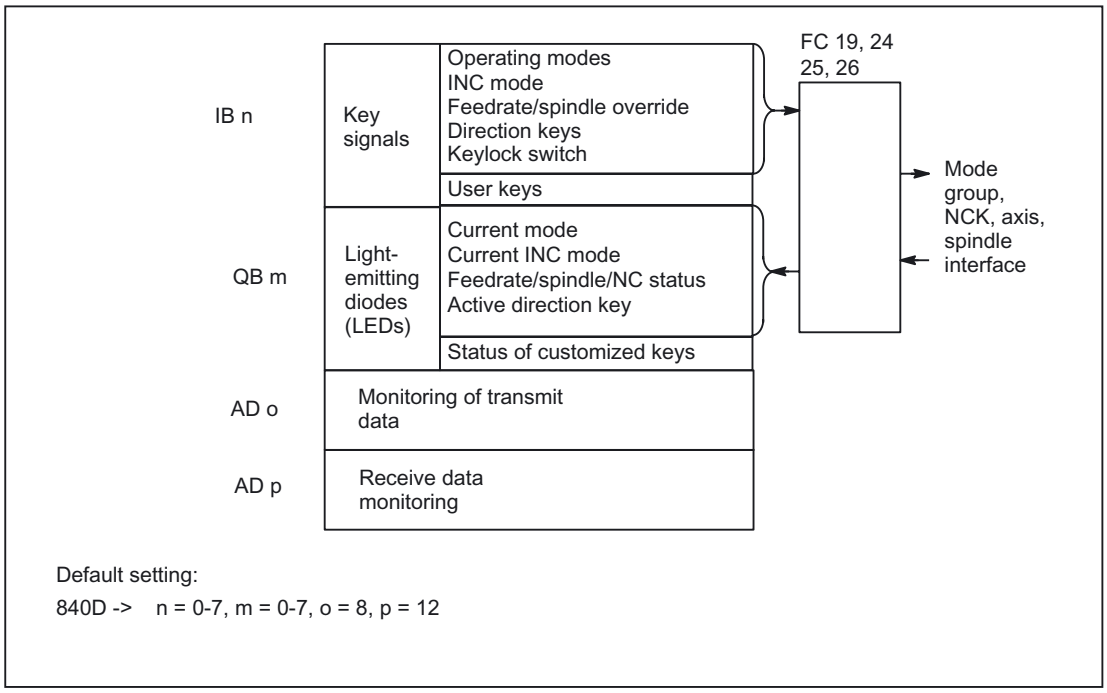


Fig. 2-11 Interface to and from machine control panel

2.7 Structure and functions of the basic program

General

The program is modular in design, i.e., it is structured according to NCK functions.

In the operating system, a distinction is made between the following levels of execution:

- Start-up and synchronization (OB 100)
- Cyclic mode (OB 1)

- Process interrupt handling (OB 40)
- Diagnostic interrupt, module failure (OB82, OB86)

Each section of the basic program - as illustrated in the figure below - must be called by the user in OBs 1, 40 and 100.

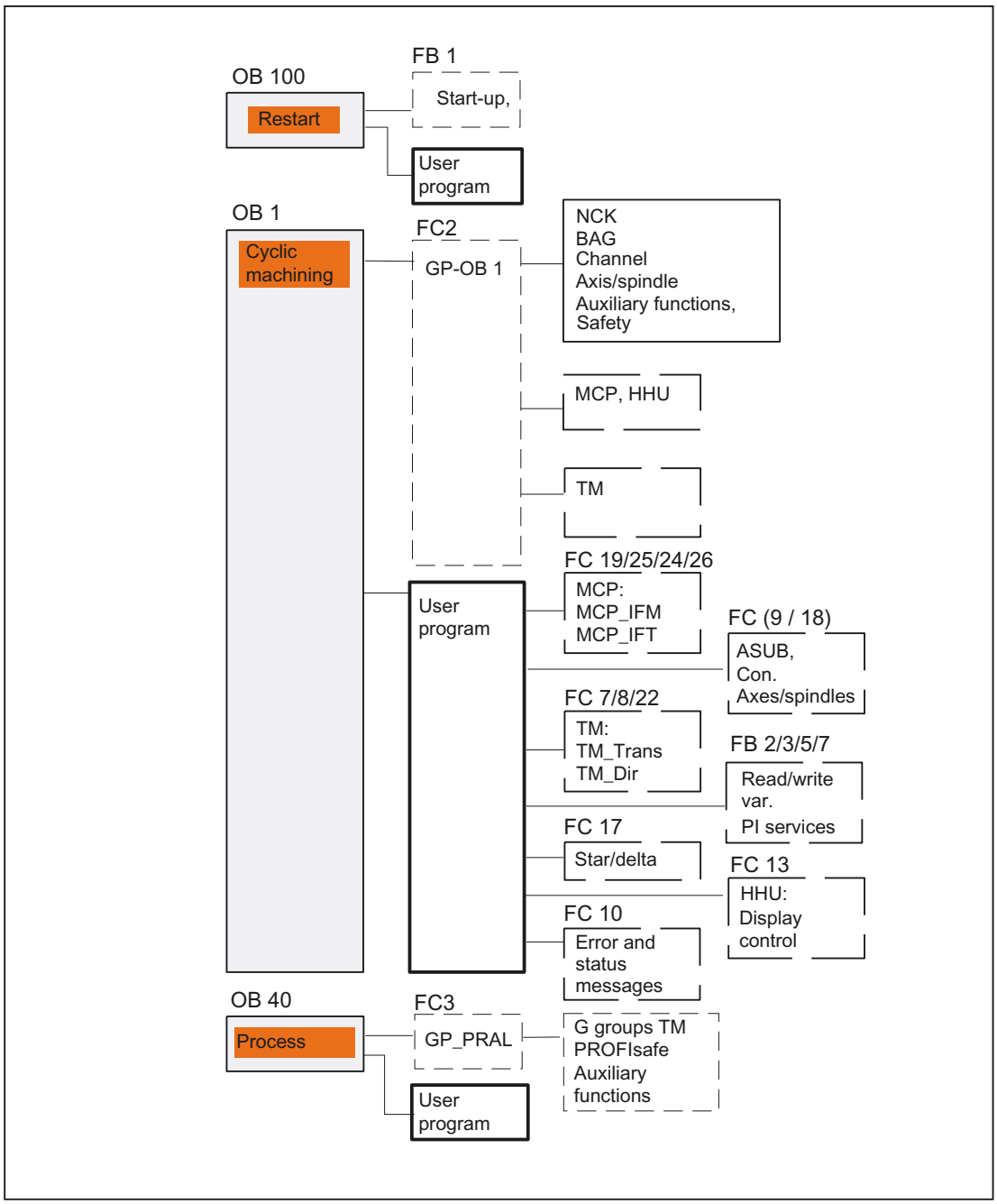


Fig. 2-12 Structure of the basic program

2.7.1 Start-up 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. An undefined state may otherwise develop on the PLC (e.g., all PLC LEDs flashing).

Start-up

The synchronization of NCK and PLC is performed during start-up. The system and user data blocks are checked for integrity and the most important basic program parameters are verified for plausibility. In the event of an error, the basic program outputs an error identifier to the diagnostic buffer and switches the PLC to STOP.

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 power-up.

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 signal from the NCK fails to arrive, the PLC/NCK interface is neutralized and the signal "NCK CPU ready" in DB 10 is set to zero.

2.7.2 Cyclic operation (OB1)

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
- Transmission of the MCP signals via the NCK (on the 840D only)
- 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 operating modes
- 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 parts program block)
- Transfer is acknowledge-driven.
- The acknowledgment times have an immediate effect on the execution time of NC blocks containing auxiliary functions requiring acknowledgment.

The value range is presented in the table below:

Function	Structure		Value range		Data type	
	1. Value	2. Value	1. Value	2. Value	1. Value	2. 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 STEP7 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 documentation "Lists of SINUMERIK 840D sl"). 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 acknowledgment is passed to the NCK. The acknowledgment 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 both 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 start-up, 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

On the 840 D, depending on the bus connection, MCP signals are transmitted to the parameterized I/O areas either directly to the PLC or indirectly 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.

2.7.3 Time-interrupt processing (OB 35)

General

The user must program OB 35 for time-interrupt 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 power-up 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.

2.7.4 Process-interrupt processing (OB 40)

General

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.

(For more information please also refer to the SIMATIC STEP7 description or the STEP7 online help).

2.7.5 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. If the cause of the error is removed, a PLC stop will not be triggered. A PLC stop does not occur on the PROFIBUS MCPs indicated in the FB1 parameters. The response can be changed by modifying the FC5 parameter setting.

2.7.6 Response to NCK failure

General

During cyclic operation, the PLC basic program continuously monitors NCK availability by polling the sign-of-life character. If the NCK is no longer reacting, then the NCK PLC interface is neutralized and IS **NCK CPU ready** in **signals from NC group (DB 10.DBX 104.7)** is reset. The signals sent by the NCK to the PLC are initialized with default settings.

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).

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 power-up 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.

2.7.7 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 spindle (FC 18),
- Read/write variables (FB 2, FB 3).

Note

The following note will later help you to check and diagnose a function call (FCs, FBs of basic program). These are FCs and FBs, which are controlled by a trigger signal (e.g., via parameter Req, Start, etc.), and which supply an execution acknowledgment as an output parameter (e.g., via parameter Done, NDR, Error, etc.). 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.
This control mechanism may be positioned 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, flags, data variables and time-related advantages over data variables).
The trigger parameter must be a global variable (e.g., flag, data variable).
In OB 100, jobs still activated by the user program (Parameter Req,
Start, etc.:= TRUE) must be set to zero at the named parameters. 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 can be traversed either from the PLC or from the NC.
- They can be started from a function call on the PLC in all operating modes.
- The start is independent of NC block boundaries.

Function calls are available for positioning, indexing axes and spindles (FC 18).

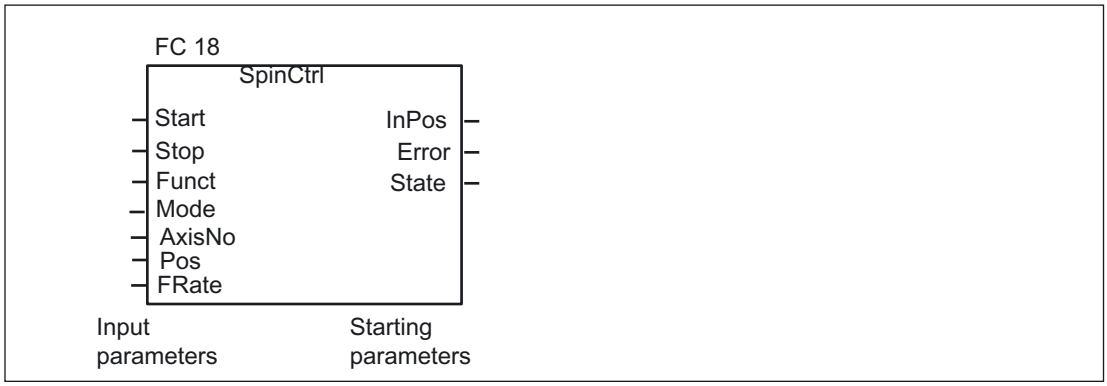
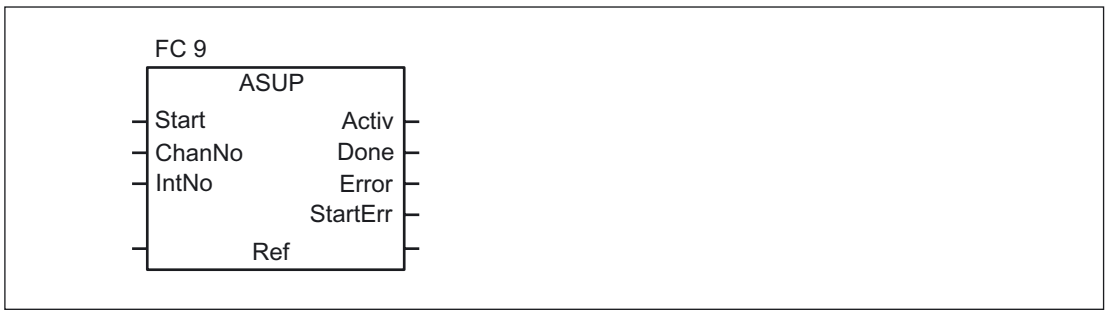


Fig. 2-13 FC 18 input/output parameters

ASUBs

Asynchronous subprograms (ASUBs) can be used to activate any function 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.



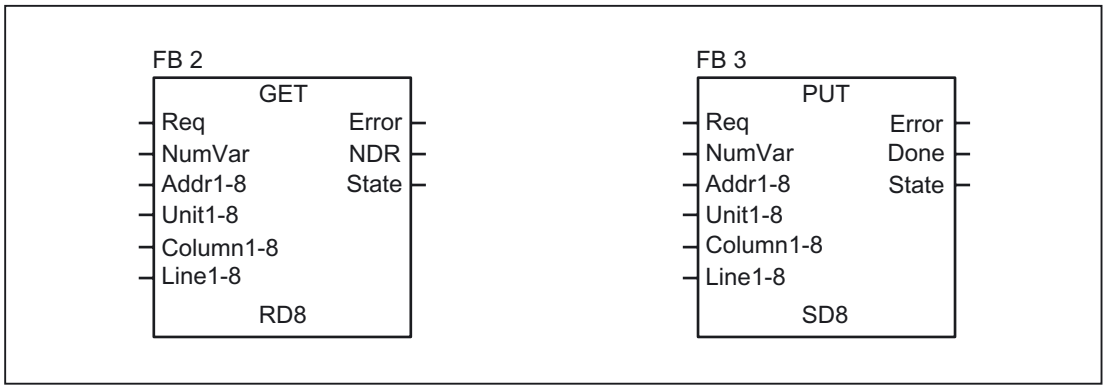
Note

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, 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 bits) are converted to PLC format (32 bits with 24-bit mantissa and 8bit exponent) and vice versa). A loss of accuracy results from the conversion from 64 bits to 32bit REAL. The maximum precision of 32bit REAL numbers is approximately 10 to the power of 7.



2.7.8 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**	**Meaning**
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
UDT77	DB 77	MCP and HHU signals with standard SDB 210

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 STEP7 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. To do so, however, a fully qualified instruction is needed for data access in the earlier program (e.g., "U DB31.DBX 60.0" - this command is converted to "AxisX.E_SpKA" when the symbols function is activated in the editor).

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 STEP7 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

2.7.9 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 between 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 read-in disable in the associated NCK channel is set simultaneously by the basic program.

The read-in 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 high-speed auxiliary function does not result in a read-in disable.

The figure below shows the structure of the **M decoding according to list**:

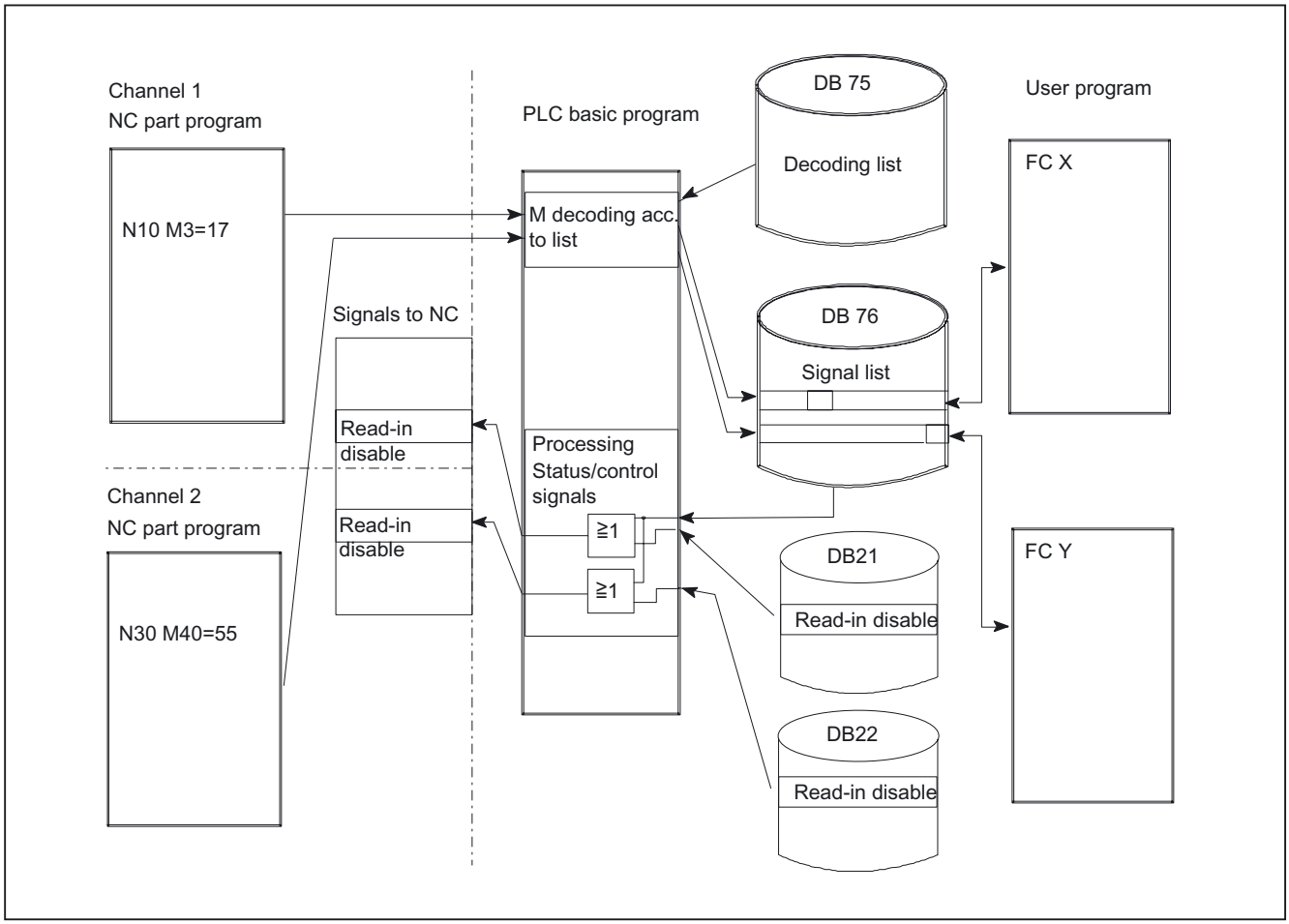


Fig. 2-14 M decoding acc. to list

Activation of the function

The number of evaluating/decoding groups is indicated in the basic program parameter "ListMDecGrp" when FB1 is called in OB 100 (see also FB1 description). 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**	**Remarks**
MExtAdr	Int	0 to 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 read-in 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	DBX0.0 to DBX0.4
2	3	12	23	DBX2.0 to DBX3.3
3	40	55	55	DBX4.0

```
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 FB1 in 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 DB75 (decoding list) to the PLC 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 DB76 (see decoding list DB75). At the same time, the basic program sets the read-in disable and the processing of the NC program is halted (and the "expanded address M function" and "M function no." are not entered in the associated NC channel DB).

The read-in 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 (DB76).

2.7.10 PLC machine data

General

The user has the option of storing PLC-specific machine data in the NCK. These machine data can then be processed during power-up of the PLC (OB 100). This enables, for example, user options, machine expansion levels, machine configurations, etc., to be implemented.

The interface to read these data is stored 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 BP parameters UDInt, UDHex and UDReal is greater than zero.

The size of the individual ranges and, therefore, the total length of DB 20,
is set using PLC machine data
MD14504 MAXNUM_USER_DATA_INT,
MD14506 MAXNUM_USER_DATA_HEX,
MD14508 MAXNUM_USER_DATA_FLOAT
and specified for the user in BP parameters
UDInt, UDHex and UDReal.
The basic program stores the data in DB 20 in the following order:
Integer MD, Hex field MD, Real MD.

The integer and real values are stored in DB 20 in S7 format.
Hexadecimal data are stored in DB20 in the order in which they are input (use as bit fields).

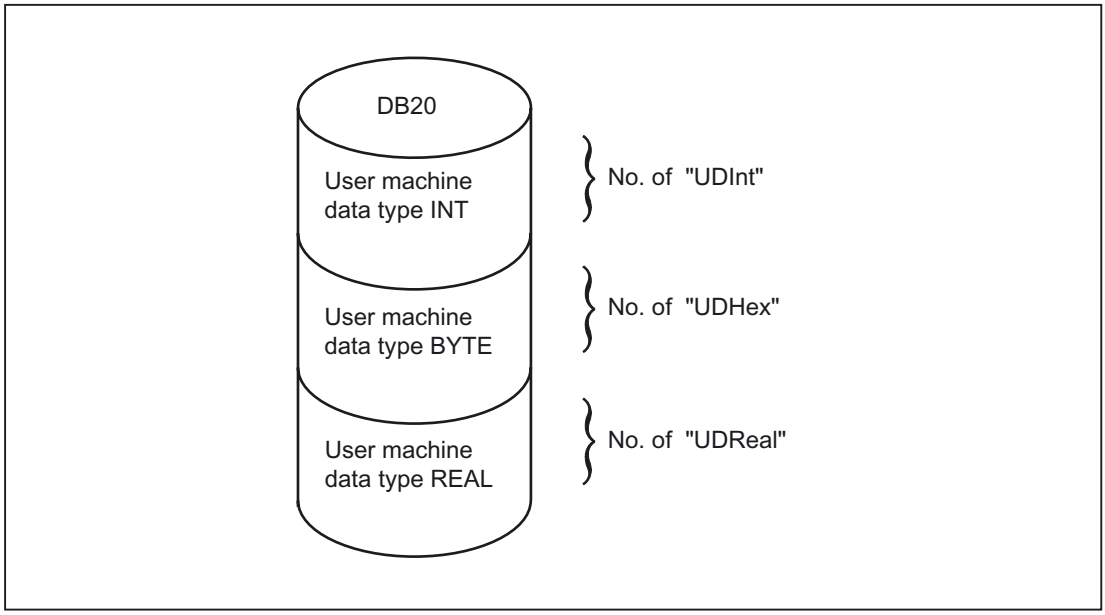


Fig. 2-15 DB 20

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.

Interrupts	
400120	**Delete DB 20 in PLC and restart**
Explanation	DB length is not the same as the required DB length
Reaction	Interrupt 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 USER_DATA_INT[0]	123
MD14510 USER_DATA_INT[1]	456
MD14510 USER_DATA_INT[2]	789
MD14510 USER_DATA_INT[3]	1011
...	
MD14512 USER_DATA_HEX[0]	12
MD14512 USER_DATA_HEX[1]	AC
...	
MD14514 USER_DATA_FLOAT[0]	123.456

BP parameters (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

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.

An entry is made in the symbol table to allow data access in symbolic form:

Interrupts

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];
...
```

2.7.11 Configuring machine control panel, handheld unit

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, MPI) 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 first MCP and the left-hand decade for the second 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 and output addresses are also used in FC 19, FC 24, FC 25, FC 26 and FC 13. Addresses for status information, MPI, PROFIBUS/Ethernet (a GD parameter set must be set for the MPI variant of the handheld unit rather than an MPI address) must also be defined. The default time settings for timeout and cyclic forced retriggering should not be changed.

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 (MPI)

The handheld unit addresses the MPI by means of a GD parameter set. These parameter values must be assigned according to the handheld unit settings. However, the parameter names on the handheld unit are the reverse of the parameter names in the basic program. In the basic program, all *Send type parameters must be defined by the handheld unit as *Rec type (and *Rec as *Send).

Configuring

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 CP840D. The mechanism is parameterized completely via the MCP/HHU parameters in FB1.

In the second case, data are transferred via the PLC operating system by means of the evaluation of SDB210 (global data) or via the PROFIBUS configuration.

The mechanism is parameterized via STEP7->Global Data or in HW Config. To allow the basic program to access these data and implement MCP/HHU failure monitoring, the addresses set via SDB210 (global data) must be declared in the FB1 parameters in the basic program.

An overview of the various coupling mechanisms appears below. Mixed operation can also be configured. If an error is detected due to a timeout monitor, an entry is made in the diagnostic buffer of the PLC CPU (interrupts 400260 to 400262). In this case, the input signals from the MCP or from the handheld unit (MCP1In/MCP2In or BHGIn) are initialized with 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

In the following tables, "(n.r.)" indicates "not relevant".

840D: Ethernet connection

Without further configuration settings being made, communication takes place directly from the PLC GP via the CP 840D sl. The FB1 parameters listed below are used for parameterization.

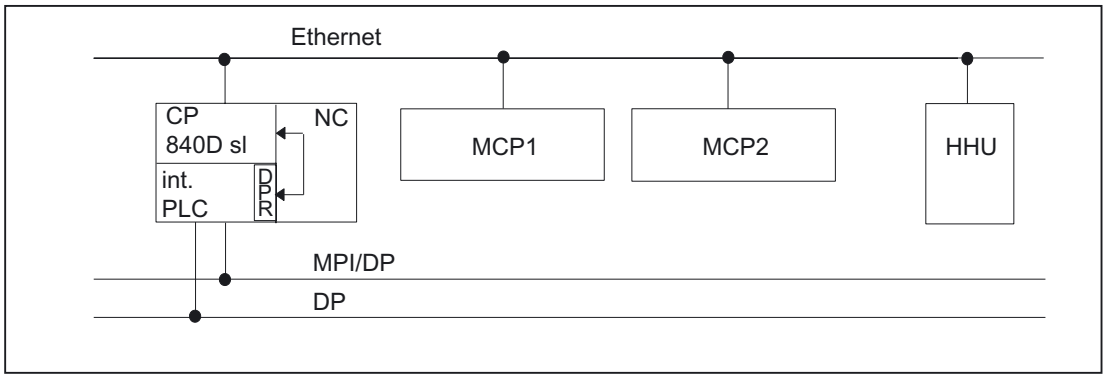


Fig. 2-16 840D: Ethernet connection

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	MCP2StatSend	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl	MCP2Cycl	BHGTimeout
MCPMPI = FALSE		BHGCycl
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType = b#16#55 (via CP 840D sl)		BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPSDB210= FALSE		BHGSendGDNo (n.r.)
MCPCopyDB77 = FALSE		BHGSendGBZNo (n.r.)
		BHGSendObjNo (n.r.)
		BHGMPI = FALSE
		BHGStop

Status information		
Available in	**Bit No.**	**Description**
MCP1StatSend MCP2StatSend BHGStatSend	4	Syntax error in data package: Error in parameter set (FB1)
MCP1StatSend MCP2StatSend BHGStatSend	27	Transmitter: Timeout
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Timeout

An error entry is also made in the PLC diagnostic buffer for timeouts (bits 10 and 27),

resulting in the following error messages on 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 have yet been exchanged between the MCP/HHU and PLC.
The monitoring function is activated as soon as all components have signaled "Ready" after power-up.

840D: PROFIBUS connection

With an MCP PROFIBUS connection, these components must be considered in the STEP7 hardware configuration. The MCP may only be interfaced with the standard DP bus on the PLC (not 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 MCP1BusAdr and MCP2BusAdr. Enter the pointer to the configured diagnostic address (e.g., P#A8190.0) in MCPxStatRec.

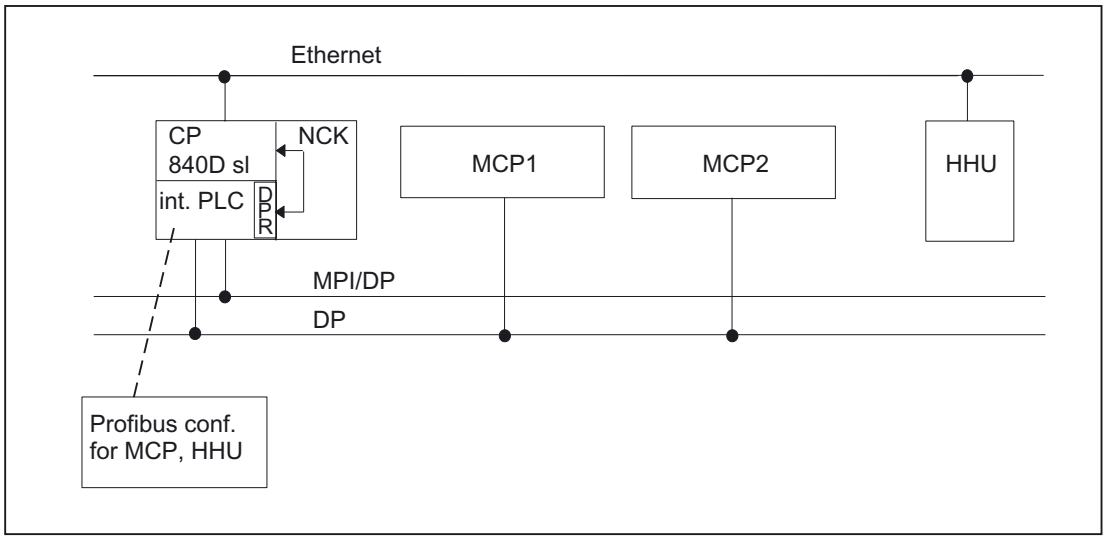


Fig. 2-17 PROFIBUS connection

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	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl (n.r.)	MCP2Cycl	BHGTimeout
MCPMPI = FALSE		BHGCycl
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 switches the PLC to the STOP state. If this is undesirable, OB 82, OB 86 can be used to avoid a stop. Setting MCPxStop := True deactivates the MCP as a slave via SFC 12. If the PLC does not switch to the stop state following the failure of or a fault on the MCP, an interrupt message will be generated via the basic program. The interrupt is deleted when the station recovers.

840D: MPI connection

If an MPI connection is used for the MCP and HHU, these components must be taken into account in the configuration for global data in STEP7 (NetPro). Although the addresses of the MCP and HHU can be stored anywhere, we recommend a free input and output image area. These start addresses must also be stored in the pointer parameters of the FB1. The FB1 parameters listed below are used for further parameterization.

Setting MCP1Stop, MCP2Stop, BHGStop = 1 will suppress time monitoring. MCP failed interrupts are suppressed.

Once the configuration has been compiled in NetPro, the SDB 210 in the SDB container will need to be loaded on the PLC.

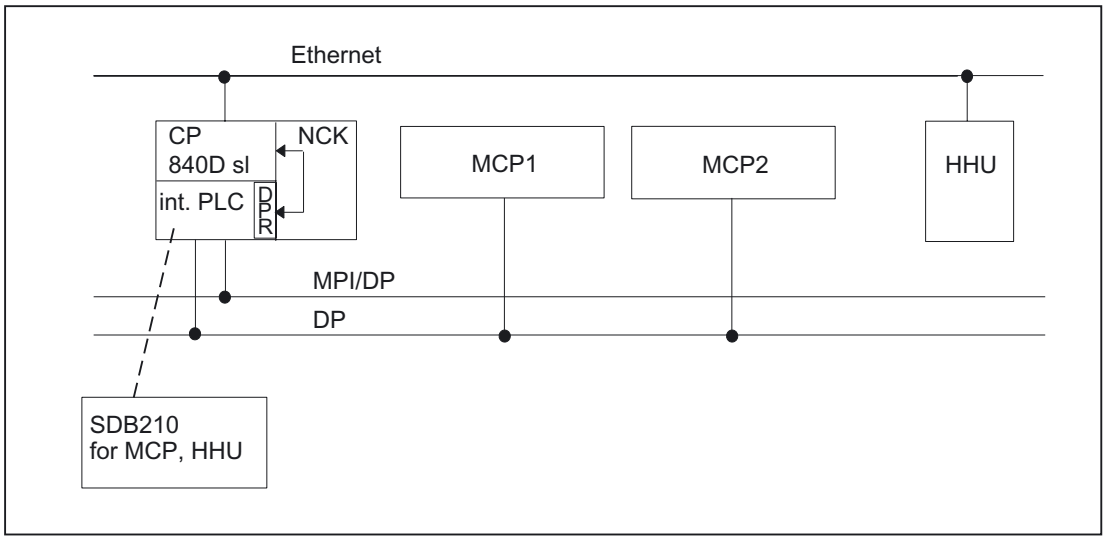


Fig. 2-18 MPI connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum=1 or 2 (number of MCPs)		HHU = 1 (MPI)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend	MCP2StatSend	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1Timeout	MCP2Timeout	BHGTimeout
MCPMPI = TRUE		BHGMPI = TRUE
MCP1Stop	MCP2Stop	BHGStop
MCPSDB210 = TRUE		
MCPCopyDB77 = TRUE or FALSE		
MCPBusType = B#16#00		

(all pointer entries as parameterized in SDB210 global data)

Status information (MCP1 and MCP2 see above)		
Available in	**Bit No.**	**Description**
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Timeout

An error entry is also made in the PLC diagnostic buffer for timeouts (bits 10 and 27), resulting in the following error messages on the HMI:

- 400260: MCP 1 failure

  or

- 400261: MCP 2 failure
- 400262: HHU failure

An MCP/HHU failure is detected only following a cold restart if the MCP/HHU has previously been involved in data exchange. The first exchange of data with the MCP/HHU activates the monitoring function.

Configuring global data

An additional SIMATIC 300 station for each component needs to be added to the machine project for the configuration of the machine control panel/handheld unit. Any type of CPU must be inserted in location 2 on row 0 in this station by means of the hardware configuration (HW Config.). The desired MPI address of the operator component must be set as the MPI address. MPI network(1) can then be marked in the SIMATIC Manager. The global data can then be activated via the Extras menu item. The rest of the procedure is described in detail in the Installation and Start-Up Guide.

A special variant of GD communication with a standard SDB210 is to be represented here. The standard SDB210 is designed for all MCPs and HHUs on DB77. Other pointers can be wired to the FB1. The basic program manages data transfer if the FB1 parameter MCPCopyDB77 is set to True (see figure below). If you are using the standard SDB210, the following settings must be made on the MCPs/HHUs:

1. MCP must be set to bus address 14. MCP2 must be set to bus address 10. In respect of the GD parameters, HHU must be set to the default settings of 2.1.1 and 2.2.1. All components must be set to a transfer rate of 187.5 kbaud.

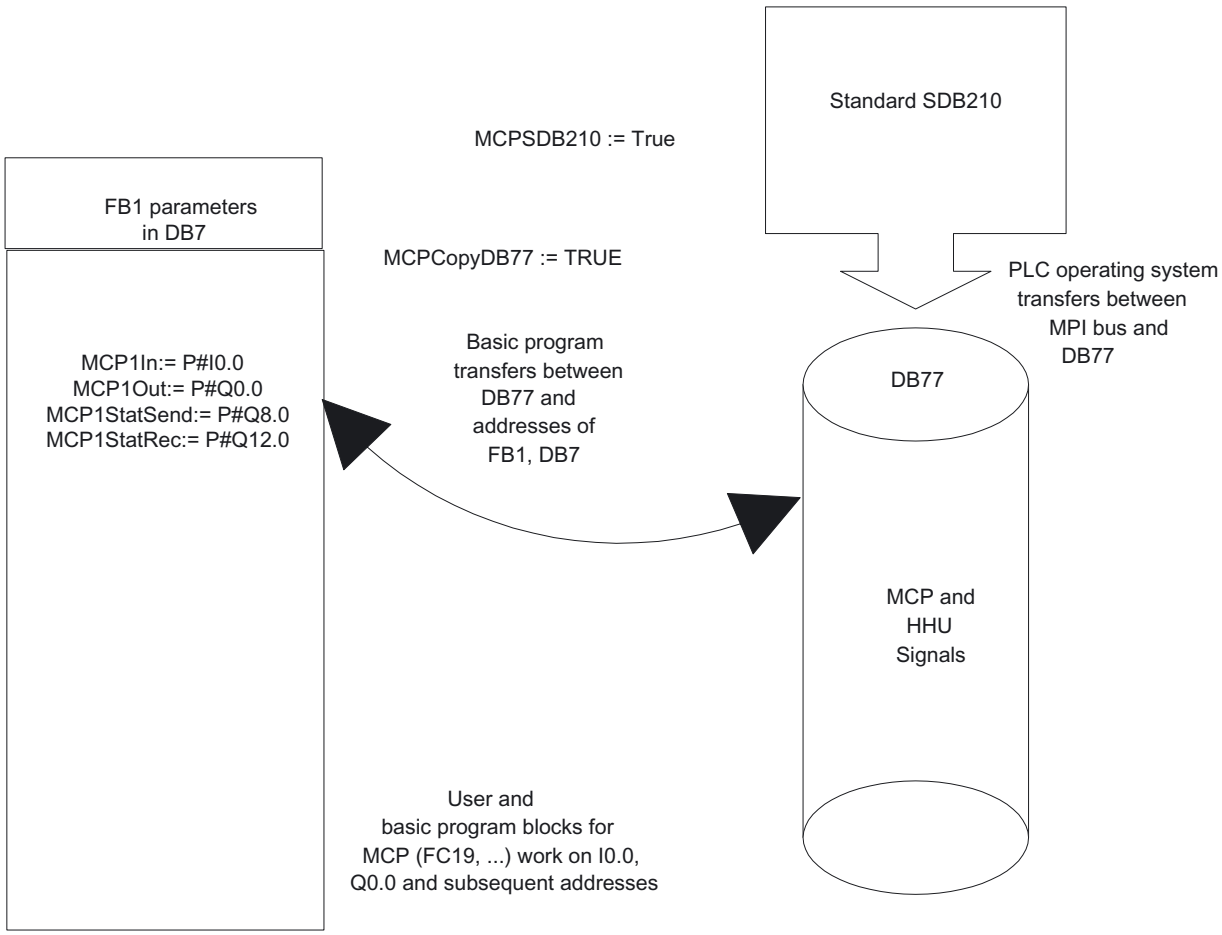


Fig. 2-19 Configuring global data

2.7.12 Switchover of machine control panel, handheld unit

Requirements

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 first 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.

Bus address switchover

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"; 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 DB10 byte 104 (relevant bits 0, 1, 2 are set to 1).

2.7.13 Special functions of the machine control panel

General

Special functions are available for some MCP variants:

Switch off MCP flashing

On the MCP MPI variant, flashing can be suppressed in offline mode. No communication takes place in offline mode (e.g., even if the MCP connection fails). During active communication, a value can be specified in the output data using status bits 24 or 25 in MCP1StatSend and MCP2StatSend. The checkback signal indicating that the value has been transferred is sent back in the same bits in MCP1StatRec, MCP2StatRec. Following a successful checkback signal, the status bits in send status must be reset.

There is an implementation example in the Toolbox.

Temperature monitoring of the MCP

With the MCP MPI variant, an increased temperature is signaled back via bit 28 = 1 in MCP1StatRec, MCP2StatRec.

2.8 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 (DB18) available for Safety SPL signals and runs a data comparison to ensure the consistency of SPL program data in the NCK.

See:
References:
/FBSI/ Description of Functions, Safety Integrated

2.9 Assignment overview

2.9.1 Assignment: NCK/PLC interface

The assignment of the NC/PLC interface is comprehensively described in:

References:
/LIS/ Lists

2.9.2 Assignment: FB/FC

Number	Meaning
FB 15	Basic program
FB 1, FC2, FC 3, FC5	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. For FC and FB assignment, see Section 6.4 (Memory requirements of PLC basic program for 810D, 840D).

2.9.3 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**	**Pa-ckage**
1		Reserved for Siemens	BP
2-4	PLC-MELD	PLC messages	BP
5-8		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 - upper 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.

2.9.4 Assignment: Timers

Timer No.	Meaning
T 0 – to 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.	

2.10 Memory requirements of basic PLC program for 810D, 840D

General

The basic program consists of basic and optional functions. The **basic functions** include cyclic signal exchange between the NC and PLC.
The **options** include, for example, the FCs, which can be used if required.

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 with SINUMERIK 840D			
			Block size (bytes)
Block type no.	Function	Remark	Working memory
Basic functions in basic program			
FB 1, (15)		Must be loaded	678
FC 2, 3, 12, 20		Must be loaded	2146
DB 4, 5, 7, 8, 17, 19		Must be loaded	1480
DB 2, 3, 6		Are generated by the BP	812
OB 1, 40, 100		Must be loaded	282
		Total value	5398

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 21, 30	PLC/channel signals	Are generated by BP as a function of NCMD	412 each
DB 31, ...61	PLC/axis or spindle signals	Are generated by BP as a function of NCMD	14 each

Basic program options			
	Machine control panel		
FC 19	Transfer of MCP signals, M variant	Must be loaded when M variant of MCP is installed	1538
FC 25	Transfer of MCP signals, T variant	Must be loaded when T variant of MCP is installed	1328
FC 24	Transfer of MCP signals, slim variant	Must be loaded when slim variant of MCP is installed	1522
FC 26	Transfer of MCP signals, HPU variant	Must be loaded for HPUs	1216
FC 14	MPI/OPI transfer	Must be loaded if MCPNum > 0	802
	Handheld unit		
FC 13	Display control HHU	Can be loaded for handheld units	1032
	Error/operating messages		
FC 10	Acquisition FM/BM	Load when FM/BM is used	1354
	ASUB		
FC9	ASUB start	Load when PLC ASUBs are used	518

Basic program options			
	Star/delta changeover		
FC 17	Star/delta switchover of MSD	Load for star/delta switchover	492
	Spindle control		
FC 18	Spindle control	Load for spindle control from PLC	882
	PLC/NC communication		
FB 2	Read NC variable	Load for Read NC variable	350
DB n1)	Read NC variable	One instance DB per FB 2 call	270 each
FB 3	Write NC variable	Load for Write NC variable	350
DB m1)	Write NC variable	One instance DB per FB3 call	270 each
FB 4	PI services	Load for PI services	296
DB o1)	PI services	One instance DB per FB4 call	130 each
DB 16	PI services description	Load for PI services	618
FB5	Read GUD variables	Load for PI services	472
DB p	Read GUD variables	One instance DB per FB5 call	166 each
FB 6	General communication	Load for Read/write NC variables and PI services	5512
DB 15	General communication	Instance DB for FB 6	280
FB7	PI services 2	Load for PI services	300
DB o	PI services 2	One instance DB per FB4 call: every	144
FC 21	Transfer	Load with dual-port RAM, ...	1074
	m:n		

Basic program options				
FB 9	Switchover M to N	Load with M to N		2512
	Safety Integrated			
FB10	Safety relay	Load with Safety option		354
FB 11	Brake test	Load with Safety option		780
FB 18	Safety data	Load with Safety option		226
	Tool management			
FC 7	Transfer function turret	Load for tool management option		420
FC 8	Transfer function	Load for tool management option		1062
FC 22	Direction selection	Load if direction selection is required		298
DB71	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	(	(B+ Sp+R)*20
	Compile cycles			
DB 9	Interface PLC compile cycles	Is generated by BP as a function of NC option		436
1): DB number must be specified by PLC user				

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		5398
	Interface DBs		1174
	MCP		1328
		Total value	7900

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		5398
See above	Interface DBs		2326
See above	MCP		1328
See above	Error/status messages		1354
See above	ASUBs	1 ASUB initiation	518
See above	Concurrent axis	For 2 turrets	882
See above	PLC/NC communication	1 x read variable and 1 x write variable	1240
See above	Tool management	2 turrets with one loading point each	1940
See above	Compile cycles		436
		Total value	15422

2.11 General conditions and NC-VAR_Selector

2.11.1 Supplementary conditions

2.11.1.1 Programming and parameterizing tools

Hardware

Programming devices or PCs with the following equipment are required for the PLCs installed on the 810D and 840D:

	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	

	Minimum	Recommendation
Graphic	SVGA (1024*768)	
Mouse	Yes	
Operating system	Windows 2000 /XP Professional, STEP7 version 5.3 SP2 or higher	

The required version of **STEP7** 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
- Parameterization
  - 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
  - PLC STOP/complete restart/overall reset triggering from the programming device
  - 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 statuses 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 configuring modules
    (e.g., CP3425 -> NCM package)

Note

More information about possible functions can be found in SIMATIC catalogs and STEP7 documentation.

2.11.1.2 SIMATIC documentation required

References:
SIMATIC S 7 System Overview
S7-300 Operation List
Programming with STEP7
STEP7 User Manual
STEP7 Programming Manual; Designing User Programs
STEP7 Reference Manual; STL Statement List
STEP7 Reference Manual; LAD Ladder Diagram
STEP7 Reference Manual; Standard and System Functions
STEP7 Manual: Conversion of STEP5 Programs
STEP7 General Index
CPU 317-2DP Manual

2.11.1.3 Relevant SINUMERIK documents

References:
/IAD/ Installation and Start-Up Guide 840D, PLC Interface
/BH/ Operator Components (HW) Manual 840D
/FB/ Descriptions of Functions 840D
/LIS/ Lists 840D / 810D

2.11.2 NC VAR selector

2.11.2.1 Overview

General

A catalog with a user-defined catalog name must be set up via the Windows Explorer. The selected data of the VAR selector (data.VAR and data.AWL (STL)) must be stored in this catalog. The "Daten.AWL" (STL data) file must then be inserted into the STEP7 machine project via "Insert", "External Source" in the STEP7 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 STL files must be compiled with STEP7.

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 STEP7 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".

Note

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 version, i.e., selected variables in an older software version must not be reselected when the software is upgraded.

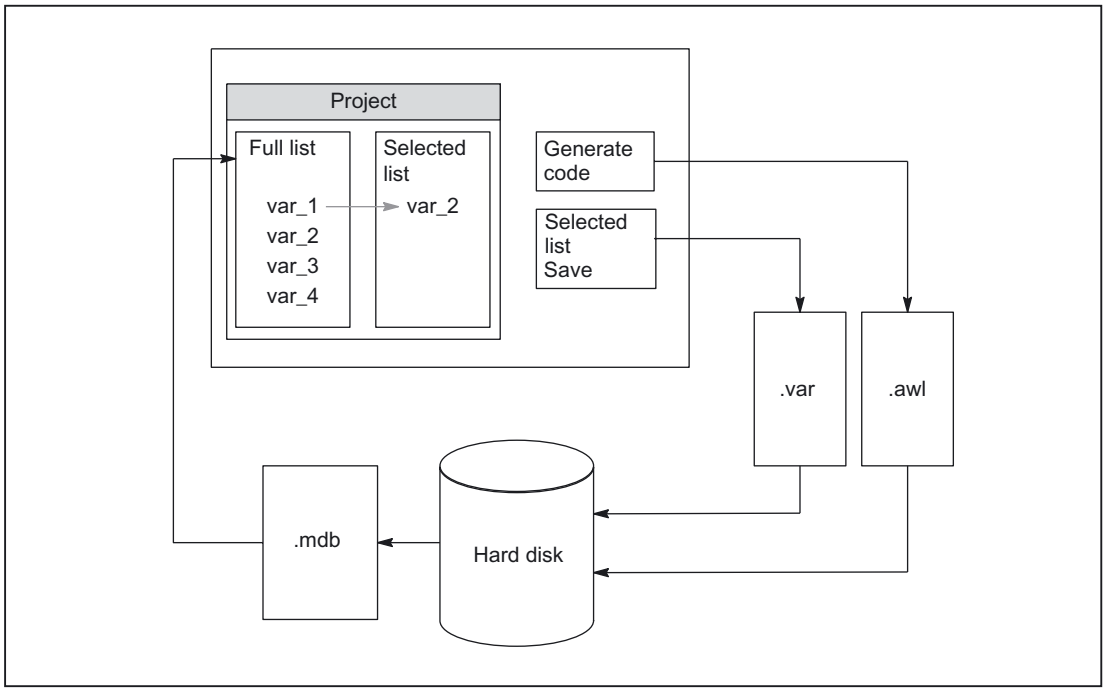


Fig. 2-20 NC VAR selector

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.

In SW V6.3 and higher, the variable lists ncv*.mdb are separated according to:	
NC variables including machine and setting data:	ncv_NcData.mdb
Machine data for 611D drive:	ncv_611d.mdb
Machine data for 611D linear drive:	ncv_611dLinear.mdb
Machine data of the 611D drive, Performance 2:	ncv_611d_P2.mdb
Machine data of the 611D linear drive, Performance 2:	ncv_611d_P2Linear.mdb
Machine data of the hydraulic drive:	ncv_Hydraulics.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 STEP7 source file (.awl) and stored.

Once he has generated a PLC data block by means of the STEP7 compiler, the programmer is able to read or write NCK variables via the basic program function blocks "PUT" and "GET" using the STEP7 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 95 (or later operating system).

The assignment of names to variables is described in:
References:
/LIS/ Lists; Chapter: Variables,
or in the Variables Help file (integrated in NC VAR selector).

2.11.2.2 Description of functions

Overview

The figure below illustrates how the NC VAR selector is used within the STEP7 environment.

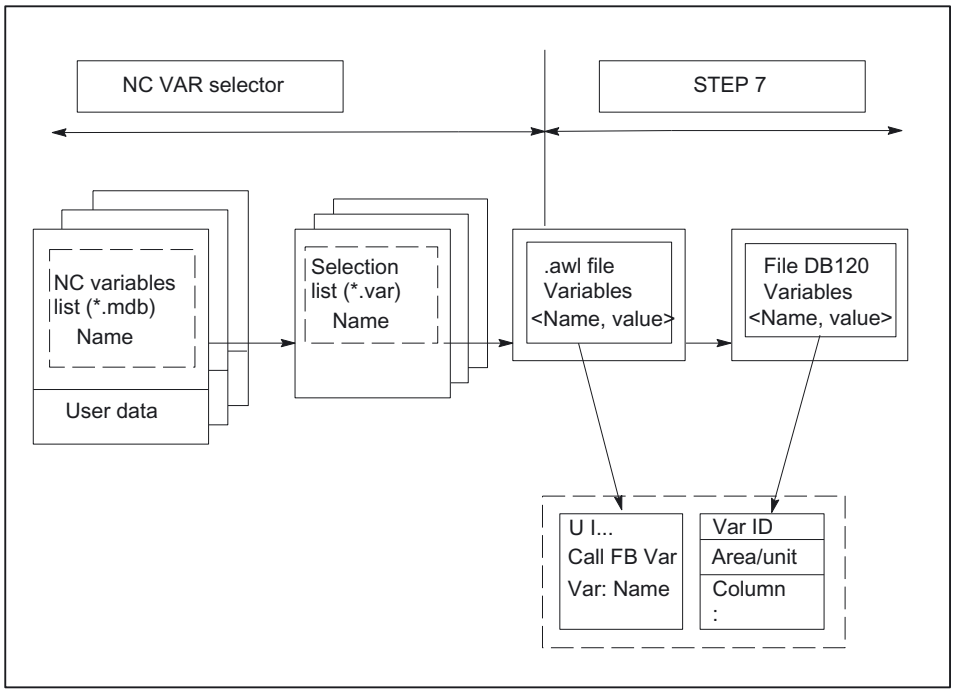


Fig. 2-21 Application of NC VAR selector in the STEP7 environment

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 STEP7 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 STEP7 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 STEP7 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.

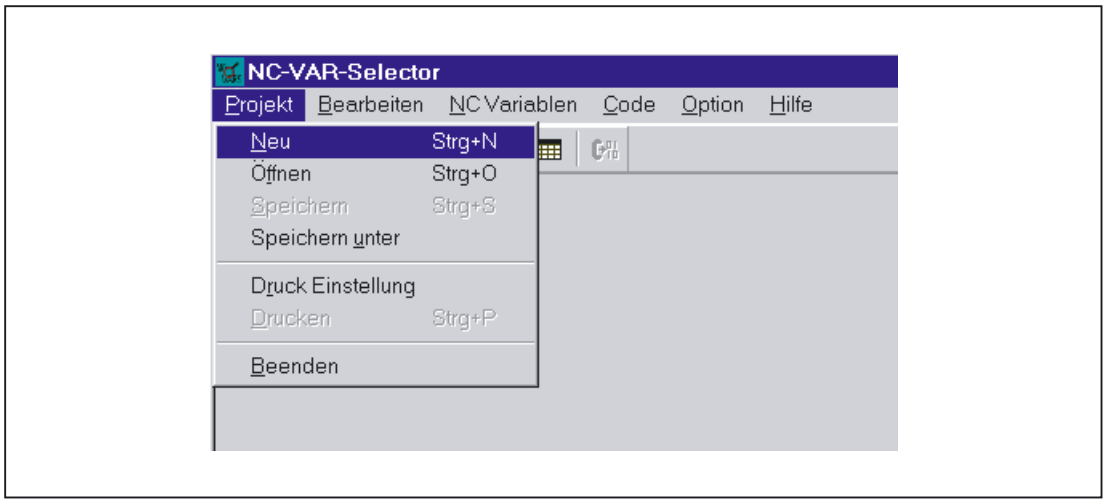


Fig. 2-22 Basic display with basic menu

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).

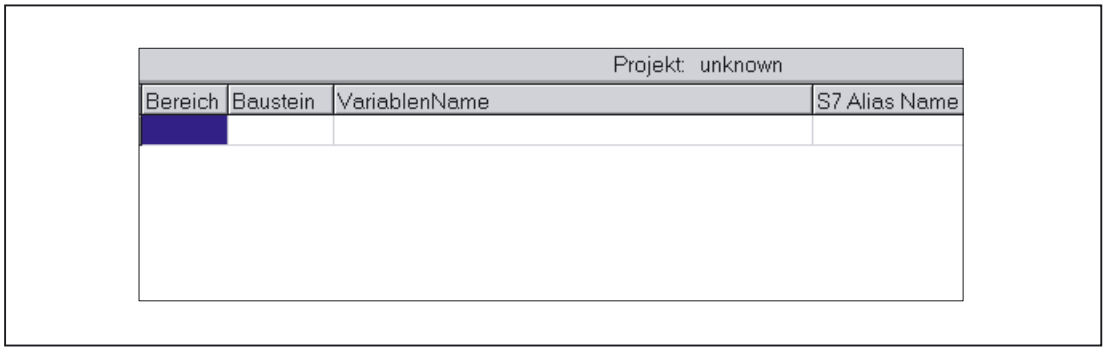


Fig. 2-23 Window with selected variables for new project

The selected variables are displayed in a window.

Opening an 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.

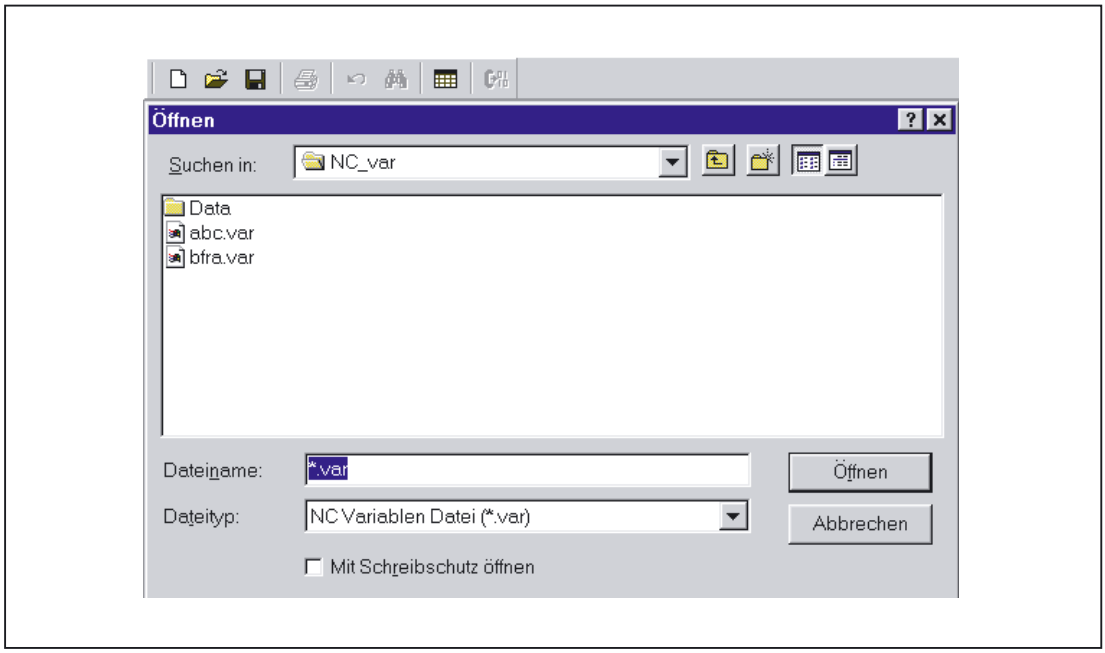


Fig. 2-24 Selection window for existing projects

If, after selection of the project, new variables are to be added, a complete list of NCK variables must be selected (see: Selecting complete list). 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 the NC Var selector path Data\Swxy (xy stands for SW no., e.g., SW 5.3:=xy=53). This list can be selected as an NC variables list. The available variable lists are provided thematically.

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.

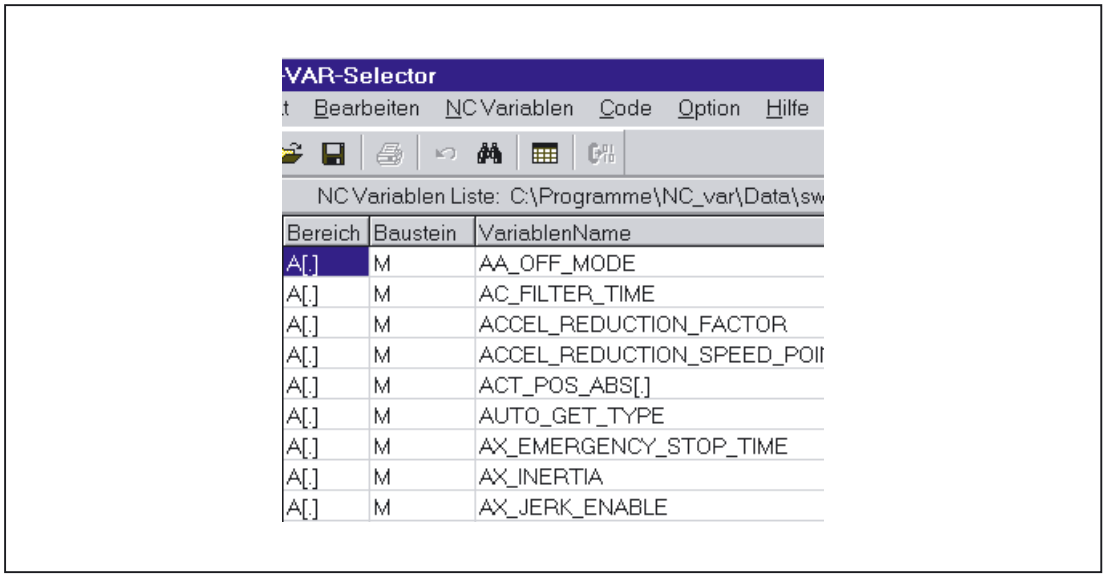


Fig. 2-25 Window with selected Complete List

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.

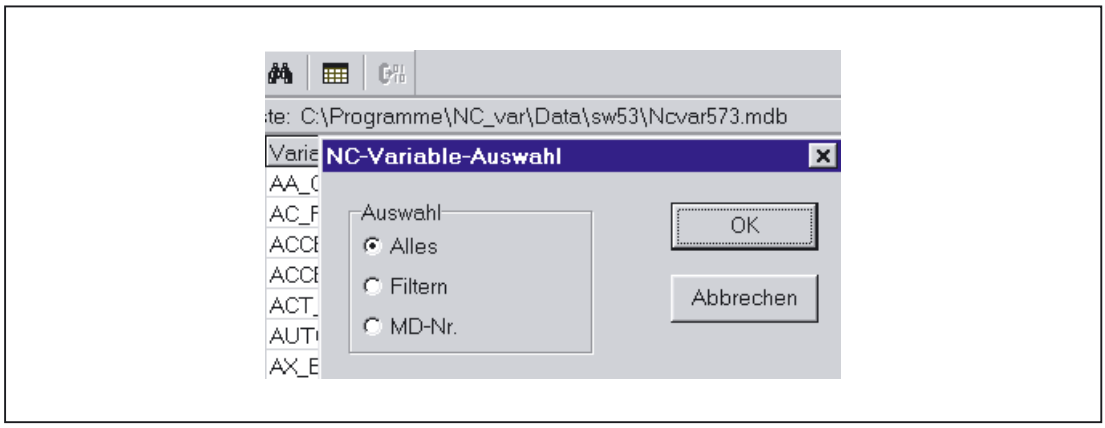


Fig. 2-26 Window with filter criteria for displaying list of variables

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

- Selecting variables
  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 STEP7 compiler recognizes only 24 ASCII characters as an unambiguous STEP7 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 STEP7 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.

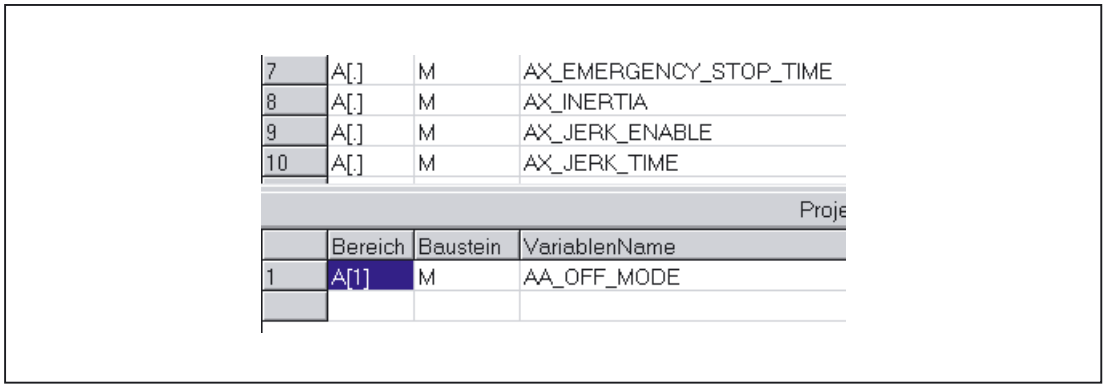


Fig. 2-27 Screen with complete list and selected variables

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 multidimensional 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:
/LIS/ **Lists;** 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.

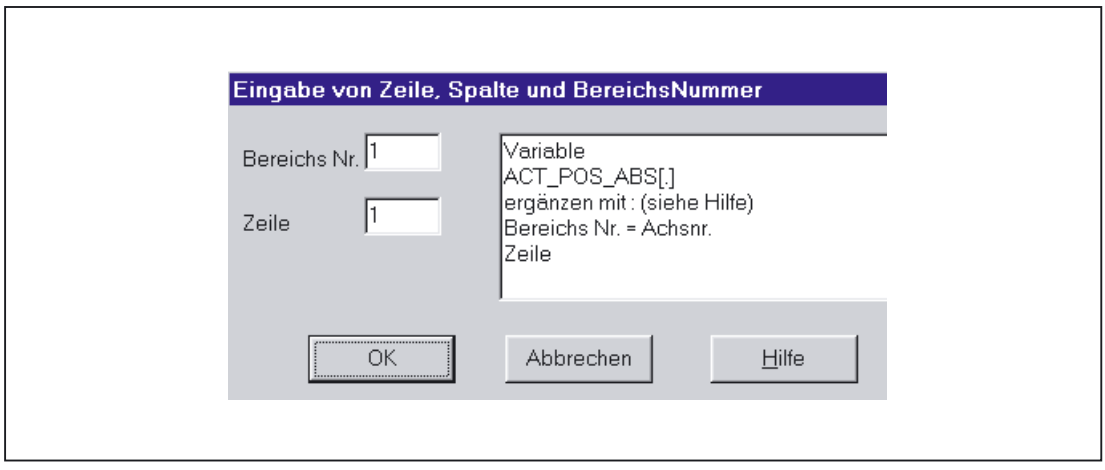


Fig. 2-28 Entry field for line, column and block no.

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 "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.

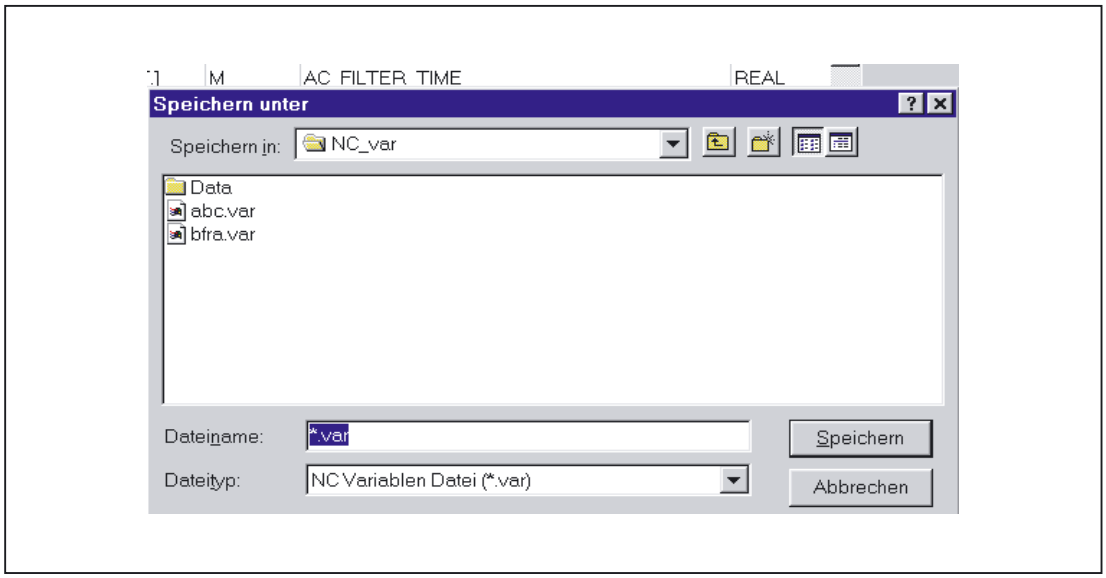


Fig. 2-29 Window for project path and name of file to be stored

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 STEP7 project (transferring the data block to a STEP7 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 "Unit 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 STEP7 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 STEP7 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 a STEP7 project

The generated AWL file is transferred to a selectable SIMATIC project (program path) and compiled. Furthermore, the symbol can also be transferred. This function will be available as of STEP7 Version 5.1 and NCVar Selector 6.04.05. This process take some time due to the time require by STEP7. Before transferring a new AWL file, the file window of the AWL file must be closed in the LAD/FUP/AWL 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 Operator's Guide
- The Description of Variables

The copyright and the version number can also be displayed.

2.11.2.3 Start-up, installation

The Windows application "NC Var selector" is installed using the SETUP program supplied with the package.

2.12 Block descriptions

2.12.1 FB 1: RUN_UP Basic program, start-up section

Description of functions

The synchronization of NCK and PLC is performed during start-up. 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 start-up routine, therefore, the CPUs perform "subsidiary start-up functions" and exchange ID information to ensure that the procedure is functioning correctly.
Since the start-up 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

An additional SDB 210 must be generated via the STEP7 NetPro tool for the **operator components** connected on the **MPI interface**.
The corresponding procedure is explained in the Installation and Start-up Guide. For more information about the assignment of MCP and HHU parameters, see "MCP/HHU configuration".

840D declaration

```
Code                                                Comment
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;
```

```
Code                                                    Comment
    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
    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 display
```

```
Code                                                    Comment
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;
END_VAR
```

Description of formal parameters 840D

The table below lists all formal parameters of the RUN_UP function for the 840D:

Signal	Type	Type	Value range	Remarks
MCPNum	I	Int	Up to 2	Number of active MCPs 0: No MCPs available
MCP1In MCP2In	I	Pointer	I0.0 to I120.0 or F0.0 to F248.0 or DBn DBX0.0 to DBXm.0	Start address for input signals of relevant machine control panel
MCP1Out MCP2Out	I	Pointer	Q0.0 to Q120.0 or F0.0 to F248.0 or DBn DBX0.0 to DBXm.0	Start address for output signals of relevant machine control panel
MCP1StatSend MCP2StatSend	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for data transmission to the machine control panel: DW#16#08000000: Timeout, otherwise 0
MCP1StatRec MCP2StatRec	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for data reception by machine control panel: DW#16#00000400: Timeout, otherwise 0
MCP1BusAdr MCP2BusAdr	I	Int	1...15 1 ... 126	MPI/MCP bus address of MCP DP slave: PROFIBUS address
MCP1Timeout MCP2Timeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for machine control panel

Signal	Type	Type	Value range	Remarks
MCP1Cycl MCP2Cycl	I	S5time	Recommendation: 200 ms	Time reference for cyclic updating of signals to machine control panel
MCPMPI	I	Bool		1: All machine control panels connected to the MPI bus (without GD parameterization)
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		0: No SDB 210 for MCP 1: Activate timeout monitors on SDB 210 for MCP
MCPCopyDB77	I	Bool		1: Copy between DB 77 and MCP pointers on DB 7 Can only be used with standard SDB 210 configuration on DB 77.
MCPBusType	I	Byte		0: MPI or MCP b#16#33: Profibusb #16#55: Ethernet Mixed operation possible, see MCP/HHU configuration
HHU	I	Int		HHU interface 0: No HHU 1: HHU on MPI with SDB 210 configuration 5: HHU on Ethernet
BHGIn	I	Pointer	I0.0 to I124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address PLC receive data from HHU
BHGOut	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address PLC transmit data to HHU
BHGStatSend	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for transmitting data to HHU: DW#16#08000000: Timeout, otherwise 0

Signal	Type	Type	Value range	Remarks
BHGStatRec	I	Pointer	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for receiving data from HHU: DW#16#00000400: Timeout, otherwise 0
BHGInLen	I	Byte	HHU default: B#16#6 (6 bytes)	Quantity of data received from handheld unit
BHGOutLen	I	Byte	HHU default: B#16#14 (20 bytes)	Quantity of data transmitted to handheld unit
BHGTimeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for handheld unit
BHGCycl	I	S5time	Recommendation: 100 ms	Time reference for cyclic updating of signals to handheld unit
BHGRecGDNo	I	Int	HHU default: 2	Receive GD circle no. or bus address
BHGRecGBZNo	I	Int	HHU default: 2	Receive GI no.
BHGRecObjNo	I	Int	HHU default: 1	Object number for receive GI
BHGSendGDNo	I	Int	HHU default: 2	Transmit GD circle no.
BHGSendGBZNo	I	Int	HHU default: 1	Transmit GI no.
BHGSendObjNo	I	Int	HHU default: 1	Object number for transmit GI
BHGMPI	I	Bool		1: Handheld unit coupled to MPI (without SDB 210 config.) Parameter HHU must be set to 2.
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	Value range	Remarks
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
MsgUser	I	Int	0...25	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	Default, false	Evaluate T function in OB40
IRAuxfuH	I	Bool	Default, false	Evaluate H function in OB40
IRAuxfuE	I	Bool	Default, false	Evaluate DL function in OB40
UserVersion	I	Pointer	DBxx	Pointer to string variable. The associated string variable is indicated in the version display (max. 41 characters).
MaxBAG	Q	INT	1..10	Number of mode groups
MaxChan	Q	INT	1..10	Number of channels
MaxAxis	Q	INT	1..31	Number of axes
ActivChan	Q	ARRAY[1..10] OF BOOL		Bit string for active channels
ActivAxis	Q	ARRAY[1..31] OF BOOL		Bit string for active axes
UDInt	Q	Int		Quantity of integer machine data in DB 20
UDHex	Q	Int		Quantity of hexadecimal machine data in DB20
UDReal	Q	Int		Quantity of real machine data in DB 20

MCP/HHU monitoring (840D)

For communication with the machine control panels (MCPs), in the event of an error, the following status information will be displayed (MPI, Ethernet only):

Available in:	Bit No.	Description
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Timeout
SINUMERIK 840D only: MCP1StatSend MCP2StatSend BHGStatSend	27	Transmitter: Timeout

In addition, an error entry is generated in the diagnostics buffer of the PLC, resulting in the following error messages on the HMI:

- 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.

840D example call

An example call for the FB 1 in OB 100 appears below. This example is part of the diskette with basic program for 840D.

```
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
```

2.12.2 FB 2: Read GET NC variable

Description of functions

The PLC user program can read variables from the NCK area using FB GET. 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-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 process lasts for several PLC cycles (normally 1-2). The block can be called up in cyclic mode only.

Any errors are indicated by 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.
"DB name.S7 name" is transferred as the actual parameter of the NCK variable address (Addr1 to Addr8) when FB 2 is called.

Variable addressing

For some NC variables, it is necessary to select area no. and/or line or column from the NC-VAR selector. A basic type can be selected for these variables,
i.e., area/column/line are preset to "0".
The contents of the area number, line and column specified by the NC VAR selector are checked for a "0" in the FB. If a "0" is present, the value is transferred to the input parameter. The user must supply the required parameters (UnitX/ColumnX/LineX) before calling FB GET.
Here unit corresponds to the area number.

Note

After communication between the PLC and NC (read/write NC variables, FB2, 3, 5, or PI general services, FB4) has been aborted by POWER OFF, the start jobs must be deleted in the first OB1 run after cold restart or reset (signal: Req = 0).

FB 2 can read NC variables only if basic program parameter NCKomm ="1" has been set (in OB 100: FB 1, DB7). The call is permitted only in cyclic program OB1.
When **channel-specific** variables are read, only the variables of
one and the same channel may be addressed in one job (FB 2 call) via Addr1 to Addr8.

In areas V and H, different logic axis numbers must not be assigned in one job.
(Failure to observe this rule results in Error:= TRUE, State:= W#16#02).

NCK variables within **one** group can be combined in a job:

	Area				
Group 1	C[1]	N	B	Q	T
Group 2	C[2]	N	B	Q	T
Group 3	V[.]	H[.]			

The same rules apply to channels 3 to 10 as illustrated as examples in the above table in groups 1 and 2.

Note

Especially when reading several long strings, the number of usable variables can be less than 8.

Declaration

```
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 table below list all formal parameters of the GET function.

Signal	Type	Type	Value range	Remarks
Req	I	Bool		Job start with positive signal edge
NumVar	I	Int	1 to 8 (corresponds to use of Addr1 to Addr8)	Number of variables to be read
Addr1 to Addr8	I	Any	[DBName]. [VarName]	Variable identifiers from **NC Var selector**
Unit1 to Unit8	I	Byte		Area address, optional for variable addressing
Column1 to Column8	I	Word		Column address, optional for variable addressing
Line1 to Line8	I	Word		Line address, optional for variable addressing
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
NDR	Q	Bool		Job successfully executed Data are available
State	Q	Word		See error identifiers
RD1 to 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 to 8	1	Access error	In high byte number of Var in which error occurred
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgment, job not executable	Internal error, try: NC reset
1 to 8	4	Insufficient local user memory available	Read var. is longer than specified in RD1 (to RD8); in high byte number of var in which error occurred
0	5	Format conversion error	Error on conversion of var. type double: Var. is not within S7 REAL area

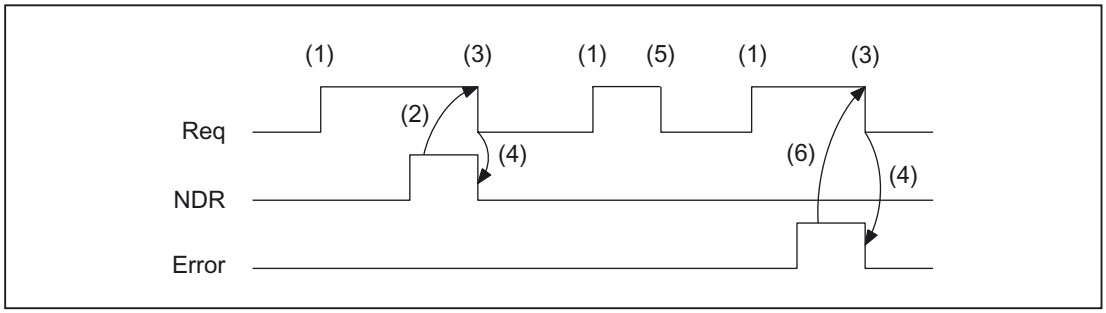
State		Meaning	Note
WORD H	WORD L		
0	6	FIFO full	Job must be repeated since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
1 to 8	8	Incorrect target area (RD)	RD1 to RD8 may not be local data
0	9	Transmission occupied	Job must be repeated
1 to 8	10	Error in variable addressing	Unit 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 to 8	13 (0x0d)	ANY date reference incorrect	NcVar date 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.
- Set FB 2 parameters

Pulse diagram



(1) Activation of function
(2) Positive acknowledgment: Receive new data
(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 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: AXCONF_MACHAX_USED[1]	char	20070	1	C1AxConfMachAxUsed1
C[1]	M	MD20070: AXCONF_MACHAX_USED[2]	char	20070	1	C1AxConfMachAxUsed2
C[1]	M	MD20090: SPIND_DEF_MASTER_SPIND	int	20090	1	C1SpindDefMasterSpind

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 DB 120):

Symbol	Operand	Data type
NCVAR	DB120	DB120

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 acknowledgment
    U        M 102.0;        //Error pending
    R        M 100.0;        //Terminate job
    Call fb 2, db 110(
             Req :=      M 100.0,
             NumVar :=   3,                    //Read 3 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 DB120 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 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
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

2.12.3 FB 3: PUT write NC variables

Description of functions

The PLC user program can write variables in the NCK area using FB PUT. 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 process lasts for several PLC cycles (normally 1-2). The block can be called up in cyclic mode only.

Any errors are indicated by 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. "DB name.S7 name" is transferred as the actual parameter of the NCK variable address (Addr1 to Addr8) when FB3 is called.

Variable addressing

For some NC variables, it is necessary to select area no. and/or line or column in the NC VAR selector. A basic type can be selected for these variables,
i.e., area/column/line are preset to "0".
The contents of the area number, line and column specified by the NC VAR selector are checked for a "0" in the FB. If a "0" is present, the value is transferred to the input parameter. The user must supply the required parameters (UnitX/ColumnX/LineX) before calling FB PUT.
Here unit corresponds to the area number.

Machine data, GUD

In order to define machine data and GUDs without a password, the protection levels of the data you want to access must be redefined to the lowest level. The procedure is described in the Installation Guide (in the Section describing the protection level concept) and in the Programming Guide Advanced (protection levels for user data).

Notice

FB 3 can only write NC variables if basic program parameter NCKomm has been set to "1" (in OB100: FB 1, DB 7). The call is permitted only in cyclic program OB1.

When channel-specific variables are written, only variables from one and the same channel may be addressed via Addr1 to Addr8 in one job. In areas V and H, different logic axis numbers must not be assigned in a single job. (Failure to observe this rule results in Error:= TRUE, State:= W#16#02).

NCK variables can be combined in a group in a single job: NCK variables can be combined in **one** group in a single job:

	Area				
Group 1	C[1]	N	B	Q	T
Group 2	C[2]	N	B	Q	T
Group 3	V[.]	H[.]			

The same rules apply to channels 3 to 10 as illustrated as examples in the above table in groups 1 and 2.

Note

Especially when reading several long strings, the number of usable variables can be less than 8.

Declaration

```
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	Value range	Remarks
Req	I	Bool		Job start with positive signal edge
NumVar	I	Int	1 to 8 (corresponds to use of Addr1 to Addr8)	Number of variables to be written
Addr1 to Addr8	I	Any	[DBName].[VarName]	Variable identifiers from **NC Var selector**
Unit 1 to Unit 8	I	Byte		Area address, optional for variable addressing
Column 1 to Column 8	I	Word		Column address, optional for variable addressing
Line 1 to Line 8	I	Word		Line address, optional for variable addressing
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
Done	Q	Bool		Job successfully executed
State	Q	Word		See error identifiers
SD1 to 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:

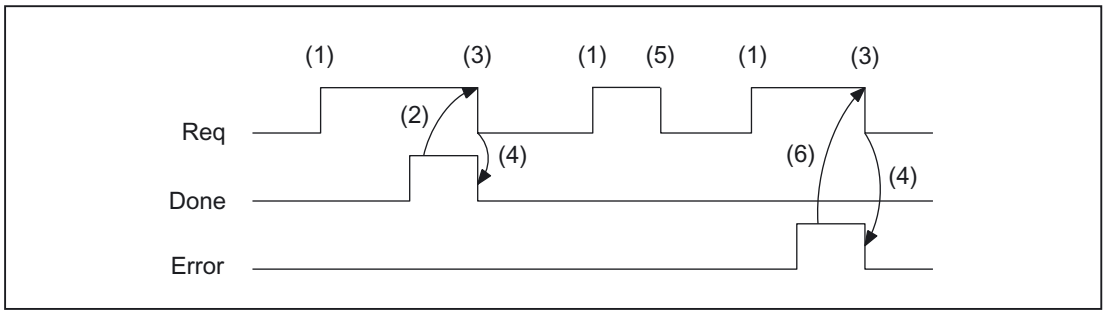
State		Meaning	Note
WORD H	WORD L		
1 to 8	1	Access error	In high byte number of Var in which error occurred
0	2	Error in job	Incorrect compilation of Var in a job

State		Meaning	Note
WORD H	WORD L		
0	3	Negative acknowledgment, job not executable	Internal error, try: Check job, NC Reset
1 to 8	4	Data areas or data types do not match or string is empty	Check data to be written in SD1 to SD8; in high byte number of the Var in which 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 to 8	8	Incorrect target area (SD)	SD1 to SD8 may not be local data
0	9	Transmission occupied	Job must be repeated
1 to 8	10	Error in variable addressing	Unit or column/line contains value 0
0	11	Variable addr. invalid or var. is read-only	Check Addr (or variable name), area, unit
0	12	NumVar = 0	Check parameter NumVar
1 to 8	13 (0x0d)	ANY data reference incorrect	NcVar date required has not been parameterized

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 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 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	DB120	DB120

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 acknowledgment
     U              M 102.0;            //Error pending
     R              M 100.0;            //Terminate job
     Call fb 3, db 111(
             Req :=     M 100.0,
             NumVar :=   3,                              //Write 3
                                                         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);
```

2.12.4 FB 4: PI_SERV General PI services

Description of functions

FB PI_SERV can be used to start program-instance services in the NCK area. The FB is multi-instance-capable. The possible services are described in this Section.

Recommendation: Use the expanded FB7 instead of FB4.

A program section, which carries out a particular function (e.g., with tool management, search for empty location in a magazine), is executed in the NCK by making a request via the PI service.
Every FB 4 call must be assigned an instance DB from the user area.

The specified service is referenced via the PIService parameter. The selected PI service is supplied via the freely assignable additional input variables with varying data types (Addr1 to Addr4 for strings, WVar1 to WVar 10 for integer or word variables). A 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 logic "1" in status parameter Done. Any errors are indicated by Error and State.
The "PI" data block (DB16) contains internal descriptions of the possible PI services. A name must then be assigned to this DB 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 extends over several PLC cycles (generally 1 to 2). The block can be called up in cyclic mode only.

Note

After communication between the PLC and NC (read/write NC variables, FB2, 3, 5, or PI general services, FB4) has been aborted by POWER OFF, the start jobs must be deleted in the first OB1 run after cold restart or reset (signal: Req = 0).

FB 4 can start PI services only if the basic program parameter NCKomm has been set to "1" (in OB100: FB 1, DB7). The call is permitted only in cyclic program OB1.

Declaration

```
FUNCTION_BLOCK FB 4
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 ;
END_VAR
VAR_OUTPUT
    Error :             BOOL ;
    Done :              BOOL ;
    State :             WORD ;
END_VAR
```

Description of formal parameters

The following table shows all formal parameters of the function PI_SERV.

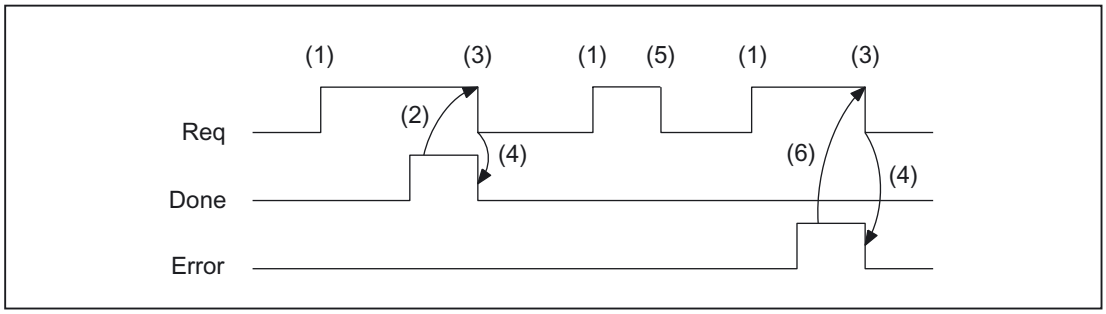
Signal	Type	Type	Value range	Remarks
Req	I	Bool		Job request
PIService	I	Any	[DBName].[VarName] default is: "PI".[VarName]	PI service description[1)]
Unit	I	Int	1...	Area number
Addr1 to Addr4	I	Any	[DBName].[VarName]	Reference to strings specification according to selected PI service
WVar1 to WVar10	I	Word	1...	Integers or word variables. Specification according to selected PI service.
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
Done	Q	Bool		Job successfully executed
State	Q	Word		See error identifiers
[1)] See README file on basic program diskette supplied				

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 block output State. The error identifiers, which may be encountered, are as follows:

State		Meaning	Note
3		Negative acknowledgment, job not executable	Internal error, try: NC reset
6		FIFO full	Job must be repeated since queue is full
7		Option not set	BP parameter "NCKomm" is not set
9		Transmission occupied	Job must be repeated
	13 (0x0d)	ANY date reference incorrect	String date required has not been parameterized
	14 (0x0e)	PIService parameter reference incorrect	No valid PI description

Pulse diagram



(1) Activation of function
(2) Positive acknowledgment: PI service has been executed
(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

Overview of PI services

The following Section provides an overview of the PI services that can be started from the PLC. The meaning and application of the general FB 4 input variables (Unit, Addr ..., WVar ...) depend on the individual PI service concerned.

PI service	Function	Available in 840D
ASUB	Assign interrupt	x
CANCEL	Execute cancel	x
CONFIG	Reconfiguration of tagged machine data	x
DIGION	Digitizing on	x
DIGIOF	Digitizing off	x
FINDBL	Activate block search	x
LOGIN	Activate password	x
LOGOUT	Reset password	x
NCRES	Initiate NC Reset	x
SELECT	Select program for processing for one channel	x
SETUFR	Activate user frame	x
PI service	Tool management function	
CRCEDN	Create new cutting edge	x
CREACE	Create cutting edge	x
CREATO	Generate tool	x
DELECE	Delete a cutting edge	x
DELETO	Delete tool	x
MMCSEM	Semaphores for various PI services	x
TMCRTO	Create tool	x
TMFDPL	Empty location search for loading	x
TMFPBP	Empty location search	x
TMMVTL	Prepare magazine location for loading, unload tool	x
TMPOSM	Position magazine location or tool	x
TMPCIT	Set increment value for workpiece counter	x
TMRASS	Reset active status	x
TRESMO	Reset monitoring values	x
TSEARC	Complex search using search screen forms	x
x: PI service is available		

Possible block types	
Block types	
Workpiece directory	WPD
Main program	MPF
Subprogram	SPF
Cycles	CYC
Asynchronous subprograms	ASP
Binary files	BIN

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.SELECT	Program selection
Unit	INT	1 to 10	Channel
Addr1	STRING		Path name
Addr2	STRING		Program name

PI service: ASUB

Assign interrupt

Function:

A program stored on the NCK is assigned an interrupt signal for a channel. This is possible only if the file may be executed. The path name and program name must be entered as described in the Programming Guide (Chapter "File and Program Management", Section "Program Memory"). Please also refer to example of FB 4 for notation of path and program names.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.ASUP	Assign interrupt
Unit	INT	1 to 10	Channel
WVar1	WORD	1 to 8	Interrupt number
WVar2	WORD	1 to 8	Priority
WVar3	WORD	0/1	LIFTFAST
WVar4	WORD	0/1	BLSYNC
Addr1	STRING		Path name
Addr2	STRING		Program name

Note

The SETINT instruction is also used to make the assignment.

The ASUB PI service may only be executed when the channel to be activated is in Reset.

References:
/PA/ Programming Guide

PI service: CANCEL

Execute cancel

Function:

The CANCEL command activates the Cancel function (in the same way as the key on the HMI).

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CANCEL	Cancel

PI service: CONFIG

Reconfiguration

Function:

The reconfiguration command activates machine data, which have been entered sequentially by the operator or the PLC, almost in parallel.
The command can only be activated when the control is in Reset or the program is interrupted (NC stop at block limit). An FB 4 error checkback signal is output if these conditions are not fulfilled (state = 3).

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CONFIG	Reconfiguration
Unit	INT	1	
WVar1	INT	1	Classification

PI service: DIGION

Digitizing on

Function:

Select digitizing in the specified channel.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.DIGION	Digitizing on
Unit	INT	1 to 10	Channel

PI service: DIGIOF

Function:

Deactivating digitizing in the specified channel

Parameterization:

Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.DIGIOF	Digitizing off
Unit	INT	1 to 10	Channel

PI service: FINDBL

Activate block search

Function:

A channel is switched to block search mode and the appropriate acknowledgment then transmitted. The block search is then executed immediately by the NCK. The search pointer must already be in the NCK 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. The operator is responsible for providing a collision-free approach path.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.FINDBL	Block search
Unit	INT	1 to 10	Channel
WVar1	WORD	x	Preprocessing mode
x: Describes the preprocessing mode x = 1 without calculation x = 2 with calculation x = 3 with main block observation			

PI service: LOGIN

Create password

Function:

Transfers the parameterized password to the NCK. The passwords generally consist of 8 characters. If required, blanks must be added to the string of the password.

Example:

Password: STRING[8] := 'SUNRISE?';

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.LOGIN	Create password
Unit	INT	1	NCK
Addr1	STRING	8 characters	Password

PI service: LOGOUT

Reset password

Function:

The password last transferred to the NCK is reset.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.LOGOUT	Reset password
Unit	INT	1	NCK

PI service: NCRES

Initiate NC Reset

Function:

Initiates an NCK Reset. The Unit and WVar1 parameters must be assigned 0.

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI.NCRES	Initiate NC Reset
Unit	INT	0	0
WVar1	WORD	0	0

PI service: Select

Function:

A program stored on the NCK is selected for processing for one channel. This is possible only if the file may be executed. The path name and program name must be entered as described in the Programming Guide (Chapter "File and program management", Section "Program memory"). Please also refer to example of FB 4 for notation of path and program names.

Possible block types

Workpiece directory	WPD
Main program	MPF
Subprogram	SPF
Cycles	CYC
Asynchronous subprograms	ASP
Binary files	BIN

Signal	Type	Value range	Meaning
PIService	ANY	PI.SELECT	Program selection
Unit	INT	1 to 10	Channel
Addr1	STRING		Path name
Addr2	STRING		Program name

PI service: SETUFR

Activate user frames

Function:

User frames are loaded to the NCK. All necessary frame values must be transferred to the NCK beforehand by writing variables with FB 3.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.SETUFR	Activate user frames
Unit	INT	1 to 10	Channel

Tool management services: PI service: CRCEDN

Create new cutting edge

Function:

If the T number of an existing tool is specified in parameter “T number” in the PI service, then a cutting edge is set up 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 fails. If a value of 00000 is entered for the T number (model of absolute D numbers), then the D number value range might extend from 00001 to 31999. The new cutting edge is set up with the specified D number. If the specified cutting edge already exists, then the PI service is aborted in both cases.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CRCEDN	Create new cutting edge
Unit	INT	1, 2	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

PI service: CREACE

Create cutting edge

Function:

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.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CREACE	Create cutting edge
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: CREATO

Create tool

Function:

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.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.CREATO	Create tool
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: DELECE

Delete a cutting edge

Function:

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 fails. If a value of 00000 is entered for the T number (model of absolute D numbers), then the D number value range might extend from 00001 to 31999. If the specified cutting edge does not exist, then the PI service is aborted in both cases.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.DELETE	Delete cutting edge
Unit	INT		TOA
WVar1	INT		T number of tool for which the cutting edge must be deleted. 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

PI service: DELETO

Delete tool

Function:

Deletes the tool assigned to the transferred T number with all cutting edges (in TO, in some cases TU, TUE and TG (type 400), TD and TS blocks).

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.DELETO	Delete tool
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: MMCSEM

Semaphores for various PI services

For use by HMI and PLC

10 semaphores are provided for each channel. These protect critical functions for the HMI/PLC. By setting the semaphore for the corresponding function number, several HMI/PLC units can be synchronized with it in cases where a function contains a critical section with respect to data to be fetched by the NCK. Semaphores are managed by the HMI/PLC. A semaphore value of 1stipulates a Test & Set operation for the semaphore 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. The operation must be repeated later.

Note

On completion of the operation (reading data of this PI service) it is **essential** that the **semaphore** is **enabled again**.

Parameter:

WVar1=FunctionNumber

This function number represents a 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 ...    Reserved
10:

WVar2=SemaphorValue

0:       Reset semaphore
1:       Test and set semaphore
```

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.MMCSEM	Set semaphore
Unit	INT	1, 2 to 10	Channel
WVar1	INT	1 to 10	FunctionNumber
WVar2	Word	0, 1	SemaphoreValue

PI service: TMCRTO

Create tool

Function:

Creation of a tool with specification of an identifier, a duplo number and a T number (optional). 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 NCK. If a TU block exists, it will contain the data set for the tool.
After execution of the PI, 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 PI service MMCSEM for more information.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMCRTO	Create tool
Unit	INT	1, 2	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			

PI service: TMFDPL

Empty location search for loading

Function:
(dependent on 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, 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 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 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 PI service MMCSEM for more information.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMFDPL	Empty location for loading
Unit	INT	1, 2	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

PI service: TMFPBP

Empty location search

Function:
(dependent on parameter assignment)

See description in FB 7.

PI service: TMMVTL

Prepare magazine location for loading, unload tool

Function:

This PI service is used both to load and unload tools. Whether the PI 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 movements.

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 <- "acknowledgment"

The following variables from the TMC block are used to monitor case 2), 6):

magCBCmd (area no. = TO unit)

magCBCmdState <- "acknowledgment"

Load function

Prepares the specified real magazine for the specified channel for loading, i.e., traverses the magazine 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, 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, the number of the location for tool loading is listed in the TM area in component **magCMCmdPar2** for the real magazine of the channel.

Unload function

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 via a T number or by means of the location and magazine numbers. The value -1 is entered at unused specification points.

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 PI service MMCSEM for more information.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMMVTL	Make magazine location ready for loading, unload tool
Unit	INT	1, 2	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

PI service: TMPOSM

Position magazine location or tool

Function:
(dependent on 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.

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").

or

- 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).

or

- 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").

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMPOSM	Position magazine location or tool
Unit	INT	1, 2	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

PI service: TMPCIT

Set increment value for workpiece counter

Function:

Incrementing the workpiece counter of the spindle tool

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI.TMPCIT	Set increment value for workpiece counter
Unit	INT	1 to 10	TOA
WVar1	WORD	0 ... Maximum	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 ... Maximum	Increment value; indicates the number of spindle revolutions after which the workpiece counter is incremented

PI service: TMRASS

Reset active status

Function:

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 VDI 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	Value range	Meaning
PIService	ANY	PI. TMRASS	Reset active status
Unit	INT	1 to 10	TO area

PI service: TRESMO

Reset monitoring values

This PI service resets the monitoring values of the designated edges of the designated tools to their setpoint (initial) values.
This only relates to tools with active monitoring.

Compare this with the RESETMON NC command.

Parameterization:

Signal	Type	Value range	Meaning
PIService	ANY	PI. TRESMO	Reset monitoring values
Unit	INT	1 to 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 ... Maximum	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: Total-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.

PI service: TSEARC

Complex search using search screen forms

Function:
(dependent on 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 operations. 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. This is achieved by selecting the relevant comparison criteria in the operand masks (parMaskT..) in the TF block (i.e., which tool data are to be compared?), entering the types of comparison logic (==, <, >, <=, >=, &&) in the comparison operator data (parDataT..), and entering the comparison values 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. The function number provided for the semaphore feature (PI service MMCSEM) is the function number for TSEARC.

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	MagNrTo	PlaceNrTo	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, location #P2 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 symmetrical search (see parameter "SearchDirection"), the search domain must only include one magazine (cases 2 and 5 in the table above). If another search domain is specified, the service will malfunction. A reference location must be specified in the parameters MagNrRef and PlaceNrRef for a symmetrical search (see parameter "SearchDirection"). The reference location is specified in the parameters MagNrRef and PlaceNrRef. 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 associated WVar1 parameters for this PI service. See PI service MMCSEM for more information.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TSEARC	Complex search using search screen forms
Unit	INT	1, 2	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: Symmetrical with real magazine location positioned in front of the location specified by MagNrRef and PlaceNrRef

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
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

Call example

Program selection in channel 1 (main program and workpiece program)

Entry of PI for DB 16 and STR for DB 124 with the S7 SYMBOL editor:

Parameterization:

Symbol	Operand	Data type
PI	DB16	DB16
STR	DB124	DB124

```
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]:=               //Main program
                                 '/_N_MPF_DIR/';
           PName_WST:            string[32]:=
                                 '_N_ABC_MPF';
            Path_WST:            string[32]:=               //Workpiece program
                                 '/_N_WKS_DIR/_N_ZYL_WPD';
   end_struct
BEGIN
END_DATA_BLOCK
Function FC "PICall" : VOID
   call fb4,db126(
   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 acknowledgment
U       M 1.0;                //Error pending
R       M 0.0;                //Terminate job
                  Req :=      M0.0,
                  PIService   PI.SELECT,
                  :=
                  Unit :=     1,                          // CHAN 1
                  Addr1 :=    STR.Path,
                  Addr2 :=    STR.PNam  //Main-program
                              e,        selection
                                        //Addr1:=STR.Path
                                        _WST,
                                        //Addr2:=STR.PNam
                                        e_WST,
                                        //Workpiece-
                                        program selection
                  Error :=    M1.0,
                  Done :=     M1.1,
                  State :=    MW2);
```

2.12.5 FB 5: GETGUD read GUD variable

Description of functions

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 is multi-instance-capable. 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 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 the 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 the variable pointer. GUD variable fields are an exception. In these, Line1 .. Line8 must also be parameterized with the field index of this variable.

Successful completion of the read process is indicated by a logic "1" in status parameter Done.

The read process extends over several PLC cycles generally 1 to 2).
The block can be called up in cyclic mode only.

Any errors are indicated by Error and State.

Note

After communication between the PLC and NC (read/write NC variables, FB2, 3, 5, or PI general services, FB4) has been aborted by POWER OFF, the start jobs must be deleted in the first OB1 run after cold restart or reset (signal: Req = 0).

FB 5 can read GUD variables only if basic program parameter NCKomm has been set to "1" FB 1, DB 7).

Declaration

```
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 = 1
    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	Remarks
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 variable 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 variable token
VarToken	I	Any	[DBName].[VarName]	Address to a 10byte token (see example)
Error	Q	Bool		Job negatively acknowledged or not executable
Done	Q	Bool		Job successfully executed.
State	Q	Word		See error identifiers
RD	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:

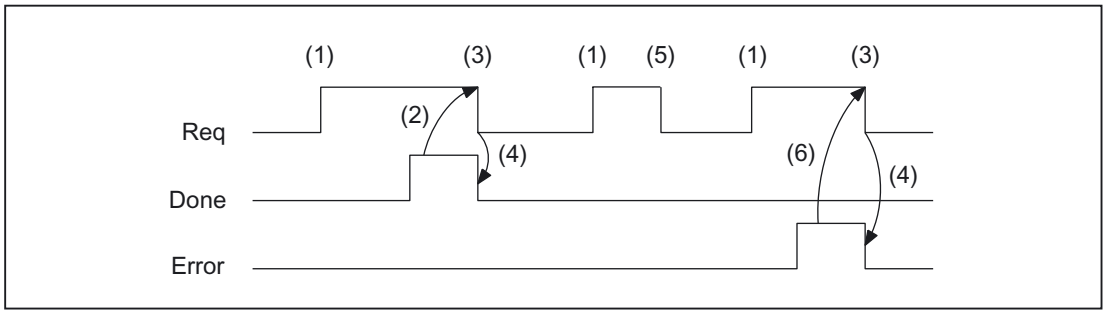
State		Meaning	Note
WORD H	WORD L		
0	1	Access error	
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgment, job not executable	Internal error, try: NC reset
0	4	Data areas or data types do not tally	Check data to be read in RD

State		Meaning	Note
WORD H	WORD L		
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	Check Addr (or variable name), area, unit
1 to 8	13 (0x0d)	ANY data reference incorrect	String/NcVar data required has not been parameterized

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 output parameter state

Call example

Reading of a GUD variable with the name "GUDVAR1" as an integer variable (see also table in FB 2: Assignment of NC data type in SIMATIC data type).
Call and parameterization of FB 5 with instance DB 111:

```
DATA_BLOCK DB GUDVAR                                    //Assignment to symbol table
    STRUC
    GUDVar1 : STRING[32] := 'GUDVAR1';                  //Name is defined by user
    GUDVar1T :
    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
DATA_BLOCK DB 111                                       //Unassigned user DB, as instance for FB 5
FB 5
BEGIN
END_DATA_BLOCK

DATE_BLOCK 112                                          //Unassigned user DB, as instance for FB 3
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 a 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 :=         GUDVAR.GUDVar1,
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 :=     GUDVAR.GUDVar1T,
Error :=        M102.0,
Done :=         M100.1,
State :=        MW104,
RD :=           P#DB99.DBX0.0 DINT 1
);
```

Following a successful FB5 call, fb3 can be written via the returned address.

```
Call fb 3, db 112(
Req := M 200.0,
NumVar := 1,                      //Write 1 GUD variable
Addr1 := GUDVAR.GUDVar1T,
Error := M102.0,
Done := M100.1,
State := MW104,
SD1:= P#DB99.DBX0.0 DINT 1);
```

2.12.6 FB 7: PI_SERV2 General PI services

Description of functions

A detailed description of the FB 7 is contained in the description of FB 4. The FB is multi-instance-capable. The call is permitted only in cyclic program OB1. The only difference to FB 4 is the number of WVar1 and subsequent parameters. In FB 7, WVar1 to WVar16 are defined in VAR_INPUT (FB4 has WVar1 to WVar10). All other parameters are identical to FB 4. This PI server can be used for all PI services previously implemented with FB 4. In addition, the PI services listed below can only be handled with FB 7.

Declaration

```
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

The following table shows all formal parameters of the function PI_SERV2.

Signal	Type	Type	Value range	Remarks
Req	I	Bool		Job request
PIService	I	Any	[DBName].[VarName] default 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...	Integers or word variables. Specification according to selected PI service
Error	Q	Bool		Negative acknowledgment of job or execution of job impossible
Done	Q	Bool		Job has been executed successfully
State	Q	Word		See error identifiers

Overview of additional PI services

The following section provides an overview of the PI services that can be started from the PLC. The meaning and application of the general FB 7 input variables (Unit, Addr..., WVar...) depends on the individual PI service concerned.

