SIEMENS

SINUMERIK

SINUMERIK 840D sl / 828D Basic Functions

Function Manual

Valid for

Control
SINUMERIK 840D sl / 840DE sl / 828D

CNC software   Version 4.8 SP3

08/2018
6FC5397-0BP40-6BA2

Legal information

Warning notice system

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

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

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

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

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Document order number: 6FC5397-0BP40-6BA2
℗ 08/2018 Subject to change

Preface

SINUMERIK documentation

The SINUMERIK documentation is organized into the following categories:

- General documentation/catalogs
- User documentation
- Manufacturer/service documentation

Additional information

You can find information on the following topics at the following address (https://support.industry.siemens.com/cs/de/en/view/108464614):

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

If you have any questions regarding the technical documentation (e.g. suggestions, corrections), please send an e-mail to the following address (mailto:docu.motioncontrol@siemens.com).

mySupport/Documentation

At the following address (https://support.industry.siemens.com/My/ww/en/documentation), you can find information on how to create your own individual documentation based on Siemens' content, and adapt it for your own machine documentation.

Training

At the following address (http://www.siemens.com/sitrain), you can find information about SITRAIN (Siemens training on products, systems and solutions for automation and drives).

FAQs

You can find Frequently Asked Questions in the Service&Support pages under Product Support (https://support.industry.siemens.com/cs/de/en/ps/faq).

SINUMERIK

You can find information about SINUMERIK at the following address (http://www.siemens.com/sinumerik).

Target group

This publication is intended for:

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

Benefits

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

Standard version

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

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

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

Technical Support

Country-specific telephone numbers for technical support are provided in the Internet at the following address (https://support.industry.siemens.com/sc/ww/en/sc/2090) in the "Contact" area.

Information on the structure and contents

Structure

This Function Manual is structured as follows:

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

- Appendix with:
  - List of abbreviations
  - Documentation overview
- Index of terms

Note

For detailed descriptions of data and alarms see:

- For machine and setting data:
  Detailed machine data description
- For NC/PLC interface signals:
  NC Variables and Interface Signals List Manual
- For alarms:
  Diagnostics Manual

Notation of system data

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

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

Note

Signal address

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

Quantity structure

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

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

Data types

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

Type	Meaning	Value range
INT	Signed integers	-2,147,483,648 ... +2,147,483,647
REAL	Numbers with decimal point	≈ ±5.0*10^{-324} … ≈ ±1.7*10^{+308}
BOOL	Boolean values	TRUE (≠0) , FALSE (0)
CHAR	ASCII characters and bytes	0 ... 255 or -128 ... 127
STRING	Character string, null-terminated	Maximum of 400 characters + /0 (no special characters)
AXIS	Axis names	All axis names available in the control system
FRAME	Geometrical parameters for moving, rotating, scaling, and mirroring	---

Arrays

Arrays can only be formed from similar elementary data types. Up to 3-dimensional arrays are possible.

Example: `DEF INT ARRAY[2, 3, 4]`

Number systems

The following number systems are available:

- Decimal: `DEF INT number = 1234` or `DEF REAL number = 1234.56`
- Hexadecimal: `DEF INT number = 'H123ABC'`
- Binary: `DEF INT number = 'B10001010010'`

Querying REAL variables

We recommend that querying REAL or DOUBLE variables in NC programs and synchronized actions is programmed as limit value evaluation.

Example: Querying the actual value of an axis for a specific value

```
Program code                                                  Comment
DEF REAL AXPOS = 123.456
IF ($VA_IM[<axis>] - 1ex-6) <= AXPOS <= ($VA_IM[<axis>] + 1ex-6)   ; actual position
   ...                                                        == AXPOS
```

Program code	Comment
ELSE	
...	<> AXPOS
ENDIF	

Table of contents

1 Fundamental safety instructions

1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

If the safety instructions and residual risks in the associated hardware documentation are not observed, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

1.2 Warranty and liability for application examples

Application examples are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. Application examples do not represent specific customer solutions, but are only intended to provide support for typical tasks.

As the user you yourself are responsible for ensuring that the products described are operated correctly. Application examples do not relieve you of your responsibility for safe handling when using, installing, operating and maintaining the equipment.

1.3 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit:

Industrial security (http://www.siemens.com/industrialsecurity)

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at:

Industrial security (http://www.siemens.com/industrialsecurity)

Further information is provided on the Internet:

Industrial Security Configuration Manual (https://support.industry.siemens.com/cs/ww/en/view/108862708)

⚠ WARNING

Unsafe operating states resulting from software manipulation

Software manipulations (e.g. viruses, trojans, malware or worms) can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a holistic, state-of-the-art industrial security concept for the installation or machine.
- Make sure that you include all installed products into the holistic industrial security concept.
- Protect files stored on exchangeable storage media from malicious software by with suitable protection measures, e.g. virus scanners.
- Protect the drive against unauthorized changes by activating the "know-how protection" drive function.

A2: Various NC/PLC interface signals and functions 2

2.1 Brief description

Contents

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

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)

2.2 NC/PLC interface signals

2.2.1 General information

NC/PLC interface

The NC/PLC interface comprises the following parts:

- Data interface
- Function interface

Data interface

The data interface is used for component coordination:

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

Data exchange is organized by the basic PLC program.

Cyclic signal exchange

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

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

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

PLC to NC:

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

NC to PLC:

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

Channel-specific signals (DB21, ...)

PLC to NC:

- Control signal "Delete distance-to-go"

NC to PLC:

- NC status signals (NC alarm active)

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

PLC to NC:

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

NC to PLC:

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

Function interface

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

References

- Description of the basic PLC program:
  → Function Manual, Basic Functions, Basic PLC Program (P3)
- Description of the event-driven signal exchange (auxiliary and G commands):
  → Function Manual, Basic Functions; Auxiliary Function Output to PLC (H2)
- Overview of all interface signals, functional and data components:
  → List Manual 2

2.2.2 Ready signal to PLC

DB10 DBX104.7 (NC-CPU ready)

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

DB10 DBX108.3 (operating software ready)

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

DB10 DBX108.5 (drives in cyclic operation)

Precondition: For **all** machine axes of the NC, the associated drives are in the cyclic operation. That is, they cyclically exchange PROFIdrive telegrams with the NC.

DB10 DBX108.6 (drive ready)

Precondition: For **all** machine axes of the NC, readiness of the associated drives and also the third-party drives is signaled via PROFIBUS:

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

DB10 DBX108.7 (NC ready)

The NC is ready.

2.2.3 Status signals to PLC

DB10 DBX103.0 (remote diagnosis active)

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

DB10 DBX109.6 (air temperature alarm)

The ambient temperature or fan monitoring function has responded.