PI service	Function	Available in
		840D/810D
TMFPBP	Empty location search	+

Empty location search

TMFPBP

Function:
(dependent on parameter assignment)

This service searches the magazine(s) named in the relevant parameters for an empty location, which meets the specified criteria (tool size and location type). The result of the empty location search can be fetched from variables magCMCmdPar1 (magazine number) and magCMCmdPar2 (location number) in block TMC when the service has functioned correctly. As the PI service stores a result in variables magCMCmdPar1 and magCMCmdPar2, the service must be protected by the semaphore mechanism (PI service MMCSEM) with the function number for _N_TMFDPL in cases where several control units or PLCs are operating on one NC.

The search area can be predefined in the following way by setting parameters MagazineNumber_From, LocationNumber_From, MagazineNumber_To, LocationNumber_To:

Magazine Number _From	Location Number _From	Magazine Number_To	Location Number_To	Search area
WVar1	WVar2	WVar3	WVar4	
#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, location #P2 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

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 PI service MMCSEM for more information.

Parameter assignment			
Signal	**Type**	**Value range**	**Meaning**
PIService	ANY	PI.TMFPBP	Empty location search
Unit	INT	... max. TOA	TOA
WVar1	INT		MagazineNumber_From: Magazine number of magazine from which search must begin (start of search area)
WVar2	INT		LocationNumber_From: Location number of location in magazine MagazineNumber_From at which search must begin
WVar3	INT		MagazineNumber_To: Magazine number of magazine at which search must end
WVar4	INT		LocationNumber_To: Location number of location in magazine MagazineNumber_To at which search must end
WVar5	INT		MagazineNumber_Ref:
WVar6	INT		LocationNumber_Ref:
WVar7	INT	0, 1 .. 7	Number of required half locations to left
WVar8	INT	0, 1 .. 7	Number of required half locations to right
WVar9	INT	0, 1 .. 7	Number of required half locations in upward direction
WVar10	INT	0, 1 .. 7	Number of required half locations in downward direction
WVar11	INT	0, 1 .. 7	Number of required location type
WVar12	INT	0: default 1: forwards 2: backwards 3: Symmetrical	Specifies the required search direction 0: Empty location search strategy is set in $TC_MAMP2

2.12.7 FB 9: M : N operating-unit switchover

Description of functions

This block allows switchover between several **operating 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:
/FB2/ Description of Functions, Expansion 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 data block DB19 (see FB 2 above, Chapter 5 Signal Description). FB 9 uses 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 operating-unit switchover:

Tabulated overview of functions:	
Basic function	**Meaning**
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 HMI are connected online on one NCU simultaneously, one of the two is always in active 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 non-actuated signal level on a falling edge of DB10.DBX104.0. The override settings should remain unchanged.
Measures for deactivating keys must be implemented in the PLC user program (see below: Example of override switchover).

The call is permitted only in cyclic program OB1.

Declaration of function

```
FUNKTION_BLOCK FB9
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: Sign-of-life 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**	**Remarks**
Ack	I	BOOL	Acknowledge interrupts
OPMixedMode	I	BOOL	Mixed operation deactivated for OP without M to 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_MCP_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 not multi-instance-capable.

FB9 call

```
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 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 to N errors visualized on the HMI by the appearance of interrupts 410900 - 410906.

If execution of an HMI function has failed (and an appropriate error message cannot be displayed), status parameter ErrorMMC is set to 'logic 1' (e.g., booting error when no connection is made).

Example of FB1 call (call in OB100)

```
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
      MCP1Cycl                := S5T#200MS,
      MCP1Stop                := TRUE,         // MCP switched off
      NCCyclTimeout           := S5T#200MS,
      NCRunupTimeout           := S5T#50S);
```

Example of override switchover

```
// Auxiliary flags used M100.0, M100.1, M100.2, M100.3
// Positive edge of MCP1Ready must check override and actions for activation
// Initiate MCP block
// This example covers feed override; for spindle override, interfaces and
// input bytes must be exchanged.
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 assign. input or memory 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       EB28;                           //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       M100.3;                         //Yes, set auxiliary flag 2
// When override values match, call the MCP program again
MCP: CALL "MCP_IFM"(                    //FC 19
        BAGNo           := B#16#1,
        ChanNo          := B#16#1,
        SpindleIFNo     := B#16#0,
        FeedHold        := M 101.0,
        SpindleHold     := M 101.1);
wei2: NOP               0;
```

2.12.8 FB 10: Safety relay (SI relay)

Description of functions

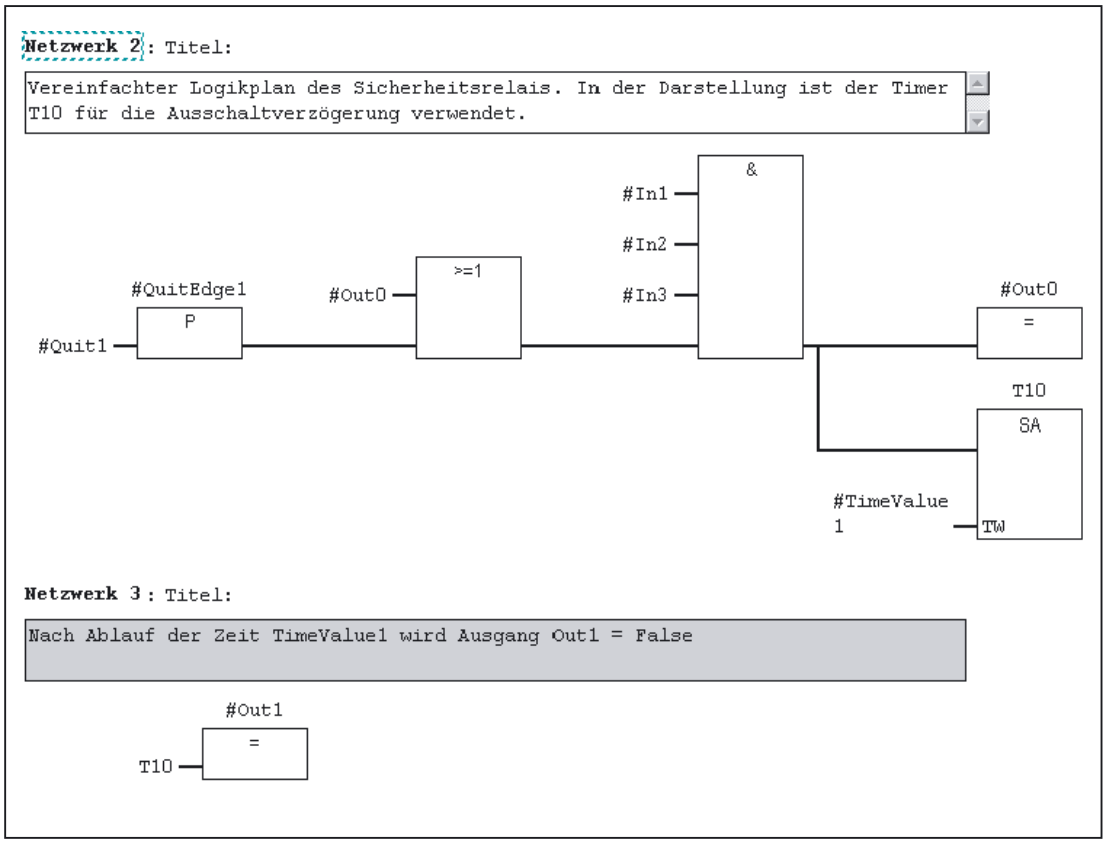
The SPL block "Safety relay" for "Safety Integrated" is the PLC equivalent of the NC function of the same name. The standard SPL "Safety relay" block is designed to support the implementation of an emergency stop function with safe programmable logic. However, it can also be used to implement other similar 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. Parameter FirstRun must be switched to the value TRUE via a retentive data (memory bit, bit in data block) on the first run after control booting. The 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 separate instance.

The corresponding NCK SPL block is described in:
References:
/FBSI/ Description of Functions, Safety Integrated.

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
    Quit1 :          : BOOL ;           //Ack 1 signal
    Ack2 :           :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 initial 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**	**Remarks**
In1	I	BOOL	Input 1
In2	I	BOOL	Input 2
In3	I	BOOL	Input 3
Ack1	I	BOOL	Acknowledge input 1
Ack2	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	Q	BOOL	Output undelayed
Out1	Q	BOOL	Output delayed by TimeValue1
Out2	Q	BOOL	Output delayed by TimeValue2
Out3	Q	BOOL	Output delayed by TimeValue3
FirstRun	I/O	BOOL	Activation of initial setting

Note

The block must be called cyclically by the user program once following SPL program start-up. The user must provide an instance DB with any number for this purpose. The call is multi-instance-capable.

2.12.9 FB 11: Brake test

Description of functions

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 so-called "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 apply additional torque/additional force on top of the weight force 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 required parameterization of NC and drive is described in:
References:
/FBSI/ Description of Functions, 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 servo disable, feedrate enable signals). The "PLC monitoring axis" (DB "Axis".DBX 28.7) must also be set to status 1 by the user program for the entire duration of the test. Before the "PLC monitoring axis" signal is activated, the axis must be switched to "neutral axis" status (e.g., byte 8 must be set to channel 0 in the axis DB, activation signal must be set in the same byte, current-status checkback can be polled in byte 68). The block must not be started until the NC checkback via the appropriate bit (DB "Axis".DBX 63.1) has arrived. 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

Note on FB 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 FC 18 repeatedly, whereby the inactive FC 18 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 checkback**	**Monitoring time value**
Start brake test	DBX 71.0 = 1	TV_BTactiv
Close brake	Bclosed = 1	TV_Bclose
Issue travel command	DBX 64.6 Or DBX 64.7	TV_FeedCommand
Issue test travel command	DBX62.5 = 1	TV_FXSreached
Wait for hold time	DBX62.5 = 1	TV_FXShold
Deselect brake test/ open brake	DBX71.0 = 0	TV_BTactiv
Issue Test O.K.		

Declaration of the function

FUNCTION_BLOCK FB 11

```
VAR_INPUT
    Start               : BOOL ;        //Start of brake test
    Ack                 : 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           : S5TIME ;      //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**	**Remarks**
Start	I	BOOL	Start brake test
Ack	I	BOOL	Acknowledge error
Bclosed	I	BOOL	Checkback input whether Close Brake is activated (single-channel - 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, check of axis signal DBX71.0
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 -> issue travel command 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	Q	BOOL	Request for close brake
MoveAxis	Q	BOOL	Request to initiate travel movement
Done	Q	BOOL	Test successfully completed
Error	Q	BOOL	Error occurred.
State	Q	BYTE	ErrorStatus

Error identifiers

Error identifiers	
State	**Meaning**
0	No fault
1	Start conditions not fulfilled, e.g., axis not under closed-loop control/brake closed/axis disabled
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 end stop cannot be reached. TV FXSreached monitoring timeout
7	Brake is not holding at all (end position is reached)/approach velocity too high
8	Brake opens during the holding period
9	Error in brake test deselection

Error identifiers	
State	**Meaning**
10	Internal error
11	"PLC monitoring axis" signal not activated by 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 FB11 call:

```
UN      M        111.1;    //Request to close brake, Z axis of FB
=       Q         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
L       B#16#10
T       DBB      8;        //Request neutral axis
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
R       DBX      8.4;
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
         Ack              := I    3.7,     //Acknowledge error with Reset key
         Bclosed          := I    54.0,    //Checkback signal, brake close
                                           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: Travel command
                                            issued
          TV_FXSreache    := S5T#1S,        //Monitoring time value: Fixed stop
                                            reached
          TV_FXShold       := S5T#2S,       //Monitoring time value: Test time Brake
          CloseBrake      :=M      111.1,   //Request to close brake
          MoveAxis        :=M      111.2,   //Request to initiate travel movement
          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
                  ";
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
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, 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
```

2.12.10 FB 29: Signal recorder and data trigger diagnostics

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 Boolean signals (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 (byte value), and Var2 and Var3 (integer values) is 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 is set up by the user. The DB of the ring buffer must be passed to the diagnostics FB in the BufDB parameter. The ring buffer must use an array structure, as specified 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 associated instance DB for the FB is a DB from the user area.

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.

FUNCTION_BLOCK FB 29

```
VAR_INPUT
Func : INT ;                                //Function number
//0 = No function, 1 = Signal recorder, 2 = Data trigger
    Signal_1 : BOOL ;
    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:

Signal	Type	Type	Value range	Remarks
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
VAR	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	Q	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 is complete (function 1), read out the ring buffer DB in STEP7 by 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
```

2.12.11 FC 2: GP_HP Basic program, cyclic section

Description of functions

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
                          Ack :=            I6.1);           //Acknowledgment of error
                                                             messages via I 6.1
END_ORGANIZATION_BLOCK
```

2.12.12 FC 3: GP_PRAL Basic program, interrupt-driven section

Description of functions

Block-synchronized transfers from the NCK to the PLC (auxiliary and G functions) are carried out in the interrupt-driven part of the basic program. **Auxiliary functions** are subdivided into normal and high-speed auxiliary functions. The high-speed 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. Normal auxiliary functions are only acknowledged when they have existed for the duration of one cycle. This allows the application to issue a read disable to the NC.

High-speed 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.

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. (only if 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 high-speed, job-driven 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 interrupt-driven 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 channel-specifically 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). The change signals and function value are available to the user in the associated channel DB. The change signal for this 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 and
                                        //indexing axes, spindles
GP_AuxFunction : ARRAY[1..10] OF BOOL;  //Channel-oriented for //auxiliary
                                        functions
GP_FMBlock : ARRAY[1..10] OF BOOL;      //Channel-oriented for block
                                        //transfer to FM (available soon)

//Further local user data may be defined from this point onwards
END_VAR
BEGIN
    CALL FC 3;
    //INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

2.12.13 FC 5: GP_DIAG Basic program, diagnostic alarm, and module failure

Description of functions

Description of functions

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. The PROFIBUS MCPs assigned on FB1 are excluded from stopping the PLC.

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 OB82 and OB86 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
```

2.12.14 FC 7: TM_REV Transfer block for tool change with revolver

Description of functions

After a revolver has been changed, the user will call this block. 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.

Output parameter "Ready" is set to the value TRUE when the job has been executed correctly.

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

It is not permissible to abort the transfer (e.g., by an external signal reset). 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. Machine data 20310 bit 12 must not be set to 1 for circular magazines.

Manual revolver switching

If a manual action is used to rotate the revolver, this information must be forwarded to the tool management. The asynchronous transfer function of FC 8 must be used to transfer the modified positions of the revolver. This must only occur once on the first manual rotation in the sequence. In this case, the following parameterization of the asynchronous transfer is needed via FC 8:

TaskIdent = 4
TaskIdentNo = channel
NewToolMag = Magazine number of the revolver
NewToolLoc = Original location of the tool
OldToolMag = Magazine no. buffer storage (spindle) = 9998
OldToolLoc = Buffer storage number of the spindle
Status = 1

This action also causes the same T command to be resent to the tool-management interface if the previous T is programmed again.

Declaration of the function

STL representation

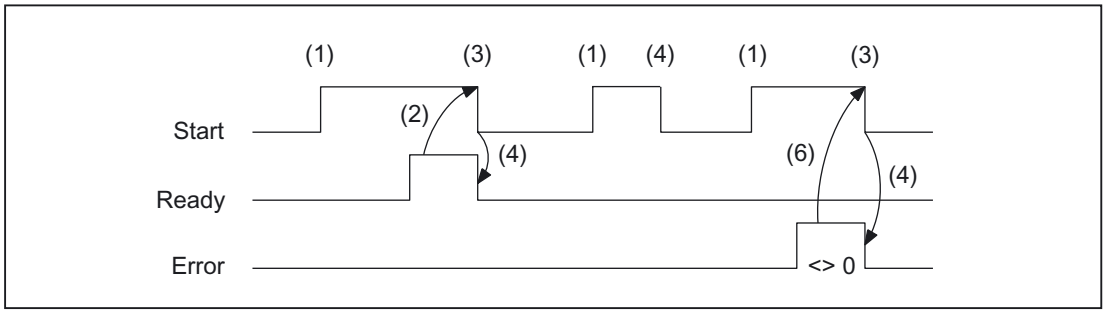
```
FUNCTION FC 7:          void
//NAME :TM_REV
VAR_INPUT
    Start:            BOOL ;
    ChgdRevNo:        BYTE ;
END_VAR
VAR_OUTPUT
    Ready:            BOOL ;
    Error :           INT ;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the TM_REV function.

Signal	Type	Type	Value range	Remarks
Start	I	Bool		= transfer is started
ChgdRevNo	I	Byte	1..	Number of revolver interface
Ready	Q	Bool		= transfer completed
Error	Q	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
(2) Positive acknowledgment: Tool management has been transferred
(3) Reset function activation after receipt of acknowledgment
(4) Signal change by means of FB
(5) This signal chart is not permissible. The job usually has to be terminated as the new tool positions need to be forwarded to the tool management in the NCK.
(6) Negative acknowledgment: Error has occurred, error code in output parameter error

Call example

```
CALL FC 7 (                 //Tool management transfer of 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
```

2.12.15 FC 8: TM_TRANS transfer block for tool management

Description of functions

In the case of changed tool positions or status changes, the user will call FC TM_TRANS. This FC is parameterized with parameter "TaskIdent":

1. For loading/unloading positions
2. For spindle change positions
3. For revolver change positions as transfer identifier
4. Asynchronous transfer
5. 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 transmitted 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 so-called "old tool" (e.g., on loading), the value 0 is assigned to parameters "OldToolMag", "OldToolLoc".

Block FC TM_TRANS may be started (with "Start" parameter = "TRUE") only 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.

Output parameter "Ready" is set to the value TRUE when the job has been executed correctly.

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 complete). 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 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

It is not permissible to abort the transfer (e.g., by an external signal reset). 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 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	Value range	Remarks
Start	I	Bool		1 = Start of transfer
TaskIdent	I	Byte	1..5	Interface or tank 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 new tool -1: Tool remains at its location NewToolLoc = any value. Only with TaskIdent = 2 permissible.
NewToolLoc	I	Int	0 to max. location number	Current location number of new tool

Signal	Type	Type	Value range	Remarks
OldToolMag	I	Int	-1, 0..	Current magazine number of tool to be changed -1: Tool remains at its location OldToolLoc = any value. Only with TaskIdent = 2 permissible.
OldToolLoc	I	Int	Max. location number	Current location number of tool to be changed
Status	I	Int	1..6, 103..105	Status information about transfer operation
Ready	Q	Bool		1= transfer completed
Error	Q	Int	0..65535	Error checkback 0: No error has occurred 1: Unknown "TaskIdent" 2: Unknown "TaskIdentNo" 3: Illegal job ("interface active" signal for 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

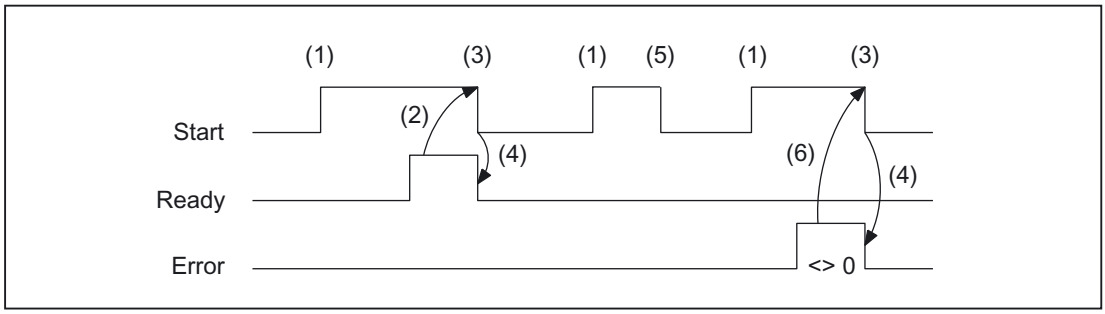


Fig. 2-30 Pulse diagram

(1) Activation of function
(2) Positive acknowledgment: Tool management has been transferred
(3) Reset function activation after receipt of acknowledgment
(4) Signal change by means of FB
(5) This signal chart is not permissible. The job usually has to be terminated since the new tool positions need to be forwarded to the tool management in the NCK.
(6) Negative acknowledgment: Error has occurred, error code in output parameter error

Status list

Status = 1:
The operation is complete (loading/unloading/reloading, prepare change, change).

The parameters of FC 8 (FC TM_TRANS) "NewToolMag", "NewToolLoc", "OldToolMag" and "OldToolLoc" must be parameterized to the tool target positions specified in the interface (except for Prepare change). For further information, please refer to description of parameters of FC TM_TRANS or general Section of tool management in the PLC.

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 tool-change 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 acknowledgment, 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 complete.

Status = 2:
The "new" tool cannot be made available.

This status is only admissible in conjunction with the "Change tool" 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 has 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 TM_TRANS. 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 take the specified magazine position into account in the subsequent command. However, it can only do so if the 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 "positioning to load point" is enabled, status 5 (not status 1) cannot be used for the function termination.

Status = 6:
The tool management job is 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 = 103:
The "new" tool can be inserted.

This status is only permissible in conjunction with the preparations for a tool change if the PLC may reject the new tool (MD: MC_TOOL_MANAGEMENT_MASK, bit 4). The tool positions have remained unchanged. This status is required to ensure that preprocessing continues in the NCK (otherwise processing is stopped).

References:
/FBW/ Description of Functions, 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, chain-type magazines and disk magazines for the "Tool change preparation" phase. Status 104 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 acknowledgment status is that the state information 1 to 7 leads to the termination of the command. If FC 8 receives one of the statuses, 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
```

2.12.16 FC 9: ASUB start-up of asynchronous subprograms

Description of functions

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 interrupt number must tally with the parameters in FC 9.
An ASUB prepared in this way can be started by the PLC at any point in time. 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 required 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. 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 start-up. 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 FB4 call must be terminated before FC9 can be started. FC9 cannot be started if "Emergency off" is set. Neither can FC9 be started if the channel reset is active.

Declaration

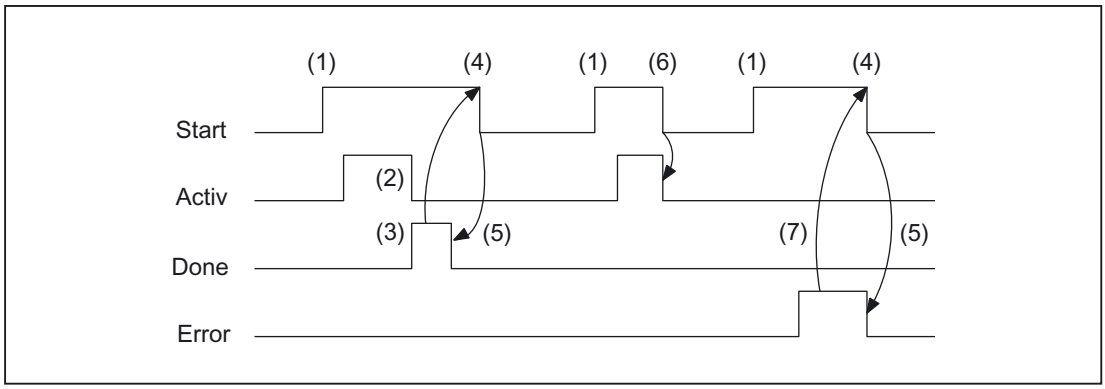
```
FUNCTION FC 9: VOID                                          //ASUB
VAR_INPUT
    Start:     BOOL ;
    ChanNo:    INT ;
    IntNo:     INT ;
END_VAR
VAR_OUTPUT
    Active:    BOOL ;
    Done :     BOOL ;
    Error :    BOOL ;
    StartErr:  BOOL ;
END_VAR
VAR_IN_OUT
    Ref:       WORD ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the ASUB function.

Signal	Type	Type	Value range	Remarks
Start	I	Bool		
ChanNo	I	Int	1 - 10	No. of NC channel
IntNo	I	Int	1 - 8	Interrupt no.
Active	Q	Bool		1 = Active
Done	Q	Bool		1 = ASUB terminated
Error	Q	Bool		1 = Interrupt switched off
StartErr	Q	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) ASUP active
(3) Positive acknowledgment: ASUB ended
(4) Reset function activation after receipt of acknowledgment
(5) Signal change by means of FB
(6) Not permitted If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.
(7) Negative acknowledgment: Error has occurred

Call example

```
CALL FC 9 (          //Start an asynchronous subroutine
                     //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);
```

2.12.17 FC 10: AL_MSG error and operating messages

Description of functions

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 operating 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, i.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;                  //Set disable condition, reset via
                            //FC AL_MSG, if M 50.0 outputs the signal "0".
```

Error and operational messages are provided by the user in data block DB 2.

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.

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	Value range	Remarks
ToUserIF	I	Bool		1 = Transfer 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); //Acknowledgment of
                                //error message carried out via
                                //input I6.1
```

2.12.18 FC 12: AUXFU call interface for user with auxiliary functions

Description of functions

FC AUXFU is generally called on an event-driven 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 job-driven 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

```
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	Remarks
Chan	I	Byte	0 to 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
```

2.12.19 FC 13: BHGDisp Display control for handheld unit

Description of functions

This block carries out the display control of the handheld unit (HHU). The information to be output on the display is stored as 32 characters in string data ChrArray. A fixed text assignment of 32 characters is, therefore, required for this string when the data block is created. Variable components within this string can be inserted by means of the optional numerical converter, for which 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. 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).

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

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.

Note

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. In this case, it is advisable to use the input signal from the HHU to the PLC "Acknowledgment digital display" (DB m+5.7) for parameter "Convert". In this way it can be ensured that the most recent numerical information is displayed.

Declaration of the function

STL representation

```
DATA_BLOCK "strdat"                                  //The data block number
                                                     //is defined in the symbol file
    STRUCT
    disp:           STRING [32]:= 'line 1 line 2  //32 characters are defined
                    2 ';
    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]
    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 to 32)
    Digits :          BYTE ;            //Number of decimal places (1 to 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	Remarks
Row	I	Byte	0-3	Display line 0: No display output 1: Line 1 2: Line 2 3: Line 1 and line 2
ChrArray	I	String	>= string[32] [DBName].[VarName]	This string contains the entire display content
Convert	I	Bool		Activation of numerical conversion
Addr	I	Pointer		Points to the variable to be converted
DataType	I	Byte	1-8	Data type of variable 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 (see parameter Digits)
StringAddr	I	Int	1-32	Address within variable ChrArray
Digits	I	Byte	1-4	Relevant only for data type Real with sign 1: 6.1 digits unsigned 2: 5.2 digits unsigned 3: 4.3 digits unsigned 4: 3.4 digits unsigned
Error	Q	Bool		Conversion error, numerical overflow or error in StringAddr

Ranges of values

Value ranges of data types	
Data type	**Representable numerical range**
BOOL	0, 1
BYTE	0 to 255
WORD	0 to 65535
INT	- 32768 to + 32767
DWORD	0 to 9999999
DINT	- 9999999 to + 9999999

Value ranges of data types	
Data type	**Representable numerical range**
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

Call example

```
CALL FC 13 (
           Row :=           MB 26,
           ChrArray :=      "strdat".disp,  //DB with name strdat in
                                            //data element symbol table
                                            //disp is declared as string[32]
                                            //and assigned in full with characters
           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);
```

2.12.20 FC 17: YDelta star/delta changeover

Description of functions

The block for 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:	Delete the "Motor selection in progress" interface signal in the relevant axis DB (DB 31, ... DBX21.5) and connect the changeover process using "Motor selection" A (DB 31, ... DBX21.5).
Step 2:	As soon as the "Pulses enabled" = 0 (DB 31, ... DBX93.7) checkback signal 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 "Motor selection in progress" (DB31, ... DBX21.5).

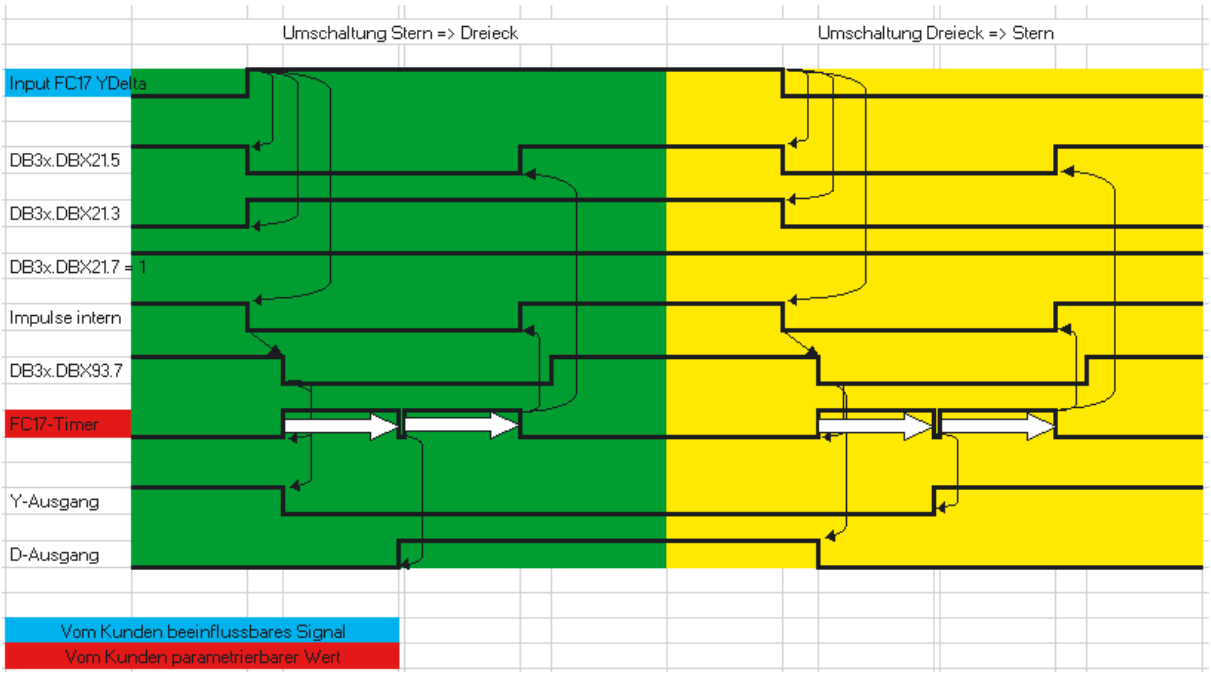


Fig. 2-31 Star/delta switchover

More information about motor speed adjustment can be found in:
References:
/FB/ Description of Functions, Basic Machine; Spindles (S1);
Section: Configurable Gear Adjustments
/FB/ Description of Functions, Basic Machine; Velocities, Setpoint/Actual Value Systems, Closed-Loop Control (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.

Restrictions

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 pulses on the drive, IS "Current controller active" (DB31, ... DBX61.7) and "Speed controller active" (DB31, ... DBX61.6) are deactivated simultaneously to IS "Pulses enabled" (DB31, ... DBX93.7).
- 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 interrupt 25050 "Contour monitoring".
- Once the star/delta changeover has been initiated with FC17, 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	Remarks
YDelta	I	Bool		= Star = Delta 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

Signal	Type	Type	Value range	Remarks
TimerNo	I	Int	10..	Timer for programming the wait time
Y	Q	Bool		Energizing of star contactor
Delta	Q	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
```

2.12.21 FC 18: SpinCtrl spindle control

Description of functions

FC SpinCtrl can be used to control spindles and axes from the PLC.

References:
/FB1/ Description of Functions, Basic Machine; Spindles (S1)
/FB2/ Description of Functions, Expansion Functions; Positioning Axes (P2)
/FB/ Description of Functions, Expansion 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" state 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 (DB 31, ...) (see interface lists). Once the axis/spindle is operating under PLC control, the travel command for the active state 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 reaction 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 axis status signals that may need to be evaluated (e.g., exact stop, traverse command).

InPos on spindle - rotation/oscillation:

For the "Rotate spindle" and "Oscillate spindle" functions, the meaning of the "InPos" parameter is defined as follows:
Setpoint speed is output --> function was started without error.
The reaching of the required spindle speed must be evaluated using 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, value from MD35300: SPIND_POSCTRL_VELO (position control activation speed) is used
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:	Error:	With positioning error = "1"
State:	Error code	

The oscillation speed is taken from machine data:
MD35400 SPIND_OSCILL_DES_VELO

MD35010 GEAR_STEP_CHANGE_ ENABLE = 0	Function	MD35010: GEAR_STEP_ CHANGE_ENABLE = 1	Function
Pos = 0	Oscillation	Pos = 0	
Pos = 1	Oscillation	Pos = 1	Oscillation with gear stage change M41
Pos = 2	Oscillation	Pos = 2	Oscillation with gear stage change M42
Pos = 3	Oscillation	Pos = 3	Oscillation with gear stage change M43
Pos = 4	Oscillation	Pos = 4	Oscillation with gear stage change M44
Pos = 5	Oscillation	Pos = 5	Oscillation with gear stage change M45

4. Traverse indexing axis:

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

```
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	Value range	Remarks
Start	I	Bool		Start spindle control from PLC
Stop	I	Bool		Stop spindle control from PLC
Funct	I	Byte	1 to B#16#0B	1: Position spindle 2: Rotate spindle 3: Oscillate spindle 4: Indexing axis 5: PosAxis metric 6: PosAxis inch 7: PosAxis metric with handwheel override 8: PosAxis inch with handwheel override 9: Rotate spindle with 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: Rotational direction 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	Q	Bool		1 = Position reached, or function executed
Error	Q	Bool		1 = error
State	Q	Byte	0 to 255	Error code

FRate

The feed rate in FC 18 can also be specified as:

1. Cutting rate with unit m/min or feet/min
2. Constant grinding wheel surface speed in m/s or feet/s

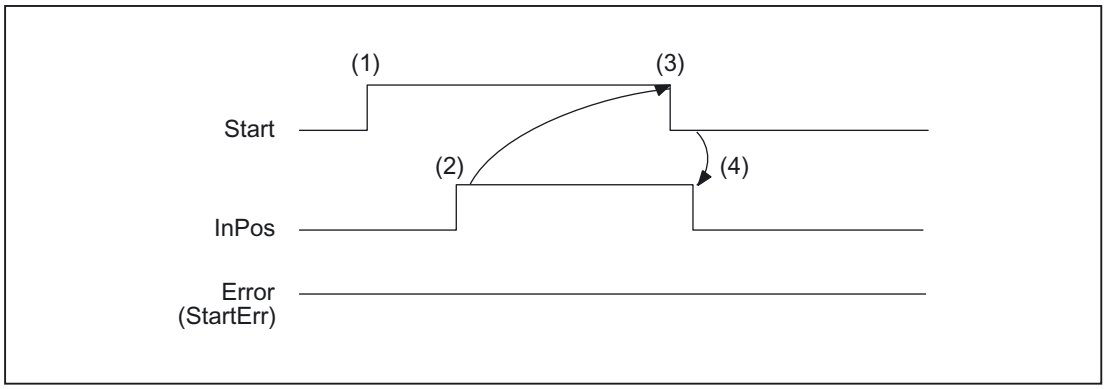
These alternative velocity settings can be used only if this function is activated by the NC program. Checkback signals for successful activation can be found in byte 84 on the axis interface.

Error identifiers

If a function could not be executed, this is indicated by the "Error" status parameter being set to 'logic 1'. The error cause is coded at block output "State":

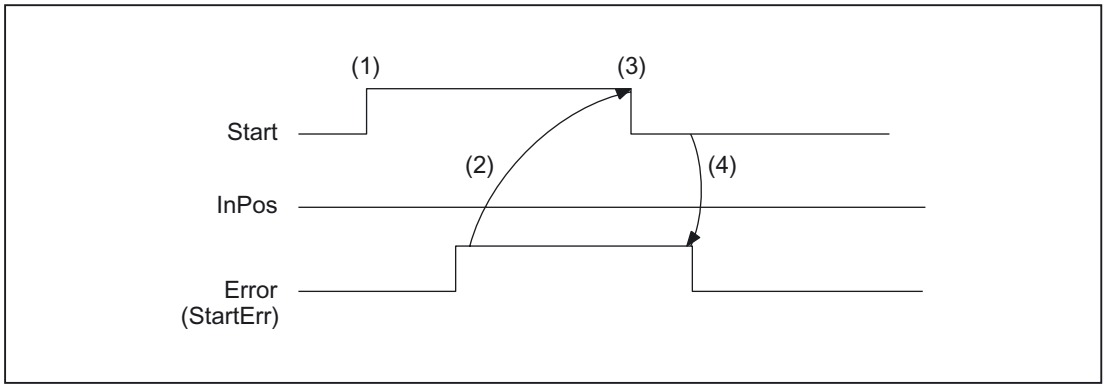
State		Meaning
Errors caused by PLC handling:		
1	B#16#1	Several axis/spindle functions have been activated simultaneously
20	B#16#14	A function has been started without the position being reached
30	B#16#1e	The axis/spindle has been transferred to the NC while still in motion
40	B#16#28	The axis is programmed by the NC program, NCK internal error
Error can be attributed to actions on the NCK. The interrupt numbers are described in the Diagnostics Guide for the 840D:		
100	B#16#64	Corresponds to interrupt number 16830
105	B#16#69	Corresponds to interrupt number 16770
106	B#16#6a	Corresponds to interrupt number 22052
107	B#16#6b	Corresponds to interrupt number 22051
108	B#16#6c	Corresponds to interrupt number 22050
109	B#16#6d	Corresponds to interrupt number 22055
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
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	Not an indexing axis
121	B#16#79	Indexing position error
125	B#16#7d	DC (shortest path) not possible
126	B#16#7e	Minus absolute value not possible
127	B#16#7f	Plus absolute value not possible
130	B#16#82	Software limit switch Plus
131	B#16#83	Software limit switch minus
132	B#16#84	Working area limitation plus
133	B#16#85	Working area limitation minus
System or other serious interrupts:		
200	B#16#c8	Corresponds to system interrupt number 450007

Pulse diagram



(1) Activation of function by means of a positive signal edge with start or stop
(2) Positive acknowledgment: Function executed/position reached
(3) Reset function activation after receipt of acknowledgment
(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 acknowledgment: Error has occurred
(3) Reset function activation after receipt of acknowledgment
(4) Signal change using FC

Call examples

1. Position spindle:

```
//Positive acknowledgment resets Start:
U M112.0;            //InPos
R M 100.0;           //Start
//Negative acknowledgment, 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 :=          Bq16q1,           //Position spindle
           Mode :=           Bq16q2,           //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 :=          Bq16q2,           //Rotate spindle
           Mode :=           Bq16q5,           //Rotational direction 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);
```

2.12.22 FC 19: MCP_IFM transmission of MCP signals to interface

Description of functions

FC MCP_IFM (M variants) is used to transfer the following from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode Groups
- Axis selections
- WCS/MCS switchover commands
- Traversing keys
- Overrides
- Keylock 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 geo-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 associated (machine) axis interface or the geo-axis interface of the handwheel selected (only if the "HWheelMMC parameter := TRUE" in FB 1).

The LEDs on the machine control panel operate on the basis of the checkback signals from the selections made.

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 MCP fails, the signals it outputs are preset to zero along with the "FeedHold" and "SpindleHold" output signals.

Multiple calls of FC 19, FC24, FC25, and even FC 26 are permitted within the same PLC cycle. In this case, the first call in the cycle activates 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 command is performed by this block once after power-up via DB10.DBX 57.0. Machine control panels can still be handled in parallel by this block. The block call for the 2nd machine control panel in the OB1 cycle must come after the call for the 1st MCP. Support for two MCPs is still provided in the control panel blocks up to certain limits (support is not provided as standard for mutual interlocking of axis selections 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 at byte 32 for the second MCP (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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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 a different set of axis numbers in the axis table (DB10) and switch off LED via this key.

Declaration of the function

```
FUNCTION FC 19: void
//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	Value range	Remarks
BAGNo	I	Byte	0 - b#16# 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	Q	Bool		Feed stop from MCP, modal
SpindleHold	Q	Bool		Spindle stop from MCP, modal

MCP selection signals to the user interface

Keylock 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

Operating modes and machine functions

Source: MCP button	Destination: Interface DB (parameter ChanNo)
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.DBB2, Bits 0 to 5

Direction keys rapid traverse override

The transfer is dependent upon the selected axis. The associated interface bits are deleted for axes, which are not selected.

Source: MCP button	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 button	Destination: Interface DB (all axis DBs)
Direction key +	DB31, ... DBX4.7
Direction key -	DB31, ... DBX4.6
Rapid traverse override	DB31, ... DBX4.5

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)

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

Feedrate, spindle signals

Source: MCP keys	Destination: FC output parameters
Feed stop Feed enable	Parameter: "FeedHold" latched, LEDs are activated
Spindle stop Spindle enable	Parameter: "SpindleHold" latched, LEDs are activated

Checkback signals from user interface for controlling displays

Operating modes and machine functions

Destination: MCP LED	Source: Interface DB (parameter ModeGroupNo)
AUTOMATIC	DB11, ... DBX6.0
MDI	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 ModeGroupNo)
INC 1 ... 10 000, INC Var.	DB11.DBB8, bits 0 to 5

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 activated by pressing the direction keys.
Axis selection and WCS/MCS LEDs are activated by pressing 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 to 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 required flexibility can be obtained by applying the solution described below.

1. Before FC 19 is called, the information (RLO) relating to the newly defined axis selection key is transferred to the key selection identified by an axis number.
2. After FC 19 has been called, the information (RLO) relating to the LED identified by the axis number is transferred to the LED of the new axis selection key and the RLO of the previous axis LED is then deleted.

Example:

The spindle is defined as the 4th axis and must be selected via axis key 9.

STL extract:

```
u i 5.2;                          //Selection of ninth axis
= e 4.2;                          //Selection of fourth axis
    call fc 19(
               BAGNo :          b#16#1,
               ChanNo :         b#16#1,
               SpindleIFNo :    b#16#4,
               FeedHold :       m 30.0,
               SpindleHold :    m 30.1);
u a 2.5;                          //LED fourth axis
= a 3.3;                          //LED ninth axis
clr;
= a 2.5;                          //Switch off LED on fourth axis
```

2.12.23 FC 21: transfer PLC NCK data exchange

Description of functions

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".

Declaration of 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	Remarks
Enable	I	Bool		1 = FC 21 active
Funct	I	Byte	1.. 7	1: Synchronized actions to 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	Q	Bool		
ErrCode	Q	Int		Depends on "Funct"

Functions

1: Signals for synchronized actions to channel:

2: Signals for synchronized actions from channel:

Synchronized actions can be disabled or enabled by the PLC.
The data area is stored on the user interface in DB21 to DB30.DBB 300..307 (to channel) and DBB 308..315 (from channel). 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	Remarks
Enable	I	Bool		1 = FC 21 active
Funct	I	Byte	1, 2	1: Synchronized actions to channel 2: Synchronized actions from channel
S7Var	I	Any	S7 data storage area	Not used
IVAR1	I	Int	1..MaxChannel	Channel number
Error	Q	Bool		
ErrCode	Q	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:           TO DB21;
    SET;
    S DBX300.2;     //ID3
    S DBX301.1;     //ID10
    S DBX303.6;     //ID31
    L B#16#1;
    T MB11;
    SPA TRAN;

// Synchronized actions from NCK channel:
SYVK:               L B#16#2;
                    T MB11;
```

```
TRAN: CALL FC 21 (
                     Enable :=          M 10.0,         // If True, FC 21 is active
                     Funct :=           MB 11,
                     S7Var :=           #myAny,         //Not used
                     IVAR1 :=           1,              //Channel no.
                     IVAR2 :=           0,
                     Error :=           M 10.1,
                     ErrCode :=         MW 12);
...
END_FUNCTION
```

Functions

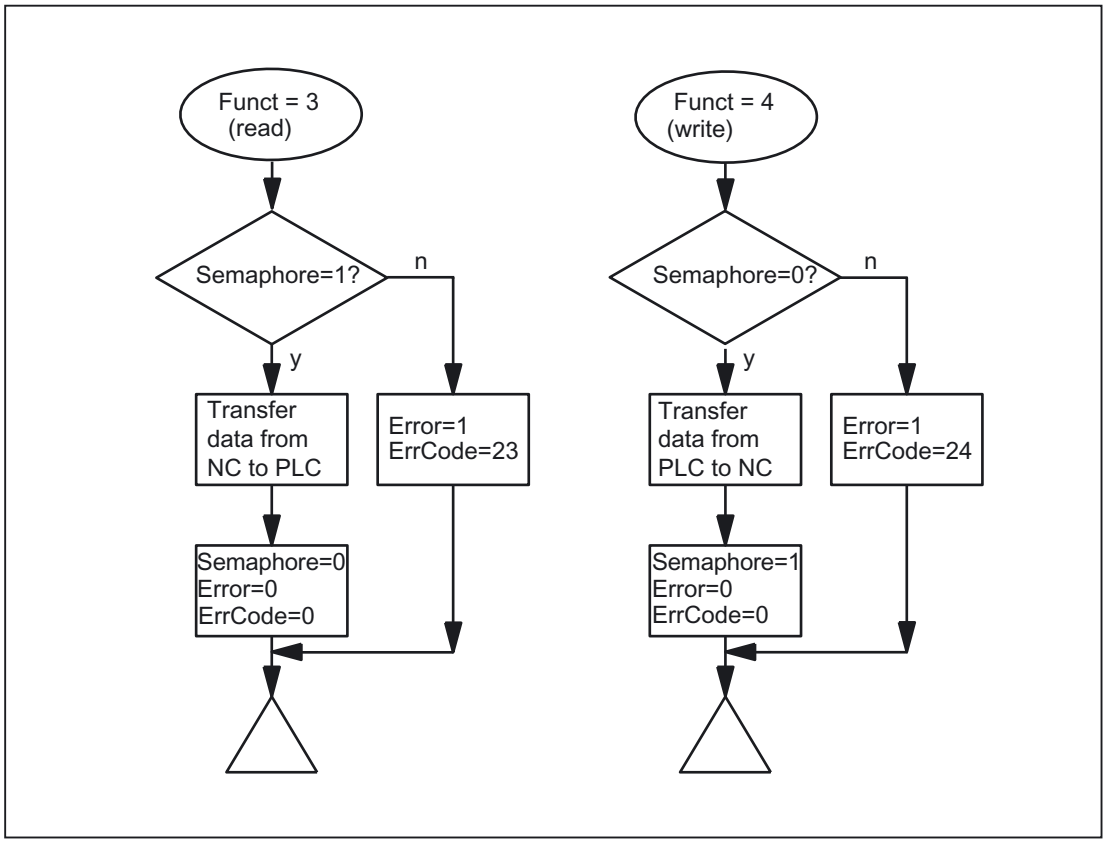
3, 4: Rapid PLC NCK data exchange

General

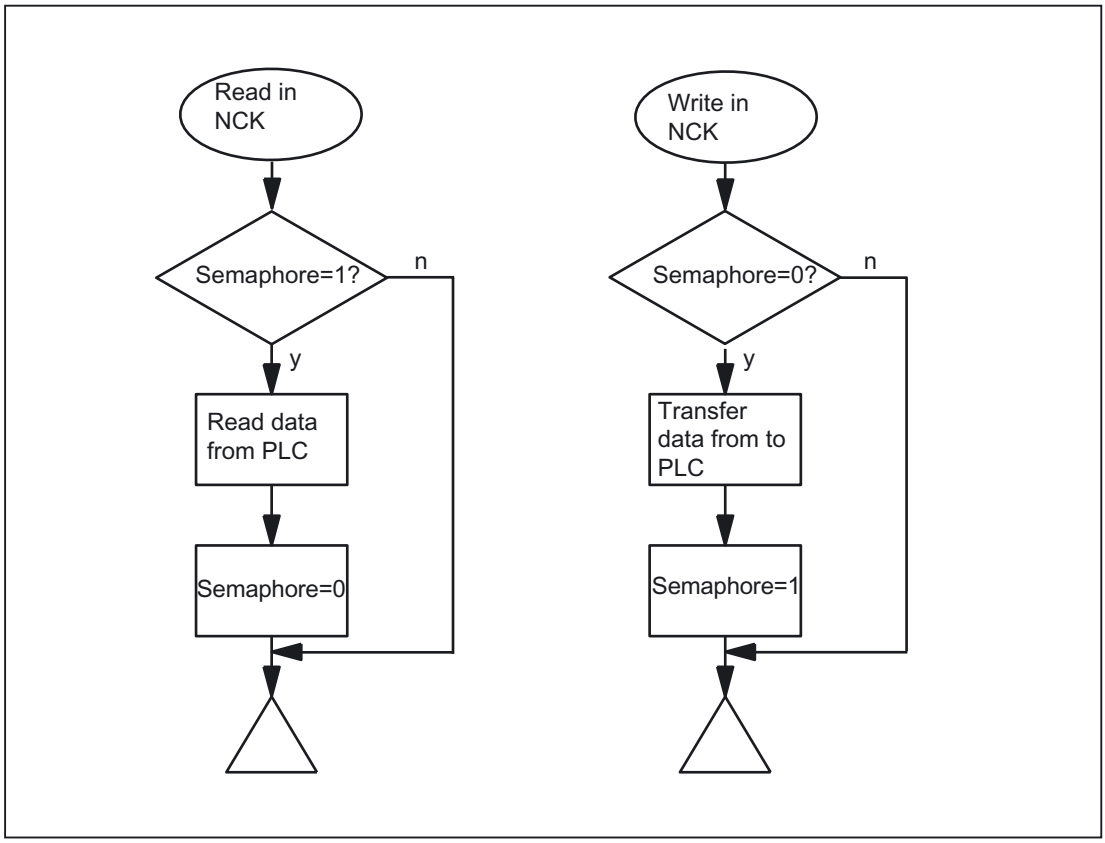
A separate, internal data area is provided to allow the high-speed exchange of data between the PLC and NCK. The size of the internal data field is preset to 1024 bytes. The NCK is accessed (read/written) from the PLC via FC21. The assignment of this area (structure) must be declared identically in both the NC parts 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 Programming Guide). The concrete address is the data field is specified by a byte offset (0 to 1023) in parameter IVAR1. In this case, the alignment must be selected according to the data format, i.e., a Dword starts at a 4-byte limit and a word at a 2-byte limit. Bytes can be positioned on any chosen offset within the data field, single-bit 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. However, 2 byte consistency is only ensured for the data types Word or Int, 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 1023 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 = -1, data are transferred without a semaphore.

Data exchange with semaphore in PLC (schematic of FC 21)



Basic structure in NCK:



Variable value ranges

The following signals are relevant:

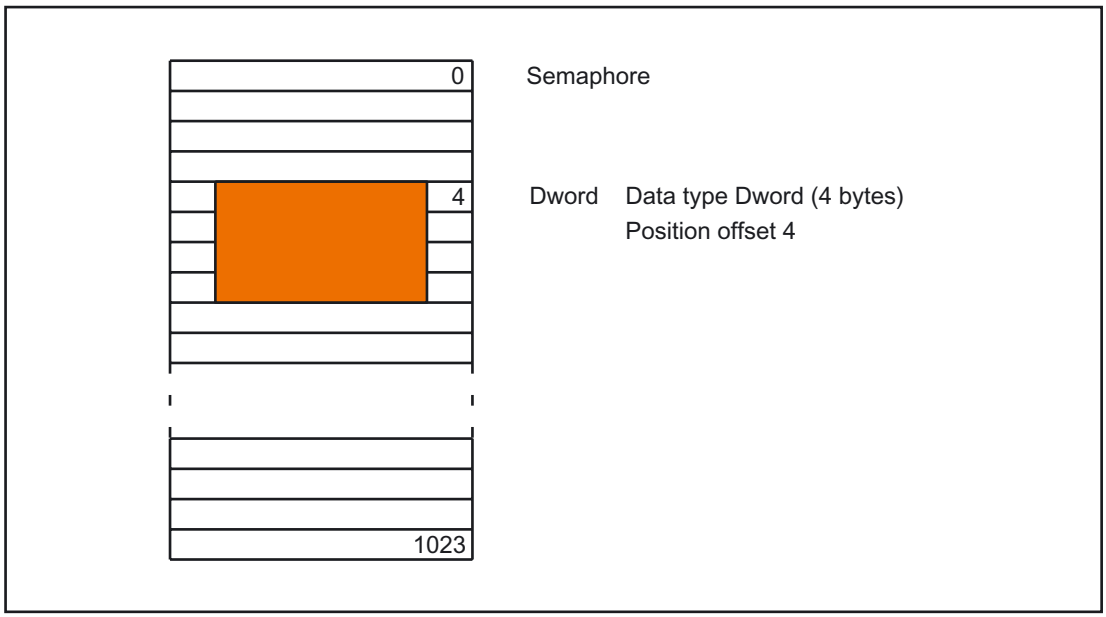
Signal	Type	Type	Value range	Remarks
Enable	I	Bool		= FC 21 active
Funct	I	Byte	3 ,4	3: Read data 4: Write data
S7Var	I	Any	S7 data area, except local data	Source/destination data storage area
IVAR1	I	Int	0..1023	Position offset
IVAR2	I	Int	-1 .. 1023	Semaphore byte Transfer without semaphore: -1
Error	Q	Bool		
ErrCode	Q	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 MD 100:

Data type Dword (4 bytes)

Position offset 4




```
CALL FC 21 (
           Enable :=      M 10.0,                    // If True, FC 21 is 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 F10.0;
```

Examples of NCK programming:

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 is 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 function 5 is to transmit important control signals at high speed in between cyclic data transfers. Data bytes 6 and 7 of user interfaces DB21 to DB30 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	Remarks
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	Q	Bool		
ErrCode	Q	Int		1: "Funct" invalid 10: Channel no. invalid

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. Data byte 2 of application interface DB31 to DB61 is transferred to the NC. The transfer is performed for all activated axes. This allows the servo enable to be transferred outside the PLC cycle, for example.

The following signals are relevant:

Signal	Type	Type	Value range	Remarks
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	Q	Bool		
ErrCode	Q	Int		1: "Funct" invalid

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. Data byte 4 of application interface DB31 to DB61 is transferred to the NC. The transfer is performed for all activated axes. This function can be used, for example, to transfer a feed stop outside the PLC cycle.

The following signals are relevant:

Signal	Type	Type	Value range	Remarks
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	Q	Bool		
ErrCode	Q	Int		1: "Funct" invalid

2.12.24 FC 22: TM_DIR Direction selection for tool management

Description of functions

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	Value range	Remarks
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	Q	Bool		1 = Move magazine clockwise
Ccw	Q	Bool		1 = Move magazine counterclockwise

Signal	Type	Type	Value range	Remarks
InPos	Q	Bool		1 = In position
Diff	Q	Int	0..	Differential path (shortest path)
Error	Q	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
              );
```

2.12.25 FC 24: MCP_IFM2 transmission of MCP signals to interface

Description of functions

With FC MCP_IFM2 (M variant slim operator panel), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- 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 geo-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 of the HMI** are decoded and activated in the associated (machine) axis interface or geo-axis interface of the relevant handwheel (only if FB 1 parameter "HWheelMMC := TRUE").

The LEDs on the machine control panel operate on the basis of the checkback signals from the selections made.

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 activates the LED displays. Moreover, all actions of the parameterized block are executed 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 command is performed by this block once after booting via DB10.DBX 57.0. Furthermore, two machine control panels can be handled in parallel by this block. The block call for the 2nd machine control panel in the OB1 cycle must come after the call for 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 at byte 32 for the second MCP (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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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.

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	Value range	Remarks
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. ModeGroupNo >= 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	Q	Bool		Feed stop from MCP, modal
SpindleHold	Q	Bool		Spindle stop from MCP, modal
SpindleDir	Q	Bool		Spindle direction 0 equals + (counterclockwise) 1 equals - (clockwise)

Call example

```
CALL FC 24 (           //Slim 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 to 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 checkback signal supplied via parameter SpindleDir can be used as a direction input for an additional FC 18 call.

2.12.26 FC 25: MCP_IFT transfer of MCP/OP signals to interface

Description of functions

With FC MCP_IFT (T variants), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- Direction keys of four axes
- WCS/MCS switchover commands
- Overrides
- Keylock 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 HMI. Transmission of HMI signals to the interface can be deactivated by setting the FB 1 (DB 7) parameter "MMCToIF" to "FALSE".

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 geo-axis interface of the parameterized channel.

The **handwheel selection signals from the HMI** are decoded and activated in the associated (machine) axis interface or the geo-axis interface of the handwheel selected (only if the "HWheelMMC parameter := TRUE" in FB 1).

The LEDs on the machine control panel operate on the basis of the checkback signals from the selections made.

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.

In the case of machine control panel failure, the signals it supplies are set to zero.

Multiple calls of FC 25 are permitted in a single PLC cycle. In this case, the first call in the cycle activates 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.
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. The block call for the 2nd machine control panel in the OB1 cycle must come after the call for the 1st 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 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 at byte 32 for the second MCP (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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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.

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	Value range	Remarks
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. ModeGroupNo >= 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	Q	Bool		Feed stop from MCP, modal
SpindleHold	Q	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 to 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.

2.12.27 FC 26: HPU_MCP Transfer of HPU/HT6 signals to the interface

Description of functions

With FC HPU_MCP (machine control panel signals of the handheld programming device), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- WCS/MCS switchover
- Traversing keys
- Override

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).

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 (for 6 axes).

  – When the WCS is selected, the signals of the first three axes are transferred to the geo-axis interface of the parameterized channel. The remaining three axes are transferred to the interface of the selected machine axis.

The **handwheel selection signals of the HMI** are decoded and activated in the associated (machine) axis interface or geo-axis interface of the relevant handwheel (only if FB 1 parameter "HWheelMMC := TRUE").

The LEDs on the machine control panel operate on the basis of the checkback signals from the selections made.

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 MCP fails, the signals it supplies are set to zero.

Multiple calls of FC 19, FC24, FC 25, and even FC 26 are permitted within the same PLC cycle. In this case, the first call in the cycle activates the LED displays. Moreover, all actions of the parameterized block are executed 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.

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 command is performed by this block once after power-up via DB10.DBX 57.0. Furthermore, two machine control panels can be handled in parallel by this block. The block call for the 2nd machine control panel in the OB1 cycle must come after the call for 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 at byte 32 for the second MCP (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 impermissible 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 limit is set for the appropriate MCP in DB10.DBW30 (symbolic name: MCP1MaxAxis) or DB10.DBW54 (symbolic name: MCP2MaxAxis). 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.

Declaration of the function

```
FUNCTION FC 26: void
// NAME:                HPU_MCP
VAR_INPUT
    BAGNo :             BYTE ;
    ChanNo:             BYTE ;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below shows all formal parameters of the "HPU_MCP" function.

Signal	Type	Type	Value range	Remarks
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	B#16#0A	Channel no. for the channel signals

2.12.27.1 MCP selection signals to the user interface

Operating modes and machine functions

Source: MCP key	Destination: Interface DB (parameter ModeGroupNo) representation for mode group 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.DBB2, Bits 0 to 5

Direction keys rapid traverse override

The transfer is dependent upon the selected axis. The corresponding interface bits are deleted for axes that are not selected.

Source: MCP button	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 button	Destination: Interface DB (6 assigned axis DBs)
Direction key +	DB31, ... DBX4.7
Direction key -	DB31, ... DBX4.6
Rapid traverse override	DB31, ... DBX4.5

Override

Source: MCP setting	Destination: Interface DB (parameter ChanNo)
Feedrate override	DB21, ... DBB4

Source: MCP setting	Destination: Interface DB (6 assigned axis DBs)
Feedrate override	DB31, ... DBB0

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

2.12.27.2 Checkback signals from user interface for controlling displays

Operating modes and machine functions

Operating modes and machine functions	
Destination: MCP output	**Source: Interface DB (parameter ModeGroupNo) representation for mode group 1**
AUTOMATIC	DB11, ... DBX6.0
MDI	DB11, ... DBX6.1
JOG	DB11, ... DBX6.2
REPOS	DB11, ... DBX7.1
REF	DB11, ... DBX7.2
TEACH IN	DB11, ... DBX7.0

WCS/MCS output is operated through key actuation.

Call example

```
CALL FC 26 (                            //Machine control panel of the HPU/HT6
                                        //Signals to interface
            BAGNo :=    B#16#1,         //Mode group no. 1
            ChanNo :=   B#16#1);        //Channel no. 1
```

With these parameter settings, the signals from the first parameterized machine control panel are sent to the 1st mode group, the 1st channel and all axes.

2.12.28 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 HPU/HT6 variant)

Associated blocks

DB 7, no. of BAGs, channels, axes

DB 7, pointer of machine control panel

DB 8, storage for the next cycle

FC 20, output of error messages

Resources used

None.

General

Blocks FC 19 (M version), FC 24 (slim-line version), FC 25 (T version) and FC 26 (HPU/HT6 version) transfer 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. "ChanNo" selects the channel to be processed. The "SpindleIFNo" parameter defines the axis interface of the spindle. The spindle override is transferred to this spindle interface. The parameters are checked for errors. 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 four blocks have a similar structure. The blocks are organized into separate sections for 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 start-up.

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 modifications should be made to this Section of the program. The user-specific modifications will usually mainly involve axis selections.

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. If the number is too large, message 401901 or 402501 is output and the control switches to Stop mode.

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. If the number is too large, message 401902 or 402502 is output and the control switches to Stop mode.

The Spindle network transfers the spindle override to the interface configured via SpindleIFNo.

The Axes network transfers the feedrate 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.

2.12.29 Signal/data descriptions

2.12.29.1 Interface signals NCK/PLC, MMC/PLC, MCP/PLC

General

The NCK/PLC, HMI/PLC and MCP/PLC interface signals are contained in the Lists document.

References:
/LIS/ Lists

for SINUMERIK 840D, with references to the Description of Functions in which the signals are described.
The NCK signals, which are evaluated and prepared by the basic program and transferred to the user interface, are listed in the following Sections.

2.12.29.2 Decoded M signals

General

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: DB21
- M functions from channel 2: DB22
- Etc.

The signal length is one PLC cycle.

Note

The spindle-specific M functions below are not decoded: M3, M4, M5, and M70.

Addresses in DB21, ...	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. In order to securely identify the end of a part program in the channel,
DB21, ... DBX33.5 (M02/M30 active)
must be evaluated. 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.

2.12.29.3 G Functions

General

The G 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: DB21
- G functions from channel 2: DB22
- 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 in the 8 G groups specified in the machine data element:
MD22510 $NC_ GCODE_GROUPS_TO_PLC
are overwritten according to the initial setting defined via the machine data, as are the values programmed in the part program.

Addresses in DB21, ...	Variable	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

Addresses in DB21, ...	Variable	Type	Basic position	Comment
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
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

G functions (values)

A full list of all G functions can be found in:
References:
/PG/ Programming Guide, Basics

2.12.29.4 Message signals in DB 2

General

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:
/IAD/ Installation and Start-Up Guide; Message Numbers

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.

Channel areas in DB 2

Area	Address	Message number
Channel 1	DBX0.0 - DBX11.7	510.000 -- 510.231
Channel 1, geo axes	DBX12.0 - DBX17.7	511.100 - 511.315
Channel 2	DBX18.0 - DBX29.7	520.000 - 520.231
Channel 2, geo axes	DBX30.0 - DBX35.7	521.100 - 521.315
Channel 3	DBX36.0 - DBX47.7	530.000 - 530.231
Channel 3, geo axes	DBX48.0 - DBX53.7	531.000 - 531.315
Channel 4	DBX54.0 - DBX65.7	540.000 - 540.231
Channel 4, geo axes	DBX66.0 - DBX71.7	541.100 - 541.315
Channel 5	DBX72.0 - DBX83.7	550.000 - 550.231
Channel 5, geo axes	DBX84.0 - DBX89.7	551.100 - 551.315
Channel 6	DBX90.0 - DBX101.7	560.000 - 560.231
Channel 6, geo axes	DBX102.0 - DBX107.7	561.100 - 561.315
Channel 7	DBX108.0 - DBX119.7	570.000 - 570.231
Channel 7, geo axes	DBX120.0 - DBX125.7	570.100 - 571.315
Channel 8	DBX126.0 - DBX137.7	580.000 - 580.231
Channel 8, geo axes	DBX138.0 - DBX143.7	581.100 - 581.315
Channels 9 and 10 are not currently implemented		

User areas in DB 2

Area	Address	Message number
Axis/spindle 1	DBX144.0 - DBX145.7	600.100 - 600.115
Axis/spindle 2	DBX146.0 - DBX147.7	600.200 - 600.215
Axis/spindle 3	DBX148.0 - DBX149.7	600.300 - 600.315
Axis/spindle 4	DBX150.0 - DBX151.7	600.400 - 600.415
Axis/spindle 5	DBX152.0 - DBX153.7	600.500 - 600.515
Axis/spindle 6	DBX154.0 - DBX155.7	600.600 - 600.615
Axis/spindle 7	DBX156.0 - DBX157.7	600.700 - 600.715
Axis/spindle 8	DBX158.0 - DBX159.7	600.800 - 600.815
Axis/spindle 9	DBX160.0 - DBX161.7	600.900 - 600.915
Axis/spindle 10	DBX162.0 - DBX163.7	601.000 - 601.015

Area	Address	Message number
Axis/spindle 11	DBX164.0 - DBX165.7	601.100 - 601.115
Axis/spindle 12	DBX166.0 - DBX167.7	601.200 - 601.215
Axis/spindle 13	DBX168.0 - DBX169.7	601.300 - 601.315
Axis/spindle 14	DBX170.0 - DBX171.7	601.400 - 601.415
Axis/spindle 15	DBX172.0 - DBX173.7	601.500 - 601.515
Axis/spindle 16	DBX174.0-DBX175.7	601.600 - 601.615
Axis/spindle 17	DBX176.0 - DBX177.7	601.700 - 601.715
Axis/spindle 18	DBX178.0 - DBX179.7	601.800 - 601.815
Axes 19 to 31 are not currently implemented		

User areas in DB 2

Area	Address	Message number
User area 0	DBX180.0 - DBX187.7	700.000 - 700.063
User area 1	DBX188.0 - DBX195.7	700.100 - 700.163
User area 2	DBX196.0 - DBX203.7	700.200 - 700.263
User area 3	DBX204.0 - DBX211.7	700.300 - 700.363
User area 4	DBX212.0 - DBX219.7	700.400 - 700.463
User area 5	DBX220.0 - DBX227.7	700.500 - 700.563
User area 6	DBX228.0 - DBX235.7	700.600 - 700.663
User area 7	DBX236.0 - DBX243.7	700.700 - 700.763
User area 8	DBX244.0 - DBX251.7	700.800 - 700.863
User area 9	DBX252.0 - DBX259.7	700.900 - 700.963
User area 10	DBX260.0 - DBX267.7	710.000 - 701.063
User area 11	DBX268.0 - DBX275.7	710.100 - 701.163
User area 12	DBX276.0 - DBX283.7	710.200 - 701.263
User area 13	DBX284.0 - DBX291.7	710.300 - 701.363
User area 14	DBX292.0 - DBX299.7	710.400 - 701.463
User area 15	DBX300.0 - DBX307.7	710.500 - 701.563
User area 16	DBX308.0 - DBX315.7	710.600 - 701.663
User area 17	DBX316.0 - DBX323.7	710.700 - 701.763
User area 18	DBX324.0 - DBX331.7	710.800 - 701.863
User area 19	DBX332.0 - DBX339.7	710.900 - 701.963
User area 20	DBX340.0 - DBX347.7	702.000 - 702.063
User area 21	DBX348.0 - DBX355.7	702.100 - 702.163
User area 22	DBX356.0 - DBX363.7	702.200 - 702.263
User area 23	DBX364.0 - DBX371.7	702.300 - 702.363
User area 24	DBX372.0 - DBX379.7	702.400 - 702.463

2.12.30 Useful Tips on Programming with STEP 7

2.12.30.1 General

General

Some useful tips on programming complex machining sequences in STEP7 are given in the following. This information concentrates mainly on the handling of data type POINTER and ANY. Detailed information about the structure of data types POINTER and ANY can be found in Chapter "CPU register and storage of data" in STEP7 manual "Designing user programs".

2.12.30.2 Copying data

The following is an example of how to copy data at high speed from one DB into another.

```
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               42;                 //Transfer 84 bytes
    M001:
    L               DBW [AR1,P#0.0];    //Copy word-oriented
    L               DBW [AR1,P#0.0];
    T               DIW [AR2,P#0.0];
    +AR1            P#2.0;
    +AR2            P#2.0;
    TAK;
    LOOP            M001;
```

2.12.30.3 ANY and POINTER

Multi-instance DB

With version 2 and higher of STEP 7, FBs might have a multi-instance capability, i.e., they might incorporate multi-instance DBs. The primary characteristic of multi-instance DBs is that they can be used for various instances of FBs (see STEP 7 documentation), thus allowing the DB data quantity to be optimized.

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

For more complex programs using pointers and address registers in FBs, which are to be multi-instance capable, certain rules specified by the programmer must be complied with. With multi-instance 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 multi-instance 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. It is best to save the AR2 register of the instance in a local variable (VAR_TEMP).
The 'load pointer to an instance variable' command returns a pointer value at 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

```
FUNCTION_BLOCK FB 99                    Comment
VAR_INPUT
    varin: INT ;
END_VAR
VAR
    variable1: ARRAY[0 to 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. If cross-area
                                        processing is to take place, then an area should be skipped
                                        when these two pointers are added.
AD              DW#16#00FF_FFFF,        //Skipping of an area
LAR1                                    //Load into AR1
TAR2;
+AR1 AR2;                               //AR2 instance offset to be added
                                        //The ARRAY of variable1 can now be accessed indirectly via
                                        AR1.
L               DIW [AR1, P#0.0];       //E.g., access to first element
END_FUNCTION_BLOCK
```

Strings

General

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 STRING

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 STEP7 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 string is stored in byte 2 by the 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.

Determining offset addresses for data block structures

General

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.

```
FUNCTION FC 99: VOID                    Comment
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
```

Supplementary Conditions

3

3.1 Supplementary conditions

There are no supplementary conditions to note.

Examples

4

4.1 Examples

No examples are available.

Data Lists

5

5.1 Machine data

5.1.1 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)
Machine data in integer/hex format is operated in the NC as DWORD. Machine data in floating comma format are operated in the NC as FLOAT (IEEE 8 byte). They are stored in the NC/PLC interface and can be read by the PLC user program during PLC power-up from the DB 20.		

5.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
28150	MM_NUM_VDIVAR_ELEMENTS	Number of elements for writing PLC variables

Index

A

AS, 2-19
Assignment of DBs, 2-65
ASUBs, 2-43

B

BAG, 2-38
Basic PLC Program (P3)|Physical interfaces on 840D, 2-19
Basic PLC program (P3)|PLC interface on SINUMERIK 840D, 2-19

C

Concurrent axes, 2-43
Configurability of machine control panel, handheld unit, 2-54
Cyclic operation, 2-37
Cyclic signal exchange, 1-1

E

Event-driven signal exchange, 1-1, 1-2

F

FB 10
    Safety relay, 2-143
FB 11
    Brake test, 2-146
FB 2
    GET, 2-92
FB 29
    Signal recorder and data trigger diagnostics, 2-151
FB 3
    PUT, 2-99
FB 4
    PI_SERV General PI services, 2-106
FB 5
    GETGUD read GUD variable, 2-129
FB 7
    PI_SERV2 General PI services, 2-134
FB 9
    Operating unit switchover, 2-84
FB 9 M to N operating-unit switchover, 2-138
FC 10
    AL_MSG, 2-174
FC 13
    BHGDisp, 2-177
FC 17
    YDelta, 2-180
FC 18
    SpinCtrl, 2-183
FC 19
    MCP_IFM, 2-194
FC 22
    TM_DIR, 2-208
FC 24
    MCP_IFM2, 2-210
FC 25
    MCP_IFT, 2-214
FC 7
    TM_REV, 2-160
FC 8
    TM_TRANS, 2-164
FC2
    GP_HP, 2-154
FC3
    GP_PRAL, 2-156, 2-158
FC9
    ASUB, 2-171

I

Interface
    840D, 2-19
    PLC/MCP, 2-33
    PLC/MMC, 2-30

M

M decoding acc. to list, 2-46
MAXNUM_USER_DATA_FLOAT, 5-1
MAXNUM_USER_DATA_HEX, 5-1

MAXNUM_USER_DATA_INT, 5-1
MD14504, 2-50
MD14506, 2-50
MD14508, 2-50
MD35400, 2-186
Memory requirements of basic PLC program, 2-66
  Maximum, 2-69
  Minimum, 2-68
Message signals in DB2, 2-226
MM_NUM_VDIVAR_ELEMENTS, 5-1

N

NC failure, 2-41
NC tags, 2-78
NC VAR selector, 2-72
  Start-up, installation, 2-84

P

PI services
  Overview, 2-109
PLC, 2-19
PLC messages, 2-30
PLC/NCK interface, 2-23
PLC_CYCLIC_TIMEOUT, 5-1
Process-interrupt processing, 2-40
Programming and parameterizing tools, 2-69
Programming devices or PCs, 2-69

R

Read/Write NC variables, 2-44

S

Signals
  PLC/axes, spindles, 2-29
  PLC/Mode group, 2-26, 2-27
  PLC/NC, 2-25
  PLC/NCK channels, 2-27
Start-up and synchronization of NCK PLC, 2-37
Symbolic programming of user program with interface DB, 2-45

U

Useful Tips on Programming with STEP 7, 2-229
Useful tips on programming with Step7
  Multi-instance DB, 2-229
  Strings, 2-231
USER_DATA_FLOAT[n], 5-1
USER_DATA_HEX[n], 5-1
USER_DATA_INT[n], 5-1

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Reference Point Approach (R1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 Axis-specific referencing 2-1

2.2 Channel-specific referencing 2-3

2.3 Reference point approach from part program (G74) 2-5

2.4 Referencing with incremental measurement systems 2-6
2.4.1 Chronological sequence 2-6
2.4.2 Phase 1: Traversing to the reference cam 2-7
2.4.3 Phase 2: Synchronization with the zero mark 2-10
2.4.4 Phase 3: Traversing to the reference point 2-16
2.4.5 Buffered actual value 2-19

2.5 Referencing with distance-coded reference marks 2-21
2.5.1 General overview 2-21
2.5.2 Basic parameter assignment 2-21
2.5.3 Chronological sequence 2-24
2.5.4 Phase 1: Reference mark search 2-25
2.5.5 Phase 2: Travel to fixed stop 2-27

2.6 Referencing with absolute value encoders 2-29
2.6.1 Information about calibration 2-29
2.6.2 Calibration by entering a reference point offset 2-31
2.6.3 Adjustment by entering a reference point value 2-32
2.6.4 Automatic calibration with probe 2-34
2.6.5 Calibration with BERO 2-36
2.6.6 Reference point approach with absolute encoders 2-37
2.6.7 Automatic encoder replacement detection 2-38
2.6.8 Enabling the measurement system 2-40
2.6.9 Referencing variants that are not supported 2-41

2.7 Referencing by means of actual value adjustment 2-41
2.7.1 Actual value adjustment to the referencing measurement system 2-41
2.7.2 Actual value adjustment to the referenced measurement system 2-42
2.7.3 Actual value adjustment for measuring systems with distance-coded reference marks 2-43

2.8 Referencing in follow-up mode 2-44

2.9 Zero mark selection with BERO 2-47

3 Supplementary Conditions 3-1

3.1 Large traverse range 3-1

4 Examples 4-1

4.1 Examples 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Channel-specific machine data 5-1
5.1.3 Axis/spindle-specific machine data 5-1

5.2 Signals 5-3
5.2.1 Signals to BAG 5-3
5.2.2 Signals from BAG 5-3
5.2.3 Signals to channel 5-3
5.2.4 Signals from channel 5-3
5.2.5 Signals to axis 5-4
5.2.6 Signals from axis 5-4

Index Index-1

Brief Description

1

1.1 Brief description

Functionality

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 follow-up mode
- Referencing with cam switch at the drive

Start

Reference point approach of a machine axis can be started by:

- Operating modes JOG and MDA, machine function REF: Manually
- Part program: Part program command G74

Detailed Description 2

2.1 Axis-specific referencing

In axis-specific reference point approach, reference point approach must be initiated individually for each machine axis that is to be referenced.

Selection of 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, ... DBB0.2 (JOG mode)

DB11, ... DBB0.1 (MDA mode)

Then, select machine function REF (reference point approach):

DB11, ... DBB1.2 (REF machine function)

Start 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, ... DBB4.6 (minus traversing key)

DB31, ... DBB4.7 (plus traversing key)

Direction enable

To avoid operator errors, direction enable must be parameterized.

The direction enable specifies which traversing key starts the reference point approach:

MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point 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

When reference point approach is started, the referencing status of the machine axis is reset:

DB31, ... DBB60.4 (referenced/synchronized 1)

DB31, ... DBB60.5 (referenced/synchronized 2)

DB21, ... DBB36.2 (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 type:

- SINUMERIK 840D: Max. 8 machine axes
- SINUMERIK 810D: Max. 5 machine axes

Terminating reference point approach

Acknowledgment that reference point approach of a machine axis has been successfully completed is given by setting the referencing status:

DB31, ... DBB60.4 (referenced/synchronized 1)

DB31, ... DBB60.5 (referenced/synchronized 2)

Aborting 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

Therefore, to cancel reference point approach, either a mode group reset:
DB11, ... DBX0.7 (mode group reset)
or a channel reset for the master channel of the machine axis:
DB21, ... DBB7.7 (reset)
must be initiated.

All machine axes that have not yet successfully completed reference point approach when the action is cancelled remain in status "Not referenced":

DB31, ... DBB60.4 (referenced / synchronized 1)

DB31, ... DBB60.5 (referenced / synchronized 2)

2.2 Channel-specific 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, ... DBB0.2 (JOG mode)

DB11, ... DBB0.1 (MDA mode)

Then, select machine function REF (reference point approach):

DB11, ... DBB1.2 (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	Meaning
-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 are referenced simultaneously.

Simultaneous reference point approach of several machine axes

Several machine axes can be referenced simultaneously depending on the control type:

- SINUMERIK 840D: Max. 8 machine axes
- SINUMERIK 810D: Max. 5 machine axes

Start reference point approach

Channel-specific reference point approach is started with:

DB21, ... DBB1.0 (activate referencing)

The status of channel-specific reference point approach is indicated by the channel with:

DB21, ... DBB33.0 (referencing active)

Referencing status

When reference point approach is started, the referencing status of the machine axis is reset:

DB31, ... DBB60.4 (referenced/synchronized 1)

DB31, ... DBB60.5 (referenced/synchronized 2)

Terminating 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)

Aborting 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.

To cancel reference point approach, either a mode group reset:
DB11, ... DBX0.7 (mode group reset)
or a channel reset for the relevant channel:
DB21, ... DBB7.7 (Reset)
must be initiated.

All machine axes that have not yet successfully completed reference point approach when the action is cancelled remain in status "Not referenced":

DB31, ... DBB60.4 (referenced/synchronized 1)

DB31, ... DBB60.5 (referenced/synchronized 2)

2.3 Reference point approach 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, ... DBB1.5 (position measuring system 1) = 0

  DB31, ... DBX1.6 (position measuring system 2) = 0
- DB31, ... DBB2.1 (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 points to be noted

G74 must be programmed in a separate part program block.

Reset response

Using mode group reset:
DB11, ... DBX0.7 (mode group reset)
or a channel reset:
DB21, ... DBB7.7 (reset),
reference point approach is cancelled for all programmed axes.

All machine axes that have not yet successfully completed reference point approach when the action is cancelled remain in status "Not referenced":

DB31, ... DBB60.4 (referenced/synchronized 1)

DB31, ... DBB60.5 (referenced/synchronized 2)

2.4 Referencing with incremental measurement systems

2.4.1 Chronological sequence

Reference point approach with incremental measuring systems can be divided into three phases:

- Phase 1: Traversing to the reference cam
- Phase 2: Synchronization with the zero mark of the position measuring system (encoder zero mark)
- Phase 3: Traversing to the reference point

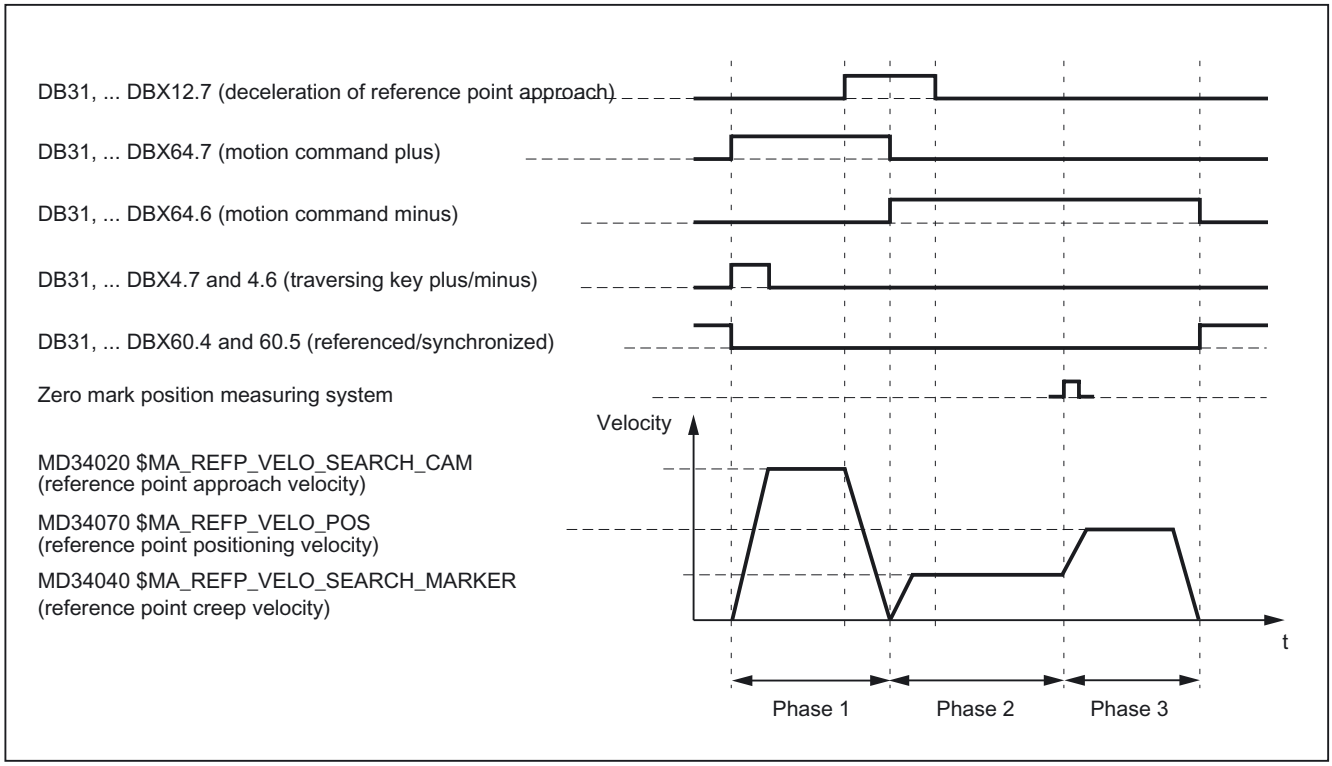


Fig. 2-1 Time sequence when referencing with incremental measuring systems (example)

2.4.2 Phase 1: Traversing to the reference cam

Phase 1: Graphic representation

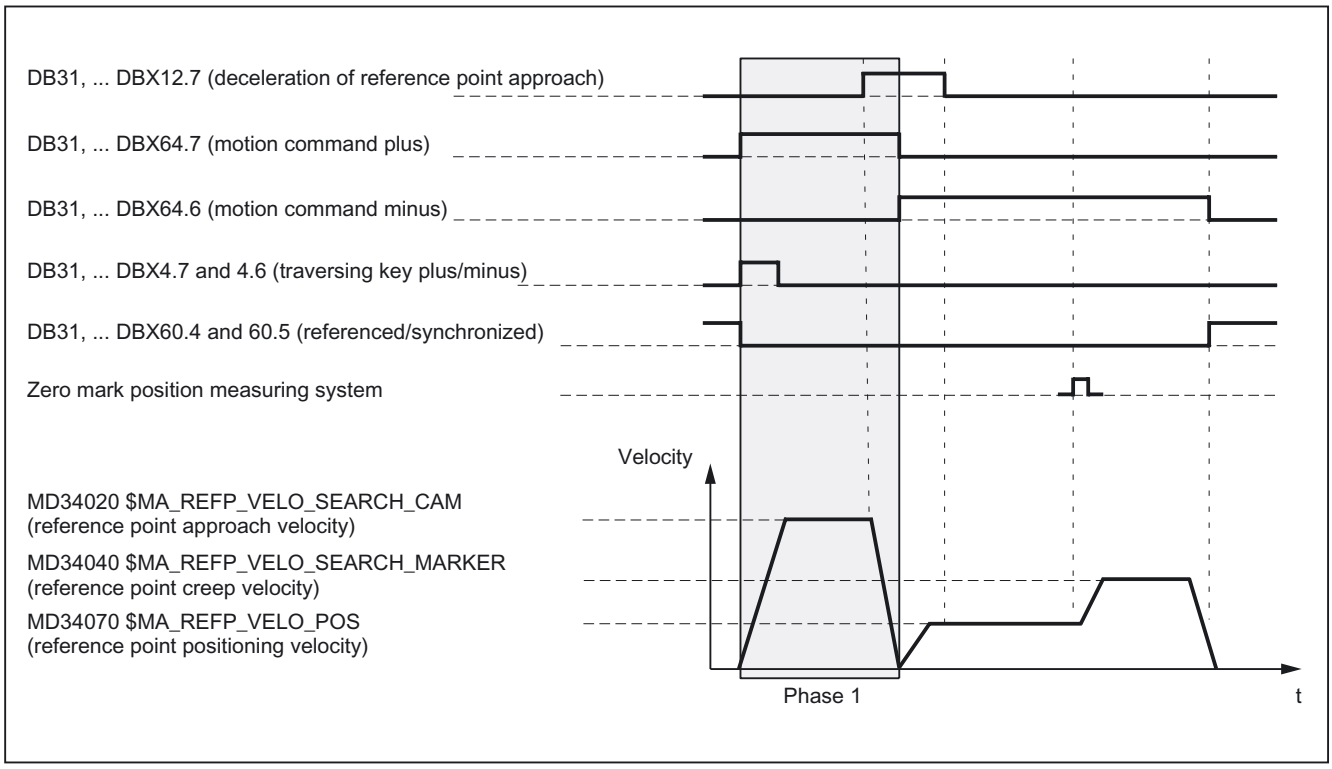


Fig. 2-2 Phase 1: Traversing to the reference cam

Phase 1: Start

For information on starting reference point approach, refer to "Axis-specific referencing" and "Channel-specific referencing."

Phase 1: Sequence

In phase 1, depending on the position of the machine axis with reference to the homing 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

When reference point approach is started, the machine axis accelerates in the assigned direction:
MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction) to the assigned reference point approach velocity before the reference cam:
MD34020 $MA_REFP_VELO_SEARCH_CAM.

The PLC user program communicates to the NC that the reference cam has been reached via the following interface signal:

DB31, ... DBX12.7 (reference point approach deceleration)

As a result the NC decelerates to zero speed. In so doing, the following distance S_{min} is traversed, at a minimum:

S_{min} =	(MD34040 $MA_ REFP_VELO_SEARCH_MARKER) ^{2} / 2 * (MD32300 $MA_MAX_AX_ACCEL)

This minimum distance must be covered to ensure that the machine axis exits the reference cam in phase 2 at the assigned reference point creep velocity.

Phase 1 is now complete. Reference point approach is continued with Phase 2.

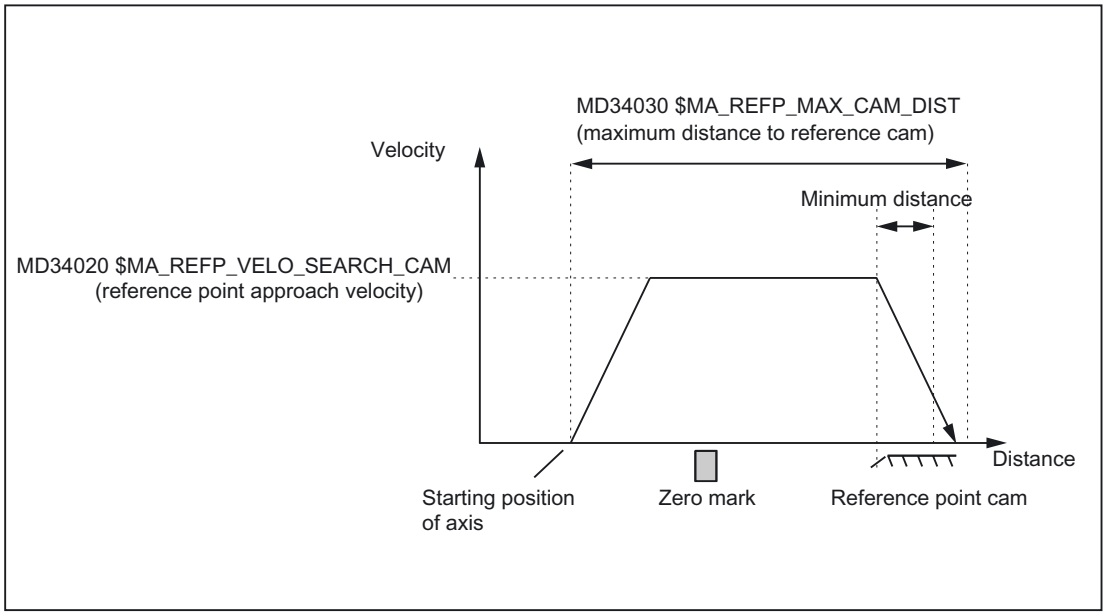


Fig. 2-3 Minimum distance for deceleration

Case 2: The machine axis is positioned on the reference cam

The machine axis remains at its starting position.
Phase 1 now complete. Reference point approach is continued with Phase 2.

Case 3: The machine axis has no reference cam

Machine axes without reference point 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 point cam:

MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) **= 0**

Phase 1 is now complete. Reference point approach is continued with Phase 2.

Phase 1: Features

- Feed override active.
- Feed stop (channelspecific and axisspecific) is active.
- NC-STOP and NC-START are active.
- If in Phase 1 the machine axis does not reach the reference cam:
  DB31, ... DBX12.7 (deceleration of reference point approach) = 1
  within the assigned distance:
  MD34030 $MA_REFP_MAX_CAM_DIST (maximum distance to the reference cam),
  the machine axis is stopped.

2.4.3 Phase 2: Synchronization with the zero mark

Phase 2: Graphic representation

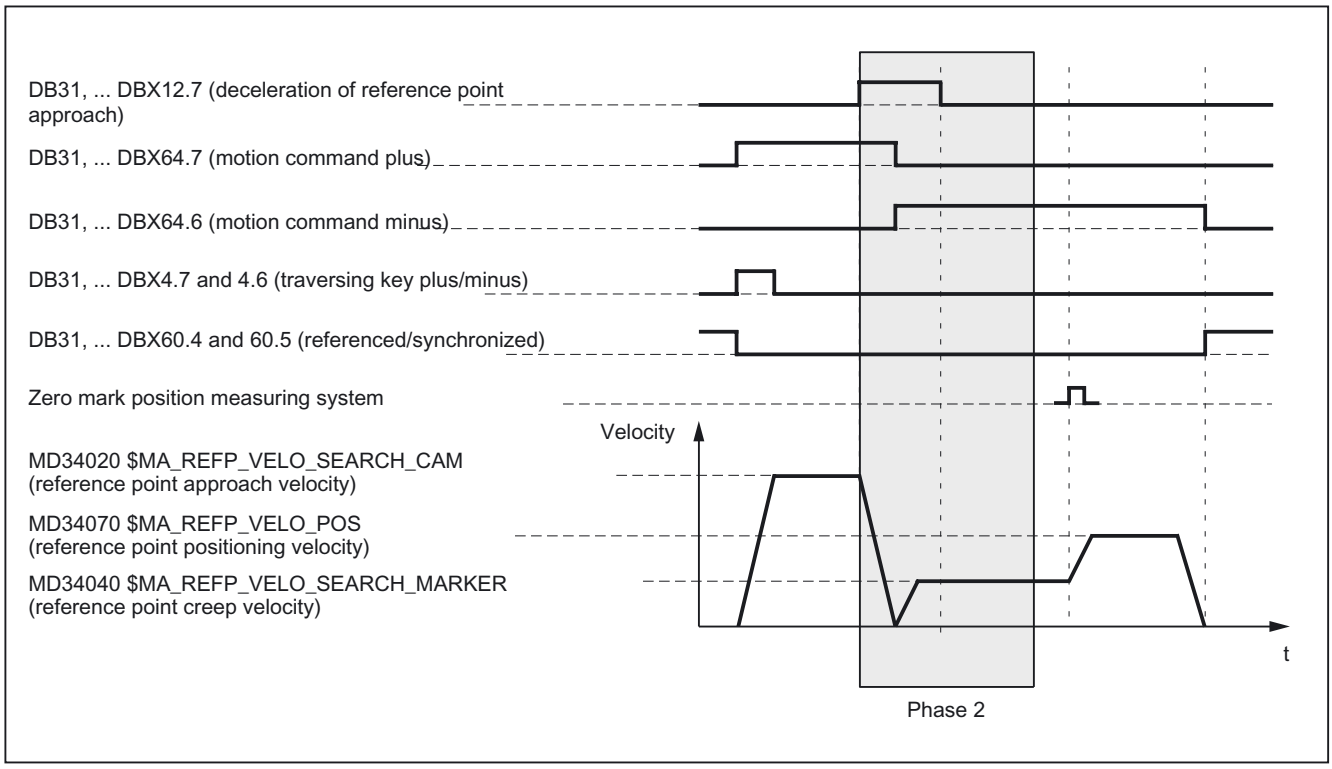


Fig. 2-4 Phase 2: Synchronization with the zero mark

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 zero mark is searched for in the parameterized direction and approached:

MD34050 $MA_REFP_SEARCH_MARKER_REVERSE (direction reversal on reference cam)

Phase 2: Sequence

In Phase 2, there are two cases, depending on the assigned synchronization method:
MD34050 $MA_REFP_SEARCH_MARKER_REVERSE (direction reversal on reference cam):

1. Synchronization with falling reference cam edge:

   MD34050 $MA_REFP_SEARCH_MARKER_REVERSE = 0

2. Synchronization with rising reference cam edge:

   MD34050 $MA_REFP_SEARCH_MARKER_REVERSE = 1

Note

If, while approaching the reference cam, the actual velocity of the machine axis does not reach the set velocity of Phase 2 within the assigned tolerance limits:
MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)
Phase 1 is started over again.

This will be the case, for example, if the machine axes are positioned at the reference cam when reference point approach starts.

Case 1: Synchronization with falling reference cam signal edge

During synchronization with falling reference cam edge, the machine axis accelerates to the assigned reference point creep velocity:
MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)
along the assigned reference point approach direction (traversing direction of Phase 1):
MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction).

After the reference cam is exited:
DB31, ... DBX12.7 (deceleration of reference point approach) = 0,
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.

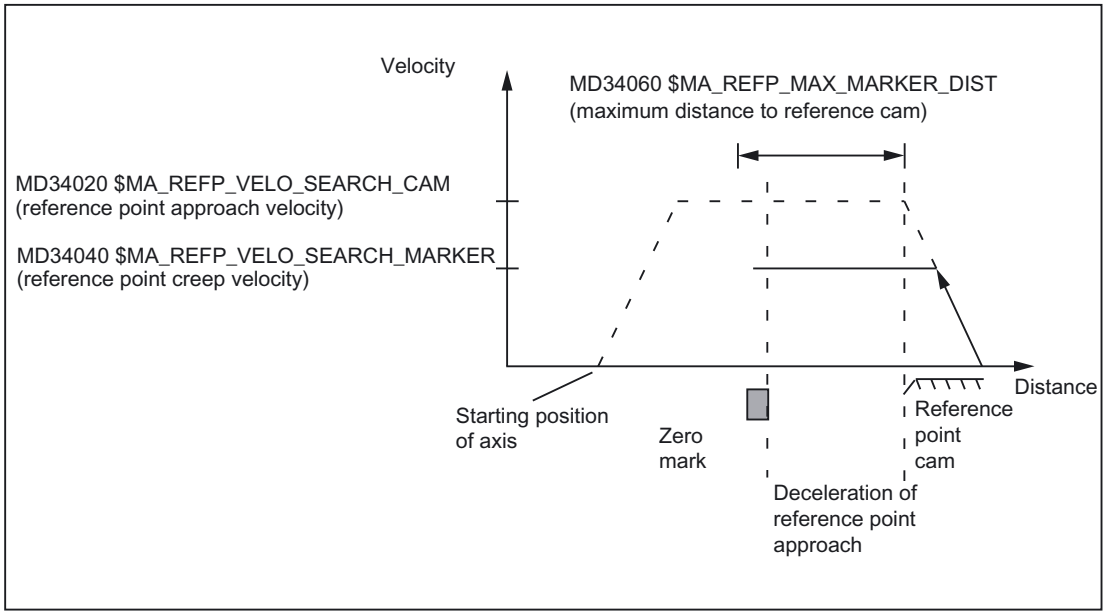


Fig. 2-5 Synchronization with falling reference cam signal edge

Case 2: Synchronization with rising reference cam edge

During synchronization with rising reference cam edge, the machine axis accelerates to the assigned reference point approach velocity:
MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)
along the assigned reference point approach direction (traversing direction of Phase 1):
MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction).

After the reference cam is exited:
DB31, ... DBX12.7 (deceleration of reference point approach) = 0
the machine axis is decelerated to a standstill.

Then, the machine axis moves back on the reference cam at the assigned reference point creep velocity:
MD34040 $MA_REFP_VELO_SEARCH_MARKER
(reference point creep velocity).

After the reference cam is reached:
DB31, ... DBX12.7 (deceleration of reference point approach) = 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.

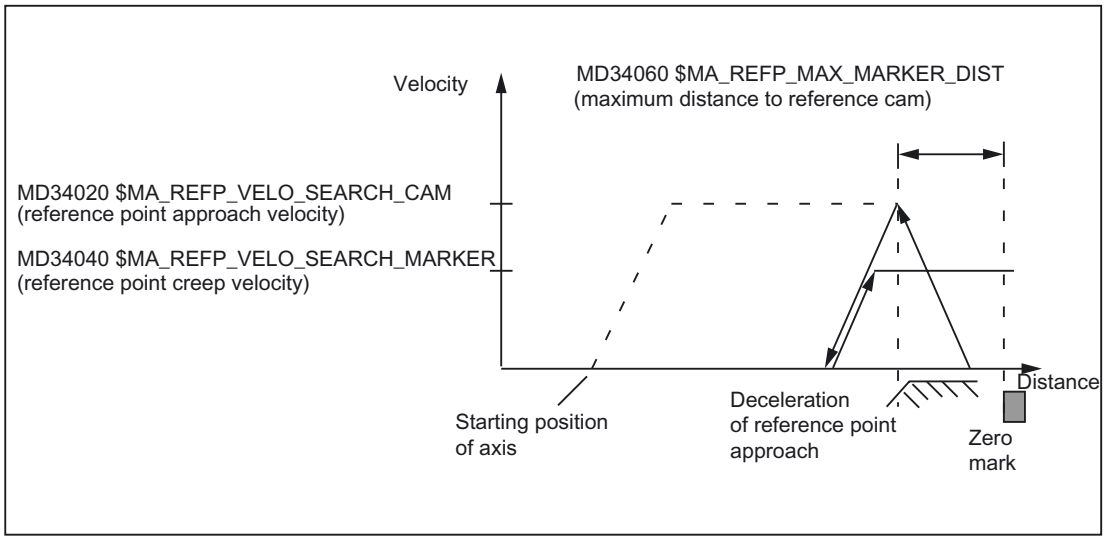


Fig. 2-6 Synchronization with rising reference cam signal edge

Electronic reference cam offset

Electronic reference cam offset is used for compensating reference cam length increases caused by temperature:

MD34092 $MA_ REFP_CAM_SHIFT (electronic reference cam offset for incremental measuring systems with equidistant zero marks)

After a rising or falling reference cam edge is detected the axis is synchronized for the next encoder zero mark only after the cyclically calculated offset path has been covered. Because the offset path s_{shift} is calculated by the NC in IPO cycles, the following minimum and maximum offset paths s_{shift_min} and s_{shift_max} will result:

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 reference cam offset acts in the direction of zero mark search.

Prerequisite

The reference cam offset is only active for machine axes for which a reference cam has been parameterized:

MD34000 $MA_REFP_CAM_IS_ACTIVE = 1

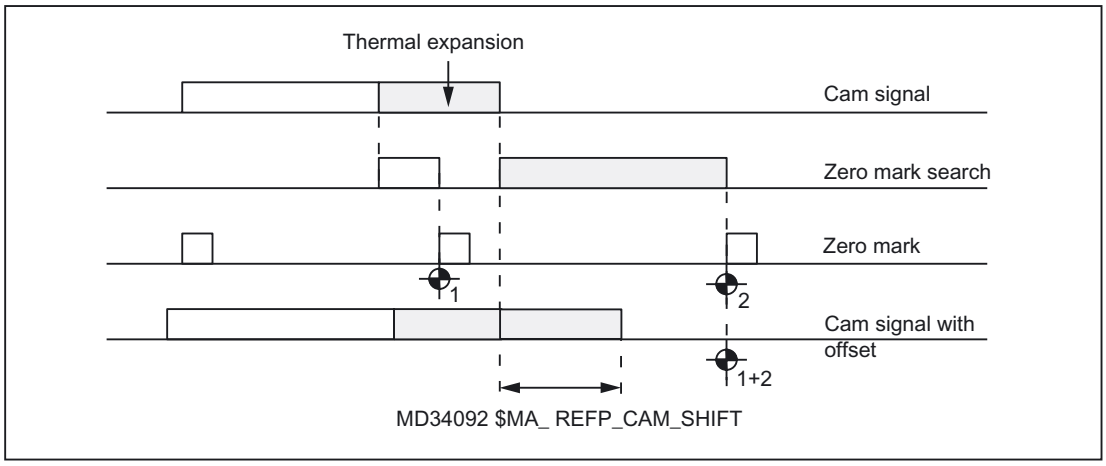


Fig. 2-7 Electronic reference cam offset

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 will influence the dynamic response between arrival at the reference cam on the machine and detection of the reference cam signal transmitted 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 VDI interface
- Interpolation cycle
- Position control cycle

Setting notes

- 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 offset

  Information needed for parameterizing the electronic reference cam offset 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 nondeterministic, 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 control 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 2 zero marks.

Phase 2: Features

- Feedrate override is **not** active.

  Internal motion with feedrate override = 100%.

  If a feedrate override of 0% is specified, motion is aborted.

- The feed stop (channel-specific and axis-specific) is active.
- NC-STOP and NC-START are **not** active.
- If in Phase 2 the machine axis does not reach the reference mark (encoder zero mark) within the assigned distance:
  MD34060 $MA_REFP_MAX_ MARKER_DIST (maximum distance to reference mark)
  the machine axis is stopped.

2.4.4 Phase 3: Traversing to the reference point

Phase 3: Graphic representation

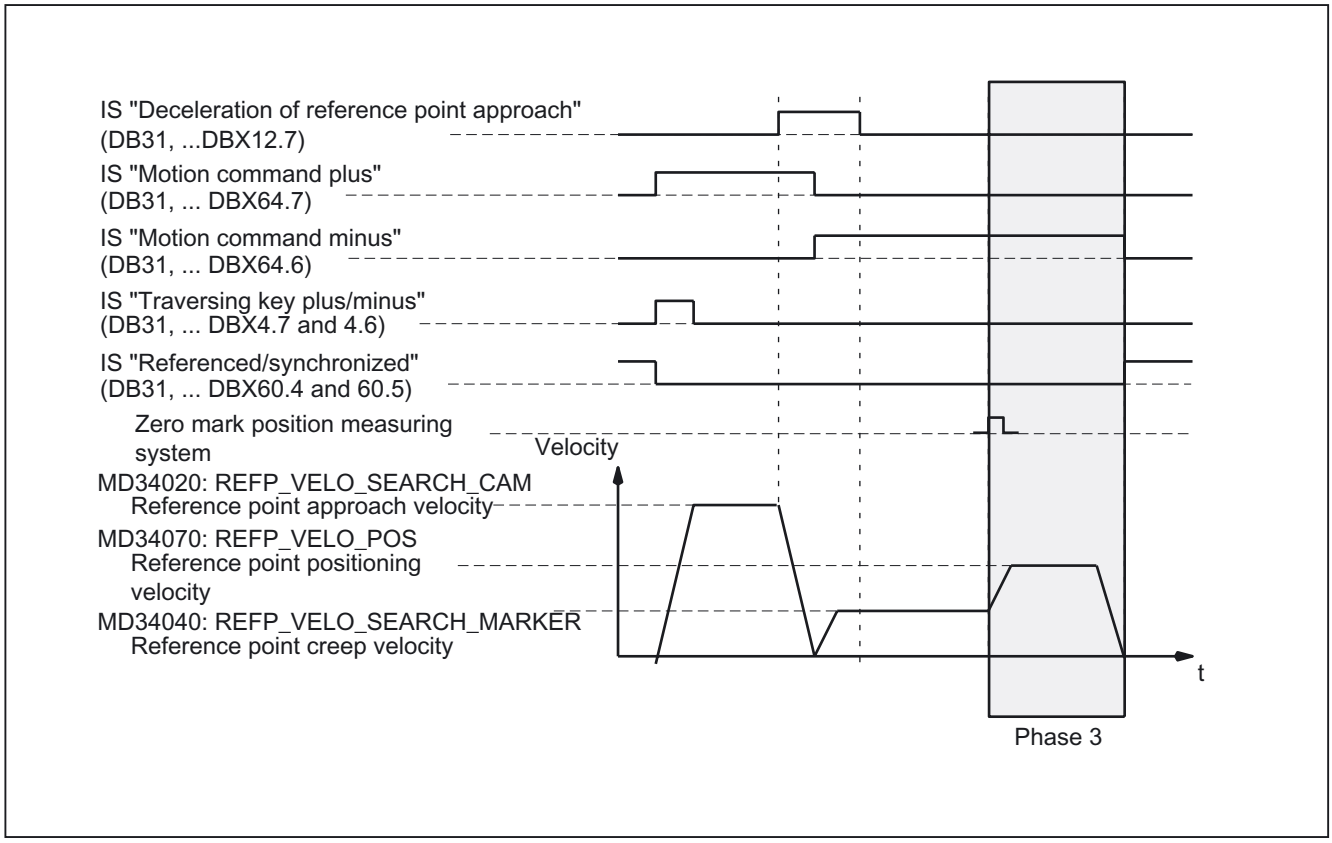


Fig. 2-8 Phase 3: Traversing to the reference point

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)

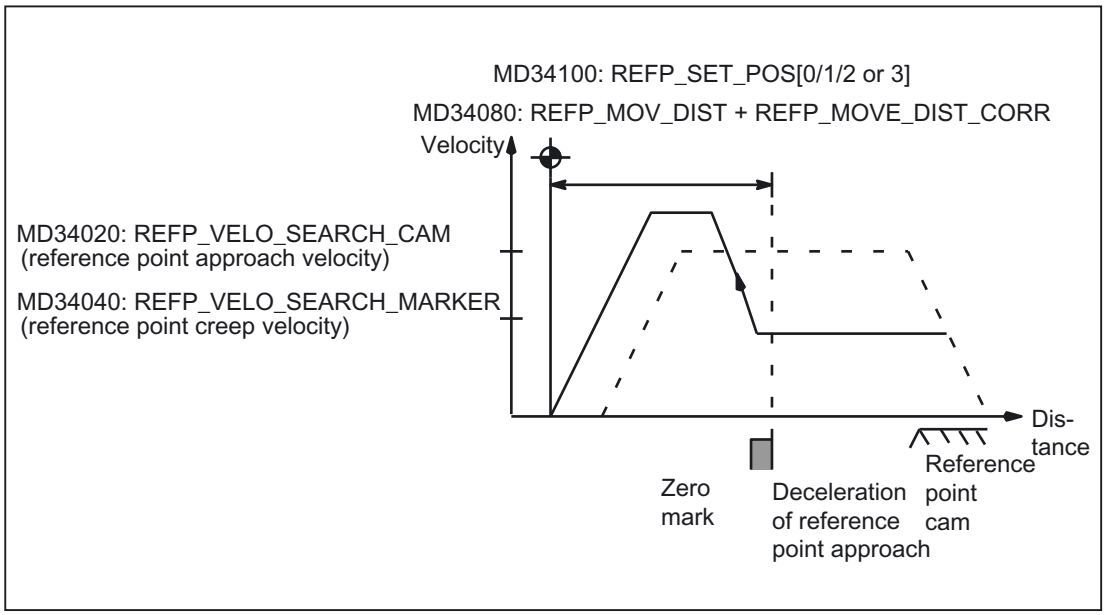


Fig. 2-9 Reference point position

When the reference point is reached the machine axis is stopped and the actual value system of the machine axis is synchronized with the reference point value specified by the PLC user program.

Reference position of machine axis = MD34100$MA_ REFP_SET_POS[**n**] (reference point value [n])

The reference point value is specified by the PLC user program by means of:
DB31, ... DBX2.4/.5/.6/.7 (reference point value 1/2/3/4)

In so doing, the NC assumes the reference point value that was selected by the PLC user program when the reference cam was reached in Phase 1:
DB31, ... DBX12.7 (deceleration of reference point approach) = 1

The machine axis is now referenced. To identify this, the NC sets the appropriate interface signal based on which measuring system is currently active:
DB31, ... DBB60.4 / .5 (referenced/synchronized 1/2) = 1

Features of phase 3

- Feed override active
- The feed stop (channel-specific and axis-specific) is active
- NC-STOP and NC-START are active

Special feature of phase 3

If the assigned distance between the encoder zero mark and the reference point, i.e., the sum of the reference point distance and the reference point offset:
MD34080 $MA_REFP_MOVE_DIST (reference point distance)
MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
is less than the deceleration distance that is needed to stop the axis on the reference point based on the reference point positioning velocity:
MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)
the machine axis first stops "behind" the reference point and then moves back to the reference point.

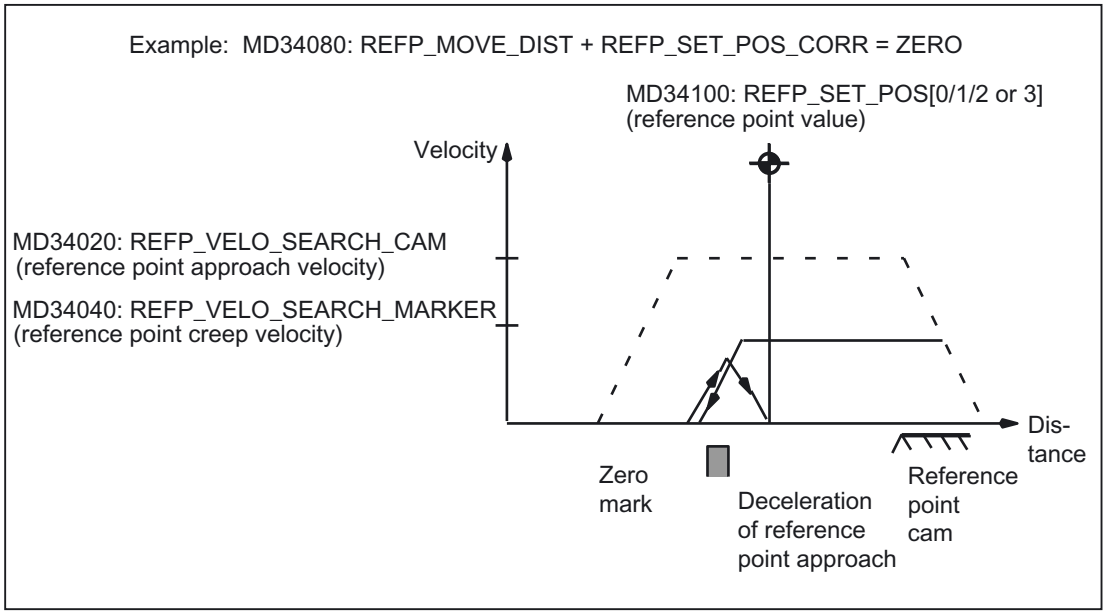


Fig. 2-10 Reference point distance plus reference point offset smaller than braking distance

2.4.5 Buffered actual value

Automatic referencing

Buffering the actual value enables "Automatic referencing" to be performed. With automatic referencing, machine axes with incremental measuring systems do not have to repeat the reference point approach in order to reference the measuring system following situations when the status of these axes is "Non-referenced". For example, this is the case after:

- POWER OFF/ON
- "Parking" selected

  DB31, ... DBB1.5/1.6 (position measuring system 1/2) = 0

  DB31, ... DBB2.1 (servo enable) = 0

If the function is active, referencing is performed without axis motion by synchronizing the actual value system of the machine axis to the last valid actual value. The actual value is buffered in a retentive memory area of the control even after POWER OFF.



Warning

The machine axis must not be moved mechanically while the measuring system (encoder) of the machine axis is switched off (POWER OFF, axis "parked" etc.). This must be supported by the machine manufacturer with such measures as holding brakes, etc., and ensured by the user. Otherwise the actual value system of the machine axis will no longer be synchronized resulting in danger to personnel and machine.

Precondition for automatic referencing

At the time that automatic referencing occurs, e.g., after a POWER ON, when "parking" of machine axis is deactivated again, etc., the encoder status of the active measuring system of the machine axis must have the value 2:

MD34210 $MA_ENC_REFP_STATE = **2**

That is, the encoder is referenced, an exact stop is reached, and automatic referencing is active the next time the encoder is activated.

The following conditions must be fulfilled:

- "Automatic referencing" is assigned for the measuring system (encoder):

  MD34210 $MA_ENC_REFP_STATE = 1

Encoder status = 1:	Automatic referencing enabled, but encoder not yet referenced or exact stop not reached.

- The measuring system has already been successfully referenced with reference point approach.

  This is identified by the NC by changing the encoder status from 1 to 2.

Encoder status = 2:	Encoder is referenced, an exact stop is reached, and automatic referencing is active the next time the encoder is activated.

- The machine axis was disabled in "Exact stop fine" status.

Note

As part of the actual value buffering, the axis-specific interface signal:
DB31, ... DBB60.7 (position reached with exact stop fine)
is evaluated.

The actual values of machine axes that do not use the signal cannot be buffered.

Functional sequence

Two different cases apply for automatic referencing, which depend on the encoder status:

- **Case 1: Encoder status = 2**

  Automatic referencing in parameterized, the measuring system has been referenced and the machine axis was switched off in status "Exact stop fine".

  As a result:

  – The actual value system of the machine axis is synchronized with the buffered actual value.

  – The status of the machine axis is set to "Referenced".

    DB31, ... DBB60.4/60.5 (referenced/synchronized 1/2) = **1**

- **Case 2: Encoder status = 1**

  Automatic referencing in parameterized but the measuring system has either not yet been referenced or the machine axis was not switched off in status "Exact stop fine".

  As a result:

  – The actual value system of the machine axis is synchronized with zero.

  – The status of the machine axis is set to "Not referenced":

    DB31, ... DBB60.4/60.5 (referenced/synchronized 1/2) = **0**

2.5 Referencing with distance-coded reference marks

2.5.1 General overview

Distance-coded 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 in different ways. 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).

2.5.2 Basic parameter assignment

Linear measuring systems.

The following data must be set to parameterize linear measuring systems:

- The absolute offset between machine zero and the position of the 1st 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)

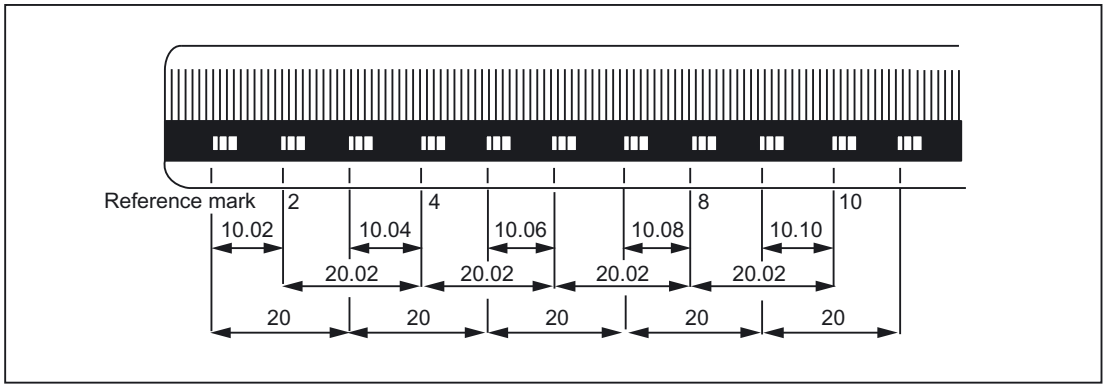


Fig. 2-11 DIADUR graduated glass scale with distance-coded reference marks (dimensions in mm for 20 mm scale division)

Rotary measuring system

For rotary measuring systems, the same applies as for linear measuring systems (see above).

Determining the absolute offset

In order to determine the absolute offset between machine zero and the position of the 1st reference mark of a machine axis, the following procedure is recommended:

1. The value zero must be entered 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 HMI user interface.
4. Measure the current position of the machine axis with reference to machine zero.
5. Calculation of absolute offset.

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:

Equidirectional orientation:	Absolute offset = Measured position + indicated actual position
Inverse orientation:	Absolute offset = Measured position - indicated actual position



Warning

After the absolute offset is determined and an entry is made in machine data element:
MD34090 $MA_REFP _MOVE_DIST_CORR
the reference point approach for the machine axis is repeated.

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

2.5.3 Chronological sequence

Chronological sequence

Referencing with distance-coded reference marks can be divided into two phases:

- Phase 1: Travel across the reference marks with synchronization
- Phase 2: Traveling to a fixed destination point

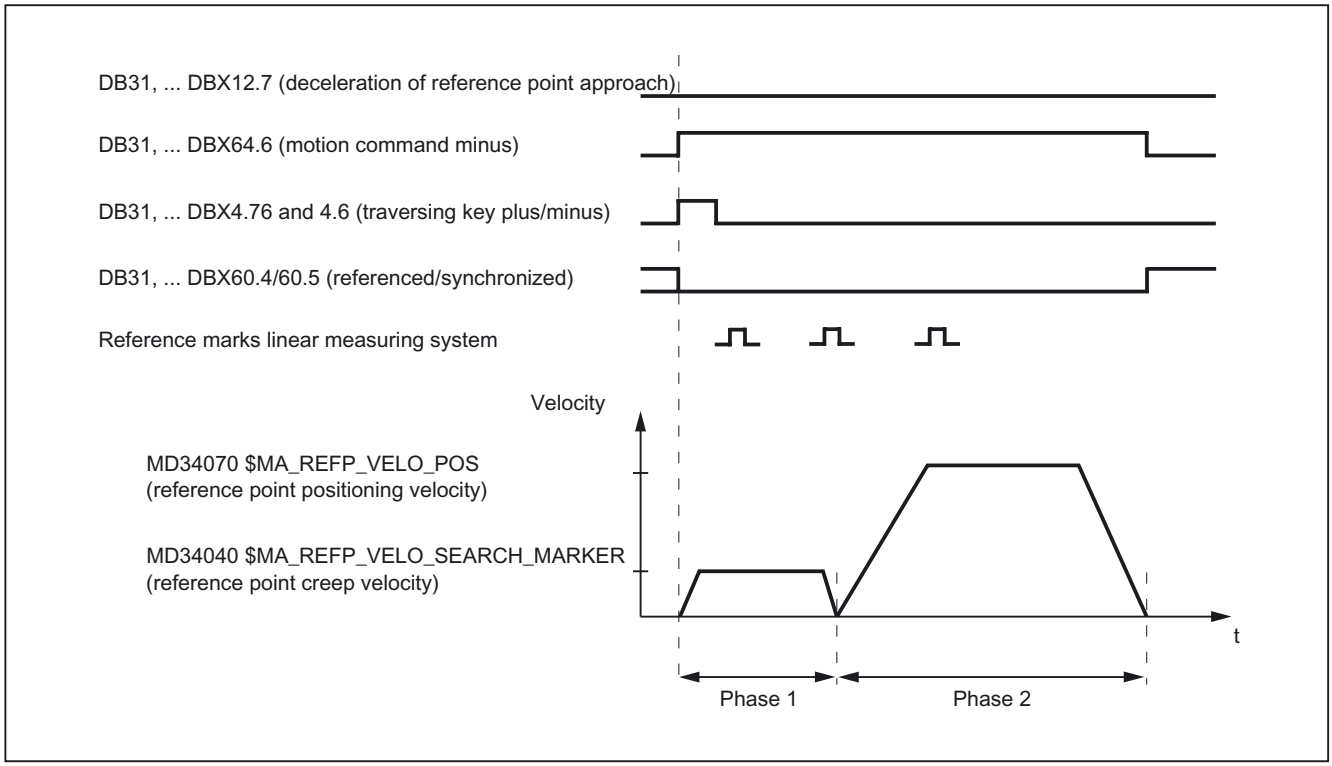


Fig. 2-12 Distancecoded reference marks

2.5.4 Phase 1: Reference mark search

Phase 1: Start

For information on starting reference point approach, refer to "Axis-specific referencing" and "Channel-specific referencing."

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 touching a reference cam

Once reference point approach is started, the machine axis accelerates to the assigned reference point creep velocity:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Once the parameterized number of reference marks 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 at the start of reference point approach the machine axis is positioned on the reference cam, the machine axis accelerates to the assigned reference point creep velocity:
MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)
along the assigned reference point approach direction:
MD34010 $MA_CAM_DIR_IS_MINUS (reference point approach in minus direction)
i.e., away from reference cam. 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 parameterized number of reference marks 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 reference point approach is started, the machine axis accelerates to the assigned reference point creep velocity:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Before the machine axis travels over the parameterized number of reference marks, it touches the reference cam. It is then reversed and reference mark search is restarted in the opposite direction.

Once the parameterized number of reference marks 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

During reference point approach, if the NC determines a distance greater than twice the assigned reference mark distance:
MD34300 $MA_ENC_REFP_MARKER_DIST (reference mark distance)
for two consecutive reference marks, an error occurs.

The machine axis is then moved in the opposite direction at half the assigned reference point creep velocity:
MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)
and the reference mark search is restarted.

If another incorrect reference mark is detected, the machine axis is stopped, reference point approach is aborted and the following alarm is displayed:

Alarm 20003 "Error in measuring system"

Abort criterion

If the assigned number of reference marks is not detected within the assigned distance:
MD34060 $MA_REFP_MAX_ MARKER_DIST (maximum distance to reference mark)
the machine axis is stopped and reference point approach is aborted.

Features of phase 1

- After phase 1 is successfully completed, the actual value system of the machine axis is synchronized.

2.5.5 Phase 2: Travel to fixed stop

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 = *Mode*

Mode = 0:	Travel to target position
Mode = 1:	No travel to target position

Mode = 0: Travel to target position (normal case)

The machine axis accelerates to the assigned reference point positioning velocity:
MD34070 $MA_REFP_VELO_POS
and moves to the assigned end point (reference point):
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, ... DBB60.4/60.5 (referenced/synchronized 1/2) = 1

Mode = 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, ... DBB60.4/60.5 (referenced/synchronized 1/2) = 1

Features of phase 2

Irrespective of whether a reference point cam is parameterized for the machine axis, phase 2 exhibits different properties:

- **Machine axis without reference point cam**

  MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference point cam) = **0**

  Properties:

  – Feed override active.

  – Feed stop (channel-specific and axis-specific) is active.

  – NC STOP and NC START are active.

- **Machine axis with reference point cam**

  MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) = **1**

  Properties:

  – Feedrate correction 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.

  – After the reference cam is exited, if the assigned number of reference marks is not detected within the assigned distance:
  MD34060 $MA_REFP_MAX_ MARKER_DIST (maximum distance to reference mark)
  the machine axis is stopped.

Special features of rotary measuring systems

On rotary distancecoded 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 (extent of modulo range)

MD30340 $MA_MODULO_RANGE_START (starting position of 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

2.6 Referencing with absolute value encoders

2.6.1 Information about calibration

Referencing with absolute value encoders

The advantage of machine axes with absolute encoders is that the reference point approach required with incremental measuring systems after, for example:
Power on of controller (POWER ON)
or
Deselection of "Parking" of machine axis
can be omitted after a one-time adjustment operation, and the actual value system of the machine axis is 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 indicated in the following axis-specific machine data element of the machine axis on which the absolute encoder is connected.

MD34210 $MA_ENC_REFP_STATE (status of absolute encoder)

Status	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 value encoder
- Removal/installation of the motor with the absolute value encoder
- SRAM data loss of the NC
- Battery failure
- Function: PRESET



Warning

The NC automatically resets the status of the absolute encoder to 0 = "Encoder not adjusted" only when there is a gear change:

MD34210 $MA_ENC_REFP_STATE = 0

In all other cases, it is the sole responsibility of the user to indicate the unadjusted state of the absolute encoder to the NC by manually resetting the status to 0 = "Encoder not adjusted" and to repeat the adjustment.



Warning

Data backup

When machine data of Machine A are backed up, the encoder status:
MD34210 $MA_ENC_REFP_STATE (status of absolute encoder)
of the machine axes is also backed up.

When this data block is loaded into machine B of the same type, e.g., as part of a series start-up or after servicing, the NC automatically regards the machine axes in question as adjusted/referenced. It is therefore the special responsibility of the machine manufacturer/user to perform a readjustment.

See the description of machine data element:
MD30250 $MA_ACT_POS_ABS (absolute encoder position at switch-off time)

2.6.2 Calibration by entering a reference point offset

Function

If adjustment is performed by entering a reference point offset, the difference between the actual position displayed on the HMI user interface and the real actual position in the machine is determined and passed on to the NC as the reference point offset.

Sequence

Proceed as follows to perform an adjustment by entering the reference point offset:

1. Determine the position of the machine axis relative to the machine zero point, for example, by:
   - Measuring the position (e.g., laser interferometer)

     or
   - Moving the machine axis to a known position (e.g., fixed stop)
2. Establish the actual position of the machine axis displayed by the NC on the HMI user interface.
3. Enter the reference point offset as the difference between the actual positions determined under Items 1 and 2:

   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
4. Identify the absolute encoder as calibrated:

   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. Check the machine axis position displayed on the HMI user interface.

Note

Backlash compensation

If backlash compensation has been parameterized for a measuring system with absolute encoders, attention must be paid that the machine axis is moved to the machine position at which the adjustment is to take place in such a way that no backlash occurs.

Activate reference point offset permanently

The entered reference point offset:
MD34090 $MA_REFP_MOVE_DIST_CORR
is activated permanently following a 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).

2.6.3 Adjustment by entering a reference point value

Function

In the case of adjustment by entering a reference point value, the absolute position of the machine axis relative to machine zero is determined, for example,
by measuring the position (e.g., with a laser interferometer)
or
by moving the machine axis to a known position (e.g., fixed stop)
and this absolute position is made known to the NC as a reference point value.

The NC then calculates the reference point offset from the difference between the encoder absolute value and the reference point value.

Sequence

Proceed as follows when for adjustment by entering the reference point value:

1. Set the referencing mode to "Assume the reference point value":

   MD34200 $MA_ENC_REFP_MODE = 0

2. Move the machine axis in JOG mode to the position to be measured (e.g., with a laser inferometer) or to a known position (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. Enable encoder adjustment:

   MD34210 $MA_ENC_REFP_STATE = 1

5. Initiate NC-RESET to transfer 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 then calculates the reference point offset from the reference point value signaled by the absolute encoder and the entered reference point value, enters this reference point offset in machine data element:
   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
   and sets the status of the absolute encoder 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.

   To identify this, 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 $MA_REFP_MOVE_DIST_CORR
is activated permanently following a 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 an 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).

2.6.4 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. Calculate and write the reference point offset to machine data:

   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. Initiate NC-RESET to transfer the entered machine data values.
4. Start part program.
5. Disable part program start again for non-referenced machine axes:

   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 $MA_REFP_MOVE_DIST_CORR
is activated permanently following a 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 an 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).

2.6.5 Calibration 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.

Sequence

Proceed as follows for adjustment with BERO:

1. Set referencing mode to "Referencing with BERO":

   MD34200 $MA_ENC_REFP_MODE = 2

2. Assign reference point value:

   MD34100 $MA_REFP_SET_POS = reference point value

3. Start reference point approach.

   Reference point approach can be started manually in JOG-REF mode or in AUTOMATIC or MDA mode from a part program (G74).

   After a successful reference point approach, the absolute encoder is calibrated, the actual value system of the machine axis is synchronized, and the machine axis is referenced.

   To identify this, the NC sets an interface signal for the measuring system that is currently active:

   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 assigned to "Referencing with absolute encoder".

MD34200 $MA_ENC_REFP_MODE = 0

Signal propagation delay compensation

The signal propagation time can cause corruption of the absolute position detected by the NC. The signal propagation time can be compensated for in any direction:

MD31122 $MA_BERO_DELAY_TIME_PLUS (BERO delay time plus)

MD31123 $MA_BERO_DELAY_TIME_MINUS (BERO delay time minus)

Note

A requirement for correct compensation of the signal propagation delay are SIMODRIVE 611-D drives. The assignments made in the factory for the compensation times do not usually have to be changed.

Creep velocity

If the BERO is to be approached at the assigned reference point creep velocity:
MD34040 $MA_REFP_VELO_SEARCH_MARKER
reference mode "BERO with configured approach velocity for spindles" must be assigned:
MD34200 $MA_ENC_REFP_MODE = 7

2.6.6 Reference point approach with absolute encoders

MD34330

If reference point approach is initiated for a machine axis with adjusted absolute encoder as the active measuring system (manually in JOG-REF mode or automatically via part program instruction (G74)), the movement of the machine axis is dependent on the assigned traversing enable:

MD34330 $MA_REFP_STOP_AT_ABS_MARKER = <Value>

Value	Meaning
FALSE	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.
TRUE	Traversing is **not** enabled. When reference point approach is initiated, the machine axis does not move and reference point approach is terminated immediately.

2.6.7 Automatic encoder replacement detection

Functionality

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 during power-up machine axis specifically in:

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 = FALSE

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

Because not all drives connected via PROFIBUS-DP can supply the encoder serial number in a timely manner when the controller is powered up and some of these drives cannot supply the encoder serial number at all, unnecessary readjustments can be avoided by permanently setting the range of the encoder serial numbers for PROFIBUS drives to FALSE internally in the NC:

MD34232 $MA_EVERY_ENC_SERIAL_NUMBER = FALSE

Manual parameterization to TRUE has no effect.

2.6.8 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, ... DBX1.5/1.6 (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. Axis travel movements are disabled during synchronization. Spindle travel movements are not.

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 operates at rotational speeds above the encoder limit frequency of the incremental track, a significantly lower limit frequency of the absolute track must be assigned 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 limit frequency corresponding to the limit speed:
MD36300 $MA_ENC_FREQ_LIMIT
is calculated as follows:

(4 * MD31020 $MA_ENC_RESOL) / (MD10050 $MN_SYSCLOCK_CYCLE_TIME * MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO)

Note

The closed-loop positioning speed:
MD35300 $MA_SPIND_POSCTRL_VELO
that is relevant for spindles must be set in accordance with the encoder limit frequency of the absolute encoder of the spindle:
MD36300 $MA_ENC_FREQ_LIMIT (encoder limit frequency).

2.6.9 Referencing variants that are not supported

The following referencing variants are not supported when used with absolute encoders:

- Referencing/calibrating with encoder zero mark
- Distancecoded reference marks
- BERO with two-edge evaluation

2.7 Referencing by means of actual value adjustment

2.7.1 Actual value adjustment to the referencing measurement system

Functionality

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

Actual value adjustment to the referencing measuring system is activated machine axis specifically by means of:

MD34102 $MA_REFP_SYNC_ENCS = 1

2.7.2 Actual value adjustment to the referenced measurement system

Functionality

If a machine axis has several measuring systems and one of them is referenced, the remaining measuring systems can be referenced by actual value adjustment to the measuring system already referenced.

Advantage

Referencing does not take as long because the path to be traversed only has to be sufficiently along to allow elimination of the backlash in both measuring systems.

If the distance to the traversing range limits is large enough, referencing can be started at any point along the traversing range of the machine axis in any direction because no zero mark, reference point cam, etc., is required.

Parameterization

The following machine data must be parameterized for actual value adjustment to a referenced measuring system:

- Homing mode: Actual value adjustment to a referenced measuring system

  MD34200 $MA_ENC_REFP_MODE[*measuring system*] = 6
- Traverse path for backlash recovery:

  MD34080 $MA_REFP_MOVE_DIST[*measuring system*] (reference point distance)

Note

Reference point distance greater than zero

To ensure reliable backlash recovery, the assigned value for the reference point distance must be greater than the maximum backlash of both measuring systems.

Reference point distance equal to zero

The reference point distance can also equal zero. In this case, actual value adjustment is performed without a traversing movement as soon as reference point approach starts. In this case, the active backlash should be less than half the increment of the measuring system whose actual value is taken.

2.7.3 Actual value adjustment for measuring systems with distance-coded reference marks

Functionality

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)

2.8 Referencing in follow-up mode

Functionality

Incremental measuring systems and measuring systems with distance-coded reference marks can be referenced even when the machine axis is in follow-up mode. This requires that reference point approach be properly parameterized according to the measuring system in use (see "Referencing with incremental measuring systems" and "Referencing with distance-coded reference marks").

When referencing in follow-up mode the machine axis is moved not by the NC but by means of an external travel movement 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 distancecoded reference marks
- Modulo rotary axes (absolute position within one revolution)

Zero mark selection when several zero marker 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 to the relevant drive module (SIMODRIVE 611D) via a BERO input in order to uniquely specify the reference point. The position of the reference point is then derived from the combination of BERO signal and encoder zero mark.

Zero mark evaluation with BERO must be parameterized as the referencing mode:

MD34200 $MA_ENC_REFP_MODE = 5

Negative edge evaluation

In the case of a referencing operation with a negative edge evaluation of the BERO signal:
MD34120 $MA_REFP_BERO_LOW_ACTIVE = FALSE
synchronization is with the next encoder zero mark encountered after the BERO is exited.

Positive edge evaluation

In the case of a referencing operation with a positive edge evaluation of the BERO signal:
MD34120 $MA_REFP_BERO_LOW_ACTIVE = TRUE
synchronization is with the next encoder zero mark encountered after the BERO is approached.

Mechanically, the BERO must be sized in such a way that the positive BERO signal covers the entire width of the encoder zero mark.

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 (follow-up mode active) == TRUE
at the start of reference point approach, the measuring system is referenced in follow-up mode.
(If the machine axis is not in follow-up mode at the start of reference point approach, "normal" NC-controlled reference point approach is performed.)

Referencing in follow-up mode can be started in the following modes:

- JOG-REF: Traversing keys
- AUTOMATIC: Part program command G74

Sequence of referencing operation (JOG-REF mode)

1. Activate follow-up mode of machine axis:

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

   DB31, ... DBX2.1 (servo enable) = 0
2. Take into account activation of follow-up mode:

   DB31, ... DBX61.3 (follow-up mode active) = 1
3. Switch to JOG-REF mode
4. External movement 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
- NC-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 instruction 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 (servo enable) = 0
4. Take into account activation of follow-up mode:

   DB31, ... DBX61.3 (follow-up mode active) = 1
5. The referencing operation is started internally in the NC as soon as part program instruction G74 is processed.
6. External movement 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
- NC-RESET

Response when measuring systems are already referenced

A measuring system that you have already referenced can be referenced again.

2.9 Zero mark selection with BERO

Functionality

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 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 to the relevant drive module (SIMODRIVE 611D) via a BERO input in order to uniquely specify the reference point. The position of the reference point is then derived from the combination of BERO signal and encoder zero mark.

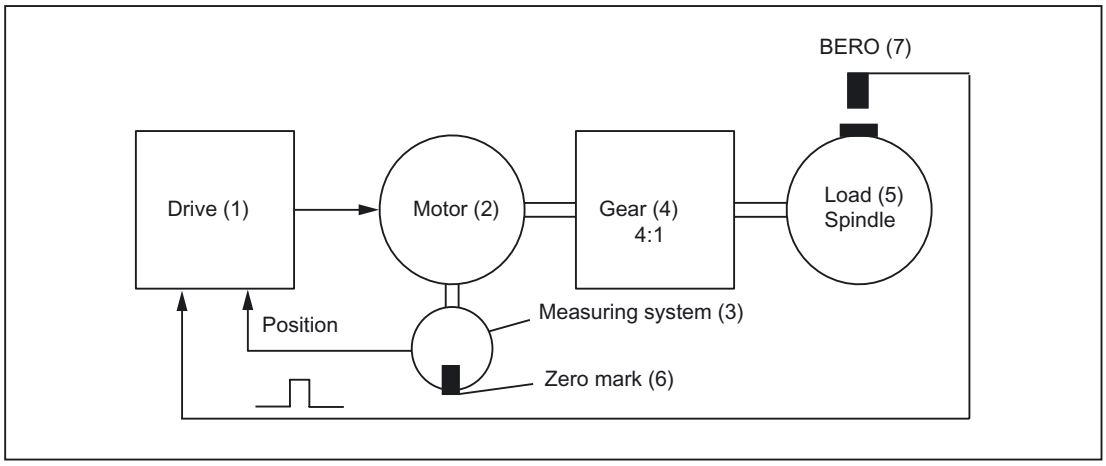


Fig. 2-13 Zero mark selection with BERO

Zero mark evaluation with BERO must be parameterized as the referencing mode:

MD34200 $MA_ENC_REFP_MODE = 5

Negative edge evaluation

In the case of a referencing operation with a negative edge evaluation of the BERO signal:
MD34120 $MA_REFP_BERO_LOW_ACTIVE = FALSE
synchronization is with the next encoder zero mark encountered after the BERO is exited.

Positive edge evaluation

In the case of a referencing operation with a positive edge evaluation of the BERO signal:
MD34120 $MA_REFP_BERO_LOW_ACTIVE = TRUE
synchronization is with the next encoder zero mark encountered after the BERO is approached.

If, mechanically, the BERO is sized in such a way that the positive BERO signal covers the entire width of the encoder zero mark, the encoder zero mark will be reliably detected in both traversing directions.

Supplementary Conditions

3

3.1 Large traverse range

Notes on uniqueness of encoder position

Linear absolute encoder

The absolute value of linear position encoders, e.g., Heidenhain LC181, is always unique for the scale lengths available.

Rotary absolute encoder

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 connected to 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.

Supplementary conditions

- Linear axes with a traversing range greater than 4,096 encoder revolutions, an EQN 1325 rotary absolute encoder, and an assigned 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 to 360 degrees).

    MD34220 $MA_ENC_ABS_TURNS_MODULO

    Otherwise, the machine axis may require a very long traverse path to reach absolute zero when the measuring system is activated.
- Machine axes with absolute encoders:
  - To ensure that the NC correctly determines the current actual position when the measuring system is reactivated, the machine manufacturer/user must ensure that the machine axis is moved less than half the assigned traversing range when the measuring system is deactivated (POWER OFF, "Parking" selected).

    MD34220 $MA_ENC_ABS_TURNS_MODULO

Examples

4

4.1 Examples

No examples are available.

Data Lists

5

5.1 Machine data

5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
11300	JOG_INC_MODE_LEVELTRIGGRD	INC/REF in jog/continuous mode

5.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20700	REFP_NC_START_LOCK	NC start disable without reference point

5.1.3 Axis/spindle-specific 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

Number	Identifier: $MA_	Description
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
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, distance-coded
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 follow-up mode
34110	REFP_CYCLE_NR	Axis sequence for channel-specific 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	Distance-coded 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

5.2 Signals

5.2.1 Signals to BAG

DB number	Byte.Bit	Name
11, ...	0.7	Mode group RESET
11, ...	1.2	Machine function: REF
11, ...	5.2	Active machine function: REF

5.2.2 Signals from BAG

DB number	Byte.Bit	Name
11, ...	5.2	Active machine function REF

5.2.3 Signals to channel

DB number	Byte.Bit	Name
21, ...	1.0	Activate referencing
21, ...	28.7	OEM channel signal (MMC → PLC) REF

5.2.4 Signals from channel

DB number	Byte.Bit	Name
21, ...	33.0	Referencing active
21, ...	35.7	Reset
21, ...	36.2	All axes referenced

5.2.5 Signals to axis

DB number	Byte.Bit	Name
31, ...	1.4	Follow-up mode (request)
31, ...	1.5/1.6	Position measuring system 1/position measuring system 2
31, ...	2.4 - 2.7	Reference point value 1 to 4
31, ...	4.6/4.7	Traversing keys minus/plus
31, ...	12.7	Decelertion of reference point approach

5.2.6 Signals from axis

DB number	Byte.Bit	Name
31, ...	60.4/60.5	Referenced, synchronized 1/Referenced, synchronized 2
31, ...	61.3	Follow-up mode active
31, ...	64.6/64.7	Traverse command minus/plus

Index

D

DB11, ...
- DBB0.1, 2-1, 2-3
- DBB0.2, 2-1, 2-3
- DBB1.2, 2-1, 2-3
- DBX0.7, 2-3, 2-4, 2-5

DB21, ...
- DBB1.0, 2-4
- DBB33.0, 2-4
- DBB36.2, 2-2
- DBB7.7, 2-5
- DBX36.2, 2-4
- DBX7.7, 2-3, 2-4

DB31, ...
- DBB1.5, 2-5, 2-19, 2-40
- DBB1.6, 2-5, 2-19, 2-40
- DBB2.1, 2-5, 2-19, 2-40
- DBB4.6, 2-1
- DBB4.7, 2-1
- DBB60.4, 2-2, 2-3, 2-4, 2-5, 2-20, 2-27, 2-33, 2-36, 2-41
- DBB60.5, 2-2, 2-3, 2-4, 2-5, 2-20, 2-27, 2-33, 2-36, 2-41
- DBB60.7, 2-20
- DBX1.4, 2-46, 2-47
- DBX12.7, 2-8, 2-9, 2-11, 2-12
- DBX2.1, 2-46, 2-47
- DBX60.4, 2-44, 2-46, 2-47
- DBX60.5, 2-44, 2-46, 2-47
- DBX61.3, 2-46, 2-47
- DBX61.4, 2-46

H

Homing
- With incremental measurement system, 2-6

M

Machine zero, 1-2

MD
- 34000, 2-9
- 34010, 2-8
- 34020, 2-8
- 34050, 2-10
- 34080, 2-42
- 34210, 3-2
- 34220, 3-2
- 35150, 2-11

MD10050, 2-41
MD10070, 2-41
MD11300, 2-1
MD20700, 2-35
MD30242, 2-44
MD30250, 2-30
MD30330, 2-28
MD30340, 2-28
MD30455, 2-29
MD30550, 2-3
MD31020, 2-41
MD31122, 2-37
MD31123, 2-37
MD32300, 2-8
MD34000, 2-13, 2-28
MD34010, 2-1, 2-11, 2-12, 2-25, 2-33
MD34020, 2-12
MD34030, 2-9
MD34040, 2-8, 2-11, 2-12, 2-13, 2-25, 2-26, 2-37
MD34050, 2-11
MD34060, 2-15, 2-26, 2-28
MD34070, 2-17, 2-18, 2-27
MD34080, 2-17, 2-18, 2-42
MD34090, 2-17, 2-18, 2-21, 2-23, 2-31, 2-32, 2-33, 2-34, 2-35, 2-36
MD34092, 2-13
MD34093, 2-15
MD34100, 2-17, 2-27, 2-33, 2-35, 2-36
MD34102, 2-42, 2-43
MD34104, 2-45
MD34110, 2-3
MD34120, 2-45, 2-48
MD34200, 2-23, 2-33, 2-36, 2-37, 2-43, 2-45, 2-48
MD34210, 2-19, 2-29, 2-30, 2-31, 2-33, 2-35, 2-38
MD34230, 2-38
MD34232, 2-39

MD34300, 2-26
MD34320, 2-21
MD34330, 2-27, 2-37
MD35300, 2-41
MD36300, 2-40

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Spindles (S1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief Description 1-1

2 Detailed Description 2-1

2.1 Spindle modes 2-1
2.1.1 General information 2-1
2.1.2 Spindle control mode 2-3
2.1.3 Spindle oscillation mode 2-7
2.1.4 Spindle positioning mode 2-7
2.1.5 Axis mode 2-21
2.1.6 Default setting of the spindle mode 2-25

2.2 Reference/Synchronize 2-26

2.3 Configurable gear adaptation 2-30
2.3.1 Gear steps for spindles and gear step change 2-30
2.3.2 Intermediate gear 2-41
2.3.3 If the gear step change is not acknowledged 2-42
2.3.4 Gear step change with oscillation mode 2-43
2.3.5 Gear step change at fixed position 2-50

2.4 Selectable spindles 2-55

2.5 Programming 2-60
2.5.1 Programming in the part program 2-60
2.5.2 Programming via synchronized actions 2-63
2.5.3 Programming of spindle controls via PLC with FC18 2-64
2.5.4 Special spindle motions via PLC interface 2-64
2.5.5 Gear step change with DryRun, program testing and SERUPRO 2-72
2.5.6 Programming from external (PLC, MMC) 2-75

2.6 Spindle monitoring 2-75
2.6.1 Speed ranges 2-75
2.6.2 Axis/spindle stationary (n < n min) 2-77
2.6.3 Spindle in setpoint range 2-77
2.6.4 Minimum/maximum gear step speed 2-79
2.6.5 Maximum encoder frequency 2-80
2.6.6 End point monitoring 2-81

3 Supplementary Conditions 3-1

3.1 Supplementary conditions 3-1

4 Examples 4-1

4.1 Example of automatic gear step selection (M40) 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Channelspecific machine data 5-1
5.1.3 Axis/spindle-specific machine data 5-2

5.2 Setting data 5-3
5.2.1 Channel-specific setting data 5-3
5.2.2 Axis/spindle-specific setting data 5-4

5.3 Signals 5-4
5.3.1 Signals to axis/spindle 5-4
5.3.2 Signals from axis/spindle 5-5

Index Index-1

Tables

Table 2-1 Change over to axis mode: 2-24

Table 2-2 Switch back to spindle mode 2-24

Brief Description

1

1.1 Brief Description

Spindle functions

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....)
- 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 with and without compensating chuck (G331, G332)
- 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

Detailed Description 2

2.1 Spindle modes

2.1.1 General information

Spindle modes

The spindle can have the following modes:

- Control mode
- Oscillation mode
- Positioning mode
- Synchronous mode synchronous spindle (S3)
- Rigid tapping

References:
/PA/ Programming Guide, Fundamentals
/PAZ/ Programming Guide, Cycles

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:

Switching between spindle modes

Interface signals or programming commands can be used to switch between the spindle modes and axis operation:

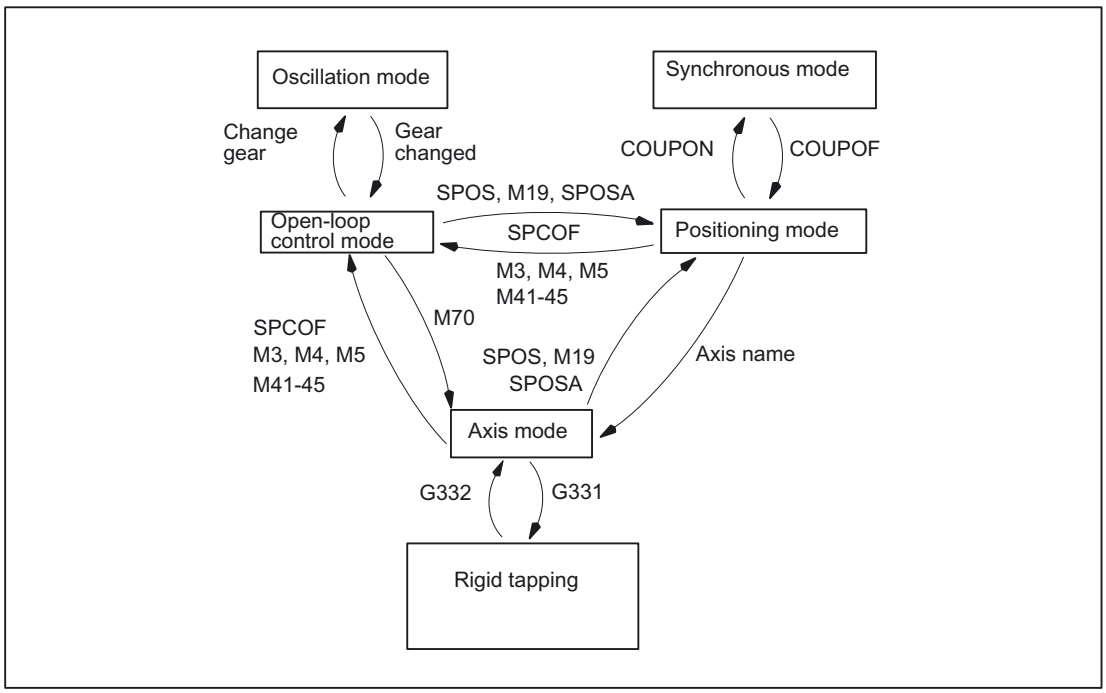


Fig. 2-1 Switching between spindle modes

- Open-loop control mode → Oscillation mode

  The spindle changes to oscillation mode if a new gear step 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 → Open-loop control mode

  When the new gear is engaged, IS:
  DB31, ... DBX84.6 (Oscillation mode)
  is reset and the spindle is switched
  to open-loop control mode with IS:
  DB31, ... DBX16.3 (Gear changed).
  The last programmed spindle speed (S value) is reactivated.

- Open-loop 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 → Open-loop control mode

  M3, M4 or M5 are used to change to open-loop 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.

- Open-loop control mode → Axis mode

  Switching from open-loop control mode to axis mode can be also achieved by programming M70. In this case, a rotating spindle is decelerated in the same way as for M5, position control is activated and the zero parameter set is selected.

- Axis mode → Open-loop control mode

  To terminate axis mode, M3, M4 or M5 can be used to change to open-loop 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.

2.1.2 Spindle control mode

When open-loop control mode?

The spindle is in control mode with the following functions:

Constant spindle speed	S.... M3/M4/M5 and G93, G94, G95, G97, G971
Constant cutting rate	G96/G961 S.... M3/M4/M5
Constant spindle speed	S.... M3/M4/M5 and G33, G34, G35
Constant spindle speed	S.... M3/M4/M5 and G63

Preconditions

- The spindle need not be synchronized.
- No spindle position actual-value sensor 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)

- A spindle position actual-value sensor is an essential requirement for M3/M4/M5 in connection with:
  - Revolutional feedrate (G95)
  - Constant cutting rate (G96, G961, G97, G971)
  - Thread cutting (G33, G34, G35)
  - Tapping (G331, G332)
  - Activate position control (SPCON, M70)

Position control ON/OFF

The spindle can be operated with or without position control:

SPCON	(Position control ON)
SPCOF	(Position control OFF)

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.
- NC/PLC IS:
  DB31, ... DBX84.7 (open-loop control mode)
  is set.
- NC/PLC IS:
  DB31, ... DBX61.5 (Position controller active)
  is reset if position control is not used.
- Acceleration in speed control mode is defined independent of the gear step in machine data:
  MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL
  and should, whenever possible, correspond to the physical properties.

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

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 activated in the part program with SPCON (spindle number).
- NC/PLC IS:
  DB31, ... DBX61.5 (position controller active)
  is set.
- Acceleration in position control mode is defined independent of the gear step in machine data:
  MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL.

Independent spindle reset

Machine data:
MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (independent spindle reset)
defines how the spindle behaves after a reset or program end (M2, M30):

- MD35040 $MA_SPIND_ACTIVE_AFTER_RESET is reset.

  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.

- MD35040 $MA_SPIND_ACTIVE_AFTER_RESET is set.

  On a 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 decelerated.

If prior to a reset or end of program the constant cutting speed (G96, G961) is active, the current spindle speed (in relation to 100% spindle override) is applied internally as the last programmed spindle speed.

The spindle can only be stopped with NC/PLC IS:
DB31, ... DBX2.2
(delete distancetogo/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)

Machine data:
MD36060 $MA_STANDSTILL_VELO_TOL
defines the speed at which the spindle is considered "stationary".

The speed set in the machine data should be dimensioned
so that NC/PLC IS:
DB31, ... DBX61.4 (axis/spindle stationary) is sure to be
pending on standstill.

If NC/PLC IS:
DB31, ... DBX61.4 (axis/spindle stationary)
is signaled and no position control is active for the spindle, zero is displayed for the actual speed on the MMC and zero is read with system variable $AA_S[n].

References:
/FB1/Descriptions of Functions, Basic Machine; Various Interface Signals and Functions (A2)

Spindle behavior with G96, G961

Constant cutting rate

- At the start of machining (transition from G0 to Gx) and after NC stop, G60 (exact stop, modal) and G09 (exact stop, non-modal) the system waits until the actual speed has reached the speed setpoint tolerance range before starting the path.

  NC/PLC IS:
  DB31, ... DBX83.5 (nAct = nSet)
  is then set.

- NC PLC IS:
  DB31, ... DBX83.1 (setpoint speed limited)
  and
  DB31, ... DBX83.1 (setpoint speed limited)
  are also set to defined values even if severe 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 (axle/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.

Machine data:
MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START
can also influence spindle behavior significantly.

Spindle behavior at the end of gear step change

End of gear step change

- NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  tells the NC that the new gear step
  (NC/PLC IS DB31, ... DBX16.0-16.2 (actual gear step 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 step should correspond to the set gear step.

  The actual gear step signaled is relevant for selection of the parameter set.

- Once the gear step 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 step change, then the last speed and direction of rotation will be reactivated after the gear step change.

2.1.3 Spindle oscillation mode

Operating principle

Oscillation mode is activated for the spindle during the gear step change.

A detailed description of "Spindle oscillation mode" appears in:
References:
/FB1/ Descriptions of Functions, Basic Machine; Spindles (S1), Chapter: "Gear step change in oscillation mode".

2.1.4 Spindle positioning mode

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.

The spindle is in positioning mode with the following functions (the parameter [n], where n=spindle number, is not required for the main spindle):

- `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(.....)` identical to `SPOSA [n]= .....`
- `M19` or `M [n]=19`

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 distancetogo.

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 distance to be traveled.

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]=.....

Identical 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.

MD20850

To achieve transparency of `M19` and `SPOS` or `SPOSA` in respect of the response at the VDI interface, the auxiliary function `M19` is output to the VDI interface configured for `SPOS` and `SPOSA` and activated by setting machine data:
MD20850 SPOS_TO_VDI = 1.

After activation, the minimum duration of an `SPOS`/`SPOSA` block is increased to the time for output and acknowledgment of the auxiliary functions by the PLC.

M19 (DIN 66025)

`M19` can be used to position the spindle.

The position and positioning approach mode are read from setting data:
SD43240 M19_SPOS
and
SD43250 M19_SPOSMODE.

The positioning options with `M19` are identical to those of
`SPOS = <approach mode> <position/path>`.

`M19` is output as an auxiliary function to the VDI interface as an alternative to `M3`, `M4`, `M5`, and `M70`. The `M19` block remains active in the interpolator for the duration of the positioning motion (like `SPOS`).

Part programs using M19 as a macro (e.g., DEFINE M19 AS SPOS = 0) or as a subroutine, 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 functions group: 4
MD22010 $MC_AUXFU_ASSIGN_TYPE[0] = "M"               ; Auxiliary functions type: "M
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[0] = 0            ; Auxiliary functions
                                                       expansion: 0
MD22030 $MC_AUXFU_ASSIGN_VALUE[0] = 19               ; Auxiliary function value: 19
```

SD43240 and SD43250

The positioning data for M19 are stored in axis-specific setting data:
SD43240 $SA_M19_SPOS[n] (position)
and
SD43250 $SA_M19_SPOSMODE[n] (position approach mode).

End of positioning

The end of the positioning motion can be programmed using the following commands:

FINEA [Sn]	End of motion on reaching "Exact stop fine" (DB31, ... DBX60.7)
COARSEA [Sn]	End of motion on reaching "Exact stop coarse" (DB31, ... DBX60.6)
IPOENDA [Sn]	End of motion when “IPO stop” is reached

Block change

Settable block change time

In addition to the existing programmable endofmotion criterion with FINEA, COARSEA, IPOENDA, for singleaxis interpolation, a new endofmotion criterion can be programmed for block change with IPOBRKA in the braking ramp (100-0%).

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.

For more detailed information about the settable block change time of "settable positioning axes" for single-axis interpolation, please refer to:
References:
/FB2/ Description of Functions, Extended Functions; Positioning Axes (P2)

SPOS, M19 and SPOSA have the same functionality, but differ in their block change behavior:

- Programming with SPOS and M19

  The block change is carried out if all functions programmed in the block have reached their endofblock 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.

- Programming with 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:**

  If NC/PLC IS:
  DB31..., DBX 83.5 (spindle in setpoint range)
  is not active, channel-specific signal:
  DB2x, DBX 6.1 (read-in disable)
  can be set in order to wait for a particular spindle 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.

  Setting:
  MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 2
  stops traveling path axes before the start of machining at the last G0.

  Machining continues:

  – On 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)

- **In the part program:**

  Coordination actions in the part program have the following advantages:

  – The part programmer 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.



Caution

The part programmer 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 programmer must ensure that WAITS starts at the earliest in the phase in which the spindle of another channel has already started to accelerate or decelerate towards the new speed or direction of rotation.

- WAITS instructions are used for coordination in the part program:

WAITS	For main spindle (master spindle)
WAITS[n]	For main and other spindles
WAITS[n, m,...,q]	For several spindles up to the maximum number of spindles

	The control waits before executing subsequent blocks until:
Position(s) programmed with SPOSA	Are reached.
M5	The spindle is stationary. NC/PLC IS: DB31, ... DBX61.4 (spindle stationary) taking the tolerance: MD36060 $MA_STANDSTILL_VELO_TOL into account. WAITS is terminated and the next block loaded when the first occurrence of the signal is detected.
M3, M4 (speed control mode)	The speed is in the setpoint range. NC/PLC IS: DB31, ... DBX83.5 (spindle in setpoint range) 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 function of WAITS 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 IS:
  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 WAITS 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.

M function output

Auxiliary function M19 is output to the VDI interface by:

M[n]=19	Always output to the IS
SPOS [n]	Output of M19 to the IS with MD setting: MD20850 $MC_SPOS_TO_VDI = 1
SPOSA[n]	Output of M19 to the IS with MD setting: MD20850 $MC_SPOS_TO_VDI = 1

If M19 is programmed, auxiliary function M [n]=19 is always output to the interface.

For SPOS and SPOSAM19is output depending on machine data:
MD20850 $MC_SPOS_TO_VDI
to the IS.

Feed

The positioning speed is configured using machine data:
MD35300 $MA_SPIND_POSCTRL_VELO
and can be modified by programming or by synchronized actions:

`FA[Sn]`	Where n = spindle number
`FA[Sn]=0`	The configured speed is active

The speed is specified in [degrees/min].

Acceleration

The dynamic response during positioning can be modified by programming or by synchronized actions:

`ACC[Sn]`	Programming or synchronized action
`ACC[Sn]=0`	The configured acceleration is active

n: Spindle number, 0 ... max. spindle number

Positioning from rotation

The spindle can be in speed control mode or in position control mode when positioning starts (`SPOS`, `M19` or `SPOSA` command in the program).

The following sequence is obtained:

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

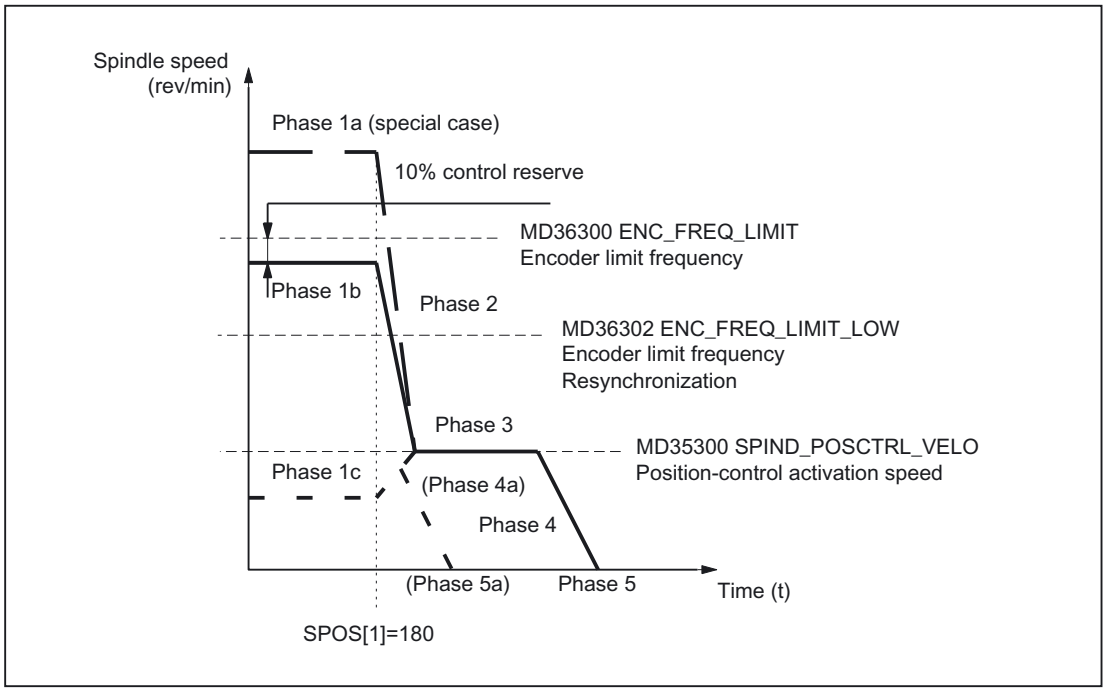


Fig. 2-2 Positioning from rotation

Note

The speed from machine data:
MD36302 $MA_ENC_FREQ_LOW
must be greater than the position-control activation speed
(MD35300 $MA_SPIND_POSCTRL_VELO).

Phase 1

Possible positioning from Phase 1a:

The spindle is rotating at a higher speed than the encoder limit frequency. The spindle is not synchronized.

Possible positioning from Phase 1b:

The spindle is rotating at a lower speed than the encoder limit frequency. The spindle is synchronized.

Note

If 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).

Possible positioning from Phase 1c:

The spindle rotates at the programmed spindle speed,
when the speed is lower than the speed entered in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO (position-control activation speed).
The spindle is synchronized.

Phase 2

Spindle speed > Position-control activation speed

The command `SPOS`, `M19` or `SPOSA` starts deceleration of the spindle with the acceleration stored in machine data:
MD35200 $MA_GEAR_STEP_SPEEDCTL_ACCEL
until it reaches the position-control activation speed.
When the speed drops below the encoder limit frequency, the spindle is synchronized.

Spindle speed < Position-control activation speed

When `SPOS`, `M19` or `SPOSA` is programmed, the spindle switches to position control mode (if it is not yet in position control mode).
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL (acceleration in position control mode) is activated. The travel path to the end 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, the operational sequences in phases 2 - 3 - 4 - 5, or -2 - 4a - 5a are executed.

Phase 3

Spindle speed > Position-control activation speed

When the position-control activation speed stored in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO
is reached:

- Position control is activated (if not already active)
- The distance-to-go (to end point) is calculated
- Acceleration in machine data:
  MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL (Acceleration in position control mode) is activated (or this level of acceleration is maintained)

Spindle speed < Position-control activation speed

To reach the end point, the spindle is accelerated up to the speed defined in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO (position-control activation speed).

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.

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 end point.

Spindle speed < Position-control activation speed

At the time identified by the braking start point calculation in phase 3,
the spindle brakes to standstill with the acceleration from machine data:
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 up to machine data:
MD35300 $MA_SPIND_POSCTRL_VELO.

The spindle is braked to standstill with the acceleration in machine data:
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 measurement system limit speed.

NC/PLC IS:
DB31, ... DBX83.1 (programmed speed too high)
is set.

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 has reached the end point. Position control is active and holds 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 machine data:
MD36010 $MA_STOP_LIMIT_FINE
and
MD36000 $MA_STOP_LIMIT_COARSE.

Note

The positioning action is considered complete when the end-of-positioning criterion is met and signaled.
This requires "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.

Positioning from standstill

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.

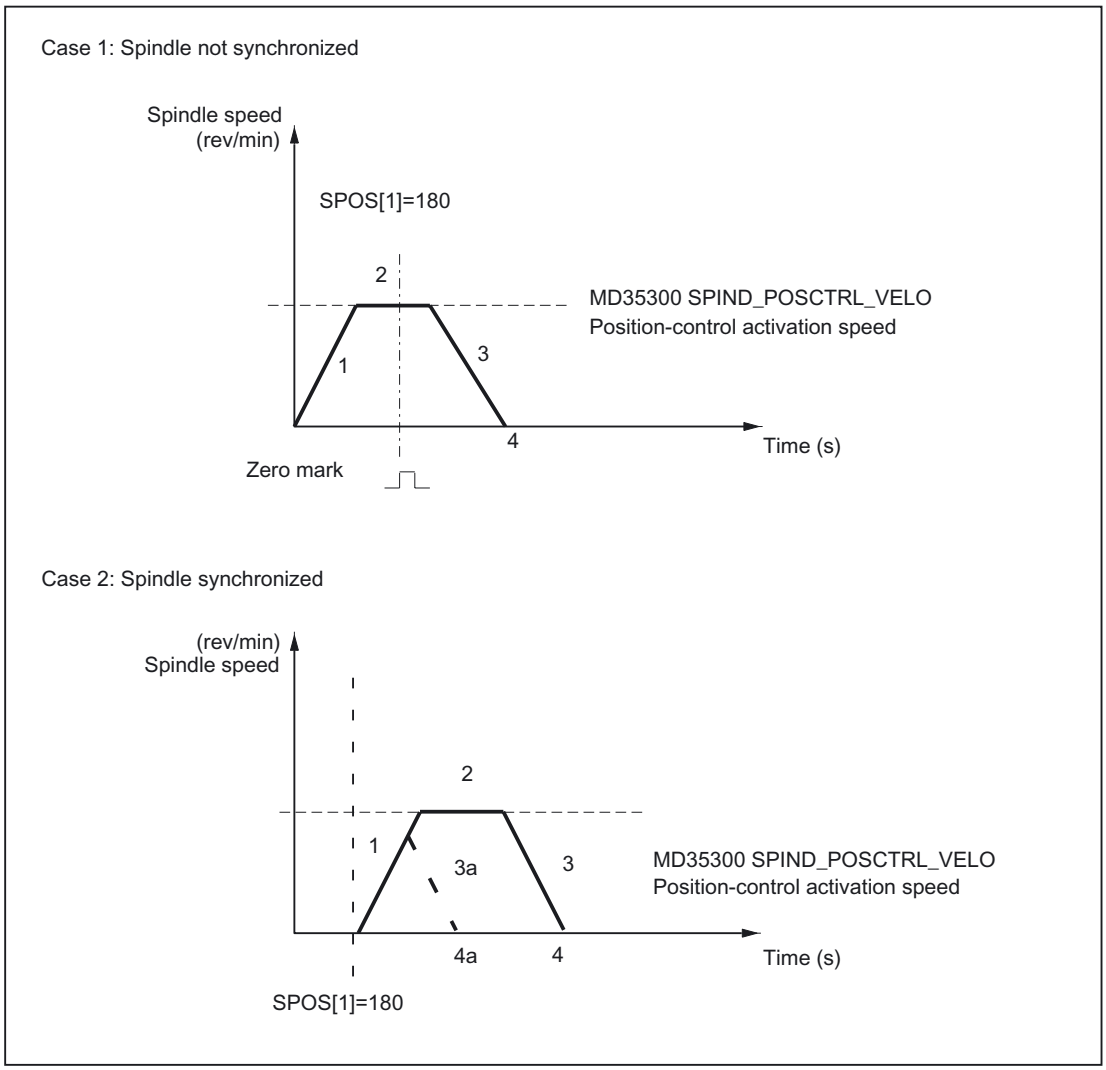


Fig. 2-3 Positioning with stationary spindle

Phase 1

Case 1: Spindle not synchronized

The programming of `SPOS`, `M19` or `SPOSA` accelerates the spindle with the acceleration from machine data:
MD35200 $MA_GEAR_STEP_POSCTRL_ACCEL (Acceleration in positioning mode).

The direction of rotation is defined in machine data:
MD35350 $MA_SPIND_POSITIONING_DIR (Direction of rotation when positioning from standstill).

Exception:
If `ACN`, `ACP`, `IC` is used for positioning, the programmed direction of travel is activated.

The spindle is synchronized with the next zero mark of the spindle position actual-value encoder and switches to position control mode.

A watchdog checks whether the zero mark can be found in the path stored in machine data:
MD34060 $MA_REFP_MAX_MARKER_DIST
(excluding IC).

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 creep speed (the spindle is not accelerated further).

Case 2: Spindle synchronized

`SPOS`, `M19` or `SPOSA` will switch the spindle to position control mode.

Acceleration from machine data:
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL (acceleration in position control mode)
is activated.

The direction of rotation is defined by the programmed motion (`ACP`, `ACN`, `IC`, `DC`) or by the distancetogo.
The speed defined in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO (position-control activation speed)
is not exceeded.

The travel path to the end 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, the operational sequences in 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 time identified by the braking start point calculation in phase 1, the spindle brakes to standstill with the acceleration from machine data:
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL.

Phase 3

At the time identified by the braking start point calculation in phase 2, the spindle brakes to standstill with the acceleration from 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 standstill with the acceleration in machine data:
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL.

Phase 4, 4a

The spindle is stationary and has reached the end point. Position control is active and holds 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 machine data:
MD36010 $MA_STOP_LIMIT_FINE
and
MD36000 $MA_STOP_LIMIT_COARSE.

Phase 3:

At the time identified by the braking start point calculation in phase 2,
the spindle brakes to standstill with the acceleration from machine data:
MD35210 $MA_GEAR_STEP_ POSCTRL_ACCEL.

Phase 4:

The spindle is stationary and has reached the end point. Position control is active and holds 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 machine data:
MD36010 $MA_STOP_LIMIT_FINE
and
MD36000 $MA_STOP_LIMIT_COARSE.

Aborting the positioning process

The positioning action can only be aborted with NC/PLC IS:
DB31, ... DBX2.2 (delete distance-to-go/Spindle reset).

The positioning action is aborted with each reset (Operator panel front reset/NC/PLC IS DB31, ... DBX2.2 (delete distance-to-go/Spindle reset)),
independent of machine data:
MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (individual spindle reset).

Special points to be noted

- The accelerations are defined in the following machine data:
  - MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL
    (acceleration in position control mode)
  - MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL
    (acceleration in speed control mode)
- The spindle override switch is active.
- The positioning action (`SPOS`, `M19` and `SPOSA`) is canceled with each reset.
- The positioning action is canceled with `NC STOP`.
- The positioning velocity can also be programmed with `FA[Sn]`.

2.1.5 Axis mode

Why axis mode?

For certain machining tasks (e.g., on turning machines with end-face machining), the spindle not only has to be rotated with `M3`, `M4` and `M5` and positioned with `SPOS`, `M19` and `SPOSA`, but also addressed as an axis with its own identifier (e.g., C).

Prerequisites

- 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.
- A position actual-value encoder is a mandatory requirement for axis mode.
- If the axis is not synchronized, e.g., `M70` is programmed after POWER ON, the axis must first be homed with `G74`. Only then does the mechanical position match the programmed one.

  Example:

```
M70
G74 C1=0 Z100
G0 C180 X50
```

M function output

The M function, which switches the spindle to axis mode, can be configured in machine data:
MD20094 $MC_SPIND_RIGID_TAPPING_M_NR.
The value on delivery is 70.

Functionality

If the axis mode is active and the rotary axis homed, all axis functions can be used.

References:
/FB2/ Description of Functions, Expansion Functions; Rotary axes (R2)

The most important functions are:

- Programming with axis name
- Use of zero offsets (`G54, G55`, `TRANS`, etc.)
- `G90`, `G91`, `IC`, `AC`, `DC`, `ACP`, `ACN`
- Use of kinematic transformations (e.g., `transmit`)
- Interpolation with other axes (path interpolation)
- Programming as a positioning axis

Special points to be noted

- The feed override switch is active.
- NC/PLC IS:
  DB21, ... DBX7.7 (Reset)
  does not terminate axis mode as standard.

- NC/PLC IS:
  DBB16 to DBB19 and DBB82 to DBB91 in DB31, ...
  are of no significance if NC/PLC IS:
  DB31, ... DBX60.0 (axis/no spindle)
  is set to zero.
- Axis mode can be activated in all gear steps.
  If the position actual-value encoder is installed on the motor (indirect measurement system), the positioning and contouring accuracy may 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 machine data of the parameter set with index zero are effective in order to carry out adaptation in this mode.

Transition to axis mode

Transition to axis mode by programming:

- The spindle with its own identifier or by M70 and by means of the `M` function:
  MD20094 $MC_SPIND_RIGID_TAPPING_M_NR
- The relevant machine data when changing the servo parameter set are:

MD31050 $MA_DRIVE_AX_RATIO_DENOM	(Measuring gear denominator)
MD31060 $MA_DRIVE_AX_RATIO_NUMERA	(Numerator load gear unit)
MD32200 $MA_POSCTRL_GAIN	(Servo gain factor (Kv))
MD32452 $MA_BACKLASH_FACTOR	(Weighting factor for backlash)
MD32610 $MA_VELO_FFW_WEIGHT	(Weighting factor for feedforward control)
MD32800 $MA_EQUIV_CURRCTRL_TIME	(Equivalent time constant current control circuit for feedforward control)
MD32810 $MA_EQUIV_SPEEDCTRL_TIME	(Equivalent time constant speed control loop for feedforward control)
MD32910 $MA_DYN_MATCH_TIME	(Time constant for dynamic matching)
MD36012 $MA_STOP_LIMIT_FACTOR	(Factor for exact stop coarse/fine and zero speed control)
MD36200 $MA_AX_VELO_LIMIT	(Threshold value for velocity monitoring)

- The dynamic limits of the axis stored in the machine data are applicable in axis operation.
- The axis switches to the current feedforward control mode as designated by the MD and the commands `FFWON` and `FFWOF`.

  For more information about the servo parameter set, please refer to:
  References:
  /FB1/ Descriptions of Functions, Basic Machine; Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)
- When using resolution changes in (analog) drive actuators, the following NC program steps are required:

Table 2-1 Change over to axis mode:

`SPOS=...`	
`M5`	`Closed-loop controller enable off (by PLC)`
	`→ Output to PLC`
`M70`	`Switch actuator (by PLC on account of M70)`
	`Closed-loop controller enable on (by PLC)`
`C=...`	`NC travels with axis parameter set`

Table 2-2 Switch back to spindle mode

`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`	`NC travels with spindle parameter set`
`or`	
`SPOS=...`	

Change to spindle mode

The interpolation parameter (set 1 ... 5) is selected according to the currently valid gear step.

The feedforward control function is always activated, except for tapping with compensating chuck.
Machine data:
MD32620 $MA_FFW_MODE (feedforward control type)
must always be not equal to 0.

Feedforward control should always be operated with the value 100% to avoid alarms being output during positioning.

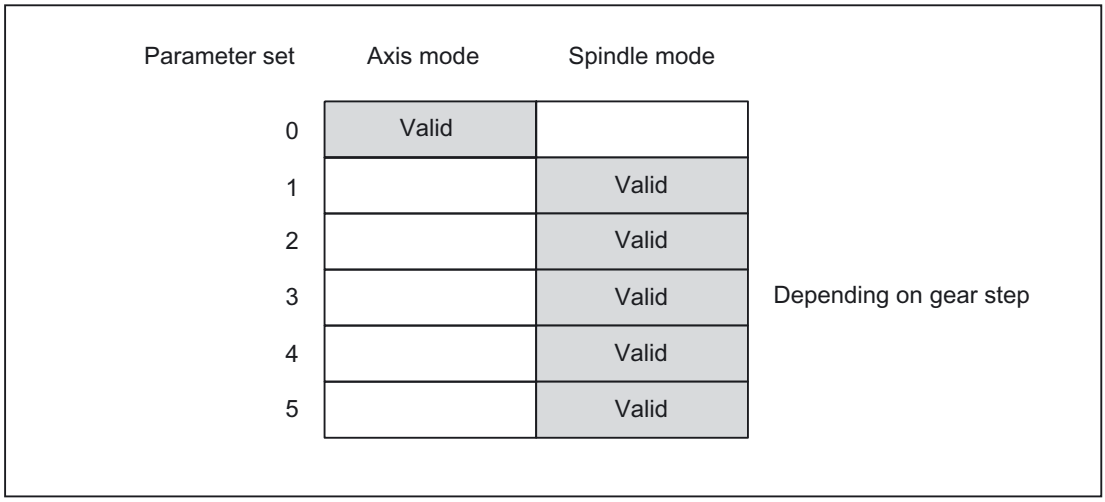


Fig. 2-4 Validity of parameter sets for axis and spindle modes

2.1.6 Default setting of the spindle mode

Machine data

The default setting of the spindle mode allows configuration of the basic spindle setting following Power On, NC-START and RESET using machine data:
MD35020 $MA_SPIND_DEFAULT_MODE
and
MD35030 $MA_SPIND_DEFAULT_ACT_MASK.

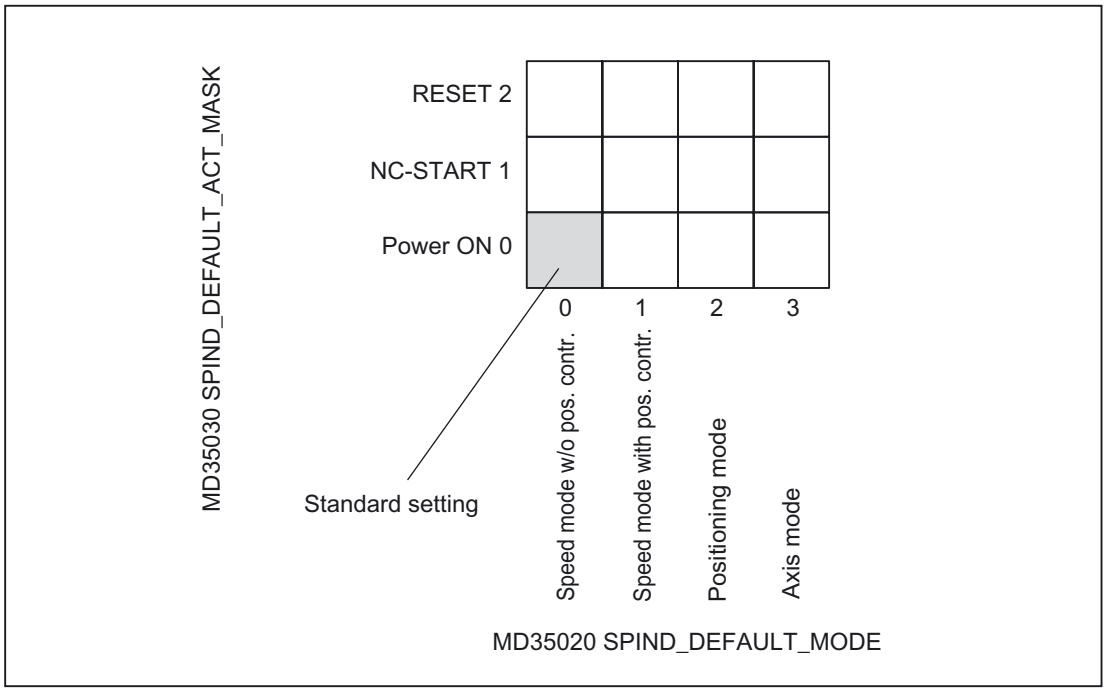


Fig. 2-5 Programmable defaults for spindle operating mode

2.2 Reference/Synchronize

Why synchronize?

In order to ensure that the control detects the exact position when it is switched on, the control must be synchronized with the position measurement system of the spindle. This process is known as synchronization.

Only a synchronized spindle is capable of:

- Thread cutting
- Rigid tapping
- Being programmed as an axis

Additional information about synchronizing the spindle can be found in
References:
/FB1/ Description of Functions, Basic Machine; Homing (R1)

Why reference?

In order to ensure that the control detects the exact machine zero when it is switched on, the control must be synchronized with the position measurement system of the rotary axis. This process is known as homing. The sequence of operations required to home an axis is known as homing.

Only a homed axis can approach a programmed position accurately on the machine.

Additional information about homing the rotary axis can be found in
References:
/FB1/ Description of Functions, Basic Machine; Homing (R1)

Installation position of the position measurement system

The position measurement systems can be installed as follows:

- Direct 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 measurement systems are provided, they can be installed either in the same location or separately.

Synchronization procedure

When the spindle is switched on, the spindle 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 movement (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 negative or positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.
- If does not make any difference whether the procedure is initiated from the part program, `FC 18` or synchronized actions.
- Crossing the zero mark in JOG mode by means of direction keys in speed control mode.

Note

During synchronization of the spindle, all four possible reference point values and reference point offsets are effective as appropriate to the measurement 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/end point for distance-coded system)

MD34090 $MA_REFP_MOVE_DIST_CORR
(reference point offset/absolute offset, distance-coded)

MD34100 $MA_REFP_SET_POS
(reference point value, of no relevance in the case of distance-coded system).

If a non-homed 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 at position 2)
 and
 DB31, ... DBX17.4 (resynchronize spindle at position 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.

A signal runtime compensation (dead time) for positive and negative directions of motion in conjunction with the machine data setting:
MD34200 $MA_ENC_REFP_MODE = 2 or 7
is entered in machine data:
MD31122 $MA_BERO_DELAY_TIME_PLUS
or
MD31123 $MA_BERO_DELAY_TIME_MINUS.

- Setting:
  MD34200 $MA_ENC_REFP_MODE = 7
  executes position synchronization only at a fixed velocity/speed defined in machine data:
  MD34040 $MA_REFP_VELO_SEARCH_MARKER.

  The velocity defined in machine data:
  MD34040 $MA_REFP_VELO_SEARCH_MARKER
  is also effective when homing in operating mode JOG-REF and through the part program with `G74`.

- Setting:
  MD34200 $MA_ENC_REFP_MODE = 2
  executes position synchronization without specifying a specific velocity/speed.

Note

The overriding of the signal propagation delay by the NC requires the use of type 611D drives.

Signal propagation delays are preset on delivery so that the content generally does not have to be changed.

Homing sequence

If the spindle is to be programmed in axis mode directly after control power up, it must be ensured that the axis is homed.

When the control is switched on, the spindle can be homed (condition is one zero mark per revolution).

Information about the homing sequence can be found in
References:
/FB1/ Descriptions of Functions, Basic Machine; Homing (R1)

The rotary axis is homed at the same time as the spindle is synchronized
(see Synchronization procedure) if the position measurement 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 control, but only one can be active.

NC/PLC IS:
DB31, ... DBX1.5 (position measurement system 1)
and
DB31, ... DBX1.6 (position measurement system 2)
are used to select the active position measurement system.

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 value linkage

NC/PLC IS DB31, ... DBX16.4 (Resynchronize 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 step change is performed. Synchronization is triggered internally once the spindle is rotating in the new gear step (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 control. When the system switches from the vertical to the horizontal spindle, the spindle must be resynchronized.

  This synchronization action is triggered with NC/PLC IS:
  DB31, ... DBX16.4 (resynchronize spindle 1)
  or
  DB31, ... DBX16.5 (resynchronize spindle 2).

  The spindle must be in open-loop control mode.

2.3 Configurable gear adaptation

2.3.1 Gear steps for spindles and gear step change

Why do we need gear steps?

Gear steps are used on spindles to step down the speed of revolution 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 steps

5 gear steps can be configured for each spindle.
If the spindle motor is mounted to the spindle directly (1:1) or with a non-adjustable gear ratio, machine data:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE
must be set to zero.
In this case, the 1st gear step is active. This is the default setting.

Selection of gear change type

Configuring machine data:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (gear step change is possible)
determines the gear step change type as follows:

0:	Spindle motor installed directly (1:1) or with a non-variable transmission ratio.
1:	Spindle motor with up to 5 gear steps. The gear step change is executed in oscillation mode.
2:	Spindle motor with up to 5 gear steps. The gear step change is executed at the configured fixed position.

Requirement for a gear step change:

Activated by the NC by setting machine data:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE **not equal to 0**
and by the part program at `M40` with `S`... or `M41`...`M45`.

In principle, the gear step change is only performed, however, if the requested gear step is not the same as the active gear step.

The following section explains who can request a gear step change and how it is triggered, initiated and completed.

Ways of specifying a gear step change

A gear step change can be requested:

- In the part program

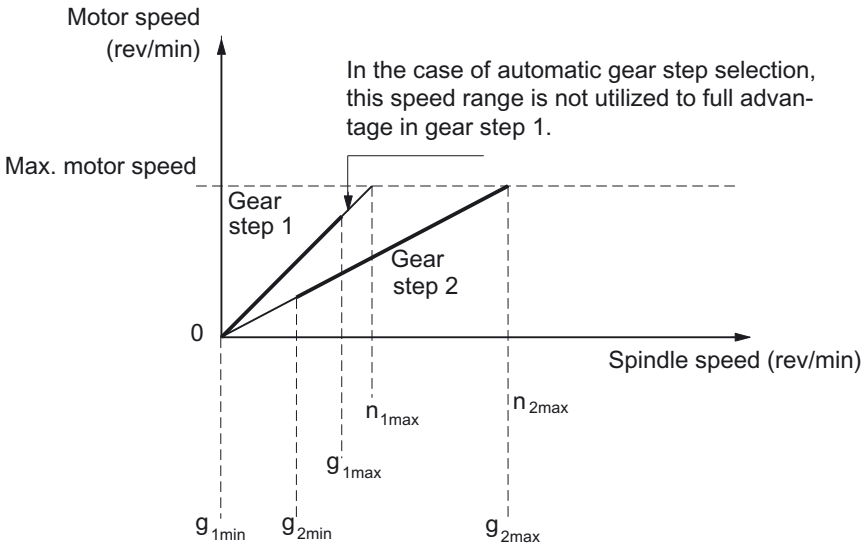
  Automatically by means of the programmed spindle speed for `M40` without `S` or by means of programming with `M41` to `M45`.
- By the PLC using function module `FC 18`.
- By synchronized actions with `M40` and `S` or `M41` to `M45`.
- In the reset state by writing to the VDI interface.

  In particular after a power on, the NC can be informed of the latest gear step.
- In the reset state in the event of NC stop.

  In the case of a manual gear step change directly at the installation location.

Selection between two gear steps

Gear step selection between two gear steps with specification of a maximum spindle speed is shown in the example below:



Specifiable using MD35110 GEAR_STEP_MAX_VELO:

n_{1max} ... Maximum spindle speed for gear step 1

g_{1min} ... Minimum spindle speed for gear step 1, automatic gear step selection

g_{1max} ... Maximum spindle speed for gear step 1, automatic gear step selection

n_{2max} ... Maximum spindle speed for gear step 2

g_{2min} ... Minimum spindle speed for gear step 2, automatic gear step selection

g_{2max} ... Maximum spindle speed for gear step 2, automatic gear step selection

Fig. 2-6 Gear step change with selection between two gear steps

Parameters on gear step change

During the gear step change interpolation parameters and, as standard, servo parameter sets of the position controller are activated depending on machine data:
MD35590 $MA_PARAMSET_CHANGE_ENABLE.

Servo parameter sets of position controller 1 to 6:
The servo parameter sets are used for fast matching of the position controller to changed properties of the machine during operation on gear change of the spindle.

If the spindle is operating in axis mode, the new gear step is stored internally and parameter set index "0" remains active in the servo. The new gear step is activated the **next** time the spindle is programmed.

For more information about closed-loop control and the servo parameter set, please refer to:
References:
/FB1/ Descriptions of Functions, Basic Machine; Velocities, Setpoint/Actual-Value Systems, ClosedLoop Control (G2)

/PGA/ Programming Guide Advanced/Programmable Servo Parameter Set

The **interpolation parameters** specify gear steps 1 to 5
at the VDI interface for:
DB31, ... DBX82.0-82.2 (set gear step A to C).

The appropriate
servo parameter set is activated depending on:
DB31, ... DBX16.0-16.2 (actual gear step A to C).

One set of parameters, with the following assignment, is provided by the NC for each of the 5 gear steps:

Data set Spindle data	VDI interface	Parameter set, index n	Data/content of the data set
Data for axis mode	*	0	Monitoring functions
Data for 1st gear step	001	1	M40 speed Min./max. speed Acceleration rates K_V factor Transmission ratio
Data for 2nd gear step	010	2	
Data for 3rd gear step	011	3	
Data for 4th gear step	100	4	
Data for 5th gear step	110 111	5	

* = the last active gear step

The parameter sets of gear steps 1 to 5 can be configured with the following machine data:

	Min./max. speed
MD35110 $MA_GEAR_STEP_MAX_VELO[n]	Max. gear step change
MD35120 $MA_GEAR_STEP_MIN_VELO[n]	Min. gear step change
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[n]	Max. gear step
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[n]	Min. gear step
	Acceleration in:
MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL[n]	Speed control mode:
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL[n]	Position controller mode
MD35012 $MA_GEAR_STEP_CHANGE_POSITION[n]	Gear step change position

Changing parameter sets and formatting

The gear step change also switches over the servo parameter set,
if machine data:
MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0 or 1.

0:	The parameter sets **cannot be controlled**.
1:	The servo parameter set is defined primarily **by the internal NC switchover** at the VDI interface.

If the parameter sets cannot be controlled:

- For axes, the 1st parameter set with the index 0 is always the one that is applicable.
- For spindles, the 2nd to the 6th parameter set **plus one** is always active depending on the gear step selected.

On stipulation of the servo parameter set by the VDI interface, parameter sets 1 to 6 can be activated via indices 0 to 5.

The following applies:

- For the axes involved, the parameter set number that corresponds to the master spindle gear step **incremented by one** is active. This then corresponds to parameter set numbers 2 to 6.
- For spindles, the 2nd to the 6th parameter set plus one is always active depending on the gear step selected.

If:
MD35590 $MA_PARAMSET_CHANGE_ENABLE = 2,
the servo parameter set is **specified by the PLC**.
A gear step can be specified by the PLC at any time.

Sequence for switchover

If the new gear step is preselected, the following sequence is implemented:

NC/PLC interface signals:
DB31, ... DBX82.0-82.2 (set gear step A to C)
and
DB31, ... DBX82.3 (change gear)
are set.

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
(NC/PLC IS: DB31, ... DBX61.4 (axis/spindle stationary)
with NC/PLC IS:
DB31, ... DBX18.5
(oscillation speed).

In principle, the new gear step can also be engaged without oscillation

When the new gear step is engaged,
the NC/PLC interface signals DB31, ... DBX16.0-16.2 (actual gear step A to C)
and
DB31, ... DBX16.3 (gear changed)
are set by the PLC program.

End of gear step change

The gear step change is considered completed ("oscillation" spindle mode is deselected) and the spindle is switched to the servo and interpolation parameter set of the new actual gear step.

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).

NC/PLC IS:
DB31, ... DBX16.3 (gear changed)
tells the NC that the new gear step applies with NC/PLC IS:
DB31, ... DBX16.0-16.2 (actual gear step A to C)
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 step, which should correspond to the set gear step, is relevant for selecting the parameter set.

If this is not the case, machine data setting:
MD11410 $MN_SUPPRESS_ALARM_MASK, Bit 3 = 0
triggers alarm 22010.

Following acknowledgment of gear step change via the PLC
with NC/PLC IS:
DB31, ... DBX16.3 (gear changed)
the spindle is in speed control mode (DB31, ... DBX84.7).

For more information about signal exchange between the PLC and NC, please refer to:
References:
/FB1/ Descriptions of Functions, Basic Machine; Various Interface Signals (A2)

Specify gear step in the part program

In the case of `M40`, the spindle must be in open-loop control mode for automatic gear step selection with an S value. The gear step change is otherwise rejected and alarm 22000 "Gear change not possible" is output.

Automatic selection with active M40

The gear step is automatically selected by the control. The control checks which gear step is possible for the programmed spindle speed (S value).

If the suggested gear step is not the same as the current (actual) gear step,
NC/PLC IS:
DB31, ... DBX82.3 (change gear)
and
DB31, ... DBX82.0-82.2 (set gear step A to C)
are set.

While the appropriate gear step is being determined, a gear step change is only requested if the new speed is not within the permissible speed range of the active gear step.

If necessary, the speed is limited to the max. speed of the current gear step or increased to the minimum speed of the current gear step, and NC/PLC IS:
DB31, ... DBX83.1 (setpoint speed limited)
is set.

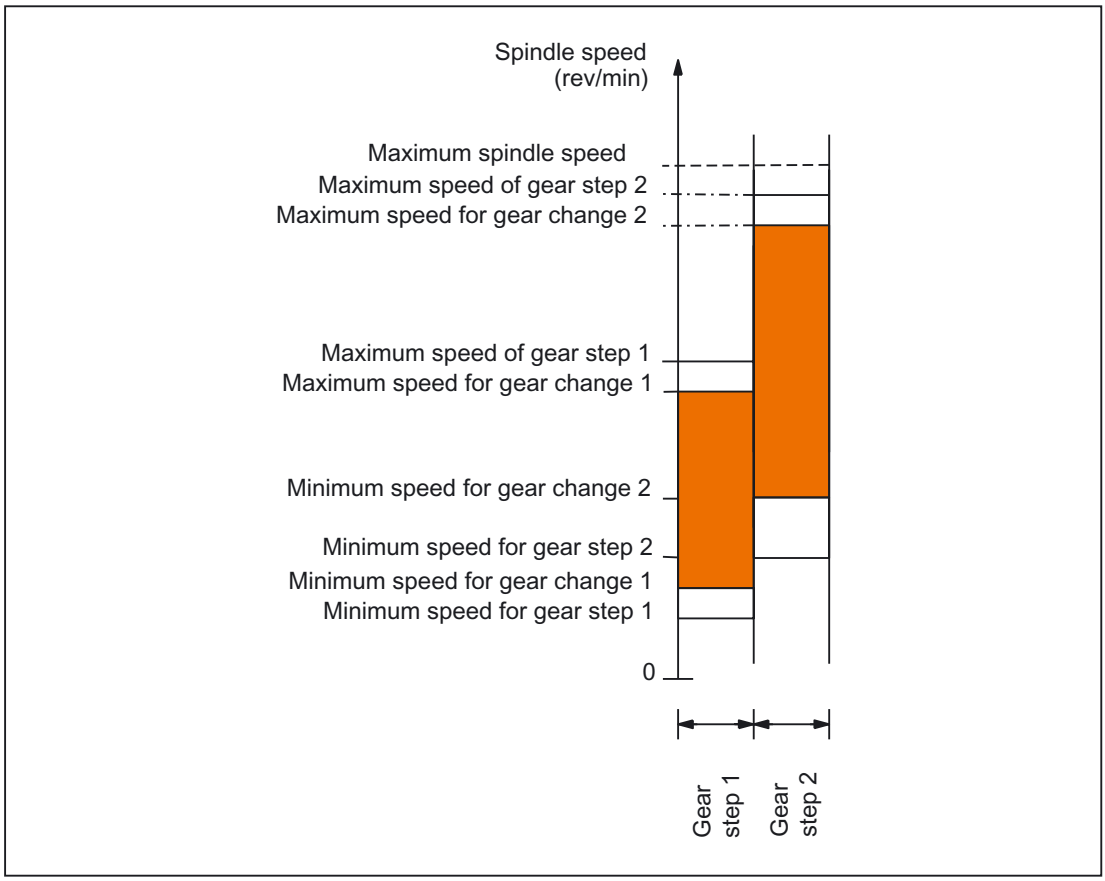


Fig. 2-7 Example for two "gear steps with overlapping speed ranges" for automatic gear step change (M40)

Permanently defining the gear step with M41 to M45

The gear step can be permanently defined in the part program with M41 to M45.

If a gear step is specified by M41 to M45, which is not the same as the current (actual) gear step, NC/PLC IS:
DB31, ... DBX82.3 (Change gear)
and
DB31, ... DBX82.0-82.2 (Set gear step A to C)
are set.

The programmed spindle speed (S value) then refers to this permanently defined gear step.

If a spindle speed is programmed, which exceeds the max. speed of the defined gear step, the spindle speed is limited to the max. speed of the gear step:
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT
and NC/PLC IS:
DB31, ... DBX83.1 (setpoint speed limited)
is set.

Block change

When programming the gear step change in the part program, the gear step change set remains active until the gear step change is aborted by PLC. (Same effect, as if NC/PLC IS: DB21, ... DBX6.1 (Read-in enable) is set.)

Specification of gear step via PLC with FC18

The gear step 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 step as appropriate for the speed. This corresponds to an automatic gear step change with `M40`.

The gear step is not changed if the spindle is positioned by FC18 or if traversing in axis mode.

For more information about function block FC18, please refer to:
References:
/FB1/ Descriptions of Functions, Basic Machine; PLC Basic Program (P3)

Specification of gear step with synchronized actions

The gear step change can be requested by synchronized actions:

- In the event of automatic gear step change for `M40` with S

  or

- When specifying gear steps 1 to 5 with `M41` to `M45`.

The gear step is not changed if the spindle is positioned by synchronized actions or if traversing in axis mode.

Manual specification of a gear step

Outside a part program that is running, the gear step can also be changed without a request from the NC or the machine. This is the case, for example, when a gear step is changed directly by hand.

To select the appropriate parameter sets, the NC must be informed of the current gear step. For this to work, the control or the part program must be in the reset state.

Supplementary conditions

Transfer of the gear step to the NC is initiated when
NC/PLC IS:
DB31, ... DBX16.0-16.2 (actual gear step 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 step 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 step is specified by the PLC with NC/PLC IS:
DB31, ... DBX16.0-16.2) (actual gear step A to C),
it is disabled throughout the gear change action.

NC stop during gear step 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 step change
- NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  is not set.

Note

Options for aborting:

NC/PLC IS:
DB31, ... DBX2.2 (delete distance-to-go/spindle reset)
or
DB31, ... DBX16.3 (gear changed)
with corresponding acknowledgment from actual gear step
(DB31, ... DBX16.0-16.2 (actual gear step)).

Spindle response after a gear step change

How the spindle behaves once the gear step has been changed depends on the following initial conditions:

- If the spindle was in the stop state before the gear step change (`M5`, FC18: "Stop rotate spindle"), in positioning or axis mode, `M5` (spindle stop) is active after completion of the gear step change.
- If a direction of rotation (`M3`, `M4`, or FC18: "Start spindle rotation"), then the last speed and direction of rotation will become active again after the gear step change. In the new gear step, the spindle accelerates to the last spindle speed programmed (`S` value).

  The next block in the part program can be executed.

Special points to be noted

The following points must be observed on gear step change:

- The gear step change is not terminated by selecting
  NC/PLC IS:
  DB31, ... DBX20.1 (ramp-up switchover to V/f mode).
  Setpoint 0 is output.

  The acknowledgment of the gear step change is carried out as usual
  by the NC/PLC IS:
  DB31, ... DBX16.3 (gear changed).
- The "Ramp-function generator rapid stop" signal must be reset by the PLC before the gear step change is completed by the PLC.
- The gear step change procedure is terminated without an alarm output on an NC reset.
  The gear step output with NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear step 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 more information about function block FC17, please refer to:
References:
/FB1/ Descriptions of Functions, Basic Machine; PLC Basic Program (P3)

Parameter set of the load gearbox ratio

It is possible to configure positive or **negative load gearbox factors** for each gear step and in axis mode.

The setting is defined in machine data:

MD31060 $MA_DRIVE_AX_RATIO_NUMERA.

The setting range is the same size for positive and negative load gearbox factors.
It is not possible to enter the value 0.

Note

The reference will be lost if an indirect encoder is configured and the load gearbox ratio changes.

NC/PLC IS:
DB31, ... DBX60.4/60.5 (Homed/Synchronized 1 or 2)
is reset for the relevant measuring system.

2.3.2 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`.

```
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 more information about closed-loop control conditions, please refer to
References:
/FB1/ Descriptions of Functions, Basic Machine; Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)

2.3.3 If the gear step change is not acknowledged

Mode change

A gear step 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 step 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 step 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 step change.
If this time elapses without the gear step 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 replacement
- 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`
- Subroutine level abort, subroutine abort

Response after POWER ON

The active gear step on the machine can be specified by the PLC control 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 step A to C)
to the PLC.

2.3.4 Gear step change with oscillation mode

What is oscillation?

Oscillation in this context means that the spindle motor rotates alternately in the clockwise and counterclockwise directions. This oscillation movement makes it easy to engage a new gear step.

Oscillation mode

NC/PLC IS:
DB31, ... DBX82.3 (change gear)
displays that a gear step change is required.

In principle, the new gear step can also be engaged without oscillation

1. Machine data:
   MD35010 $MA_GEAR_STEP_CHANGE_ENABLE
   must be set to 1.
2. NC/PLC IS:
   DB31, ... DBX84.6 (oscillation mode)
   is set.
3. The acceleration is defined in machine data:
   MD35410 $MA_SPIND_OSCILL_ACCEL.

DB31, ... DBX18.5 (oscillation speed)

The spindle is in oscillation mode if a new gear step was defined using automatic gear step 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 step is defined
that is not the same as the current actual gear step.

If NC/PLC IS:
DB31, ... DBX18.5 (oscillation speed)
is simply set by the PLC without a new gear step 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:
/FB1/ Description of Functions, Basic Machine, 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).

The start direction is defined in machine data:
MD35430 $MA_SPIND_OSCILL_START_DIR (oscillation start direction).

The time t1 (or t2) is started
depending on the start direction defined in 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 passed, 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 counterclockwise rotation) must be simulated on the PLC.

Special points to be noted

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 step change is requested.
- The gear step 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 step 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 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 (homed/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 step.

Functionality

Machine tools of conventional design require a gear step change 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 step 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 reengaging of the gear wheels.

- Writing of NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear step 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.