DB10 DBX109.7 (NC 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 result in loss of data.

DB10 DBX109.0 (NC alarm pending)

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

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

The NC sends this signal to the PLC to indicate that at least one NC alarm is pending for the affected channel.

DB21, ... DBX36.7 (NC alarm with machining stop pending)

The NC sends this signal to the PLC to indicate that at least one NC alarm that has interrupted or aborted the current part program processing is active for the relevant channel.

DB21, ... DBX39.1 (NC alarm with program stop)

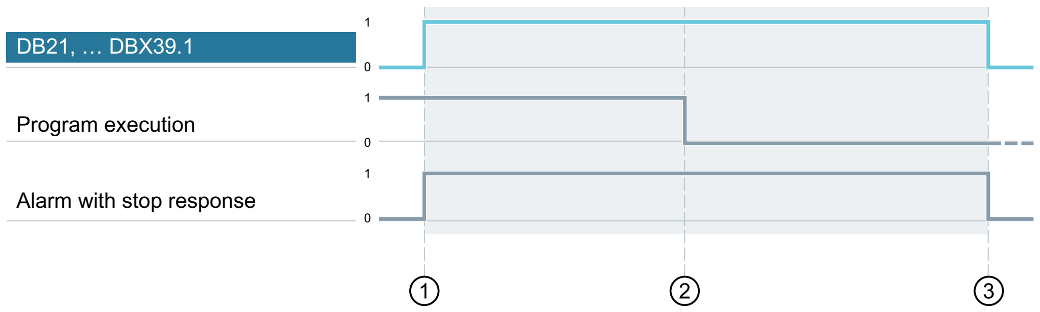
With interface signal DB21, ... DBX39.1 (NC alarm with program stop), the NC signals to the PLC that at least one NC alarm is pending for the affected channel that is blocking the program from advancing.

The signal is reset as soon as the alarm responses that activated the signal are no longer active. This depends on the reset conditions of the alarms.

Signal chart of alarms with stop response

Interface signal DB21, ... DBX39.1 is always set when an alarm with a stop response is generated. No distinction is made between the various stop responses:

- Stop on path, reset signal "NC ready" (DB10 DBX108.7).
- Stop all axes of the mode group, reset signal "Mode group ready" (DB11 DBX6.3)
- Immediate stop on the path.
- Interpreter is stopped, content of IPO buffer is processed.
- Stop at the end of the block.



① Alarm with stop response is generated.
NC/PLC interface signal DB21, ... DBX39.1 (NC alarm with program stop) is set for the affected channel.
② The alarm stops program execution (machining stops).
③ When the alarm is reset, interface signal DB21, ... DBX39.1 is also reset.

Figure 2-1 Signal chart of alarms with stop response

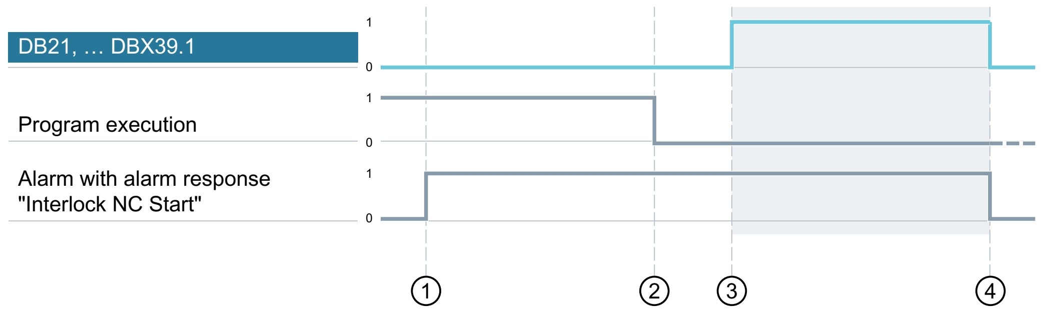
Signal chart of alarms with alarm response "NC Start disable"

Alarms with alarm response "NC Start disable" are also taken into account.

Alarms with this response prevent execution of the following sequences:

- Start of a part program/teach in start/continuation of a jog movement with the "NC START" key.
- Start of a part program from the part program of another channel.
- Start of an ASUB from the "Reset" state (in JOG or AUTOMATIC mode).
- Start of an ASUB from the "interrupted" state in JOG mode.
- Cascaded search after a successfully completed search
- Start of a search from the reset state (search type 1,2,4)
- Start of a search from the "interrupted" state (search type 1,2,4)
- Start of SERUPRO (search type 5)
- Start of a simulation search (search type 102)

If one of these sequences is aborted in response to a pending alarm with alarm response "NC Start disable", interface signal DB21.DBX39.1 is set.



① Alarm with alarm response "NC start disable" is generated.

② The NC program executed to the end.

③ NC start / ASUB start / Search is triggered. Because the alarm has not yet been cleared, NC/PLC interface signal DB21, ... DBX39.1 (NC alarm with program stop) is set for the affected channel.

When the alarm is reset, interface signal DB21, ... DBX39.1 is also reset.

Figure 2-2 Signal chart of alarms with alarm response "NC Start disable"

Note

Allow ASUB start

With machine data MD20194 $MC_IGNORE_NONCSTART_ASUP, interrupt inputs can be parameterized such that an assigned ASUB is even started if an alarm with alarm response "NC Start disable" is pending (see Chapter "Perform ASUB start for user alarms (Page 603)"). In this case, interface signal DB21, ... DBX39.1 (alarm with program stop) is **not** set.

2.2.4 Signals to/from the operator panel front

DB19 DBX0.0 (brighten screen)

The screen blanking is disabled.

DB19 DBX0.1 (darken screen)

The operator panel screen is darkened.

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

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

Note

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

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

Screen darkening via keyboard / automatic screen saver

If no keys are pressed on the operator panel front within the assigned time (default = 3 minutes):
MD9006 $MM_DISPLAY_SWITCH_OFF_INTERVAL (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 does not generate an operator action.

Parameterization:

- DB19 DBX0.1 = 0
- MD9006 $MM_DISPLAY_SWITCH_OFF_INTERVAL > 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 / MCS)

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

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

DB19 DBB13 (control of the file transfer)

Order byte for controlling the program selection of PLC. The orders relate to the program specified via the following interface signals:

- DB19 DBX16.0-6 (program selection from the PLC: index of the program list)
- DB19 DBB17 (program selection from the PLC: program index in the program list)

DB19 DBB16 (program selection from the PLC: index of the program list)

Control byte for specifying the program list.