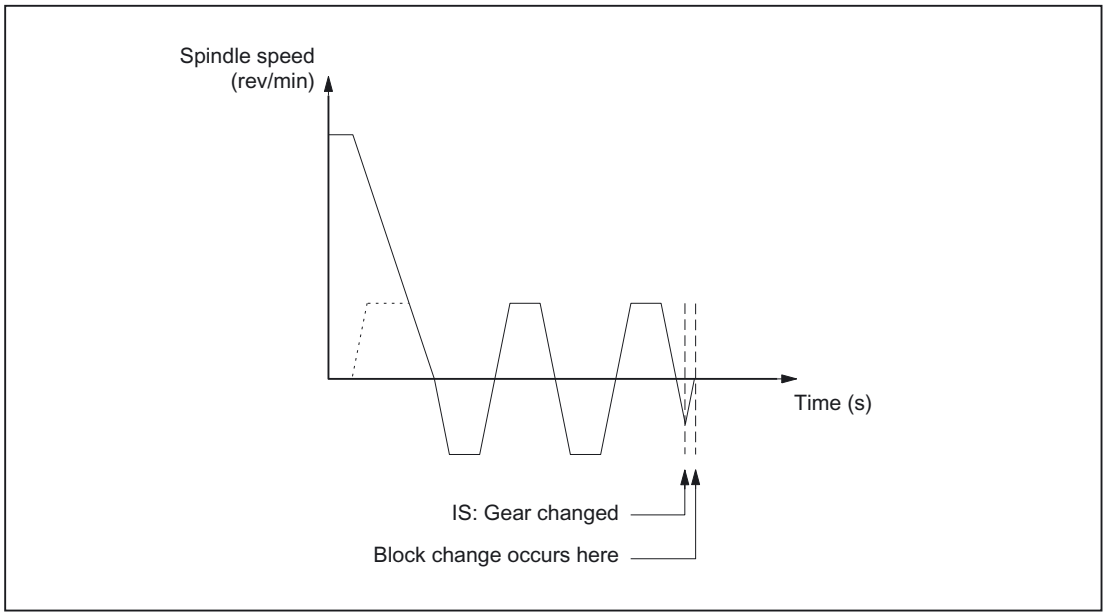
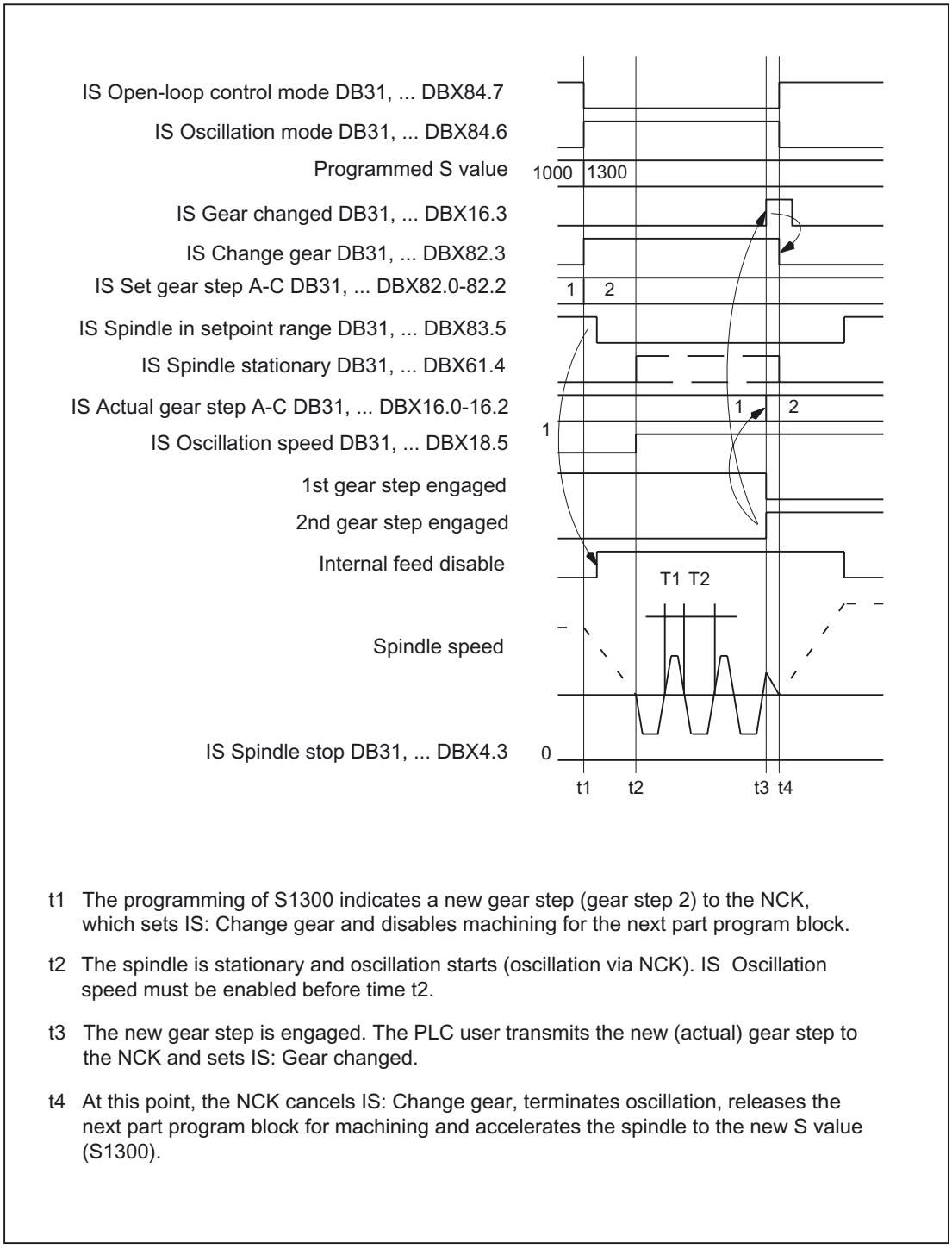


Fig. 2-8 Block change following oscillation mode

Oscillation mode

Typical time sequence for the gear step change with a spindle:



t1 The programming of S1300 indicates a new gear step (gear step 2) to the NCK, which sets IS: Change gear and disables machining for the next part program block.

t2 The spindle is stationary and oscillation starts (oscillation via NCK). IS Oscillation speed must be enabled before time t2.

t3 The new gear step is engaged. The PLC user transmits the new (actual) gear step to the NCK and sets IS: Gear changed.

t4 At this point, the NCK cancels IS: Change gear, terminates oscillation, releases the next part program block for machining and accelerates the spindle to the new S value (S1300).

Fig. 2-9 Gear step change with stationary spindle

2.3.5 Gear step change at fixed position

Application and advantages

Machine tools increasingly use standardized spindle drives, firstly to save technological dead time on a gear step change and secondly to gain the cost benefits of using standardized components.

The Gear step change at fixed position function supports the "directed gear step change" of load gearboxes that need to be activated in a different way than the NC. The gear step change can in this case only be performed at a defined spindle position. An oscillation motion as required by conventional load gearboxes is thus no longer necessary.

Sequence for gear step change at fixed position

Gear step 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 across the position configured in machine data:
  MD35012 $MA_GEAR_STEP_CHANGE_POSITION.

  If the gear step 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:
  DB 31, ... 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 step 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 step change dialog starts.
- Output of VDI interface signals:
  DB31, ... DBX84.6 (oscillation mode)
  DB31, ... DBX82.3 (change gear)
  DB31, ... DBX82.0-82.2 (set gear step 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 "Servo enable":
  DB31, ... DBX2.1 = 0.

- Mechanical switchover of the gear step on the machine.

  No oscillation motion 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 step A to C)
  by the PLC.

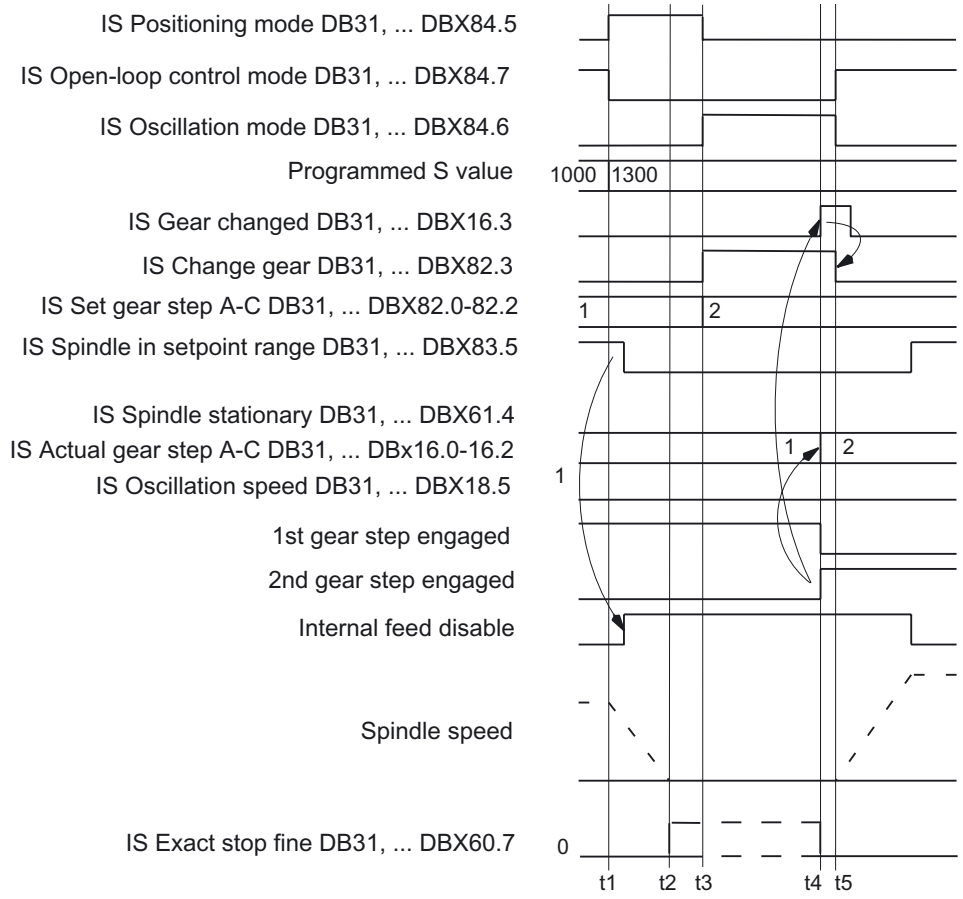
- After signal:
  DB31, ... DBX16.3 (gear step changed),
  the last movement to be active is continued, if available.

  For indirect encoders (motor encoders), the homing status is cleared:
  DB31, ... DBX60.4/5 = 0.

  The spindle is in speed control mode and NC/PLC IS:
  DB311, ... DBX84.7 (open-loop control mode)
  is output.

Gear step change at fixed position

Typical time sequence for the gear step change at fixed position:



t1 The programming of S1300 indicates a new gear step (gear step 2) to the NCK, which sets IS: Positioning mode and disables machining for the next part program block.

t2 The spindle is stationary and exact stop is signaled.

t3 Gear step change - wait time

t4 The new gear step is engaged. The PLC user transmits the new (actual) gear step to the NCK and sets IS: Gear changed.

t5 At this point, the NCK cancels the Change gears, releases the next part program block for machining and accelerates the spindle to the new S value (S1300).

Fig. 2-10 Gear step change with stationary spindle

Gear step change position MD35012

The gear step change position is defined in machine data:
MD35012 $MA_GEAR_STEP_CHANGE_POSITION
for each gear step.

Gear step 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 step 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 stationary
(e.g., after power ON), the approach is determined by machine data:
MD35350 $MA_SPIND_POSITIONING_DIR.

If an adjustable gear step change position is required, then this can be achieved by writing the machine data and by a subsequent NewConfig.
The MD value can be changed using the part program or via the MMC.

If the system is unable to reach the preset position, then alarm 22020 is signaled and the gear step 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.

Velocity

The positioning speed is taken from machine data:
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 instruction `OVRA[Sn]`.

Note

`OVRA[Sn]` is valid modally. After the gear step change, a value appropriate for the machining should be reset.

The part-program instruction `FA[Sn]` does not change the positioning speed during gear step change.

Acceleration

The acceleration vales are defined independent of the gear step in machine data:
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 step 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 step change time and secondly on the accuracy of the approach to the preset gear step change position.

Block change

The block change is stopped and the machining blocks are not started until the gear step has been changed by the PLC (DB31, ... DBX16.3).

End of gear step change

Once the gear step change has been completed, the spindle returns to open-loop control mode and will automatically change to the controller mode defined by `SPCON` or `SPCOF`.

All gear-specific limit values (min./max. speed of gear step, etc.) correspond to the check-back values of the actual gear step.

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 the gear step change at fixed position
is activated by setting:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2.

2.4 Selectable spindles

Application

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.

Functionality

Each spindle is uniquely mapped to a machine axis using a configurable number with machine data:
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX...].
This number is always applies to a spindle, regardless of the channel in which 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.

The setting data table
(SD42800 $SC_SPIND_ASSIGN_TAB[...]; Spindle number converter)
assigns each logical spindle used in the part program a physical spindle.

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.

The logical spindles can be changed by modifying the setting data:
SD42800 $SC_SPIND_ASSIGN_TAB[...].
The switchover can be initiated from the part program, from the PLC and/or from the MMC.

Note

In setting data:
SD42800 $SC_SPIND_ASSIGN_TAB[0]
contains the logical master spindle.

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 affected by the spindle mapping table are:
$P_S, $P_SDIR, $P_SMODE, $P_GWPS, $AC_SDIR, §AC_SMODE, $AC_MSNUM, $AA_S.

References:
/PGA/ Programming Guide Advanced

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 "AutoGet" function is triggered for the physical spindle or alarm 16105 "Assigned spindles do not exist" is output, depending on the configuration variant.
- If the setting data:
  SD42800 $SC_SPIND_ASSIGN_TAB[...]
  is defined by the PLC or the MMC, the channel whose table is modified must have be in the reset state and the spindle to be switched must not be used in the part program currently running.

  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, ambiguities can arise, according to the conversion table, when logical spindles are displayed on the MMC.
- Spindle conversion operates on spindles via FC 18.

Activation

Setting data:
SD42800 $SC_SPIND_ASSIGN_TAB[...]
is enabled by activating machine data:
MD20092 $MC_SPIND_ASSIGN_TAB_ENABLE=1.

Basic position SD42800

After switching on the NC in installation switch position 1 (Delete SRAM)
the setting data:
SD42800 $SC_SPIND_ASSIGN_TAB[...]
is in the basic position.

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

Assumptions: Spindle configurations:

Specifying the 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
```

Applying a machine axis in the 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 table setting
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 the
             configured No. "1" (No. of 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.
                                           Caution: 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, spindle number 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
```

2.5 Programming

2.5.1 Programming in the part program

Programming statements

Statement	Description	
SETMS	The master spindle is the spindle stored in machine data: MD20090 $MC_SPIND_DEF_MASTER_SPIND (initial setting for master spindle on channel).	
SETMS(n)	The spindle with the number (n) is the master spindle (can differ from the initial setting in machine data: MD20090 $MC_SPIND_DEF_MASTER_SPIND). The master spindle must be defined for the following functions:	
	G95	Rev. feedrate
	G96 S../G961 S...	Constant cutting speed in m/min or ft/min
	G97/G971	Cancel G96/G961 and freeze last spindle speed
	G63	Tapping with compensating chuck
	G33/G34/G35	Thread cutting
	G331/G332	Rigid tapping
	G4 S...	Dwell time in spindle revolutions
	Programming M3, M4, M5, S, SPOS, M19, SPOSA, M40, M41 to M45 and WAITS without entering the spindle number.	
	The current master spindle setting can be retained via RESET and START. The setting is made in machine data: MD20110 $MC_RESET_MODE_MASK and MD20112 $MC_START_MODE_MASK. **References:** /FB1/Descriptions of Functions, Basic Machine; Mode Group, Channel, Program Operation (K1)	
M3 M1=3	Direction of spindle rotation clockwise for master spindle. Direction of spindle rotation clockwise for spindle number 1	
M4 M2=4	Direction of spindle rotation counterclockwise for master spindle. Direction of spindle rotation counterclockwise for spindle number 2	
M5 M1=5	Spindle stop without orientation for master spindle. Spindle stop without orientation for spindle number 1	
S .... S2=	Spindle speed in rpm for master spindle. Spindle speed in rpm for spindle number 2	
SPOS =270 SPOS[n] =270	Spindle positioning for the master spindle or the spindle with number n to the position 270 degrees. The block change is only performed when the spindle is in position.	

Statement	Description
SPOSA=90 SOSSA[n]=90	Spindle positioning for the master spindle or the spindle with number n to the position 90 degrees. The block change is executed immediately. Spindle positioning continues, regardless of further part program processing, until the spindle has reached its position.
SPOS=DC(Pos), SPOS[n]=DC(Pos), SPOSA=DC(Pos), SPOSA[n]=DC(Pos)	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(Pos), SPOS[n]=ACN(Pos), SPOSA=ACN(Pos), SPOSA[n]=ACN(Pos)	The position is always approached with negative direction of motion. If necessary, the direction of motion is inverted prior to positioning.
SPOS=ACP(Pos), SPOS[n]=ACP(Pos), SPOSA=ACP(Pos), SPOSA[n]=ACP(Pos)	The position is always approached with positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.
SPOS=IC(Pos), SPOS[n]=IC(Pos), SPOSA=IC(Pos), SPOSA[n]=IC(Pos)	The travel path is specified. The direction of travel is obtained 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	Spindle positioning for the master spindle or the spindle with number n to the position entered in setting data: SD43240 $SA_M19_SPOS. The block change is only performed when the spindle is in position.
M70 M1=70	Bring spindle to standstill and activate position control, select zero parameter set, activate axis mode for the master spindle or for the spindle with number 1.
SPCON, SPCON(n), SPCON(n,m)	Spindle position control for the master spindle ON Spindle position control mode for the spindle number n ON Spindle position control mode for the spindle number n and m ON
PCOF, SPCOF(n), SPCOF(n,m)	Spindle position control for the master spindle OFF, activate speed control mode. Spindle position control for the master spindle OFF, activate speed control mode for the spindle number n as well as for the spindle number n and m.
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.
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 S2...	Programmable minimum spindle speed limitation. Programmable minimum spindle speed limitation for spindle number 2.
G26 S...., G26 Sn...	Programmable maximum spindle speed limitation. Programmable maximum spindle speed limitation for spindle number n.
LIMS=	Programmable maximum spindle speed limitation with `G96`, `G961`, `G97`.

Statement	Description
WAITS	Part program, synchronization command for master spindle. Execution of the following blocks is suspended until the spindle(s) programmed with `SPOSA` has/have reached their position(s) with exact stop fine. Waits until the spindle is at standstill after `M5`. Waits until the spindle has reached its setpoint speed after `M3`/`M4`.
WAITS(n) WAITS(n,m)	Synchronization command for spindle n. Synchronization command for spindles n and m.
FA[Sn]	Programming of positioning speed (axial feed) for spindles in [deg/min]. The value configured in machine data: MD35300 $MA_SPIND_POSCTRL_VELO is reactivated with FA[Sn]=0.
OVRA[Sn]	Programming of axial override value for spindle n in [%].
ACC[Sn]	Programming of the axial acceleration capacity of spindle n in [%].
SPI(n)	Axis functions for a spindle with `SPI(n)` with `SPI(n)` (spino) are converted into the data type AXIS according to machine data: MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[ ]. `SPI` is used if axis functions are programmed using the spindle number. The following instructions are possible with SPI: • Frame instructions with SPI: – CTRANS() – CFINE() – CMIRROR() – CSCALE() • Velocity and acceleration values for following spindles with SPI: – FA[SPI(n)] – ACC[SPI(n)] – OVRA[SPI(n)] • System variable with SPI: – $P_PFRAME[SPI(1),TR] = 2.22 frames can be written, for example. – $P_PFRAME = CTRANS (X, axis value, Y, axis value, SPI(1), axis value) – $P_PFRAME = CSCALE (X, scale, Y, scale, SPI(1), scale) – $P_PFRAME = CMIRROR (S1, Y, Z) – $P_UBFR = CTRANS (A, 10) : CFINE (19, 0.1) Detailed information about programming of SPI can be found in **References:** /PGA/ Programming Guide Advanced
M40, M1=40	Automatic gear step selection for master spindle. Automatic gear step selection for spindle number 1.
M41 to M45 M1=41 to M1=45	Select gear step 1 to 5 for master spindle. Select gear step 1 to 5 for spindle number 1.

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.

Detailed information about programming the spindle can be found in
References:
/PG/ Programming Guide Fundamentals

2.5.2 Programming via synchronized actions

M40 to M45

M functions M40 to M45 can also be programmed in synchronized actions.

Please note:

- 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.

For synchronized actions:

- 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 the first to fifth gear step according to the programming instructions in the part program.

Please note:

- 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 as M3, M4, M5 instructions).

Detailed information about programming of spindles and spindle movements from synchronized actions can be found in:
References:
/PGA/ Programming Guide Advanced
/FBSY/ Description of Functions Synchronized Actions

2.5.3 Programming of spindle controls via PLC with FC18

Automatic gear step change with FC18

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.

For detailed explanations on how to program spindle controls via the PLC with FC18, please refer to:
References:
/FB1/ Descriptions of Functions, Basic Machine; Basic PLC Program (P3)

2.5.4 Special spindle motions via PLC interface

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

Special VDI interface signals are provided to start and stop spindles outside a running part program. In this regard, the channel status and the program status need not be in the active mode. These states will occur, e.g., on RESET and in JOG mode.

The spindle concerned must have the state "Channel axis" or "Neutral axis" and must not be moved using the JOG keys or positioned by FC18 or synchronized actions. If these conditions are fulfilled, spindle jobs will be accepted via the DBB30 spindle interface.

The spindle settings are retained after a change in mode (e.g., from JOG mode to AUTOMATIC). The spindle settings (direction of rotation, speed or cutting rate) are applied to the part program at the start of the program and can be modified again using part-program instructions. In JOG mode, the spindle can be moved at the speed last programmed.

Multichannel 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.

This channel can be determined on the PLC by reading NC/PLC IS:
DB31, ... DBX68.0-68.3 (NC axis/spindle in channels A to D).

Spindle job

In order to start a job, the channel handling the spindle must be in the acceptance status. A spindle job is always started on the LowHigh edge of a DBB30 signal.

Generally, the DBB30 start signals do not have any meaning in the static status and do not prevent the spindle being programmed by FC18, synchronized actions, the part program or JOG traversing movements (e.g., when the STOP signal is statically at "1").

Conditions for the acceptance status

Outside a running part program, spindles can be started and stopped using the special VDI interface signals NCK → PLC.

For this, **the channel status** in "Interrupted" mode:
DB21, ... DBX35.6 = 1
or "Reset":
DB21, ... DBX35.7=1
and the **program status** must be in "Interrupted" mode:
DB21, ... DBX35.6 = 1
or "Aborted":
DB21, ... DBX35.4 = 1.

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 status "Channel axis" or "Neutral axis" and must not be moved by means of the JOG keys.
- When the spindle is specified, no positioning movement may be carried out by FC18 or synchronized actions.

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 assumed that the acceptance range will be indicated to the operator by the PLC program.

Spindle jobs outside the acceptance range can also be carried out using FC18 or ASUB.

SD43200 Overwrite

Setting data:
SD43200 $SA_SPIND_S
can be overwritten as follows:

- Through speed programming
- Through direct programming in the part program
- Through the MMC

Note

The setting data is written immediately and asynchronously to part-program execution.

Conditions for writing

When overwriting the setting data:
SD43200 $SA_SPIND_S
the following conditions apply:

Programming through:	Conditions for programming:
Speed programming	MD35035 $MA_SPIND_FUNCTION_MASK Bit 4 = 1 must be set.
	Constant cutting rate `G96`, `G961` must not be active.
	NC/PLC IS: DB31, ... DBX84.0 = 0 (constant cutting speed) must be set.
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.
MMC: Direct programming in the part program	Only positive values including zero can be accepted. Otherwise, a corresponding message is generated.

Spindle commands

The following basic logic functions can be specified for the spindle:

Basic logic functions:	Spindle commands:
Motion specification: Spindle stop Spindle start clockwise rotation Spindle start counterclockwise rotation Select gear step Spindle positioning	Independently of a running part program: Input from the PLC via axial VDI interface signals DB31, ... DBB30
Speed MD35035 $MA_SPIND_FUNCTION_MASK Bit 4 = 1 When movement starts the speed is read from setting data: SD43200 $SA_SPIND_S.	Speed specifications from part program or FC18 are programmed in setting data: SD43200 $SA_SPIND_S.
Setpoint speed: MD35035 $MA_SPIND_FUNCTION_MASK Bit 5 = 1 The content from setting data: SD43200 $SA_SPIND_S is used as the setpoint speed.	You can use the JOG keys to operate the spindle at the speed defined in setting data: SD43200 $SA_SPIND_S.
Constant cutting speed: MD35035 $MA_SPIND_FUNCTION_MASK Bit 8 = 1 Read from setting data: SD43202 $SA_SPIND_CONSTCUT_S.	Specifications for the constant cutting speed from part program, FC18 and synchronized actions are programmed in setting data: SD43200 $SA_SPIND_S.

Notes on machine data MD35035 $MA_SPIND_FUNCTION_MASK:

For bit 4 and bit 8, the following also applies:

- Programmed S values that are not programmed speed values are **not** written to the corresponding setting data.

  This includes S values in the case of a rotation-related dwell time (`G4`).
- If this conditions are met, spindle jobs will be accepted via the DBB30 interface (acceptance status).

Motion settings

The direction of rotation is specified in DBB30 and transmitted by the NCK input signals at the axial VDI interface (PLC -> NCK) in data bytes DB31, ... DBB30 (JobShop).

Interface signals

"Spindle stop" (equivalent to M5)	DB31, ... DBX30.0
"Spindle start clockwise" (equivalent to M3)	DB31, ... DBX30.1
"Spindle start counter-clockwise" (equivalent to M4)	DB31, ... DBX30.2
"Select gear step" (function in preparation)	DB31, ... DBX30.3 (reserved signal)
"Spindle positioning" (corresponds to M19)	DB31, ... DBX30.4

Priorities

If several DBB30 signals are set at the same time, the following order of priority is defined:

Spindle stop (M5)	1. Priority
Spindle start/Positioning mode (M3)	2. Priority
Spindle start/Positioning mode (M4)	3. Priority
Spindle start/Positioning mode (M19)	4. Priority

Acknowledgment spindle start/stop

A spindle start can be detected at the VDI interface based on traversing commands output:
DB31, ... DBX64.6 = 1 (traversing command minus)
or
DB31, ... DBX64.7 = 1 (traversing command plus).

A spindle stop is executed if NC/PLC IS:
DB31, ... DBX61.4 = 1 (axis/spindle stationary)
is output.

Direct acknowledgment signals for the DBB30 input signals are not output.

Invert M3/M4

NC/PLC interface signal:
DB31, ... DBX17.6 (invert M3/M4)
acts on NC/PLC IS:
DB31, ... DBX30.1 (spindle start CW)
and
DB31, ... DBX30.2 (spindle start CCW)
in the same way as for direction of rotation programming for M3 or M4 via part program, synchronized action, or FC18.

Speed default

Speed specifications from part program, FC18 or synchronized actions are programmed in setting data:
SD43200 $SA_SPIND_S
and always programmed from the usual sources.

Read SD43200

When movement starts the speed is read from setting data:
SD43200 $SA_SPIND_S [rpm]
with the positive edge of the start signal:
DB31, ... DBX30.1 (spindle start CW)
and
DB31, ... DBX30.2 (spindle start CCW).

Note

Rewriting of the setting data is not activated until the next positive edge of the start signals:
DB31, ... DBX30.1 (spindle start CW)
and
DB31, ... DBX30.2 (spindle start CCW)
for the current spindle speed.

The content of the setting data is only accepted with the next spindle start signal.

Gear step change and effect on speed

In the current version, no gear step change is triggered if the setpoint speed is out of the speed range of the gear step. The usual speed limitations and the speed increase to the setpoint speed are active.

Constant cutting rate setting

Constant cutting speed specifications from part program, FC18 or synchronized actions are programmed in setting data:
SD43200 $SA_SPIND_S
and always programmed from the usual sources.

Read SD43202

The constant cutting speed is read from **setting data:**
SD43202 $SA_SPIND_CONSTCUT_S [m/min] or [feet/min]
with the positive edge of the start signal:
DB31, ... DBX30.1 (spindle start CW)
and
DB31, ... DBX30.2 (spindle start CCW).

Note

Rewriting of the setting data is not activated until the next positive edge of the start signals:
DB31, ... DBX30.1 (spindle start CW)
and
DB31, ... DBX30.2 (spindle start CCW)
for the current spindle speed.

The content of the setting data is only accepted with the next spindle start signal.

Supplementary condition

To ensure that constant cutting rate settings are active, the spindle concerned must be a master spindle in the channel handling the spindle.

This condition is met when NC/PLC IS:
DB31, ... DBX84.0 = 1 (constant cutting rate active)
is set on the axial VDI interface.

Writing from the part program

When writing from the part program, the value for the constant cutting rate 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]

With `G70`, `G71` and when programming from external (MMC) the setting in machine data:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC
determines the interpretation of the programmed value.

For more detailed information about the metric/inch unit system, please refer to
References:
/FB1/ Descriptions of Functions, Basic Machine; Velocities, Setpoint/Actual-Value Systems, ClosedLoop Control (G2)

Setting via FC18 synchronized actions

If the constant cutting rate is set via FC18, the setting of bit 6 in byte to in the "Signals to concurring positioning axes" area determines how the speed value (bytes 8...11) is interpreted.

In the case of setting via synchronized actions, the feedrate type will determine how the S value is interpreted, analogously to the part program.

Reading from part program and synchronized actions

The programmed cutting rate value can be determined both in the part program and in synchronized actions by reading system variables:
$P_CONSTCUT_S
and
$AC_CONSTCUT_S.

The programmed cutting rate value can also be read via the OPI interface.

System variables

RV: Defined **r**ange of **v**alues of the two new system variables.

Description	NCK variable
Last constant cutting rate programmed RV = {0, DBL_Max}	$P_CONSTRUCT_S[n]
Current constant cutting rate RV = {0, DBL_Max}	$AC_CONSTRUCT_S[n]

Spindlespecific functions

MD35035

Machine data:
MD35035 $MA_SPIND_FUNCTION_MASK
defines spindle-specific functions from the part programs, FC18 and synchronized actions as follows:

No gear step change with DryRun, program testing and SERUPRO.	
Bit 0 = 1	Gear step change for blocks with `M40`, `M41` to `M45` or via FC18 and synchronized actions are suppressed with DryRun.
Bit 1 = 1	Gear step change for blocks with `M40`, `M41` to `M45` or via FC18 and synchronized actions are suppressed with Program Test and SERUPRO.
Bit 2 = 1	Gear step change for programmed gear step is performed after deselection of functions DryRun and SERUPRO.
Acceptance of programmed speed and cutting rate including settings.	
Bit 4 = 1	The programmed speed including speed specifications are accepted in setting data: SD43200 $SA_SPIND_S. through FC18 and synchronized actions.

Bit 5 = 1	The content of setting data: SD43200 $SA_SPIND_S serves as the setpoint speed in JOG mode. You can use the JOG keys to operate the spindle at the speed defined in SD43200. If the content is zero, other JOG speed specifications are active (see setting data SD41200 $SN_JOGSPIND_SET_VELO).
Bit 8 = 1	The programmed cutting speed including specifications for FC18 and synchronized actions are accepted in SD43202 $SA_SPIND_S.
Spindle override is also actively influenced during zero marker search with `M19`, `SPOS`, or `SPOSA`.	
Bit 12 = 1	Spindle override is effective during zero marker search with `M19`, `SPOS`, or `SPOSA`.
For additional information about machine data: MD35035 $MA_SPIND_FUNCTION_MASK please refer to: **References:** /FB1/ Descriptions of Functions, Basic Machine; Acceleration (B2)	

2.5.5 Gear step change with DryRun, program testing and SERUPRO

Part program, synchronized actions and FC18

The behavior of the gear step change from the part program, FC18 and synchronized actions for the DryRun, program testing and SERUPRO (SearchRunByProgramtest) functions can be configured using the existing machine data:
MD35035 $MA_SPIND_FUNCTION_MASK.

These functions do not generally require a gear step change and can therefore be suppressed in machine data:
MD35035 $MA_SPIND_FUNCTION_MASK
with bits 0 to 2 as follows:

Dry run feedrate (DryRun)	
Bit 0 = 0	Gear steps are activated even with the DryRun function active for part program blocks with `M40`, `M41` to `M45` or via FC18 and synchronized action programming (response).
Bit 0 = 1	Gear step change for blocks with `M40`, `M41` to `M45` or via FC18 and synchronized actions are suppressed with DryRun.
Program testing and SERUPRO	
Bit 1 = 0	Gear steps are activated even with the Program Test function active for part program blocks with `M40`, `M41` to `M45` or via FC18 and synchronized action programming (response).
Bit 1 = 1	Gear step change for blocks with `M40`, `M41` to `M45` or via FC18 and synchronized actions are suppressed with Program Test and SERUPRO.

DryRun, program testing and SERUPRO	
Bit 2 = 0	Gear step change for programmed gear step is **not** performed subsequently on REPOS after deselection of functions DryRun, Program Test and SERUPRO.
Bit 2 = 1	Gear step change for programmed gear step is performed after deselection of functions DryRun and SERUPRO if possible.

Gear step change suppression

If a gear step change is suppressed, if necessary, the interpolator will limit the programmed spindle speed to the permissible speed range of the active gear step.

NC/PLC IS:
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 step change is suppressed, no new gear step setpoint is output to the PLC with NC/PLC IS:
DB31,...DBX82.0-82.2 (gear step setpoint).

The gear step change request:
DB31,...DBX82.3 (change gear)
is also suppressed.
This ensures that no gear step change information has to be processed by the PLC program.

Determining the last active gear step

System variable:
$P_GEAR
returns the gear step 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 step from the part program, synchronized actions and operator panel interface.

The DryRun function can be deselected within a running part program. Once it has been deselected, the correct gear step 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 step has been activated. Any necessary gear step change is performed in the system REPOS started on deselection if the spindle is in speed control mode. A complete gear step change dialog takes place with the PLC and the last programmed gear step is activated.

If there is a mismatch between the gear step programmed in the part program and the actual gear step returned via the VDI interface with REPOS, no gear step change takes place.

The same applies to the SERUPRO function.

For more information about SERUPRO block search, please refer to
References:
/FB1/ Descriptions of Functions, Basic Machine; Mode Group, Channels, Program Operation Mode (K1)

Supplementary conditions

If the gear step change is suppressed, the output spindle speed moves within the speed range specified by the current gear step.

The following restrictions apply to the subsequent activation of a gear step change with REPOS:

- The gear step 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 step cannot be activated because the spindle is in position or axis mode or a link is active, alarm 22011"Channel%1 block%3 spindle% Change to programmed gear step not possible" is signaled.

Example

Gear step change in DryRun

```
                          1. Activate 1st gear step (GS) for output state;
N00 M3 S1000 M41                    ;   1. GS is selected
M0                                  ;  Part program stops


                          PI service: Activate dry run feedrate (DryRun);
                                    ;  (Configuring)
N10 M42                             ;   2. 2nd GS requested, no gear step 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 step change to the 2nd gear step
                                       takes place
N20 G1 Z100 F1000                   ;
...                                 ;
N99 M30                             ;  Part program end
```

2.5.6 Programming from external (PLC, MMC)

SD43300 and SD42600

The revolutional feedrate behavior 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 feedrate 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)`.

2.6 Spindle monitoring

2.6.1 Speed ranges

Permissible speed ranges for the spindle

The spindle monitoring functions and the currently active functions (`G94`, `G95`, `G96`, `G961`, `G97`, `G971`, `G33`, `G34`, `G35`, `G331`, `G332`, etc.) define the permissible speed ranges of the spindle.

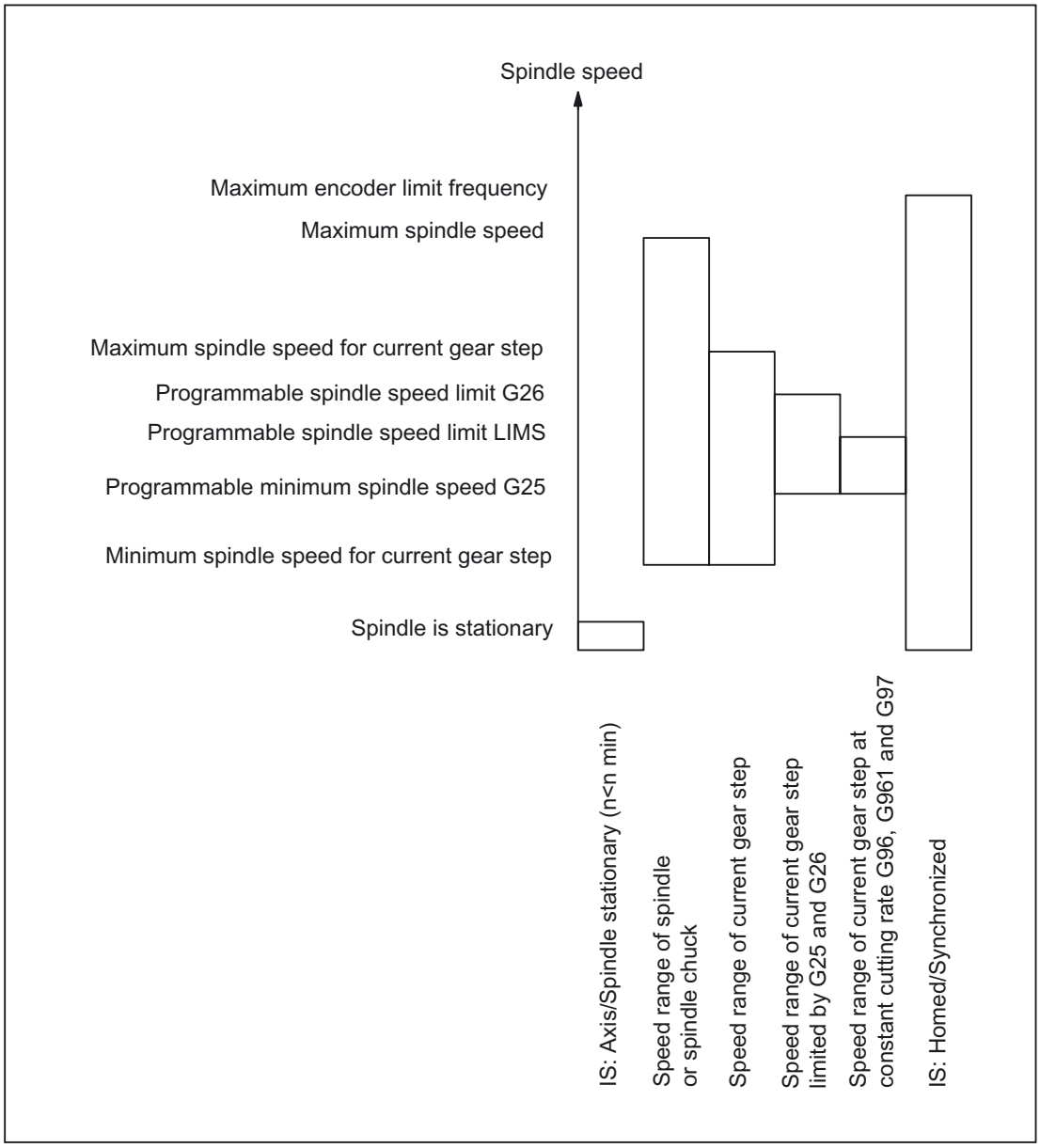


Fig. 2-11 Ranges of spindle monitoring functions/speeds

2.6.2 Axis/spindle stationary (n < n min)

Spindle monitoring with stationary axis/spindle

Only if the axis/spindle is stationary, i.e., the actual spindle speed falls below a specifiable value in machine data:
MD36060 $MA_STANDSTILL_VELO_TOL (maximum velocity/speed "Axis/spindle stationary")
and no longer generates any setpoints, is it possible to perform certain functions on the machine, such as tool change, open machine doors, path feed enable, etc.

If the spindle is stationary,

- NC/PLC IS:
  DB31, ... DBX61.4 (axis/spindle stationary)
  is set to 1.
- The next machining block is released
  and machine data:
  MD35510 $MA_SPIND_STOPPED_AT_IPO_START
  is set to 0.

  Exception:
  MD35510 $MA_SPIND_STOPPED_AT_IPO_START =1

  Path interpolation is not influenced, the traveled path axis is not stopped.

Monitoring is effective in all spindle modes and in axis mode.

2.6.3 Spindle in setpoint range

Function

"Spindle in setpoint range" spindle monitoring checks whether:

- The programmed spindle speed is reached.
- The spindle is stationary (NC/PLC IS:
  DB31, ... DBX61.4 (axis/spindle stationary)).
- The spindle is still in the acceleration or deceleration phase.

In spindle control mode, the setpoint speed
(programmed speed * spindle offset including active limitations)
is compared with the actual speed.

If the actual speed deviates from machine data:
MD35150 $MA_SPIND_DES_VELO_TOL
by more than the spindle speed tolerance:

- Axial NC/PLC IS:
  DB31, ... DBX83.5 (spindle in setpoint range)
  is set to 0.
- The next machining block is not enabled.
- Machine data:
  MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START
  is set to 1.

  Exception:
  MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 0
  Path interpolation is not influenced, the traveled path axis is not stopped.

Tolerance range for setpoint speed

MD35150 $MA_SPIND_DES_VELO_TOL = 0.1

The spindle actual speed may deviate +/- 10% from the setpoint speed.

The spindle setpoint speed is derived from the programmed speed taking the current limits into account.

Any limitation or increase of the programmed speed
is indicated either by axial NC/PLC IS:
DB31, ... DBX83.1 (set speed limited)
or
DB31, ... DBX83.2 S (set speed increased)
and does **not** prevent the axis reaching the speed tolerance range.

If the spindle lies inside the tolerance range,
axial NC/PLC IS:
DB31, ... DBX83.5 (spindle in setpoint range)
is set to 1 at the VDI interface.

Special case:
If the speed tolerance is set to 0, axial NC/PLC IS:
DB31, ... DBX83.5 (spindle in 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 violated, e.g., due to an overload, the path movement is not automatically brought to a standstill.

2.6.4 Minimum/maximum gear step speed

Minimum speed

The minimum gear step speed is entered in machine data:
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT.

This setpoint speed cannot be undershot by programming an S value, which is too small.

NC/PLC IS:
DB31, ... DBX83.2 (setpoint speed increased) (programmed speed too low) is set.

The minimum gear step speed is effective only in speed mode and can only be undershot by:

- Spindle override 0%
- `M5`
- `S0`
- NC/PLC IS DB31, ... DBX4.3 (spindle stop)
- Withdraw NC/PLC IS:
  DB31, ... DBX2.1 (controller enable).
- NC/PLC IS DB21, ... DBX7.7 (reset)
- NC/PLC IS DB31, ... DBX2.2 (delete distance-to-go/spindle reset)
- NC/PLC IS DB31, ... DBX18.5 (oscillation speed)
- NC/PLC IS DB21, ... DBX7.4 (NC STOP axes plus spindles)
- NC/PLC IS DB31, ... DBX1.3 (axis/spindle disable)
- NC/PLC IS DB31, ... DBX16.7 (delete S value)

Maximum speed

The maximum gear step speed is entered in machine data:
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT.

When the gear step is engaged, this setpoint speed cannot be exceeded.

When the programmed spindle speed is limited, NC/PLC IS:
DB31, ... DBX83.1 (setpoint speed limited) (programmed speed too high) is set.

2.6.5 Maximum encoder 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 (homed/synchronized 1)
or
DB31, ... DBX60.5 (homed/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 undershot

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 points to be noted

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

2.6.6 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.

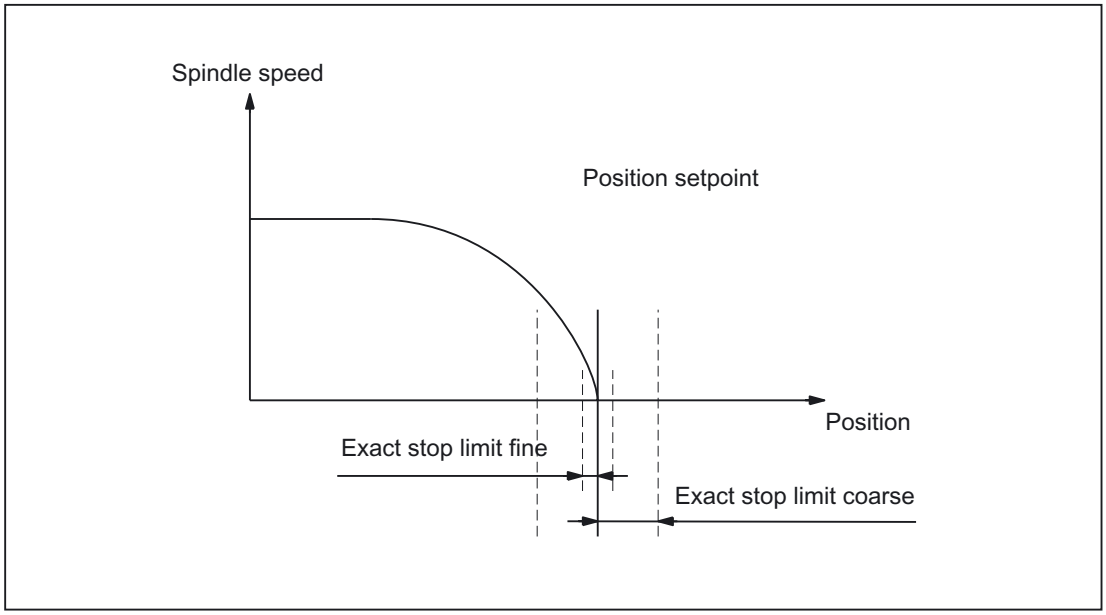


Fig. 2-12 Exact stop zones of a spindle

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.

Supplementary Conditions

3

3.1 Supplementary conditions

None.

Examples

4

4.1 Example of 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)

Data Lists

5

5.1 Machine data

5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10192	GEAR_CHANGE_WAIT_TIME	Wait time for acknowledgment of a gear step 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

5.1.2 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 auxiliary function "M19" to the VDI interface
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET

5.1.3 Axis/spindle-specific 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, distance-coded
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	Gear step change options expandable at fixed position
35012	GEAR_STEP_CHANGE_POSITION	Gear step change position
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
35100	SPIND_VELO_LIMIT	Maximum spindle speed
35110	GEAR_STEP_MAX_VELO[n]	Maximum speed for gear change
35120	GEAR_STEP_MIN_VELO[n]	Minimum speed for gear change
35130	GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum speed of gear step
35140	GEAR_STEP_MIN_VELO_LIMIT[n]	Minimum speed of gear step

Number	Identifier: $MA_	Description
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
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 non-synchronized 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
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.

5.2 Setting data

5.2.1 Channel-specific 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`)

5.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43200	SPIND_S	Specification of the spindle speed
43202	SPIND_CONSTCUT_S	Specification of the constant cutting rate for the master spindle
43210	SPIND_MIN_VELO_G25	Progr. Spindle speed limitation `G25`
43220	SPIND_MAX_VELO_G26	Progr. Spindle speed limitation `G26`
43230	SPIND_MAX_VELO_LIMS	Progr. spindle speed limitation `G96/G961`
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

5.3 Signals

5.3.1 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	0	Feed override
31, ...	1.7	Override active
31, ...	1.6	Position measuring system 2
31, ...	1.5	Position measuring system 1
31, ...	1.4	Follow-up mode
31, ...	1.3	Axis/spindle disable
31, ...	2.2	Spindle reset/delete distance-to-go
31, ...	2.1	Servo enable
31, ...	3.6	Velocity/spindle speed limitation
31, ...	16.7	Delete S value
31, ...	16.5	Resynchronize spindle 2
31, ...	16.4	Resynchronize spindle 1
31, ...	16.3	Gear changed
31, ...	16.2-16.0	Actual gear step A to C
31, ...	17.6	Invert M3/M4
31, ...	17.5	Resynchronize spindle during positioning 2
31, ...	17.4	Resynchronize spindle during positioning 1
31, ...	18.7	Direction of rotation setpoint left

DB number	Byte.Bit	Description
31, ...	18.6	Direction of rotation setpoint right
31, ...	18.5	Oscillation speed
31, ...	18.4	Oscillation via PLC
31, ...	19.7 - 19.0	Spindle offset H - A
31, ...	30.4	Spindle positioning
31, ...	30.3	Select gear step
31, ...	30.2	Spindle start CCW
31, ...	30.1	Spindle start CW
31, ...	30.0	Spindle stop
31, ...	60.7	Position reached with exact stop fine
31, ...	60.6	Position reached with exact stop coarse
31, ...	60.5	Homed/synchronized 2
31, ...	60.4	Homed/synchronized 1
31, ...	60.3	Encoder limit frequency exceeded 2
31, ...	60.2	Encoder limit frequency exceeded 1
31, ...	60.0	Axis/no spindle
31, ...	61.7	Current controller active
31, ...	61.6	Speed control loop active
31, ...	61.5	Position controller active
31, ...	61.4	Axis/spindle stationary ($n < n_{min}$)
31, ...	82.3	Change gear
31, ...	82.2-82.0	Set gear step AC
31, ...	83.7	Actual direction of rotation clockwise
31, ...	83.5	Spindle in setpoint range
31, ...	83.2	Setpoint speed increased
31, ...	83.1	Setpoint speed limited
31, ...	83.0	Speed limit exceeded

5.3.2 Signals from axis/spindle

DB number	Byte.Bit	Description
31, ...	84.7	Active spindle control mode
31, ...	84.6	Active spindle mode oscillation mode
31, ...	84.5	Active spindle positioning mode
31, ...	84.3	Rigid tapping active
31, ...	86 and 87	M function for spindle
31, ...	88-91	S function for spindle

Index

$

$AA_S[n], 2-5
$AC_SGEAR, 2-73
$P_GEAR, 2-73

C

Constant cutting rate setting, 2-69

D

DB 31, ...
- DBB16-19, 2-23
- DBB68ff., 2-63
- DBB82-91, 2-23
- DBX1.3, 2-79
- DBX1.5, 2-29
- DBX1.6, 2-29
- DBX16.0-16.2, 2-6, 2-33, 2-35, 2-38, 2-39, 2-40, 2-47, 2-51
- DBX16.3, 2-2, 2-6, 2-35, 2-39, 2-40, 2-46, 2-47, 2-48, 2-51, 2-54
- DBX16.4, 2-30
- DBX16.5, 2-30
- DBX16.7, 2-79
- DBX17.0, 2-53
- DBX17.4, 2-28
- DBX17.4/17.5, 2-50
- DBX17.5, 2-28
- DBX17.6, 2-68
- DBX18.4, 2-44, 2-46, 2-51
- DBX18.5, 2-6, 2-35, 2-44, 2-45, 2-46, 2-47, 2-51, 2-79
- DBX18.6, 2-46
- DBX18.7, 2-46
- DBX2.1, 2-51, 2-79
- DBX2.2, 2-5, 2-21, 2-39, 2-79
- DBX20.1, 2-40
- DBX30.0, 2-68
- DBX30.1, 2-68
- DBX30.2, 2-69, 2-70
- DBX30.3, 2-68
- DBX30.4, 2-68
- DBX4.3, 2-46, 2-79
- DBX60.0, 2-23
- DBX60.2, 2-80
- DBX60.3, 2-80
- DBX60.4, 2-80
- DBX60.4/60.5, 2-40, 2-47, 2-50, 2-51
- DBX60.5, 2-80
- DBX60.6, 2-9, 2-17, 2-20, 2-21, 2-82
- DBX60.7, 2-17, 2-20, 2-21, 2-54, 2-82
- DBX61.4, 2-5, 2-6, 2-35, 2-68, 2-77
- DBX61.5, 2-4, 2-47, 2-51
- DBX64.6, 2-68
- DBX64.7, 2-68
- DBX68.0-68.3, 2-64
- DBX7.7, 2-79
- DBX82.0-82.2, 2-33, 2-35, 2-36, 2-37, 2-38, 2-43, 2-47, 2-50, 2-53
- DBX82.3, 2-35, 2-36, 2-37, 2-44, 2-47, 2-48, 2-50, 2-53
- DBX83.1, 2-6, 2-16, 2-36, 2-37, 2-78, 2-80
- DBX83.2, 2-78, 2-79
- DBX83.5, 2-6, 2-10, 2-78
- DBX84.0, 2-66, 2-70
- DBX84.5, 2-50, 2-54
- DBX84.6, 2-2, 2-44, 2-47, 2-50, 2-53, 2-54
- DBX84.7, 2-4, 2-35, 2-51

DB21, ...
- DBB194, 2-63
- DBB202, 2-63
- DBB68ff., 2-63
- DBX35.4, 2-65
- DBX35.6, 2-65
- DBX35.7, 2-65
- DBX6.1, 2-38
- DBX7.4, 2-39, 2-79
- DBX7.7, 2-22

DB2x
- DBX6.1, 2-10

DB31, ...
- DBX30.1, 2-69, 2-70
- DBX30.2, 2-68

DB31,...
- DBX82.0-82.2, 2-73
- DBX82.3, 2-73

DBX83.1, 2-73
DBX83.2, 2-73
DB31...
DBX61.4, 2-11
DBX83.5, 2-11, 2-12
Dry run feedrate (DryRun), 2-72

G

Gear step change from NC, 2-31
Gear steps, 2-30

I

Intermediate gear, 2-41
Invert M3/M4, 2-68

M

Manual specification of a gear step, 2-38
MD10192, 2-42
MD10240, 2-70
MD11410, 2-35
MD20070, 2-58
MD20090, 2-60
MD20092, 2-57, 2-59
MD20094, 2-22, 2-23
MD20110, 2-60
MD20112, 2-60
MD20850, 2-8, 2-12
MD22000, 2-9
MD22010, 2-9
MD22020, 2-9
MD22030, 2-9
MD31040, 2-17, 2-51
MD31044, 2-41
MD31050, 2-23, 2-47
MD31060, 2-23, 2-40
MD31064, 2-41
MD31066, 2-41
MD31122, 2-28
MD31123, 2-28
MD32000, 2-80
MD32200, 2-23
MD32452, 2-23
MD32610, 2-23
MD32620, 2-25
MD32800, 2-23
MD32810, 2-23
MD32910, 2-23
MD34040, 2-28
MD34060, 2-19
MD34080, 2-28
MD34090, 2-28
MD34100, 2-28
MD34200, 2-28, 2-29
MD35000, 2-56, 2-58, 2-62
MD35010, 2-31, 2-44, 2-47, 2-50, 2-55
MD35012, 2-34, 2-50, 2-53
MD35020, 2-25
MD35030, 2-25
MD35035, 2-66, 2-67, 2-71, 2-72
MD35040, 2-5, 2-21
MD3510, 2-54
MD35110, 2-34
MD35120, 2-34
MD35130, 2-34, 2-37, 2-79
MD35140, 2-34, 2-79
MD35150, 2-10, 2-11, 2-12, 2-78
MD35200, 2-4, 2-15, 2-18, 2-21, 2-34, 2-54
MD35210, 2-4, 2-15, 2-16, 2-19, 2-20, 2-21, 2-34, 2-54
MD35300, 2-13, 2-14, 2-15, 2-16, 2-19, 2-20, 2-53, 2-62
MD35310, 2-50, 2-53
MD35350, 2-18, 2-53
MD35400, 2-45, 2-46
MD35410, 2-44, 2-46
MD35430, 2-45
MD35440, 2-45
MD35450, 2-45
MD35500, 2-6, 2-10, 2-78
MD35510, 2-10, 2-77
MD35590, 2-32, 2-34
MD36000, 2-17, 2-20, 2-21, 2-81, 2-82
MD36010, 2-17, 2-20, 2-21, 2-81, 2-82
MD36012, 2-23
MD36060, 2-5, 2-11, 2-77
MD36200, 2-23
MD36302, 2-14
Motion settings, 2-68

N

Nonacknowledged gear step change, 2-42

P

Program testing and SERUPRO, 2-72

S

SD41200, 2-72
SD42600, 2-75
SD42800, 2-56, 2-57, 2-59

SD43200, 2-66, 2-67, 2-69, 2-71
SD43202, 2-67, 2-70
SD43240, 2-8, 2-9, 2-61
SD43250, 2-8, 2-9
SD43300, 2-75
Special spindle interface, 2-64
Specify gear step, 2-31
Specify gear step from the PLC, 2-34, 2-38
Speed default, 2-69
SPI(n, 2-62
Spindle commands, 2-67
Spindle job, 2-65
Spindlespecific functions, 2-71
SPOS [n], 2-7
Star/delta switchover with FC17, 2-40

T

Tolerance range for setpoint speed, 2-78

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Feeds (V1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

1.1 Brief description 1-1

2 Detailed Description 2-1

2.1 Path feedrate F 2-1
2.1.1 General 2-1
2.1.2 Type of feedrate G93, G94, G95 2-3
2.1.3 Type of feedrate G96, G961, G97, G971 2-5
2.1.4 Feedrate with G33, G34, G35 (thread cutting) 2-9
2.1.4.1 General 2-9
2.1.4.2 Programmable run-in and run-out path for G33, G34 and G35 2-11
2.1.4.3 Linear progressive/degressive thread-lead change with G34 and G35 2-13
2.1.4.4 Stop for thread cutting 2-16
2.1.5 Feedrate for G331/G332, rigid tapping 2-20
2.1.6 Feedrate for G63, tapping with compensation chuck 2-21

2.2 Feedrate FA for positioning axes 2-21

2.3 Feedrate control 2-23
2.3.1 General 2-23
2.3.2 Feedrate disable and feedrate/spindle stop 2-24
2.3.3 Feedrate override on machine control panel 2-25
2.3.4 Programmable feedrate override 2-30
2.3.5 Dry run feedrate 2-31
2.3.6 Multiple feedrate values in one block 2-33
2.3.7 Fixed feedrate values (840D, 810D) 2-38
2.3.8 Feedrate for chamfer/rounding FRC, FRCM 2-40
2.3.9 Non-modal feedrate FB 2-41
2.3.10 Programmable single-axis dynamic response 2-42

3 Supplementary Conditions 3-1

3.1 General constraints 3-1

3.2 Supplementary conditions for feedrate programming 3-1

4 Examples 4-1

4.1 Feedrate programming for chamfer/rounding FRC, FRCM 4-1

5 Data Lists 5-1

5.1 Machine data 5-1
5.1.1 NC-specific machine data 5-1
5.1.2 Channel-specific machine data 5-2
5.1.3 Axis/spindle-specific machine data 5-2

5.2 Setting data 5-3
5.2.1 Channel-specific setting data 5-3
5.2.2 Axis/spindle-specific setting data 5-3

5.3 Signals 5-4
5.3.1 Signals to channel 5-4
5.3.2 Signals from channel 5-4
5.3.3 Signals to axis/spindle 5-5

Index Index-1

Tables

Table 2-1 Input byte for "Multiple feedrates in one block" 2-33

Brief Description

1

1.1 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 commands).

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 with thread cutting (G33, G34, G35)
- Feedrate for tapping with compensating chuck (G63)
- Feedrate for rigid tapping (G331, G332)
- Feedrate for chamfer/rounding FRC, FRCM
- Non-modal feedrate FB

Programmable run-in/run-out path for G33

The thread run-in and run-out paths can be programmed. The thread axis is accelerated and decelerated within the defined distance.

Note

If a very short distance is specified, the axis can be overloaded.

Axis assignment for feedrates

Feedrates can be assigned variably to axes according to technological requirements.

The following assignments are supported:

- Separate feedrates for working plane and infeed axis
- Variable axis assignment for path feedrate
- Feedrate for positioning axes

Feedrate control

The programmed feedrate can be modified for adaptation to changes in technological conditions during machining or for test purposes.

- Via the machine control panel
- Via the operator panel
- Via the PLC
- Via a program command

Feedrate interpolation

To permit flexible definition of the feedrate characteristic, the feedrate programming according to DIN 66205 has been extended by linear and cubic characteristics.

The cubic profiles can be programmed directly or as interpolating spline.

You can program the following feedrate profiles:

- `FNORM`

  Behavior as per 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 positioned from the current value at the start of the block to the end of the block linearly across the path distance and applies as a modal value thereafter.

- `FCUB`

  The `F` values programmed non-modally (relative to the end of the block) are connected by a spline. The spline starts and ends at a tangent to the previous or subsequent define feedrate. If the `F` address is missing in a block, the last `F` value programmed is used instead.

- `FPO`

  The `F` address [syntax: `F=FPO (...,...,...)`] designates the feedrate response over a polynomial from the current value to the end of the block in which it has been programmed. The end value then applies as a modal value.

Machine data:
MD20172 $MC_COMPRESS_VELO_TOL
supports the definition of a tolerance for the path feedrate, if FLIN and FCUB are used in conjunction with compression COMPON.

For more information about programmable feedrate paths, please see
References:
/PGA/ Programming Guide, Advanced

Feedrate for chamfer/rounding FRC, FRCM

The machining conditions can change significantly during surface transitions to chamfer/rounding. The chamfer/rounding contour elements therefore need their own optimized feedrate values in order to achieve the required surface finish.

You can program the feedrate for the chamfer/rounding with FRC (non-modal) or FRCM (modal).

Non-modal feedrate FB

A separate feedrate can be specified for a single block with the FB command. During this block the previously active path feedrate is overwritten, after the block the previously active modal path feedrate becomes active once more.

Programmable single-axis dynamic response

The dynamic response of single axes can be changed directly via the programming:

- Percentage acceleration correction (ACC) in the part program and in synchronized actions
- Programmable movement end criterion: FINEA (Exact stop fine), COARSEA (Exact stop coarse), IPOENDA (Interpolator stop) in the part program and in synchronized actions
- Programmable servo parameter set (SCPARA) in the part program and in synchronized actions

Detailed Description 2

2.1 Path feedrate F

2.1.1 General

Path feedrate F

The path feedrate represents the geometrical total of the velocity components in the participating axes. It is therefore generated from the individual motions of the interpolating axes.

The standard setting uses the axial velocities 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 the program until a new F value or a new type of feedrate is programmed.

Value range for path feedrate F

References:
/PA/ Programming Guide, Fundamentals
/FB/ Description of Functions, Basic Machine; Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)

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).

The related interface signals (modification signal, F value) are described in:
References:
/FB/ Description of Functions, Basic Machine, Auxiliary Function Output to PLC (H2)

Feedrate with transition circle

For more information, please see:
References:
PA/ Programming Guide, 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 commands 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:
/PA/ Programming Guide, Fundamentals

Maximum tool path velocity

The maximum path velocity is the result of the maximum velocities of the linear or rotary axes involved (MD32000 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 MAX_AX_VELO limitation.

Limit velocity for path axes

In addition, the FL[x]=... command can be used to program a limit velocity for path axes (geometry and synchronized axes).

References:
/PA/ Programming Guide, Fundamentals

This enables separate feedrates to be programmed for the working plane and infeed axis. This means that a feedrate is specified for both 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:

```
. .. .. 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
```

2.1.2 Type of feedrate G93, G94, G95

Effectivity

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
/FB/Description of Functions, Extended Functions; Jog With/Without Handwheel (H1)

Reciprocal feedrate (G93) (applies only to 840D/810D)

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:

$$F = \frac{v}{s} \;;\; F\ [1/min]$$

Where F: Inverse-time feedrate
v: Required path velocity in mm/min or inch/min
s: Path length in mm/inch

Programming example:

```
N10 G1 G93X100 Y200 F2    ; The programmed path is traversed in 0.5 min.
```

Note

G93may 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:

$$V = n * F$$

Where V: Path velocity
n: Master spindle speed
F: Programmed revolutional feedrate

Note

The programmed F value is deleted when the system switches between the feedrate types G93, G94 and G95 .

In JOG mode the behavior of the axis/spindle also depends on the setting of setting data: SD41100 JOG_REV_IS_ACTIVE (revolutional feedrate when JOG active).

- If this setting data is active, an axis/spindle is always moved with revolutional feedrate machine data:
  MD32050 JOG_REV_VELO (revolutional feedrate with JOG)
  or
  MD32040 JOG_REV_VELO_RAPID
  (revolutional feedrate with JOG with rapid traverse overlay)
  depending on the master spindle.

- If the setting data is not active, the behavior of the axis/spindle depends on the setting data:
  SD43300 ASSIGN_FEED_PER_REV_SOURCE
  (revolutional feedrate for positioning axes/spindles).
- If the setting data is not active, the behavior of a geometry axis on which a frame with rotation is effective depends on the channel-specific setting data:
  SD42600 JOG_FEED_PER_REV_SOURCE
  (in JOG mode, revolutional feedrate for geometry axes on which a frame with rotation acts).

"Revolutional feedrate active" DB31, ... DBX62.2

A programmed, active revolutional feedrate (G95) is displayed using this interface signal.

Alarms

- If no F value is programmed, alarm 10860 "No feedrate programmed" is output.
  The alarm is not generated with G0 blocks.
- If a negative path velocity is programmed, alarm 14800 "Programmed path velocity smaller than or equal to zero" is output.
- If a revolutional feedrate (G95) is programmed but no master spindle has been defined, alarm 10810 "No master spindle defined" is output.

2.1.3 Type of feedrate G96, G961, 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. When the command is reselected, a new cutting rate may be entered.

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.

Based on the programmed cutting rate S_{Speed} , either (S_{G96}) or (S_{G961}) and the part-oriented actual value of the transverse axis (radius r), the control system uses the following formula to determine the spindle speed:

$$n = \frac{S_{Speed}}{2 * \pi * r}$$

π = Circle constant

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 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 DIAMETER_AX_DEF
(geometry axis functioning as transverse axis).

The function G96, G961 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 movements 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 when G96, G961 and G0 blocks are active can be defined in channel-specific 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

The programming command LIMS can be used to define a maximum spindle speed for the "Constant cutting rate" function in setting data:
SD43230 SPIND_MAX_VELO_LIMS
(spindle speed limiting for G96/G961)
and in the part program (for the master spindle).
The last value changed (LIMS or SD) is active.
LIMS acts on G96, G961, G97, and can be defined for up to four speed limitations in a single block in the part program. Spindle number Sn=1, 2, 3, or 4 of the master spindle that is possible in the particular instance can be programmed in part progam instruction LM[Sn].

Note

When the block is processed, all programmed values are transferred to setting data SD43230 SPIND_MAX_VELO_LIMS.

Depending on machine data:
MD10710 PROG_SD_RESET_SAVE_TAB[n] (setting data to be updated),
the speed limitation written with LIMS will remain stored after the control system has been switched off.
When G96, G961, G97 are reactivated, this spindle speed limitation is also activated.

The maximum permissible spindle speed set in G26 or in setting data:
SD43220 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 SPIND_MAX_VELO_G26) to be exceeded, the "Programmed speed too high" interface signal (DBX83.1) is set.

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 in setting data:
SD43210 SPIND_MIN_VELO_G25 (minimum spindle speed)
and for each gear stage with machine data:
MD35140 GEAR_STEP_MIN_VELO_LIMIT
(minimum speed of gear stage).

The minimum spindle speed can be changed in the part program with G25. In the event of incorrect programming that would mean one of the spindle limits (G25 or SPIND_MAX_VELO_G25) is not reached, the "Set speed too low" interface signal (DB31, ... DBX83.2) is set.

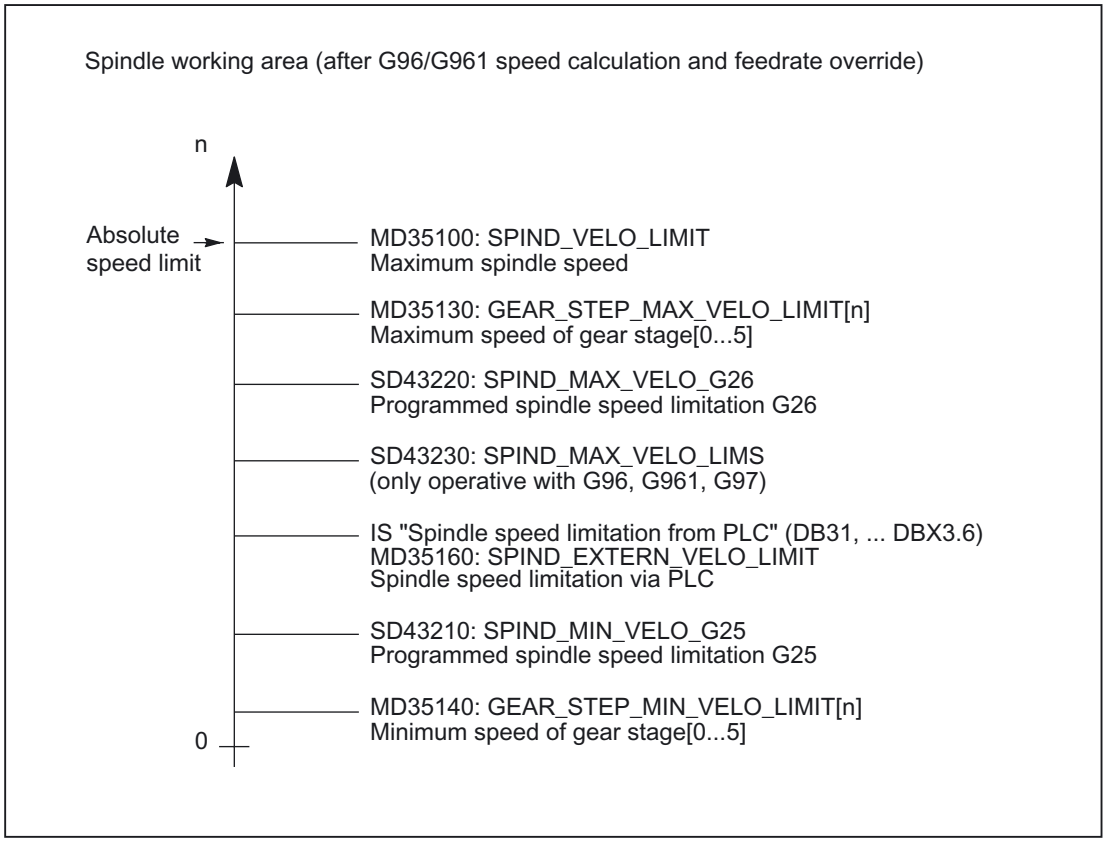


Fig. 2-1 Spindle speed limitations

The various spindle speed limits are illustrated in the figure above. For more information about this and the way in which setting data work, please see:
References:
/FB1/ Description of Functions, Basic Machine; Spindles (S1);
Chapter: Spindle Monitoring Functions, Setting Data

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

- If no F value is programmed, alarm 10860 "No feedrate programmed" is output. The alarm is not generated with G0 blocks.
- If a negative path velocity is programmed, alarm 14800 "Programmed path velocity smaller than or equal to zero" is output.
- If a transverse axis has not been defined in machine data:
  MD20100 DIAMETER_AX_DEF (geometry axes functioning as transverse axes)
  when G96, G961 are active, 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.

2.1.4 Feedrate with G33, G34, G35 (thread cutting)

2.1.4.1 General

Application of G33

The function G33 can be used to machine threads with constant lead of the following type:

References:
/PA/ Programming Guide, Fundamentals
/PAZ/ Programming Guide, Cycles

Speed S, feedrate F, thread lead

A revolutional feedrate [mm/revolution] is used for G33 threads. The revolutional feedrate is defined by programming the thread lead increase [mm/revolution].

The speed of the axes for the thread length is calculated from the programmed spindle speed S and the thread lead.

Feedrate F [mm/min] = speed S [rev/min] * thread lead [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 in setting data:
SD43210 SPIND_MIN_VELO_G25 (minimum spindle speed)
and for each gear stage with machine data:
MD35140 GEAR_STEP_MIN_VELO_LIMIT
(minimum speed for gear stage change).1

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 subsequent G33 blocks and the first subsequent non G33 block are traversed like a single 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 lead. 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 SUPPRESS_ALARM_MASK
if the ROT instruction is used specifically in the application.

All other frames are accepted by the NC without alarm. Attention is drawn to the lead-changing effect of SCALE.

2.1.4.2 Programmable run-in and run-out path for G33, G34 and G35

Application

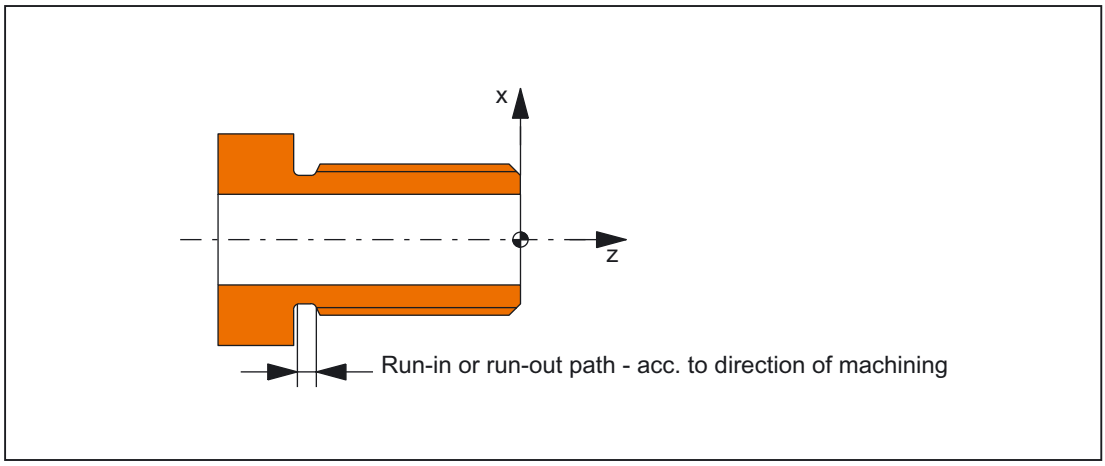


Fig. 2-2 Run-in or run-out path too short

- **Short run-in path**

  Due to the collar on the thread run-in, little room is left for the tool (T) start ramp. This must therefore be set shorter via `DITS`.

- **Short run-out path**

  Due to the collar on the thread run-out, little room is left for the tool deceleration ramp, leading to the risk of collision between the workpiece and the tool edge. The tool braking ramp can be set shorter via `DITE`. Due to the inertia of the mechanical system, however, a collision can nevertheless occur.

  Remedy: Program a shorter thread, reduce the spindle speed.

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.

Activation

The `DITS` and `DITE` functions are always active during thread cutting.

Example:

```
N...
N59 G90 G0 Z100 X10 SOFT M3 S500
N60 G33 Z50 K5 SF=180 DITS=1 DITE=3            ; Start of smoothing with Z=53
N61 G0 X20
```

SD42010

Only paths, and not positions, are programmed with DITS and DITE.

Setting data:
SD42010 THREAD_RAMP_DISP[0,1]
corresponds with the part program instructions and defines the following acceleration response of the axis during thread cutting:

- **SD42010 ≤ 0 to –1**

  Identical with previous response.

  Starting/braking of the feedrate axis at configured acceleration rate.

  Jerk according to current BRISK/SOFT programming.

- **SD42010 = 0**

  Abrupt starting/braking of the feedrate axis on thread cutting.

- **SD42010 ≥ 0**

  The thread run-up/deceleration distance is specified.

  In order to avoid technology alarm 22280, the acceleration limits of the axis must be taken into account for very small run-in/run-out paths.

Note

DITE acts at the end of the thread as a rounding clearance. This achieves a smooth change in the axis movement.

Compatibility

Machine data:
MD20650 THREAD_START_IS_HARD
is omitted and replaced by:
SD42010 THREAD_RAMP_DISP[0]
or
SD42010 THREAD_RAMP_DISP[1].

The response of the new setting data at:
SD42010 THREAD_RAMP_DISP[0] = 0
or
SD42010 THREAD_RAMP_DISP[1] = 0
is identical with the previous machine data setting:
MD20650 THREAD_START_IS_HARD = 1.

The response at:
SD42010 THREAD_RAMP_DISP[0] = -1
or
SD42010 THREAD_RAMP_DISP[1] = -1
is identical with the previous machine data setting:
MD20650 THREAD_START_IS_HARD = 0
(default setting).

Constraints

When a block is loaded into the interpolator with the DITS and/or DITE command, the path programmed under DITS is written to setting data:
SD42010 THREAD_RAMP_DISP[0]
and the path programmed under DITE is written to setting data:
SD42010 THREAD_RAMP_DISP[1].

The programmed run-in path is handled according to the current setting (inches, metric).

If no run-in/deceleration path is programmed before or in the first thread block, it is determined by the current content of setting data:
SD42010 THREAD_RAMP_DISP[0,1].

A RESET sets setting data:
SD42010 THREAD_RAMP_DISP[0,1]
to the value "-1".

2.1.4.3 Linear progressive/degressive thread-lead change with G34 and G35

Application G34, G35

The functions can be used to produce self-shearing threads.

Functionality

The thread lead increase (G34) defines the numerical increase in the lead value. A larger pitch results in a larger distance between the threads on the workpieces. The velocity of the thread axis thus increases assuming that the spindle speed is constant.
The opposite naturally applies to decreasing thread lead (G35).

The following definitions have been made for the thread-lead changes with respect to the new G codes:

- G34: Increase in thread lead corresponds to progressive change.
- G35: Decrease in thread lead corresponds to degressive change.

Both functions G34 and G35 imply the functionality of G33 and additionally provide the option of programming an absolute lead change value for the thread under F. If the start and end lead of a thread is known, the thread-lead change can be determined using the following equation:

$$F = \frac{|k_e^2 - k_a^2|}{2 * I_G}$$

The identifiers have the following meanings:

F: The thread-lead change to be programmed [mm/rev^2]

k_e: Thread lead of axis target point coordinate, thread axis [mm/rev]

k_a: Initial thread lead (programmed under I, J or K) [mm/rev]

I_G: Thread length [mm]

The absolute value of F must be applied to G34 or G35 according to the desired lead increase or lead decrease.

When the thread length I_G, lead change F and initial lead k_a are known, the lead increase at the end of block k_e can be determined as follows by modifying the formula:

- G34 (progressive lead):

$$k_e = \sqrt{k_a^2 + F * 2 * I_G}$$

- G35 (degressive lead):

$$k_e = \sqrt{k_a^2 - F * 2 * I_G}$$

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.

Sample program

Thread cutting G33 with degressive thread lead G35

```
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 lead decrease 17.045455 mm/rev2

                                      ; Thread lead 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                             ;
```

Suppress special alarms

Any lead 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 lead increase too high" or alarm 10605 "Thread lead decrease too high" are output if bit 10 in machine data:
MD11410 SUPPRESS_ALARM_MASK
is not set.

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.

The output of the alarm generated during monitoring (10604 or 10605) can be suppressed by setting bit 10 in machine data:
MD11410 SUPPRESS_ALARM_MASK.

Block preparation continues as normal when alarms are suppressed.

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.

In these cases,
alarm 22270 "Maximum velocity of thread axis reached"
or
alarm 22275 "Zero velocity of thread axis reached"
is output.

Alarms

Thread cutting G33, G34, G35

- The following alarms are output when programming is incorrect:

Alarm	10604	"Thread lead increase too high"
Alarm	10605	"Thread lead decrease too high"
Alarm	10607	"Thread with frame (ROT) cannot be executed"
Alarm	16005	"Illegal lift-off path"
Alarm	16710	"Master spindle not programmed"
Alarm	16720	"Thread lead is zero"
Alarm	16730	"Incorrect parameters"
Alarm	16740	"No geometry axis programmed"

- If the spindle speed is too high when G33, G34, G35 are active, e.g., spindle override set to 200%, alarm 22270 "Spindle speed for thread cutting too high" is output.

  Alarm 22270 is output when the rapid traverse speed of the thread axis is exceeded. It is possible to reduce the spindle speed using the spindle override switch to prevent serious alarms.

References:
/FB1/ Description of Functions, Basic Machine; Diagnostic Tools (D1);
Chapter: General Machine Data
/DA/ Diagnostics Guide

2.1.4.4 Stop for thread cutting

Stop for thread cutting

Note

The non-destructive interrupt function should only be used for thread cutting, not for tapping with G33.

Retraction movement

The retraction motion (liftfast) for thread cutting is controlled by the following keywords:

- LFON

  ⇒ Enable liftfast for thread cutting.

- LFOF

  ⇒ Disable liftfast for thread cutting.

These G functions can always be programmed.

The default setting for NC Reset and/or NC Start is specified in machine data:
MD20150 GCODE_RESET_VALUES.

The following sources can initiate a retraction motion during thread cutting:

- Fast inputs (programming with `SETINTLIFTFAST` for "LIFTFAST" option)
- NC Stop
- Alarms that implicitly initiate the NC stop.

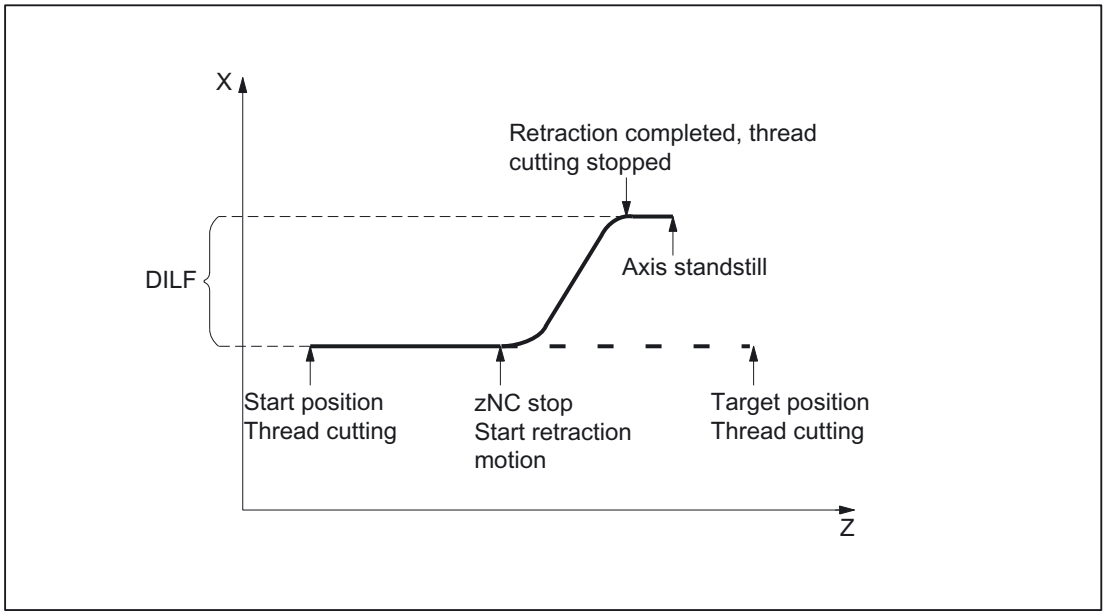


Fig. 2-3 Interruption of `G33` through retraction motion

Retraction path

The retraction path can be configured in machine data:
MD21200 LIFTFAST_DIST.

If required, this path in the part program can be changed by writing `DILF` at any point.

Following an NC reset, the value entered in machine data:
MD21200 LIFTFAST_DIST
(default value) is always active.

Retraction direction

The method for defining the retraction direction is controlled in conjunction with the variable ALF using the following keywords:

- LFTXT

  The plane of the retraction movement is determined by the path tangent and the tool direction. This G code (default setting) is used to program the response on a fast lift.

  For information about ALF programming, refer to:

  References:
  /PGA/ Programming Guide, Advanced
  /FB1/ Description of Functions, Basic Machine; Mode Group, Channel, Program Operation (K1);
  Chapter: Asynchronous Subroutines (ASUBs), Interrupt Routines

- LFWP

  The plane of the retraction movement is the active working plane selected with G codes G17, G18 or G19. The direction of the retraction plane is independent of the path tangent. This allows a fast lift to be programmed parallel to the axis.

These G functions can always be programmed.

The default setting for NC Reset and/or NC Start is specified in machine data: MD20150 GCODE_RESET_VALUES.

In the plane of the retraction movement, ALF is used, as before, to program the direction in discrete steps of 45 degrees. With LFTXT, retraction in the tool direction was 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 X direction
		ALF=3	Retraction in Y direction
G18:	Z/X plane	ALF=1	Retraction in Z direction
		ALF=3	Retraction in X direction
G19:	Y/Z plane	ALF=1	Retraction in Y direction
		ALF=3	Retraction in Z direction

Note

The extension for programming the plane of the retraction movement can be used independently of thread cutting.

Retraction speed

Retraction is performed at maximum axis velocity.
This can be configured in machine data:
MD32000 MAX_AX_VELO.

Retraction acceleration, jerk

Acceleration is performed at maximum possible values.
This can be configured in machine data:
MD32300 MAX_AX_ACCEL.

Example

```
N55 M3 S500 G90 G18                              ; Set active machining plane
...
N65 MSG ("thread cutting")
MM_THREAD:
N67 $AC_LIFTFAST=0                               ; Reset before beginning of
                                                   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
                                                   (due to G18)
                                                 ; Retraction direction -X
                                                   (with ALF=3; retraction
                                                   direction +X)
N71 G33 Z55 X15
N72 G1                                           ; Deactivate thread cutting
N69 IF $AC_LIFTFAST GOTOB MM_THREAD              ; If thread cutting
                                                 ; was aborted
N90 MSG ("")
...
N70 M30
N55 M3 S500 G90 G0 X0 Z0
...
N87 MSG ("tapping")
N88 LFOF                                         ; Deactivate fast retraction
                                                   before tapping
N89 CYCLE...                                     ; Thread drilling cycle with G33
N90 MSG ("")
...
N99 M3012.97
```

Control system response

At POWER ON and RESET, the retraction path is set as configured (MD), as is the state of LFON or LFOF and LFTXT or LFWP using machine data:
MD20150 GCODE_RESET_VALUES
from the corresponding G group.

Smoothed actual values

If low-resolution encoders are being used, smoothed actual values can be applied for constant coupled path and axis motions.

Machine data:
MD34990 ENC_ACTVAL_SMOOTH_TIME
can be used to change the time constant for more effectively smoothed actual values for:

- Thread cutting with feedrate for G33, G34, G35
- Revolutional feedrate for G95, G96, G97, FRAPON
- Display of actual positions and actual speed/velocity

The larger the time constant, the better the smoothing of the actual values and the longer the overtravel.

2.1.5 Feedrate for G331/G332, rigid tapping

Application

G331 (tapping) and G332 (tapping retraction) can be used to tap a thread without a compensating chuck (rigid tapping) if the spindle is technically capable of operating in position control mode.

Speed S, feedrate F, thread lead

A revolutional feedrate [mm/revolution] is used for G331and G332 threads. The revolutional feedrate is defined by programming the thread lead increase [mm/revolution].

The speed of the axes for the thread length is calculated from the programmed spindle speed S and the thread lead.

Feedrate F [mm/min] = speed S [rev/min] * thread lead [mm/rev]

Note

For further information about programming G63/G331/G332, please see:
References:
/PA/ Programming Guide, Fundamentals
/PAZ/ Programming Guide, Cycles

Inhibiting stop events for G331/G332

During tapping, a stop can be prevented if the block contains a path movement or a G4.

For this purpose, bit0 must be set to 0 in machine data:
MD11550 STOP_MODE_MASK.

The stop, which was activated previously, is possible again after G331/G332 has been executed.

2.1.6 Feedrate for G63, tapping with compensation chuck

Application

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 lead

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 lead.

Feedrate F [mm/min] = speed S [rev/min] * thread lead [mm/rev]

2.2 Feedrate FA for positioning axes

Syntax

```
FA[<positioning axis>] = <feedrate value>
```

Functionality

The velocity of a positioning axis is programmed with axis-specific feedrate FA.

- Effectiveness: modal
- Special points to be noted

  – No more than 5 axis-specific feedreates can be programmed in each part program block.

  – The feedrate is always G94.

  – If no axial feedrate FA is programmed, the axial default:
  MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity) takes effect.

Parameter: Positioning axis

- Value range: identifier of channel axes
  MD20080 $MC_AXCONF_CHANAX_NAME_TAB

Parameter: Feed value

- Value range
  0.001...999,999.999 mm/min, degrees/min or 0.001, ...39 999,9999 inch/min

  For more information about the value range of the axial feedrate see:

  References:
  /PA/ Programming Guide, Fundamentals"
  /FB/ Description of Functions, Basic Machine; Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)

- NC/PLC interface (channel-specific)
  The feedrate is output at the channel-specific NC/PLC interface:
  DB21, ... DBB158 - DBB193
  For a description of the channel-specific NC/PLC interface signals (modification signal, machine axis number of positioning axis, etc.) see:

  References:
  /FB/Description of Functions, Basic Machine; Various Interface Signals and Functions (A2)

- NC/PLC interface (axis-specific)
  The feed value is output at the axis-specific NC/PLC interface:
  DB31, ... DBB78 - DBB81

Note

Maximum axis velocity
The maximum axis velocity is not exceeded:
MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

F function output to the NC/PLC interface
Output of the F functions to the NC/PLC interface is not recommended. Activating the output of F functions to the NC/PLC interface can cause a drop in velocity in continuous-path mode. The output of F functions to the NC/PLC interface can be suppressed:
MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of the F functions)

For a detailed description please refer to:

References:
/FB/ Description of Functions - Basic Machine, Auxiliary Functions (H2)

RESET response

After end of program or NC-RESET, the feedrate value takes effect depending on machine data:
MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET = <value>

Value	Meaning
0	After NC RESET the default values are effective.
1	After NC RESET the last FA values to be programmed are effective.

2.3 Feedrate control

2.3.1 General

Programming and controlling the feedrate

The figure below shows the options for feedrate programming and control.

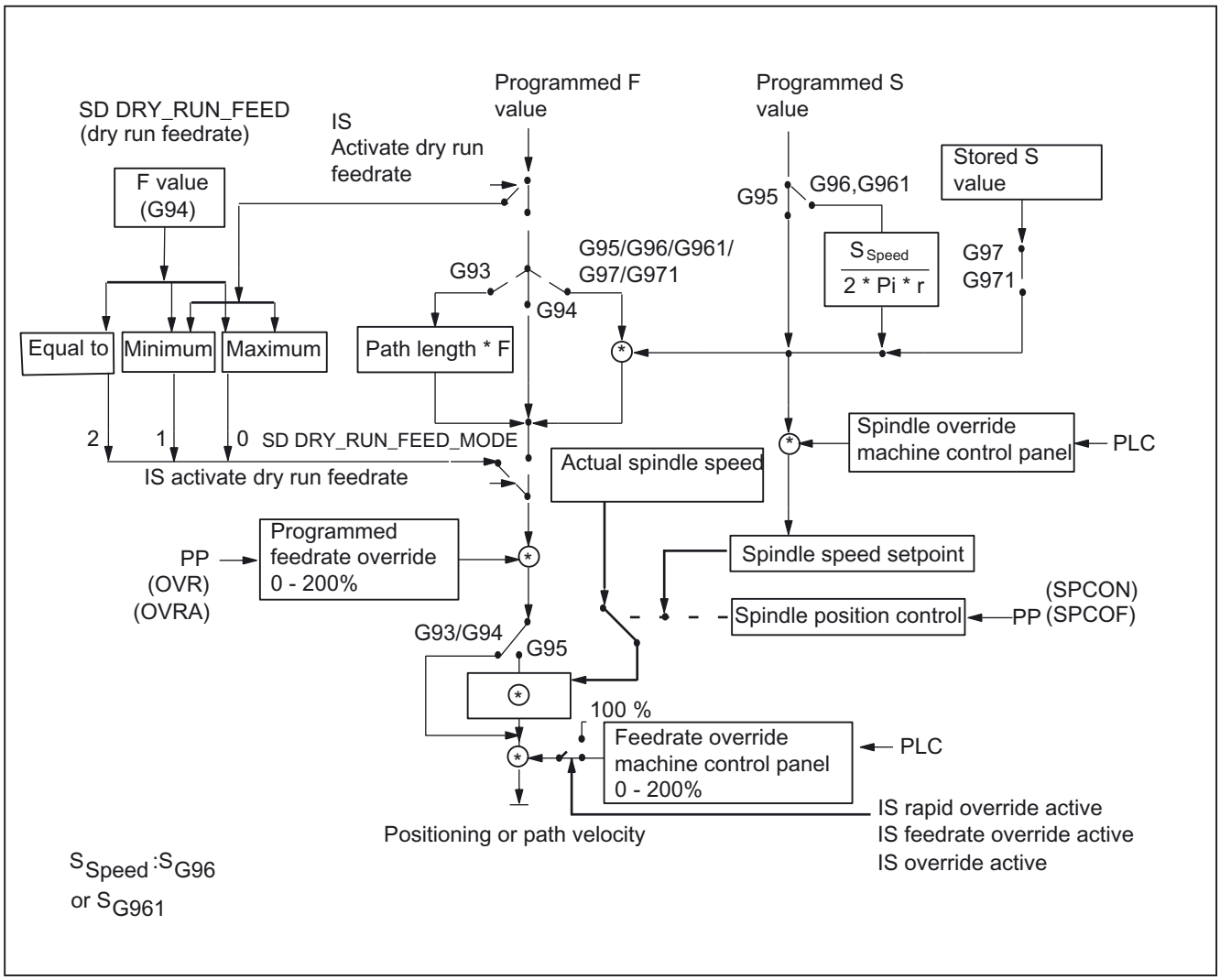


Fig. 2-4 Programming and controlling the feedrate

2.3.2 Feedrate disable and feedrate/spindle stop

General

The feedrate disable or feedrate/spindle stop brings the axes to a standstill with adherence to the braking characteristics and the path contour (exception: G33 block).

References:
/FB/ Description of Functions, Basic Machine; Continuous-Path Mode, Exact Stop and Look Ahead (B1)

"Feedrate disable" DB21, ... DBX6.0

Interface signal "Feedrate disable" (DB21, ... DBX6.0) shuts down all axes (geometry and auxiliary axes) of a channel in all modes.

Activation of channel-specific feedrate disable	
If G33, G34, G35 active:	**Not** effective
If G63 active:	Effective
If G331, G332 active:	Effective

"Feed stop" DB21, ...DBX12.3 geometry axes

Interface signals "Feed stop" (DB21, ... DBX12..3 and the following signals) for geometry axes 1, 2, and 3 stop the geometry axes of a channel in JOG mode.

"Feed stop" DB31, ...DBX4.3 axis-specific

The axis-specific "Feed stop" interface signal (DB31, ... DBX4.3) is used to stop the relevant 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.

Activation of axis-specific function "Feed stop"	
If G33, G34, G35 active:	Effective (causes contour deviations)
If G63 active:	Effective
If G331, G332 active:	Effective

Axis/spindle disable" DB31, ... DBX1.3

Activating "Axis/spindle disable" (DB 31, ...DBX1.3) disables the axial PLC interlocks "No controller enable" and "Feed stop".

The axial and channel-specific override, however, is active.

"Spindle stop" DB31, ... DBX4.3

IS "Spindle stop" (DB31, ... DBX4.3) is used to stop the relevant spindle.

Activation of the "Spindle stop" function	
If G33, G34, G35 active:	Effective (may cause contour deviations depending on dynamic characteristics)
If G63 active:	Effective
If G331, G332 active:	**Not** effective

2.3.3 Feedrate override on machine control panel

General

The operator can use the feedrate override switch to increase or decrease the path feedrate relative to the programmed percentage with immediate effect. The feedrates are multiplied by the override values.

An override between 0 and 200% can be programmed for the path feedrate.

The rapid traverse override switch is used to reduce the traversing velocity when testing a part program.

An override between 0 and 100% can be programmed for the rapid traverse.

The feedrate can be changed axis-specifically for positioning axes. The override can be between 0 and 200%.

The spindle override can be used to modify the spindle speed and the cutting rate (with G96, G961). The override can be between 0 and 200%.

The override is not permitted to exceed the machine-specific acceleration and speed limits or generate a contour error.

The feedrate override can be changed separately for path and positioning axes.

The overrides act on the programmed values or on the limits (e.g., G26, LIMS for spindle speed).

A sequence of G63 blocks is traversed without taking into account the feedrate override. A sequence of G63 blocks consists of directly consecutive blocks with G code G63. It starts with the first G63 block and finishes with the first path motion block, which is not a G63 block.

Channel-specific "Feedrate override" DB21, ...DBB4/"Rapid traverse override" DB21, ...DBB5

One enable signal and one byte is provided on the PLC interface for the override factor for each of the feedrate types.

IS "Feedrate override" (DB21, ... DBB4)

IS "Feedrate override active" (DB21, ... DBX6.7)

IS "Rapid traverse override" (DB21, ... DBB5)

IS "Rapid traverse override active" (DB21, ... DBX6.6)

The interface for the override can be provided by the PLC in binary-coded or Gray-coded format.

Machine data:
MD12020 OVR_FEED_IS_GRAY_CODE (path feedrate override switch Gray-coded)
and
MD12040 OVR_RAPID_IS_GRAY_CODE (rapid traverse override switch Gray-coded)
are used to define whether binary or Gray coding is active.

The following permanent assignment applies to binary coding:

Code	Override factor
00000000	0,00 ≙ 0%
00000001	0,01 ≙ 1%
00000010	0,02 ≙ 2%
00000011	0,03 ≙ 3%
00000100	0,04 ≙ 4%
.	.
.	.
.	.
01100100	1,00 ≙ 100%
.	.
.	.
.	.
11001000	2,00 ≙ 200%

With Gray coding, the override factors corresponding to the switch setting are entered in machine data:
MD12030 OVR_FACTOR_FEEDRATE [n]
(analysis of path feedrate override switch)
or
MD12050 OVR_FACTOR_RAPID_TRA [n] (analysis of rapid traverse override switch).

An active feedrate override acts on all path axes assigned to the current channel.

An active rapid traverse override acts on all axes moving with rapid traverse and assigned to the current axis.

If no dedicated rapid traverse override switch is used, the system can be switched between rapid traverse override and feedrate override. In this case, feedrate overrides above 100% are limited to 100% rapid traverse override.

The override, which is to be active, can be selected from the PLC or operator panel. When rapid traverse override is activated via the operator front panel, interface signal "Feedrate override for rapid traverse selected" (DB21, ... DBX25.3) is transferred by the PLC basic program to interface signal "Rapid traverse override active" (DB21, ... DBX6.6) and interface signal "Feedrate override" (DB21, ... DBB4) is copied to IS "Rapid traverse override" (DB21, ... DBB5).

If selected via the PLC, interface signal "Rapid traverse override active" (DB21, ...DBX6.6) must be set by the PLC user program and the interface signal of the feedrate override (DB21, ... DBB4) must be copied to the interface signal of the rapid traverse override (DB21, ... DBB5).

Activation of channel-specific feedrate and rapid traverse override	
If G33, G34, G35 active:	**Not** effective
If G63 active:	**Not** effective
If G331, G332 active:	**Not** effective

Reference speed for path feedrate override

The reference velocity for the path feedrate override specified via the machine control panel can be set to a value other than the default value.

The selection is made via machine data:
MD12082 OVR_REFERENCE_IS_MIN_FEED.

Axis-specific "Feedrate override" DB31, ...DBB0

One enable signal and one byte for the feedrate override factor are available on the PLC interface for each positioning axis.

IS "Feedrate override" (DB31, ... DBB0)

IS "Override active" (DB31, ... DBX1.7)

The interface for the feedrate override can be provided by the PLC in binary-coded or Gray-coded format.

Machine data:
MD12000 OVR_AX_IS_GRAY_CODE (axis feedrate override switch Gray-coded)
is used to define whether binary or Gray coding is active.

The following permanent assignment applies to binary coding:

Code	Override factor
00000000	0,00 ≙ 0%
00000001	0,01 ≙ 1%
00000010	0,02 ≙ 2%
00000011	0,03 ≙ 3%
00000100	0,04 ≙ 4%
.	.
.	.
.	.
01100100	1,00 ≙ 100%
.	.
.	.
.	.
11001000	2,00 ≙ 200%

With Gray coding, the override factors corresponding to the switch setting are entered in machine data:
MD12010 OVR_ FACTOR_AX_ SPEED [n] (analysis of axis feedrate override switch).

Activation of axis-specific feedrate override	
If G33, G34, G35 active:	**Not** effective
If G63 active:	**Not** effective (the override is set to a fixed value of 100% in the NC)
If G331, G332 active:	**Not** effective (the override is set to a fixed value of 100% in the NC)

Spindle override" DB31, ...DBB0"

One enable signal and one byte is provided on the PLC interface for the override factor for each of the spindles.

IS "Spindle override" (DB31, ... DBB19)

IS "Override active" (DB31, ... DBX1.7)

The interface for the spindle override can be provided by the PLC in binary-coded or Gray-coded format.

Machine data:
MD12060 OVR_SPIND_IS_GRAY_CODE (spindle override switch is Gray-coded)
is used to define whether binary or Gray coding is active.

The following permanent assignment applies to binary coding:

Code	Override factor
00000000	0,00 ≙ 0%
00000001	0,01 ≙ 1%
00000010	0,02 ≙ 2%
00000011	0,03 ≙ 3%
00000100	0,04 ≙ 4%
.	.
.	.
.	.
01100100	1,00 ≙ 100%
.	.
.	.
.	.
11001000	2,00 ≙ 200%

With Gray coding, the override factors corresponding to the switch setting are entered in machine data:
MD12070 OVR_FACTOR_SPIND_SPEED [n]
(analysis of spindle override switch).

Activation of spindle override	
If G33, G34, G35 active:	Effective (if the spindle is in position control mode, the override switch may be activated)
If G63 active:	**Not** effective (the override is set to a fixed value of 100% in the NC)
If G331, G332 active:	Effective

Limiting the override factor

On a binary-coded interface, the maximum possible override factors for path feedrate, axis feedrate and spindle speed can be further limited with machine data:
MD12100 OVR_FACTOR_LIMIT_BIN
(limit of binary-coded override switch)

Override active DB21, ...DBX6.6/DB21, ... DBX6.7/DB31, ... DBX1.7

The override values set with the selector switch on the machine control panel are immediately active for all operating modes and machine functions, provided that IS "Rapid traverse override active", (DB21, ... DBX6.6) "Feedrate overide active" (DB21, ... DBX6.7) and "Override active" (DB31, ... DBX1.7) are set.

An override factor of 0% acts as a feedrate disable.

Override inactive

When the override is inactive (i.e., the above interface signals are set to “0”), the override factor “1” is used internally on the NC (i.e., the override is 100%). In this case, the value entered in the PLC interface has no effect.

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. This should be initialized with "0" .

An override factor of 0% acts as a feedrate disable.

Spindle override reference

An entry is made in machine data:
MD12080 OVR_REFERENCE_IS_PROG_FEED (override reference velocity)
to indicate whether the spindle override relates to the speed limited using MD or SD or to the programmed speed.

2.3.4 Programmable feedrate override

Function

The programmable feedrate override can be used to change the velocity level of path and positioning axes by means of a command in the part program.

A separate feedrate override can be programmed for positioning axes.

Programming

The feedrate override can be changed with the following commands:

`OVR= ....`	Feedrate change for path feedrate F
`OVRA[X1]= ...`	Feedrate change for positioning feedrate FA

The programmable range is between 0 and 200%.
The default setting is 100%.

Effectivity

Interface signals "Rapid traverse and feedrate override" (DB21, ... DBB6) and "Axis-specific override active" (DB31, ... DBX1.7) do not apply to the programmable feedrate override. The programmable feedrate override remains active when these signals are deactivated.

The active override is calculated from the product of the programmable feedrate override and the feedrate override from the machine control panel.

The default setting is 100%. It is active if no feedrate override has been programmed or following a RESET, if machine data:
MD22410 F_VALUES_ACTIVE_AFTER_RESET (F function active via RESET)
is not set.

OVR is not active with G33, G34, G35.

2.3.5 Dry run feedrate

Application

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.

"Dry run feedrate selected" DB21, ... DBX24.6

The dry run feedrate can be activated from the PLC or operator panel.
When activated from the operator panel, interface signal "Dry run feedrate selected" (DB21, ... DBX24.6) is set and transferred by the PLC basic program to interface signal "Activate dry run feedrate" (DB21, ... DBX0.6).

When selected on the PLC, the "Activate dry run feedrate" interface signal is required to be set by the PLC user program.

The execution of the program is triggered with G94.

The dry run feedrate also takes precedence over the feedrates for G93, G95 and G33, G34, G35.

The programmed feedrate is compared with the dry run feedrate in:
SD42100 DRY_RUN_FEED
and then the higher of the two feedrates is applied during travel.

Dry run feedrate change

The dry run feedrate (SD42100 DRY_RUN_FEED) can be changed via the operator panel in the operating area "Parameters".

No changes can be made once the program has started up.

Evaluation

The effect of:
SD42100 DRY_RUN_FEED
can be controlled using an additional setting data:
SD42101 DRY_RUN_FEED_MODE.

SD42101 DRY_RUN_FEED_MODE	Effectivity
	As long as the "Activate dry run feedrate" interface signal is set, instead of the programmed feedrates, the feedrate value set in SD42100 DRY_RUN_FEED is effective as follows:
0	The maximum value from SD42100 DRY_RUN_FEED and the programmed velocity is effective. This is the same as the default setting.
1	The minimum value from SD42100 DRY_RUN_FEED and the programmed velocity is effective.
2	Setting data SD42100 DRY_RUN_FEED is effective directly, regardless of the programmed velocity.
3 -9	Reserved
10	As for configured value 0 except for thread cutting (G33, G34, G35) and tapping (G33, G34, G35). These functions are executed as programmed.
11	As for configured value 1 except for thread cutting (G33, G34, G35) and tapping (G33, G34, G35). These functions are executed as programmed.
12	As for configured value 2 except for thread cutting (G33, G34, G35) and tapping (G33, G34, G35). 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.