DB19 DBB17 (program selection from the PLC: program index in the program list)

Control byte for specifying the program number.

DB19 DBB26 (program selection from the PLC: status signals)

Status byte for the actual data transfer status.

DB19 DBB27 (program selection from the PLC: error identification)

Output byte for the error values for data transfer.

References

For more information about the program selection of the PLC, see:

- Program selection (Page 933) for SINUMERIK 840D sl
- Program selection (Page 1141) for SINUMERIK 828D

2.2.5 Signals to channel

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

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

2.2.6 Signals to axis/spindle

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

NOTICE
Specifications for the drive test 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.

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

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

The motion is carried out once the motion is enabled:

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

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

Stationary axis

If the interface signal is set for a stationary axis, all travel request are ignored as of this time.

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

Traversing axis

If the interface signal is set for a traversing axis, the axis is stopped using the currently active brake characteristic. If the interface signal is reset while the travel request is still present, traversing is continued.

Spindle

- Open-loop control mode: Speed setpoint zero is output immediately
- Positioning mode: See "Stationary axis" / "Traversing axis" above

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

The interface signal is only effective with the DB31, ... DBX2.1 (controller enable) interface signal

DB31, ... DBX2.1	DB31, ... DBX1.4	Mode
1	Ineffective	In closed-loop control
0	1	Follow-up
0	0	Hold

Follow-up

In follow-up mode, the set position of the machine axis is continuously corrected to the actual position (set position = actual position).

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

During follow-up mode, clamping or standstill monitoring are **not active**.

Note

If the DB31, ... DBX2.1 (controller enable) interface signal is set during follow-up mode, repositioning to the last programmed position is performed with `REPOSA` (approach on a straight line with all axes) when the NC program is active. In all other cases, all subsequent motions start at the actual position.

Hold

In "Hold" mode, the set position of the machine axis does not follow the actual position. If the machine axis is moved away from the set position, the difference between the set position and the actual position increases constantly (following error). The following error is suddenly corrected if the controller enable is set without complying with the axial acceleration characteristic (speed jump).

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

During "hold", clamping or standstill monitoring are **active**.

Note

The following error is suddenly corrected if the controller enable is set without complying with the axial acceleration characteristic (speed jump).

Application example

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

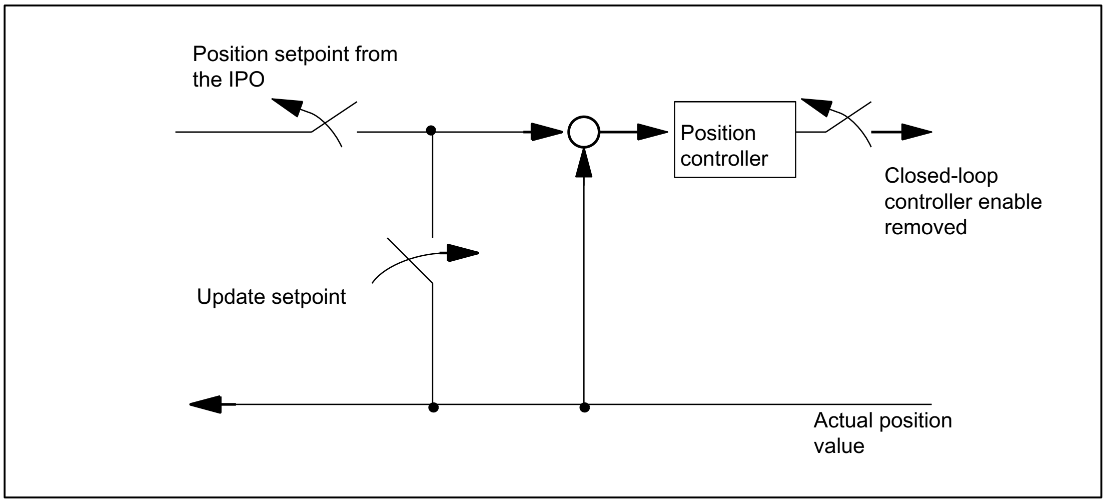

Position setpoint from the IPO
Position controller
Closed-loop controller enable removed
Update setpoint
Actual position value

Figure 2-3 Effect of controller enable and follow-up mode

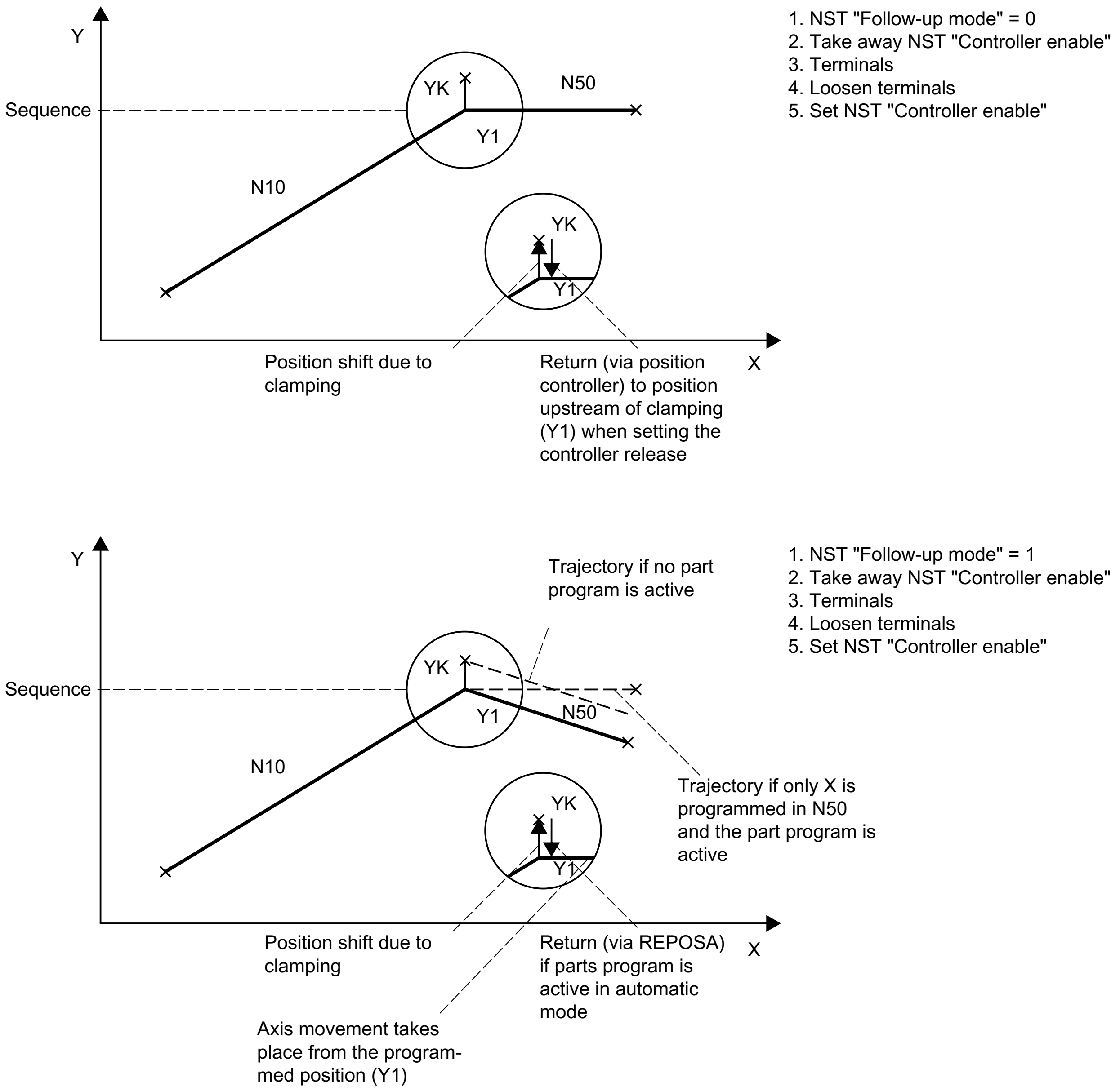

Y
Sequence
YK
N50
Y1
N10
YK
Y1
X
Position shift due to clamping
Return (via position controller) to position upstream of clamping (Y1) when setting the controller release
1. NST "Follow-up mode" = 0
2. Take away NST "Controller enable"
3. Terminals
4. Loosen terminals
5. Set NST "Controller enable"
Y
Trajectory if no part program is active
Sequence
YK
Y1
N50
N10
YK
Y1
X
Trajectory if only X is programmed in N50 and the part program is active
Position shift due to clamping
Return (via REPOSA) if parts program is active in automatic mode
Axis movement takes place from the programmed position (Y1)
1. NST "Follow-up mode" = 1
2. Take away NST "Controller enable"
3. Terminals
4. Loosen terminals
5. Set NST "Controller enable"

Figure 2-4 Trajectory for clamping and "hold"

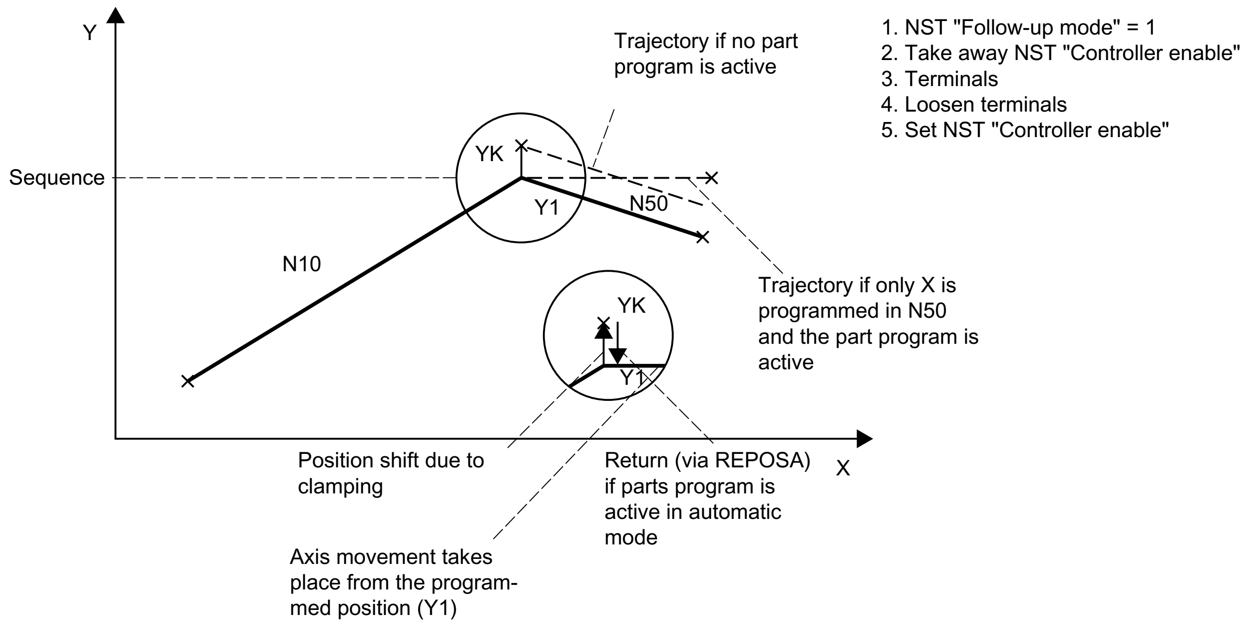

Y
Sequence
Trajectory if no part program is active
1. NST "Follow-up mode" = 1
2. Take away NST "Controller enable"
3. Terminals
4. Loosen terminals
5. Set NST "Controller enable"
YK
Y1
N50
N10
Trajectory if only X is programmed in N50 and the part program is active
YK
Y1
Position shift due to clamping
Return (via REPOSA) if parts program is active in automatic mode
X
Axis movement takes place from the programmed position (Y1)

Figure 2-5 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, referencing is not required.

The following procedure is recommended:

1. Activate follow-up mode:
   DB31, ... DBX2.1 = 0 (controller enable)
   DB31, ... DBX1.4 = 1 (follow-up mode) (in the same or preceding OB1 cycle)
   → The axis/spindle is operating in follow-up mode
2. Deactivate external controller enable and external speed setpoint
   → Axis/spindle moves with external setpoint
   → NC continues to detect the actual position and corrects the set position to the actual position
3. Deactivate external controller enable and cancel external speed setpoint
   → Axis/spindle stops
4. Cancelling follow-up mode
   DB31, ... DBX2.1 = 1 (controller enable)
   DB31, ... DBX1.4 = 0 (follow-up mode)
   → NC synchronizes to actual position. The next traversing motion begins at this position.