If the machine data setting:
MD10704 DRYRUN_MASK = 1
is set, the dry run feedrate can also be activated during execution of the program (in the part program block).

Note

Activation during machining triggers an internal reorganization task on the control which causes the axes to be stopped for a short time. This can affect the surface finish of the workpiece being machined.

2.3.6 Multiple feedrate values in one block

Application

The functionality described in the following is used primarily for grinding, but is not restricted to it.

References:
/FB/ Description of Functions, Extensions; Grinding (W4)
/FB/ Description of Functions, Extensions; Oscillation (P5)

Functionality

The function "Multiple feedrates in one block" can be used to activate 6 different feedrate values of an NC block, a dwell time or a retraction motion synchronously, depending on the external digital and/or analog inputs.

Retraction

Retraction by the programmed amount is initiated in the IPO cycle.

Signals

The HW input signals are combined in one input byte for the function "Multiple feedrate in one block". A permanent functional assignment applies within the byte:

Table 2-1 Input byte for "Multiple feedrates in one block"

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 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
I1	Activation of dwell time ST (in seconds)
I0	Activation of retraction motion SR

Priorities

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 - F7.
SR also ends ST, i.e., the complete function.

The highest-priority signal determines the current feedrate.

Machine data:
MD21230 MULTFEED_STORE_MASK
(store input signals for the "Multiple feedrates in one block" function)
can be used to define the response on loss of the highest-priority input in each case (F2 - F7).

The end-of-block criterion is satisfied when:

- The programmed end position is reached
- The retraction motion ends (SR)
- The dwell time elapses (ST)

Delete distance-to-go

A retraction motion or dwell time causes the distance-to-go to be deleted.

Hardware assignment

The channel-specific machine data:
MD21220 MULTFEED_ASSIGN_FASTIN
(assignment of the input byte of the NCK I/Os for "Multiple feedrates in one block")
can be used to assign up to two digital input bytes or comparator input bytes to the NCK I/O.

Machine data:
MD21220 MULTFEED_ASSIGN_FASTIN
can also be used to invert the input bits.

If a 2nd byte is entered, the contents of the 1st and 2nd byte are ORed before being used.

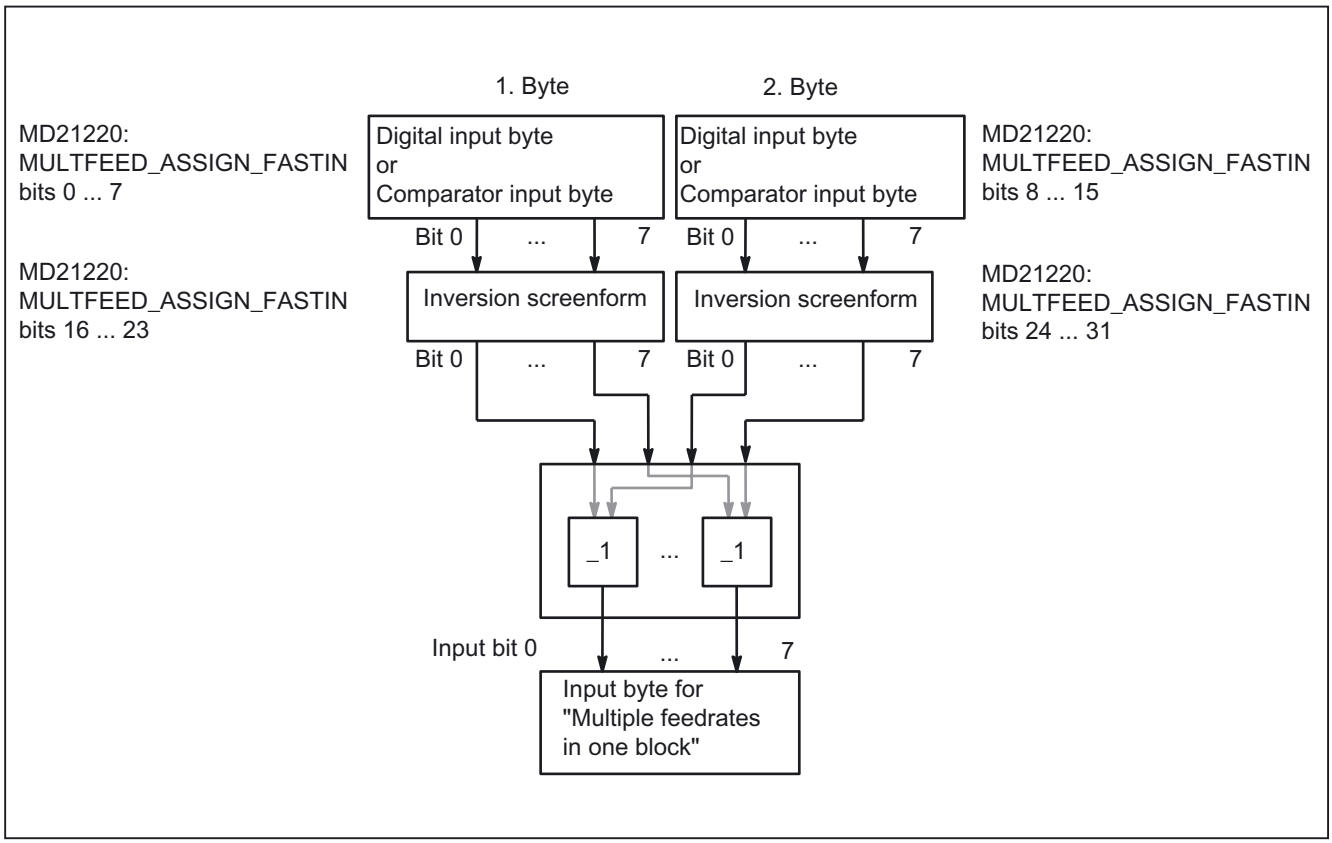


Fig. 2-5 Signal routing for "Multiple feeds in one block"

The assignment of the digital input bytes and parameterization of the comparators are described in:
References:
/FB/ Description of Functions, Basic Machine; Various Interface Signals (A2)

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 6 further feedrates in the block.

The numerical expansion indicates the bit number of the input that activates the feedrate when changed:

e.g., F7=1000 ; 7 corresponds to input bit 7.

Bits 2 to 7 are permissible for numerical expansion of the feedrate. 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 (for grinding sparking out time)
SR=...	Retraction path

These addresses apply non-modally.

Programming axial motion

The axial feedrates are programmed under address FA and remain valid until an input signal is present. They act modally.

FMA[2,x]=... to FMA[7,x]=... can be used to program up to 6 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
```

Bits 2 to 7 are permissible for the numerical expansion of the axial feedrate. The values programmed under FMA are active 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[x]=...	;Dwell (sparking out time), axis-specific
SRA[x]=...	;Retraction path, axis-specific

The expression in square brackets indicates the axis for which the sparking out time and retraction path apply.

```
STA[X]=2.5              ;Sparking out time 2.5 seconds
SRA[X]=3.5              ;Retraction path 3.5 (unit e.g.,: mm)
```

These addresses apply non-modally.

If feedrates, a sparking out time (dwell) or retraction path are programmed for an axis on account of an external input, this axis must not be programmed as a POSA axis in this block (positioning axis beyond end of block).

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 or retraction is started.

Note

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.

It is also possible to poll the status of an input for synchronous commands of various axes.

Look Ahead is also active for multiple feedrates in one block. In this way, the current feedrate is restricted by the Look Ahead value.

Application

Typical applications include:

- Analog or digital calipers

  Depending on whether the external inputs are analog or digital, various feedrate values, a dwell 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.

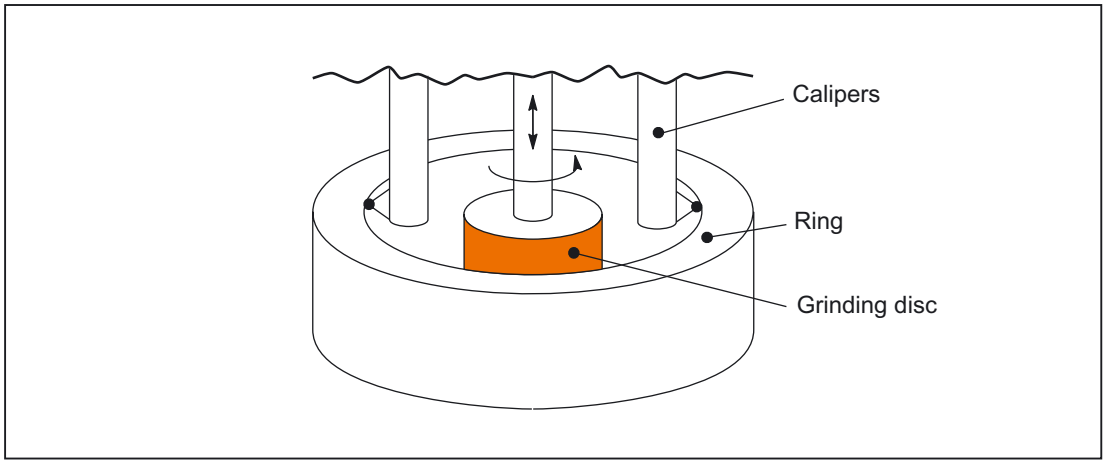
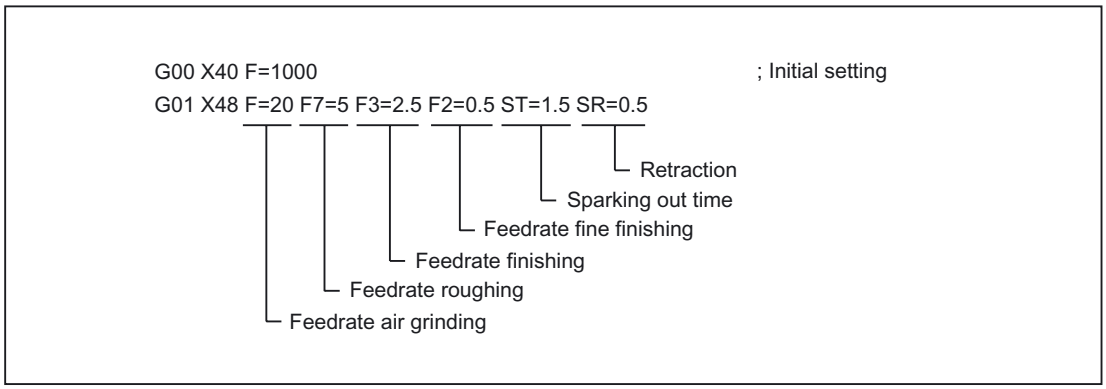


Fig. 2-6 Calipers

Part program



Note

The axial feedrate/path feedrate (F value) is the 100% feedrate.

Feedrates smaller than or equal to the axial feedrate/path feedrate can be implemented with "Multiple feedrate values in one block" (F2 to F7 values).

The "Multiple feedrate values in one block" functionality is only available in conjunction with the "Synchronized actions" function.

References:
/FB/Description of Functions, Basic Machine; Various Interface Signals and Functions (A2)

2.3.7 Fixed feedrate values (840D, 810D)

Function

The machine data can be used to define 4 fixed feedrate values, which can be activated via the interface signal. The function is possible in AUTOMATIC and JOG modes.

Behavior in AUTOMATIC mode

The fixed feedrate, which has been selected, is used instead of the programmed feedrate. The following MDs and interface signals can be used to select fixed feedrates for path/geometry axes:

- MD12202 $MN_PERMANENT_FEED[n] (fixed feedrates for linear axes)
- MD12204 $MN_PERMANENT_ROT_AX_FEED[n] (fixed feedrates for rotary axes)
- IS "Activate fixed feedrate x for path/geometry axes" (DB21, ... DBX29.0- 29.3)

The contour travels at the activated fixed feedrate, instead of using the programmed feedrate.

Behavior in JOG mode

The fixed feedrate selected via the interface signal is applied during travel instead of the JOG velocities set.

The travel direction is specified via the interface signal.

The following MDs and interface signals can be used to select fixed feedrates for path/geometry axes and for machine data:

- MD12202 $MN_PERMANENT_FEED[n] (fixed feedrates for linear axes)
- MD12204 $MN_PERMANENT_ROT_AX_FEED[n] (fixed feedrates for rotary axes)
- IS "Activate fixed feedrate x for machine axes" (DB31, ... DBX3.2-3.5)

The axis travels at the activated fixed feedrate, instead of at the JOG velocity/JOG - rapid traverse velocity set in MD.

Constraints

- The fixed feedrate is not active for spindles, positioning axes and tapping.
- During travel at fixed feedrate, the override is determined by machine data:

  MD12200 $MN_RUN_OVERRIDE_0 (travel with override 0)
- When fixed feedrate is selected, it is not possible to activate DRF offset.
- The fixed feedrates are always linear feedrate values. Revolutional feedrates are converted to linear feedrates internally.

Interface signals

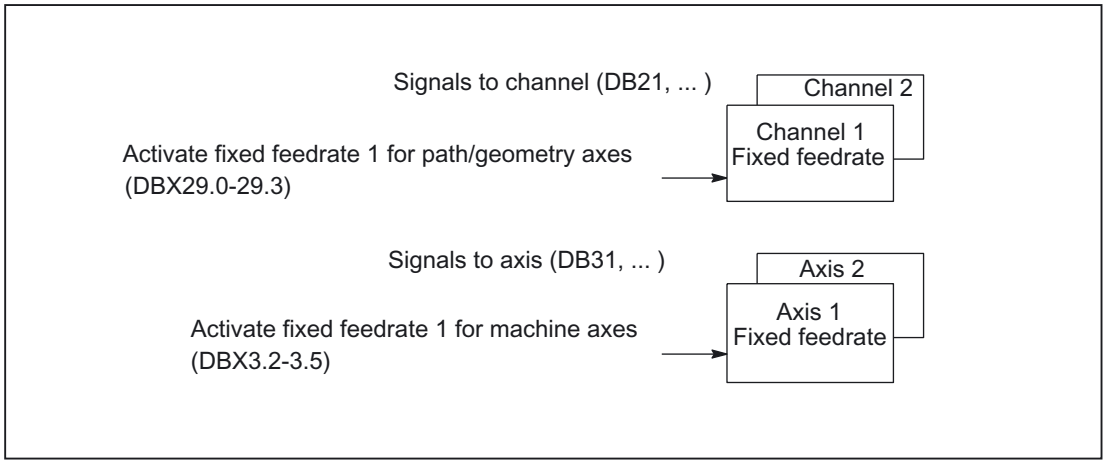


Fig. 2-7 Overview of interface signals for fixed feedrate

2.3.8 Feedrate for chamfer/rounding FRC, FRCM

General

The machining conditions can change significantly during surface transitions to chamfer/rounding. The chamfer/rounding contour elements therefore need their own optimized feedrate values in order to achieve the required surface finish.

Function

You can program the feedrate for the chamfer/rounding with FRC (non-modal) or FRCM (modal).

The feedrate value is interpreted according to the type of feedrate active:

G94, G961, G971:	Feedrate in mm/min, inch/min or °/min
G95, G96, G97:	Revolutional feedrate in mm/rev or inch/rev

The FRC/FRCM value is applied depending upon machine data:
MD20201 CHFRND_MODE_MASK:

Bit 0 = 0:	FRC/FRCM from following block (default setting)
Bit 1 = 0:	FRC/FRCM from preceding block (recommended setting)

Reason: The feedrate type (G94, G95, G96, G961 etc.) and thus the conversion to the internal format must be consistent within the block for F and FRC/FRCM.

Empty blocks

When chamfer or rounding is active, the possible number of blocks containing no traversing information is limited.

The maximum number is defined in machine data:
MD20200 CHFRND_MAXNUM_DUMMY_BLOCKS.

The possible blocks without traversing information in the compensation plane are pure dummy commands and are called empty blocks. For this reason, they may only be written between two blocks with traversing information.

Constraints

- Feedrate interpolation FLIN and FCUB is not possible for chamfer/rounding.
- FRC/FRCM has no effect if a chamfer is being machined with G0; the command can be programmed according to the F value without error message.
- FRC is only effective if a chamfer/rounding is programmed in the same block or if RNDM has been activated.
- FRC overwrites the F or FRCM value for chamfer/rounding in the current block.
- FRC/FRCM must be set to a value greater than zero.
- FRCM=0 activates the feedrate programmed in F for the chamfer/rounding.
- If FRCM is programmed, the FRCM value must be reprogrammed, equivalent to F, on a G94 ↔ G95 change, etc. If only F is reprogrammed and FRCM > 0 before the feedrate type is changed, error message 10860 (no feedrate programmed) appears.

2.3.9 Non-modal feedrate FB

General

A separate feedrate can be specified for a single block with the command FB (Feed Block). For this block the previously active path feedrate is overwritten. After this block, the previously active modal path feedrate becomes active once more.

Function

The non-modal feedrate is programmed with FB=<value>.

The feedrate value is interpreted according to the type of feedrate active:

G94, G961, G971:	Feedrate in mm/min, inch/min or °/min
G95, G96, G97:	Revolutional feedrate in mm/rev or inch/rev

Constraints

- The programmed value of FB=<value> must be greater than zero.
- If no traversing motion is programmed in the block (e.g.: computation block), FB has no effect.
- If no explicit feedrate for chamfering/rounding is programmed, then the value of FB also applies for any contour element chamfering/rounding in this block.
- The path velocity profile programmed with FLIN or FCUB does not act with the rotational feedrate for G95 and with constant cutting speed for G96/G961 and G97/G971.
- Feedrate interpolations FLIN, FCUB, .. are possible without restriction.
- Simultaneous programming of FB and FD (handwheel travel with feedrate overlay) or F (modal path feedrate) is not possible.

2.3.10 Programmable single-axis dynamic response

Single axes

Single axes can be programmed:

In the part program	POS[Axis]=...
	POSA[Axis]=...
	SPOS[Axis]=...
	SPOSA[Axis]=...
	OS[Axis]=...
	OSCILL[Axis]=...
In synchronized actions (command axes)	EVERY ... DO
	POS[Axis]=...
	SPOS [Spindle] = ...
	MOV[Axis]=...
In the PLC	FC15/FC16 and FC18

Dynamic response of an axis/spindle

The dynamic response of an axis is dependent on:

- The preset feed value (MD32060 POS_AX_VELO)

  Can be programmed in the part program with `FA[axis]= ...`, or a feed override `OVRA[axis]= ...` in percent.

  Programmable in synchronized actions using `FA[Axis]= ...`.

  Modifiable from the PLC by entering FRate, or overwriting the axial override.

- The acceleration value (MD32300 MAX_AX_ACCEL)

  Programmable indirectly in the part program by overwriting the machine data with subsequent `NewConfig`, or directly using a percentage acceleration override `ACC[Axis]= ...`.

  Modifiable indirectly in synchronized actions by overwriting the MD (no `NewConfig` possible) or by executing an ASUB, or programmable directly using a percentage acceleration override `ACC[Axis]= ...` (cannot be preset by the PLC).

  The PLC has the same options as synchronized actions.

- The acceleration characteristics

  Programmable in the part program using `BRISKA`(axis), `SOFTA`(axis), `DRIVEA`(axis) and `JERKA`(axis).

  Not programmable in synchronized actions (only directly using an ASUB).

  Cannot be preset by the PLC (only indirectly using an ASUB).

- The selected servo parameter set

  A parameter set describes the most important setting options for control purpose in the servo.

  It comprises, e.g., the following axis-specific/spindle-specific machine data:

31050 DRIVE_AX_RATIO_DENOM[n]	Parameter set n for 0 to 5
31060 DRIVE_AX_RATIO_NUMERA[n]	Parameter set n for 0 to 5
32200 POSCTRL_GAIN[n]	Parameter set n for 0 to 5
32452 BACKLASH_FACTOR[n]	Parameter set n for 0 to 5
32610 VELO_FFW_WEIGHT[n]	Parameter set n for 0 to 5
32800 EQUIV_CURRCTRL_TIME[n]	Parameter set n for 0 to 5
32810 EQUIV_SPEEDCTRL_TIME[n]	Parameter set n for 0 to 5
32910 DYN_MATCH_TIME[n]	Parameter set n for 0 to 5
36012 STOP_LIMIT_FACTOR[n]	Parameter set n for 0 to 5
36200 AX_VELO_LIMIT[n]	Parameter set n for 0 to 5

For more information about parameter sets and programming, please refer to:
References:
/FB/ Description of Functions, Basic Machine; Speeds, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)
/PG/ Programming Guide, Fundamentals; Path Response and Feed Control

Dynamic criteria and feedforward control

With dynamic response criteria, it must be distinguished from where they were set:

- In the part program

  or

- From a main-run interpolation (synchronized action or PLC default)

Note

Changes in the dynamic response made in a part program do not affect command axis or PLC axis movements.

Changes from synchronized actions do not affect the movements from a part program.

Travel with feedforward control ON/OFF

The type of feedforward control and the path axes to which feedforward is to be applied are determined:

- Programmable in the part program with FFWON/ FFWOF for axes selected using machine data.
- Only indirectly programmable in synchronized actions (ASUB).
- Only indirectly programmable from the PLC (ASUB).

Percentage acceleration override

ACC[Axis]

`ACC[Axis]= 0 .. 200` can be used to modify the acceleration set in machine data: MD32300 MAX_AX_ACCEL
in a range between 0% and 200% in part programs and synchronized actions.

`ACC[Axis]= <value>`	Where	Axis =	Channel axis name (X, Y ....), spindle (S1, ...)
		Value =	Percent of MD32300 MAX_AX_ACCEL (0 <= value <= 200)

The current acceleration value can be called with the system variables $AA_ACC[Axis]. It is determined by:

$AA_ACC[Axis]= Content(MD32300 MAX_AX_ACCEL[Axis]) * ACC[Axis]/100

Note

The acceleration offset programmed with `ACC[...]` is always considered as specified above for the output in $AA_ACC. However, $AA_ACC is not output in the part program at the same time as in a synchronized action.

The value described in the part program is only considered in the system variable $AA_ACC as described in the part program, if ACC has not been changed in the meantime by a synchronized action.

The value described in the synchronized action is only considered in the system variable $AA_ACC as described in the synchronized action, if ACC has not been changed in the meantime by a part program.

Main run axes

Main run axes (MR axes) are axes that are interpolated by the main run and can be:

- Command axes (activated by synchronized actions)
- PLC axes (started by PLC via function block)
- Asynchronous oscillating axes (setting data or from part program)
- Neutral axes

Note

Depending on whether $AA_ACC is programmed in a part program or in a synchronized action, the ACC value is output for the NC axes or the main run axes.

Variable $AA_ACC must always be queried in the mode in which the acceleration was written (either part program or synchronized action).

Examples

The last programmed value is retained following a RESET.

In the part program

```
...
N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]
...
```

Or via a synchronized action

```
...
N100 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]
...
```

Acceleration factor written in part program:

```
...
ACC[X]=50
RO $AA_ACC[X]

IF (RO <> $MA_ MAX_AX_ACCEL[X] * 0,5
SETAL(61000)
ENDIF
```

Acceleration factor is set by the synchronized action:

```
WHEN TRUE DO ACC[X]=25 $R1 = $AA_ACC[X]
G4 F1

IF (RO <> $MA_ MAX_AX_ACCEL[X] * 0.25
SETAL(61001)
ENDIF

M30
```

End-of-motion criterion for single axes

Similar to the block change criterion for path interpolation (G601, G602 and G603) it is also possible to program the movement end criterion for single-axis interpolation in a part program or in synchronized actions for main run axes: command/PLC axes.

Programmable criterion	End-of-motion on reaching
FINEA[Axis]	"Exact stop fine"
COARSEA[Axis]	"Exact stop coarse"
IPOENDA[Axis]	"Interpolator stop" (IPO stop)

Axis: Channel axis name (X, Y ....), spindle (S1, ...)

The end-of-motion criterion set will affect how quickly or slowly part program blocks and technology cycle blocks with single-axis movements are completed.
The same applies for PLC positioning instructions via FC15/16/18.

The set end-of-motion criterion can be scanned with system variable $AA_MOTEND[Axis].

$AA_MOTEND[Axis] = 1	End-of-motion with "Exact stop fine"
$AA_MOTEND[Axis] = 2	End-of-motion with "Exact stop coarse"
$AA_MOTEND[Axis] = 3	End-of-motion with "IPO stop"

Note

Depending on whether $AA_MOTEND is programmed in a part program or a synchronized action, the MOTEND value is output for the NC axes or the main-run axes.

The last programmed value is retained following a RESET.

Example:

```
...
N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]
...
```

Or via a synchronized action

```
...
N100 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]
```

For more information about block changes and end-of-motion criteria for FINEA, COARSEA and IPOENDA, please refer to:
References:
/FB/ Description of Functions, Extended Functions; Positioning Axes (P2);
Chapter: Block change

Programmable servo parameter set

SCPARA[Axis] = ...

SCPARA[axis]= ... can be used to program the parameter set (consisting of MDs) in the part program and in synchronized actions (previously, this could only be done via the PLC).

SCPARA[Axis]= <value>	Where	Axis =	Channel axis name (X, Y ....), spindle (S1, ...)
		Value =	Desired parameter set (1≤ value ≤ 6)

DB3n DBB9 bit3

To prevent conflicts between the PLC user request and NC user request, a further bit is defined on the PLC→NCK interface:

DB3n DBB9 bit3 "Parameter set selection by SCPARA disabled".

The PLC user is thereby able to set up a structured sequence when using PLC parameter switchover and entries from synchronized actions or part programs simultaneously.

On a change of bit 3 (0 ⇒ 1 or 1 ⇒ 0), the entry is written to bits 0 - 2.

Note

If parameter set selection via SCPARA is disabled, there is no error message if the latter is programmed nevertheless.

The current parameter set can be scanned by system variable $AA_SCPAR[Axis].

Example:

```
...
N100 SCPARA[X]=3          ; The 3rd parameter set is selected for axis X.
...
```

Constraints

- 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.
- The PLC user program must be expanded if the servo parameter set is to be changed both inside a part program or synchronous action and the PLC.
- 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 defined by the NC	1

- When the operating mode is changed from AUTO ⇒ JOG, the programmed dynamic response changes remain valid.
- In the event of a 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.

  Example:

```
N01 G01 POS[X]=20 POS[Y]=30 IPOENDA[X] IPOENDA[Y]
N02 POS[Z]=55 FINEA[Z]
N03 $A_OUT[1]=1
N04 POS[X]=100 COARSEA[X]
N05 ......
TARGET:                                              ;  Block search target
```

- In this example, N04 serves as a container for all programmed end-of-motion criteria. Two action blocks are saved. The digital output (N03) is issued in the first action block and in the second, COARSEA is set for the X axis, IPOENDA for the Y axis and FINEA for the Z axis.

  The same applies to the programmed servo parameter set. The last programmed acceleration override is effective from the first approach block.

Supplementary Conditions

3

3.1 General constraints

Several feeds in one block

The "Multiple feeds in one block" function is only available in conjunction with the "Synchronous actions" function.

References:
/FBSY/ Description of Functions Synchronized Actions

3.2 Supplementary conditions for feedrate programming

Unit of measurement

The applicable unit of measurement for feedrates is based on the value entered in machine data:
MD12240 SCALING_SYSTEM_IS_METRIC
(standard control system metric/imperial)
and the type of axis entered in machine data:
MD30300 IS_ROT_AX
(rotary or linear axis).

Standard setting for feedrate type

G94 is displayed on the screen as the standard setting.

The initial setting (standard programming setting) for the feedrate type is only displayed when the part program starts up.

The initial setting is set in machine data:
MD20150 GCODE_RESET_VALUES (initial setting for G groups).

Activation of the F values

Machine data:
MD22140 F_VALUES_ACTIVE_AFTER_RESET (F function active via RESET)
can be used to define whether the following most recently programmed F values are to remain active following a RESET.

- Programmed path feedrate: F=...
- Programmed feedrate change for path feedrate: OVR=...
- Programmed positioning feedrate: FA=...
- Programmed feedrate change for positioning feedrate: OVRA[U]=...

For more detailed information about syntax, please see:
References:
/PA/ Programming Guide, Fundamentals

Position spindle

If G95, G96, G961, G97, G971, G33, G34, G35 is active, spindle positioning should not be performed, because the derived path feedrate following spindle positioning = 0.

⇒ If the programmed axis position has not yet been reached, the block cannot be completed.

4 Examples

4.1 Feedrate programming for chamfer/rounding FRC, FRCM

Example 1: Following block feedrate

MD20201 CHFRND_MODE_MASK Bit0 = 0
(apply feedrate from following block; default setting)

```
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
                                              FRC=100 mm/min
N80     Y30 CHF=3 FRC=100                   ; Chamfer N80-N90 with
                                              FRCM=50 mm/min  (modal)
N90     X40                                 ; Modal rounding N90-N100 with
                                              F=100 mm/min (deselect 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: Preceding block feedrate

MD20201 CHFRND_MODE_MASK Bit0 = 1
(apply feedrate from preceding block; recommended setting)

```
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  (modal)
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
```

Data Lists

5

5.1 Machine data

5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10704	DRYRUN_MASK	Activating the dry run feed
10710	PROG_SD_RESET_SAVE_TAB	Setting data to be updated
11410	SUPPRESS_ALARM_MASK	Mask for suppressing special alarms
11550	STOP_MODE_MASK	Inhibit stop events in program section
12000	OVR_AX_IS_GRAY_CODE	Axis feed override switch Gray-coded
12010	OVR_FACTOR_AX_SPEED	Evaluation of the axis feed override switch
12020	OVR_FEED_IS_GRAY_CODE	Path feed override switch Gray-coded
12030	OVR_FACTOR_FEEDRATE	Evaluation of the path feed override switch
12040	OVR_RAPID_IS_GRAY_CODE	Rapid traverse override switch Gray-coded
12050	OVR_FACTOR_RAPID_TRA	Evaluation of the rapid traverse override switch
12060	OVR_SPIND_IS_GRAY_CODE	Spindle override switch Gray-coded
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 for the path override
12100	OVR_FACTOR_LIMIT_BIN	Limit for binary-coded 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
12240	SCALING_SYSTEM_IS_METRIC	Basic system metric

5.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20100	DIAMETER_AX_DEF	Geometry axes with transverse axis functions
20172	COMPRESS_VELO_TOL	Maximum permissible deviation from path feed for compression
20150	GCODE_RESET_VALUES	Reset G groups
20200	CHFRND_MAXNUM_DUMMY_BLOCKS	Maximal number of dummy blocks for chamfer/radii
20201	CHFRND_MODE_MASK	Feed for chamfer/rounding
20660	THREAD_AUTO_LIFTFASTANGLE	Determination of retraction angle (thread cutting)
20750	ALLOW_GO_IN_G96	G0 logic in G96
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour
21220	MULTFEED_ASSIGN_FASTIN	Assignment of input bytes of NCK I/Os for "Multiple feeds in one block"
21230	MULTFEED_STORE_MASK	Store input signals for the "Multiple feeds 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

5.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis
32000	MAX_AX_VELO	Maximum axis velocity
32060	POS_AX_VELO	Initial setting for positioning-axis velocity
32300	MAX_AX_ACCEL	Axis acceleration
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 step
35140	GEAR_STEP_MIN_VELO_LIMIT	Minimum speed of gear step
35160	SPIND_EXTERN_VELO_LIMIT	Spindle-speed limitation via PLC

5.2 Setting data

5.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42010	THREAD_RAMP_DISP	Run-in and run-out path of feed axis with thread cutting
42100	DRY_RUN_FEED	Dry run feed
42101	DRY_RUN_FEED_MODE	Dry run feed mode
42110	DEFAULT_FEED	Default value for path feed
42600	JOG_FEED_PER_REV_SOURCE	In the JOG mode, revolutional feedrate for geometry axes on which a frame with rotation acts
43300	ASSIGN_FEED_PER_RES_SOURCE	Revolutional feedrate for positioning axes

5.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43210	SPIND_MIN_VELO_G25	Minimum spindle speed
43220	SPIND_MAX_VELO_G26	Maximum spindle speed
43230	SPIND_MAX_VELO_LIMS	Spindle-speed limitation with G96

5.3 Signals

5.3.1 Signals to channel

DB number	Byte.Bit	Description
21, ...	0.6	Activate dry run feed
21, ...	4	Feed override
21, ...	5	Rapid traverse override
21, ...	6.0	Feed disable
21, ...	6.6	Rapid traverse override active
21, ...	6.7	Feed override active
21, ...	12.3	Feed stop, geometry axis 1
21, ...	16.3	Feed stop, geometry axis 2
21, ...	20.3	Feed stop, geometry axis 3

5.3.2 Signals from channel

DB number	Byte.Bit	Description
21, ...	24.6	Dry run feed selected
21, ...	25.3	Feed override for rapid traverse selected
21, ...	29.0	Activate fixed feedrate 1 for path/geometry axes
	29.1	Activate fixed feedrate 2 for path/geometry axes
	29.2	Activate fixed feedrate 3 for path/geometry axes
	29.3	Activate fixed feedrate 4 for path/geometry axes

5.3.3 Signals to axis/spindle

DB number	Byte.Bit	Description
31, ...	0	Feed/spindle override
31, ...	1.7	Override active
31, ...	3.2	Activate fixed feedrate 1 for machine axis
	3.3	Activate fixed feedrate 2 for machine axis
	3.4	Activate fixed feedrate 3 for machine axis
	3.5	Activate fixed feedrate 4 for machine axis
31, ...	4.3	Feed stop/spindle stop
31, ...	62.2	Revolutional feedrate active
31, ...	81	F function for positioning axis
31, ...	83.1	Programmed speed too high

Index

A

Acceleration correction
  In the part program/synchronized action, 2-45
Axis/spindle disable DB31, ... DBX1.3, 2-25

C

CFC, 2-2
CFCIN, 2-2
CFTCP, 2-2
Constant cutting rate (G96), 2-5
Constant speed G9, 2-6

D

Deletion of distance-to-go, 2-34
Dry run feedrate, 2-31
Dynamic response of an axis/spindle, 2-43

E

End-of-motion criterion
  Programmable in program/synchronized action, 2-47

F

Feed disable, 2-24
Feedrate disable DB21, ... DBX6.0, 2-24
Feedrate FB
  Non-modal, 2-41
Feedrate for chamfer/rounding, 2-40
Feedrate override switch, 2-25
Feedrate/spindle stop, 2-24
Feedrates
  Dry run feedrate, 2-31
  Feed disable, 2-24
  Feed override, 2-25, 2-30
  Feedrate control, 2-23
  Feedrate/spindle stop, 2-24
  Feedrates in one block, 2-33
  G93, G94, G95, 2-3
  Path feedrate F, 2-1
  Rigid tapping G331/G332, 2-20
  Spindle override, 2-28
  Thread cutting G33, 2-9
Fixed feedrate values, 2-38

G

G96, G961, G97, G971, 2-5

I

Inverse-time feedrate (G93), 2-3

L

Linear feedrate (G94), 2-4

M

Master-slave switchover with G96, G961, 2-9
MD10704, 2-32
MD10710, 2-7
MD11410, 2-10, 2-15
MD11550, 2-21
MD12000, 2-27
MD12010, 2-28
MD12020, 2-26
MD12030, 2-26
MD12040, 2-26
MD12050, 2-26
MD12060, 2-28
MD12070, 2-29
MD12080, 2-30
MD12082, 2-27
MD12100, 2-29
MD12200, 2-39
MD12202, 2-39
MD12204, 2-39
MD12240, 3-1
MD20100, 2-6, 2-9
MD20150, 2-17, 2-18, 2-20, 3-1

MD20172, 1-3
MD20200, 2-41
MD20201, 2-40, 4-1, 4-2
MD20650, 2-12
MD20750, 2-7
MD21200, 2-17
MD21220, 2-34
MD21230, 2-34
MD22140, 3-2
MD22410, 2-31
MD30300, 3-1
MD32000, 2-19
MD32040, 2-4
MD32050, 2-4
MD32060, 2-43
MD32300, 2-19, 2-43, 2-45
MD34990, 2-20
MD35140, 2-8, 2-10
Multiple feedrate values in one block, 2-33

N

Non-modal feedrate FB, 2-41

P

Parameter set
  Servo, 2-43
Programmable end-of-motion criterion, 2-47
Programmable run-in and run-out paths, 2-11
Programmable servo parameter set
  In the part program/synchronized action, 2-48
Programmable single-axis dynamic response, 1-3, 2-42

R

Rapid traverse override switch, 2-25
Revolutional feedrate (G95), 2-4
  IS revolutional feedrate active, 2-5
Run-in and run-out paths, programmable, 2-11
Run-up axes, 2-45

S

SD41100, 2-4
SD42010, 2-12, 2-13
SD42100, 2-31, 2-32
SD42101, 2-32
SD42600, 2-5
SD43210, 2-8, 2-10
SD43220, 2-8
SD43300, 2-5
Servo parameter set, 2-43
  Programmable, 2-48
Single axes, 2-42
Single-axis dynamic response, 2-42
Single-axis dynamic response?, 1-3
Spindle override factor, 2-28
Spindle speed limitation with G96, 2-7

T

Types of feedrate, 1-1

SIEMENS

Brief Description 1

Detailed Description 2

Supplementary Conditions 3

Examples 4

Data Lists 5

SINUMERIK 840D sl/840D/840Di/810D

Tool Compensation (W1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1

2 Detailed Description 2-1

2.1 Tool 2-1
2.1.1 General 2-1
2.1.2 Offset memory structure 2-3
2.1.3 Calculatin of the tool offset 2-5
2.1.4 Address extension for NC addresses T and M 2-5
2.1.5 Free D number allocation 2-7
2.1.6 Correction block for error during tool change 2-13
2.1.7 Definition of the effect of the tool parameters 2-16

2.2 Flat D number structure 2-17
2.2.1 General 2-17
2.2.2 Creating a new D number 2-18
2.2.3 Read D number from the PLC 2-18
2.2.4 D number programming 2-19
2.2.5 Programming the T number 2-23
2.2.6 Programming M6 2-24
2.2.7 Program test 2-24
2.2.8 Tool management or “Flat D numbers” 2-25

2.3 Tool cutting edge 2-26
2.3.1 General 2-26
2.3.2 Tool type (tool parameters) 2-27
2.3.3 Cutting edge position (tool parameter 2) 2-30
2.3.4 Geometry tool length compensation (tool parameters 3 to 5) 2-31
2.3.5 Geometry tool radius compensation (tool parameters 6 to 11) 2-33
2.3.6 Wear tool length compensation (tool parameters 12 to 14) 2-34
2.3.7 Wear tool length compensation (tool parameters 15 to 20) 2-34
2.3.8 Base dimension/adapter dimension tool length compensation (tool parameters 21 to 23) 2-35
2.3.9 Technology – tool clearance angle (tool parameter 24) 2-36
2.3.10 Tools with relevant tool point direction (SW 5 and higher) 2-37

2.4 Tool: Tool radius compensation 2D (TRC) 2-39
2.4.1 General 2-39
2.4.2 Selecting the TRC (G41/G42) 2-40
2.4.3 Approach and retraction behavior (NORM/KONT/KONTC/KONTT) 2-41
2.4.4 Smooth approach and retraction 2-46
2.3.5 Deselecting the TRC (G40) 2-60
2.4.6 Compensation at outside corners 2-60
2.4.7 Compensation at inner corners 2-65
2.4.8 Collision detection and bottleneck detection 2-67
2.4.9 Blocks with variable compensation value 2-69
2.4.10 Keep tool radius compensation constant 2-71
2.4.11 Alarm response 2-74
2.4.12 Intersection procedure for polynomials 2-75
2.4.13 G461/G462: Approach/retract strategy extension 2-75

2.5 Toolholder with orientation capability ... 2-80
2.5.1 General ... 2-80
2.5.2 Kinematic interaction and machine design ... 2-87
2.5.3 Inclined machining with 3 + 2 axes ... 2-95
2.5.4 Machine with rotary table ... 2-96
2.4.5 Procedure when using orientational toolholders ... 2-99
2.5.6 Programming ... 2-104
2.5.7 Supplementary conditions and control system response for orientation ... 2-105

2.6 Incrementally programmed compensation values ... 2-108
2.6.1 G91 extension ... 2-108
2.6.2 Machining in direction of tool orientation ... 2-110

2.7 Basic tool orientation ... 2-111

2.8 Special handling of tool compensations ... 2-115
2.8.1 Relevant setting data ... 2-115
2.8.2 Mirroring of tool lengths (SD 42900: MIRROR_TOOL_LENGTH) ... 2-116
2.8.3 Mirroring of wear lengths (SD 42920: WEAR_SIGN_CUTPOS) ... 2-117
2.8.4 Tool length and plane change (SD 42940: TOOL_LENGTH_CONST) ... 2-119
2.8.5 Tool type (SD 42950: TOOL_LENGTH_TYPE) ... 2-120
2.8.6 Temperature offsets in the tool direction (SD 42960: TOOL_TEMP_COMP) ... 2-121
2.8.7 Tool lengths in the WCS allowing for the orientation ... 2-121
2.8.8 Tool length offsets in tool direction ... 2-122

2.9 Sum offsets and setup offsets ... 2-127
2.9.1 General ... 2-127
2.9.2 Description of function ... 2-128
2.9.3 Activation ... 2-131
2.9.4 Examples ... 2-137
2.9.5 Upgrades for Tool Length Determination ... 2-138

2.10 Working with tool environments ... 2-138
2.10.1 General ... 2-138
2.10.2 Saving with TOOLENV ... 2-139
2.10.3 Delete tool environment ... 2-141
2.10.4 How many environments and which ones are saved? ... 2-143
2.10.5 Read T, D, DL from a tool environment ... 2-143
2.10.6 Read tool lengths, tool length components ... 2-144

2.11 Tool lengths L1, L2, L3 assignment: LENTOAX ... 2-149

3 Supplementary conditions ... 3-1

3.1 Flat D number structure ... 3-1

3.2 SD42935 expansions ... 3-2

4 Examples ... 4-1

4.1 Toolholder with orientation capability ... 4-1
4.1.1 Example of orientational toolholders ... 4-1
4.1.2 Example of orientational toolholders with rotary table ... 4-2
4.1.3 Example basic tool orientation ... 4-5
4.1.4 Calculation of offsets according to location of use or workpiece ... 4-5

4.2 Examples 3–6: SETTCOR function for tool environments ... 4-8

5 Data lists ... 5-1

5.1 Machine data ... 5-1
5.1.1 NC-specific machine data ... 5-1
5.1.2 Channelspecific machine data ... 5-2
5.1.3 Axis-/spindlespecific machine data ... 5-3

5.2 Setting data 5-4
5.2.1 Channelspecific setting data 5-4
5.3 Signals 5-5
5.3.1 Signals from channel 5-5
Index **Index-1**

Brief Description

1

Calculating tool compensation data

The SINUMERIK 840D/810D controls can be used to calculate the following tool compensation data:

- Length compensation
- Radius compensation
- Storage of tool data in a flexible tool compensation 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

Type		
Geometry: Length	Wear: Length	Base/adapter dimension
Geometry: Radius	Wear: Radius	
Technology		

- 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 outside 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 5Axis transformation is not required for this case.

References:
/FB3/ Description of Functions, Special Functions; 3- to 5Axis Transformation (F2)

Appropriate selection of the tool data and toolholder data describes the kinematics for the control such that it can make allowance for the tool length compensation.
The control can take some of the description data direct from the current frame.

Note

Please refer to the following documentation for more information about tools and tool compensations and a full technical description of the general and specific programming features for tool compensation (TLC and TRC):
References:
/PA/ Programming Guide, Fundamentals.

Flat/Unique D number structure

Compensations can be selected via unique D numbers with management function.

Special handling of tool compensations

Setting data
SD42900 $SC_MIRROR_TOOL_LENGTH
to
SD42960 $SC_TOOL_TEMP_COMP
can be used to control the evaluation of the signs for tool length and wear.

The same applies to the response of the wear components when mirroring geometry axes or changing the machining plane via setting data.

References:
/PG/ Programming Guide, Fundamentals; Tool Compensations

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 and 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 control calculates an intersection by extending the compensation curve of this block with a circle whose center point coincides with the end point of the non-corrected 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 control 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
- Deletion
- 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.

Detailed Description 2

2.1 Tool

2.1.1 General

Select tool

A tool is selected in the program with the T function.

The setting in MD 22550

$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

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

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.

Machine data:
MD22560 $MC_TOOL_CHANGE_M_CODE (M function for tool change)
is used to remove the old tool from the spindle and load the new tool onto the spindle.

This tool change must be programmed with the M function `M06`, in accordance with DIN 66025.

The next tool is preselected using machine data:
MD20121 $MC_TOOL_PRESEL_RESET_VALUE
Allowances are made for its tool length compensation values on RESET and during power-up, in accordance with machine data:
MD20110 $MC_RESET_MODE_MASK.

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`.

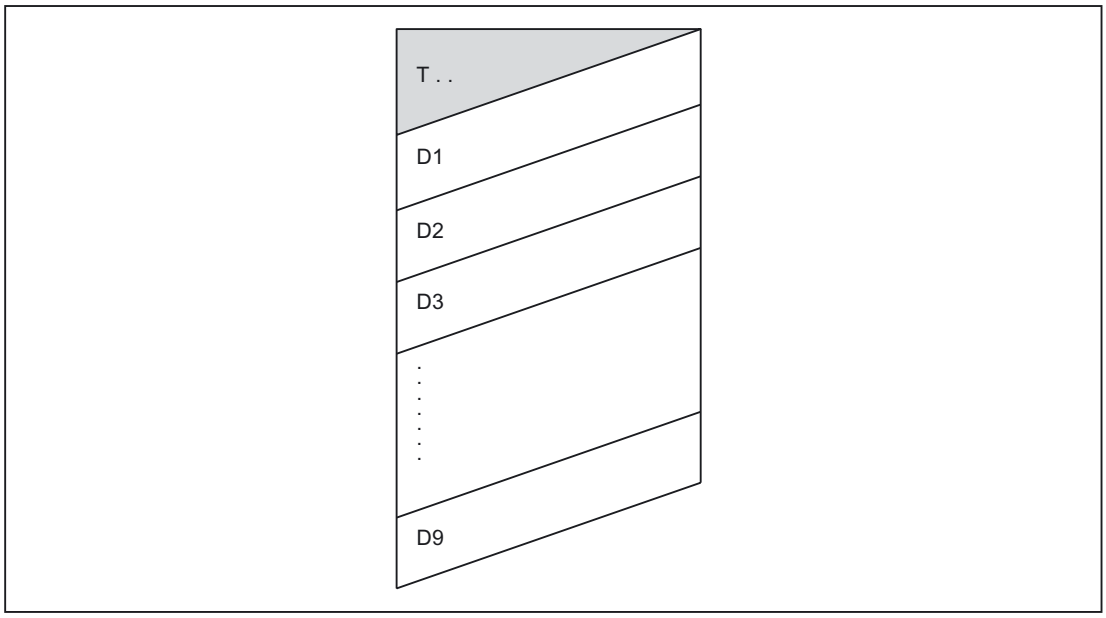


Fig. 2-1 Example of a tool T... with 9 cutting edges (D1 to D9)

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 predefined using machine data:
  MD20270 $MC_CUTTING_EDGE_DEFAULT:

= 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 compensations

`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, tool length compensation is applied with the first axis motion, in which it is operative.

- **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).

2.1.2 Offset memory structure

Tool compensation memory size

Each channel can have a dedicated tool compensation memory (TO unit).

Machine data:
MD28085 $MC_MM_LINK_TOA_UNIT (assignment of TO unit to a channel)
is set to define the TO memory, which exists for the appropriate channel.

The maximum number of cutting edges for all tools managed by the NCK
is set with machine data:
MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA
(number of cutting edges in the 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 compensation memory is created. The individual values in the compensation 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.

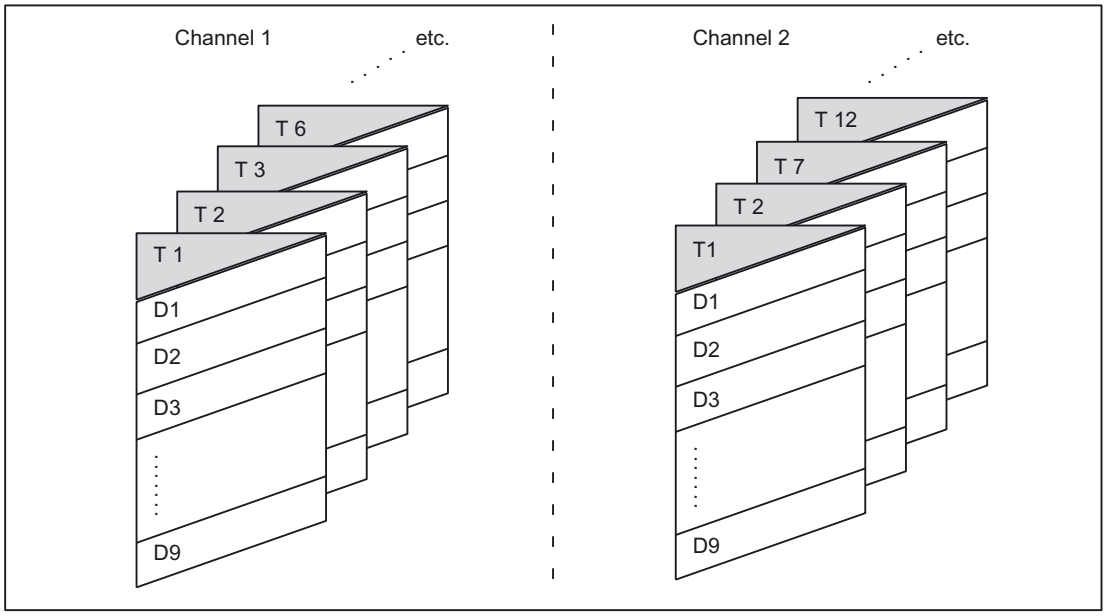
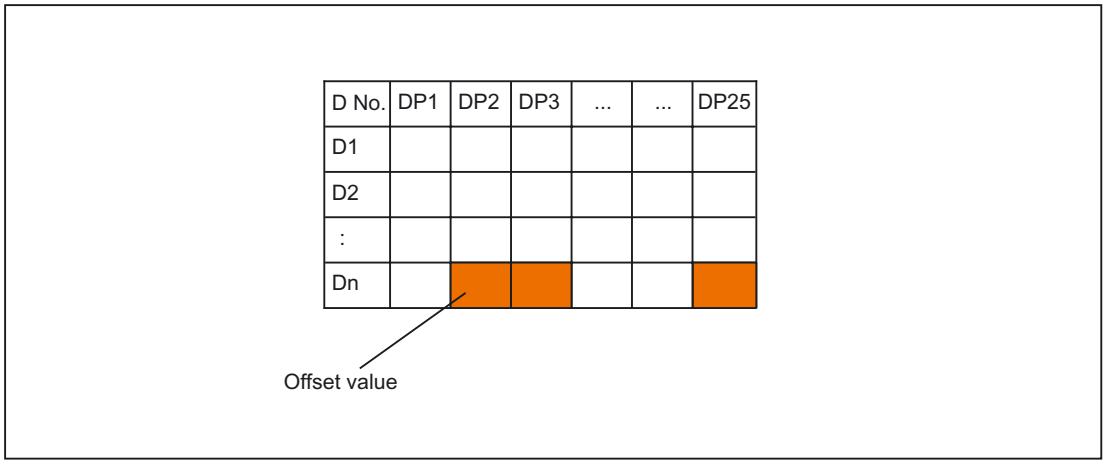


Fig. 2-2 Example of a tool compensation memory structure for 2 channels

2.1.3 Calculation of the tool offset

D No.

The **D no.** is sufficient for calculating the tool compensations (can be set via MD).



Example:

The above compensation block is to be calculated in the NC.

Part program call:

```
...
Dn
```

2.1.4 Address extension for NC addresses T and M

MD20096

Machine data:
MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO
can be used to define whether the address extension of T and M is to be interpreted as a spindle number even if tool management is **not** activated.
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.

If machine data:
MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE,
the programmed D address refers to the active tool in relation to the master spindle (same as for tool management function).

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.

If machine data:
MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE
(spindle number as address extension), only the T value programmed with reference to the master spindle is displayed as programmed/active.

Example

The example below shows the effect of the machine data.

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.

- If tool management is not active and
  MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = **FALSE**, `D4` refers to tool "50".

  The address extensions of neither T nor Mare evaluated in the NCK.

  Each tool change command defines a new path compensation.

- If tool management is not active and
  MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = **TRUE**, `D4` refers to tool "5" (as with active tool management).

  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 meaning of the extension is defined by the user (in the PLC user program).

2.1.5 Free D number allocation

"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:

- Machine data:
  MD18105 $MN_MM_MAX_CUTTING_EDGE_NO
  is used to define the maximum permissible D number.

  The default value is 9, in order to maintain compatibility with existing applications.
- The number of cutting edges (or compensation data sets) can also be defined on a **tool-specific** basis using machine data:
  MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL.

  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).

If
MD18105 ≤ MD18106,
the CE cutting edge number is equal to the D number
(for compatibility with the previous response).
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

If the maximum cutting edge number (MD18105) is greater than the maximum number of cutting edges per tool (MD18106), the following commands are available:

• 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:
/PA/ Programming Guide, 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).

Examples

MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 1

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

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:

```
$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

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:

```
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:

```
$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,
D number 2 of cutting edge 1 cannot have the same name as the D number of cutting edge 3, i.e.,
SETDNO(1, 1, 17)
returns the status = FALSE.

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:

```
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:

```
SETDNO( 1, 1,     ;T=1, cutting edge 1 receives the (new) D number 100
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:

```
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.
```

2.1.6 Correction block for 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 can be programmed in a variety of ways, depending on machine data: MD22550 $MC_TOOL_CHANGE_MODE:

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 machine data:
MD 22550 $MC_TOOL_CHANGE_MODE).

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 MMC.
- 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 intervention options as for = 0.
Missing data can be reloaded. 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 if MD22562 $MC_TOOL_CHANGE_ERROR_MODE bit0 = 1).

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

During program testing, the program also stops immediately at the Txx block containing the error if MD22562 $MC_TOOL_CHANGE_ERROR_MODE bit0=1,
to allow the program to be corrected.

2.1.7 Definition of the effect of the tool parameters

MD20360

Machine data:
MD20360 $MC_TOOL_PARAMETER_DEF_MASK
can control the effect of the tool parameters on the transverse axis in connection with diameter programming. 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:
MD11346 $MN_HANDWHEEL_TRUE_DISTANCE = 1

Define the DRF compensation in the transverse axis as a diameter compensation with:
MD20360 $MC_TOOL_PARAMETER_DEF_MASK bit 9 = 1

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:
/FB/ Description of Functions, Extended Functions; Manual Travel and Handwheel Travel (H1)
/PG/ Programming Guide, Fundamentals (the Programming Guide contains a complete technical description of axis-specific deselection of the DRF compensation.)

2.2 Flat D number structure

2.2.1 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

Machine data:
MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE
can be used to define the type of D number management that is active:

Value = 0	As previously = default setting
Value = 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 the MMC.

2.2.2 Creating a new D number

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.

Machine data:
MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA
is used to set the maximum number of D or compensation data blocks (max. 600).

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.
The same also applies in reverse for transfer.

D range

1 - 99 999 999

2.2.3 Read D number from the PLC

Reading from system variables

The programmer has various options for specifying the D number in the part program.

One option is to read it from the system variables
$A_DNO[n] ; (n=1 to 9).
("DNO" stands for D number.)

Example

Compensation selection in the part program
`D9` is written in the program. With `$A_DND[9]`, the absolute D number stored in the 9th table location at the time of the call is read.

System variable $A_DNO[n]; (n=1 to 9)	
Meaning	The variable reads the value of the nth position from a table containing the D numbers. The variable is typically used for indirect D programming (via parameters): D=$A_DNO[1] The variable reads a field element (=the actual D number) in the D number table of VDI. The PLC writes the matching D number for this T number to the field after the NCK has output the auxiliary function T to the PLC. Field elements have the value **zero** if no D number has been defined for them. A maximum of 9 different D numbers can be stored from the PLC for 1 T in the VDI interface. D number values can range between 0 and 32000, i.e., the table may contain gaps.
Data type	INT
Value range	99 999 999
Indices	n=1-9. Index n indicates the position in the D number table that is to be read.
Access	Read in part program
Preprocessor stop	Yes
Synchronized action	Read in synchronized actions
Validity	Only relevant in conjunction with the "Flat D numbers" function or tool management on PLC.

2.2.4 D number programming

MD18102

Machine data:
MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE
can be used to set how the D number is programmed.

D0 still contains the previous meaning, "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.

Direct, absolute programming

Programming in the part program is carried out as before. Only the value range of the programmed D number is increased.

Example 1:

MD22550 $MC_TOOL_CHANGE_MODE = 0
MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE = 1
MD20270 $MC_CUTTING_EDGE_DEFAULT = –1

```
...
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

```
T1
T2
T3
D777              No wait, D777 is activated,
                  T3 = programmed and active tool in display
                  D777 = programmed and active compensation
```

Note

The tool change and the assignment of a D compensation to an actual tool are not the responsibility of NCK.

Indirectly parameterized program

D=$A_DNO[n]
Meaning: Select the D number entered at position n (=1 to 9) in the VDI D number table.

The $A_DNO variable is read during run-up with advance forced synchronization with the main run.

The D numbers available in the VDI interface can be read with $A_DND.

Note

Before the NCK main run reads the D number, it must be ensured beyond doubt that the PLC has written the D numbers that match the previously programmed T value. This mechanism (in the NCK) usually initiates a wait for PLC in the NCK.

Indirect indexed programming

Dn or **D=n**

Meaning: Select the D number entered at position n (=1 to 9) in the VDI D number table. Although this syntax is identical to the conventional one, it activates the compensation block for the D number determined for index n internally.

Example of indirect D programming sequence:

MD20270 = 0

MD22550 = 0

Spindle no. 2 is not the master spindle

Part program	Action
T4	Tool change command: NCK outputs the value 4 as T auxiliary function. PLC evaluates it and provides the associated absolute D numbers in the VDI. The PLC only does this if the address extension received is the number of the master spindle. A simple communication protocol allows the NCK to detect whether the associated D numbers are available for the subsequent programmed D.
X1	
...	
D1	The NCK feed is synchronized with the main run, checks whether the new D numbers are present in the VDI (and waits if necessary) and accepts the desired (absolute) D number at position 1 (D1), e.g., the number 4711. The NCK now determines the compensation block 4711 and calculates the geometry. The NC copies the entire content of the D number table to the VDI interface (each time a tool change command is detected).
D=$P_DNO[2]	Contents correspond to the programming of D2 (for indirect programming)

Part program	Action
D3	The NC accesses the absolute D number of location 3; a synchronized action is not required, because a new T has not been programmed.
T2 = 12000000	The NC does not detect a tool change, because spindle no. 2 is not the number of the master spindle.
D4	The NC accesses the absolute D number of location 4; a synchronized action is not required, because a new T has not been programmed.

Before outputting the T number to the PLC, the NCK stores the status of the VDI with reference to the queued D number(s). A counter is contained in the VDI for this purpose. The PLC increments the counter each time the D number is refreshed. If necessary the next D number request from the NCK waits until the counter value has changed. The VDI then contains the new D number table, which belongs to the previously programmed value.

Note

D can only be programmed without an address extension. D always refers to the master spindle. T can be programmed with an address extension. PLC only has to write the D compensations to VDI for T values or `M6` commands with reference to the master spindle. The NCK assumes that this is the case when performing synchronization actions during reading of the D numbers from the VDI.

Example

```
MD22550 TOOL_CHANGE_MODE = 1
MD22560 TOOL_CHANGE_M_CODE = 6          ; i.e., tool change with M6 programming
                                          Spindle number 3 is not the master
                                          spindle
T1
M6                                      ; Tool change command
T3
T3 = 11
D = 2                                   ; Wait until the counter in the VDI has
                                          changed;
                                          this is the signal that the PLC has
                                          written the D values for T1 to the D
                                          number table (e.g., at position 2 = 4711;
                                          then activate compensation 4711 in NCK).
                                          T3 = programmed T
                                          T1 = active T in the display
```

Delete D no. via part program

- **With** flat D number

  $TC_DP1[d] = 0

  Compensation data set is deleted from NCK with number d.

  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

  Delete all D compensations from NCK.

Active compensation data blocks (D numbers) cannot be deleted. It may, therefore, 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:

- MD20270 $MC_CUTTING_EDGE_DEFAULT
- MD20130 $MC_CUTTING_EDGE_RESET_VALUE
- MD20120 $MC_TOOL_RESET_VALUE
- MD20121 $MC_TOOL_PRESEL_RESET_VALUE
- MD22550 $MC_TOOL_CHANGE_MODE
- MD22560 $MC_TOOL_CHANGE_M_CODE
- MD20110 $MC_RESET_MODE_MASK
- MD20112 $MC_START_MODE_MASK

2.2.5 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 with 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.

2.2.6 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.

Machine data:
MD22550 $MC_TOOL_CHANGE_MODE
is used to define whether the tool change command is performed with an M function.
T is used as the tool preparation command.

Machine data:
MD22560 $MC_TOOL_CHANGE_M_CODE
is used to define the name of the M function for the 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
```

2.2.7 Program test

MD20110

Machine data:
MD20110 $MC_RESET_MODE_MASK, bit 3
can be used to define that the active tool and the tool compensation are transferred as follows:

Bit 3=1	From the last test program to finish in test mode
Bit 3=0	From the last program to finish before activation of the program test

Prerequisite

Bits 0 and 6 of MD20110 must be enabled.

2.2.8 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.

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 time-critical 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.

2.3 Tool cutting edge

2.3.1 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 this purpose:

Tool parameter	Meaning	Note	Reserved for expansions
1	Tool type		
2	Cutting edge position	only for turning tools	
Geometry tool length compensation			
3	Length 1		
4	Length 2		
5	Length 3		
Geometry tool radius compensation			
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	for 3D face milling	
11	Angle 2	for 3D face milling	
Wear tool length compensation			
12	Length 1		
13	Length 2		
14	Length 3		

Tool parameter	Meaning	Note	Reserved for expansions
Wear tool radius compensation			
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	for 3D face milling	
20	Angle 2	for 3D face milling	
Tool base dimension/adapter dimension tool length compensation			
21	Basic length 1		
22	Basic length 2		
23	Basic length 3		
Technology			
24	Undercut angle	only for turning tools	
25			Reserved*

* "Reserved" means that this tool parameter on the 840D/810D is not used (reserved for expansions).

Milling cutter types 111, 120, 121, 130, 155, 156 and 157 3D are given special treatment for face milling by evaluating tool parameters (1 - 23).

For more information about the different tool types, see
Subsection 7.7.1 Tool type (tool parameters) and
References:
/FB1/ Description of Functions, Basic Machine; Tool Compensation (W1),
Chapter "Tool type (tool parameters)"
/FB3/ Description of Functions, Special Functions; 3D Tool Radius Compensation (W5)

2.3.2 Tool type (tool parameters)

Meaning

A 3digit number is used to define the tool type. The operator/machine setter/programmer selects the tool type. This determines further components such as geometry, wear and tool base dimensions in advance. The tool type has no meaning in the grinding and turning tool groups. Non-listed numbers are also permitted, in particular with grinding tools (400-499).

Table of the main tool parameters available (only the tool type is necessary)

Tool type	CLDATA*1 "Cutter Location Data" Cutter location data in accordance with DIN 66215, BI 1GWPS*2 Grinding wheel peripheral speed	Length of cutting edge	Geometry – Length 1	– Length 2	– Length 3	Geometry – Radius – Length	– Length	– Radius	– Min. wheel radius	– Min./curr. wheel width	– Angle of inclined wheel	Wear – Length	Wear – Radius	Tool base dimension – Length 1 to 3	– Peripheral speed	Maximum speed	Undercut angle
Milling tools and drilling tools (all other)																	
100	Milling tool 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			
121	End mill with corner rounding		x					x				x	x	x			
130	Angle head mill		x	x	x			x				x	x	x			
131	Angle head mill with corner rounding		x	x	x			x				x	x	x			
140	Facing tool		x					x				x	x	x			
145	Thread cutter		x					x				x	x	x			
150	Side mill		x					x				x	x	x			
155	Truncated cone mill		x	x	x			x				x	x	x			
200	Twist drill		x									x		x			
205	Drill		x									x		x			
210	Boring bar		x									x		x			
220	Center drill		x									x		x			
230	Countersink		x									x		x			
231	Counterbore		x									x		x			
240	Tap regular thread		x									x		x			
241	Tap fine thread		x									x		x			
242	Tap/Whitworth thread		x									x		x			
250	Reamer		x									x		x			
Grinding tools and turning tools (400 – 599)																	
400	Surface grinding wheel		x	x	x		x	x			x	x	x	x	x		
401	Surface grinding wheel with monitoring		x	x	x		x	x	x	x	x	x	x	x	x	x	
403	Same as 401 but without tool base dimension for GWPS*2		x	x	x		x	x	x	x	x	x	x	x		x	
410	Face grinding wheel		x	x	x		x	x				x	x	x	x		
411	Face grinding wheel with monitoring		x	x	x		x	x	x	x		x	x	x	x	x	
413	Same as 411 but without tool base dimension for GWPS*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	Finishing tool	x	x	x	x			x				x	x	x			x
520	Plunge cutter	x	x	x	x			x				x	x	x			x
530	Parting 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			

Special points to be noted

- 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 compensation data for slotting saw type

The following compensation data (TOA data) can be specified for the slotting saw tool type (type 700):

	Geometry	Wear	Base	
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

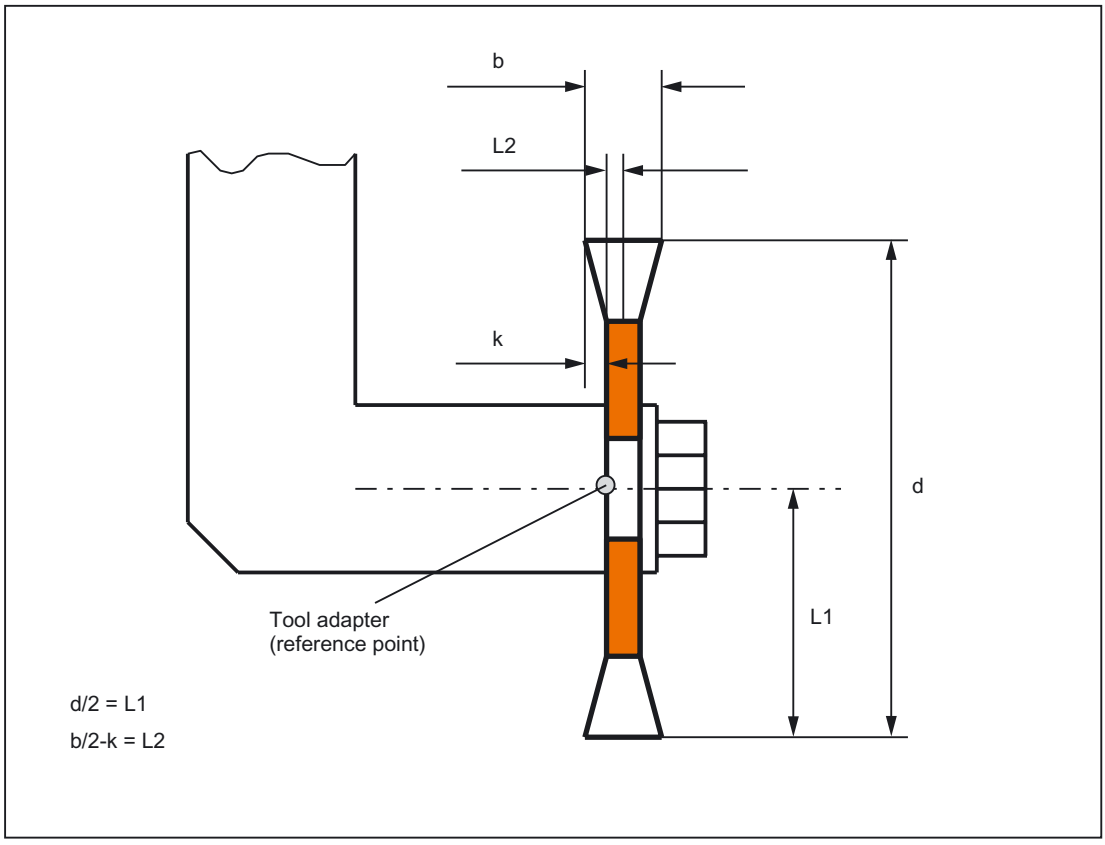


Fig. 2-3 Geometry of slotting saw (analogous to angle head cutter)

The width of the saw blade is accounted for with `G40` to `G42`:

`G40`	No saw blade compensation
`G41`	Saw blade compensation left
`G42`	Saw blade compensation right

2.3.3 Cutting edge position (tool parameter 2)

Tool radius compensation for turning tools

The control also needs the following parameters in order to calculate the tool radius compensation for turning tools (tool type 5xx):

- Cutting edge position (tool parameter 2)
- Cutting edge radius (tool parameter 8)

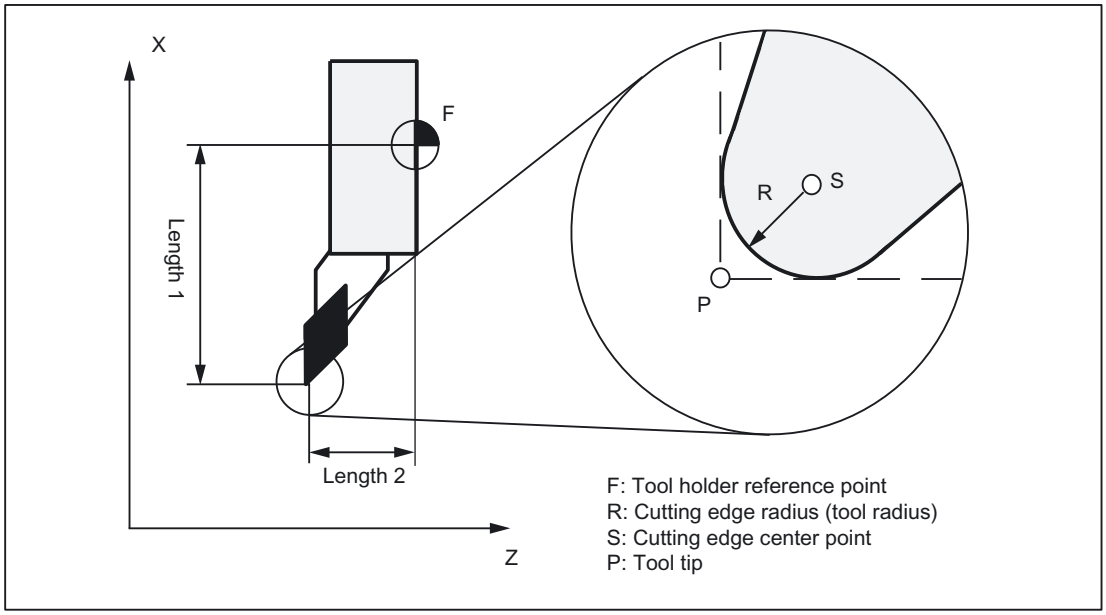


Fig. 2-4 Dimensions for turning tools: Turning tool

Length of cutting edge

The cutting edge position describes the position of the tool tip P in relation to the cutting edge center point S. The cutting edge position is entered in tool parameter 2 (shown as P2 in the figure).

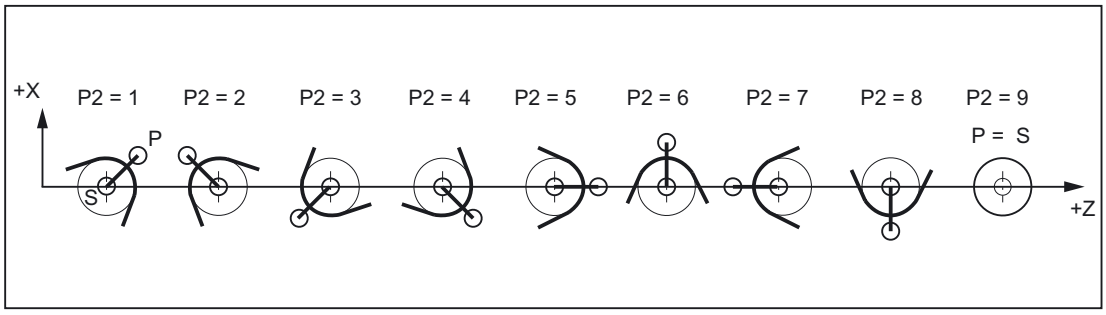


Fig. 2-5 Tool parameter 2c (P2): Machining behind the turning center

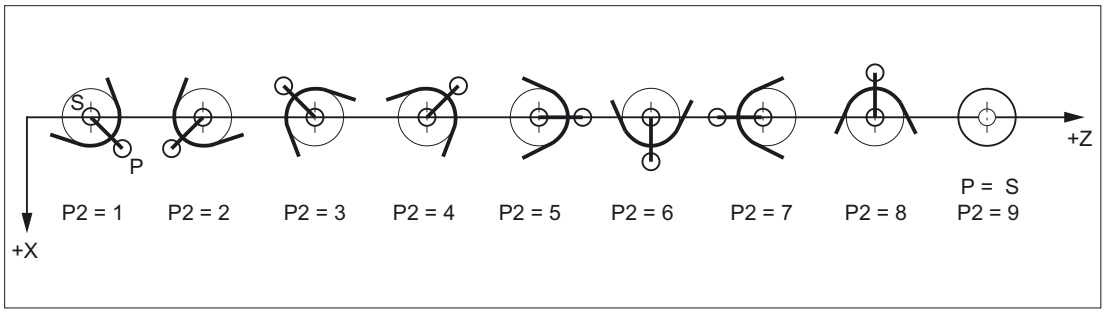


Fig. 2-6 Tool parameter 2 (P2): Machining in front of the turning center

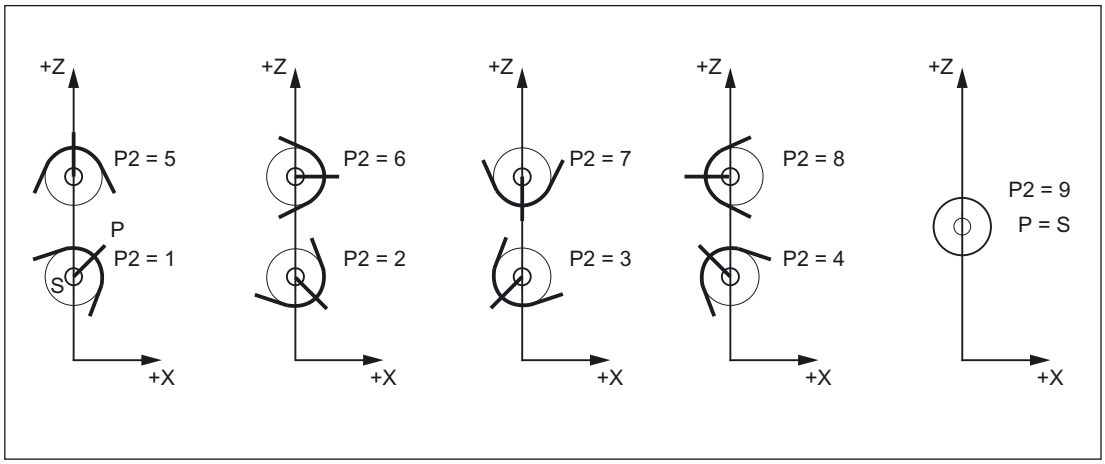


Fig. 2-7 Tool parameter 2 (P2): Cutting edge position for vertical boring and turning mills

Special feature

- 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.

2.3.4 Geometry tool length compensation (tool parameters 3 to 5)

Tool length 1 to 3

The lengths of the tools for tool length compensation are entered as tool lengths 1 to 3 (tool parameters 3 to 5). The following length specifications must be entered as a minimum for each tool type:

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

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 than the minimum required are entered in tool parameters 3 to 5 for one tool type (only tool length 1 is required for tool type 2xx), these additional tool lengths are calculated in the geometry axes without an alarm.

Note

Please refer to the following documentation for information about entering tool dimensions (lengths) in tool parameters 3 to 4 (tool lengths 1 to 3) and how these are calculated in the three geometry axes:
References:
/BA/ Operator's Guide.

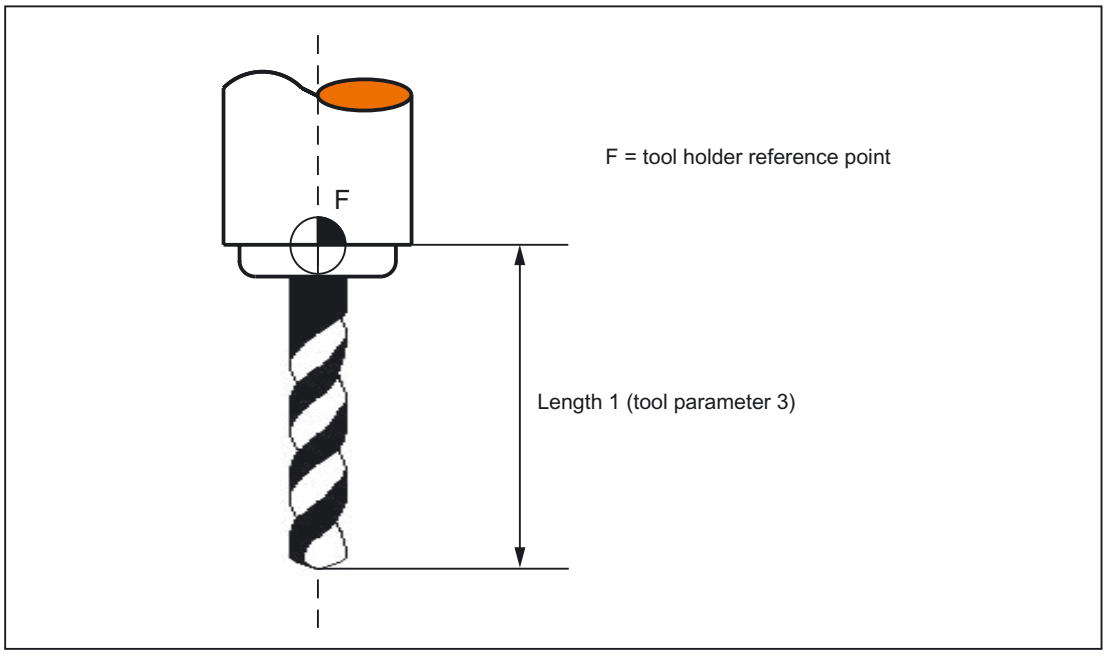


Fig. 2-8 Twist drill (tool type 200) with tool length (tool parameter 3)

Special feature

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.

2.3.5 Geometry tool radius compensation (tool parameters 6 to 11)

Tool geometry

Geometry tool radius compensation defines the shape of the tool.

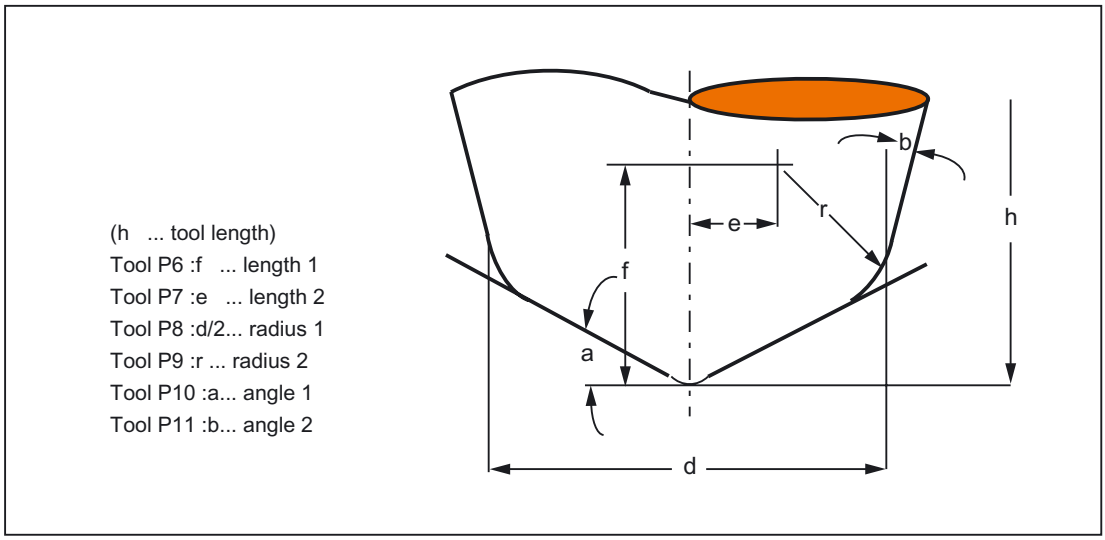


Fig. 2-9 Description of the tool geometry

Note

The tool description as given in the figure is required only for 3D face milling.

Otherwise,

on SINUMERIK 840D/810D, only tool parameter 6 (tool radius 1) is used from tool parameters 6 to 11.

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 tool radius compensation in the three geometry axes:
References:
/PA/ Programming Guide, Fundamentals
/FB3/ Description of Functions, Special Functions; 3D Tool Radius Compensation (W5).

Tool length 1	Not used	
Tool length 2	Not used	
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).	
Tool radius 2 Tool angle 1 Tool angle 2	Not used	

2.3.6 Wear tool length compensation (tool parameters 12 to 14)

Meaning

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 size 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 a finishing allowance.

2.3.7 Wear tool length compensation (tool parameters 15 to 20)

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.

Meaning

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 size can change due to:

- Tool wear caused during service life by machining
- Definition of a finishing allowance.

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.

2.3.8 Base dimension/adapter dimension tool length compensation (tool parameters 21 to 23)

Meaning

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.

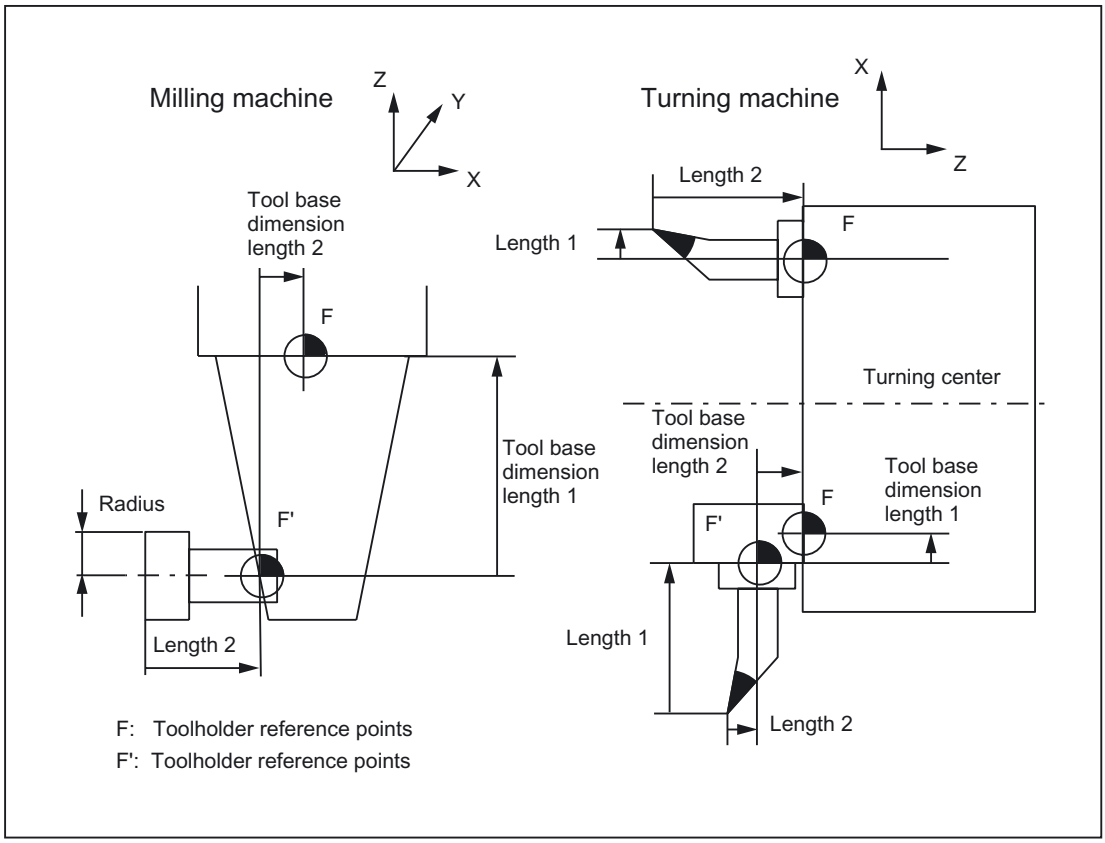


Fig. 2-10 Example applications for base-dimension/adapter-dimension TLC

Basic length 1 to 3

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.

Note

Please refer to the following documentation for more information about base-dimension/adapter-dimension tool length compensation:
References:
/PA/ Programming Guide, Fundamentals.

2.3.9 Technology – tool clearance angle (tool parameter 24)

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 degrees with no leading sign) is entered in tool parameter 24 as the tool clearance angle.

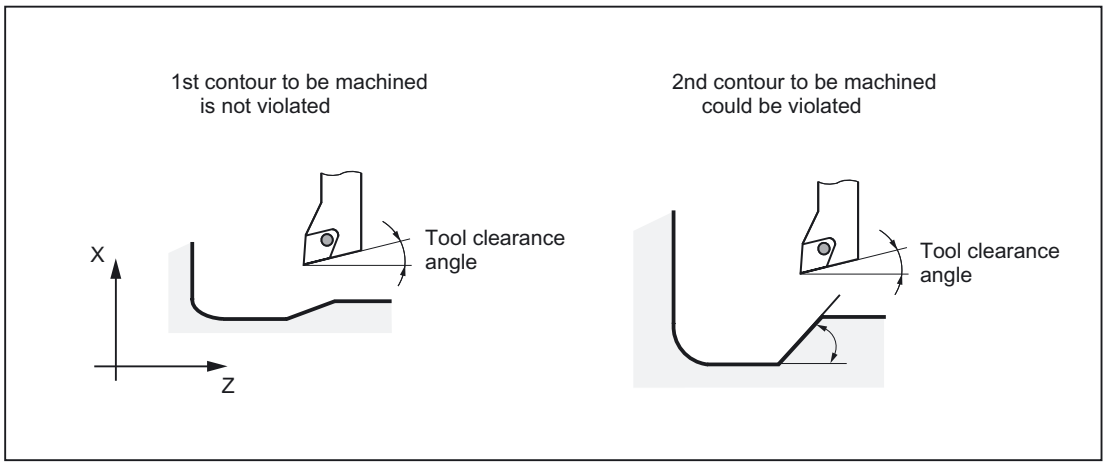


Fig. 2-11 Tool clearance angle of the turning tool during relief cutting

Longitudinal/face

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.

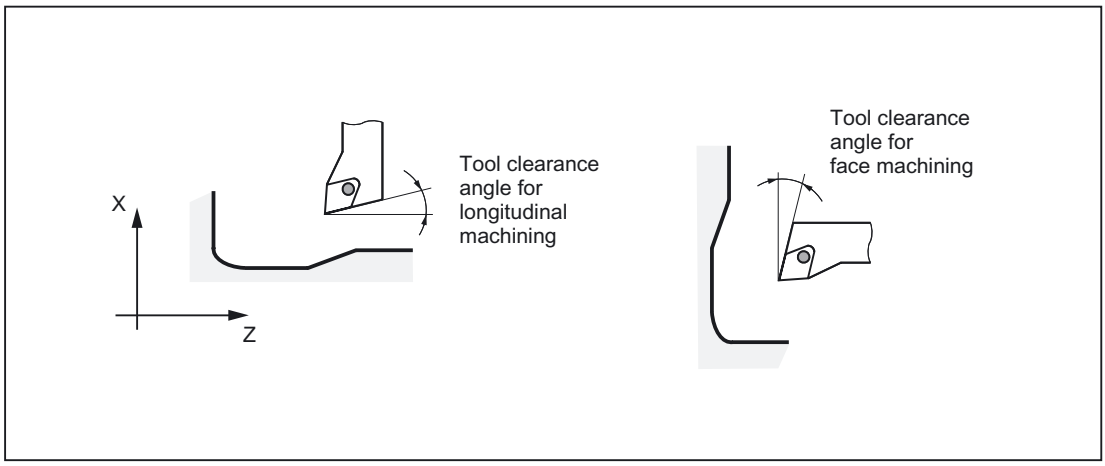


Fig. 2-12 Tool clearance angle for longitudinal and face machining

Note

If a tool clearance angle (tool parameter 24) of zero is entered, relief cutting is not monitored in the turning cycles.

Please refer to the following documentation for more detailed information about the tool clearance angle:
References:
/PAZ/ Programming Guide, Cycles.

2.3.10 Tools with relevant tool point direction (SW 5 and higher)

Up to SW 4.x

The change from `G40` to `G41`/`G42` or vice versa is treated as a tool change for tools with relevant tool point direction. This causes a preprocessor stop when a transformation (e.g., `TRANSMIT`) is active, and may lead to deviations from the intended part contour.

SW 5 and higher

The following changes have been made:

1. The change from `G40` to `G41`/`G42` or vice versa is no longer treated as a tool change. Therefore, a preprocessor stop no longer occurs with `Transmit`.

2. 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 tool point direction are identical. Deviations from previous behavior occur only in relatively rare cases where the approach or retraction block does not intersect with an adjacent motion block:

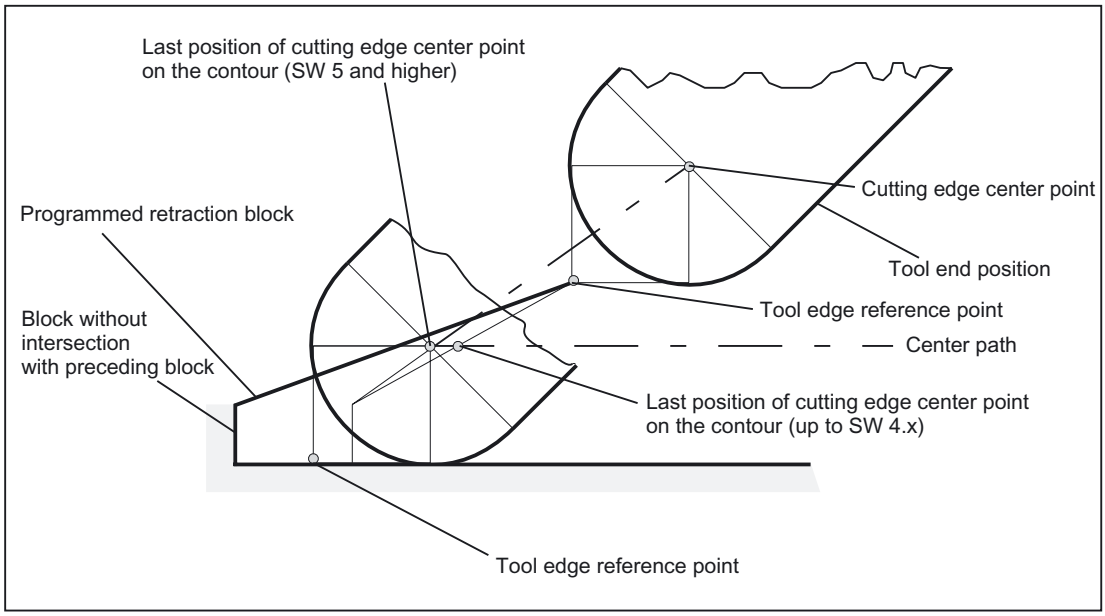


Fig. 2-13 Retraction behavior for tool with relevant tool point direction

1. 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, a change is permitted even when a transformation (e.g., `Transmit`) is active, in contrast to previous versions.
2. 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 compensation. Tools with a relevant tool point direction 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 tool point direction 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 tool point direction parameter is ignored.)

2.4 Tool: Tool radius compensation 2D (TRC)

2.4.1 General

Note

For TRC (tool radius compensation), see:
References:
/PA/ Programming Guide, Fundamentals.
Only the Programming Guide contains a complete technical description of TRC (Tool Radius Compensation) 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 compensation memory. Tool radius compensation can be used to calculate the equidistant path to the programmed contour from the current tool radius.

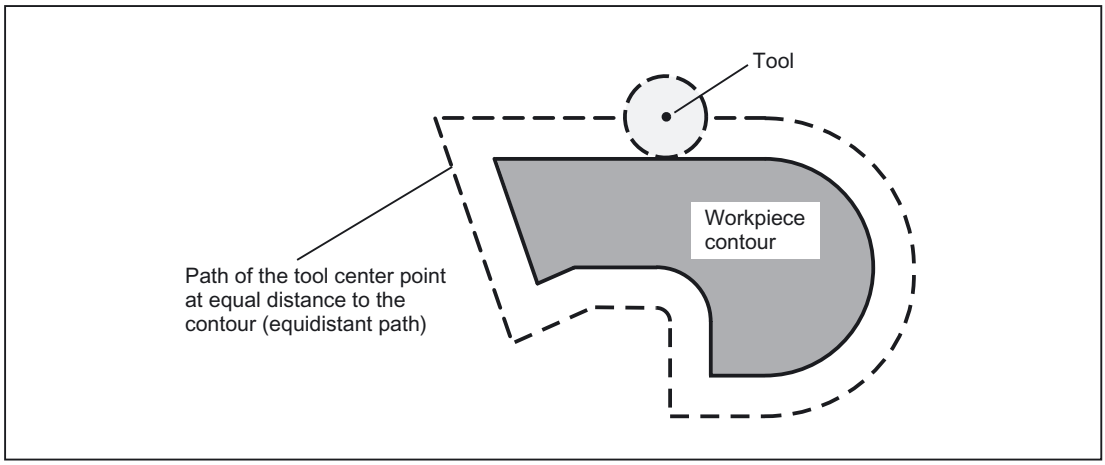


Fig. 2-14 Workpiece contour (geometry) with equidistant path

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

2.4.2 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 right-hand side of the programmed contour in the direction of motion.

• G41	...	TRC on the left-hand side of the contour in the direction of motion
• G42	...	TRC on the right-hand 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 max. number of dummy blocks can be predefined in machine data:
MD20250 $MC_CUTCOM_MAXNUM_DUMMY_BLOCKS
(number of blocks with no traversing movements with 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.

2.4.3 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):

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.

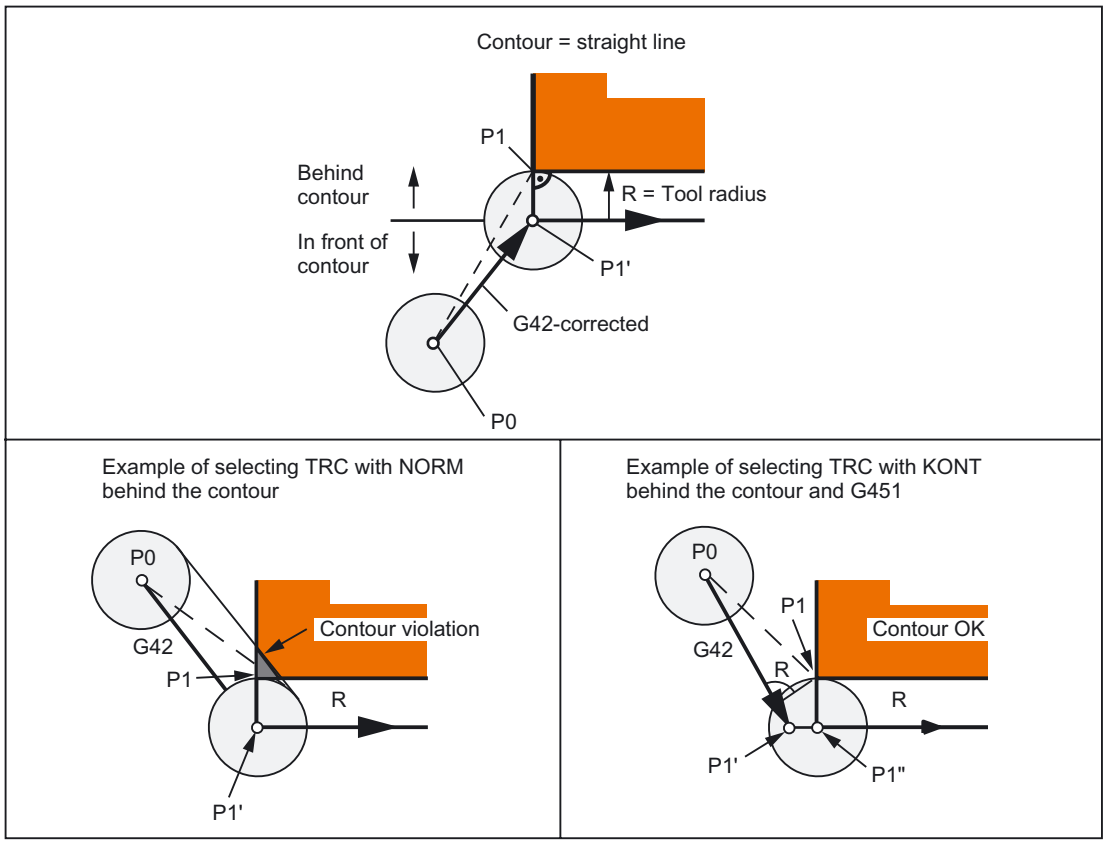


Fig. 2-15 Example for selecting TRC with KONT or NORM in front of and behind the contour

- 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.

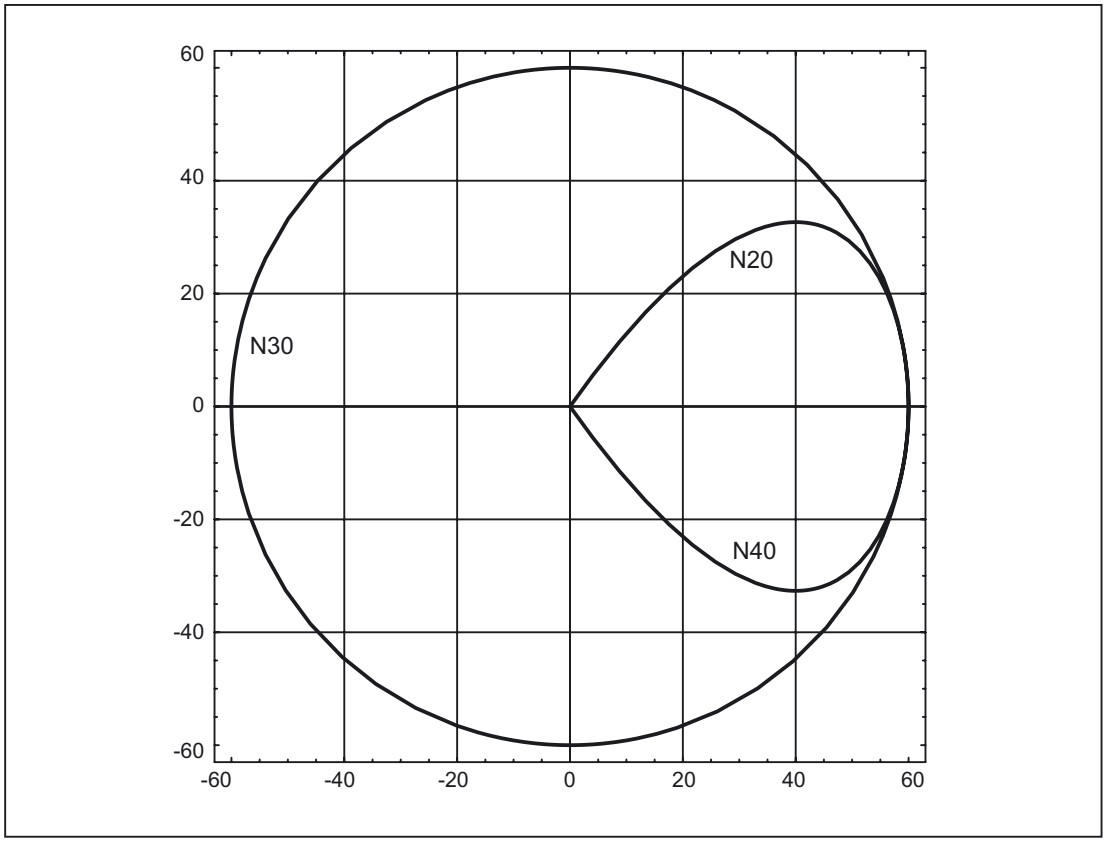


Fig. 2-16 Approach and retraction with constant curvature during inside machining of a full circle. Projection in the X-Y plane.

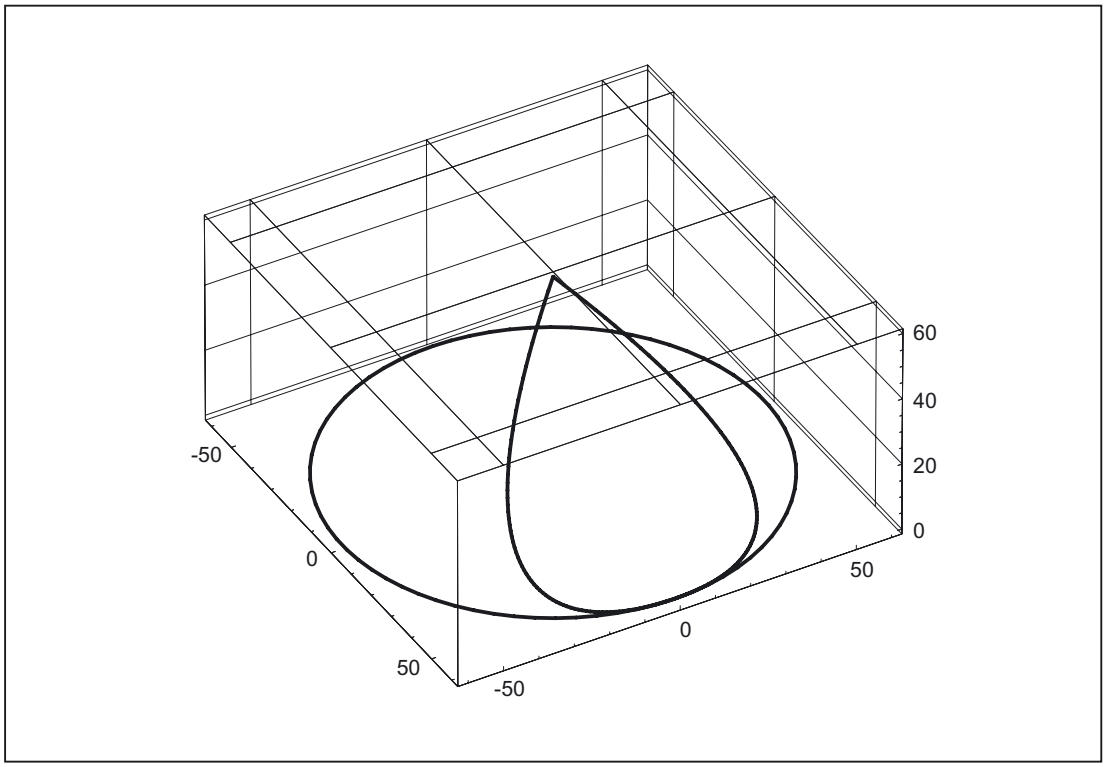


Fig. 2-17 Approach and retraction with constant curvature during inside machining of a full circle. 3D representation.

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.**

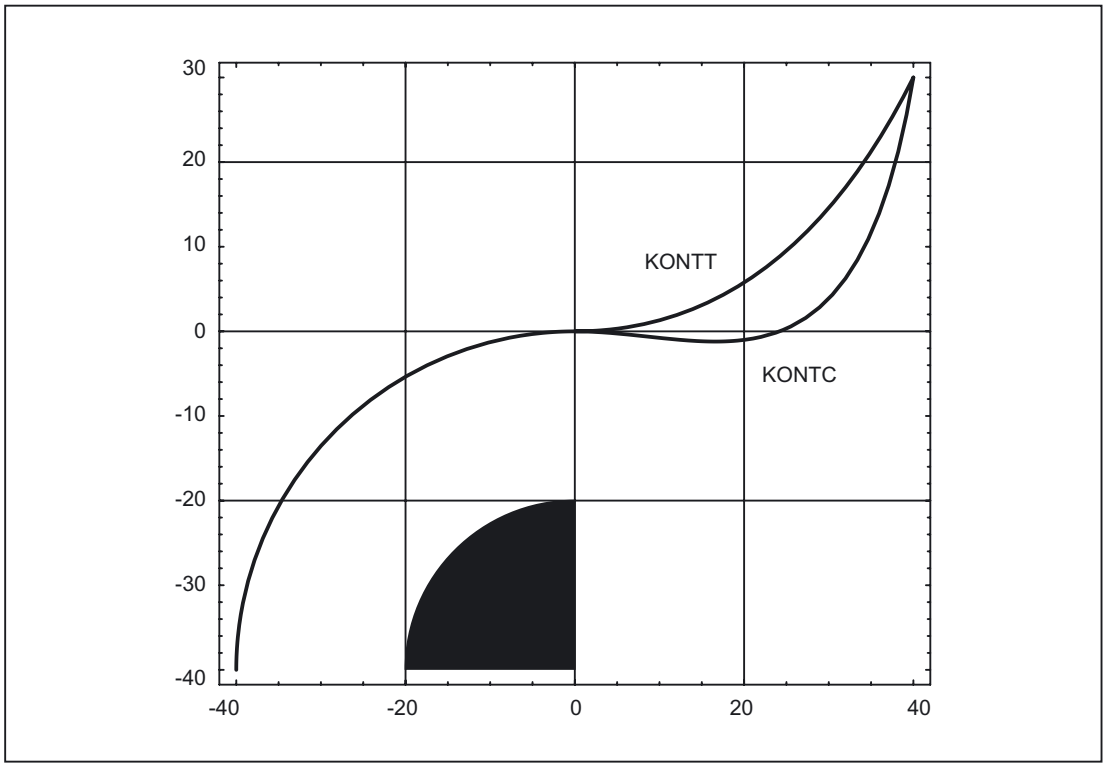


Fig. 2-18 Differences between KONTT and KONTC

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.

2.4.4 Smooth approach and retraction

Meaning

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.

Approach movement

A maximum of 4 sub-movements:

- 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.

Retraction movement

On retraction, the points are traversed in the reverse direction, i.e., starting at P_4 and ending at P_0.

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

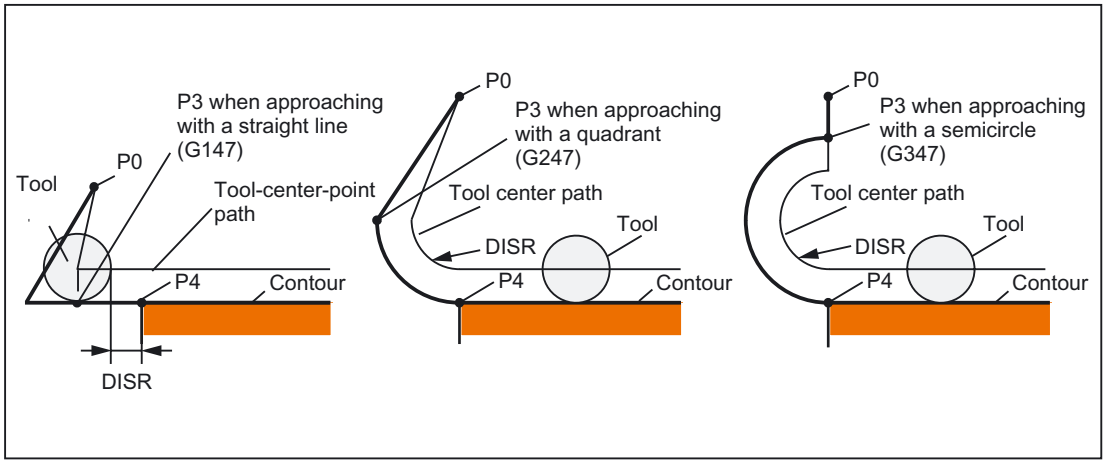


Fig. 2-19 Approach behavior depending on G147 to G347 and DISR (with simultaneous activation of tool radius compensation)

- **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**

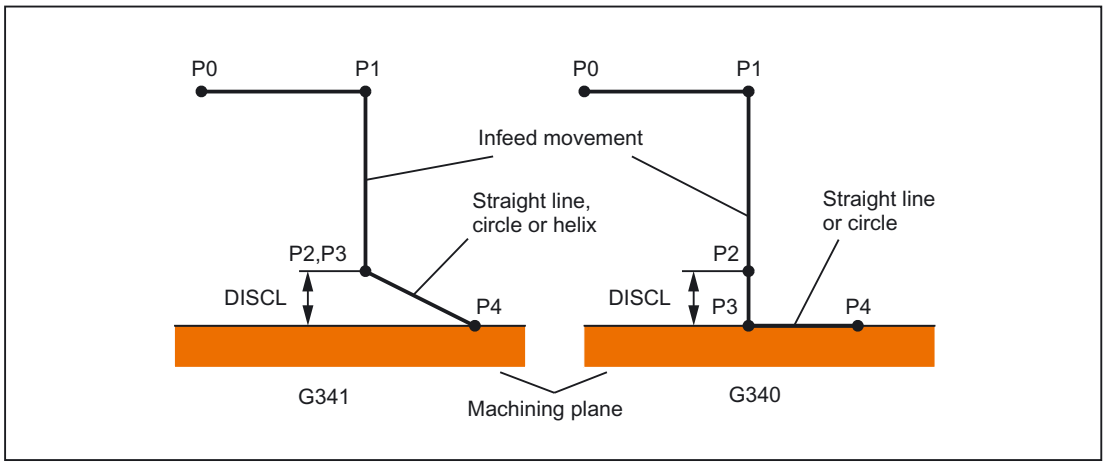


Fig. 2-20 Sequence of the approach movement depending on G340/G341

`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: Specifies the length of a straight approach line or the radius of an approach arc.**

  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 output on approach and retraction with straight lines:

  – 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)

  With circles, `DISR` always specifies the radius of the tool center 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

  Or

  – If the radius value ≤ 0.

- **DISCL: Specifies the distance from point $P_2$to 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 reversal of direction, a tolerance defined by the machine data:
  MD20204 $MC_SAR_CLEARANCE_TOLERANCE
  is permitted.

  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....**

1. Possible programming of end point P4 for approach:

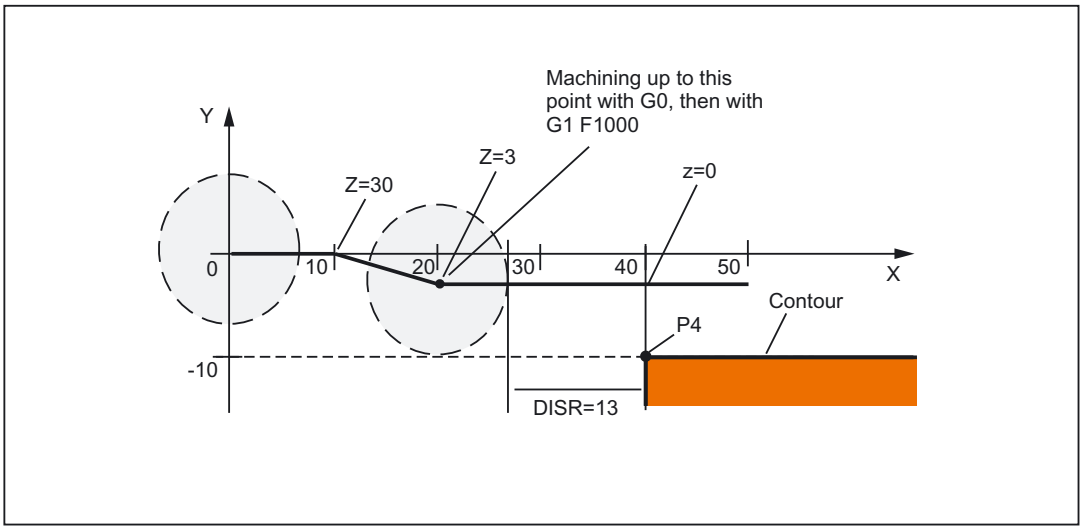
End point P_4 can be programmed in the actual SAR block.

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

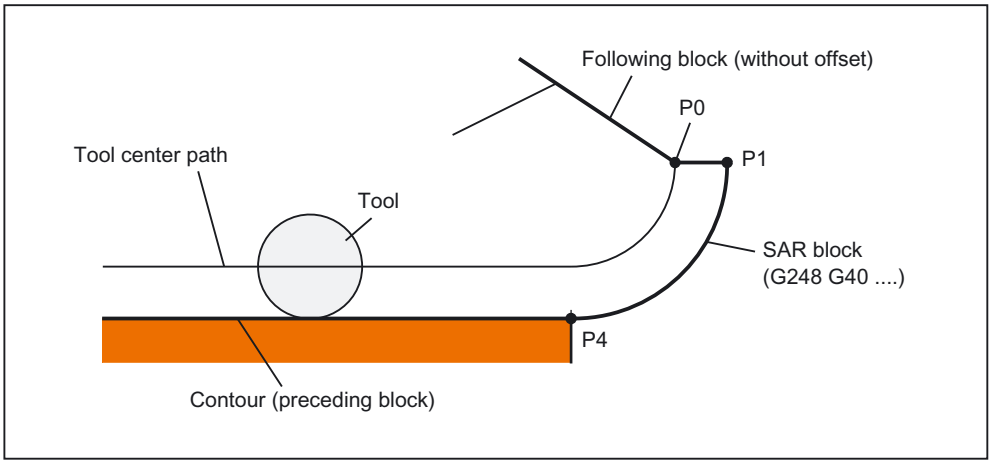
N10 G90 G0 X0 Y0 Z30 D1 T1
N20 X10
N30 G41 G147 DISCL=3 DISR=13 Z=0 F1000
N40 G1 X40 Y-10
N50 G1 X50
...
...
```