Note

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

Cancelling follow-up mode

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

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

Note

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

Monitoring

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

- Standstill monitoring
- Clamping monitoring
- Positioning monitoring

Effects on other interface signals:

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

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

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

- Indirect motor measuring system
- Direct measuring system on load

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

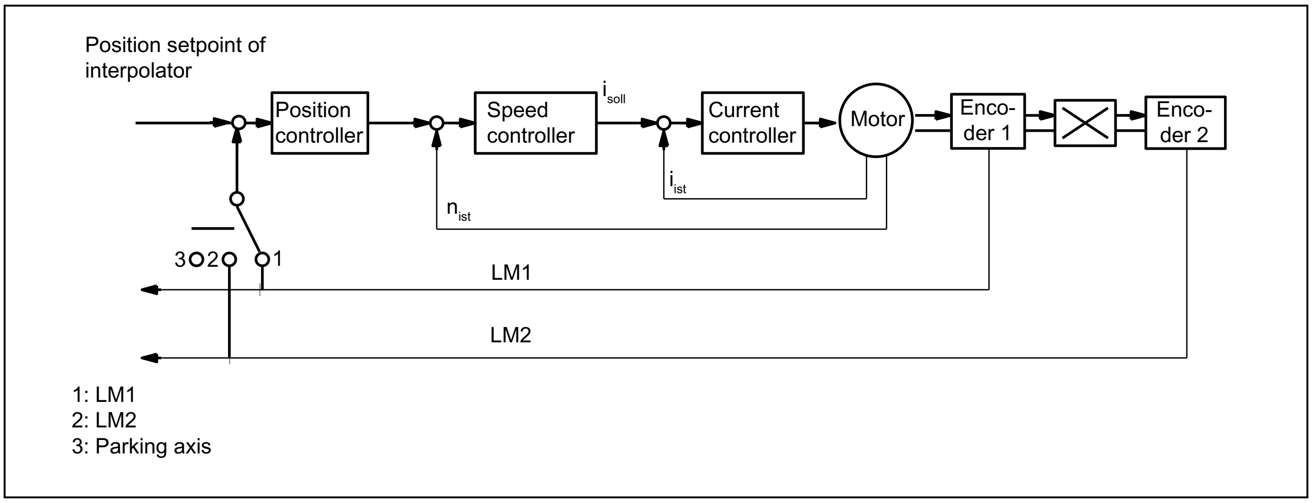


Figure 2-6 Position measuring systems 1 and 2

Functionality of the "Position measuring system 1 / 2" interface signals in conjunction with the "Controller enable":

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

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

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

DB31, ... DBX2.1 == 1

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

DB31, ... DBX2.1 == 0

Activation methods

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

- NC/PLC interface signal:
  - DB31, ... DBX2.1 (controller enable)
  - DB31, ... DBX21.7 (pulse enable)
  - DB31, ... DBX93.5 (drive ready)
  - DB10, DBX56.1 (emergency stop)
- NC-internal
  Alarms that trigger cancellation of the controller enable on the machine axes. The alarm responses are described in:
  References:
  Diagnostics Manual

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

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

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

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

Alarm: "21612 Controller enable reset during motion"

- The machine axis is decelerated taking into account the parameterized duration of the braking ramp for error states with a fast stop (speed setpoint = 0):
  MD36610 $MA_AX_EMERGENCY_STOP_TIME (max. duration of the braking ramp in the event of errors)
  An alarm is displayed:
  Alarm: "21612 Controller enable reset during motion"

  Note

  The controller enable is cancelled at the latest when the cut-off delay expires:

  MD36610 $MA_AX_EMERGENCY_STOP_TIME

- The machine axis position control loop opens. Feedback via interface signal:
  DB31, ... DBX61.5 == 0 (position controller active).
  The time for the parameterized cut-off delay of the controller enable is started by the machine data:
  MD36620 $MA_SERVO_DISABLE_DELAY_TIME (OFF delay of the controller enable)
- As soon as the actual speed has reached the zero speed range, the drive controller enable is cancelled. Feedback via interface signal:
  DB31, ... DBX61.6 == 0 (speed controller active)
- The actual position value of the machine axis continues to be acquired by the controller.
- At the end of the braking operation, the machine axis is switched to follow-up mode, regardless of the corresponding NC/PLC interface signal. Standstill and clamping monitoring are not effective. See the description above for the interface signal:
  DB31, ... DBX1.4 (follow-up mode).

Synchronizing the actual value (reference point approach)

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

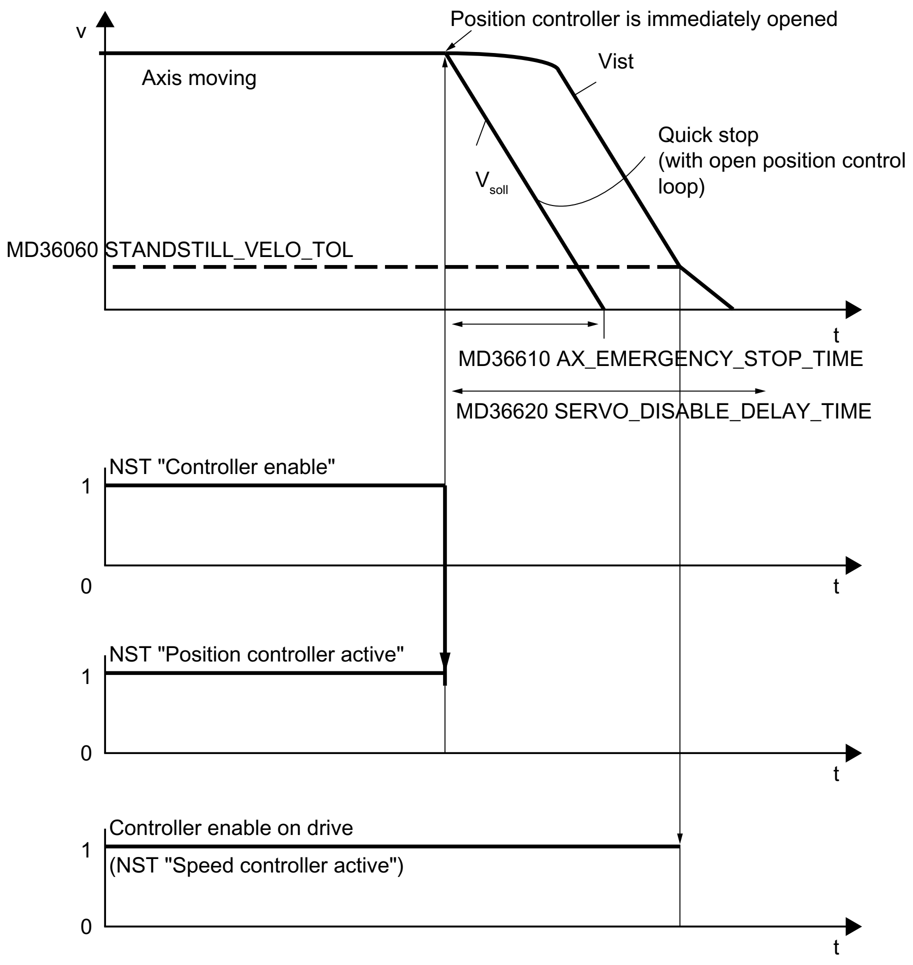

v
Position controller is immediately opened
Vist
Axis moving
Quick stop
(with open position control loop)
V soll
MD36060 STANDSTILL_VELO_TOL
t
MD36610 AX_EMERGENCY_STOP_TIME
MD36620 SERVO_DISABLE_DELAY_TIME
NST "Controller enable"
1
0
t
NST "Position controller active"
1
0
t
Controller enable on drive
(NST "Speed controller active")
1
0
t

Figure 2-7 Cancelling the controller enable when the machine axis is in motion

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

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

Spindle reset

For a detailed description of the spindle reset, see Section "S1: Spindles (Page 1273)".