`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
```

2. Possible programming of end point P0 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.

  – `FAD` programmed with `G341`:

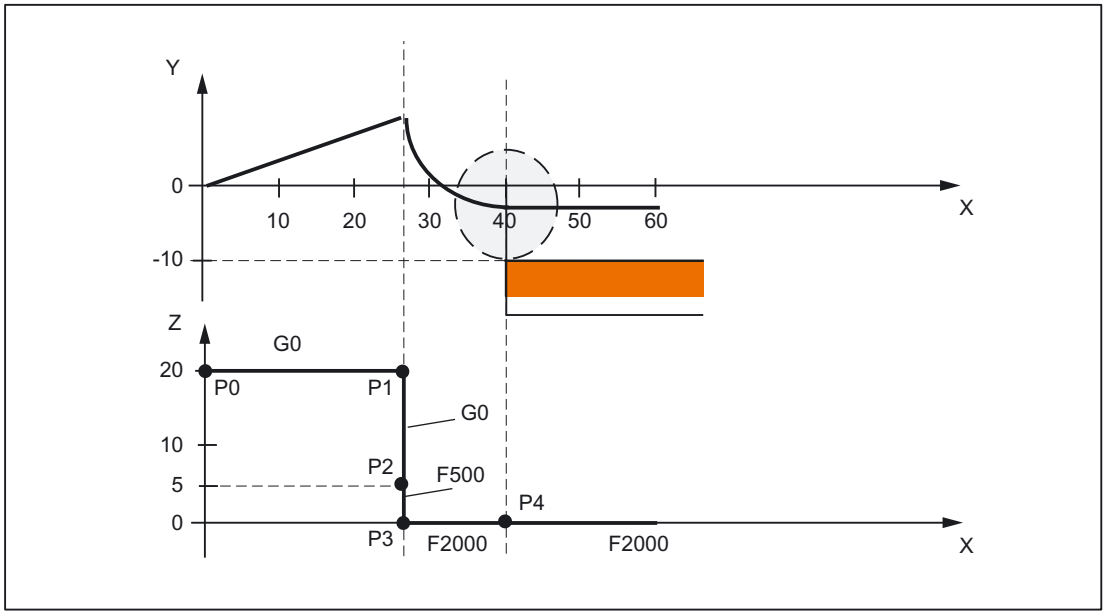
  Feedrate of the infeed movement perpendicular to the machining plane from P_2 to P_3.

  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 7 mm 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.

Velocities

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.

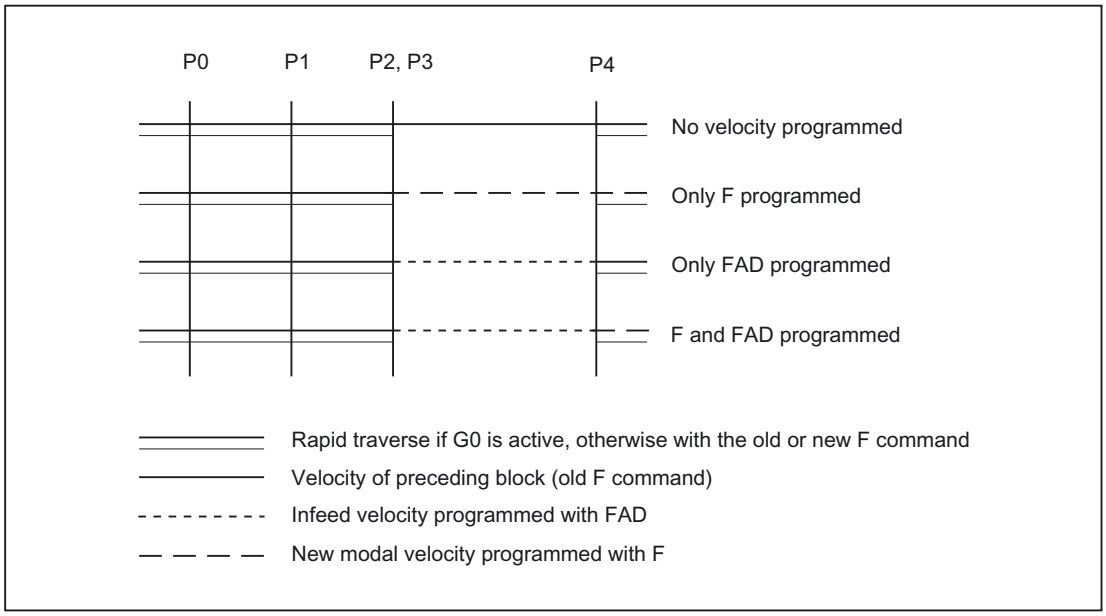


Fig. 2-21 Velocities in the SAR subblocks on approach with G340

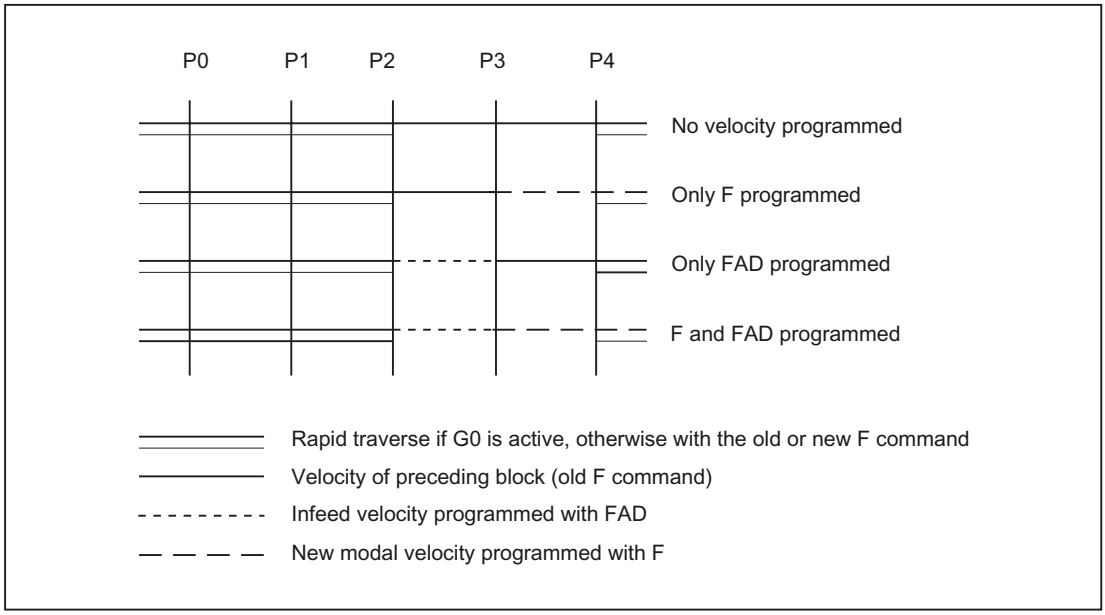


Fig. 2-22 Velocities in the SAR subblocks on approach with G341

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.

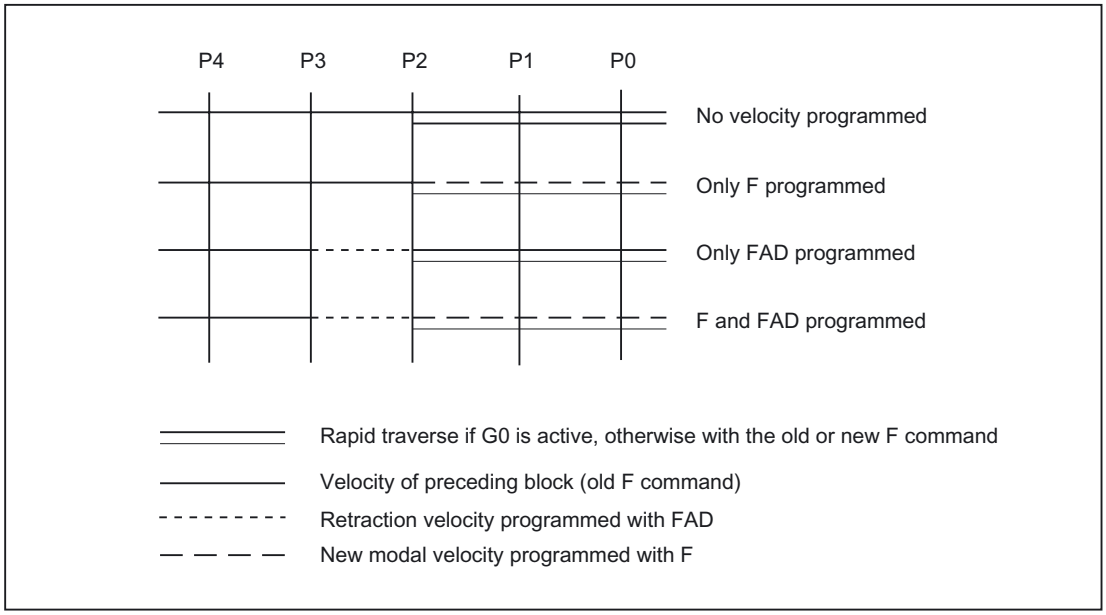


Fig. 2-23 Velocities in the SAR subblocks on retraction

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.

Supplementary conditions

- Any further NC commands (e.g., auxiliary function outputs, synchronous axis movements, positioning axis movements, etc.) can be programmed in a 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 taken not 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.

  Machine data:
  MD20202 $MC_SAR_MAXNUM_DUMMY_BLOCKS
  is used to limit the number of possible dummy blocks.
- 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 movement 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 controller, a preprocessor stop results, in this case, in an alarm.

Response with REPOS

If an SAR cycle is interrupted and repositioned, it resumes at the point of interruption on RMI. With RME, the contact point is the end point of the last SAR block; with RMB, it is the start point of the first SAR block.

If RMI 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.

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)

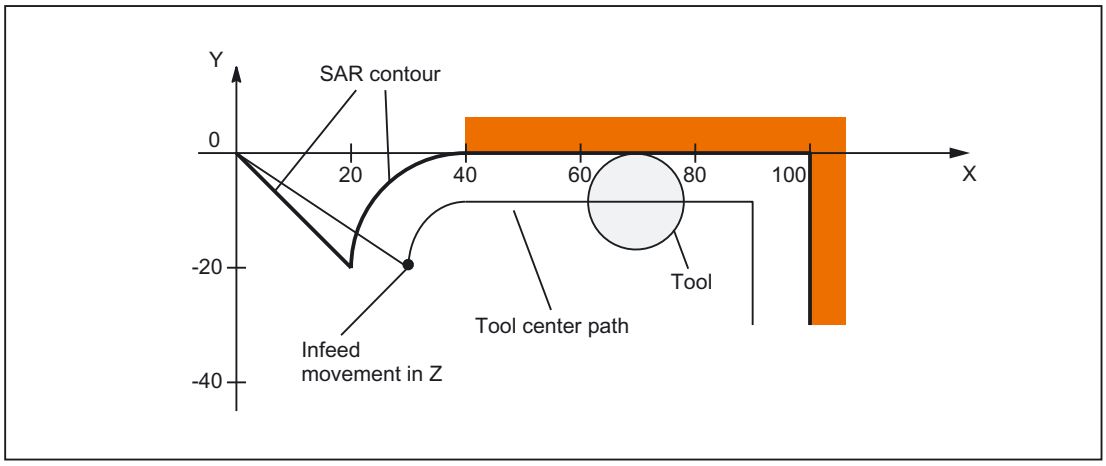


Fig. 2-24 Contour example 1

Part program:

```
$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 compensation `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 movement
  - 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`. (For this velocity to be active in the following blocks, the active G code `G0` must be overwritten in `N30` 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.

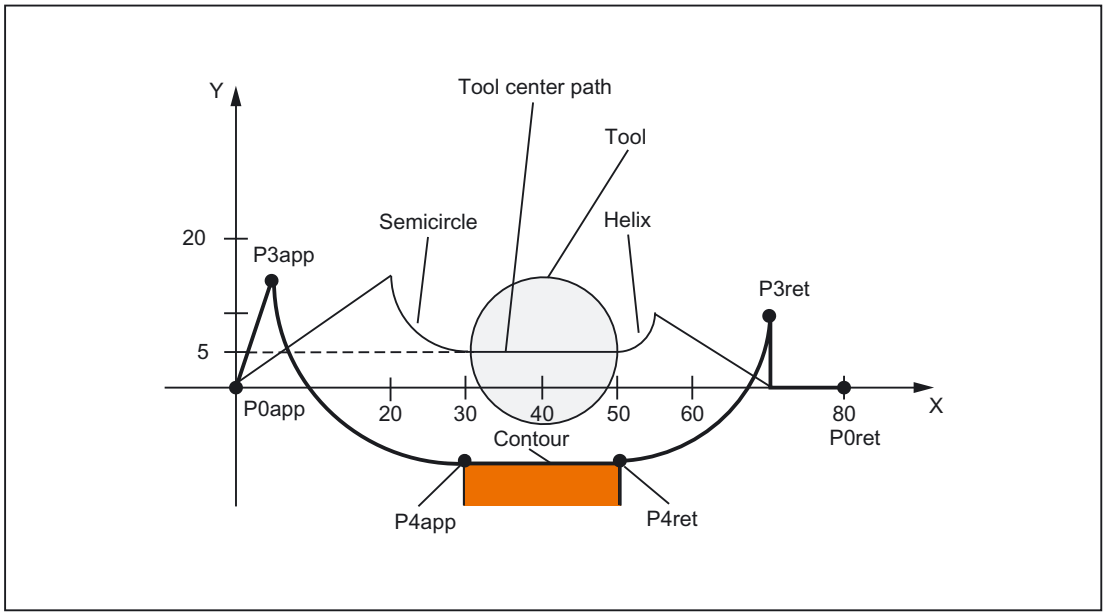


Fig. 2-25 Contour example 2

Part program:

```
$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.

Tool radius compensation allows for an effective radius of 15, which is the sum of the tool radius (10) and the contour compensation (5). The resulting radius of the tool center path in the approach block is therefore 10, and 5 in the retraction block.

2.4.5 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.

2.4.6 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:

`G450`	...	Discontinuous block transitions with transition circle
`G451`	...	Discontinuous block transitions with intersection of equidistant paths

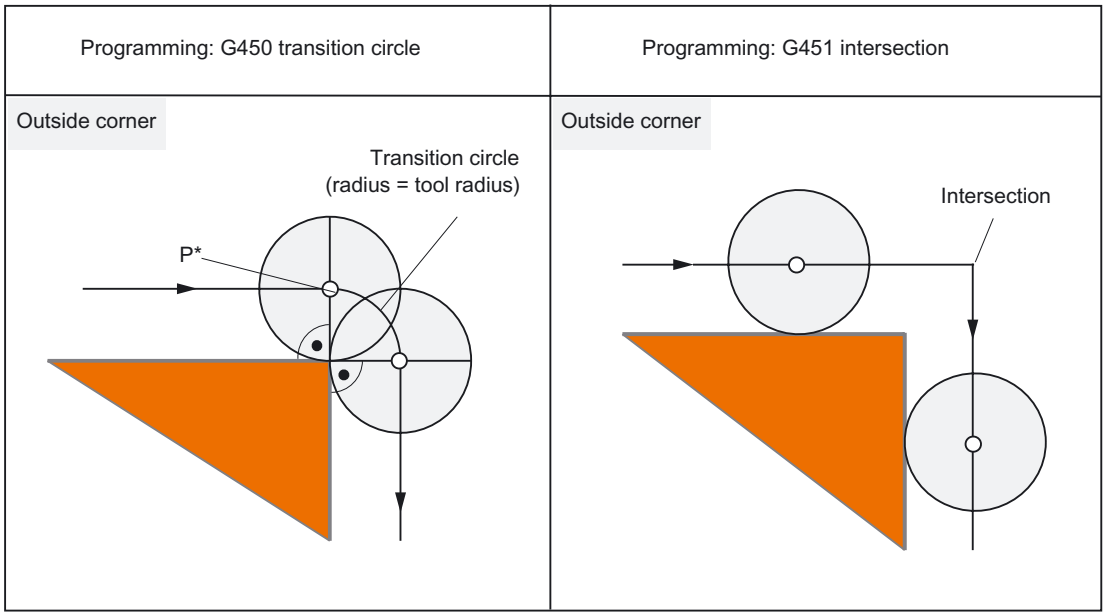


Fig. 2-26 Example of a 90 degree outside corner with G450 and G451

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 begins perpendicular to the path tangent at the end point of the previous path section (program block) and ends in the perpendicular position at the start point of the new path section (program block).
In cases where the 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 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.

DISC = 0	...	Overshoot disabled, transition circle active
DISC = 100	...	Overshoot large enough to theoretically produce a response similar to intersection (G451).

Machine data:
MD20220 $MC_CUTCOM_MAX_DISC (maximum value for DISC)
can be used to define the maximum value that can be programmed with `DISC`.

Values greater than 50 are generally not advisable with `DISC`.

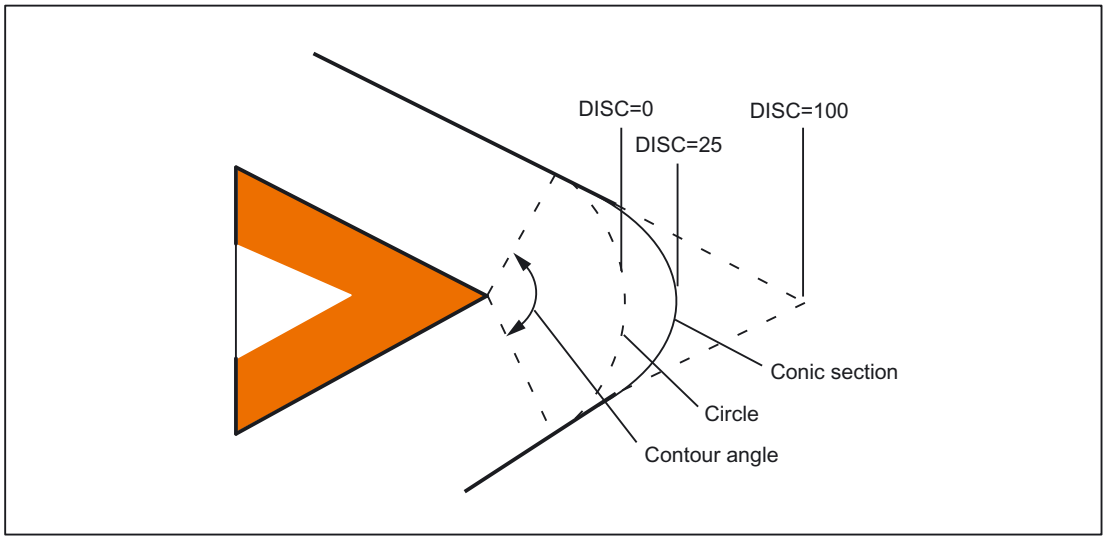


Fig. 2-27 Example: Overshoot with DISC=25

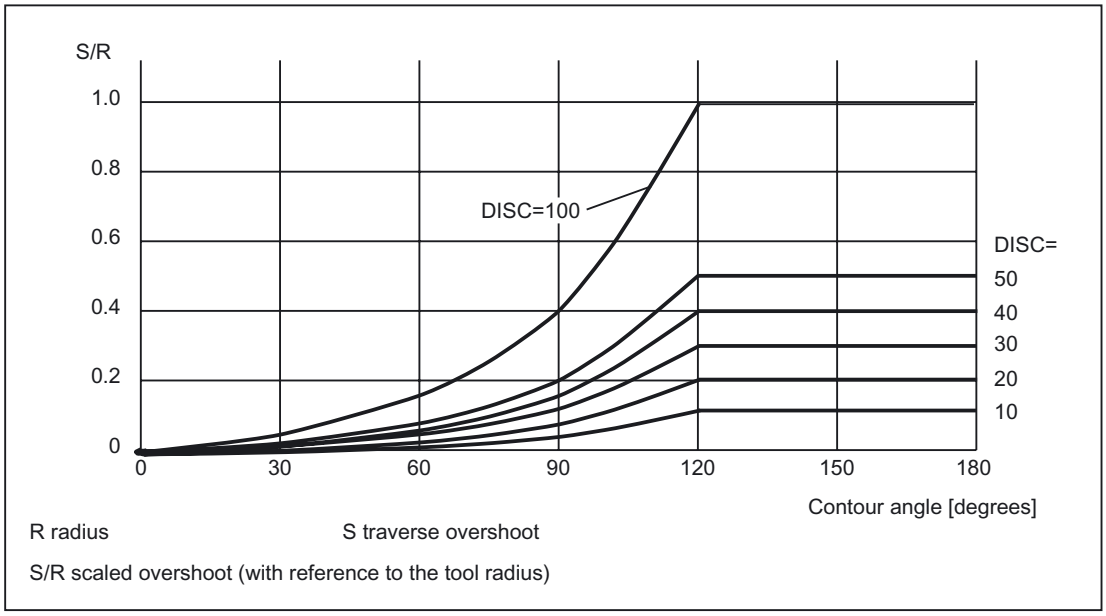


Fig. 2-28 Overshoot with DISC depending on contour angle

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 contour angle, which can be traversed following this automatic switchover (intersection → transition circle), can be defined in machine data:
MD20210 $MC_CUTCOM_CORNER_LIMIT (maximum angle for compensation blocks with TRC).

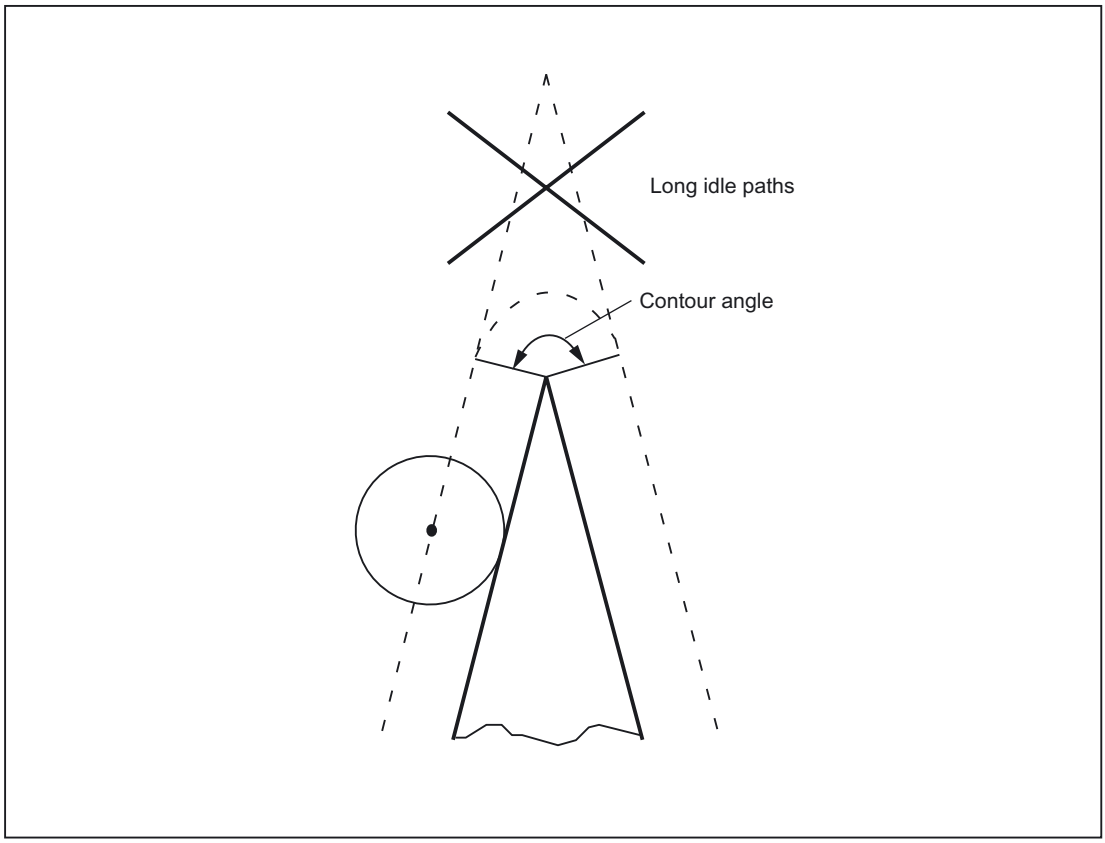


Fig. 2-29 Example of automatic switchover to transition circle

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 contour angle, which can be traversed following this automatic switchover (transition circle → intersection), can be defined in machine data:
MD20230 $MC_CUTCOM_CURVE_INSERT_LIMIT
(maximum angle for intersection calculation with TRC).

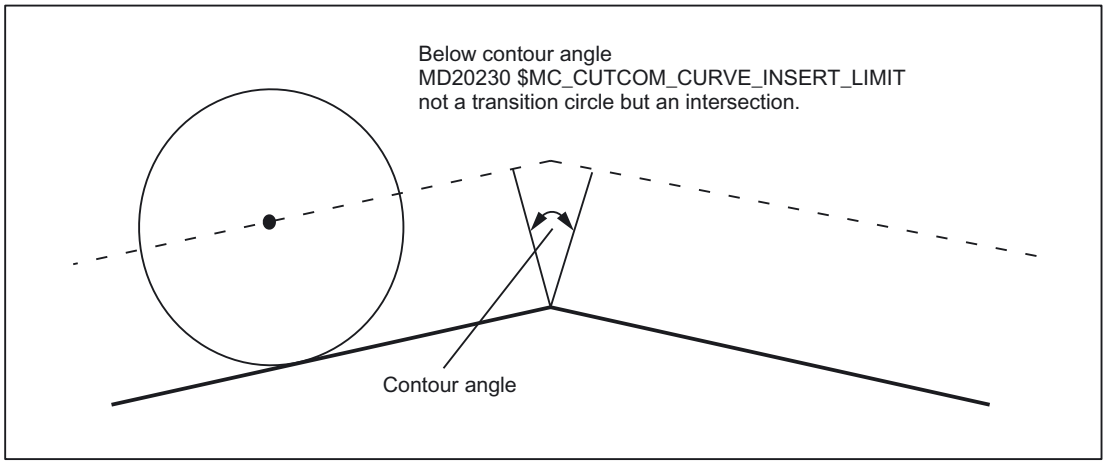


Fig. 2-30 Example of automatic switchover to intersection

2.4.7 Compensation at 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).

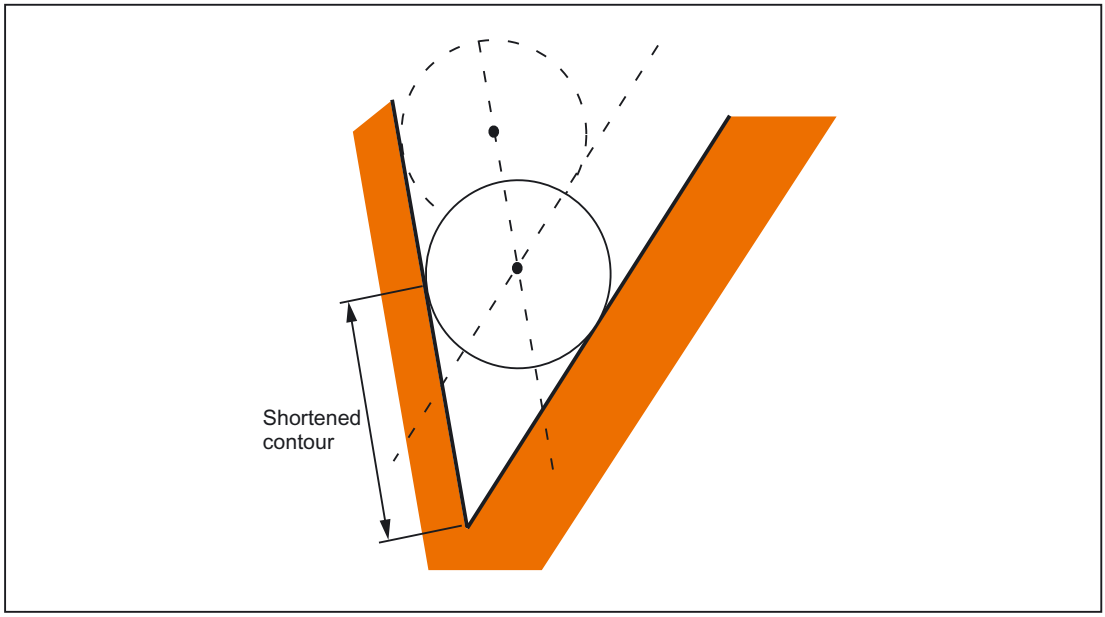


Fig. 2-31 Example of a shortened contour

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.

This maximum number of blocks used for the predictive check can be entered in machine data:
MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS
(blocks for predictive contour calculation with TRC).

If no intersection is found within the number of blocks defined for the check, program execution is interrupted and an alarm is output.

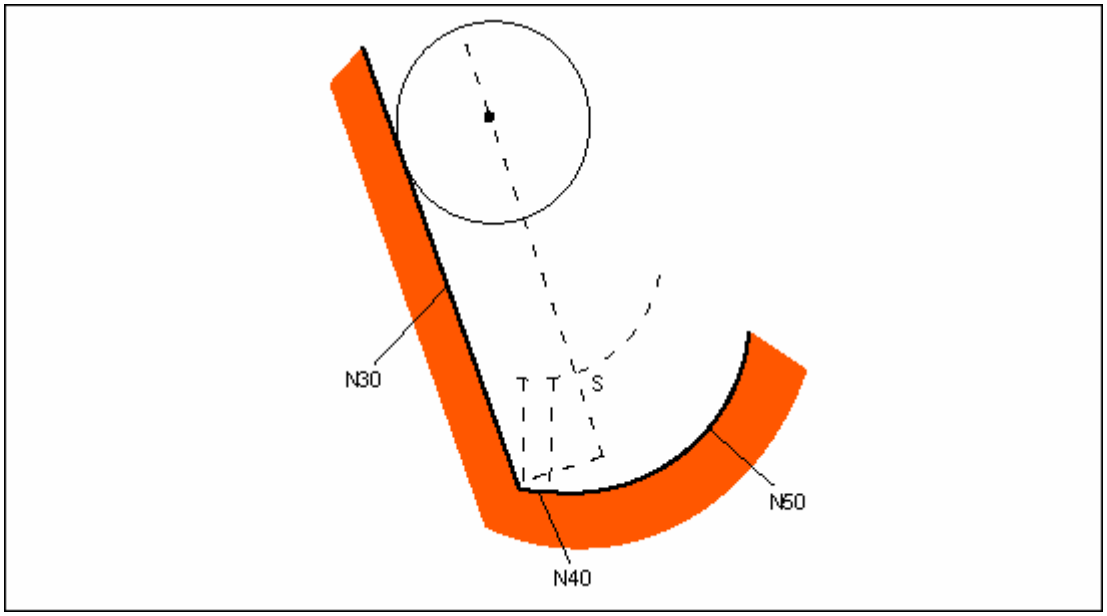


Fig. 2-32 If there is no intersection between N30 and block N40, the intersection between block N30 and block N50 is calculated.

Multiple intersections

→ see also "Collision detection"

It can be the case with inside corners that multiple intersections of the equidistant paths are found in several consecutive blocks. In these cases, the last intersection is always used as the valid intersection (see figure below).

The maximum number of blocks used for the predictive check can be entered in machine data:
MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS
(blocks for predictive contour calculation with TRC).

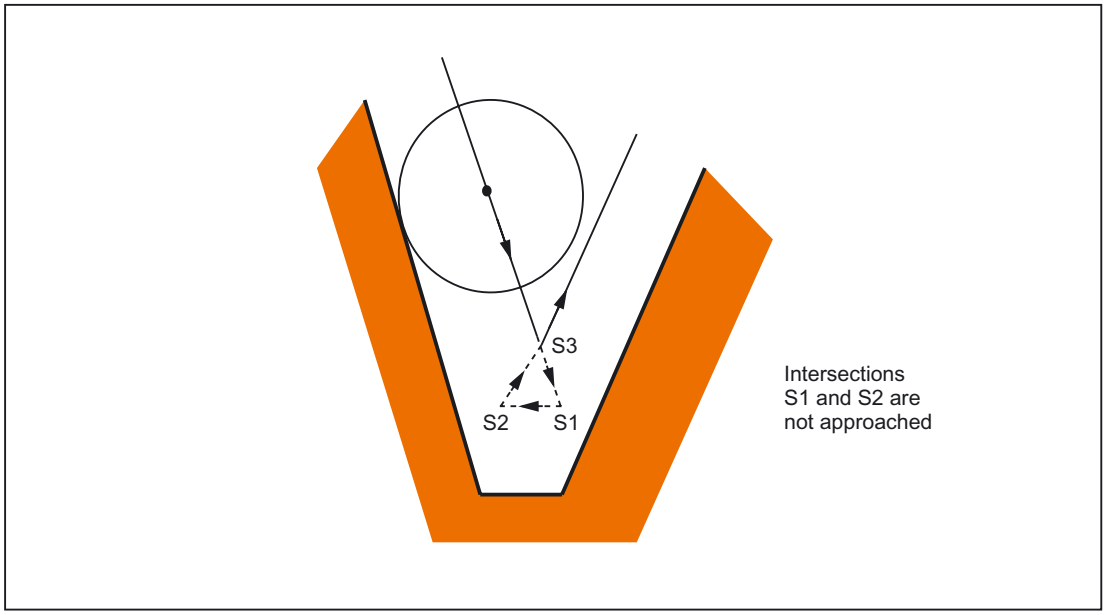


Fig. 2-33 Example: Inside corner with TRC without contour violation (predicting 3 blocks)

Special points to be noted

Where multiple intersections with the next block are found, the intersection nearest the start of the next block applies.

2.4.8 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 machine data:
MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS
(blocks for predictive contour calculation with TRC).

Programming

Collision detection can be activated or deactivated in the program:

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 meaning 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.

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.

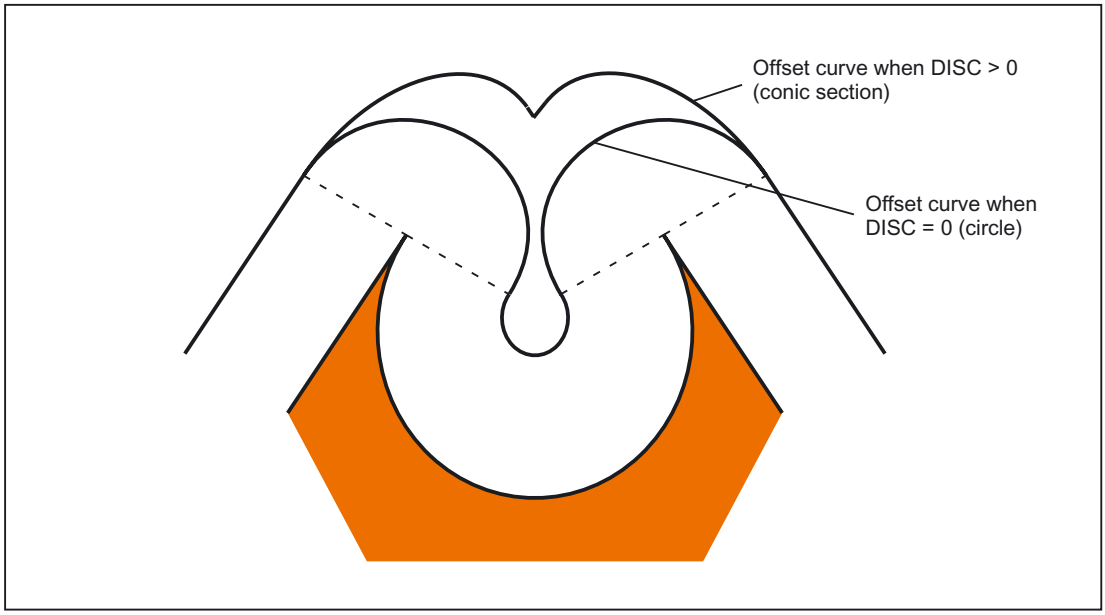


Fig. 2-34 Bottleneck detection and outside corners

2.4.9 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).

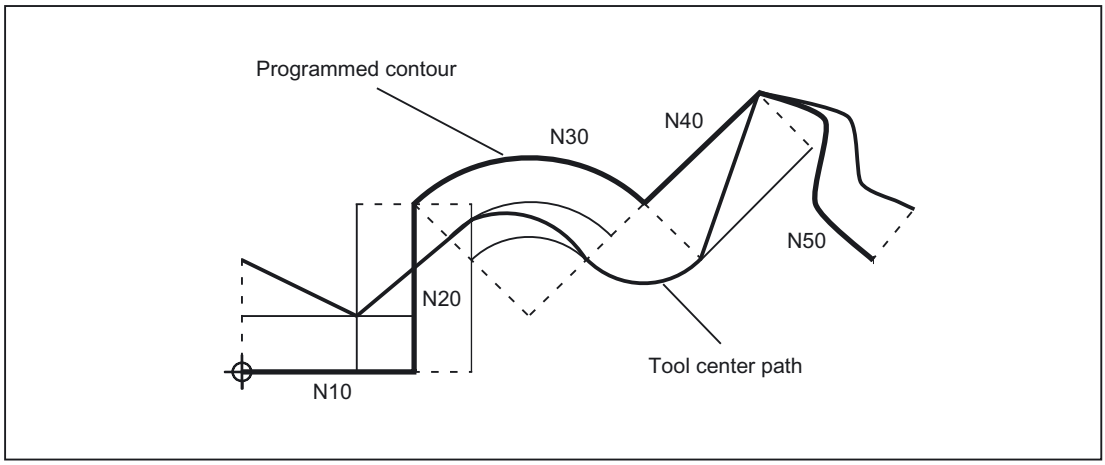


Fig. 2-35 Tool radius compensation with variable compensation value

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.

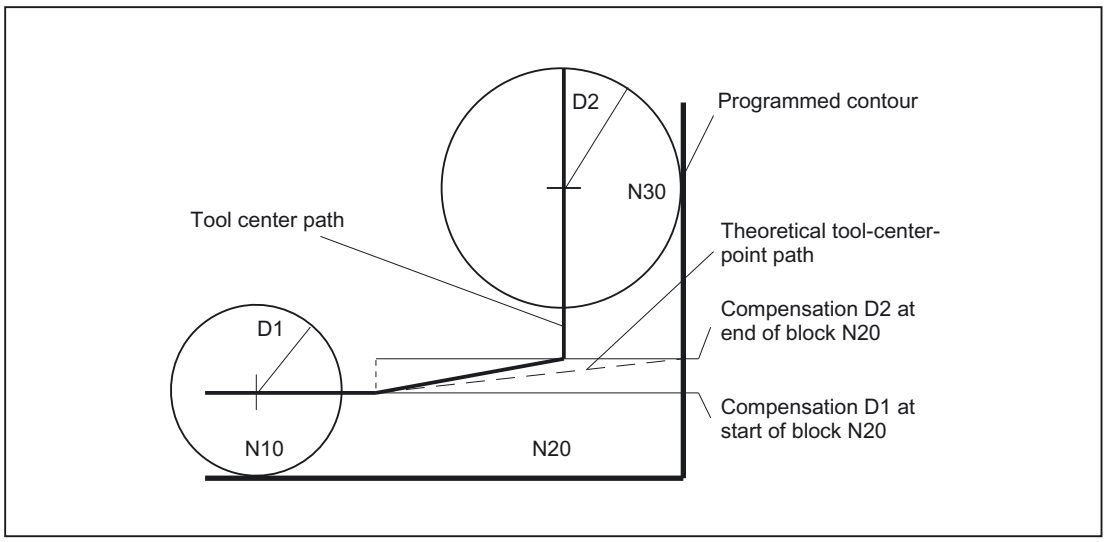


Fig. 2-36 Intersection calculation with variable compensation value

Restrictions

If the compensation radius is programmed as being larger than the programmed circle radius for machining on the inside of the circle, the machining operation is rejected with 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.

Assigning a value other than zero to setting data:
SD42496 $SC_CUTCOM_CLSD_CONT
provides a means of achieving a stable closed contour by choosing the first intersection point instead of the second. In that case, the second intersection point is always reached, even if the block is skipped. A third block is then not required.

2.4.10 Keep tool radius compensation constant

Meaning

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
```

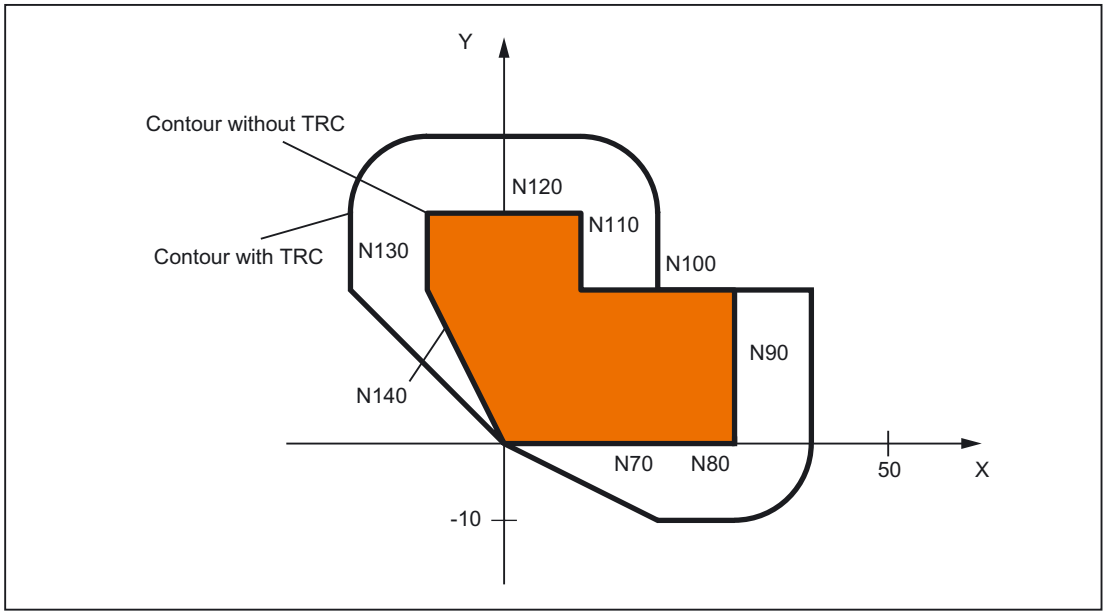


Fig. 2-37 Sample program for contour suppression

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

  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.

2.4.11 Alarm response

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.

Setting data:
SD42480 $MC_STOP_CUTCOM_STOPRE
is, therefore, inserted so that the tool radius compensation response 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.

2.4.12 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.

Machine data:
MD20256 $MC_CUTCOM_INTERS_POLY_ENABLE
is used to enable the intersection point procedure 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.

2.4.13 G461/G462: Approach/retract strategy extension

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
```

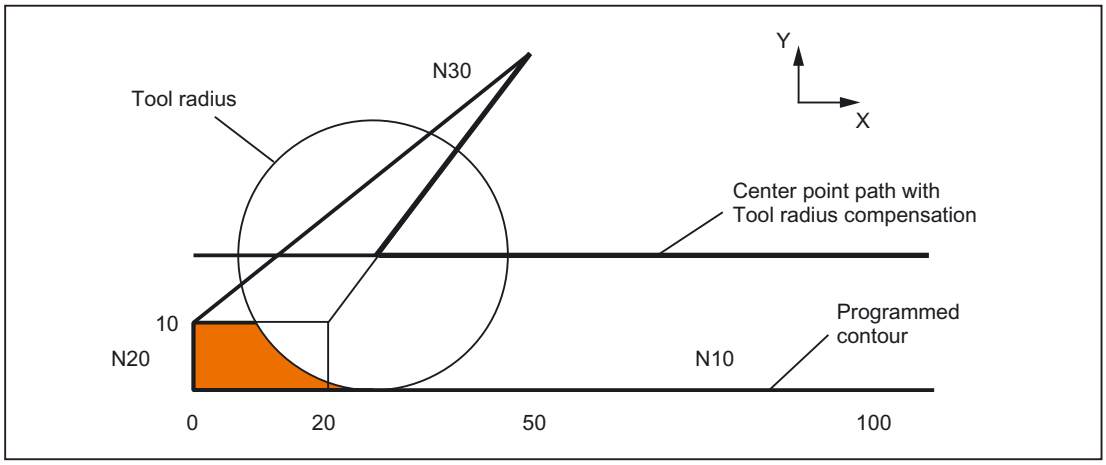


Fig. 2-38 Retraction behavior with G460

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.

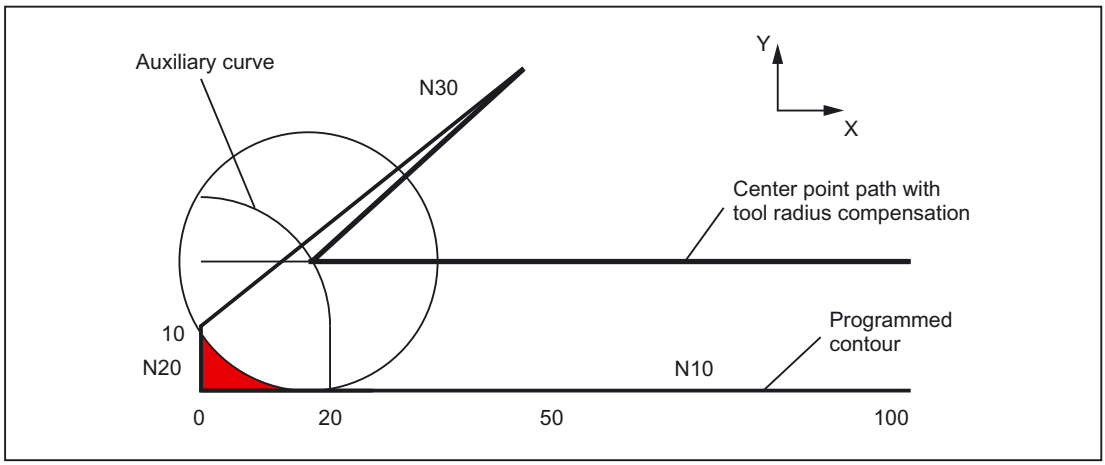


Fig. 2-39 Retraction behavior with G461

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, alarm 10751 (collision danger) is output.

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).

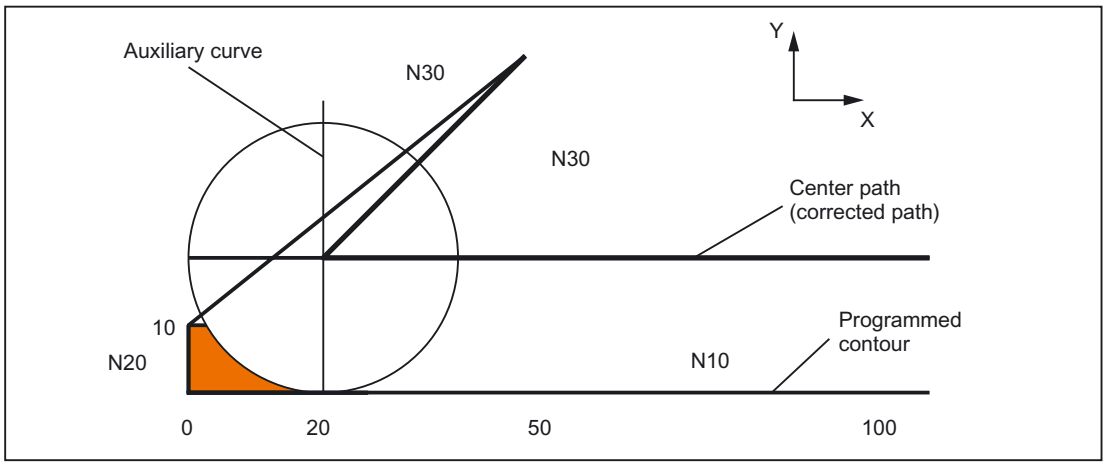


Fig. 2-40 Retraction behavior with G462

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
```

2.5 Toolholder with orientation capability

2.5.1 General

Introduction

The orientation of the tool can vary (e.g., **due to retooling**) for a 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.

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

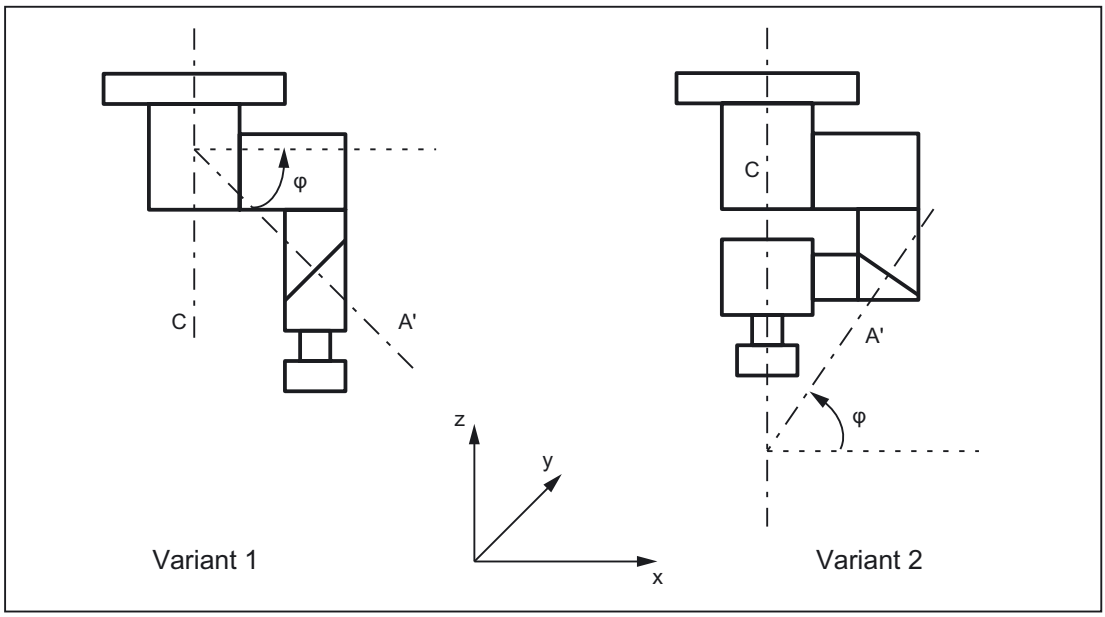


Fig. 2-41 Cardan toolholder with two axes

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	Then
No toolholder is active	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
Toolholder without orientation change is active	

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 kinematic, are not perpendicular to each other and if the tool axis, which defines the tool direction, does not stand 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 meaning 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 14153 if it is tried to activate the toolholder with orientation capability.

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 meaning

System variables for toolholders with orientation capability

$TC_CARRn[m]

n: Parameters 1 to 33

m: Number of toolholder with orientation capability 1...value of MD18088 $MN_MM_NUM_TOOL_CARRIER

Description	NCK variables	Language format	Preassignment
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 variables	Language format	Preassignment
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.		

2.5.2 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:

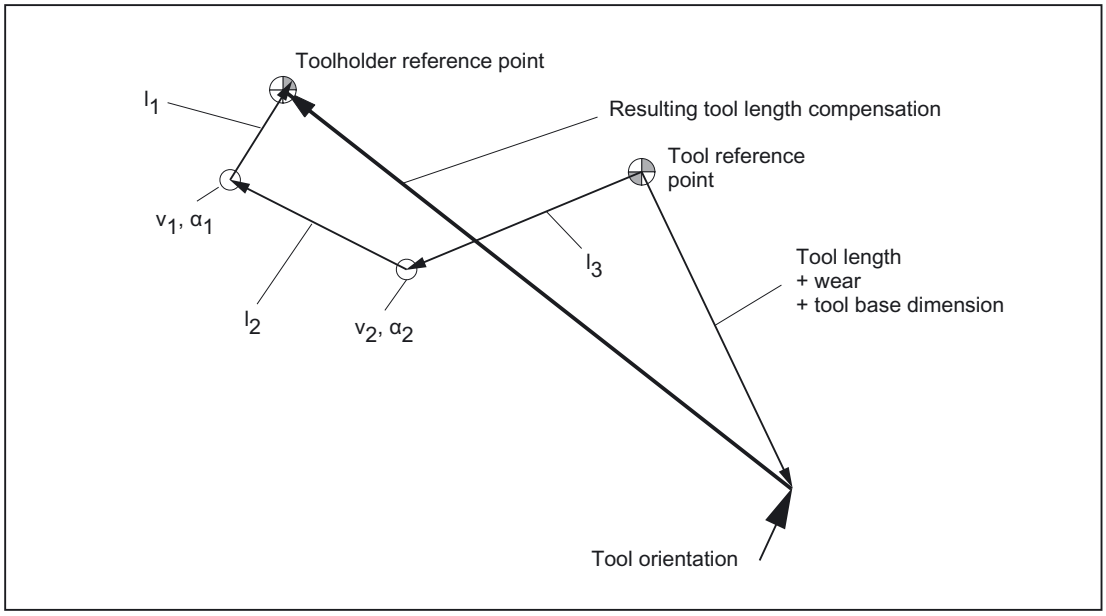


Fig. 2-42 Kinematic chain to describe a tool with orientation

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.

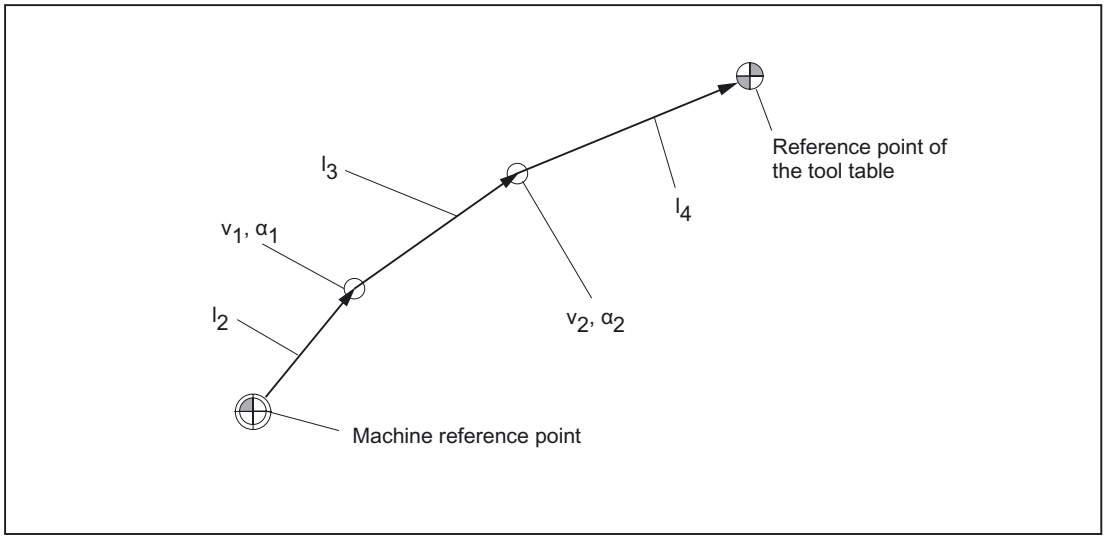


Fig. 2-43 Kinematic chain to describe a rotary table

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.

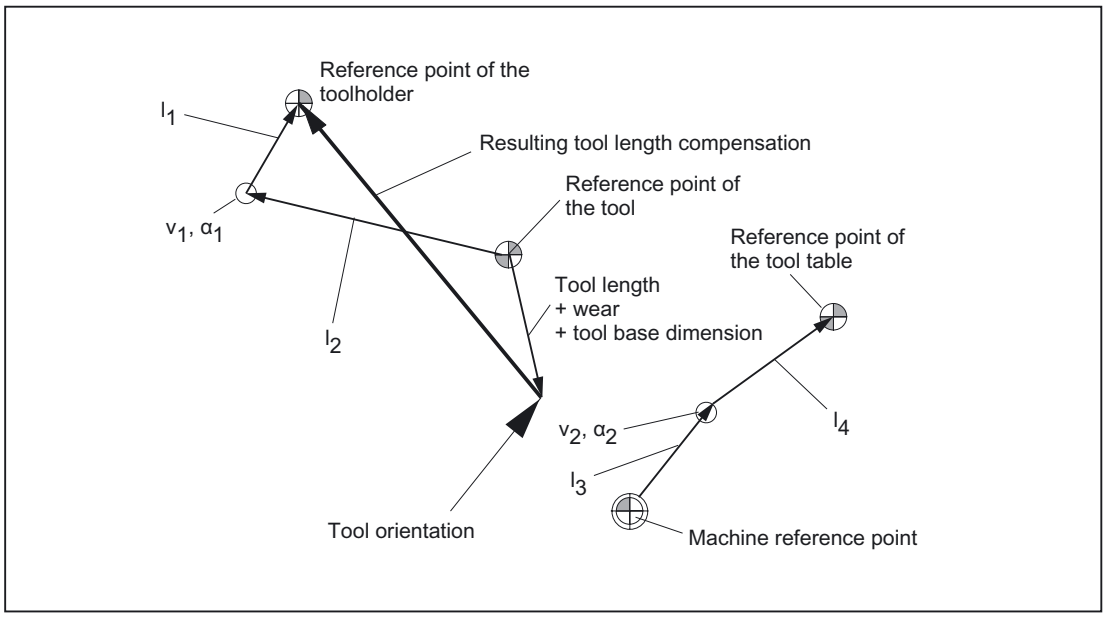


Fig. 2-44 Kinematic sequence with extended kinematics

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 more information about the meaning of system variables $TC_CARR41 to $TC_CARR65, which are available for the fine offset, please refer to:
References:
/PGA/ Programming Guide, Production Planning; Tool Compensations.

Activation

The fine offset values are added to the basic values if
SD42974 $SC_TOCARR_FINE_CORRECTION = 1.

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

- Setting data:
  SD42974 $SC_TOCARR_FINE_CORRECTION
  is enabled simultaneously.

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 α1and 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 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 more information about the reference points in the working area, please refer to:
References:
/FB1/ Description of Functions, Basic Machine; Axes, Coordinate Systems, Frames (K2).

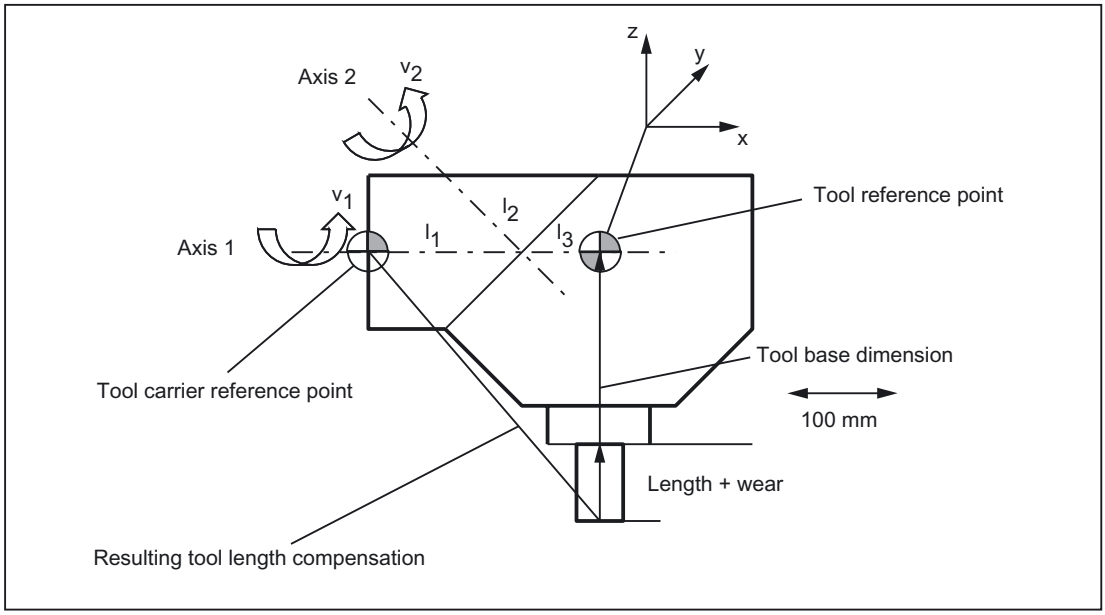


Fig. 2-45 Assignment of the toolholder data

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 variables	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

Description	NCK variables	Value
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

Explanation

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.

2.5.3 Inclined 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 information about 5-axis transformation, please refer to:
References:
/FB3/ Description of Functions, 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.

2.5.4 Machine with rotary 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

If machine data:
MD20180 $MC_TOCARR_ROT_ANGLE_INCR[i] is zero,
the rotary axis position is used with its programmed or calculated value.

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]
when n is an integer.

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 movement by the machine itself. The ratios are shown in the figure below:

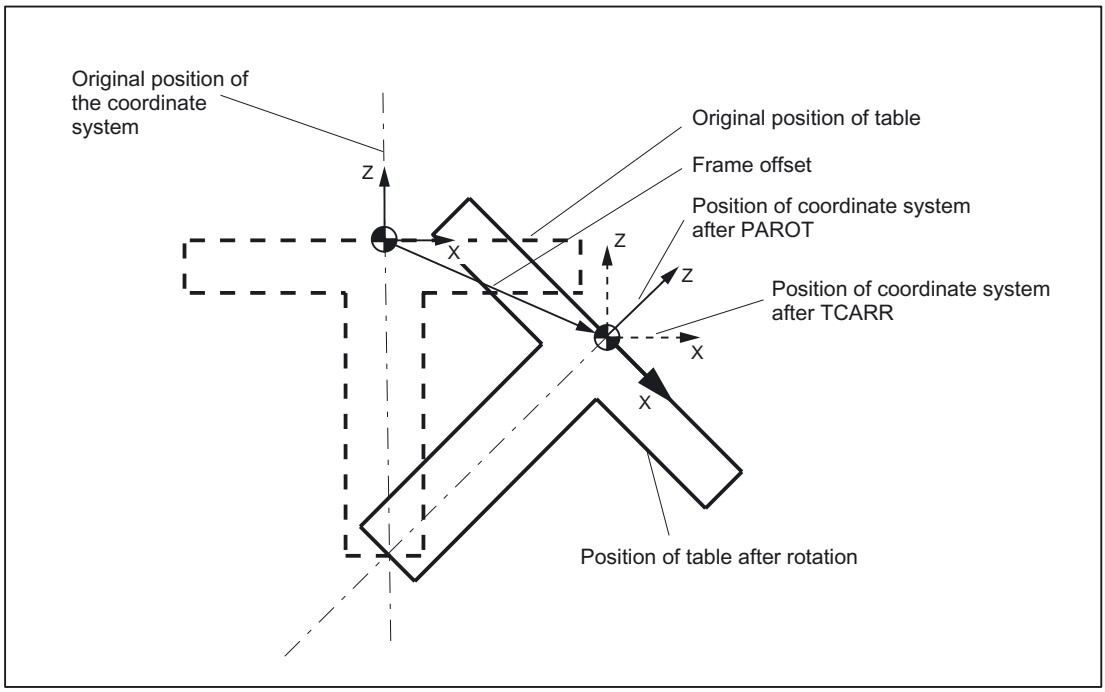


Fig. 2-46 Zero offset on activation of a rotary table with TCARR

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, bit 2 must be enabled in machine data:
MD28082 $MC_MM_SYSTEM_FRAME_MASK.

The basic frame identified by machine data:
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER
is then no longer required for the zero 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 possible, therefore, when using TOROT, for example, to rotate a workpiece such that an oblique plane lies parallel to the X/Y 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 more information about the TCARR and TOROT functions, as well as PAROT in relation to channel-specific system frames, please refer to:
References:
/FB1/ Description of Functions, Basic Machine; Axes, Coordinate systems, Frames (K2).

2.5.5 Procedure when using orientational toolholders

Creating a toolholder

The number of toolholder data sets available in the NCK is defined by the machine data: MD18088 $MN_MM_NUM_TOOL_CARRIER.

1. The value is calculated as follows:

   MD18088 $MN_MM_NUM_TOOL_CARRIER =
   "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 more information about defining and assigning a TO unit via machine data:
MD28085 $MC_MM_LINK_TOA_UNIT,
see:
References:
/FB/ Description of Functions, Extended Functions; Memory Configurations (S7).

1. 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`.

2. 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 NCKMMC (OPI) variable services using system variable $P_TCANG[<n>].

3. 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 NC program command `TCARR = m` (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 non-rotating 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, yielding 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 $\alpha_1$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.

2.5.6 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 new "value" is **written** to parameter **n** for toolholder **m** with:

```
$TC_CARRn[m] = value
```

The parameters of toolholder m can be **read** with:
`value = $TC_CARRn[m]` (`value` must be a REAL variable),
as long as the toolholder data set has already been defined.

The toolholder data set number must lie within the range defined by machine data:
MD18088 $MN_MM_NUM_TOOL_CARRIER
(total number of definable toolholder data blocks).

This number of toolholder data sets, divided by the number of active channels, can be defined for a channel.

Exception:
If non-standard settings are selected via machine data:
MD28085 $MC_MM_LINK_TOA_UNIT.

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 α1and 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 information about programming tool compensations with toolholder kinematics and system variables, please refer to:
References:
/PGA/ Programming Guide, Production Planning.

2.5.7 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 be defined (i.e., 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 α1or 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

Machine data:
MD20126 $MC_TOOL_CARRIER_RESET_VALUE
can be used to select a toolholder automatically on RESET or program start, and is treated in the same way as the controlled selection of a tool via machine data:
MD20120 $MC_TOOL_RESET_VALUE.

Behavior on RESET or during program start is controlled via bit 6 in machine data:
MD20110 $MC_RESET_MODE_MASK
or
MD20112 $MC_START_MODE_MASK;
this is the same bit that is used in tool selection.

References:
/FB1/ Description of Functions, Basic Machine; Mode Group, Channel, Program Operation, Reset Response (K1)

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

Machine data:
MD22530 $MC_TOCARR_CHANGE_M_CODE
can be set so that, optionally, a constant or an M code is output when the toolholder is selected. The number of the code is derived from the toolholder number.

References:
/FB1/ Description of Functions, Basic Machine; Output of Auxiliary Functions to PLC (H2)

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.

References:
/FB1/ Description of Functions, Basic Machine; Mode Group, Channel, Program Operation, Reset Response (K1)

2.6 Incrementally programmed compensation values

2.6.1 G91 extension

Prerequisites

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

With setting data:
SD42442 $SC_TOOL_OFFSET_INCR_PROG,
it is possible to define whether a changed tool length is traversed with `FRAME` and incremental programming of an axis, or whether only the programmed path is traversed.

Zero offset/frames G91

With setting data:
SD42440 $SC_FRAME_OFFSET_INCR_PROG,
it is possible to define 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.

References:
/FB1/ Description of Functions, Basic Machine; Axes, Coordinate Systems, Frames (K2), Chapter: "Frames"

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), 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.

2.6.2 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:

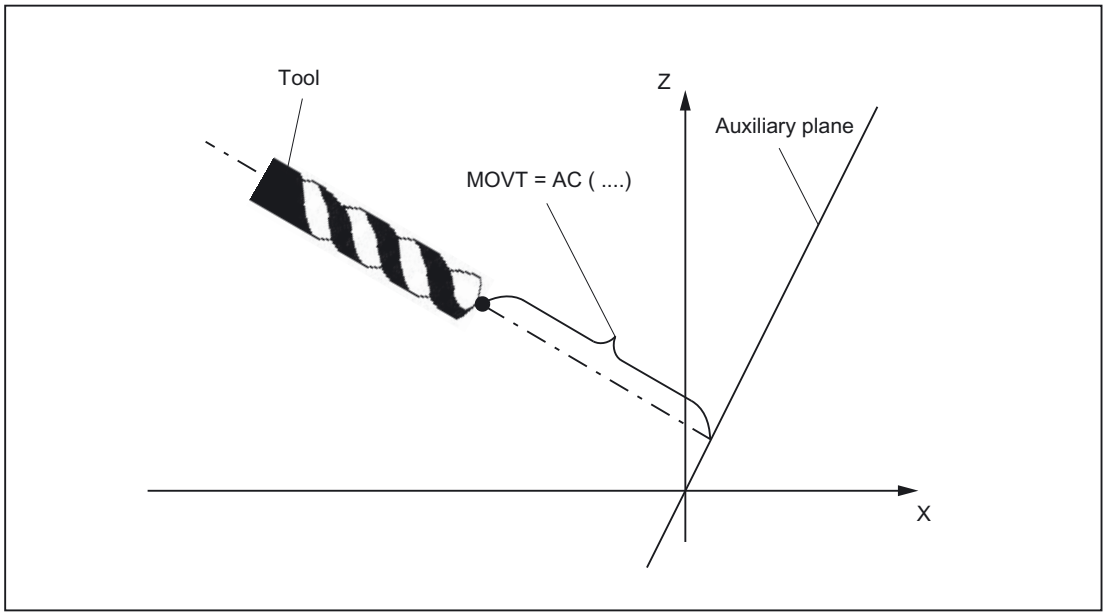


Fig. 2-47 Definition of the position for absolute programming of a motion in tool direction

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).

2.7 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:
MD18114 $MN_MM_ENABLE_TOOL_ORIENT
is set to a value other than zero.

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.

If setting data:
SD42950 $SC_TOOL_LENGTH_TYPE
is not equal to zero, 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.

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 setting data:
SD42940 $SC_TOOL_LENGTH_CONST
is not equal to zero, the tool orientation is not rotated on a change of plane.

Tool length components

In relation to setting data:
SD42910 $SC_MIRROR_TOOL_LENGTH
and
SD42950 $SC_TOOL_LENGTH_TYPE,
tool orientation components are treated in the same way as tool length components, i.e., the components are interchanged accordingly and assigned to the geometry axes.

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 equivalent 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

2.8 Special handling of tool compensations

2.8.1 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

For detailed information about sum and tool compensations, please refer to:
References:
/FBW/ Description of Functions, Tool Management
/PG/ Programming Guide, Fundamentals; Tool Compensations

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)

2.8.2 Mirroring of tool lengths (SD 42900: MIRROR_TOOL_LENGTH)

Activation

Tool length mirroring is activated by:

SD42900 $SC_MIRROR_TOOL_LENGTH <> 0 (TRUE)

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

Inverting the sign mirrors the wear values of the tool length components whose associated axes are mirrored.

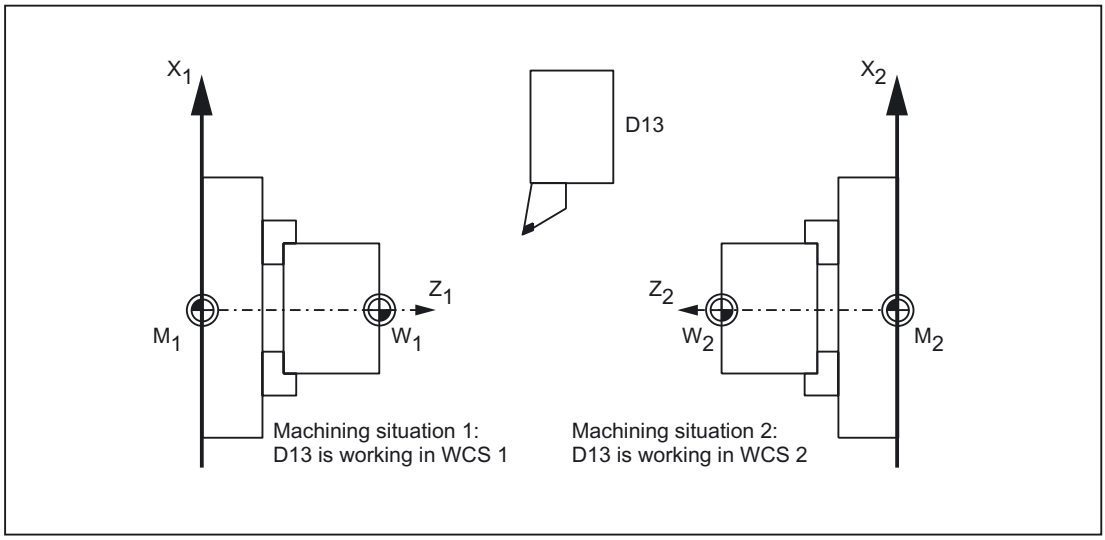


Fig. 2-48 Application example: Double-spindle turning machine

2.8.3 Mirroring of wear lengths (SD 42920: WEAR_SIGN_CUTPOS)

Activation

Wear length mirroring is activated by:

SD42920 $SC_WEAR_SIGN_CUTPOS <> 0 (TRUE)

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

Activating the
Mirror tool lengths (SD42900 <> 0)
and
Mirror wear lengths (SD42920 <> 0)
functions can disable mirroring (sign inversion) in one or a number of components.

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.

2.8.4 Tool length and plane change (SD 42940: 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 setting data:
SD42940 $SC_TOOL_LENGTH_CONST
in accordance with the following table:

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.

2.8.5 Tool type (SD 42950: TOOL_LENGTH_TYPE)

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.

- Value = 1: Always as for milling tools
- 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.

2.8.6 Temperature offsets in the tool direction (SD 42960: TOOL_TEMP_COMP)

Note

Temperature offset in the tool direction is operative only as an option with generic 5-axis transformation, types 24 and 56.

2.8.7 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.

2.8.8 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.

Compensation applies to the "Temperature compensation" option and is active only when the axis to be compensated is really referenced.

Activation

Temperature compensation in the tool direction is an option, which must be enabled beforehand.

It is activated by setting machine data:
MD20390 $MC_TOOL_TEMP_COMP_ON
to a value **other than zero**.

Bit 2 must also be enabled in machine data:
MD32750 $MA_TEMP_COMP_TYPE[<axis index>]
for each affected channel axis.

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

Temperature compensation in the tool direction is operative only 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 movement 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.

Supplementary conditions

Temperature compensation in the tool direction is an option, which must be enabled in advance and is available:

- For generic 5-axis transformation
- For transformation with rotated linear axis for transformation types 64 to 69

Limit values

Compensation values are restricted to maximum values via machine data:
MD20392 $MC_TOOL_TEMP_COMP_LIMIT[0]
to
MD20392 $MC_TOOL_TEMP_COMP_LIMIT[2].

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 the setting data:
SD42960 $SC_TOOL_TEMP_COMP[0]
to
SD42960 $SC_TOOL_TEMP_COMP[2].

The setting data are 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 about 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:
SD42960 $SC_TOOL_TEMP_COMP[2]
is also effective in the direction of the machine X axis if transformation is active.

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 are listed below:

The first rotary axis rotates about Z. C axis
The second rotary axis rotates about Y. B axis

The essential machine data are shown in the table below:

MD20390 TOOL_TEMP_COMP_ON = TRUE	; Temperature compensation active
Option	; Activate option
MD32750 TEMP_COMP_TYPE[AX1] = 4	; Compensation in tool direction
MD32750 TEMP_COMP_TYPE[AX2] = 4	; Compensation in tool direction
MD32750 TEMP_COMP_TYPE[AX3] = 4	; Compensation in tool direction
	; Assignment of transformation type 24:
MD24100 TRAFO_TYPE_1 = 24	; Transformer type 24 in first channel
MD24110 TRAFO_AXES_IN_1[0] = 1	; First axis of the transformation
MD24110 TRAFO_AXES_IN_1[1] = 2	; Second axis of the transformation
MD24110 TRAFO_AXES_IN_1[2] = 3	; Third axis of the transformation
MD24110 TRAFO_AXES_IN_1[3] = 5	; Fifth axis of the transformation
MD24110 TRAFO_AXES_IN_1[4] = 4	; Fourth axis of the transformation
MD24120 TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1	; Geo axis for channel axis 1
MD24120 TRAFO_GEOAX_ASSIGN_TAB_1[1] = 2	; Geo axis for channel axis 2
MD24120 TRAFO_GEOAX_ASSIGN_TAB_1[2] = 3	; Geo axis for channel axis 3
MD24570 TRAFO5_AXIS1_1[0] = 0.0	;
MD24570 TRAFO5_AXIS1_1[1] = 0.0	; Direction
MD24570 TRAFO5_AXIS1_1[2] = 1.0	; First rotary axis is parallel to Z
MD24572 TRAFO5_AXIS1_2[0] = 0.0	; Direction
MD24572 TRAFO5_AXIS1_2[1] = 1.0	; Second rotary axis is parallel to Y
MD24572 TRAFO5_AXIS1_2[2] = 0.0	;
MD25574 TRAFO5_BASE_ORIENT_1[0] = 0.0	;
MD25574 TRAFO5_BASE_ORIENT_1[1] = 0.0	; Basic tool orientation
MD25574 TRAFO5_BASE_ORIENT_1[2] = 1.0	; In Z direction

NC program

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.

```
SD42960 TOOL_TEMP_COMP[0] = -0.3                ; First compensation value
SD42960 TOOL_TEMP_COMP[1] = 0.0                 ;
SD42960 TOOL_TEMP_COMP[2] = -1.0                ; Second compensation value


                                                ; 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.

MD20390 TOOL_TEMP_COMP_ON = TRUE	; Temperature compensation active
Option	; Activate option
MD32750 TEMP_COMP_TYPE[AX1] = 4	; Compensation in tool direction
MD32750 TEMP_COMP_TYPE[AX2] = 4	; Compensation in tool direction
MD32750 TEMP_COMP_TYPE[AX3] = 4	; Compensation in tool direction

For more information about temperature compensation, please refer to:
References:
/FB2/ Description of Functions, Extended Functions; Compensations (K3).

For more information about generic 5-axis transformation, please refer to:
References:
/FB3/ Description of Functions, Special Functions; 3- to 5-axis Transformation (F2)

2.9 Sum offsets and setup offsets

2.9.1 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 MMC.

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 activated via machine data setting:
MD18080 $MN_MM_TOOL_MANAGEMENT_MASK, bit 8=1.

If kinematic transformations (e.g., 5-axis 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.

2.9.2 Description of function

Sum offsets

Several sum offsets (DL numbers) can be defined per D number. This allows you to determine, for example, **workpiece-location-dependent** 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 reset is performed
  - An operator panel start is performed

    or
  - 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 meaning 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 bit-coded 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.

2.9.3 Activation

Function

The function must be activated via a machine data.

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 in machine data:
MD20272 $MC_SUMCORR_DEFAULT
becomes active.

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: 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.

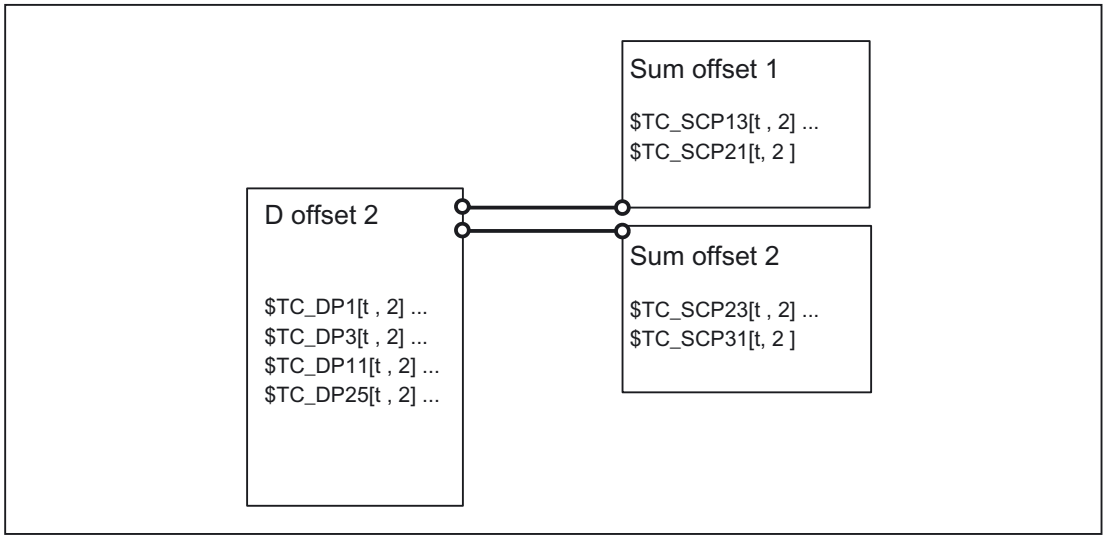


Fig. 2-49 MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4 = 0

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).

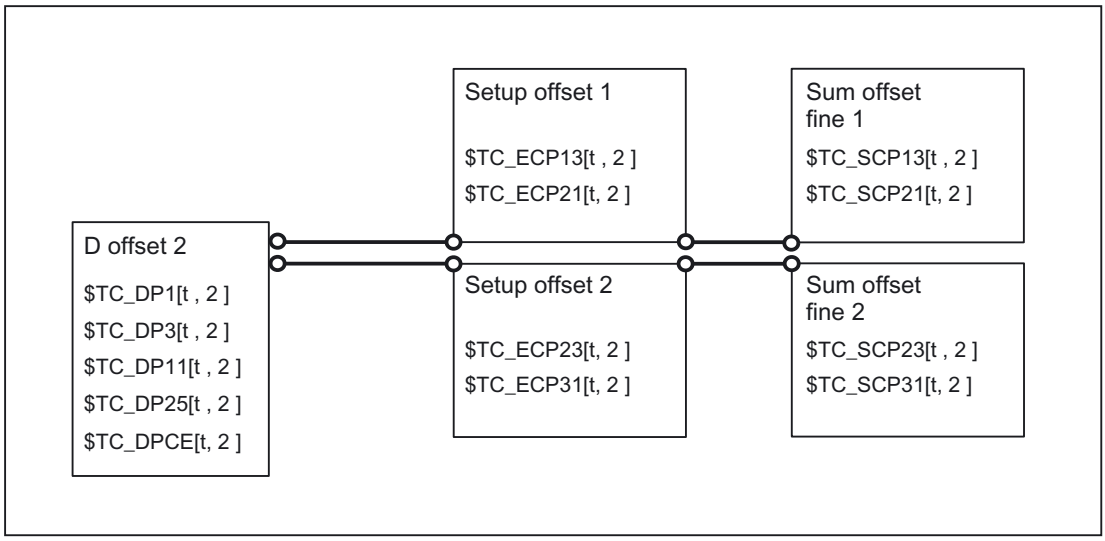


Fig. 2-50 MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4 = 1 "setup offsets" + "sum offsets fine"

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 meaning 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         ; Parameter y of setup offset x is assigned the
                                  value "r.r.". The other parameters of x have a
                                  value of 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 `DELDL` to 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 machine data:
MD20110 $MC_RESET_MODE_MASK
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.

2.9.4 Examples

Example 1

Machine data are to be used to define that no compensations and no sum offsets are active on a tool change
(MD20270 $MC_CUTTING_EDGE_DEFAULT=0,
MD20272 $MC_SUMCORR_DEFAULT=0):

```
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

Machine data are to be used to define that compensation D2 and sum offset DL=1 are activated on a tool change
(MD20270 $MC_CUTTING_EDGE_DEFAULT=2,
MD20272 $MC_SUMCORR_DEFAULT=1):

```
T5 M06              ;  Tool number 5 is loaded - D2 + DL=1 are active
                       (= values of MD).
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.
```

2.9.5 Upgrades for Tool Length Determination

2.10 Working with tool environments

2.10.1 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.

2.10.2 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 MD20360 $MC_TOOL_PARAMETER_DEF_MASK
- Machine data MD18112 $MN_MM_KIND_OF_SUMCORR
- Setting data SD42935 $SC_WEAR_TRANSFORM
- Setting data SD42950 $SC_TOOL_LENGTH_TYPE
- Setting data SD42940 $SC_LENGTH_CONST
- Setting data SD42900 $SC_MIRROR_TOOL_LENGTH
- Setting data SD42910 $SC_MIRROR_TOOL_WEAR
- Setting data SD42930 $SC_WEAR_SIGN
- Setting data SD42920 $SC_WEAR_SIGN_CUTPOS
- 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 machine data:
MD18105 $MN_MM_NUM_TOOL_ADAPTER
determines whether the adapter length or the tool base dimension is included in the tool length calculation.

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 parameters 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 location reserved for tool environments (MD18116 $MN_MM_NUM_TOOL_ENV = 0), i.e., the "Tool environments" functionality if 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.

2.10.3 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 location reserved for tool environments (MD18116 $MN_MM_NUM_TOOL_ENV = 0), i.e., the "Tool environments" functionality if 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 power-up with default machine data). There are no further automatic deletion operations (e.g., on RESET).

2.10.4 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 range of values is 0 to MD18116 $MN_MM_NUM_TOOL_ENV.

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, which does not refer to a defined data set, is passed, alarm 17020 (illegal array index1) is output.

2.10.5 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 location reserved for tool environments
(MD18116 $MN_MM_NUM_TOOL_ENV = 0),
i.e., the "Tool environments" functionality if 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 the following:
- 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 is stored in the data set specified by "_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.

2.10.6 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 location reserved for tool environments
(MD18116 $MN_MM_NUM_TOOL_ENV = 0),
i.e., the "Tool environments" functionality if 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 sign can only be used 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 meanings:

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

MD20100 $MC_DIAMETER_AX_DEF="X"
X is the 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 vector MD 24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2 is equivalent to the sum of the two vectors l_1 and l_3 for a toolholder with orientation capability type M.

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, the component assigned to the tool and the component assigned to the tool table is relevant.

A new machine data:
MD24558/24658 $MC_TRAFO5_JOINT_OFFSET_PART_1/2
has thus been introduced as an equivalent to vector l_3.

Machine data:
MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2
no longer corresponds to the sum of l_1 and l_3, but only to vector l_1.

If machine data:
MD24558/24658 $MC_TRAFO5_JOINT_OFFSET_PART_1/2 is equal to zero,
the new behavior is the same as before.

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.

2.11 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 meaning:

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 if setting data:
SD42900 $SC_MIRROR_TOOL_LENGTH
is enabled and mirroring is active.

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 format
status = LENTOAX(_AXIND, _MATRIX, "WCS")

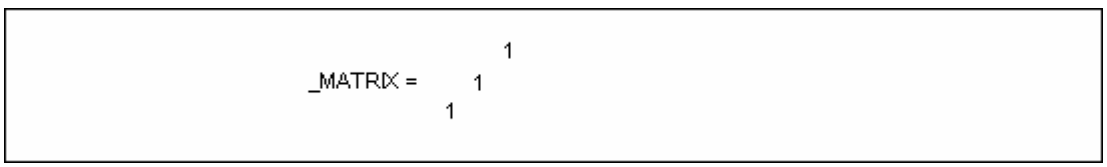
The result parameter **_AXIND** contains the values:
_AXIND[0] = 3

_AXIND[1] = 2

_AXIND[2] = 1

Or, in short: (3, 2, 1)

In this case, the associated matrix **_MATRIX** is:



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:

$$_MATRIX = \begin{matrix} sn(\,) & s(\,) \\ s(\,) & sn(\,) \end{matrix}$$
$$1$$

Note

For more information about the coordinate systems named above, please refer to:
References:
/PGA/ Programming Guide, Production Planning; "Tool Compensation"

Supplementary Conditions

3

3.1 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

When T numbers are output to the PLC, a synchronization mechanism is triggered in the NCK:
with absolute, indirect D programming the PLC supplies the D values via VDI.
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

Machine data:
$MC_AUXFU_AT_BLOCK_SEARCH_END
can be used to control the timing of the output of auxiliary functions to PLC at the end of a block search – automatically 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.

3.2 SD42935 expansions

SD42935

Setting data:
SD42935 $SC_WEAR_TRANSFORM (transformation of wear values)
can be used to define the wear components to be transformed or not transformed in connection with the `TOWMCS` and `TOWWCS` functions.

Examples

4

4.1 Toolholder with orientation capability

4.1.1 Example of orientational toolholders

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
```

4.1.2 Example of orientational toolholders with rotary table

Use of the MOV command

For use of the `MOVT` command it is assumed that the program is running on a 5-axis 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 component 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 that 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) about 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 X-Z and in the Y-Z plane each form an angle of +45 degrees or -45 degrees with the X or Y axis. Thus, the plane defined in this way is positioned so that the surface normal points in the direction of 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.

4.1.3 Example basic tool orientation

Basic orientation in the bisector

A milling tool is defined with length L1=10, whose basic orientation is in the bisector of the X/Z 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. Based on the tool length, the machine positions X0 Y0 Z10 result in this block.
In N80, an incremental traversing movement of 10 takes place in the tool direction. The resulting axis positions are thus X7.071 Y0 Z17.071.

4.1.4 Calculation of offsets according to location of use or workpiece

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. 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: 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.

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. The sum of the tool wear and insert offset evaluated in the WCS is then added as in block N410.

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 is X10.100 Y0.0071 Z0.0071. The vector generated from the tool wear and insert offset (X0 Y–0.505 Z0.875) is not affected by the plane change. The sum of the two vectors yields the dimension specified in the comment in N430.

4.2 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.

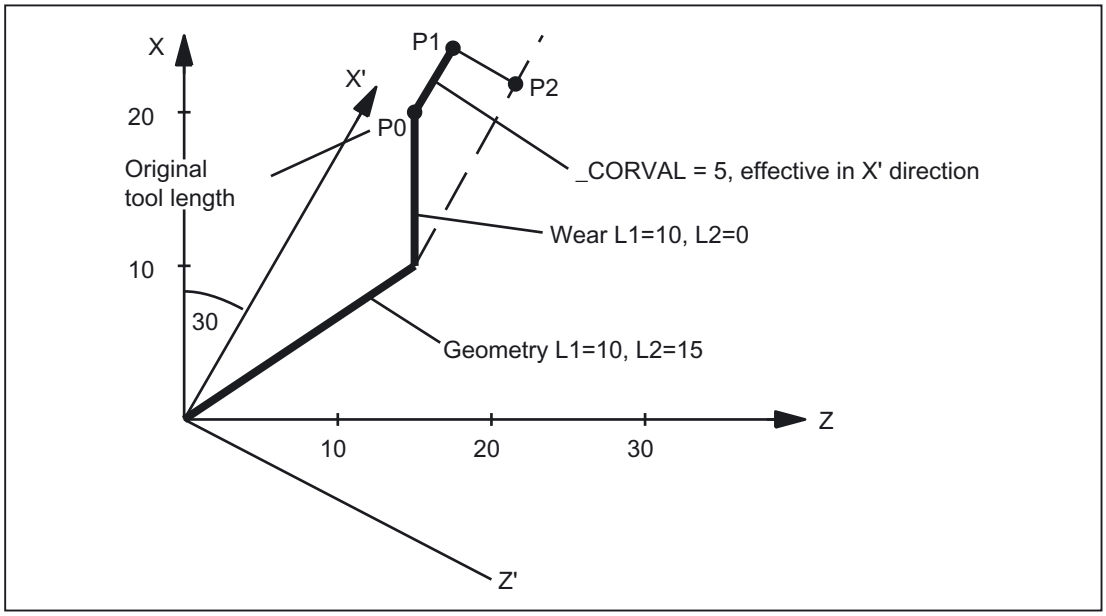


Fig. 4-1 Tool length compensation, example 6

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
```

Setting data:
SD42930 $SC_WEAR_SIGN
is enabled in N90, i.e., the wear must be evaluated 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:

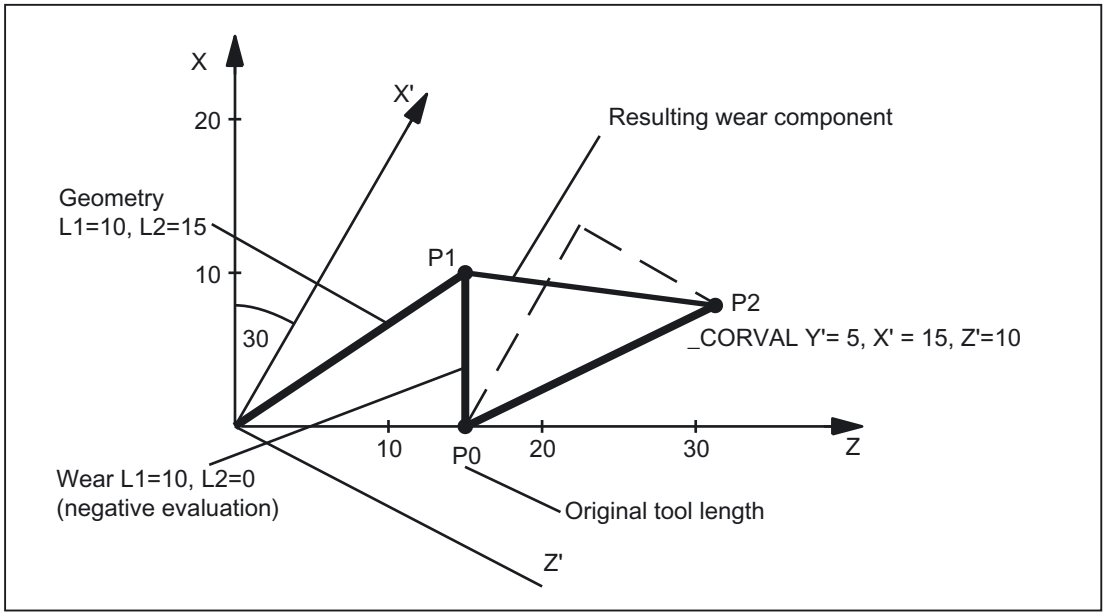


Fig. 4-2 Tool length compensation, example 8

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:

Let machine data:
MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters)
contain the value 2 (tool length must be evaluated with factor 0.5 in the diameter axis), let X be 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.