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

Request for activation of the specified controller parameter set.

Controller parameter set	DBX9.2	DBX9.1	DBX9.0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1

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

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

MD35590 $MA_PARAMSET_CHANGE_ENABLE = 1 or 2

For detailed information on the parameter set changeover, see Section "Parameter set selection during gear step change (Page 1329)".

Parameter set changeover when machine axis is in motion

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

MD32200 $MA_POSCTRL_GAIN (servo gain factor)

- "Identical servo gain factors (K_V)" or "position control not active":
  The NC responds immediately to the parameter set changeover. The parameter set is also changed during the motion.
- "Unequal servo gain factors (K_V)" or "position control not 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 stationary)
  MD36060 $MA_STANDSTILL_VELO_TOL (threshold velocity/speed "axis/spindle stationary")

Parameter set changeover from the part program

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

For detailed information on the auxiliary function output, see Section "H2: Auxiliary function outputs to PLC (Page 401)".

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

If the interface signal is set, a request for a parameter set changeover (DB31, ... DBX9.0 / 9.1 / 9.2) is ignored.

2.2.7 Signals from axis/spindle

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

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

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

The motion is carried out once the motion is enabled:

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

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

The machine axis is in follow-up mode.

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

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

- No new setpoints are to be output **AND**
- The actual speed of the machine axis is lower than the parameterized zero speed: MD36060 $MA_STANDSTILL_VELO_TOL (threshold velocity axis stationary)

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

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

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

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

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

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

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

Active parameter set Coding accordingly:

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

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

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

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

MD33050 $MA_LUBRICATION_DIST (distance for lubrication by PLC)

2.2.8 Signals to axis/spindle (digital drives)

DB31, ... DBX21.0 - 4 (motor/drive data set: selection)

The PLC issues a request to the drive to switch over to a new motor and/or drive data set.

The interface can be flexibly parameterized using: DB31, ...DBX130.0 - 4 (see Chapter "Switching over motor/drive data sets (Page 74)")

The active motor and/or drive data set is indicated using: DB31, ... DBX93.0 - 4

Note

Main spindle drive

For a main spindle drive, only motor data sets 1 and 2 are valid when switching over between a star and delta connection.

- Motor data set 1: Star
- Motor data set 2: Delta

DB31, ... DBX21.5 (motor has been selected)

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

DB31, ... DBX21.6 (request integrator disable, speed controller)

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

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

Feedback signal via: DB31, ... DBX93.6 = 1 (integrator disable, speed controller)

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

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

- Trigger equipment enable
- Controller and pulse enable
- Pulse enable (safe operating stop)
- Stored hardware input

- Setpoint enable
- "Ready to run state"
  - No drive alarm (DClink1 error)
  - DC link connected
  - Ramp-up completed

Feedback signal via: DB31, ... DBX93.7 (pulses enabled)

2.2.9 Signals from axis/spindle (digital drives)

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

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

DB31, ... DBX93.0 - 4 (motor/drive data set: display)

The drive signals back the active motor and drive data set to the PLC.

The interface can be flexibly parameterized using: DB31, ...DBX130.0 - 4 (see Chapter "Switching over motor/drive data sets (Page 74)")

The request to switch over motor and/or drive data set is realized using: DB31, ... DBX21.0 - 4

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 disable, speed controller)

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

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

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

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

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

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

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

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

Note

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

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

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

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

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

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

References:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.3 Functions

2.3.1 Screen refresh behavior for overload - only 840D sl

There are part programs, where the main run must wait until the preprocessing run provides new blocks.

Preprocessing run and display refresh compete for NC processor time.

The following machine data can be used to set how the NC should behave if the preprocessing run is too slow:

MD10131 $MN_SUPPRESS_SCREEN_REFRESH

Value	Meaning
0	If the preprocessing run of a channel is too slow, the display is not refreshed in all channels.
1	If the preprocessing run of a channel is too slow, the display is only refreshed in the time critical channels in order to obtain processor time for the preprocessing run.
2	Display refresh is not suppressed.

2.3.2 Settings for involute interpolation - only 840D sl

Introduction

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

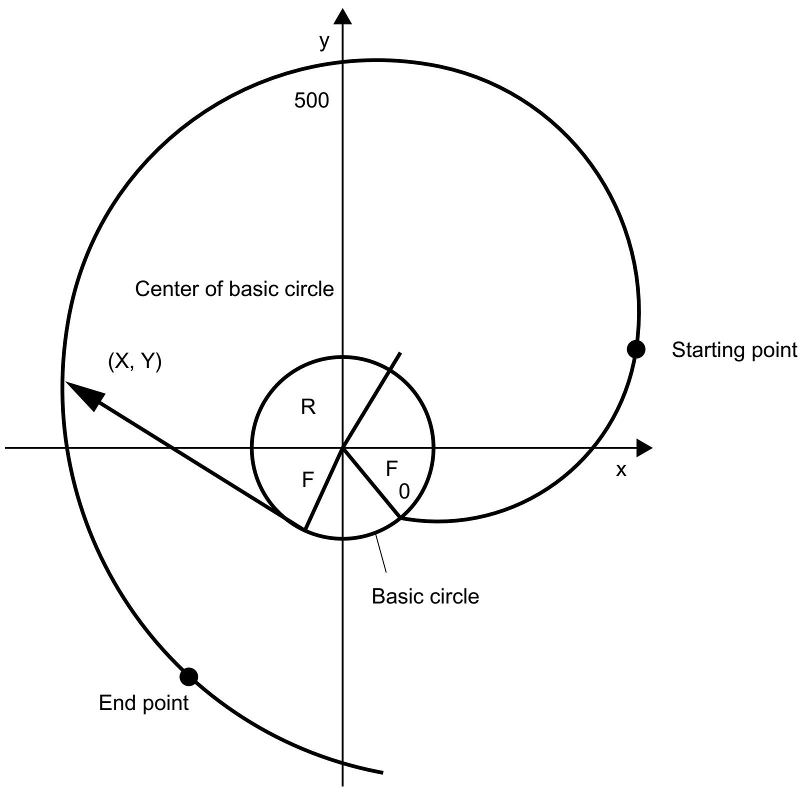


Figure 2-8 Involute (unwound from base circle)