Data Lists

5

5.1 Machine data

5.1.1 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	Number of toolholders
18094	MM_NUM_CC_TDA_PARAM	Number of TOA data
18096	MM_NUM_CC_TOA_PARAM	Number of TOA data, which can be set up per tool and evaluated by the CC
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Tool compensations per TOA module
18102	MM_TYPE_OF_CUTTING_EDGE	Activate flat D number management
18105	MM_MAX_CUTTING_EDGE_NO	Address extension interpreted as spindle number
18106	MM_MAX_CUTTING_EDGE_PERTOOL	Maximum number of cutting edges per tool
18108	MM_NUM_SUMCORR	Number of all sum offsets in NCK
18110	MM_MAX_SUMCORR_PER_CUTTEDGE	Maximum number of sum offsets per cutting edge
18112	MM_KIND_OF_SUMCORR	Properties of sum offsets in the NCK
18114	MM_ENABLE_TOOL_ORIENT	Assign orientation to cutting edges
18116	MM_NUM_TOOL_ENV	Tool environments in TO area

5.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20096	T_M_ADDRESS_EXT_IS_SPINO	Spindle number as address extension
20110	RESET_MODE_MASK	Definition of control basic setting after power-up and RESET parts program end
20120	TOOL_RESET_VALUE	Definition of tool, for which tool length compensation is selected during power-up or on reset or parts program end as a function of machine data: MD20110 RESET_MODE-MASK.
20121	TOOL_PRESEL_RESET_VALUE	Definition of the preselected tool, for which the tool length compensation is selected during power-up and on reset or parts program end as a function of MD20110.
20126	TOOL_CARRIER_RESET_VALUE	Active toolholder on RESET
20130	CUTTING_EDGE_RESET_VALUE	Definition of tool cutting edge, for which tool length compensation is selected during power-up or on reset or parts program end as a function of machine data: MD20110 RESET_MODE-MASK.
20132	SUMCORR_RESET_VALUE	Number for selecting sum offset
20140	TRAFO_RESET_VALUE	Definition of transformation set, which is selected during power-up or on reset or parts program end as a function of machine data: MD20110 RESET_MODE-MASK.
20180	TOCARR_ROT_ANGLE_INCR[i]	Value of the minimum incremental step with toolholder with orientation capability
20182	TOCARR_ROT_ANGLE_OFFSET[i]	Offset of the rotary axis with toolholder with orientation capability
20184	TOCARR_BASE_FRAME_NUMBER	Base frame of the table offset for toolholder with orientation capability with rotary table
20188	TOCARR_FINE_LIM_LIN	Limit linear fine offset TCARR
20190	TOCARR_FINE_LIM_ROT	Limit rotary fine offset TCARR
20202	WAB_MAXNUM_DUMMY_BLOCKS	Maximum number of blocks with no traversing motions with SAR
20204	WAB_CLEARANCE_TOLERANCE	
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value for tool radius compensation
20230	CUTCOM_CURVE_INSERT_LIMIT	Minimum value for intersection calculation with tool radius compensation
20240	CUTCOM_MAXNUM_CHECK_BLOCKS	Blocks for predictive contour calculation with tool radius compensation
20250	CUTCOM_MAXNUM_DUMMY_BLOCKS	Max. no. of dummy blocks with no traversing movements
20252	CUTCOM_MAXNUM_SUPPR_BLOCKS	Maximum number of blocks with compensation suppression

Number	Identifier: $MC_	Description
20256	CUTCOM_INTERS_POLY_ENABLE	Intersection process possible for polynomials
20270	CUTTING_EDGE_DEFAULT	Selected cutting edge after tool change
20272	SUMCORR_DEFAULT	Number for activating a new cutting edge compensation
20360	TOOL_PARAMETER_DEF_MASK	Defines the effect 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	Limit angle between path tangent and tool orientation with 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	Error response on programmed 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	Assignment of TO unit to a channel

5.1.3 Axis-/spindle-specific machine data

Number	Identifier: $MA_	Description
32750	TEMP_COMP_TYPE	Temperature compensation type

5.2 Setting data

5.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42442	TOOL_OFFSET_INCR_PROG	Retraction of tool compensations on incremental programming
42470	CRIT_SPLINE_ANGLE	Corner limit angle for the compressor with COMPCAD (changed in SW 7.1 and higher)
42480	STOP_CUTCOM_STOPRE	Alarm response for tool radius compensation and preprocessing stop
42494	CUTCOM_ACT_DEACT_CTRL	Approach/retraction behavior for tool radius compensation in blocks without travel information
42496	CUTCOM_CLSDT_CONT	Maintain stability of approach/retraction behavior for tool radius compensation in blocks
42900	MIRROR_TOOL_LENGTH	Mirror tool length compensation
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
42935	WEAR_TRANSFORM	Transformation of wear values
42940	TOOL_LENGTH_CONST	Retain the assignment of tool length components when changing the machining plane (G17 to G19)
42950	TOOL_LENGTH_TYPE	Assignment of the tool length components independent of tool type
42960	TOOL_TEMP_COMP	Temperature compensation value in relation to tool
42974	TCARR_FINE_CORRECTION	Fine offset TCARR on/off

5.3 Signals

5.3.1 Signals from channel

DB number	Byte.Bit	Name
21, ...	61.0	T function 1 change
21, ...	62.0	D function 1 change
21, ...	116-119	T function 1
21, ...	128-129	D function 1
21, ...	214	Active G function of group 7
21, ...	223	Active G function of group 16
21, ...	224	Active G function of group 17
21, ...	225	Active G function of group 18
21, ...	230	Active G function of group 23

Index

$

$A_DNO[n], 2-18
$A_MONIFACT, 3-1
$P_TOOLENV, 2-142
$P_TOOLENVN, 2-142
$TC_DPCE[t,d], 2-8

A

Alarm for preprocessing stop, 2-74
Assignment tool/toolholder, 2-80

B

Basic orientation of tools, 2-112
Basic tool orientation, 2-111, 4-5

C

Collision detection, 2-67
Compatibility
    GETTCOR, 2-149
Compensation number, 2-8
Constant curvature, 2-42
Constant tangent, 2-42
Cutting edge number, 2-8

D

D functions, 2-2
D number structure
    - flat (without tool management), 2-17
D numbers
    Allocation of free ..., 2-7
DELTOOLENV, 2-141
Description of a rotation, 2-92
Direction vector, 2-112
DISC, 2-61

F

Flat D number structure, 2-17
    D number programming, 2-19
    Read D number from the PLC, 2-18
Frame change, 2-82
Frame rotation, 2-95

G

G40, 2-40
G41, 2-40
G42, 2-40
G450/G451, 2-60
G451, 2-63
G460, 2-76
G461, 1-3, 2-77
G462, 1-3, 2-78
G91 extension, 2-108
    Zero point offset, 2-110
GETTCOR, 2-144
GETTENV, 2-143

I

Incrementally programmed compensation values, 2-108

K

Keep tool radius compensation constant, 2-71
Kinematic interaction and machine design, 2-87
Kinematic type, 2-84
KONT, 2-41
KONTC, 2-41
KONTT, 2-41

L

Length of cutting edge
    relevant, 2-37

LENTOAX, 2-149
Limited toolholder orientation, 2-83

M

Machine kinematics, 2-84
Machine with rotary tool, 2-88
Machine with rotary workpiece, 2-89
Machines with extended kinematics, 2-89
Machining in direction of tool orientation, 2-110
MD11346, 2-16
MD11410, 2-68
MD18080, 2-127
MD18088, 2-85, 2-99, 2-104
MD18100, 2-3, 2-18
MD18102, 2-10, 2-17, 2-19
MD18104, 2-139
MD18105, 2-7
MD18106, 2-7
MD18112, 2-139
MD18114, 2-112
MD18116, 2-140, 2-141, 2-142, 2-143, 2-144
MD20096, 2-5, 2-6
MD20110, 2-2, 2-23, 2-24, 2-106, 2-109, 2-137
MD20112, 2-23, 2-106
MD20120, 2-23, 2-106
MD20121, 2-2, 2-23
MD20126, 2-106
MD20130, 2-23
MD20180, 2-96
MD20184, 2-98
MD20188, 2-91
MD20190, 2-91
MD20202, 2-57
MD20204, 2-50
MD20210, 2-63
MD20220, 2-62
MD20230, 2-64
MD20240, 2-65, 2-66, 2-67
MD20250, 2-40
MD20256, 2-75
MD20270, 2-3, 2-14, 2-23, 2-137
MD20272, 2-131, 2-137
MD20360, 2-16, 2-17, 2-139, 2-147, 4-14
MD20390, 2-122
MD20392, 2-123
MD22530, 2-107
MD22550, 2-1, 2-13, 2-14, 2-23, 2-24
MD22560, 2-1, 2-23, 2-24
MD22562, 2-15
MD24550, 2-148
MD24558, 2-148
MD24560, 2-148
MD24650, 2-148
MD24658, 2-148
MD24660, 2-148
MD28082, 2-98
MD28085, 2-3, 2-100, 2-105
MD32750, 2-122

N

NORM, 2-41

P

Parameter sets, 2-83
Polynomial
  Intersection procedure, 2-75
Position programming, 2-95
Programming, 2-104

R

Representation of the kinematic chain, 2-87
Rotary axis parameters, 2-84
Rotation component in tool direction, 2-95

S

SD42442, 2-109
SD42480, 2-74
SD42496, 2-70
SD42900, 1-2, 2-116, 2-139, 2-151
SD42910, 2-113, 2-116, 2-117, 2-139
SD42920, 2-116, 2-117, 2-139
SD42930, 2-116, 2-118, 2-139, 4-13
SD42935, 2-116, 2-139, 3-2, 4-7
SD42940, 2-113, 2-116, 2-119, 2-139
SD42950, 2-113, 2-116, 2-139
SD42960, 1-2, 2-116, 2-123, 2-124
SD42974, 2-91
Selection of the cutting edge when changing tool, 2-3
Slotting saw, 2-29
Smooth approach and retraction
  Approach movement, 2-47
  Meaning, 2-46
  Retraction movement, 2-47
Storing angles in the toolholder data, 2-83
System parameters for toolholders with orientation capability, 2-85
System variables for the user and for fine offsets, 2-86

T

T function, 2-1
Taking the compensation values into account location-specifically and workpiece-specifically, 4-5
TCARR, 2-97
TCOABS, 2-97
TCOFR, 2-97
TOA, 2-3
Tool, 2-1
  Change, 2-1
  Change tool with M06, 2-1
  Cutting edge, 2-26
  DISC, 2-61
  Length compensation, 2-31
  Length of cutting edge, 2-30
  Radius compensation, 2-33
  Select, 2-1
  T function, 2-2
  Tool base dimension/adapter dimension, 2-35
  Tool cutting edge, 2-2
  Tool radius compensation 2D (TRC), 2-39
  Tool type, 2-27
  Undercut angle, 2-36
  Wear, 2-34
Tool change
  Compensation memory, 2-3
  D function, 2-2
Tool compensation
  Calculation, 2-5
Tool compensation types, 2-82
Tool environments, 2-138
Tool length compensation, 2-31
Tool radius compensation
  2D:approach and retraction behavior, 2-41
  2D:collision detection, 2-67
  2D:compensation at inside corners, 2-65
  2D:compensation at outside corners, 2-60
  2D:deselection, 2-60
  2D:intersection G451, 2-63
  2D:selection, 2-40
  2D:smooth approach and retraction, 2-46
  2D:transition circle, 2-61
  2D:variable compensation value, 2-69, 2-95
  Modified alarm response, 2-74
Tool radius compensation (TRC), 2-39
Tool type
  Slotting saw, 2-29
TOOL_PARAMETER_DEF_MASK, 2-147
TOOLENV, 2-139
Toolholder
  With orientation capability, 2-80
Toolholder selection, 2-80
Toolholder with orientation capability, 1-2, 2-80, 4-1
  Control system response on Reset, program start, REPOS, 2-105
  Programming, 2-96, 2-99, 2-104, 4-2, 4-5
  Supplementary conditions, 2-95, 2-99, 2-105
Toolholders with orientation capability with rotary table, 4-2
Tools with a relevant tool point direction, 2-37

Z

Zero vectors, 2-87

SIEMENS

Brief Description 1

Detailed Description 2

SINUMERIK 840D sl/840D/840Di/810D

NC/PLC Interface Signals (Z1)

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D/840DE	7.3
NCU system software for 840Di/840DiE	3.2
NCU system software for 810D/810DE	7.3

08/2005 Edition
6FC5397-0BP10-0BA0

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Caution

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

Notice

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

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

Qualified Personnel

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

Prescribed Usage

Note the following:



Warning

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG
GERMANY

Order No.: 6FC5397-0BP10-0BA0
08/2005 Edition

Copyright © Siemens AG, 2005.
Technical data subject to change

Contents

1 Brief Description 1-1
1.1 Brief description 1-1

2 Detailed Description 2-1
2.1 Various interface signals and functions (A2) 2-1
2.1.1 Signals from PLC to NC (DB10) 2-1
2.1.2 Selection/status signals from MMC to PLC (DB10) 2-2
2.1.3 Signals from NC to PLC (DB10) 2-3
2.1.4 Signals to OP (DB19) 2-7
2.1.5 Signals from OP (DB19) 2-17
2.1.6 Signals to channel (DB21, ...) 2-25
2.1.7 Signals from channel (DB21, ...) 2-25
2.1.8 Signals to axis/spindle (DB31, ...) 2-27
2.1.9 Signals from axis/spindle (DB31, ...) 2-42
2.2 Axis monitoring, protection zones (A3) 2-64
2.2.1 Signals to channel (DB21, ...) 2-64
2.2.2 Signals from channel (DB21, ...) 2-65
2.2.3 Signals to axis/spindle (DB31, ...) 2-67
2.2.4 Signals from axis/spindle (DB31, ...) 2-69
2.3 Continuous-path mode, exact stop and LookAhead (B1) 2-69
2.3.1 Signals from channel (DB21, ...) 2-69
2.3.2 Signals from axis/spindle (DB31, ...) 2-70
2.4 Acceleration (B2) 2-71
2.5 Diagnostic tools (D1) 2-71
2.6 Travel to fixed stop (F1) 2-71
2.6.1 Signals to axis/spindle (DB31, ...) 2-71
2.6.2 Signals from axis/spindle (DB31, ...) 2-73
2.7 Velocities, setpoint/actual value systems, closed-loop control (G2) 2-73
2.8 Auxiliary function output to PLC (H2) 2-74
2.8.1 Signals to channel (DB21, ...) 2-74
2.8.2 Signals from channel (DB21, ...) 2-74
2.8.3 Signals from axis/spindle (DB31, ...) 2-79
2.9 Mode group, channel, program operation, reset response (K1) 2-80
2.9.1 Signals to mode group (DB11) 2-80
2.9.2 Signals from mode group (DB11) 2-85
2.9.3 Signals to channel (DB21, ...) 2-90
2.9.4 Signals from channel (DB21, ...) 2-97
2.9.5 Signals to axis/spindle (DB31, ...) 2-110
2.9.6 Signals from axis/spindle (DB31, ...) 2-110
2.10 Axis types, coordinate systems, frames (K2) 2-112
2.10.1 Signals to axis/spindle (DB31, ...) 2-112

2.11 EMERGENCY STOP (N2) 2-113
2.11.1 Signals to NC (DB10) 2-113
2.11.2 Signals from NC (DB10) 2-114
2.12 Transverse axes (P1) 2-114
2.13 PLC basic program (P3) 2-115
2.14 Reference point approach (R1) 2-115
2.14.1 Signals to channel (DB21, ...) 2-115
2.14.2 Signals from channel (DB21, ...) 2-116
2.14.3 Signals to axis/spindle (DB31, ...) 2-117
2.14.4 Signals from axis/spindle (DB31, ...) 2-118
2.15 Spindles (S1) 2-119
2.15.1 Signals to axis/spindle (DB31, ...) 2-119
2.15.2 Signals from axis/spindle (DB31, ...) 2-126
2.16 Feeds (V1) 2-133
2.16.1 Signals to channel (DB21, ...) 2-133
2.16.2 Signals to axis/spindle (DB31, ...) 2-141
2.16.3 Signals from axis/spindle (DB31, ...) 2-147
2.17 Tool offset (W1) 2-148
Index Index-1

Brief Description

1

1.1 Brief description

The signal description contains a detailed description of the NC/PLC interface signals:

- PLC messages (DB2)
- NC (DB10)
- Mode group (DB11)
- OP (DB19)
- NCK channel (DB21-DB30)
- Axis/spindle (DB31-DB61)
- Loading/unloading a magazine (DB71)
- Spindle as change point (DB72)
- Tool turret (DB73)

Detailed Description 2

2.1 Various interface signals and functions (A2)

2.1.1 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	Access to certain data types can be disabled depending on the key-operated switch position. Input, changing, deleting data as well as certain operations on the operator panel can be disabled for certain user groups in this way. Key-operated switch position 0 offers the fewest access rights and position 3 the most access rights. The signals "key-operated switch positions 1 to 3" can either be entered directly from the key-operated switch on the machine control panel or from the PLC user program. Only one interface signal may be set in each case. If several positions are set simultaneously (interface signals), then they are no longer valid and key-operated switch position 3 is automatically set by the control. The data areas that can be disabled are assigned to the key-operated switch positions using the MMC machine data for protection levels. The following signal combinations apply:
Application example(s)	Depending on the rights assigned to the operator, programmer or installation engineer, certain functions will be disabled by the key-operated switch. Unintentional changes to data (e.g. zero offsets) or activation of program conditions (e.g. selecting dry run feed rate) by the operator can therefore be prevented.
Corresponding to	Disabling using a password

The following signal combinations apply:

	DB10, DBB56			
Key-operated switch position	Bit 7	Bit 6	Bit 5	Bit 4
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

2.1.2 Selection/status signals from MMC to PLC (DB10)

DB10 DBX103.0	**MMC alarm is active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	At least 1 MMC alarm (alarm numbers 100 000 to 105 999) is active.
Signal state 0 or edge change 1 → 0	No MMC alarm is active.

DB10 DBX103.5	**AT box ready**
Edge evaluation: no	Signal(s) updated: cyclic
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.
References	/BH/ Operator Components, Manual; Chapter: "PCU 50 components, spare parts"

DB10 DBX103.6	**MMC temperature limit**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The ambient conditions of the MMC 101/102 limits are within the permissible tolerance range of 5 to 55 degrees C.
Signal state 0 or edge change 1 → 0	The MMC temperature limit was either fallen below or exceeded. The MMC temperature monitoring has responded and disabled the MMC.

DB10 DBX103.7	MMC101/102 battery alarm
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The MMC battery monitoring function has responded. In the event of a power failure, the most recently modified data and a correct device configuration may be lost. A corresponding MMC alarm is signaled. Check the MMC backup battery. An insufficient (low) battery voltage also affects the actual time on the MMC.
Signal state 0 or edge change 1 → 0	No MMC battery alarm is active.
Signal irrelevant for ...	SINUMERIK 840/840Di/810D with PCU 50, PCU 20 or PCU 70
References	/BH/ Operator Components, Manual, Chapter: "PCU 50 component, replace battery"

2.1.3 Signals from NC to PLC (DB10)

DB10 DBX104.7	NCK-CPU-Ready
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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.
Signal state 0 or edge change 1 → 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 initiated by the PLC basic program: • Status signals from the NCK to the PLC (user interface) are deleted (cleared) • Change signals for the help functions are deleted • Cyclic processing of the user interface PLC to NCK is terminated.
Application example(s)	Individual PLC outputs can, for example, be set to a defined state from the PLC user program.
References	/DA/ Diagnostics Guide /FB1/ Description of Functions Basic Machine; PLC Basic Program (P3)

DB10 DBX108.1	**MMC CPU2 Ready (to OPI or MPI)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The MMC2 CPU is ready and registers itself cyclically with the NCK.
Signal state 0 or edge change 1 → 0	The MMC CPU is not ready.

DB10 DBX108.2 - DBX108.3	**MMC CPU1 Ready (MMC to MPI)/ MMC CPU1 Ready (MMC to OPI, standard connection)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The MMC CPU is ready and registers itself cyclically with the NCK.
Signal state 0 or edge change 1 → 0	The MMC CPU is not ready.
Application example(s)	The appropriate measures can be introduced by the PLC user program if "MMC-CPU1-Ready" = 0.
References	/DA/ Diagnostics Guide

DB10 DBX108.6	**611D-Ready**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	All existing drives signal the status drive ready (summary of axial interface signals "DRIVE ready").
Signal state 0 or edge change 1 → 0	As soon as the drive not ready status is signaled from a drive (i.e. IS "DRIVE ready" = 0).
Signal irrelevant for ...	SINUMERIK 840Di
Corresponding to	DB31, ... DBX93.5 (DRIVE ready)

DB10 DBX108.7	**NC Ready**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 control have been established • The control is in the cyclic mode
Signal state 0 or edge change 1 → 0	The control 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) • NC CPU watchdog If the signal "NC Ready" goes to 0 the following measures are introduced by the control 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 control is not ready again until after POWER ON.
Corresponding to	Relay contact "NC Ready"
References	/DA/ Diagnostics Guide /FB1/ Description of Functions Basic Machine; PLC Basic Program (P3)

DB10 DBX109.0	**NCK alarm is active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	At least one NCK alarm is present. This is a group signal for the interface signals DB21, ... DBX36.6 (channel-specific NCK alarm is active) of all existing channels.
Signal state 0 or edge change 1 → 0	No NCK alarm is active.
Corresponding to	DB21, ... DBX36.6 (channel-specific NCK alarm present) DB21, ... DBX36.6 (NCK alarm with processing stop present)
References	/DA/ Diagnostics Guide

DB10 DBX109.5	**Heatsink temperature alarm NCU 573**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The limit values of the heatsink temperature were exceeded on the NCU 573. The heatsink monitoring function has responded and the continuous operation of the operator panel is no longer assured.
Signal state 0 or edge change 1 → 0	The heatsink monitoring function of the NCU 573 has not responded.

DB10 DBX109.6	**Air temperature alarm**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The ambient temperature or fan monitoring function has responded. This could be due to the following: • 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 V DC 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 or edge change 1 → 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.
References	/DA/ Diagnostics Guide

DB10 DBX109.7	**NCK battery alarm**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The NCK battery voltage monitoring function has responded. This could be due to the following: • 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 control 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 control ran-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 or edge change 1 → 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	/DA/ Diagnostics Guide /IAD/ SINUMERIK 840D, Startup Guide /IBI/ SINUMERIK 840Di, Startup Guide /IAG/ SINUMERIK 810D Startup Guide /IAF/ SINUMERIK FMNC, Start-Up Guide

2.1.4 Signals to OP (DB19)

DB19 DBX0.0	**Screen bright**
Edge evaluation: no	Signal(s) updated: cyclic
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: cyclic
Signal state 1 or edge change 0 → 1	The screen is switched to "dark" by the PLC user program. The automatic screen blanking function is therefore inactive: i. e. the screen does not automatically become bright (blanking off) when a key on the keyboard is pressed.
Signal state 0 or edge change 1 → 0	The screen is switched to "bright" by the PLC user program. With this signal status, screen blanking from the control can be automatically derived from the keyboard. If a key is not pressed on the keyboard for a time that is defined using machine data: MD9006 $MM_DISPLAY_BLACK_TIME (time to blanken the screen), then the screen is blanked. The screen is switched to bright again (screen blanking off) the next time a key on the operator panel is pressed.
Application example(s)	Screen saver
Special cases, errors, ...	Notice: If the interface signal "Darken screen" DB19, DBX0.1 (darken screen) = 1, then the keyboard of the operator panel remains operational! 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: cyclic
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 blanked using the interface signal: DB19, DBX0.1 (darken screen), then, at the same time, the keyboard should be disabled using the interface signal: DB19, DBX0.2 (key disable), in order to prevent the operator from unintentionally entering keyboard commands.
Corresponding to	DB19, DBX0.1 (darken screen)

DB19 DBX0.3	**Clear cancel alarms**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Clear error key on the machine control panel is pressed. The master control receives the bits and transmits the "Cancel" key to the MBDD, which then acknowledges all Cancel alarms of the NCK and MMC. The PLC application acknowledges the PLC alarms itself. PowerOn 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 applicable to HMI Advanced or PCU 50.
Corresponding to	DB19, DBX20.3 (cancel alarm cleared)

DB19 DBX0.4	**Clear recall alarms**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The Clear error key is pressed on the machine control panel. The master control receives the bits and transmits the "Cancel" key to the MBDD, which then acknowledges all Cancel alarms of the NCK and MMC. 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 value in WCS**
Edge evaluation: no	Signal(s) updated: cyclic
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.

DB19 DBX0.7	**Actual value in WCS**
Corresponding to	DB19, DBX20.7 (change over MCS/WCS)
Additional references	/BA/ Operator's Guide

DB19 DBW2	**Higraph first error display**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBW4	**Higraph first error display**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX6.0...7	**Analog spindle 1, utilization in percent**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX7.0 - DBX7.7	**Analog spindle 2, utilization in percent**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX8.0 - DBX8.7	**Channel number of the machine control panel on the MMC**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX10.0	**Programming area selection**
Edge evaluation: no	Signal(s) updated: cyclic
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 DBX10.1	**Alarm area selection**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Alarm area selection active
Signal state 0 or edge change 1 → 0	Alarm area selection inactive

DB19 DBX10.2	**Tool offset selection**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Tool offset selection active
Signal state 0 or edge change 1 → 0	Tool offset selection inactive

DB19 DBX10.7	**ShopMill control signal**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX12.2	**COM2**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	COM2 active
Signal state 0 or edge change 1 → 0	COM2 inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated via RS-232-C.

DB19 DBX12.3	**COM1**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	COM1 active
Signal state 0 or edge change 1 → 0	COM1 inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated via RS-232-C.

DB19 DBX12.4	**RS-232-C Stop**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS-232-C stop active
Signal state 0 or edge change 1 → 0	RS-232-C stop inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated via RS-232-C.

DB19 DBX12.5	**RS-232-C external**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS-232-C external active
Signal state 0 or edge change 1 → 0	RS-232-C external inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated via RS-232-C.

DB19 DBX12.6	**RS-232-C off**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS-232-C off active
Signal state 0 or edge change 1 → 0	RS-232-C off inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated via RS-232-C.

DB19 DBX12.7	**RS-232-C on**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS-232-C on active
Signal state 0 or edge change 1 → 0	RS-232-C on inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated via RS-232-C.

DB19 DBX13.5	**Unload**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Unload active
Signal state 0 or edge change 1 → 0	Unloading inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated using the hard disk.

DB19 DBX13.6	**Load part program**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Load active
Signal state 0 or edge change 1 → 0	Loading inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated using the hard disk.

DB19 DBX13.7	**Selection**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Selection active
Signal state 0 or edge change 1 → 0	Selection inactive
Application example(s)	Valid for HMI Embedded; a file transfer can be initiated using the hard disk.

DB19 DBX14.0 - DBX14.6	**PLC index**
Edge evaluation: no	Signal(s) updated: cyclic
Description	This byte for controlling the RS-232-C defines the PLC index which specifies the axis, channel or TO No. for the standard control file. This file is handled from PLC → MMC in accordance with the task stored in DB19.DBB12.
Application example(s)	Valid for HMI Embedded, with reference to DB19.DBB12 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: cyclic
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)	Valid for HMI Embedded, with reference to DB19.DBB12

DB19 DBX15.0 - DBX15.7	**PLC line offset**
Edge evaluation: no	Signal(s) updated: cyclic
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)	Valid for HMI Embedded, with reference to DB19.DBB12 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: cyclic
Description	This byte for controlling file transfer via hard disk defines the index for the control file (job list). This file is handled from PLC → MMC in accordance with the task stored in DB19.DBB13.
Application example(s)	Valid for HMI Advanced, with reference to DB19.DBB13 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: cyclic
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 DBX17.0 - DBX17.7	**PLC line offset in the user control file**
Edge evaluation: no	Signal(s) updated: cyclic
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)	Valid for HMI Advanced, with reference to DB19.DBB13 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: cyclic
Signal state 1 or edge change 0 → 1	Mode change disable active
Signal state 0 or edge change 1 → 0	Mode change disable inactive

DB19 DBX45.0	**FC9 Out: Active**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX45.1	**FC9 Out: Done**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX45.2	**FC9 Out: Error**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX45.3	**FC9 Out: StartError**
Edge evaluation: no	Signal(s) updated: cyclic

2.1.5 Signals from OP (DB19)

DB19 DBX20.1	**Screen is dark**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The screen is darkened.
Signal state 0 or edge change 1 → 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 cleared**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Cancel alarm deleted active
Signal state 0 or edge change 1 → 0	Cancel alarm deleted inactive

DB19 DBX20.4	**Recall alarm cleared**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Recall alarm deleted active
Signal state 0 or edge change 1 → 0	Recall alarm deleted inactive

DB19 DBX20.6	**Simulation selected**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	On entry to simulation = 1
Signal state 0 or edge change 1 → 0	On exit from simulation = 0
Application example(s)	Can be evaluated by machine manufacturer in order to activate the test on NC start. The following must be set in the drive machine data: MD1012 $MD_FUNC_SWITCH, bit 2 = 0. Status "Ext. pulse disable active, terminal 663 open" is then not transmitted to the NC.
Corresponding to	MD1012 $MD_FUNC_SWITCH, bit 2

DB19 DBX20.7	**Switch over MCS/WCS**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 1 PLC cycle.
Signal state 0 or edge change 1 → 0	No effect
Application example(s)	The interface signal: DB19, DBX20.7 (change over MCS/WCS) must be transferred to the interface signal: DB19, DBX0.7 (actual value in WCS) in order that the changeover becomes effective.
Corresponding to	DB19, DBX0.7 (actual value in WCS)

DB19 DBX22.0 - DBX22.7	**Displayed channel number from the MMC**
Edge evaluation: no	Signal(s) updated: cyclic

DB19 DBX24.0	**Error (status RS-232-C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Error status active
Signal state 0 or edge change 1 → 0	Error status inactive
Application example(s)	The actual status of the RS 232 C is sent to the PLC in the acknowledgment byte DB19.DBB24.
Corresponding to	Applies to HMI Embedded

DB19 DBX24.1	**O.K. (status RS-232-C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	O.K. status active
Signal state 0 or edge change 1 → 0	O.K. status inactive
Application example(s)	The actual status of the RS 232C is sent to the PLC in acknowledgment byte DB19.DBB24.
Corresponding to	Applies to HMI Embedded

DB19 DBX24.2	**COM2 (status RS-232-C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	COM2 active
Signal state 0 or edge change 1 → 0	COM2 inactive
Application example(s)	The actual status of the RS 232C is sent to the PLC in acknowledgment byte DB19.DBB24.
Corresponding to	Applies to HMI Embedded

DB19 DBX24.3	**COM1 (status RS-232-C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	COM1 active
Signal state 0 or edge change 1 → 0	COM1 inactive
Application example(s)	The actual status of the RS 232C is sent to the PLC in acknowledgment byte DB19.DBB24.
Corresponding to	Applies to HMI Embedded

DB19 DBX24.4	**RS-232-C stop (status RS-232-C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS-232-C stop active
Signal state 0 or edge change 1 → 0	RS-232-C stop inactive
Application example(s)	The actual status of the RS 232C is sent to the PLC in acknowledgment byte DB19.DBB24.
Corresponding to	Applies to HMI Embedded

DB19 DBX24.5	**RS232C external (status RS232C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS-232-C external active
Signal state 0 or edge change 1 → 0	RS-232-C external inactive
Application example(s)	The actual status of the RS 232C is sent to the PLC in acknowledgment byte DB19.DBB24. The bit "RS-232-C external" is delayed until the transfer of the part program to be externally executed has been started and the selection has been made. Only then is an "NC Start" possible.
Corresponding to	Applies to HMI Embedded

DB19 DBX24.6	**RS-232-C off (status RS-232-C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS-232-C off active
Signal state 0 or edge change 1 → 0	RS-232-C off inactive
Application example(s)	The actual status of the RS 232C is sent to the PLC in acknowledgment byte DB19.DBB24.
Corresponding to	Applies to HMI Embedded

DB19 DBX24.7	**RS-232-C on (status RS-232-C from PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	RS232C on active
Signal state 0 or edge change 1 → 0	RS232C on inactive
Application example(s)	The actual status of the RS 232C is sent to the PLC in acknowledgment byte DB19.DBB24.
Corresponding to	Applies to HMI Embedded

DB19 DBX25.0 - DBX25.7	**Error RS-232-C**
Edge evaluation: no	Signal(s) updated: cyclic
Description	0 = no error 1 = invalid number for the control file (value in DB19.DBB14<127 or invalid) 2 = DB19.DBB15 could not be read 3 = control file /BD.DIR/PLC_IN_OUT_xxx.TEA not found (value in DB19.DBB14 invalid) 4 = invalid index in control file. (value in DB19.DBB15 incorrect) 5 = selected job list in the control file could not be opened (only HMI Advanced) 6 = error in the job list (job list interpreter returns error) (only HMI Advanced) 7 = job list interpreter signals empty job list (only HMI Advanced) 8 = error during the RS-232-C transmission The error text is entered into the DIENSTE PROTOKOLL [SERVICES REPORT]. 9 = error when executing job list (only HMI Advanced)

DB19 DBX25.0 - DBX25.7	**Error RS-232-C**
Corresponding to	Applies to HMI Embedded

DB19 DBX26.1	**OK (job list selection from PLC, status)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Transfer correctly completed
Signal state 0 or edge change 1 → 0	Transfer completed with error
Corresponding to	Valid for HMI Advanced

DB19 DBX26.2	**Error (job list selection from PLC, status) (previously bit 0)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Transfer completed with error
Signal state 0 or edge change 1 → 0	Transfer correctly completed
Corresponding to	Valid for HMI Advanced

DB19 DBX26.3	**Active (job list selection from PLC, status)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Job in progress
Signal state 0 or edge change 1 → 0	No task in progress
Corresponding to	Valid for HMI Advanced

DB19 DBX26.5	**Unload (job list selection from PLC, status)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Unload active
Signal state 0 or edge change 1 → 0	Unloading inactive
Corresponding to	Valid for HMI Advanced

DB19 DBX26.6	**Load (job list selection from PLC, status)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Load active
Signal state 0 or edge change 1 → 0	Loading inactive
Corresponding to	Valid for HMI Advanced

DB19 DBX26.7	**Selection (job list selection from PLC, status)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Selection active
Signal state 0 or edge change 1 → 0	Selection inactive
Corresponding to	Valid for HMI Advanced

DB19 DBX27.0 - DBX27.7	**Data transfer error**
Edge evaluation: no	Signal(s) updated: cyclic
Meaning	• 0 = no error • 1 = invalid number for control file (value in DB19.DBB16<127 or invalid) • 2 = DB19.DBB17 could not be read • 3 = control file /BD.DIR/PLC_IN_OUT_xxx.TEA not found (value in DB19.DBB16 invalid) • 4 = invalid index in control file (value in DB19.DBB17 incorrect) • 5 = selected job list in control file could not be opened • 6 = error in the job list (job list interpreter signals an error) • 7 = job list interpreter signals an empty job list • 8 = data transfer error The error text is entered into the DIENSTE PROTOKOLL [SERVICES REPORT]. • 9 = error while executing the job list
Corresponding to	Valid for HMI Advanced

DB19 DBX40.0 - DBX40.7	**Mode group number**
Edge evaluation: no	Signal(s) updated: cyclic
Meaning	This byte contains the mode group number.

DB19 DBX41.0 - DBX41.7	**Channel number (FC9: ChanNo)**
Edge evaluation: no	Signal(s) updated: cyclic
Meaning	This byte contains the channel number (FC9: ChanNo).

DB19 DBX42.0	**FC9: Start (measuring in Jog mode)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Start active
Signal state 0 or edge change 1 → 0	Start inactive

2.1.6 Signals to channel (DB21, ...)

DB21, ... DBX6.2	**Delete distance-to-go (channel-specific)**
Edge evaluation: no	Signal(s) updated: cyclic
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 not removed. The next program block is then started. IS "Delete distance-to-go (channel-specific)" is therefore ignored by positioning axes. Note: 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	DB21, ... DBX6.2 (delete distance-to-go (axis-specific))

2.1.7 Signals from channel (DB21, ...)

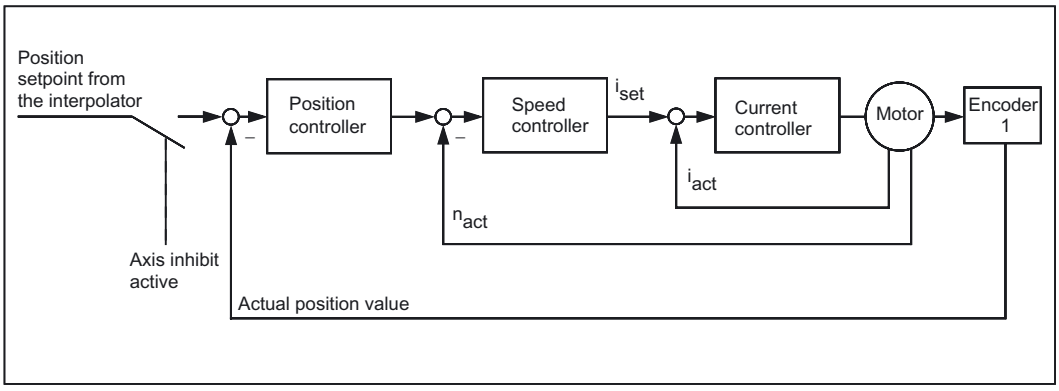
DB21, ... DBX36.6	**Channel-specific NCK alarm is active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	At least one NCK alarm is present for this channel. This means that the group interface signal is also set DB10 DBX103.0 (NCK alarm is present). The PLC user program 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).

DB21, ... DBX36.6	**Channel-specific NCK alarm is active**
Signal state 0 or edge change 1 → 0	No NCK alarm is active for this channel.
Corresponding to	DB21, ... DBX36.6 (NCK alarm with processing stop present) DB10 DBX103.0 (NCK alarm present)
Additional references	/DA/ Diagnostics Guide

DB21, ... DBX36.7	**NCK alarm with processing stop present**
Edge evaluation: no	Signal(s) updated: cyclic
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 is present)
Additional references	/DA/ Diagnostics Guide

DB21, ... DBX318.7	**Overstore active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The overstore function is enabled (with the channel-specific PI service "_N_OST_ON"). If the PI service is denied, the signal will not change.
Signal state 0 or edge change 1 → 0	The overstore function is disabled (with the channel-specific PI service "_N_OST_OFF"). If the PI service is denied, the signal will not change.

2.1.8 Signals to axis/spindle (DB31, ...)

DB31, ... DBX1.3	**Axis/spindle disable**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1 - Axis disable - Spindle disable	(Test status) If the interface signal "Axis disable" is output - for this axis - no more position partial setpoints are output to the position controller; the axis travel is therefore disabled. The position control loop remains closed and the remaining following error is reduced to zero. If an axis is moved with axis disable the actual value position display shows the setpoint position and the actual velocity value display shows the setpoint velocity even though the machine axis is not actually moving. With a RESET the position actual value display is set to the real actual value of the machine. Travel commands continue to be output to the PLC for this axis. If the interface signal is canceled again the associated axis can again traverse normally. If the interface signal "Axis disable" is set for a traversing axis, the axis is stopped with a ramp stop. If the interface signal "Spindle disable" is set, as for axis disable, for this spindle no more speed setpoints are output to the speed controller in the open-loop control mode and no more position partial setpoints are output to the position controller in positioning mode. The movement of the spindle is thus disabled. The speed actual value display displays the speed setpoint value. Spindle disable can only be canceled per "Reset" or with `M2` followed by a program restart. If interface signal "Spindle disable" is set while a spindle is turning, the spindle is stopped according to its acceleration characteristic.
Signal state 0 or edge change 1 → 0	(Normal conditions). The position setpoint values are transferred to the position controller cyclically. The speed setpoint values are transferred to the speed controller cyclically. Cancellation of the "Axis/spindle disable" (edge change 1 → 0) does not take effect until the axis/spindle is stationary (i.e. an interpolation setpoint is no longer present).
	

DB31, ... DBX1.3	**Axis/spindle disable**
Application example(s)	The interface signal "Axis disable" and "Spindle disable" is used when running-in and testing a new NC part program. In so doing, the machine axes and spindles should not execute any traversing or rotational movement.
Special cases, errors, ...	If, for an axis/spindle "axis/spindle disable" is present, then the interface signals: DB31, ... DBX2.1 (controller enable), DB2 ... (feed/spindle stop) and where relevant DB31, ... DBX12.0-12.1 (hardware limit switch) are not effective regarding braking the axis/spindle. The axis/spindle can however be brought into the "follow up" or "hold" state (refer to DB31, ... DBX1.4 (follow-up mode)). Notes: This signal inhibits setpoint output to the drive. 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. Resynchronization takes place automatically on the next traversing command for this axis, i.e. the axis traverses the remaining distance-to-go. Example: G0 X0 Y0 G1 F1000 X100 ; For X20 the signal "axis inhibit" briefly comes, ; X–axis remains stationary, the NC continues to interpolate Y100 X200 ; X travels from approx. 20 to 200 ... Behavior in conjunction with synchronous operation, refer to: **References:** /FB2/ Description of Functions Expansion Functions; Synchronous Spindles (S3) This signal is no longer effective when the coupling for FS/FA is activated. → No. 6 If the signal for the LS/LA is set, it also applies to the FS/FA(s) → No. 7 A workpiece clamped between two spindles (workpiece transfer from front to rear-side machining) cannot be destroyed.

DB31, ... DBX1.3	**Axis/spindle disable**

No.	set: 1	not set: 0	Coupling	Procedure
	LS/LA	FS/FA		
1	0	0	Off	Setpoints of axes are output
2	0	1	Off	No setpoint output for FS/FA
3	1	0	Off	No setpoint output for LS/LA
4	1	1	Off	No setpoint output for LS/LA and FS/FA
5	0	0	ON	Setpoints of axes are output
6	0	1	ON	Disable not effective for FS/FA
7	1	0	ON	Disable also effective for FS/FA
8	1	1	ON	No setpoint output for LS/LA and FS/FA

Corresponding to	DB21, ... DBX33.7 (program test active)

DB31, ... DBX1.4	**Follow-up mode**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1	Follow-up mode is selected for the axis/spindle by the PLC. The means that the position setpoint continually tracks the actual value if the controller enable for the drive is withdrawn. As soon as the follow-up mode is effective, the interface signal: DB31, ... DBX61.3 (follow-up mode active) is set to a 1 signal. The actual value continues to be acquired and updated. If the axis/spindle is moved from its current position by an external effect the zero speed and clamping monitoring do not issue an alarm. 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.

DB31, ... DBX1.4	**Follow-up mode**
Signal state 0	Follow-up mode is not selected (hold). 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 position actual 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 movements start from the setpoint position valid before "controller enable" was removed. When the position control is switched in again the axis may make a speed setpoint jump. Zero speed monitoring or clamping monitoring is still active. In order to disable (switch-out) the zero speed monitoring, when clamping an axis, the interface signal: DB31, ... DBX2.3 (clamping operation running) should be set. In the "holding state", the interface signal: DB31, ... DBX61.3 (follow-up mode active) is set to a 0 signal.
Special cases, errors, ...	If the drive controller enable is withdrawn inside the control due to faults, then the following should be carefully observed: Before an NC start, after the queued alarms have been successfully deleted (i.e. inside the control, the controller enable is re-issued), then "holding" should be activated: DB31, ... DBX1.4 (follow-up mode) = 0 Otherwise, for an NC start and selected follow-up mode, the traversing distance of the previous NC block would not be executed due to the internal delete distance to go. Notice: When changing over from the "follow-up" state to the "hold" state and in the control mode (a controller enable is issued), a delete distance-to-go command is activated in the control. As a consequence, for example, an NC block - in which only this axis is traversed - is ended directly.
Corresponding to	DB31, ... DBX2.1 (controller enable) DB31, ... DBX2.3 (clamping in progress) DB31, ... DBX61.3 (follow-up mode active)

DB31, ... DBX1.5 - DBX1.6	**Position measuring system 1 (PMS1)/Position measuring system 2 (PMS2)**
Edge evaluation: no	Signal(s) updated: cyclic
PMS1: Signal state 1 PMS2: Signal state 0	Position measuring system 1 is used for the axis/spindle (e.g. for position control, absolute value calculation, display). If a position measuring system 2 also exists (MD30200 $MA_NUM_ENCS = 2), this actual value is also acquired.
PMS1: Signal state 0 PMS2: Signal state 1	Position measuring system 2 is used for the axis/spindle (e.g. for position control, absolute value calculation, display). If a position measuring system 1 also exists, this actual value is also acquired.
PMS1: Signal state 1 PMS2: Signal state 1	As it is not possible to use both position measuring systems simultaneously for the position control of an axis/spindle, the control automatically selects position measuring system 1. If a position measuring system 2 also exists, this actual value is also acquired.
PMS1: Signal state 0 PMS2: Signal state 0	1. The axis is in the park position. This means that the following features are valid: – Both position measuring systems are inactive. – There is no actual value acquisition. – The position measuring system monitoring functions are disabled (among others, cable connection of a measured value encoder). The reference point has no effect: DB31, ... DBX60.4/5 (referenced/synchronized) has the signal condition 0 As soon as an axis is parked, the interface signals: DB31, ... DBX61.5 (position controller active), DB31, ... DBX61.6 (speed controller active) and DB31, ... DBX61.7 (current controller active) are set to a 0. After parking has been completed the axis must be re-referenced (reference point approach). If interface signals PMS1 and PMS2 are set to 0 while the axis is moving, the axis is stopped with a ramp stop without the controller enable being internally withdrawn. This is appropriate for the following situations: – Spindle encoder no longer outputs a signal above a certain speed (no longer supplies any pulses). – Spindle encoder is decoupled mechanically because it would not be able to cope with the speed. – The spindle can then continue to run in speed-controlled mode. In order to really bring the axis/spindle to a stop, the controller enable must always be removed additionally by the PLC. 2. The spindle does not have a position measuring system and is only speed controlled. In this case, the interface signal: DB31, ... DBX2.1 (controller enable) should be set to a 1 signal.

DB31, ... DBX1.5 - DBX1.6	**Position measuring system 1 (PMS1)/Position measuring system 2 (PMS2)**
Application example(s)	1. Changing over from position measuring system 1 to positioning measuring system 2 (and vice versa). If the axis in both position measuring systems was referenced and the limit frequency of the measured value encoder has not been exceeded in the meantime i.e. DB31, ... DBX60.4 and 60.5 (referenced/synchronized 1 and 2) has signal state 1, then after the changeover, a reference point approach does not have to be repeated. On switchover, the current difference between position measuring system 1 and 2 is traversed immediately. With the machine data: MD36500 $MA_ENC_CHANGE_TOL (maximum tolerance on position actual value changeover) a tolerance bandwidth can be entered - the deviation between the two actual values may lie within this bandwidth when changing over. If the actual value difference is greater than the tolerance, a switchover between the two systems does not take place and alarm 25100 "Measuring system switchover" not possible is triggered. 2. Parking axis (i.e. no PMS is active): If the encoder has to be removed - e.g. if a rotary table has to be removed from the machine - the position measuring system monitoring is switched off in the parking position. The mounted axis/spindle encoder turns so quickly in certain applications that it can no longer maintain its electrical characteristics (edge rate-of-rise, etc.). 3. Switch-off measuring system: When 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 PMS must be separately referenced.
Special cases, errors, ...	If the state "parking axis" is active, the interface signal: DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2) is ignored at NC start for this axis.
Corresponding to	DB31, ... DBX60.4 (referenced/synchronized 1) DB31, ... DBX60.5 (referenced/synchronized 2) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX2.1 (controller enable) MD36500 $MA_ENC_CHANGE_TOL (max. tolerance on position actual value switchover) MD30200 $MA_NUM_ENCS (number of encoders)
Additional references	/FB1/ Description of Functions, Basic machine; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)

DB31, ... DBX2.1	**Controller enable**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1	The position control loop of the axis/spindle is closed; the axis/spindle is in closed-loop control.
Signal state 1 or edge change 0 → 1	When "controller enable" is set by the PLC user program: • Position control loop of axis is closed. • Position actual value is no longer switched to the position setpoint. • The controller enable of the drive is output. • The interface signal: DB31, ... DBX61.5 (position controller active) is set to 1. When "controller enable" has been signaled no new actual value synchronization of the axis (reference point approach) of the axis is necessary if the maximum permissible limit frequency of the axis measuring system has not been exceeded during follow-up mode. As a function of the interface signal: DB31, ... DBX1.4 (follow-up mode) it is possible to select whether or not the axis first traverses back to the earlier setpoint position (i.e. the positional deviation caused by the clamping process is moved through to eliminate the deviation).
Edge change 1 → 0 or signal state 0	"Controller enable" will be/is removed: The procedure for removing "controller enable" depends on whether the axis/spindle or an axis of the geometry grouping is stationary or traversing at this point in time. • Axis/spindle stationary: – Position control loop of axis is opened. – When IS "Follow-up mode" = 1 the position actual value is switched to the position setpoint (i.e. the setpoint position is corrected to the actual value position). The position actual value of the axis/spindle continues to be acquired by the control. – The controller enable of the drive is removed. – The interface signals: DB31, ... DBX61.5 (position controller active) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX61.7 (current controller active) are set to a 0 signal. • Axis/spindle traverses: – The axis is stopped with rapid stop. – Alarm 21612 "VDI signal controller enable reset during movement" is triggered. – The position control loop of the axis/spindle is opened. – Independent of the interface signal: DB31, ... DBX1.4 (follow-up mode) At the end of braking the position actual value is switched to the position setpoint (i.e. the setpoint position is corrected to track the actual value position). The position actual value of the axis/spindle continues to be acquired by the control. IS "Follow-up mode" is set. – The interface signals: DB31, ... DBX61.5 (position controller active) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX61.7 (current controller active) are set to 0. The axis status cannot be changed again until after RESET.

DB31, ... DBX2.1	**Controller enable**
Application example(s)	Using controller enable when clamping the axis: The axis is positioned to the clamping position. As soon as it has stopped it is clamped and then controller enable is removed. Controller enable is removed because the axis could be mechanically pressed out of position slightly by clamping and the position controller would continuously have to work against the clamping. When clamping is to be stopped, a controller enable signal is first set again and then the axis is freed from clamping.
Special cases, errors, ...	If an attempt is made to traverse the axis/spindle without controller enable, the axis/spindle remains stationary but sends a traversing/move command to the PLC (axis only). The traversing/move command is kept and is executed when the controller enable is re-activated. If the controller enable of a traversing geometry axis is removed the programmed contour cannot be maintained. Controller enable is automatically cancelled by the control when certain faults occur at the machine, the position measuring system or the control.
Corresponding to	DB31, ... DBX61.3 (follow-up mode 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 (switch-off delay 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: cyclic
Signal state 1 or edge change 0 → 1	Delete distance-to-go (axis-specific): Depending on the operating mode, the following occurs for IS "delete distance-to-go axial": • In the 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 removed. • For AUTOMATIC and MDA: 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. Note: 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
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 (axial)" 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 not reached in the following block.
Corresponding to	DB21, ... DBX6.2 (delete distance-to-go, channel-specific)
Additional references	/FB1/ Description of Functions, Basic Machine; Spindles (S1)

DB31, ... DBX9.0 - DBX9.2	**Controller parameter set switchover (request) requested parameter set**
Edge evaluation: no	Signal(s) updated: On request
Signal state 1 or edge change 0 → 1	-
Signal state 0 or edge change 1 → 0	-
Signal irrelevant for ...	MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0
Application example(s)	The binary-coded index of the parameter set to be activated is located in the 3 bits: • 0 corresponds to the 1st parameter set • 1 corresponds to the 2nd parameter set • Etc. (max. 6 parameter sets are possible)
Special cases, errors, ...	Indices 6-7 are mapped onto parameter set 6.
Corresponding to	DB31, ...DBX69.0-69.2

DB 31, ... DBX9.3	**Disable parameter set switchover commands from NC Locks**
Signal state 1 or edge change 0 → 1	NC should not initiate any parameter set switchovers.
Signal state 0 or edge change 1 → 0	Parameter set switchover by the NC is enabled.
Corresponding to	DB31, ... DBX9.0 - DBX9.2

DB31, ... DBX20.0	**Ramp-up times**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	V/f operation is activated using machine data: MD1014 $MD_U/F_MODE_ENABLE. The time entered in the machine data: MD1126 $MD_U/F_MODE_RAMP_TIME_2 is effective.
Signal state 0 or edge change 1 → 0	V/f operation operation is activated using machine data: MD1014 $MD_U/F_MODE_ENABLE. The time entered in the machine data: MD1125 $MD_U/F_MODE_RAMP_TIME_1 is effective.
Signal irrelevant for ...	SINUMERIK 840Di
Corresponding to	MD1014 $MD_U/F_MODE_ENABLE (activate V/f operation) MD1125 $MD_UF_MODE_RAMP_TIME_1 (ramp-up time 1 for V/f mode) MD1126 $MD_UF_MODE_RAMP_TIME_2 (ramp-up time 2 for V/f mode)
Additional references	/FBA/ SIMODRIVE 611D Description of Functions, Drives

DB31, ... DBX20.1	**Ramp-function generator fast stop**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	A fast stop is triggered by the PLC for the drive (611D). Speed setpoint 0 is therefore entered. The drive is stopped without a ramp function (regenerative braking). As soon as the fast stop is recognized by the drive module, the interface signal: DB31, ... DBX92.1 (Ramp-function generator fast stop active) is returned to the PLC.

DB31, ... DBX20.1	**Ramp-function generator fast stop**
Signal state 0 or edge change 1 → 0	No rapid stop is requested by the PLC for the drive.
Signal irrelevant for ...	SINUMERIK 840Di
Corresponding to	DB31, ... DBX92.1 (ramp-function generator fast stop active)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX20.2	**Torque limit 2**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Torque limit 2 is requested by the PLC for the axis/spindle. For 611D two torque limit values can be set for each axis/spindle, whereby torque limit 2 refers to torque limit 1 (reduction factor). Torque limit 2 is selected via the interface. The limit value is defined using the drive parameters. As soon as torque limit 2 is active for the drive, the drive signals this using the interface signal: DB31, ... DBX92.2 (torque limit 2 active)
Signal state 0 or edge change 1 → 0	Only torque limit 1 has been selected by the PLC.
Signal irrelevant for ...	SINUMERIKI 840Di
Application example(s)	In order to reduce the stress on the mechanics and workpiece, the actual torque limit can be reduced using torque limit 2 for certain machining operations.
Corresponding to	DB31, ... DBX92.2 (torque limit 2 active)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX20.3	**Speed setpoint smoothing**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The PLC requests a filter to smooth the speed setpoint value. In the drive module the interface signal is only effective under the following conditions: • Speed setpoint filter 1 is active in the drive • Speed setpoint filter 1 has been configured as a low-pass filter (i.e. not as a band-stop filter) As soon as these conditions exist whereby the smoothing of the speed setpoint is active, the 611D or 611U signals this to the PLC using the interface signal: DB31, ... DBX92.3 (speed setpoint smoothing active).

DB31, ... DBX20.3	**Speed setpoint smoothing**
Signal state 0 or edge change 1 → 0	No smoothing of the speed setpoint value is requested by the PLC.
Application example(s)	With this interface signal, speed setpoint smoothing can - for example - be activated from the PLC user program for a spindle during speed control to achieve smooth torque output. Speed setpoint smoothing can therefore be deactivated when the spindle is in positioning mode.
Special cases, errors, ...	
Corresponding to	DB31, ... DBX92.3 (speed setpoint smoothing active)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX21.0 - DBX21.2	**Drive parameter set selection A, B, C**
Edge evaluation: no	Signal(s) updated: cyclic
Meaning	With bit combinations A, B and C it is possible to select 8 different drive parameter sets for the digital drives SIMODRIVE 611D/611U/SINUMERIK 810D. The following assignment applies:

Drive 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

The switchable drive parameters are as follows:

- Current setpoint filters (low-pass, band-stop); for adaptation to the mechanics
- Motor speed normalization
- Speed controller parameters
- Speed setpoint filter
- Speed monitoring data

As soon as the new drive parameter becomes effective, the drive signals this to the PLC using the interface signals:
DB31, ... DBX93.0, 1 and 2 (active parameter set).

DB31, ... DBX21.0 - DBX21.2	**Drive parameter set selection A, B, C**
Application example(s)	Drive parameter switchover can be used for the following: • Changing the gear stage • Changing-over 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 when in the zero speed state (especially when the axis is stationary).
Corresponding to	DB31, ... DBX93.0, 1 and 2 (active drive parameter set)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D or /IAC/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX21.3 - DBX21.4	**Motor selection A, B**
Edge evaluation: no	Signal(s) updated: cyclic
Meaning	The PLC can switch between 4 different motors or motor mode types using the motor selection function. The following assignment applies:

Motor selection	Application	B	A
Motor 1	Operating mode 1	0	0
Motor 2	Operating mode 2	0	1
Motor 3	res. up to SW6.3, then for	1	0
Motor 4	611D Performance 2 or 611U can be used for operating mode 3 or 4	1	1

As soon as a new motor selection is detected, the drive cancels the pulse enable (feedback signal using the interface signal DB31, ... DBX93.3 and 4 (active motor)).

Using the motor selection, it is possible, for example, to choose mode 1 as star-connected operation and mode 2 as delta-connection operation for the main spindle drive (MSD).

The drive signals the currently selected motor back to the PLC using the interface signals: DB31, ... DBX93.3 and 4 (active motor).

Signal irrelevant for ...	SINUMERIK 810D
Application example(s)	Timing for star-delta switchover
Special cases, errors, ...	**Notice:** Before a new motor is selected, the interface signal: DB31, ... DBX21.5 (motor selected) must be set to 0!

DB31, ... DBX21.3 - DBX21.4	**Motor selection A, B**
Corresponding to	DB31, ... DBX93.3 and 4 (active motor) DB31, ... DBX21.5 (motor selected)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX21.5	**Motor selection in progress**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The PLC outputs the interface signal: DB31, ... DBX21.5 (motor selected) to the 611D to confirm that the external contactor has been switched over to the new motor (e.g. that motor contactor 2 has been energized as part of the star/delta switchover process). The pulses are then enabled by the drive.
Signal state 0 or edge change 1 → 0	The interface signal: DB31, ... DBX21.5 (motor selected) must be reset to 0 by the PLC user program before a new motor is selected! Otherwise the pulses from the drive might be enabled too early.
Signal irrelevant for ...	SINUMERIK 810D
Corresponding to	DB31, ... DBX93.3 and 4 (active motor) DB31, ... DBX21.3 and 4 (motor selection A, B)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX21.6	**Speed controller integrator disable**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The interface signal is used by the 611D/611U to disable the integrator of the speed controller. The speed controller is thus changed over from a PI to a P controller. **Note:** If the speed controller integrator disable is activated, equalization processed might take place in certain applications (e.g. if the integrator was already holding a load while stationary). The 611D/611U acknowledges the integrator disable to the PLC using the interface signal: DB31, ... DBX93.6 (speed controller integrator disabled).
Signal state 0 or edge change 1 → 0	The integrator of the speed controller is enabled.
Signal irrelevant for ...	SINUMERIK 810D
Corresponding to	DB31, ... DBX93.6 (integrator n-controller disabled)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX21.7	**Pulse enable**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Pulse enable is signaled by the PLC for this drive (axis/spindle). The pulses for the drive modules are only enabled if all enable signals (hardware and software) are available (see figure for DB31, ... DBX93.5). In this case, the interface signal: DB31, ... DBX93.7 (pulses enabled) is signaled to the PLC with a signal. For additional information refer to DB31, ... DBX93.7 and references.
Signal state 0 or edge change 1 → 0	The pulses are disabled by the PLC for this drive. If pulse enable is canceled for a moving axis/spindle the axis/spindle is not longer braked in a controlled fashion. The axis coasts to rest.
Signal irrelevant for ...	SINUMERIK FM-NC
Application example(s)	Signal relevant to safety
Special cases, errors, ...	If pulse enable is canceled for a moving axis/spindle as a result of an EMERGENCY STOP, the axis/spindle only coasts to rest.
Corresponding to	DB31, ... DBX93.7 (pulses enabled)
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D or /IAC/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX1.0	**Drive test travel enable**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Safety handshake when the function generator is started by the NC. If an axis is to be traversed without further operator intervention, using: DB31, ... DBX61.0 (drive test travel request) = 1 signal, the NC requests a travel enable signal (permissive signal) from the PLC control. If all axis travel conditions are fulfilled, PLC acknowledges this with: DB31, ... DBX1.0 (drive test travel enable) = 1 signal The PLC always has the master authority in deciding whether an axis can be traversed.
Signal state 0 or edge change 1 → 0	The NC does not request permission to move axis from the PLC with: DB31, ... DBX1.0 (drive test travel enable) = 0 signal
Additional references	/IAD/ SINUMERIK 840D Startup Guide; Chapter: SIMODRIVE 611D or /IAC/ SINUMERIK 810D Installation and Start-up Guide

2.1.9 Signals from axis/spindle (DB31, ...)

DB31, ... DBX61.0	**Drive test travel request**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The control signals that all of the traversing conditions for the drives are fulfilled. Prerequisites 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 or edge change 1 → 0	The control signals that the axes cannot be moved. Axes cannot be moved for: • DB31, ... DBX61.0 (drive test, travel request) = 0 signal • In the control when faults are present This means that the prerequisites specified above are not fulfilled.
Further references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAC/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX61.3	**Follow-up mode active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The control signals that the follow-up mode for the axis/spindle is not active. Prerequisites 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 or edge change 1 → 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 above specified prerequisites are not fulfilled. In the "hold" state, the interface signal: DB31, ... DBX61.3 (follow-up mode 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).

DB31, ... DBX61.3	**Follow-up mode active**
Corresponding to	DB31, ... DBX2.1 (controller enable) DB31, ... DBX1.4 (follow-up mode!)
Additional references	/DA/ Diagnostics Guide

DB31, ... DBX61.4	**Axis/spindle stationary**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The actual velocity of the axis or the actual speed of the spindle is below the limit defined in machine data: MD36060 $MA_STANDSTILL_VELO_TOL (maximum velocity/speed for signal "Axis/Spindle stationary").
Signal state 0 or edge change 1 → 0	The actual velocity of the axis or the actual spindle speed is greater than the value specified in the MD (standstill/zero speed 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.
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-in 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**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The control signals that the position controller for the axis or spindle is closed.
Signal state 0 or edge change 1 → 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.

DB31, ... DBX61.5	**Position controller active**
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	/DA/ Diagnostics Guide

DB31, ... DBX61.6	**Speed controller active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The control signals that the speed controller is closed for the axis or spindle.
Signal state 0 or edge change 1 → 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, ...	Special case for simulation axes (MD30350 = "1"): The interface signal: DB31, ... DBX61.6 (speed controller active) is also set for simulation axes as soon as: MD30350 $MA_SIMU_AX_VDI_OUTPUT (output of the axis signals for simulation axes) = "1".
Corresponding to	DB31, ... DBX61.5 (position controller active)

DB31, ... DBX61.7	**Current controller active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The control signals that the current controller is closed for the axis or spindle.
Signal state 0 or edge change 1 → 0	The control signals that the current controller for the axis or spindle is open. The current controller output (including the feedforward quantities on the manipulated variable for the voltage) is cleared.
Corresponding to	DB31, ... DBX61.5 (position controller active) DB31, ... DBX61.6 (speed controller active)

DB31, ... DBX69.0 - DBX69.2	**Controller parameter set switchover (A (checkback signal)) Active parameter set**
Edge evaluation: no	Signal(s) updated: After switchover
Signal state 1 or edge change 0 → 1	-
Signal state 0 or edge change 1 → 0	-
Signal irrelevant for ...	MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0
Application example(s)	The binary-coded index of the activated parameter set is located in the 3 bits: • 0 corresponds to the 1st parameter set • 1 corresponds to the 2nd parameter set • Etc. (max. 6 parameter sets are possible)
Special cases, errors, ...	Index 0 is returned if the switchover function is disabled with: MD35590 $MA_PARASET_CHANGE_ENABLE = 0. In this case, the 1st parameter set is always active.
Corresponding to	DB31, ...DBX9.0 - DBX9.2 (controller parameter set switchover (request))

DB31, ... DBX76.0	**Lubrication pulse**
Edge evaluation: yes	Signal(s) updated: when a change is made
Signal state 1 or edge change 0 → 1	If the axis has traversed a distance further than entered in the machine data: MD33050 $MA_LUBRICATION_DIST (travel distance for lubrication from the PLC) then the interface signal: DB31, ...DBX76.0 (lubrication pulse) is set for one PLC cycle.
Signal state 0 or edge change 1 → 0	IS "Lubrication pulse" is automatically reset to 0 after one PLC cycle.
Application example(s)	The lubrication pump for the axis can be activated with IS "Lubrication pulse". Machine bed lubrication therefore depends on the distance traveled.
Corresponding to	MD33050 $MA_LUBRICATION_DIST (lubrication pulse distance)

DB31, ... DBX92.0	**Setting-up mode active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Setting-up mode is active for the drive (611D). The setting-up mode is selected using the terminals on the infeed/regenerative feedback module. The setting-up mode is required for optimizing the machining process. The following adaptations and additional functions are possible: • Drive: – Reducing the speed setpoint limits – Reducing the current setpoint limits • I/RF: – Disabling the closed-loop DC link voltage control
Signal state 0 or edge change 1 → 0	Normal operation is active for the drive. The following thus applies: • The maximum limit values for speed and current setpoint are active • DC link voltage control is active
Signal irrelevant for ...	SINUMERIK 840Di, 810D
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX92.1	**Ramp-function generator fast stop active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	PLC receives the signal that the ramp function generator fast stop is active. The function has been activated by the interface signal: DB31, ... DBX20.1 (ramp-function generator fast stop) The drive is stopped without a ramp function with speed setpoint = 0 and without pulse suppression.

DB31, ... DBX92.1	**Ramp-function generator fast stop active**
Signal state 0 or edge change 1 → 0	Ramp function generator fast stop is not active for the drive.
Signal irrelevant for ...	SINUMERIK 810D
Application example(s)	Bypassing the ramp function generator on the servo side
Corresponding to	DB31, ... DBX20.1 (ramp-function generator fast stop)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX92.2	**Torque limit 2 active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Drive (611D) sends an acknowledgment to the PLC that torque limit 2 is active in addition to torque limit 1. The particular limit value is defined using the drive parameters.
Signal state 0 or edge change 1 → 0	Only torque limit 1 is active.
Signal irrelevant for ...	SINUMERIK 840Di, 810D
Corresponding to	DB31, ... DBX20.2 (torque limit 2)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX92.3	**Speed setpoint smoothing active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The speed setpoint smoothing requested by the PLC with the interface signal: DB31, ... DBX20.3 (speed setpoint smoothing) is active.
Signal state 0 or edge change 1 → 0	No speed setpoint smoothing is active.
Signal irrelevant for ...	SINUMERIK 810D

DB31, ... DBX92.3	**Speed setpoint smoothing active**
Corresponding to	DB31, ... DBX20.3 (speed setpoint smoothing)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX93.0 - DBX93.2	**Active drive parameter set A, B, C**
Edge evaluation: no	Signal(s) updated: cyclic
Meaning	The drive module (611D/611U) signals back to the PLC which drive parameter set is currently active. With bit combinations A, B and C, 8 different drive parameter sets can be selected for the 611D. The following assignment applies:

Active drive 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

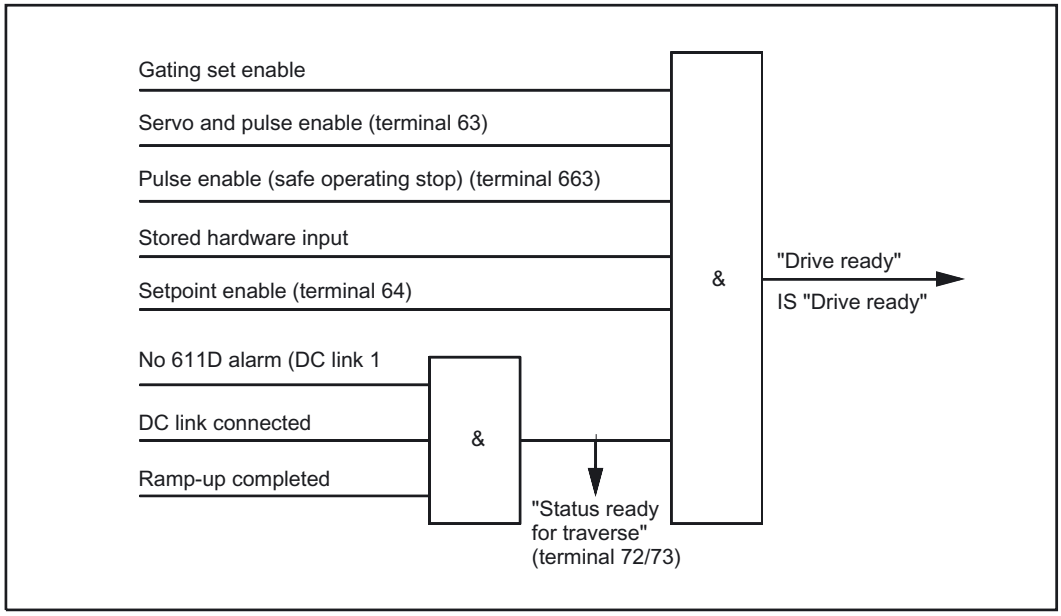
Application example(s)	Drive parameter switchover can be used for the following: • To change the gear stage • To change over the measuring circuit
Corresponding to	DB31, ... DBX21.0 - DBX21.2 (drive parameter set selection)
Additional references	DB31, ... DBX21.0 - DBX21.2 (drive parameter set selection) or /IAD/ SINUMERIK 840D Installation and Start-up Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX93.3 - DBX93.4	**Active motor A, B**
Edge evaluation: no	Signal(s) updated: cyclic
Meaning	The drive module (611D) returns signals to PLC stating which motor selection is currently active. Motor selection can be used with a main spindle drive (MSD) to switch between star and delta operation and so reduce the starting current. The following assignment applies:

Active motor	Application	B	A
Motor 1	MSD: Star mode	0	0
Motor 2	MSD: Delta mode	0	1
Motor 3	Reserved	1	0
Motor 4	Reserved	1	1

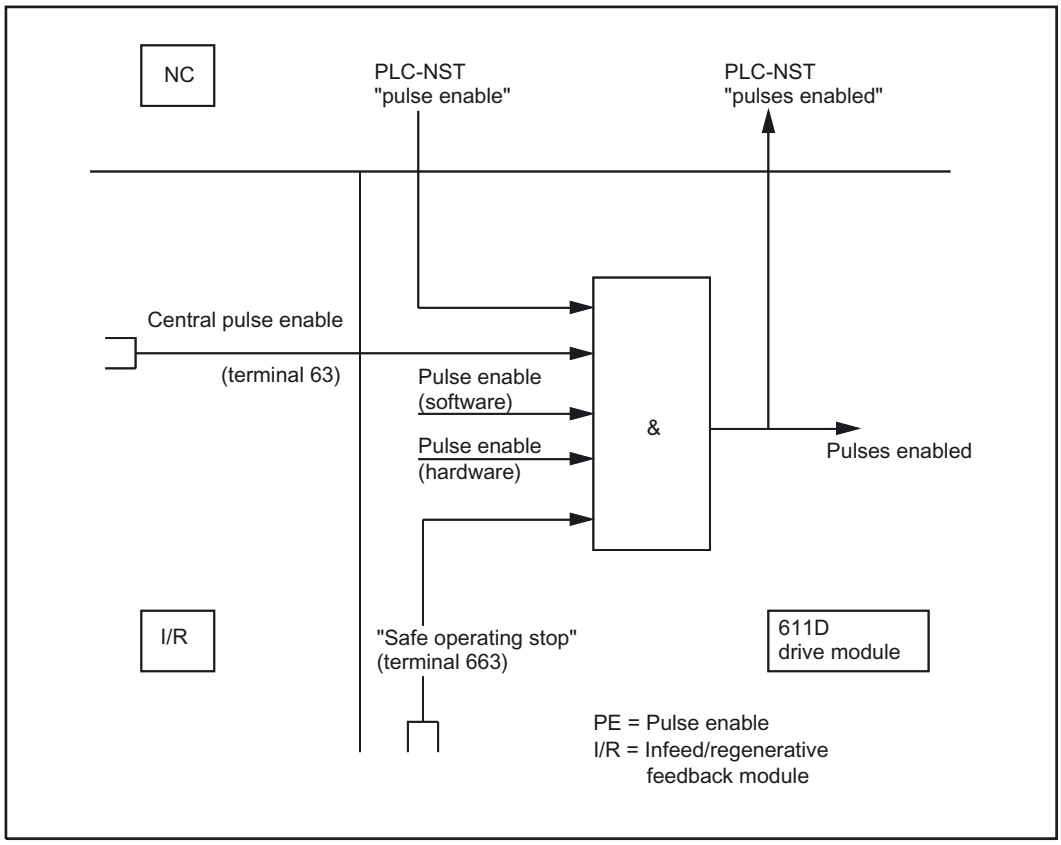
Signal irrelevant for ...	SINUMERIK 810D
Corresponding to	DB31, ... DBX21.3 and DBX21.4 (motor selection) DB31, ... DBX21.5 (motor selected)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX93.5	**Drive Ready**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Feedback signal from the drive to the PLC that the drive is ready.
Signal state 0 or edge change 1 → 0	The drive is not ready. The drive might be disabled for the following reasons (refer to Fig.): • Enable terminals not energized (e.g. terminal 63 "Controller and pulse enable"; terminal 663 "Safe operating stop", terminal 64 "Setpoint enable") • Drive alarm active (e.g. motor temperature has reached switch-off threshold) • DC link voltage is too low • Drive has not yet reached cyclic operation • Hardware fault • No position measuring system is active ("parking axis" state) • I/R is not switched on As soon as the drive is ready for operation it is stopped (depending on the fault state either with pulse disable or fast stop) or pulse disable is maintained during ramp up. The interface signals: DB10 DBX108.6 (611D ready) DB31, ... DBX61.7 (current controller active) DB31, ... DBX61.6 (speed controller active) are also withdrawn. IS "611D Ready" is not available for the 840Di when used in conjunction with the drive 611U.

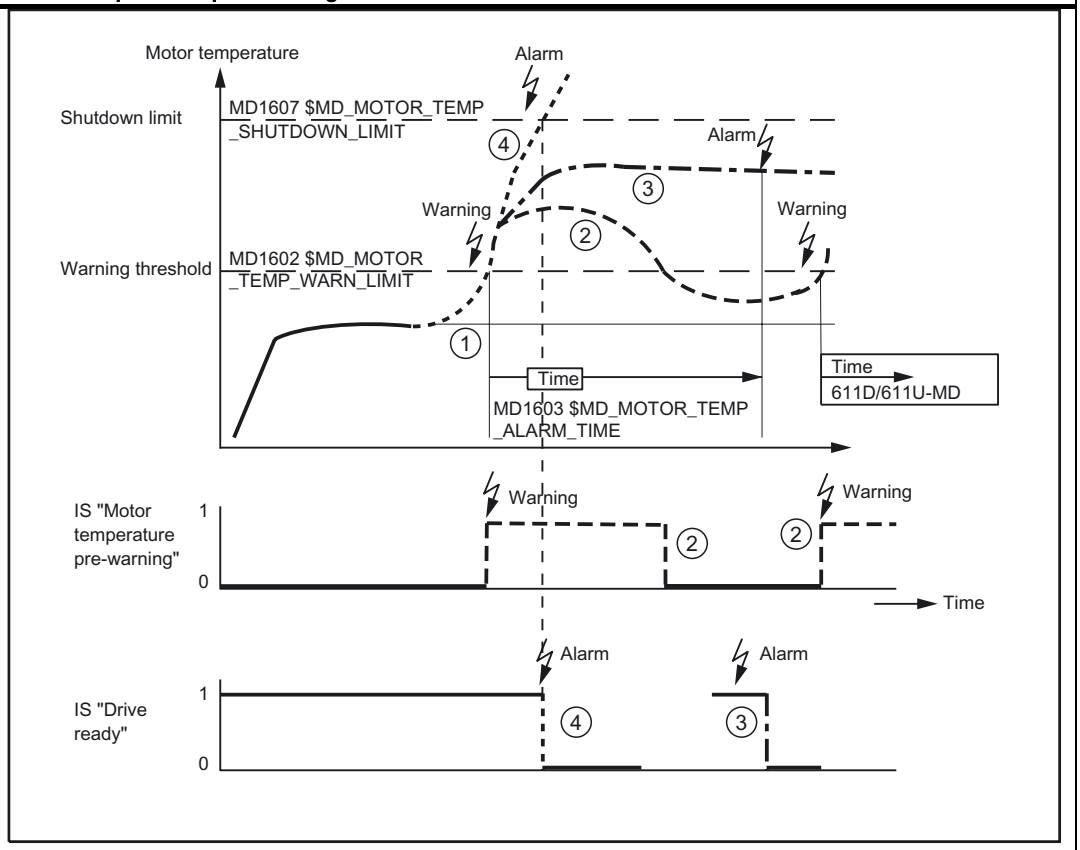
DB31, ... DBX93.5	**Drive Ready**
	
Corresponding to	DB10 DBX108.6 (611D-Ready) DB31, ... DBX61.7 (current controller active) DB31, ... DBX61.6 (speed controller active)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX93.6	**Speed controller integrator disabled**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The request from the PLC to disable the integrator of the speed controller using interface signal: DB31, ... DBX21.6 (integrator inhibit, speed controller) is active for the drive module. The speed controller has therefore switched from a PI to a P controller.
Signal state 0 or edge change 1 → 0	The integrator of the speed controller is enabled. The speed controller functions as a PI controller.
Signal irrelevant for ...	SINUMERIK 810D
Corresponding to	DB31, ... DBX21.6 (integrator disable, n-controller)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D

DB31, ... DBX93.7	**Pulses enabled**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The pulse enable for the drive module is available. The axis/spindle can now be traversed.
Signal state 0 or edge change 1 → 0	The drive module pulses are suppressed. Therefore, the axis/spindle cannot be traversed. The pulses are suppressed as soon as there is no enable signal (refer to Fig.). Also, if the "controller enable of drive" is withdrawn, the drive is stopped with setpoint 0 (regenerative braking). A timer is started in the drive module and after the configured time has expired: MD1404 $MD_PULSE_SUPPRESSION_DELAY (timer stage, pulse suppression), the pulses are inhibited. If, within this time, the actual speed reaches the shutdown speed: MD1403 $MD_PULSE_SUPPRESSION_SPEED (pulse suppression), then at this time, the pulses are inhibited. If the speed is less than/equal to the speed threshold (MD1403), and if the controller enable of the drive is withdrawn, then the pulses are immediately suppressed. Pulse suppression is also triggered if there is no position measuring system ("parking axis" state). As soon as the pulses are suppressed, then also the interface signals: DB31, ... DBX61.7 (current controller active) and DB31, ... DBX61.6 (speed controller active) are reset.

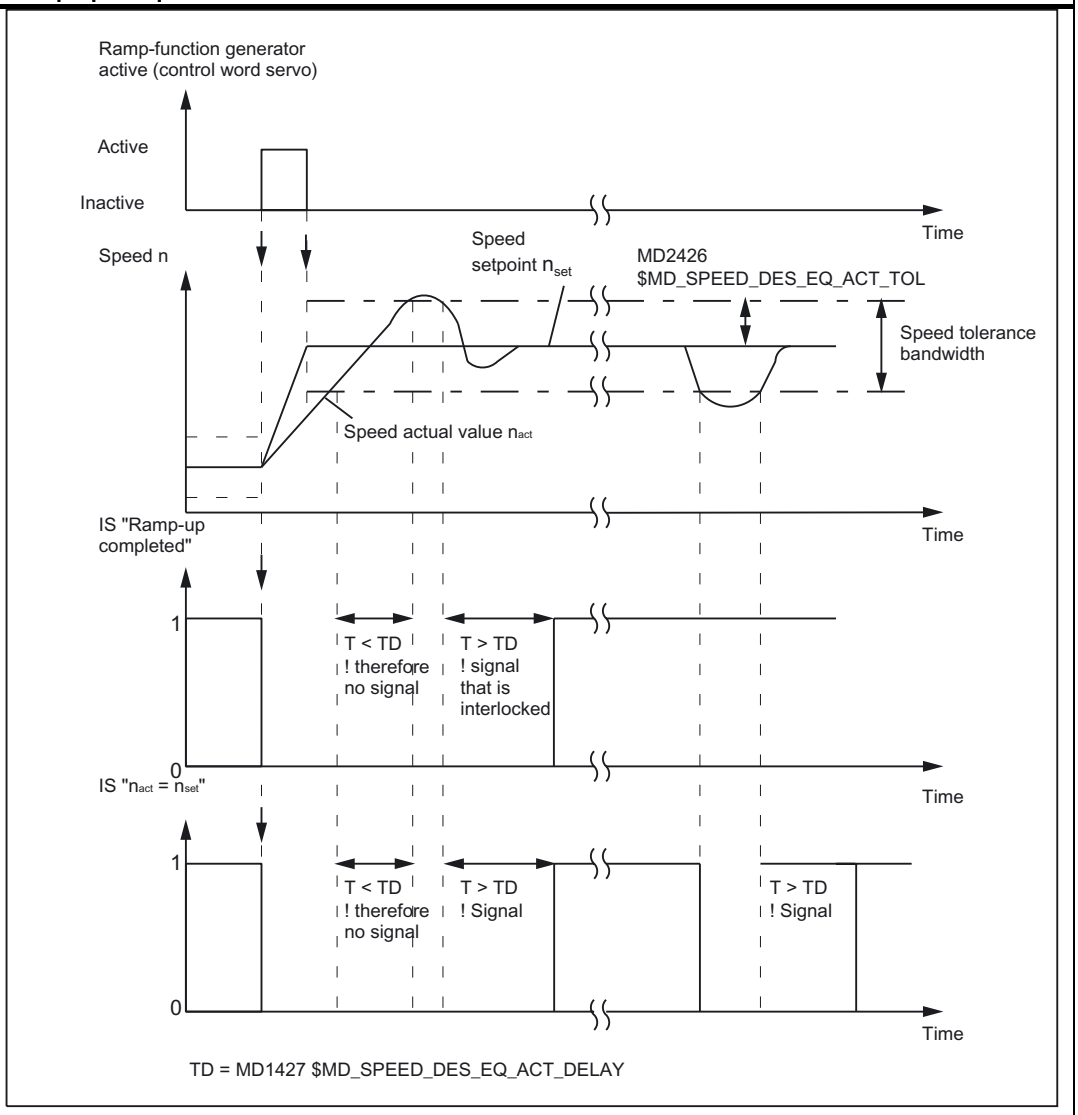
DB31, ... DBX93.7	**Pulses enabled**
	 Pulse enable for 611D drive module
Corresponding to	DB31, ... DBX21.7 (pulse enable) MD1404 $MD_PULSE_SUPRESSION_DELAY MD1403 $MD_PULSE_SUPRESSION_SPEED
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX94.0	**Motor temperature pre-warning**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The drive module signals a "Motor temperature pre-warning" to the PLC. In this case, the motor temperature has exceeded the defined warning threshold: MD1602 $MD_MOTOR_TEMP_WARN_LIMIT (maximum motor temperature; standard value 120 °C) (② in the diagram). If the motor temperature remains too high, after a defined time: MD1603 $MD_MOTOR_TEMP_ALARM_TIME (timer state, motor temperature alarm, standard value 240 s) the drive is regeneratively braked and then the pulses are inhibited (③ in the diagram). Alarm 300614 is output and the interface signal: DB10 DBX108.6 (611D ready) is withdrawn. If the motor temperature rises still further and the shutdown threshold defined in: MD1607 $MD_MOTOR_TEMP_SHUTDOWN_LIMIT (motor temperature shutdown limit, default value 155 °C) is reached, the drive is stopped immediately (refer to ④ in the diagram). An alarm is also output and IS "Drive Ready" canceled. However, if the motor temperature drops back down to below the warning threshold, the IS is reset to 0 (refer to ② in the diagram). **Exception:** If no temperature sensor signal is measured, this is interpreted as a fault in the motor PTC thermistor. In this case, the interface signal: DB31, ... DBX94.0 (motor temperature pre-warning) is also set. The procedure continues as described above.
Signal state 0 or edge change 1 → 0	The motor temperature is below the warning threshold. The actual motor temperature is displayed in the axis/spindle service display in the operating area "diagnostics". This display corresponds to the machine data: MD1702 $MD_MOTOR_TEMPERATURE (motor temperature)

DB31, ... DBX94.0	**Motor temperature pre-warning**
	
Application example(s)	As soon as "Motor temperature prewarning" has been signaled, the PLC can, for example, initiate controlled shutdown of the drives.
Corresponding to	DB31, ... DBX93.5 (drive ready) MD1602 $MD_MOTOR_TEMP_WARN_LIMIT MD1603 $MD_MOTOR_TEMP_ALARM_TIME MD1607 $MD_MOTOR_TEMP_SHUTDOWN_LIMIT
Additional references	/DA/ Diagnostics Guide /IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

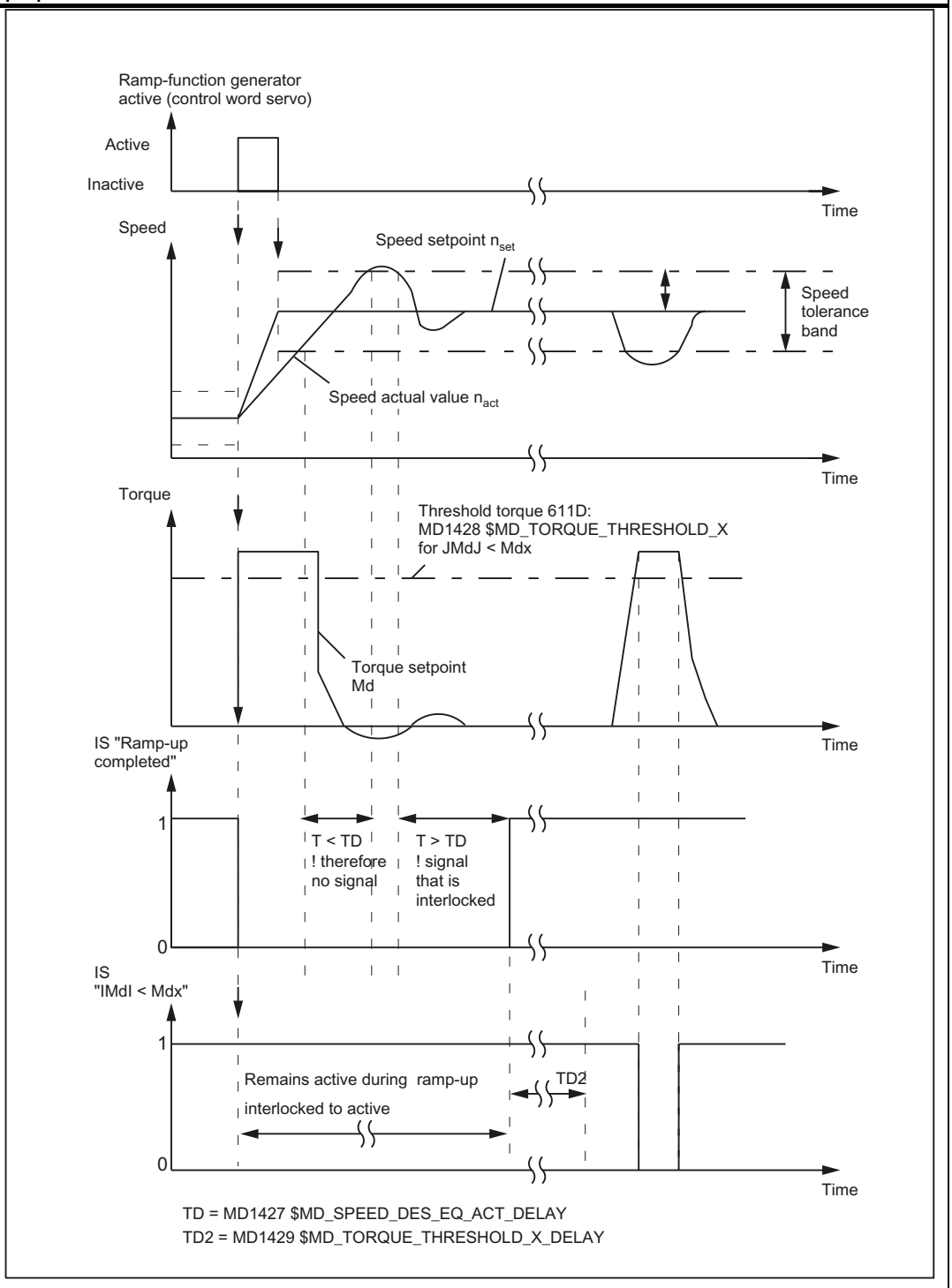
DB31, ... DBX94.1	**Heatsink temperature pre-warning**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The drive module sends the warning "heatsink temperature pre-warning" to the PLC. This triggers the following: • Terminal 5 on the infeed/regenerative feedback module is simultaneously activated. • The drive module is switched off after 20 seconds. The drives are stopped when the impulse enable is removed. Then alarm 300515 is triggered.
Signal state 0 or edge change 1 → 0	The drive module heatsink temperature pre-warning has not responded.
Application example(s)	As soon as "heatsink temperature warning" has been signaled, the PLC can, for example, initiate controlled shutdown of the drives.
Additional references	/DA/ Diagnostics Guide

DB31, ... DBX94.2	**Ramp-up completed**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The PLC is signaled, that after a new speed setpoint has entered, the speed actual value has reached the speed tolerance bandwidth: MD1426 $MD_SPEED_DES_EQ_ACT_TOL (tolerance bandwidth for n_{set} = n_{act} - signal) and has remained within this tolerance bandwidth for at least the time defined using machine data: MD1427 $MD_SPEED_DES_EQ_ACT_DELAY (delay time n_{set} = n_{act} - signal) refer to the diagram). Even if the speed actual value leaves the tolerance band (because of speed fluctuations resulting from changes in load) the "ramp-up completed" signal remains (1 signal).
Signal state 0 or edge change 1 → 0	The conditions described above have not yet been fulfilled. The ramp-up has therefore not yet been completed.

DB31, ... DBX94.2	**Ramp-up completed**
	
Corresponding to	DB31, ... DBX94.6 ("$n_{act} = n_{set}$") DB31, ... DBX94.3 ("$\|M_D\| = M_{dx}$") MD1426 $MD_SPEED_DES_EQ_ACT_TOL MD1427 $MD_SPEED_DES_EQ_ACT_DELAY
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX94.3	**$\|M_d\| < M_{dx}$**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	611D signals to the PLC that the torque setpoint $\|M_d\|$ does not exceed the threshold torque M_{dx} in the steady-state condition (i.e. ramp-up completed) (refer to the diagram). The threshold torque is set using the machine data: MD1428 $MD_TORQUE_THRESHOLD_X (threshold torque) as a % of the actual torque limit value. The torque threshold characteristic is speed-dependent. While ramping-up, the interface signal: DB31, ... DBX94.2 ($|M_d| < M_{dx}$) remains at 1. The signal $\|M_d\| < M_{dx}$ only becomes active after ramp-up has been completed: DB31, ... DBX94.2 (ramp-up completed) = 1 and the signal interlocking time for the threshold torque: MD1429 $MD_TORQUE_THRESHOLD_X_DELAY (delay time $n_d < n_{dx}$ - signal) has expired.
Signal state 0 or edge change 1 → 0	The torque setpoint $\|M_d\|$ is larger than the threshold torque M_{dx}. If necessary, the PLC user program can initiate a response.

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



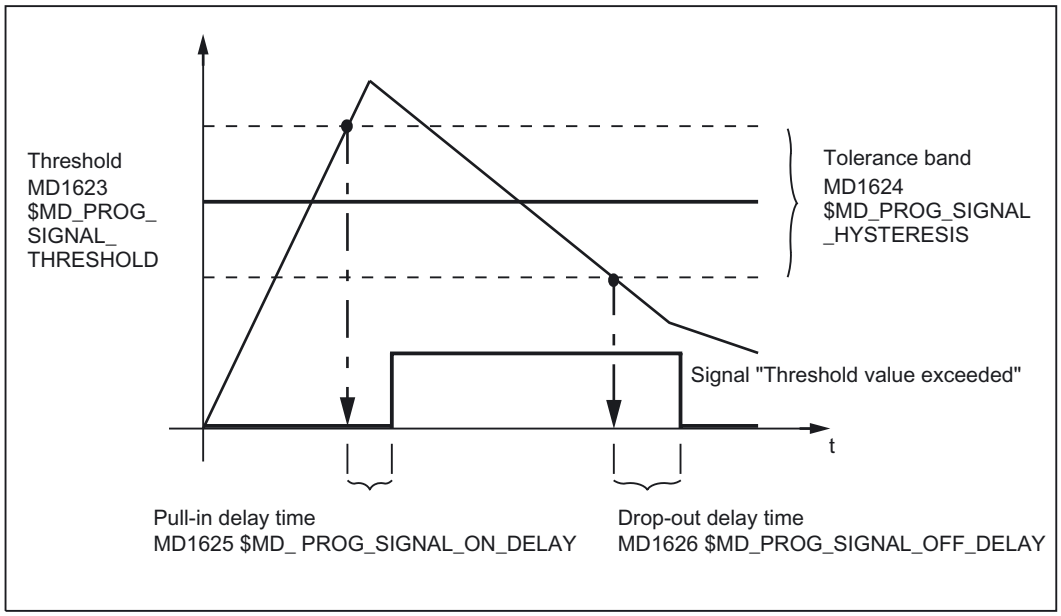
DB31, ... DBX94.3	**$\|M_d\| < M_{dx}$**
Corresponding to	DB31, ... DBX94.2 (ramp-up completed) MD1428 $MD_TORQUE_THRESHOLD_X MD1429 $MD_TORQUE_THRESHOLD_X_DELAY MD1427 $MD_SPEED_DES_EQ_ACT_DELAY
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX94.4	**$\|n_{act}\| < n_{min}$**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The SIMODRIVE 611 D/ 611D signals the PLC that the speed actual value n_{act} is less than the minimum speed (n_{min}). The minimum speed is defined using the machine data: MD1418 $MD_SPEED_THRESHOLD_MIN.
Signal state 0 or edge change 1 → 0	The speed actual value is higher than the minimum speed.
Corresponding to	MD1418 $MD_SPEED_THRESHOLD_MIN (minimum speed (n_{min} for $n_{act} < n_{min}$))
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX93.5	**$\|n_{act}\| < n_x$**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The 611D/ 611U signals the PLC that the speed actual value n_{act} is less than the threshold speed (n_x). The threshold speed is defined using the machine data: MD1417 $MD_SPEED_THRESHOLD_X.
Signal state 0 or edge change 1 → 0	The speed actual value is higher than the threshold speed.
Corresponding to	MD1417 $MD_SPEED_THRESHOLD_X (threshold speed (n_x for $n_{act} < n_x$))
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX94.6	**$n_{act} = n_{set}$**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	SIMODRIVE 611D/611U signals the PLC that after a new speed setpoint has been entered, the speed actual value n_{act} has reached the speed tolerance bandwidth: MD1426 $MD_SPEED_DES_EQ_ACT_TOL (tolerance bandwidth for $n_{set} = n_{act}$ signal) and has remained within this tolerance bandwidth for at least the time defined using machine data: MD1427 $MD_SPEED_DES_EQ_ACT_DELAY (delay time for $n_{set} = n_{act}$ signal) (refer to the diagram). If the speed actual value then leaves the tolerance bandwidth, then contrary to the signal "ramp-up completed", the interface signal: DB31, ... DBX94.6 ($n_{act} = n_{set}$) is set to 0.
Signal state 0 or edge change 1 → 0	The conditions described above have not yet been fulfilled. The speed actual value is outside the speed tolerance bandwidth.
Corresponding to	DB31, ... DBX94.2 (ramp-up completed) MD1426 $MD_SPEED_DES_EQ_ACT_TOL MD1427 $MD_SPEED_DES_EQ_ACT_DELAY
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX94.7	**Variable signaling function**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1	The SIMODRIVE 611D/611U signals the PLC that the threshold value has exceeded the value being monitored. With the variable signaling function it is possible to monitor for each axis whether a defined threshold - that can be entered - is exceeded for any programmable value of SIMODRIVE 611D/611U. A corresponding interface signal is then sent to the PLC. The parameters for the variables being monitored are set in the following machine data: • MD1620 $MD_PROG_SIGNAL_FLAGS (bits variable signal function) • MD1621 $MD_PROG_SIGNAL_NR (signal number variable signal function) • MD1622 $MD_PROG_SIGNAL_ADDRESS (address variable signal function) • MD1623 $MD_PROG_SIGNAL_THRESHOLD (threshold variable signal function) • MD1624 $MD_PROG_SIGNAL_HYSTERESIS (hysteresis variable signal function) • MD1625 $MD_PROG_SIGNAL_ON_DELAY (on delay variable signal function) • MD1626 $MD_PROG_SIGNAL_OFF_DELAY (off delay variable signal function) **Monitoring:** The parameterized variable is monitored to check whether it exceeds a defined threshold. In addition, a tolerance band (hysteresis) can be defined which is considered when scanning for violation of the threshold value. Further, the "threshold exceeded" signal can be also be logically combined with an on delay and off delay time (refer to the diagram). **Selection:** The variable to be monitored can be selected by entering a signal number or by entering a symbolic address. The machine data: MD1620 $MD_PROG_SIGNAL_FLAGS (bits, variable signaling function) can be used to enable and disable the variable signaling function on an axis-for-axis basis. This machine data can also be used to determined whether the threshold value comparison is to be signed or unsigned. For further information see References.
Signal state 0	SIMODRIVE 611D signals the PLC that the threshold value of the variable being monitored has not been exceeded or that the conditions defined in the above 611DMD are not fulfilled. If the variable signaling function is disabled (MD1620), signal state "0" is output to the PLC.

DB31, ... DBX94.7	**Variable signaling function**
	
Application example(s)	With the variable signal function the machine tool manufacturer can monitor one additional threshold value for specific applications for each axis/spindle and evaluate the result in the PLC user program. **Example:** The interface signal: DB31, ... DBX94.7 (variable signaling function) should be set to a 1 if the motor torque exceeds 50 % of the rated torque.
Corresponding to	MD1620 $MD_PROG_SIGNAL_FLAGS (bits variable signal function) MD1621 $MD_PROG_SIGNAL_NR (signal number variable signal function) MD1622 $MD_PROG_SIGNAL_ADDRESS (address variable signal function) MD1623 $MD_PROG_SIGNAL_THRESHOLD (threshold variable signal function) MD1624 $MD_PROG_SIGNAL_HYSTERESIS (hysteresis variable signal function) MD1625 $MD_PROG_SIGNAL_ON_DELAY (on delay variable signal function) MD1626 $MD_PROG_SIGNAL_OFF_DELAY (off delay variable signal function)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611D or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX95.0	**VDC link < warning threshold**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The drive signals the PLC that the DC link voltage VDC link has dropped below the DC link undervoltage warning threshold. The DC link undervoltage warning threshold is defined using: MD1604 $MD_LINK_VOLTAGE_WARN_LIMIT. The DC link undervoltage warning threshold should be defined to be greater than 400 V, depending on the application case. If the DC link voltage drops below 280 V, the unit is powered-down by the hardware.
Signal state 0 or edge change 1 → 0	The DC link voltage VDClink is greater than the DC link undervoltage warning threshold.
Application example(s)	If a warning signal is given, measures can be taken by the PLC user program, for example, to stop machining (e.g. start tool retraction) or to buffer the DC link voltage.
Corresponding to	MD1604 $MD_LINK_VOLTAGE_WARN_LIMIT (DC link undervoltage warning threshold)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611digital or /IAG/ SINUMERIK 810D Installation and Start-up Guide

DB31, ... DBX95.7	**i^2t monitoring**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The drive signals the PLC that the power unit monitoring has responded. The rated current limit of this i^2t monitoring function is defined in drive machine data MD1261 $MD_I2T_NOMINAL_REDUCTION. It is also possible to set the time for which the power unit may be at its limit in machine data: MD1262 $MD_DIAGNOSIS_I2T The i^2t monitoring function can be used to protect the power unit of digital drives against continuous overloads.
Signal state 0 or edge change 1 → 0	The i^2t monitoring function has not responded.
Application example(s)	Further measures can be initiated by the PLC user program, if necessary, when the warning signal is activated.
Corresponding to	MD1261 $MD_I2T_NOMINAL_REDUCTION (i^2t limiting, rated power unit current) MD1262 $MD_DIAGNOSIS_I2T ($i^2t$ time at the limit) MD1263 $MD_LIMIT_I2T ($i^2t$ actual limit factor) MD1264 $MD_LOAD_I2T ($i^2t$ actual utilization factor)
Additional references	/IAD/ SINUMERIK 840D Installation and Startup Guide; Chapter: SIMODRIVE 611digital or /IAG/ SINUMERIK 810D Installation and Start-up Guide

2.2 Axis monitoring, protection zones (A3)

2.2.1 Signals to channel (DB21, ...)

DB21, ... DBX1.1	**Enable protection zones**
Edge evaluation: yes	Signal(s) updated: cyclic
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: cyclic
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 active. 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 deactivated.
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: cyclic
Signal state 1 or edge change 0 → 1	The pre-activated, channel-specific protection zone 1 (...10) is activated by the PLC user program. The protection zone is immediately active. 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.