Programming

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

References:
Programming Manual, Fundamentals

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

Accuracy

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

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

MD21015 $MC_INVOLUTE_RADIUS_DELTA (end point monitoring for involute)

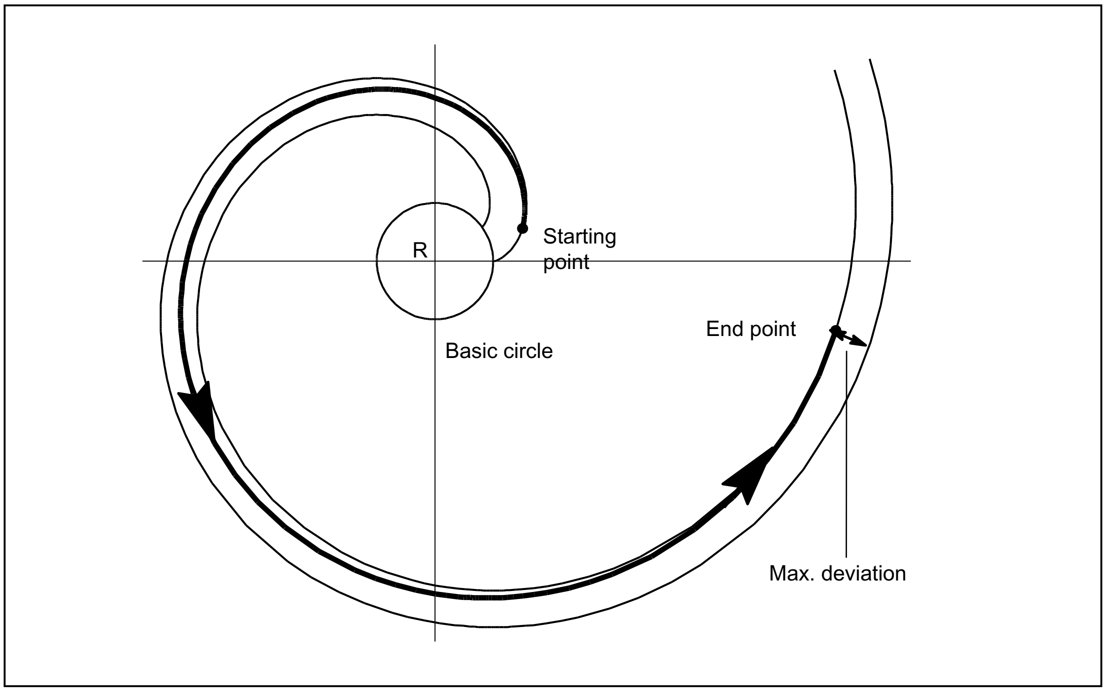


Figure 2-9 MD21015 specifies the max. permissible deviation

Limit angle

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

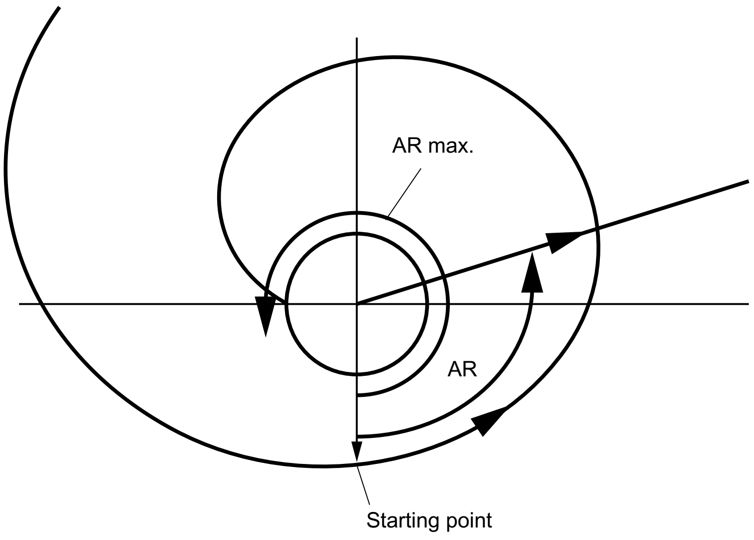


Figure 2-10 Limited angle of rotation towards base circle

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

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

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

Tool radius compensation

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

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

Dynamic response

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

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

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

2.3.3 Activate DEFAULT memory - only 840D sl

GUD start values

The DEF... / REDEF... NC commands can be used to assign default settings to global user data (GUD). To make these default values available at the parameterized initialization time, e.g. with the attribute `INIPO`, after power on, they must be saved permanently in the system. The required memory space must be enabled using the following machine data:

MD18150 $MM_GUD_VALUES_MEM (non-volatile memory space for GUD values)

References:

- Function Manual, Extended Functions; S7: "Memory Configuration"
- Programming Manual, Job Planning