2.2.2 Signals from channel (DB21, ...)

DB21, ... DBX272.0 – DBX273.1	**Machine-related protection zone 1 (...10) pre-activated**
Edge evaluation: no	Signal(s) updated: cyclic
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: cyclic
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))

DB21, ... DBX274.0 – DBX275.1	**Channel-specific protection zone 1 (...10) pre-activated**
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: cyclic
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: cyclic
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.

2.2.3 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)

2.2.4 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: cyclic
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 (encoder frequency < ENC_FREQ_LIMIT_LOW). For the edge change 1 → 0, the encoder frequency must have fallen below the value of machine data: MD36302 $MA_ENC_FREQ_LIMIT_LOW

2.3 Continuous-path mode, exact stop and LookAhead (B1)

2.3.1 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.

2.3.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.6	**Position reached with exact stop coarse**
Edge evaluation: no	Signal(s) updated: cyclic
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 mode (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 movement was stopped with NC Stop • The spindle is in the closed-loop position-controlled mode (SPCON/SPOS instruction) and is stationary • The axis is switched from closed-loop speed-controlled to closed-loop position-controlled 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 movement was stopped with NC Stop • The spindle is in the closed-loop speed-controlled mode (SPCOF/SPOSA instruction) • The "follow-up" mode is active for the axis • The "parking" mode is active for the axis • The axis is switched from closed-loop position-controlled to closed-loop speed-controlled 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: cyclic
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)

2.4 Acceleration (B2)

No signal descriptions required.

2.5 Diagnostic tools (D1)

No signal descriptions required.

2.6 Travel to fixed stop (F1)

2.6.1 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 Edge change 1 → 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. 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)

2.6.2 Signals from axis/spindle (DB31, ...)

DB31, ... data block	**Activate travel to fixed stop Signal(s) from axis/spindle (NCK → PLC)**
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 speed-controlled to current or torque-controlled 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.

2.7 Velocities, setpoint/actual value systems, closed-loop control (G2)

No signal descriptions required.

2.8 Auxiliary function output to PLC (H2)

2.8.1 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)

2.8.2 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	At the start of the next OB1 cycle, the PLC basic program resets the change signals. The value of the data concerned is not valid.

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, ... DBX59.0 - DBX59.4	**M function 1-5 not included in list**
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, ... 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 PLC power-up. • All help functions are deleted before a new function is entered.

DB21, ... DBB68 - DBB97	**M functions 1 to 5 Extended address M functions 1 to 5**
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, DBB119	**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).

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	MD 22240 $MC_AUXFU_F_SYNC_TYPE (instant in time that the F functions are output)

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	DB 21; ... DBX30.5 (activate associated M01)

2.8.3 Signals from axis/spindle (DB31, ...)

DB31, ... DBD78	**Value of F help function**
Edge evaluation: no	Signal(s) updated: Job-controlled
	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	**Value of M help function**
Edge evaluation: no	Signal(s) updated: Job-controlled
	The values for the M3, M4, M5 help functions are sent to the associated interface for the addressed spindle.

DB31, ... DBD88	**Value of S help function**
Edge evaluation: no	Signal(s) updated: Job-controlled
	The values for the S help functions are sent to the associated interface for the addressed spindle.

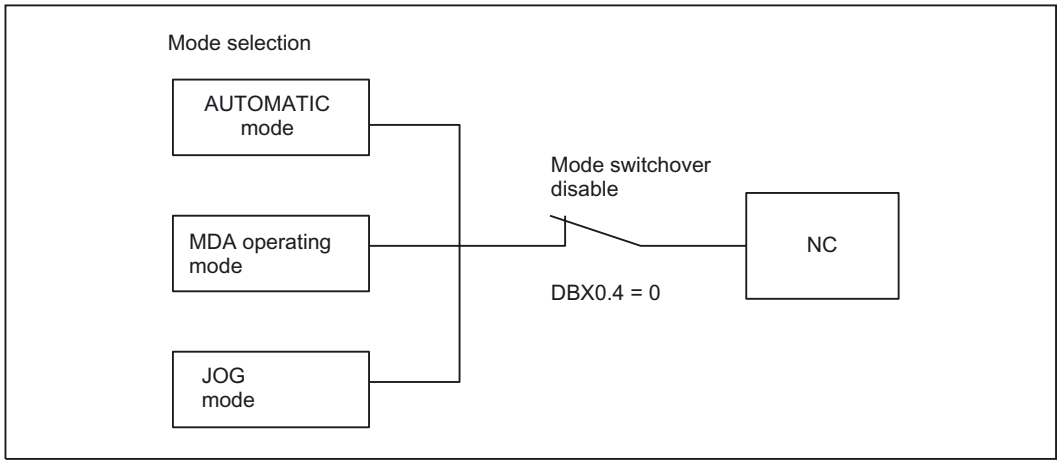
2.9 Mode group, channel, program operation, reset response (K1)

2.9.1 Signals to mode group (DB11)

DB11, ... DBX0.0	**AUTOMATIC mode**
Edge evaluation: no	Signal(s) updated: cyclic
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: cyclic
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, ... D0.4 (operating mode, changeover inhibit) = 1
Corresponding to	DB11, ... DBX6.1 (active MDA mode)

DB11, ... DBX0.2	**JOG mode**
Edge evaluation: no	Signal(s) updated: cyclic
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 changeover inhibit**
Edge evaluation: no	Signal(s) updated: cyclic
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: cyclic
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 affected.
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: cyclic
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: cyclic
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.

DB11, ... DBX0.7	**Mode group reset**
Corresponding to	DB21, ... DBX7.7 (channel reset) DB11, ... DBX6.7 (all channels in the reset state)
Special cases, errors, ...	An alarm which cancels the interface signal DB11, ... DBX6.3 (mode group ready) ensures that all of the 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: cyclic
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	/BA/ Operator's Guide

DB11, ... DBX1.1	**Machine function REPOS**
Edge evaluation: no	Signal(s) updated: cyclic
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 precise previous position using the REPOS machine function so that the program can be continued in the automatic mode.
Additional references	/BA/ Operator's Guide

DB11, ... DBX1.2	**Machine function REF**
Edge evaluation: no	Signal(s) updated: cyclic
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	/FB1/ Description of Functions, Basic Machine; Reference Point Approach (R1)

DB11, ... DBX1.6	**Single block type B**
Edge evaluation:	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. In this case, the control 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, single block type A is present. (If DB11, ... DBX1.6 and DB11, ... DBX1.7 are not set, it is impossible to determine which single block type is required. In this case, the control assumes: no single block across mode groups).
Corresponding to	Single block type A

DB11, ... DBX1.7	**Single block type A**
Edge evaluation:	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. (comparable to the 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. In this case, the control assumes: no single block across mode groups).
Signal state 0 or edge change 1 → 0	If DB11, ...DBX1.7 is not set and bit DB11, ... DBX1.6 is set, single block type B is present. (If DB11, ... DBX1.6 and DB11, ... DBX1.7 are not set, it is impossible to determine which single block type is required. In this case, the control assumes: no single block across mode groups).
Corresponding to	Single block type B

2.9.2 Signals from mode group (DB11)

DB11, ... DBX4.0	**Selected mode AUTOMATIC**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	AUTOMATIC mode is selected by the MMC.
Signal state 0 or edge change 1 → 0	AUTOMATIC mode is not selected by the MMC.

DB11, ... DBX4.1	**Selected mode MDA**
Edge evaluation:	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	MDA mode is selected by the MMC.
Signal state 0 or edge change 1 → 0	MDA mode is not selected by the MMC.

DB11, ... DBX4.2 Data block	**Selected JOG mode Signal(s) from the mode group (MMC → PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	JOG mode is selected by the MMC.
Signal state 0 or edge change 1 → 0	JOG mode is not selected by the MMC.

DB11, ... DBX5.0	**Selected machine function TEACH IN**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Machine function TEACH IN is selected in the mode group by the MMC.
Signal state 0 or edge change 1 → 0	Machine function TEACH IN is not selected by the MMC.
Additional references	/BA/ Operator's Guide

DB11, ... DBX5.1	**Selected REPOS machine function**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Machine function REPOS is selected in the mode group by the MMC.
Signal state 0 or edge change 1 → 0	Machine function REPOS is not selected by the MMC.
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	/BA/ Operator's Guide

DB11, ... DBX5.2	**Selected machine function REF**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The REF machine function is selected in the mode group by the MMC.
Signal state 0 or edge change 1 → 0	Machine function REF is not selected by the MMC.
Additional references	/FB1/ Description of Functions, Basic Machine; Reference Point Approach (R1)

DB11, ... DBX6.0 Data block	**Active mode AUTOMATIC Signal(s) from the mode group (NCK → PLC)**
Edge evaluation: no	Signal(s) updated: cyclic
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: cyclic
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: cyclic
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: cyclic
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 parts 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: cyclic
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: cyclic
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	/BA/ Operator's Guide

DB11, ... DBX7.1	**Active REPOS machine function**
Edge evaluation: no	Signal(s) updated: cyclic
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.
Further references	/BA/ Operator's Guide

DB11, ... DBX7.2	**Active machine function REF**
Edge evaluation: no	Signal(s) updated: cyclic
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.
Further references	/FB1/ Description of Functions, Basic Machine; Reference Point Approach (R1)

2.9.3 Signals to channel (DB21, ...)

DB21, ... DBX0.4	**Activate single block**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	In AUTOMATIC and MDA 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 or edge change 1 → 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.
Corresponds to	DB21, ... DBX35.3 (program status interrupted)

DB21, ... DBX0.5	**Activate M01**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Activation of program control "Conditional stop" `M01` is requested.
Signal state 0 or edge change 1 → 0	Activation of program control "Conditional stop" `M01` is not requested.
Corresponds to	DB21, ... DBX24.5 (`M01` selected) DB21, ... DBX32.5 (`M0`/`M01` active)

DB21, ... DBX1.6	**PLC action completed**
Edge evaluation: no	Signal(s) updated: cyclic
	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 or edge change 0 → 1	PLC action is completed.
Signal state 0 or edge change 1 → 0	PLC action is not yet completed.
Corresponds 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: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 0	Activation of the program test is not requested.
Corresponds to	DB21, ... DBX25.7 (program test selected) DB21, ...DBX33.7 (program test active)

DB21, ... DBX2.0	**Skip block**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 0	Skip blocks marked in the part program with an slash (/) are processed.
Corresponds 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: cyclic
Signal state 1 or edge change 0 → 1	The main run reads in no more preprocessed part program blocks. Note The signal is only active in AUTOMATIC and MDA modes.
Signal state 0 or edge change 1 → 0	The main run reads in preprocessed part program blocks.
Corresponds to	DB21, ... DBX35.0 (program status running)

DB21, ... DBX6.4	**Program level abort**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	At each edge change 0 → 1 the current program level being processed (subroutine level, ASUB level, save routine) is immediately aborted. Processing of the part program continues at the next higher program level from the exit point.
Signal state 0 or 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: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 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.
Corresponds to	DB21, ... DBX7.1 (NC START)

DB21, ... DBX7.1	**NC START**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or 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. Operating mode MDA: The entered block information or part program blocks are released for execution.
Signal state 0 or edge change 1 → 0	No effect.
Corresponds to	DB21, ... DBX7.0 (NC start disable)

DB21, ... DBX7.2	**NC STOP at block limit**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 0	No effect.
Corresponds 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: cyclic
Signal state 1 or edge change 0 → 1	AUTOMATIC or MDA 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 or edge change 1 → 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 are cleared at the next NC START.
Corresponds 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: cyclic
	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: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 0	No effect.
Corresponds to	DB11, ... DBX0.7 (mode group reset) DB21, ... DBX35.7 (channel status reset)

DB21, ... DBX31.0 - DBX31.2	**REPOS mode (REPOSPATHMODE)**
Edge evaluation: no	Signal(s) updated: cyclic

	REPOS mode specified by HMI:
	`Bit:2 1 0`
	`0 0 0 =` 0: no REPOS mode active
	`0 0 1 =` 1: Re-approach to block start `RMB`
	`0 1 0 =` 2: Re-approach to interruption point `RMI`
	`0 1 1 =` 3: Re-approach to block end point `RME`
	`1 0 0 =` 4: Re-approach to nearest path point `RMN`

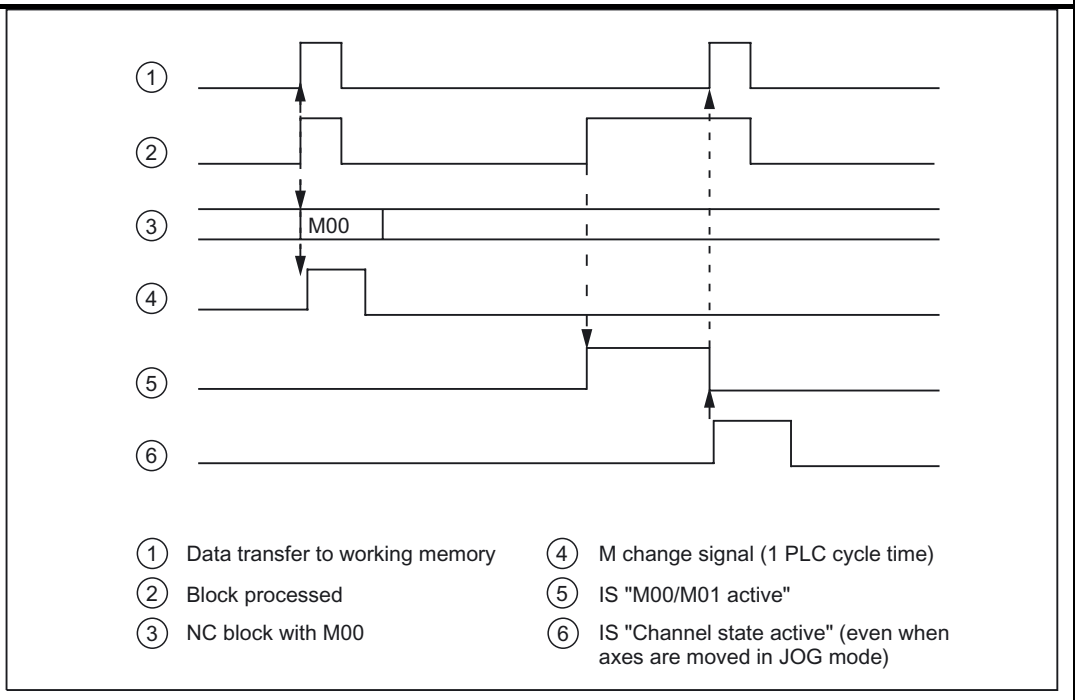
Corresponds to	DB21, ...DBX25.0 - DBX25.2 (REPOS mode (REPOSPATHMODE)) DB31, ... DBX10.0 (REPOSDELAY)

DB21, ... DBX31.4	**REPOS mode change (REPOSMODEEDGE)**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The REPOS mode has changed: DB21, ... DBX31.0 - DBX31.2 (REPOS mode (REPOSPATHMODE))
Signal state 0 or edge change 1 → 0	REPOS mode has not changed.
Corresponds to	DB21, ... DBX31.0 - DBX31.2 (REPOS mode (REPOSPATHMODE)) DB21, ... DBX319.0 (REPOSMODEEDGEACKN) DB21, ...DBX31.0 - DBX31.2 (REPOS mode (REPOSPATHMODE))

2.9.4 Signals from channel (DB21, ...)

DB21, ... DBX32.3	**Action block active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The action block is being executed.
Signal state 0 or edge change 1 → 0	No action block active.
Additional references	/BA/ Operator's Guide

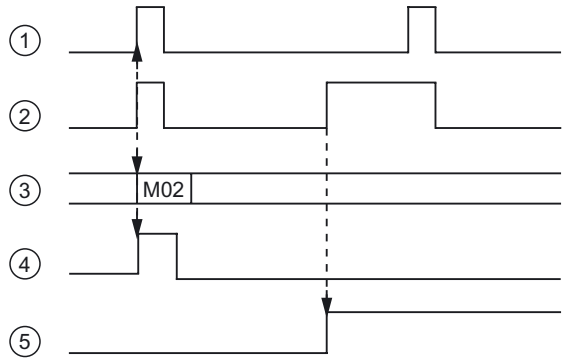
DB21, ... DBX32.5	**M00/M01 active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The part program block is processed, the auxiliary functions are output, and: • M00 is in the RAM • M01 is in the RAM and IS "Activate M01" is active The program status changes to "Stopped".
Signal state 0 or edge change 1 → 0	• With DB21, ... DBX7.1 (NC start) • For a program abort as a result of a reset

DB21, ... DBX32.5	**M00/M01 active**
Fig.	
Corresponds to	DB21, ... DBX0.5 (activate `M01`) DB21, ... DBX24.5 (`M01` selected)

DB21, ... DBX32.6	**Last action block active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The last action block is being executed.
Signal state 0 or edge change 1 → 0	The last action block is not being executed.
Further references	/BA/ Operator's Guide

DB21, ... DBX33.4	**Block search active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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)

DB21, ... DBX33.4	**Block search active**
Signal state 0 or edge change 1 → 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	/BA/ Operator's Guide

DB21, ... DBX33.5	**M02/M30 active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	• NC block with M02 or M30 (or M17 if a subroutine 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 MDA mode or machine functions REF or PRESET are selected • After DB10 DBX56.2 (acknowledge EMERGENCY STOP)
Signal state 0 or edge change 1 → 0	• No end of program or program abort • Status after the controller is switched on • Start of an NC program
Fig.	
Application example(s)	The PLC can detect the end of program processing with this signal and react appropriately.

DB21, ... DBX33.5	**M02/M30 active**
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: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 0	No transformation active.
Additional references	/PA/ Programming Guide Fundamentals

DB21, ... DBX33.7	**Program test active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 operator 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 or edge change 1 → 0	Program control "Program test" is not active.
Corresponds 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: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 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 MDA). • 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.
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"
Corresponds to	DB21, ... DBX6.1 (read-in disable)

DB21, ... DBX35.1	**Program status wait**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 0	Program status wait is not active.
Further references	/PA/ Programming Guide Fundamentals

DB21, ... DBX35.2	**Program status stopped**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 0	"Program status stopped" is not present.
Corresponds 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: cyclic
Signal state 1 or edge change 0 → 1	When the operating mode changes from AUTOMATIC or MDA (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 MDA mode when NC start is issued.
Signal state 0 or edge change 1 → 0	"Program status aborted" is not present.
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: cyclic
Signal state 1 or edge change 0 → 1	The program has been selected but not started, or the current program was aborted with a reset.
Signal state 0 or edge change 1 → 0	Program status interrupted is not active.
Corresponds to	DB21, ... DBX7.7 (reset)

DB21, ... DBX35.5	**Channel status active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	In this channel: • A part program is presently being executed in Automatic or MDA mode or • At least one axis is being traversed in JOG mode.
Signal state 0 or edge change 1 → 0	"Channel status interrupted" or "Channel status Reset" is active.

DB21, ... DBX35.6	**Channel status interrupted**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The NC part program in AUTOMATIC or MDA 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 or edge change 1 → 0	"Channel status active" or "Channel status Reset" is active.

DB21, ... DBX35.7	**Channel status reset**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 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 or edge change 1 → 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:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	One or several channels in the mode group are not in the required operating state as the result of an active interrupt routine. The signal is not set if an interrupt routine is running in a program mode.
Signal state 0 or edge change 1 → 0	All channels are operating in the required mode.
Corresponds to	MD11600 $MN_BAG_MASK

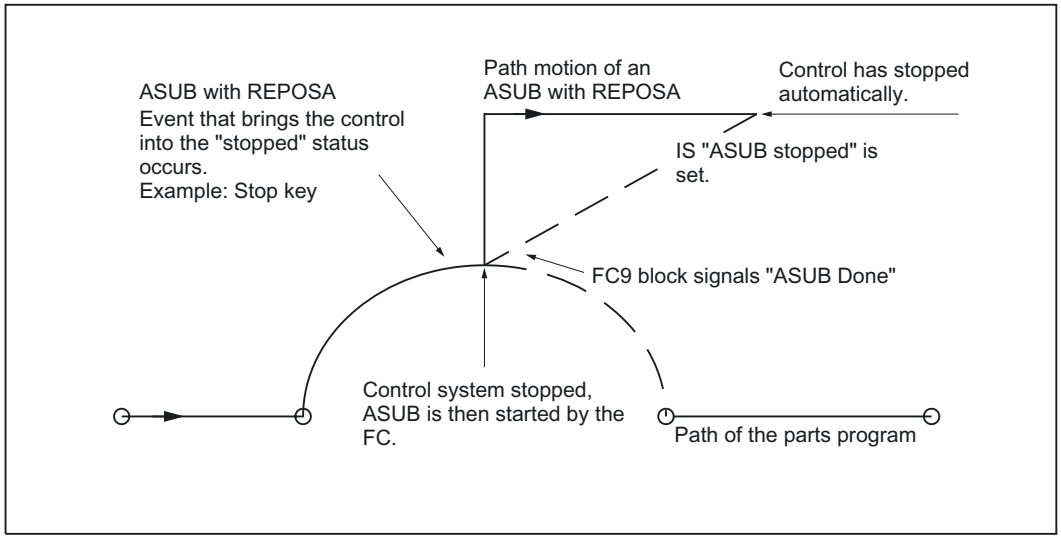
DB21, ... DBX37.6	**Read-in disable is ignored**
Edge evaluation:	Signal(s) updated:
	The following machine data are used to specify that the read-in disable (DB21, ... DBX6.1) is to be ignored: • MD11602 $MN_ASUP_START_MASK, Bit 2 = 1 (start also permitted if read-in disable is active) • MD20116 $MC_IGNORE_INHIBIT_ASUP (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)
Corresponds 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.

DB21, ... DBX37.7	**Stop at block end is ignored during single block (SBL)**
Signal state 1	Single block is active (DB21, ... DBX0.4==1) AND part program block is single block-inoperative.
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)
Corresponds 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: cyclic
Signal state <> 0	A G function or mnemonic identifier of the G group is active. The active G function can be determined from the value (starting with 1): 1 = First G function of the G group 2 = Second G function of the G group 3 = Third G function of the G group : Please refer to the programming guide for a list of the possible G groups and associated functions.
Signal state = 0	No G function or G group mnemonic identifier is active.
Application example(s)	The active G group is stored in binary code in the relevant byte. The following evaluation applies: Bit 7 6 5 4 3 2 1 0 Evaluation 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	/PA/ Programming Guide Fundamentals

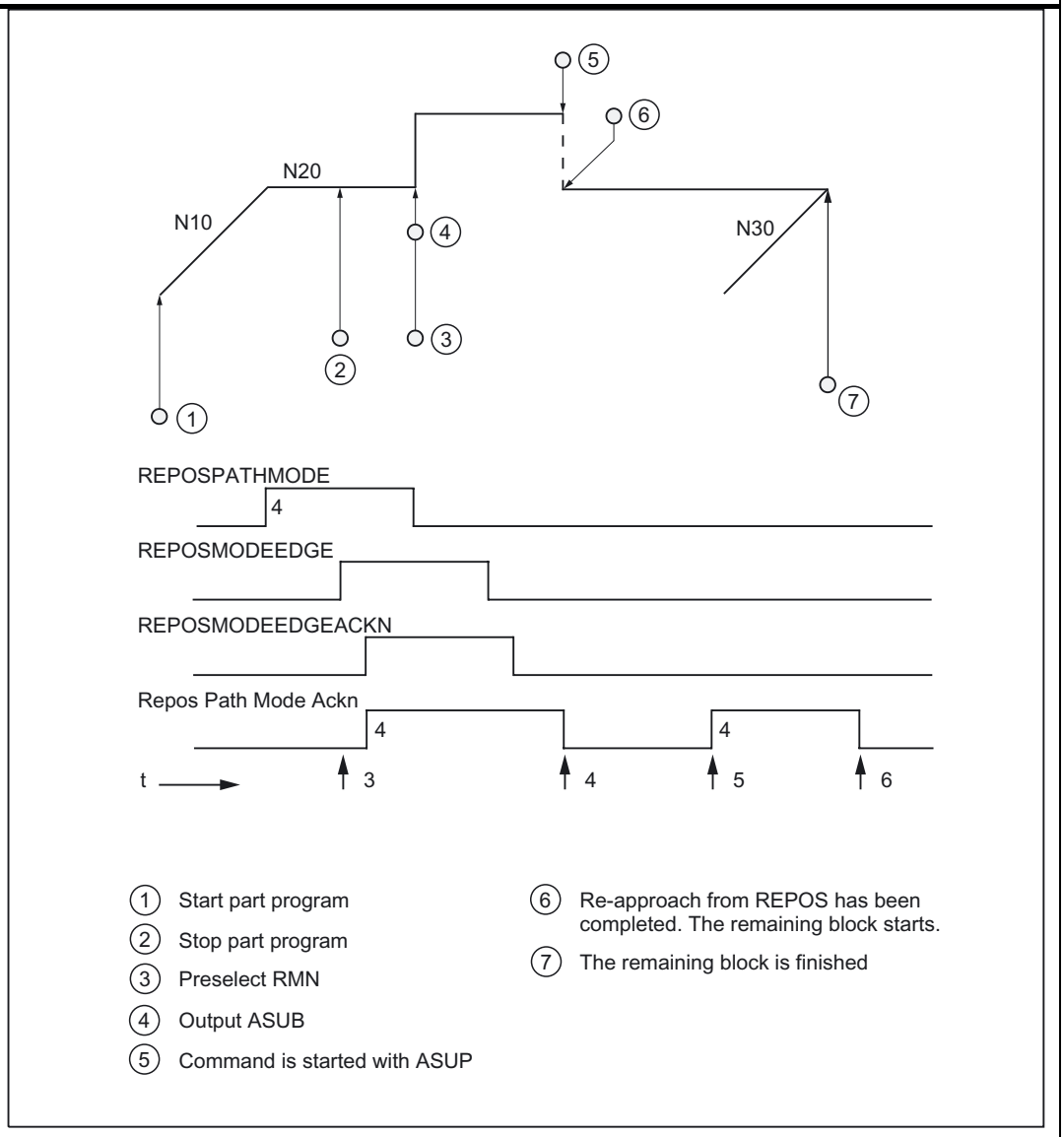
DB21, ... DBX318.0	**ASUB is stopped**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The signal is set to 1 if the control stops automatically prior to the end of the ASUB. The IS "ASUB is stopped" is only supplied in the case "Interrupt in a program mode and channel status stopped".
Signal state 0 or edge change 1 → 0	The IS "ASUB is stopped" is set to 0 with start and reset.

DB21, ... DBX318.0	**ASUB is stopped**
	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 controller assumes program status "Stopped". • The PLC initiates an ASUB via block FC9. • Before the re-approach to the contour, the controller stops and goes to program status "Stopped". The IS "ASUB is stopped" is set. • The user presses Start. The IS "ASUB is stopped" is reset, and 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**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	During processing of the part program blocks as part of the block search (internal channel status: "Program test"), up until the target block is changed to in the main run (program status: "Stopped").
Signal state 0 or edge change 1 → 0	When the target block is changed to in the main run (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	**REPOSMODEEDGEACKN**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The interface signal identified by the NCK: DB21, ... DBX31.4 (REPOSMODEEDGE) is acknowledged using the interface signal: DB21, ... DBX319.0 (REPOSMODEEDGEACKN), if level signals from: DB21, ... DBX31.0 - DBX31.2 (REPOSMODE0-2) and from: DB31, ... DBX10.0 (REPOSDELAY) are transferred to the NC. The levels relate to the current block in the main run.
Signal state 0 or edge change 1 → 0	SERUPRO-ASUB stops automatically before REPOS, and IS "REPOSMODEEDGE" does not influence the SERUPRO approach.
Corresponds to	DB21, ... DBX31.4 (REPOSMODEEDGE)

DB21, ... DBX319.1 - DBX319.3	**Repos Path Mode Ackn 0 - 2**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1	Using the interface signal: DB21, ... DBX319.1 - DBX319.3 (Repos Path Mode Ackn 0-2) with the 3 bits, one of the functions for the re-approach point RMB, RMI, RME or RMN can be acknowledged according to the following coding to the PLC: DB21, ... DBX319.1 - DBX319.3 = 1 → RMB: Re-approach at the start of the block DB21, ... DBX319.1 - DBX319.3 = 2 → RMI: Re-approach at the point of interruption DB21, ... DBX319.1 - DBX319.3 = 3 → RME: Re-approach at the end of the block DB21, ... DBX319.1 - DBX319.3 = 4 → RMN: Re-approach at the nearest path point
Signal state 0	DB21, ... DBX319.1 - DBX319.3 = 0 → RMNOTDEF: The ReposMode that is not re-defined is acknowledged to the PLC.
	Example of the sequence of REPOS acknowledgments in the part program and signal timing of the acknowledgement process from the NCK:

DB21, ... DBX319.1 - DBX319.3	**Repos Path Mode Ackn 0 - 2**
	
Corresponds to	DB21, ... DBX31.0 - DBX31.2 (REPOSPATHMODE0-2) DB21, ... DBX31.4 (REPOSMODEEGGE) DB21, ... DBX319.0 (REPOSMODEEGGEACKN) DB31, ... DBX70.2 (Repos Delay Ackn)
Additional references	/FB1/ Description of Functions Basic Machine; Mode Group, Channel, Program Operation, Reset Behavior (K1), Chapter: " Block search, type 5 SERUPRO for block search"

DB21, ... DBX319.5	**Repos DEFERAL Chan**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	All axes currently controlled by this channel have either no REPOS offset or their REPOS offsets are invalid.
Signal state 0 or edge change 1 → 0	Miscellaneous.
Corresponds to	DB31, ... DBX70.0 (Repos offset)

DB21, ... DBB376	**PROG-EVENT-DISPLAY**
Edge evaluation: no	Signal(s) updated: event-driven
Signal state 1 or edge change 0 → 1	The event associated with the bit has occurred.
Signal state 0 or edge change 1 → 0	The event associated with the bit is no longer pending.
Bit assignments	Bit 0 → part program start from channel status RESET Bit 1 → end of part program Bit 2 → operator panel reset Bit 3 → run-up Bit 4 → 1st start after the search run Bit 5 - 7 → reserved, currently always 0 part program All bits 0 → no PROG-EVENT is active Signal duration: at least one complete PLC cycle

2.9.5 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)

2.9.6 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.

DB31, ... DBX72.0	**REPOSDELAY**
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.

2.10 Axis types, coordinate systems, frames (K2)

2.10.1 Signals to axis/spindle (DB31, ...)

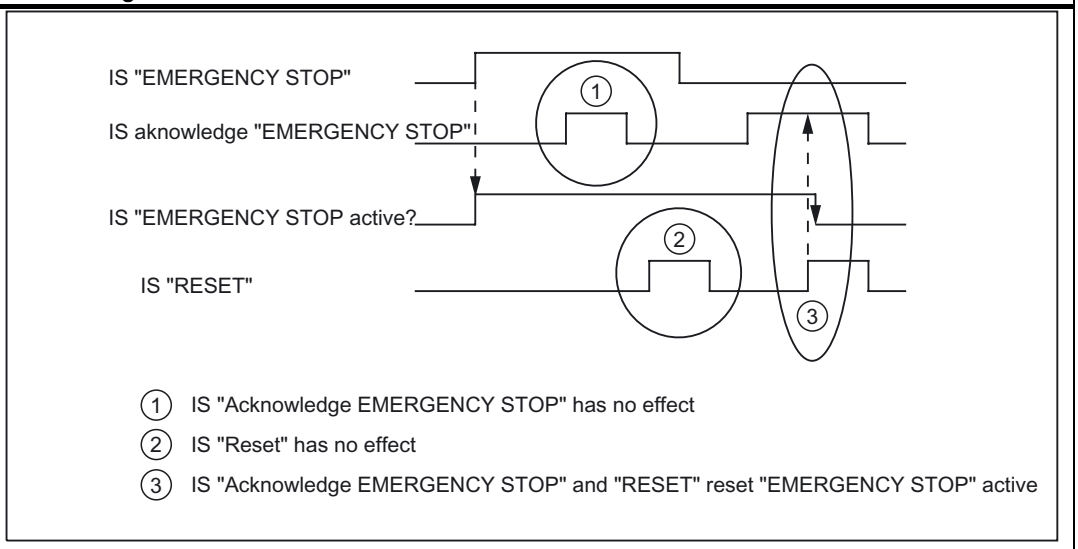
DB 31, ... DBX3.0	**External Zero Offset**
Edge evaluation: no	Signal(s) updated:
Signal state 1 or edge change 0 → 1	The preselected value of the external work offeset of an axis is used as the new value for calculating the total work offset between the basic and the workpiece coordinate systems.
Signal 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]

2.11 EMERGENCY STOP (N2)

2.11.1 Signals to NC (DB10)

DB10 DBX56.1	**EMERGENCY STOP**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The NC is switched to the EMERGENCY STOP state and the EMERGENCY STOP procedure is started on the NC.
Signal state 0 or edge change 1 → 0	The NC is not in the EMERGENCY STOP state. The EMERGENCY STOP status 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 (acknowledge EMERGENCY STOP)

DB10 DBX56.2	**Acknowledge EMERGENCY STOP**
Edge evaluation: no	Signal(s) updated: cyclic
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 canceled • Part program processing is interrupted 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 (acknowledge EMERGENCY STOP) DB11 DBX0.7 (mode group reset)

2.11.2 Signals from NC (DB10)

DB10 DBX106.1	EMERGENCY STOP active
Edge evaluation: no	Signal(s) updated: cyclic
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)

2.12 Transverse axes (P1)

No signal descriptions required.

2.13 PLC basic program (P3)

For a description of the NC/PLC interface signals, refer to:
References:
/FB1/ Description of Functions Basic Machine;
Power line basic PLC program (P3) / Solution line basic PLC program (P3 sl),
Section: "Signal/Data Descriptions".

2.14 Reference point approach (R1)

2.14.1 Signals to channel (DB21, ...)

DB21, ... DBX1.0	**Activate referencing**
Edge evaluation: yes	Signal(s) updated: cyclic
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) Each machine axis assigned to the channel can be referenced with channel-specific referencing (this is achieved internally on the control by simulating the plus/minus travel keys). Using the axis-specific machine data: MD34110 $MA_REFP_CYCLE_NR (axis sequence for channel-specific referencing) it is possible to define in which sequences the machine axes are referenced. When all of the axes entered in REFP_CYCLE_NO have reached their reference point, the interface signal: DB21, ... DBX36.2 (all axes referenced) is set.
Application example(s)	If the machine axes are to be referenced in a particular sequence, the following options are available: • 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 is used.
Corresponding to	DB21, ... DBX33.0 (referencing active) DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)

DB21, ... DBX33.0	**Referencing active**
Edge evaluation: yes	Signal(s) updated: cyclic
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). Channel-specific referencing is operational.
Signal state 0 or edge change 1 → 0	• Channel-specific referencing has been completed • Axis-specific referencing running • No referencing active
Signal irrelevant for ...	Spindles
Corresponding to	DB21, ... DBX1.0 (activate referencing)

2.14.2 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)

2.14.3 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 distance-coded 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)

2.14.4 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 (260 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)
Additional references	/FB1/ Description of Functions, Basic Machine; Spindles (S1)

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 (260 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
Additional references	/FB1/ Description of Functions, Basic Machine; Spindles (S1)

2.15 Spindles (S1)

2.15.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX2.2	**Spindle reset/delete distance-to-go**
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, ... DBX16.7	**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.

DB31, ... DBX16.7	**Delete S value**
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, ... DBX16.4 - DBX16.5	**Resynchronizing spindles 2 and 1**
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, ... DBX16.3	**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).

DB31, ... DBX16.3	**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, ... DBX16.0 - DBX16.2	**Actual gear stage A to C**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 (status-controlled)	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 program signals a different actual gear stage back to the NCK than that issued by the NCK as the setpoint gear stage, the gear stage change is still considered as having been successfully completed and the actual gear stage A to C is activated.

DB31, ... DBX16.0 - DBX16.2	**Actual gear stage A to C**
Corresponding to	DB31, ... DBX82.0 - DBX82.2 (setpoint gear stage A to C) DB31, ... DBX82.3 (change over gear stage) DB31, ... DBX16.3 (gear stage is changed over) DB31, ... DBX18.5 (oscillation speed) Parameter sets for gear stages

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, ... DBX17.4 - DBX17.5	**Re-synchronize spindle when positioning 2 and 1**
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, ... DBX18.6 - DBX18.7	**Setpoint direction of rotation, counter-clockwise/setpoint 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.

DB31, ... DBX18.6 - DBX18.7	**Setpoint direction of rotation, counter-clockwise/setpoint direction of rotation, clockwise**
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)

DB31, ... DBX18.5	**Oscillation speed**
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.

DB31, ... DBX18.5	**Oscillation speed**
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, ... DBX18.4	**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 the PLC) 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, ... DBX30.4	**Spindle positioning**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	DB31, ... DBX30.4 (spindle positioning) (corresponds to M19 position for spindle positioning)
Application example(s)	If the spindle is to be stopped from direction of rotation (M3 or M4) with orientation, or is to be re-oriented from standstill (M5), then the positioning mode is selected with SPOS, SPOSA or M19.
Corresponding to	MD20850 $MC_SPOS_TO_VDI (for SPOS/SPOSA, "M19" is output to the VSI interface) SD43240 $SA_M19_SPOS (position for spindle positioning with M19) DB31, ... DBX84.5 (active spindle operating mode, positioning mode)

DB31, ... DBX30.3	**Select gear step**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	This function is presently being prepared. This is the reason that interface signal is temporarily reserved.
Application example(s)	One parameter set each is provided for each of the 5 gear stages. When changing over to the spindle mode, the corresponding parameter set 1 to 5 is selected according to the gear stage engaged. A gear stage can be preselected as follows: • M40 automatically by the programmed spindle speed • M41 to M45 permanently by the part program • By the PLC, using function block FC 18 • In the RESET state by writing to the VDI interface
Corresponding to	MD35010 $MA_GEAR_STEP_CHANGE-ENABLE (the gear stage can be changed) MD35590 $MA_PARAMSET_CHANGE-ENABLE (a parameter set can be specified by the PLC) DB31, ... DBX82.3 (change over gear stage)

DB31, ... DBX30.2	**Spindle start, counter-clockwise rotation**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	DB31, ... DBX30.2 (spindle start, counter-clockwise) (corresponds to M4 counter-clockwise spindle direction of rotation for the master spindle)
Application example(s)	The contents of the setting date: SD43200 $SA_SPIND_S can be read and become effective with the positive edge of the start signal "spindle start, counter-clockwise direction of rotation".
Corresponding to	DB31, ... DBX18.7 (setpoint direction of rotation, counter-clockwise) DB31, ... DBX18.6 (setpoint direction of rotation, clockwise) DB31, ... DBX17.6 (invert M3/M4)

DB31, ... DBX30.1	**Spindle start, clockwise rotation**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	DB31, ... DBX30.1 (spindle start, clockwise) (corresponds to M3 spindle direction of rotation, clockwise for the master spindle)
Application example(s)	With the positive edge of the starting signal "spindle start clockwise", the contents of the setting data: SD43200 $SA_SPIND_S are read and become effective.
Corresponding to	DB31, ... DBX18.7 (setpoint direction of rotation, counter-clockwise) DB31, ... DBX18.6 (setpoint direction of rotation, clockwise) DB31, ... DBX17.6 (invert M3/M4)

DB31, ... DBX30.0	**Spindle stop**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	DB31, ... DBX30.0 (spindle stop) (corresponds to M5 spindle stop for master spindle).
Application example(s)	A spindle stop is executed if the interface signal: DB31, ... DBX61.4 (axis/spindle stationary) = 1 is signaled.
Corresponding to	DB31, ... DBX17.6 (axis/spindle stationary)

2.15.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.0	**Spindle/no axis**
Edge evaluation: yes	Signal(s) updated: cyclic
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 • Rigid tapping • 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, ... DBX82.3	**Change gear stage**
Edge evaluation: yes	Signal(s) updated: cyclic
	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 programmed spindle speed using M function `M40` If specified set speed requires gear stage change => DB31, ... DBX82.3 (change gear) = 1 DB31, ... DBX82.0 - DBX82.2 (set gear stage) = set 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
Corresponds to	DB31, ... DBX82.0 - DBX82.2 (set gear stage A to C) DB31, ... DBX16.0 - 16.2 (actual gear stage A to C) DB31, ... DBX16.3 (gear is changed)

DB31, ... DBX82.0 - 82.2	**Setpoint gear stage A to C**
Edge evaluation: yes	Signal(s) updated: cyclic
	See DB31, ... DB82.3 (change gear).
Signal state 1 or edge change 0 → 1	The set gear stage is output in coded format: 1. Gear stage 0 0 0 (C B A) 1. Gear stage 0 0 1 2. Gear stage 0 1 0 3. Gear stage 0 1 1 4. Gear stage 1 0 0 5. Gear stage 1 0 1 Invalid value 1 1 0 Invalid value 1 1 1
Signal irrelevant for ...	... Other spindle modes except oscillation mode.
Corresponds to	DB31, ... DBX82.3 (change gear) DB31, ... DBX16.0 - DBX16.2 (actual gear stage A to C) DB31, ... DBX16.3 (gear is changed)

DB31, ... DBX83.7	**Actual direction of rotation clockwise**
Edge evaluation: yes	Signal(s) updated: cyclic
	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
Corresponds to	DB31, ... DBX61.4 (axis/spindle stationary)

DB31, ... DBX83.5	**Spindle in the set range**
Edge evaluation: yes	Signal(s) updated: cyclic
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).
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 set range) == 1) THEN (DB21, ... DBX6.0 (feed disable) = 0) ELSE (DB21, ... DBX6.0 (feed disable) = 1) Note: Positioning axes are also stopped.
Corresponds to	MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START MD35500 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)

DB31, ... DBX83.2	**Setpoint speed increased (programmed speed too low)**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The set speed of the spindle that is specified directly (programmed spindle speed) or indirectly (constant cutting speed) would violate the current minimum limit value. The set speed is therefore defined at the minimum limit value. Limit values: • Minimum speed of the specified gear stage • Minimum spindle speed • Speed limiting by means of NC/PLC interface • Programmed spindle speed limiting G25 • Programmed spindle-speed limiting for G96
Signal state 0 or edge change 1 → 0	The set speed of the spindle is outside the minimum limit value.
Application example(s)	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 can accept this state as permissible and enable the path feed, or he can disable the path feed or the complete channel, IS: DB31, ... DBX83.5 (spindle in set range) is processed.
Corresponds to	DB21, ... DBX6.0 (feed disable) DB31, ... DBX4.3 (feed/spindle stop) DB31, ... DBX83.5 (spindle in set range)

DB31, ... DBX83.1	**Set speed limited (programmed speed too high)**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The set speed of the spindle that is specified directly (programmed spindle speed) or indirectly (constant cutting speed) would violate the current maximum limit value. The set speed is therefore defined at the maximum limit value. Limit values: • Maximum speed of specified gear stage • Maximum spindle speed • Speed limited by VDI • Programmed spindle speed limiting G26 • Programmed spindle speed limiting for G96
Signal state 0 or edge change 1 → 0	The set speed of the spindle is outside the maximum limit value.
Application example(s)	The interface signal: DB31, ... DBX83.1 (set speed limited) can be used to determine if the programmed speed is unattainable. The PLC user can accept this state as permissible and enable the path feed, or can disable the path feed and/or the complete channel, IS: DB31, ... DBX83.5 (spindle in set range) is processed.

DB31, ... DBX83.0	**Speed limit exceeded**
Edge evaluation: yes	Signal(s) updated: cyclic
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
Corresponds to	MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance) MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

DB31, ... DBX84.7	**Active spindle mode: Control mode**
Edge evaluation: yes	Signal(s) updated: cyclic
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
Corresponds to	DB31, ... DBX84.6 (spindle mode oscillation mode) DB31, ... DBX84.5 (spindle mode positioning mode)

DB31, ... DBX84.6	**Active spindle mode: Oscillation mode**
Edge evaluation: yes	Signal(s) updated: cyclic
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.
Corresponds to	DB31, ... DBX84.7 (spindle mode control mode) DB31, ... DBX84.5 (spindle mode positioning mode) DB31, ... DBX82.3 (change gear)

DB31, ... DBX84.5	**Active spindle mode: Positioning mode**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Positioning mode (SPOS or SPOSA) is active.
Corresponds to	DB31, ... DBX84.7 (spindle mode control mode) DB31, ... DBX84.6 (spindle mode oscillation mode)

DB31, ... DBX84.3	**Rigid tapping active**
Edge evaluation: yes	Signal(s) updated: cyclic
	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 is 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 (servo enable) = 0 • DB31, ... DBX8.3 (feed stop) = 1

DB31, ... DBB86 - DBB87	**M function for spindle**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	One of the following M functions was output from the NCK to the PLC: `M3`, `M4`, `M5`, `M70`. Output occurs by means of: See "Corresponds to ...." below
Corresponds to	DB31, ... DBB86 - DBB87 (M function for spindle), axis-specific DB21, ... DBB58, DBB68 - DBB97 (M function for spindle), channel-specific

DB31, ... DBB88 - DBB91	**S function for spindle**
Edge evaluation: yes	Signal(s) updated: cyclic
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 speed 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
Corresponds to	DB31, ... DBB88 - DBB91 (S function for spindle), axis-specific DB21, ... DBB60, DBB98 - DBB115 (S function for spindle), channel-specific

2.16 Feeds (V1)

2.16.1 Signals to channel (DB21, ...)

DB21, ... DBX0.6	**Activate dry run feed**
Edge evaluation: yes	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The dry run feed rate 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 feed rate is greater than that programmed. The dry run feed rate 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 feed rate.
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 feed rate selected) SD42100 $SC_DRY_RUN_FEED (dry run feed rate)

DB21, ... DBB4	**Feed rate override**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The feed rate 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 feed rate override. With Gray coding, the individual switch settings are assigned to the following codes:

DB21, ... DBB4	Feed rate override

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 feed rate 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 control panels is described in the Configuring Guides for 840D/810D.

Corresponding to	DB21, ... DBX6.7 (feed rate override active) MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feed rate 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: cyclic
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. With Gray coding, the individual switch settings are assigned to the following codes:

DB21, ... DBB5	**Rapid traverse override**

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 control panels is described in the Configuring Guides for 840D/810D.

Corresponding to	DB21, ... DBX6.6 (rapid traverse override active) MD12050 $MN_OVR_FACTOR_RAPID_TRA[n] (evaluation of the path feed rate 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: cyclic
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 feed rate 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: cyclic
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	**Feed rate override active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The feed rate override between 0 and a maximum of 200% entered at the PLC interface is active for the path feed rate and therefore automatically for the related axes. In JOG mode, the feed rate override acts directly on the axes. The override factor is entered using the machine data: MD12020 $MN_OVR_FEED_IS_GRAY_CODE (path feed rate override factor, gray-coded) and MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feed rate override switch).
Signal state 0 or edge change 1 → 0	The feed rate override entered at the PLC interface is ignored. When the feed rate 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 feed rate override switch on the machine control panel. Using the interface signal: DB21, ... DBX6.7 (feed rate override active), the feed rate 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 feed rate override is inactive when G33, G63, G331, G332 are active.
Corresponding to	DB21, ... DBB4 (feed rate override)

DB21, ... DBX12.3, DBX16.3, DBX20.3	**Feed stop (geometry axis 1 to 3)**
Edge evaluation: no	Signal(s) updated: cyclic
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.

DB21, ... DBX12.3, DBX16.3, DBX20.3	**Feed stop (geometry axis 1 to 3)**
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 cancelled, this is executed immediately.

DB21, ... DBX24.6	**Dry run feed rate selected**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Dry run feed rate is selected. Instead of the programmed feed rate, the dry run feed rate 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 feed rate is not selected. The programmed feed rate is active.
Corresponding to	DB21, ... DBX0.6 (activate dry run feed) SD42100 $SC_DRY_RUN_FEED (dry run feed rate)

DB21, ... DBX25.3	**Feed rate override selected for rapid traverse**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The feed rate 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 (feed rate 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 (feed rate 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 feed rate 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 feed rate 1 - 4 for path/geometry axes**
Edge evaluation: no	Signal(s) updated: cyclic
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. 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
Corresponding to	MD12202 $MN_PERMANENT_FEED[n] MD12200 $MN_RUN_OVERRIDE_0

2.16.2 Signals to axis/spindle (DB31, ...)

DB31, ... DBB0	**Feed rate override (axis-specific)**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The axis-specific feed rate 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 / Axis-specific feed rate 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%

DB31, ... DBB0	Feed rate 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 feed rate override can be additionally limited. With Gray coding, the individual switch settings are assigned to the following codes:

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 feed rate 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 control panels is described in the Configuring Guides for 840D/810D.

Corresponding to	DB31, ... DBX1.7 (override effective) MD12010 $MN_OVR_FACTOR_AX_SPEED [n] (evaluation of the axis feed rate override switch) MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

DB 31, ... DBX3.2 - DBX3.5	**Activate fixed feed rate 1 - 4 for machine axes**
Edge evaluation: no	Signal(s) updated: cyclic
Description	These signals are used to select/de-select the function "Fixed feed" and define which fixed feed should be effective for machine axes.
Corresponding to	MD12202 $MN_PERMANENT_FEED[n] MD12200 $MN_RUN_OVERRIDE_0

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

DB31, ... DBB19	**Spindle override**
Edge evaluation: no	Signal(s) updated: cyclic
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: 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.

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%

DB31, ... DBB19	**Spindle override**
Signal state 1 or edge change 0 → 1	With Gray coding, the individual switch settings are assigned to the following codes: 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 control panels is described in the Configuring Guides for 840D/810D.
Corresponding to	DB 31, ... 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)

DB31, ... DBX1.7	**Override active**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	**Feed rate override active:** The axis-specific feed rate 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 feed rate override switch gray coded) and MD12010 $MN_OVR_FACTOR_AX_SPEED [n] (evaluation of the axis feed rate 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 feed rate override or spindle override is not active. If the feed rate 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. For a binary interface, the override factor = 0. For 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 feed rate override switch or the spindle override switch on the machine control panel. The "Feed rate override active" signal can be used to enable the feed rate 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 feed rate 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, ... DBX4.3	**Feed stop/spindle stop (axis-specific)**
Edge evaluation: no	Signal(s) updated: cyclic
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 parts 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 feed rate 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.
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.

2.16.3 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 axis-specific 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 axis-specific feed rate override.
Corresponding to	MD22240 $MC_AUXFU_F_SYNC_TYPE (output time F functions)

2.17 Tool offset (W1)

No signal descriptions required.

Index

D

DB10
- DBX103.0, 2-2
- DBX103.5, 2-2
- DBX103.6, 2-2
- DBX103.7, 2-3
- DBX104.7, 2-3
- DBX106.1, 2-114
- DBX108.1, 2-4
- DBX108.2 - DBX108.3, 2-4
- DBX108.6, 2-4
- DBX108.7, 2-5
- DBX109.0, 2-5
- DBX109.5, 2-6
- DBX109.6, 2-6
- DBX109.7, 2-7
- DBX56.1, 2-113
- DBX56.2, 2-113
- DBX56.4 - DBX56.7, 2-1

DB11, ...
- DBX0.0, 2-80
- DBX0.1, 2-80
- DBX0.2, 2-81
- DBX0.4, 2-81
- DBX0.5, 2-82
- DBX0.6, 2-82
- DBX0.7, 2-82
- DBX1.0, 2-83
- DBX1.1, 2-83
- DBX1.2, 2-84
- DBX1.6, 2-84
- DBX1.7, 2-85
- DBX4.0, 2-85
- DBX4.1, 2-86
- DBX4.2, 2-86
- DBX5.0, 2-86
- DBX5.1, 2-87
- DBX5.2, 2-87
- DBX6.0, 2-87
- DBX6.1, 2-88
- DBX6.2, 2-88
- DBX6.3, 2-88
- DBX6.7, 2-89
- DBX7.0, 2-89
- DBX7.1, 2-89
- DBX7.2, 2-90

DB19
- DBW2, 2-10
- DBW4, 2-10
- DBX0.0, 2-7
- DBX0.1, 2-8
- DBX0.2, 2-8
- DBX0.3, 2-9
- DBX0.4, 2-9
- DBX0.7, 2-9
- DBX10.0, 2-10
- DBX10.1, 2-11
- DBX10.2, 2-11
- DBX10.7, 2-11
- DBX12.2, 2-11
- DBX12.3, 2-12
- DBX12.4, 2-12
- DBX12.5, 2-12
- DBX12.6, 2-13
- DBX12.7, 2-13
- DBX13.5, 2-13
- DBX13.6, 2-14
- DBX13.7, 2-14
- DBX14.0 - DBX14.6, 2-14
- DBX14.7, 2-15
- DBX15.0 - DBX15.7, 2-15
- DBX16.0 - DBX16.6, 2-15
- DBX16.7, 2-16
- DBX17.0 - DBX17.7, 2-16
- DBX20.1, 2-17
- DBX20.3, 2-17
- DBX20.4, 2-18
- DBX20.6, 2-18
- DBX20.7, 2-18
- DBX22.0 - DBX22.7, 2-19
- DBX24.0, 2-19
- DBX24.1, 2-19
- DBX24.2, 2-19
- DBX24.3, 2-20
- DBX24.4, 2-20
- DBX24.5, 2-20
- DBX24.6, 2-21
- DBX24.7, 2-21

DBX25.0 - DBX25.7, 2-21
DBX26.1, 2-22
DBX26.2, 2-22
DBX26.3, 2-22
DBX26.5, 2-23
DBX26.6, 2-23
DBX26.7, 2-23
DBX27.0 - DBX27.7, 2-24
DBX40.0 - DBX40.7, 2-24
DBX41.0 - DBX41.7, 2-24
DBX42.0, 2-24
DBX44.0, 2-16
DBX45.0, 2-16
DBX45.1, 2-16
DBX45.2, 2-17
DBX45.3, 2-17
DBX6.0...7, 2-10
DBX7.0 - DBX7.7, 2-10
DBX8.0 - DBX8.7, 2-10
DB21, ...
DBB4, 2-134
DBB5, 2-136
DBX0.6, 2-133
DBX1.0, 2-115
DBX12.3, 2-139
DBX16.3, 2-139
DBX20.3, 2-139
DBX24.6, 2-140
DBX25.3, 2-140
DBX29.0 - DBX29.3, 2-141
DBX33.0, 2-116
DBX36.2, 2-116
DBX6.0, 2-138
DBX6.6, 2-138
DBX6.7, 2-139
DB21, ...
DBB118 - DBB119, 2-76
DBB129, 2-77
DBB140 - DBB157, 2-77
DBB158 - DBB193, 2-78
DBB194 - DBB206, 2-78
DBB208 - DBB271, 2-105
DBB376, 2-109
DBB58, 2-74
DBB60 - DBB64, 2-75
DBB60 - DBB65, 2-74
DBB66 - DBB67, 2-75
DBB68 - DBB97, 2-75
DBB98 - DBB115, 2-76
DBX0.4, 2-90
DBX0.5, 2-91
DBX1.1, 2-64
DBX1.6, 2-91
DBX1.7, 2-92
DBX10.0 - DBX11.1, 2-65
DBX2.0, 2-92
DBX272.0 - DBX273.1, 2-65
DBX274.0 - DBX275.1, 2-65
DBX276.0 - DBX277.1, 2-66
DBX278.0 - DBX279.1, 2-66
DBX30.5, 2-74
DBX31.0 - DBX31.2, 2-96
DBX31.4, 2-96
DBX318.0, 2-105
DBX318.1, 2-106
DBX318.5, 2-79
DBX318.7, 2-26
DBX319.0, 2-107
DBX319.1- DBX319.3, 2-107
DBX319.5, 2-109
DBX32.3, 2-97
DBX32.5, 2-97
DBX32.6, 2-98
DBX33.4, 2-98
DBX33.5, 2-99
DBX33.6, 2-100
DBX33.7, 2-100
DBX35.0, 2-101
DBX35.1, 2-101
DBX35.2, 2-102
DBX35.3, 2-102
DBX35.4, 2-102
DBX35.5, 2-103
DBX35.6, 2-103
DBX35.7, 2-103
DBX36.3, 2-69
DBX36.4, 2-104
DBX36.6, 2-25
DBX36.7, 2-26
DBX37.6, 2-104
DBX37.7, 2-104
DBX59.0 - DBX59.4, 2-75
DBX6.1, 2-93
DBX6.4, 2-93
DBX7.0, 2-93
DBX7.1, 2-94
DBX7.2, 2-94
DBX7.3, 2-95
DBX7.4, 2-95
DBX7.7, 2-96
DBX8.0 - DBX9.1, 2-64
DB21,...
DBX6.2, 2-25
DB31
DBX1.3, 2-27
DB31, ...
DBB0, 2-141
DBB19, 2-143, 2-144

DBB78 - DBB81, 2-147
DBB86 - DBB87, 2-132
DBB88 - DBB91, 2-132
DBX1.7, 2-145
DBX12.7, 2-117
DBX16.0 - DBX16.2, 2-121
DBX16.3, 2-120
DBX16.4 - DBX16.5, 2-120
DBX16.7, 2-119
DBX17.4 - DBX17.5, 2-122
DBX17.6, 2-122
DBX18.4, 2-124
DBX18.5, 2-123
DBX18.6 - DBX18.7, 2-122
DBX2.2, 2-119
DBX2.4 - DBX2.7, 2-117
DBX3.2 - DBX3.5, 2-143
DBX30.0, 2-126
DBX30.1, 2-126
DBX30.2, 2-125
DBX30.3, 2-125
DBX30.4, 2-125
DBX4.3, 2-146
DBX60.0, 2-126
DBX60.4, 2-118
DBX60.5, 2-118
DBX62.2, 2-147
DBX82.0 - DBX82.2, 2-128
DBX82.3, 2-127
DBX83.1, 2-130
DBX83.2, 2-129
DBX83.5, 2-129
DBX83.7, 2-128
DBX84.3, 2-131
DBX84.5, 2-131
DBX84.6, 2-131
DBX84.7, 2-130

DB31, ...
DBD 78, 2-79
DBD86, 2-79
DBD88, 2-79
DBX1.0, 2-41
DBX1.1, 2-71
DBX1.2, 2-72
DBX1.4, 2-29
DBX1.5 - DBX1.6, 2-31
DBX10.0, 2-110
DBX12.0 - DBX12.1, 2-68
DBX12.2 - DBX12.3, 2-68
DBX2.1, 2-33
DBX2.2, 2-34
DBX2.3, 2-67
DBX20.0, 2-36
DBX20.1, 2-36
DBX20.2, 2-37
DBX20.3, 2-37
DBX21.0 - DBX21.2, 2-38
DBX21.3 - DBX21.4, 2-39
DBX21.5, 2-40
DBX21.6, 2-40
DBX21.7, 2-41
DBX3.0, 2-112
DBX3.1, 2-72
DBX3.6, 2-67
DBX60.2 - DBX60.3, 2-69
DBX60.6, 2-70
DBX60.7, 2-70
DBX61.0, 2-42
DBX61.3, 2-42
DBX61.4, 2-43
DBX61.5, 2-43
DBX61.6, 2-44
DBX61.7, 2-45
DBX62.5, 2-73
DBX69.0 - DBX69.2, 2-45
DBX70.0, 2-110
DBX70.1, 2-111
DBX70.2, 2-111
DBX72.0, 2-111
DBX76.0, 2-46
DBX76.4, 2-112
DBX9.0 - DBX9.2, 2-35
DBX9.3, 2-36
DBX92.0, 2-46
DBX92.1, 2-46
DBX92.2, 2-47
DBX92.3, 2-47
DBX93.0 - DBX93.2, 2-48
DBX93.3 - DBX93.4, 2-49
DBX93.5, 2-49
DBX93.6, 2-50
DBX93.7, 2-51
DBX94.0, 2-53
DBX94.1, 2-55
DBX94.2, 2-55
DBX94.3, 2-57
DBX94.4, 2-59
DBX94.5, 2-59
DBX94.6, 2-60
DBX94.7, 2-61
DBX95.0, 2-63
DBX95.7, 2-63

I

Interface signals
611D Ready, 2-4
Acknowledge EMERGENCY STOP, 2-113
Acknowledge fixed stop reached, 2-71
Action block active, 2-97
Activate associated M1, 2-74
Activate channel-specific protection zone, 2-65
Activate dry run feed, 2-133
Activate fixed feed rate 1 - 4 for machine axes, 2-143
Activate fixed feed rate 1 - 4 for path/geometry axes, 2-141
Activate M01, 2-91
Activate program test, 2-92
Activate referencing, 2-115
Activate single block, 2-90
Activate travel to fixed stop, 2-73
Active, 2-22
Active drive parameter set A, B, C, 2-48
Active G function of groups 1 to 60, 2-105
Active JOG mode, 2-88
Active machine function REF, 2-90
Active machine function TEACH IN, 2-89
Active mode AUTOMATIC, 2-87
Active mode MDA, 2-88
Active motor A, B, 2-49
Active or passive file system, 2-15, 2-16
Active REPOS machine function, 2-89
Active spindle control mode, 2-130
Active spindle mode oscillation mode, 2-131
Active spindle positioning mode, 2-131
Actual direction of rotation clockwise, 2-128
Actual gear stage A to C, 2-121
Actual value in WCS, 2-9
Air temperature alarm, 2-6
Alarm area selection, 2-11
All axes stationary, 2-69
All axes that have to be referenced are referenced, 2-116
All channels in the reset state, 2-89
Analog spindle 1, utilization in percent, 2-10
Analog spindle 2, utilization in percent, 2-10
Associated M01/M00 active, 2-79
ASUB is stopped, 2-105
AT box ready, 2-2
AUTOMATIC mode, 2-80
Axis/spindle disable, 2-27
Axis/spindle stops, 2-43
Block search active, 2-98
Block search via program test is active, 2-106
Cancel alarm cleared, 2-17
Change gear stage, 2-127
Channel number (FC9: ChanNo), 2-24
Channel number of the machine control panel on the MMC, 2-10
Channel status active, 2-103
Channel status interrupted, 2-103
Channel status reset, 2-103
Channel-specific NCK alarm is active, 2-25
Channel-specific protection zone preactivated, 2-65
Channel-specific protection zone violated, 2-66
Clamping in progress, 2-67
Clear cancel alarms, 2-9
Clear recall alarms, 2-9
COM1, 2-12, 2-20
COM2, 2-11, 2-19
Controller enable, 2-33
Controller parameter set switchover (checkback signal), 2-45
Controller parameter set switchover (request), 2-35
Current controller active, 2-45
D function 1, 2-77
Darken screen, 2-8
Data transfer error, 2-24
Delete distance-to-go (axis-specific)/Spindle reset, 2-34
Delete distance-to-go (channel-specific), 2-25
Delete S value, 2-119
Disable parameter set switchover commands from NC, 2-36
Displayed channel number from the MMC, 2-19
Drive parameter set selection A, B, C, 2-38
Drive Ready, 2-49
Drive test travel enable, 2-41
Drive test travel request, 2-42
Dry run feed rate selected, 2-140
EMERGENCY STOP, 2-113
EMERGENCY STOP active, 2-114
Enable travel to fixed stop, 2-72
Encoder limit frequency exceeded, 2-69
Error, 2-19, 2-22
Error RS-232-C, 2-21
Extended address F functions 1 to 6, 2-78
Extended address H functions 1 to 3, 2-77
Extended address M functions 1 to 5, 2-75
Extended address S functions 1 to 3, 2-76
External zero offset, 2-112
F function for positioning axis, 2-147
FC9 Out: Active, 2-16
FC9 Out: Done, 2-16
FC9 Out: Error, 2-17
FC9 Out: StartError, 2-17
FC9: Start (measuring in Jog mode), 2-24
Feed disable, 2-138
Feed rate override, 2-134
Feed rate override (axis-specific), 2-141

Feed rate override active, 2-139
Feed rate override selected for rapid traverse, 2-140
Feed stop (Geometry axis 1 to 3), 2-139
Feed stop/spindle stop (axis-specific), 2-146
Fixed stop reached, 2-73
Follow-up mode active, 2-42
Follow-up operation, 2-29
Gear is changed over, 2-120
Hardware limit switches plus and minus, 2-68
Heatsink temperature prewarning, 2-55
Higraph first error display, 2-10
i2t monitoring, 2-63
Interface signals: Activate machine-related protection zone, 2-64
Interface signals: Enable protection zones, 2-64
Interrupt processing active, 2-104
Invert M3/M4, 2-122
JOG mode, 2-81
Key disable, 2-8
Key-operated switch position, 2-1
Last action block active, 2-98
Load, 2-23
Load part program, 2-14
Lubrication pulse, 2-46
M fct. 1-5 not included in list, 2-75
M function for spindle, 2-132
M(d) less than M(dx), 2-57
M, S, T, D, H, F functions Additional info "Quick" (quick acknowledgment), 2-75
M, S, T, D, H, F functions Modification, 2-74
M00/M01 active, 2-97
M02/M30 active, 2-99
Machine function REF, 2-84
Machine function REPOS, 2-83
Machine function TEACH IN, 2-83
Machine-related protection zone preactivated, 2-65
Machine-related protection zone violated, 2-66
MDA mode, 2-80
MMC alarm is active, 2-2
MMC CPU1 Ready, 2-4
MMC CPU2 Ready (to OPI or MPI), 2-4
MMC temperature limit, 2-2
MMC101/102 battery alarm, 2-3
Mode change disable, 2-16
Mode changeover inhibit, 2-81
Mode group number, 2-24
Mode group ready, 2-88
Mode group reset, 2-82
Mode group stop, 2-82
Mode group stop axes plus spindles, 2-82
Motor selection A, B, 2-39
Motor selection in progress, 2-40
Motor temperature pre-warning, 2-53
n(act) equals n(set), 2-60
n(act) less than n(min), 2-59
n(act) less than n(x), 2-59
NC Ready, 2-5
NC start, 2-94
NC start disable, 2-93
NC stop, 2-95
NC Stop at block limit, 2-94
NC Stop axes plus spindles, 2-95
NCK alarm is active, 2-5
NCK alarm with processing stop present, 2-26
NCK battery alarm, 2-7
NCK CPU Ready, 2-3
NCU 573 heatsink temperature alarm, 2-6
OK, 2-19, 2-22
Oscillation controlled by the PLC, 2-124
Oscillation speed, 2-123
Override active, 2-145
Overstore active:DB21 DBX318.7, 2-26
Path axis, 2-112
PLC action completed, 2-91
PLC index, 2-14, 2-15
PLC line offset, 2-15, 2-16
Position controller active, 2-43
Position measuring systems 1 and 2, 2-31
Position reached with exact stop coarse, 2-70
Position reached with exact stop fine, 2-70
PROG-EVENT-DISPLAY, 2-109
Program level abort, 2-93
Program status aborted, 2-102
Program status interrupted, 2-102
Program status running, 2-101
Program status stopped, 2-102
Program status wait, 2-101
Program test active, 2-100
Programming area selection, 2-10
Pulse enable, 2-41
Pulses enabled, 2-51
Ramp-function generator fast stop, 2-36
Ramp-function generator fast stop active, 2-46
Ramp-up completed, 2-55
Ramp-up times, 2-36
Rapid traverse override, 2-136
Rapid traverse override active, 2-138
Read-in disable, 2-93
Read-in enable is ignored, 2-104
Recall alarm cleared, 2-18
Reference point approach delay, 2-117
Reference point value 1 to 4, 2-117
Referenced/synchronized 1, 2-118
Referenced/synchronized 2, 2-118
Referencing active, 2-116
Repos DEFERAL Chan, 2-109
REPOS Delay Ackn, 2-111

REPOS offset, 2-110
REPOS offset valid, 2-111
Repos Path Mode Ackn0-2, 2-107
REPOSDELAY, 2-110, 2-111
REPOSMODEEDGE, 2-96
REPOSMODEEDGEACKN, 2-107
Reset, 2-96
Re-synchronize spindle when positioning 2 and 1, 2-122
Resynchronizing spindles 2 and 1, 2-120
Revolutional feed rate active, 2-147
Rigid tapping is active, 2-131
RMB, 2-96
RME, 2-96
RMI, 2-96
RMN, 2-96
RMNOTDEF, 2-96
RS-232-C external, 2-12, 2-20
RS-232-C off, 2-13, 2-21
RS-232-C on, 2-13, 2-21
RS-232-C stop, 2-20
S function for spindle, 2-132
Screen bright, 2-7
Screen is dark, 2-17
Second software limit switch plus or minus, 2-68
Select gear stage, 2-125
Selected JOG mode, 2-86
Selected machine function REF, 2-87
Selected mode AUTOMATIC, 2-85
Selected mode MDA, 2-86
Selected mode TEACH IN, 2-86, 2-87
Selection, 2-14, 2-23
Sensor for fixed stop, 2-72
Set gear stage A to C, 2-128
Setpoint direction of rotation, counter-clockwise/setpoint direction of rotation, clockwise, 2-122
Setpoint speed increased, 2-129
Setpoint speed limited, 2-130
Setting-up mode active, 2-46
ShopMill control signal, 2-11
Simulation selected, 2-18
Single block type A, 2-85
Single block type B, 2-84
Skip block, 2-92
Speed controller active, 2-44
Speed controller integrator disable, 2-40
Speed controller integrator disabled, 2-50
Speed limit exceeded, 2-130
Speed setpoint smoothing, 2-37
Speed setpoint smoothing active, 2-47
Spindle in set range, 2-129
Spindle override, 2-143, 2-144
Spindle positioning, 2-125
Spindle reset/Delete distance-to-go, 2-119
Spindle start clockwise rotation, 2-126
Spindle start, counter-clockwise, 2-125
Spindle stop, 2-126
Spindle/no axis, 2-126
Stop at the end of block with SBL is suppressed, 2-104
Switch over MCS/WCS, 2-18
Synchronized Actions (S5, H2)|Signals for dynamic M functions: M0 - M99, 2-78
T function 1, 2-76
Tool offset selection, 2-11
Torque limit 2, 2-37
Torque limit 2 active, 2-47
Transformation active, 2-100
Unload, 2-13, 2-23
Value of F help function, 2-79
Value of M help function, 2-79
Value of S help function, 2-79
Variable signaling function, 2-61
VDClink lower than warning threshold, 2-63
Velocity/spindle speed limitation, 2-67

List of Abbreviations

A

AC	Adaptive control
ACKNLG	Acknowledge from printer
ADF	Autofeed printer
ASCII	American Standard Code for Information Interchange
ASUB	Asynchronous subroutine
AuxF	Auxiliary Function
BA	Operating mode
BB	Ready
BCD	Binary-Coded Decimals: Decimal numbers coded in binary format
BCS	Basic Coordinate System
BIN	BINary files
BOT	BoOT files: Boot files for SIMODRIVE 611D
BP	Basic Program
BUSY	Busy from printer
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CC	Compile Cycle
CNC	Computerized Numerical Control
COR	COordinate Rotation
CPA	Compiler Projecting Data: Compiler configuring data
CPU	Central Processing Unit
CR	Carriage Return
CRC	Cutter Radius Compensation
CTS	Clear To Send: Signal from serial data interfaces
CUTCOM	CUTter radius COMpensation: Tool radius compensation
DATA	Printer data bit x (x is index from 0 to 7)
DAU	Digital-to-Analog Converter
DB	Data Block in the PLC
DBB	Data Block Byte in the PLC
DBW	Data Block Word in the PLC
DBX	Data block bit in the PLC
DC	Direct Control: Movement of the rotary axis across the shortest path to the absolute position within one revolution
DCD	Data Carrier Detect
DCE	Data Communication Equipment

DIN	Deutsche Industrie Norm (German Industry Standard)
DIO	Data Input/Output: Data transfer display
DIR	DIRectory Directory
DITE	DIsplacement Thread End
DITS	DIsplacement Thread Start
DOE	Data Communication Equipment
DPM	Dual-Port Memory
DPR	Dual-Port RAM
DRF	Differential Resolver Function: (handwheel)
DRY	Dry run feedrate
DSB	Decoding Single Block
DSC	Dynamic Servo Control / Dynamic Stiffness Control
DSR	Data Set Ready
DTE	Data Terminal Equipment
DTR	Data Terminal Ready
EIA code	Special punched tape code, number of holes per character always odd
ENC	ENCoder: Actual-value sensor
EPROM	Erasable Programmable Read-Only Memory
ERROR	ERROR from printer
ESR	Extended Stop and Retract Function
FB	Function Block
FC	Function Call: Function block on the PLC
FD	Feed Drive
FIFO	First In First Out: memory, which works without address specification and whose data are read in the order in which they were stored.
FIPO	Fine InterPOlator
FOC	Force Control
FST	Feed STop
GEO	Geometry
GIA	Gear Interpolation dAta
GND	Signal GrouND
GS	Gear Step
GSC	Gear Step Change
GUD	Global User Data
GWPS	Grinding wheel peripheral speed
HASH	A software method for mapping a large number of names onto a finite memory area
HEX	Abbreviation for hexadecimal number
HMI	Human Machine Interface: Operator function of SINUMERIK for operation, programming and simulation. HMI means the same as MMC.
HW	Hardware
I/R	Infeed/regenerative feedback unit of SIMODRIVE 611(D)
IC (GD)	Implicit Communication (Global Data)
ICA	Interpolatory Interpolatory compensation

IF	Pulse enable of drive module
INC	Increment Increment
INI	INItializing data
INIT	INITialize printer
IPO	Interpolator
IS	Interface Signal
ISO code	Special punched tape code, number of holes per character always even
JOG	JOGging: Setup mode
K_{UE}	Transmission ratio
K_v	Servo gain factor
LEC	Leadscrew error compensation
LED	Light Emitting Diode: Diode
LF	Line Feed
LR	Position controller
LUD	Local User Data
MB	MegaByte
MCP	Machine control panel
MCS	Machine Coordinate System
MD	Machine Data
MDI	Manual Data Automatic: Manual input
MLFB	Machine-readable product designation
Mode group	Mode group
MPF	Main Program File: NC part program (main program)
MPI	MultiPort Interface
MSD	Main spindle drive
NC	Numerical Control: Control
NCK	Numerical Control Kernel: NC kernel with block preparation, travel range, etc.
NCU	Numerical Control Unit: NCK hardware unit
NRK	Numeric Robotic Kernel: name of NCK operating system
OB	Organization Block in the PLC
OEM	Original Equipment Manufacturer
OP	Operator Panel: Operating setup
OPI	Operator Panel Interface: Interface for connection to the operator panel
OPT	OPTions Options
PC	Personal Computer
PCU	PC Unit: PC box (computer unit)
PG	Programming device
PLC	Programmable Logic Controller
PMS	Position Measuring System
POS	POSitioning
PRAL	Process interrupt
PRT	Program test
PTP	Point To Point

PUD	Program global User Data: Global program variable
RAM	Random Access Memory: program memory that can be read and written to
REF	REFerence point approach function
REPOS	REPOSition function
RI	Ring Indicator
RISC	Reduced Instruction Set Computer: Type of processor with small instruction set and ability to process instructions at high speed
ROV	Rapid OVerride
RPA	R Parameter Active: Memory area in the NCK for R parameter numbers
RPY	Roll Pitch Yaw: Type of rotation of a coordinate system
RS-232-C	Serial interface (definition of the exchange lines between DTE and DCE)
RTLI	Rapid Traverse Linear Interpolation: Linear interpolation during rapid traverse motion
RTS	Request To Send: RTS, control signal of serial data interfaces
RxD	Receive Data
SBL	Single BLock
SD	Setting Data
SEA	Setting data active: Identifier (file type) for setting data
SERUPRO	SEarch–RUn by PROgram test
SKP	SKiP: Skip block
SLCT	SeLeCT from printer
SM	Stepper Motor
SPF	Sub Program File
SSL	Block search
STROBE	Data strobe to printer
SW	Software
SYF	SYstem Files
SYNACT	SYNchronized ACTion
T	Tool
TC	Tool Change
TCP	Tool Center Point: Tool tip
TEA	Testing Data Active: Identifier for machine data
TLC	Tool Length Compensation
TM	Tool management
TNRC	Tool Nose Radius Compensation
TO	Tool Offset
TOA	Tool Offset Active: Identifier (file type) for tool offsets
TRANSMIT	TRANSform Milling Into Turning: Coordinate conversion on turning machines for milling operations
TRC	Tool Radius Compensation
TxD	Transmit Data
UFR	User FRame: Zero offset
V.28	Definition of the electrical behavior of the signals

VDI	Communication interface between NC and PLC
WCS	Workpiece Coordinate System
WOP	Workshop-Oriented Programming
WPD	WorkPiece Directory
WZ	Tool
X_y	Connector designation (y is index)
ZO	Zero offset
ZOA	Zero Offset Active: Identifier (file type) for zero offset data

Glossary

Absolute dimension

Specifies the target of an axis motion in terms of a dimension relative to the zero point of the currently valid coordinate system. See → Incremental dimension

Acceleration with jerk limitation

To achieve optimal machine acceleration behavior while protecting the mechanical system, the machining program can switch between abrupt acceleration and continuous (jerk-free) acceleration.

Address

An address is the identifier for a particular operand or operand area, 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. The display is broken down into alarms and messages.

1. Alarms and messages in the part program

   Alarms and messages can be displayed directly from the part program in plain text.

2. Alarms and messages from the PLC

   Alarms and messages of the machine can be displayed from the PLC program in plain text. Additional function block packages are not required for this.

Archiving

Reading out of files and/or directories on an **external** memory device.

Asynchronous subroutine

Part program that can be started asynchronously to (independently of) the current program status using an interrupt signal (e.g., "Rapid NC input" signal).

Automatic

Operating mode of control (block sequence mode per 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

CNC axes are classified based on their function as follows:

- Axes: interpolating path axes
- Auxiliary axes: non-interpolating infeed and positioning axes with axis-specific feed. Auxiliary axes are not involved in actual machining, e.g., tool feeder, tool magazine.

Axis address

See → Axis identifier

Axis identifier

Axes are identifed using X, Y, and Z as defined in DIN 66217 for a right-handed, right-angled → coordinate system.

→ Rotary axes rotating around X, Y, and Z are identified using A, B, and C. Additional axes situated parallel to the specified axes can be designated using other letters.

Axis name

See → Axis identifier

Backlash compensation

Compensation for a mechanical machine backlash, e.g., backlash on reversal for ball screws. The backlash compensation can be entered separately for each axis.

Backup battery

A backup battery ensures that the → user program is stored retentively in the → CPU along with specified data areas and bit memory, timers, and counters.

Basic axis

Axis whose setpoint or actual value is used to calculate a compensation value.

Basic coordinate system

Cartesian coordinate system that is mapped to the machine coordinate system through a transformation.

The programmer uses axis names of the basic coordinate system in the → part program. The basic coordinate system exists in 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 transmission (bits/s).

Block

All files that are required for program creation and execution are designated as blocks.

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 of system program after a Power On.

C-axis

Axis around which a controlled rotary motion and positioning occurs with the workpiece spindle.

Channel

A channel is characterized by the fact that it can process a → part program independently of other channels. A channel has exclusive control over 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 feed rate, and the workpiece is thereby machined.

CNC

See → NC

CNC high-level language

The high-level language offers: → user-defined variables, → system variables, → macro techniques.

COM

Component of NC control for implementing and coordinating communication.

Compensation axis

Axis whose setpoint or actual value is modified using the compensation value.

Compensation table

Table of interpolation points. It supplies the compensation values of the compensation axis for selected positions of the basic axis.

Compensation value

Difference between the axis position measured by the encoder and the desired, programmed axis position.

Connecting cable

Connecting cables are pre-assembled or user-assembled 2-wire cables with a connector at each end. This connecting cable connects the → CPU to a → programming device or to other CPUs by means of a → multi-point interface (MPI).

Continuous-path mode

The objective of continuous-path mode is to avoid substantial deceleration of the → path axes at the part program block boundaries and to change to the next block at as close to the same path velocity as possible.

Contour

Contour of the → workpiece

Contour monitoring

The following error is monitored within a definable tolerance band as a measure of contour accuracy. An unacceptably high following error can cause the drive to become overloaded, for example. In this case, an alarm occurs and the axes are stopped.

Coordinate system

See → Machine coordinate system, → Workpiece coordinate system

CPU

Central processing unit, see → Programmable logic control

C-spline

C-spline is the best known and most frequently used spline. The transitions at the interpolation points are tangentially continuous and have constant curvature. A third-degree polynomial is used.

Cycle

Protected subroutine for implementing a repetitious machining operation on the → workpiece.

Data block

1. Data unit of the → PLC that → HIGHSTEP programs can access.
2. Data unit of the → NC: Data blocks contain the data definitions for global user data. The data can be installed directly in the definition.

Data word

Two-byte data unit within a → data block.

Diagnostics

1. Control operating area
2. The control has both a self-diagnostics program and testing aids for the service: status, alarm, and service displays

DRF

Differential resolver function: NC function that generates an incremental work offset in conjunction with an electronic handwheel in automatic mode.

Drive

The SINUMERIK 840D control system is digitally connected to the SIMODRIVE 611 converter system over a high-speed digital parallel bus.

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 enables programs, texts, and program blocks to be created, modified, expanded, moved, and inserted.

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 limit, the control behaves as though it has reached an endpoint exactly. A block advance of the → part program occurs.

External work offset

Work offset specified by the → PLC.

Fast retraction from contour

When an interrupt occurs, a motion can be initiated via the CNC machining program, enabling the tool to be quickly retracted from the workpiece contour that is currently being machined. The retraction angle and the distance retracted can also be assigned. After fast retraction, an interrupt routine can also be executed (SINUMERIK 840D).

Feed override

The programmed velocity is overriden by the current velocity setting made via the → machine control panel or from the → PLC (0 to 200%). The feed velocity can also be offset by applying a programmable percentage factor (1 to 200%) in the machining program.

Finished-part contour

Contour of the finished workpiece. See → Raw part.

Fixed-point approach

Machine tools can approach fixed points such as the tool change point, loading point, pallet change point, etc., in a defined manner. The coordinates of these points are stored in the control. The control moves the relevant axes in → rapid traverse, whenever possible.

Frame

A frame represents a calculation rule that transfers one Cartesian coordinates sytem to another Cartesian coordinate system. A frame contains the following components: → work offset, → rotation, → scaling, → mirroring.

Geometry

Description of a → workpiece in the → workpiece coordinate system.

Geometry axis

Geometry axes are used to describe a 2- or 3-dimensional area in the workpiece coordinate system.

Ground

Ground refers to all connected, inactive parts of equipment that cannot assume dangerous touch voltage even in the event of a fault.

Helical interpolation

Helical interpolation is especially suitable for easy machining inside or outside threads with form cutters and for milling lubricating grooves.

The helix consists of two motions:

- A circular movement in one plane
- A linear movement perpendicular to this plane

High-speed digital inputs/outputs

Digital inputs can be used to start high-speed CNC program routines (interrupt routines), for example. The digital CNC outputs can be used to trigger fast, program-controlled switching functions (SINUMERIK 840D).

HIGHSTEP

Summary of programming options for → PLCs of the AS300/AS400 system.

I/O 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

Identifier

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subroutines, key words, and words with multiple address letters. These additions are equivalent to the meaning of the words in the block format. Identifiers must be unique. The same identifier must not be used for different objects.

Inch system

Measuring system that 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 "inclined-surface machining" function.

Increment

Traversed distance information via the 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

Specifies a motion target of an axis in terms of the distance and direction to be traversed relative to a previously reached point. 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 that the control reads ahead can be set via system parameters.

Interpolator

Logic unit of the → NCK that defines intermediate values for the motions to be carried out in individual axes based on information on the end positions specified in the part program.

Interpolatory compensation

Interpolatory compensation is a tool that enables manufacturing-related leadscrew error and measuring system error compensations.

Interrupt routine

Interrupt routines are special → subroutines that can be started by events (external signals) in the machining process. The part program block that is being executed is aborted, and the interruption point of the axes is stored automatically.

Inverse time feedrate

With SINUMERIK 840D, the time required for the path of a block to be traversed can be programmed for the axis motion instead of the feed velocity (G93).

JOG

Control mode (setup mode): In JOG mode, the machine can be set up. Individual axes and spindles can be moved in JOG mode using 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 4 positions that are assigned functions by the operating system of the control. The key switch has three different colored keys that can be removed in the specified positions.

Keywords

Words with specified notation that have a defined meaning in the programming language for → part programs.

Leading axis

The leading axis is the → gantry axis that exists from the point of view of the operator and programmer and, thus, can be influenced like a standard NC axis.

Leadscrew error compensation

Compensation for the mechanical inaccuracies of a leadscrew participating in the feed. The control uses stored deviation values for the compensation.

Limit speed

Maximum/minimum (spindle) speed: The maximum speed of a spindle can be limited by specifying machine data, the → PLC or → setting data.

Linear axis

In contrast to a rotary axis, a linear axis describes a straight line.

Linear interpolation

The tool is moved on a straight line to the end point, and the workpiece is thereby machined.

Load memory

The load memory is the same as → RAM for the CPU 314 of the → PLC.

Look Ahead

The **Look Ahead** function is used to achieve an optimal machining speed by looking ahead over an assignable number of traversing blocks.

Machine axes

Axes that exist physically in the machine tool.

Machine control panel

Operator panel of the machine tool containing keys, knobs, etc. and simple display elements, such as LEDs. It is used for immediate control of the machine tool via the PLC.

Machine coordinate system

Coordinate system that is relative to the axes of the machine tool.

Machine fixed-point

Point that is uniquely defined by the machine tool, e.g., machine reference point.

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. In this case, a CNC channel must be regarded as a separate CNC control with decoding, block preparation, and interpolation.

Macro technique

Summary of a set of instructions under one identifier. The identifier represents the set of consolidated instructions in the program.

Main block

A block preceded with ":" that contains all information to start the operating sequence in a → part program.

Main program

The → part program designated by a number or an identifer in which additional main programs, subroutines, or → cycles can be called.

MDA

Control mode: Manual Data Automatic. In MDA mode, individual program blocks or block sequences can be entered without reference to a main program or subroutine and then executed immediately using 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. The display is broken down into alarms and messages.

Metric and inch dimensions

Position and lead values can be programmed in inches in the machining program. Irrespective of the programmable dimensions (G70/G71), the controller is set to a basic system.

Metric measuring system

Standardized system of units: For length, e.g., mm (millimeters), m (meters).

Mirroring

Mirroring inverts the signs of the coordinate values of a contour with respect to an axis. It is possible to mirror in relation to more than one axis at a time.

Mode group

Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a 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 840D controls would be: Computerized Numerical Control

NCK

Numerical control kernel: Component of NC that executes the → part programs and basically coordinates the motion operations for the machine tool.

Network

A network is the connection of multiple S7-300s and other end devices, e.g., a programming device via a → connecting cable. Data are exchanged between the connected devices over the network.

NRK

Numeric robotic kernel (operating system of → NCK)

NURBS

The motion control and path interpolation that occurs within the control is performed based on NURBS (**N**on **U**niform **R**ational **B**-**S**plines). As a result, a uniform process is available within the control for all interpolations for SINUMERIK 840D.

OEM

For machine manufacturers that want to create their own user interface or introduce technology-specific functions in the control, freedom is provided for individual solutions (OEM applications) for SINUMERIK 840D.

Offset memory

Data area in the control where the tool offset data are stored.

Operating mode

Sequential concept for operation of a SINUMERIK control. The following modes are defined: → Jog, → MDA, → Automatic.

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: In the case of machining interruptions (e.g., in case of tool breakage), the tool can be retracted a defined distance with a specifiable orientation via a program command.

Overall reset

In the event of an overall reset, the following memories of the → CPU are deleted:

- → RAM
- Read/write area of → load memory
- → System memory
- → Backup memory

Override

Manual or programmable intervention option that allows the operator to override programmed feeds or speeds in order to adapt them to a particular workpiece or material.

Part program

Series of instructions 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 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 assigned a name comprising up to 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 endpoint simultaneously.

Path feed

Path feed is applicable to → path axes. It represents the geometric sum of the feed rates of the → geometry axes involved.

Path velocity

The maximum programmable path velocity is dependent on the input resolution. For example, for 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.

PLC

Programmable **L**ogic **C**ontrol: Component of → NC: Programmable controller for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D: The PLC user program and the user data are stored together with the PLC basic program 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 that defines the position of a point on a plane in terms of its distance from the zero point and the angle formed by the radius vector with a defined axis.

Polynominal interpolation

Polynomial interpolation enables a wide variety of curve characteristics to be generated, such as **straight line, parabolic, exponential functions** (SINUMERIK 840D).

Positioning axis

Axis which performs an auxiliary movement on a machine tool (e.g. tool magazine, pallet transport). Positioning axes are axes that do not interpolate using → path axes.

Pre-coincidence

Block change occurs already when the path distance approaches an amount equal to a specifiable delta of the end position.

Program block

Program blocks contain the main program and subroutines of → part programs.

Programmable frames

Programmable → frames enable dynamic definition of new coordinate system output points while the part program is being executed. A distinction is made between absolute definition using a new frame and additive definition with reference to an existing starting point.

Programmable Logic Controller

Programmable logic controllers (PLC) are electronic controls, the function of which is stored as a program in the control unit. This means that the layout and wiring of the device do not depend on the function of the control. The programmable logic controller has the same structure as a computer; it consists of a CPU (central module) with memory, input/output modules and an internal bus system. The peripherals and the programming language are matched to the requirements of the control technology.

Programmable working area limitation

Limitation of the motion space of the tool to a space defined by programmed limitations.

Programming key

Character and character strings that have a defined meaning in the programming language for → part programs.

Protection zone

Three-dimensional zone within the → working area into which the tool tip must not pass.

Quadrant error compensation

Contour errors at quadrant transitions, which arise as a result of changing friction conditions on the guideways, can be largely eliminated using quadrant error compensation. Quadrant error compensation is parameterized by a circularity test.

RAM

RAM is a work memory in the → CPU that the processor accesses when processing the user program.

Rapid traverse

Maximum 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.

Raw part

Workpiece as it is before it is machined.

Reference point

Machine tool position that the measuring system of the → machine axes references.

Rigid tapping

This function enables rigid tapping. Using interpolating motion of the spindle as a rotary axis and the drilling axis, threads are cut exactly to the final drilling depth, e.g., blind hole threads (requirement: axis mode of spindle).

Rotary axis

Rotary axes rotate a workpiece or tool 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".

R parameters

Arithmetic parameter that can be set or queried by the programmer of the → part program for any purpose in the program.

Safety functions

The control is equipped with permanently active montoring functions that detect faults in the → CNC, the → PLC, and the machine in a timely manner so that damage to the workpiece, tool, or machine is largely prevented. In the event of a malfunction, the machining sequence is interrupted and the drives are stopped and the cause of the malfunction is saved and displayed as an alarm. At the same time, the PLC is informed that a CNC alarm is pending.

Scaling

Component of a → frame that implements axis-specific scale modifications.

Serial RS-232-C interface

For data input/output, the PCU 20 has one serial V.24 interface (RS232) while the PCU 50/70 has two V.24 interfaces. Machining programs and manufacturer and user data can be loaded and saved via these interfaces.

Setting data

Data that communicate properties of the machine tool to the NC control in a manner defined by the system software.

Softkey

A key whose name appears on an area of the screen. The selection of keys displayed is adapted dynamically 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 available for frequently recurring machining tasks.

- for drilling/milling technology
- for turning technology

In the "Program" operating area, the available cycles are listed under the "Cycle Support" menu. After selecting the desired machining cycle, the required parameters for the value assignment are displayed in plain text.

Subblock

Block preceded by "N" containing information for a sequence, e.g., positional data.

Subroutine

Sequence of statements of a → part program that can be called repeatedly with different defining parameters. The subroutine is called from a main program. It is not possible to block every subroutine against unauthorized reading and displaying. → Cycles are a form of subroutines.

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 as 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. The synchronous axis is "not available" from the perspective of the operator and the programmer.

System memory

The system memory is a memory in the CPU in which the following data are stored:

- Data required by the operating system
- The operands times, 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 variable name preceded by the character $. See → User-defined variable.

Text editor

See → Editor

TOA area

The TOA area includes all tool and magazine data. By default, this area coincides with the → channel area with regard to the reach of the data. However, machine data can be used to specify that multiple channels share one → TOA unit so that common tool management data are 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 utilized tool must be communicated to the control which then takes it into account. The curvature center is maintained equidistantly around the contour offset by the radius of curvature.

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 work offset of an axis.

Traversing range

The maximum permissible traversing range for linear axes is ± 9 decades. The absolute value depends on the selected input and positioning resolutions and the basic unit system used (inches or metric).

User-defined variable

Users can declare their own variables for any purpose in the → part program or data block (global user data). The data type and variable name are specified in a definition. See → System variable.

User interface

The user interface is the display medium of a CNC control in the form of a screen. It features horizontal and vertical softkeys.

User memory

All programs and data such as part programs, subroutines, comments, tool offsets, work offsets/frames, and channel and program user data can be stored in the shared CNC user memory.

User program

User programs for S7-300 automation systems are created with the STEP 7 programming language. The user program has a modular structure consisting 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.

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.

Working area

Three-dimensional area in which the tool tip can enter based on the design of the machine tool. See → Protection zone.

Working area limitation

The working area limitation can be used in addition to limit switches to limit the traversing range of axes. A pair of values can be used for each axis to describe the protected working area.

Work offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a → frame.

1. Settable

   SINUMERIK 840D: A configurable number of settable work offsets are available for each CNC axis. Alternatively, offsets that can be selected using G functions are active.

2. External

   In addition to all offsets that specify the position of the workpiece zero, an external work offset can be overridden by a handwheel (DRF offset) or by the PLC.

3. Programmable

   Work offsets can be programmed for all path and positioning axes using the TRANS instruction.

Workpiece

Part to be created/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. 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.

To
Siemens AG

A&D MC BMS
Postfach 3180

91050 ERLANGEN, GERMANY

Tel.: +49 (0) 180 / 5050 – 222 [Hotline]

Fax: +49 (0) 9131 / 98 – 63315 [Documentation]

E-mail: mailto:motioncontrol.docu@siemens.com

Suggestions

Corrections

For Publication/Manual:

SINUMERIK
840D sl/840D/840Di/810D

Function Manual
Basic Machine

Manufacturer/Service Documentation

From

Name

Company/dept.

Street: _____

Zip code: _____ Town: _____

Phone: _____ / _____

Fax: _____ / _____

Operator's Guide

Order no.: 6FC5397-0BP10-0BA0

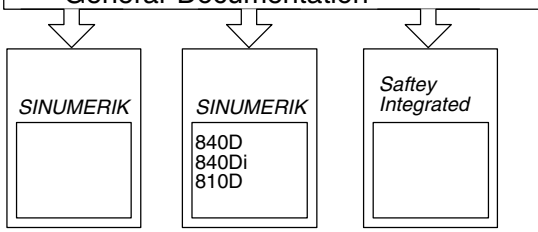
08/2005 Edition

Should you come across any printing errors when reading this publication, please notify us on this sheet. Suggestions for improvement are also welcome.

Suggestions and/or corrections

Overview of SINUMERIK 840D/840Di/810D Documentation (08/2005)

General Documentation

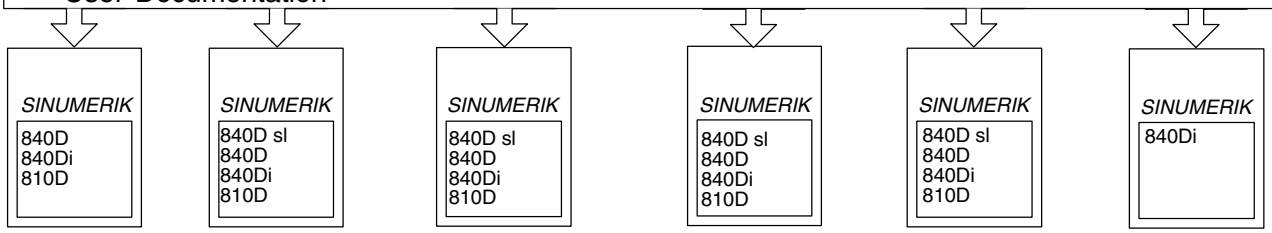


Brochure

Catalog NC 61 ***)**

Saftey Integrated
Application Manual

User Documentation



Operator's Guide
– **HMI Embedded *)**
– **ShopMill**
– **ShopTurn**
– HT6

Operator's Guide
– **HMI Advanced *)**
– Programming compact

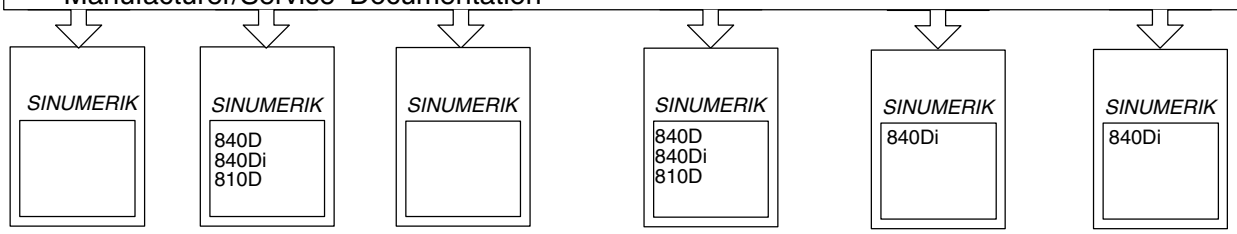
Programming Guide
– Fundamentals ***)**
– Advanced ***)**
– Programming
– Lists System Variables
– ISO Turning/Milling

Programming Guide
– **Cycles**
– **Measuring Cycles**

Diagnostics Guide *)

System Overview

Manufacturer/Service Documentation



Configuring (HW) ***)**
– 840D
– 810D

Equipment Manual
Operator Components *)

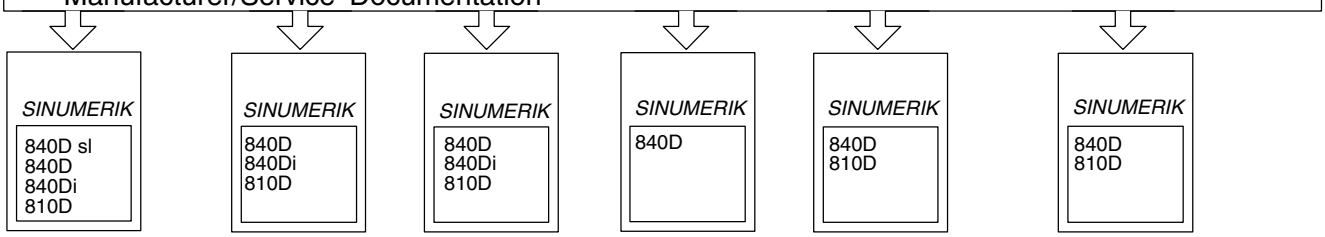
Installation and Start-up Guide *)
– 840D
– 810D
– HMI

Description of Functions
– **ShopMill**
– **ShopTurn**

Installation and Start-up Guide

List Manual *)
– Part 1
– Part 2

Manufacturer/Service Documentation



Description of Functions
– Basic Machine ***)**
– Extended Functions
– Special Functions
– **Synchronized Actions**
– **Iso Dialects**
– **EMC Guidlines**

Description of Functions
– **Drive Functins**
– **Tool Management**
– **Hydraulics Module**
– **Analog Module**

Description of Functions
– **Remote Diagnosis**
– **@Event**

Description of Functions
Saftey Integrated

Configuring
– **Operator Interface OP030**
– **HMI Embedded**

MCIS
– Computer Link
– Tool Data Inform. Sys.
– NC Data Management
– NC Data Transfer
– Tool Data Communication

Electronic Documentation



DOCONCD ***)**
DOCONWEB

***)** These documents are a minimum requirement

Overview of SINUMERIK 840D sl Documentation (08/2005)

General Documentation

- *SINUMERIK* — Brochure
- *SINUMERIK* 840D sl — Catalog NC 61 ***)**
- *SINAMICS* S120 — Catalog D21.2 Servo Control ***)**

User Documentation

- *SINUMERIK* 840D sl — Operator's Guide
  - **HMI Embedded *)**
  - **ShopMill**
  - **ShopTurn**
- *SINUMERIK* 840D sl 840D 840Di 810D — Operator's Guide
  - **HMI Advanced *)**
  - Programming compact
- *SINUMERIK* 840D sl 840D 840Di 810D — **Programming Guide**
  - Fundamentals ***)**
  - Advanced ***)**
  - Programming
  - Lists System Variables
  - ISO Turning/Milling
- *SINUMERIK* 840D sl 840D 840Di 810D — Programming Guide
  - **Cycles**
  - **Measuring Cycles**
- *SINUMERIK* 840D sl 840D 840Di 810D — **Diagnostics Guide *)**

Manufacturer/Service Documentation

- *SINUMERIK* 840D sl — Equipment Manual **NCU *)**
- *SINUMERIK* 840D sl — Equipment Manual **Operator Components *)**
- *SINUMERIK* 840D sl — **Start-up Guide** CNC ***)**
  - Part 1 NCK, PLC, Drive
  - Part 2 HMI
  - Part 3 ShopMill
  - Part 4 ShopTurn
  - Part 5 Basic Software
- *SINUMERIK* 840D sl 840D 840Di 810D — **List Manual ***
  - Part 1
  - Part 2

Manufacturer/Service Documentation

- *SINUMERIK* 840D sl 840D 840Di 810D — **Description of Functions**
  - Basic Machine ***)**
  - Extended Functions
  - Special Functions
- *SINUMERIK* 840D sl 840D 840Di 810D — Description of Functions
  - **Synchronized Actions**
  - **Iso Dialects**
- *SINAMICS* S120 — Description of Functions **Drive Functions**
- *SINUMERIK* 840D sl — Description of Functions **Safety Integrated**
- *SINUMERIK* 840D sl 840D 840Di 810D — **EMC Guidlines**

Electronic Documentation

- *SINUMERIK SINAMICS Motors* — DOCONCD ***)** DOCONWEB

***)** These documents are a minimum requirement