2.3.4 Read and write PLC variable - only 840D sl

High-speed data channel

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

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

Organization of memory area

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

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

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

Access from NC

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

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

Access from PLC

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

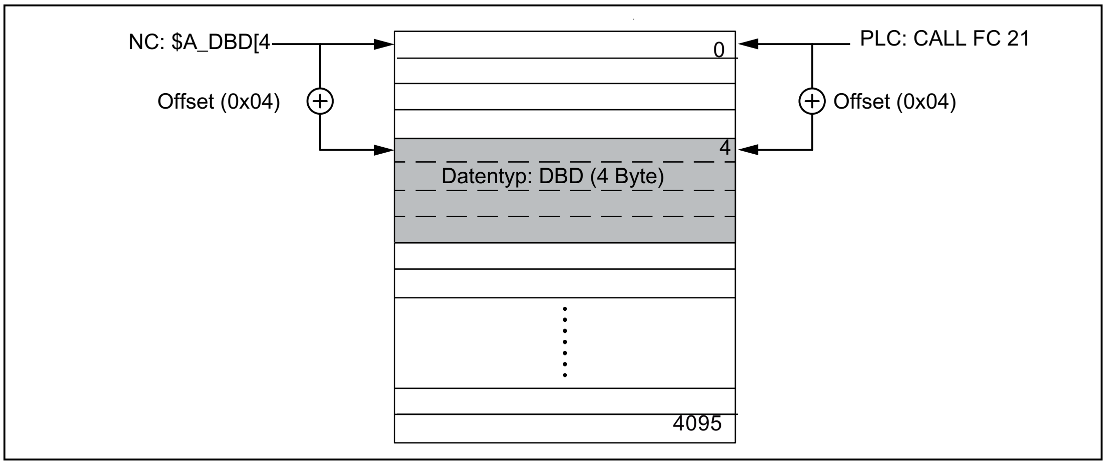


Figure 2-11 Communications buffer (DPR) for NC/PLC communication

Supplementary conditions

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

- The PLC stores data in 'Little Endian' format in the DPR.
- Values transferred with $A_DBR are subject to data conversion and hence to loss of accuracy. The data format for floating-point numbers is DOUBLE (64 bits) in the NC, but only FLOAT (32 bits) in the PLC. The format used for storage in the dual-port RAM is FLOAT. Conversion takes place respectively before/after storage in the dual-port RAM. If a read/write access is made from the NC to a variable in the dual-port RAM, the conversion is performed twice. It is impossible to prevent differences between read and written values because the data is stored in both formats.
  Example
  Bypassing the problem by means of comparison on "EPSILON" (minor deviation)

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

Activation

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

Example

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

The position offset within the NC input (PLC output area) should be the 4th byte. The position offset must be a whole-number multiple of the data width.

Writing from PLC:

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

Program code	Comment
`ErrCode :=MW12);`	
`. . .`	
`)`	

Reading in part program

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

Behavior during POWER ON, block search

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

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

Other status transitions have no effect in this respect.

References

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

SINUMERIK 840D sl: Section "Function (Page 1088)"

2.3.5 Access protection via password and keyswitch

Access authorization

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

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

Multi-level security concept

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

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

Access features

- Protection level 0 provides the greatest number of access rights, protection level 7 the least.
- If certain access rights are granted to a protection level, these protection rights automatically apply to any higher protection levels.
- Conversely, protection rights for a certain protection level can only be altered from a higher protection level.
- Access rights for protection levels 0 to 3 are permanently assigned by Siemens and cannot be altered (default).
- Access rights can be set by querying the current keyswitch positions and comparing the passwords entered. When a password is entered it overwrites the access rights of the keyswitch position.

- Options can be protected on each protection level. However, option data can only be entered in protection levels 0 and 1.
- Access rights for protection levels 4 to 7 are only suggestions and can be altered by the machine tool manufacturer or end user.

2.3.5.1 Password

Set password

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

Operating area "Commissioning" > "Password" > "Set password"

Delete password

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

Operating area "Commissioning" > "Password" > "Delete password"

Change password

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

Operating area "Commissioning" > "Password" > "Change password"

Note

Warm restart

Access rights and password status (set/deleted) are not affected by a warm restart!

New password

A password may contain up to eight characters. We recommend that you confine yourself to the characters available on the operator panel front when selecting the password.

Where a password consists of less than eight characters, the additional characters are interpreted as blanks.

When assigning a new password, the rules for assigning secure passwords must be taken into account.

Note

Assigning secure passwords

Observe the following rules when assigning new passwords:

- When assigning new passwords, make sure that you do not assign passwords that can be easily guessed, e.g. simple words, key combinations that can be easily guessed, etc.
- Passwords must always contain a combination of upper-case and lower-case letters as well as numbers and special characters. Passwords must comprise at least eight characters. PINS must comprise an arbitrary sequence of digits.
- Wherever possible and where it is supported by the IT systems, a password must always have a character sequence as complex as possible.

Further rules for the assignment of secure passwords can be found at Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, abbreviated as BSI). (https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/International/GSK_15_EL_EN_Draft.pdf?__blob=publicationFile&v=2)

You can use a program for password management for assistance when dealing with passwords. With its help, you can save passwords and PINs in encrypted form, manage them and generate secure passwords.

Defaults

As standard, the following passwords are effective for protection levels 1 - 3:

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

NOTICE

A cold restart reset the passwords to the default settings

After a cold restart (the NC powers up and standard machine data is loaded), the passwords of protection levels 1 - 3 are again reset to the default settings. For security reasons, after a cold restart we urgently recommend that the default settings of the passwords are changed.

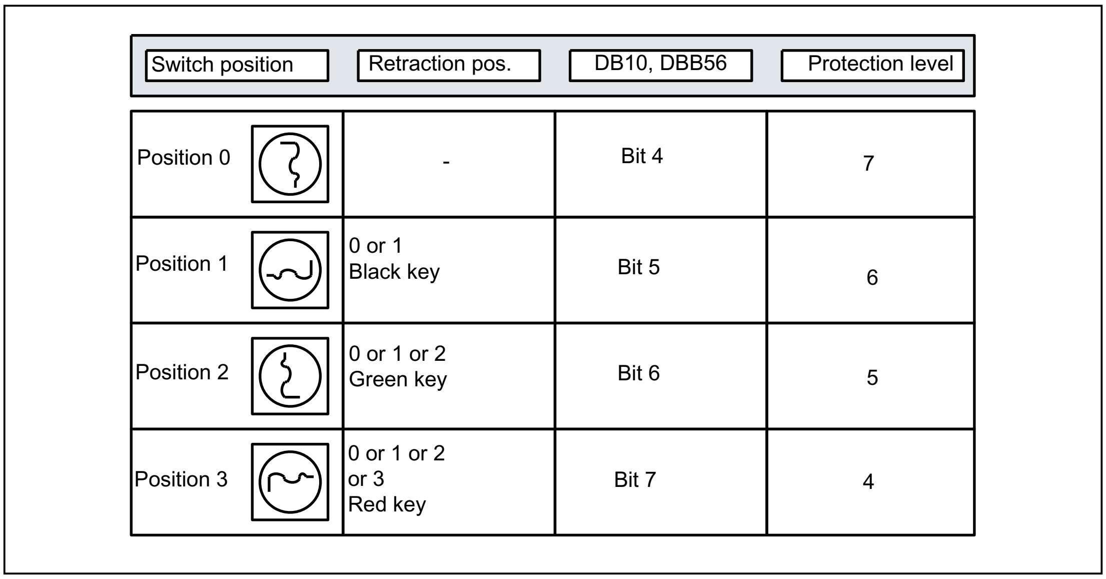
References

- "SINUMERIK 840D sl Basic Software and Operating Software" Commissioning Manual, Section "SINUMERIK Operate (IM9)" > "General settings" > "Access levels".
- Commissioning Manual, "Commissioning CNC: NC, PLC, Drive"; Section "Requirements for commissioning" > "Switch-on and power up" and "Access levels"

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

Keyswitch

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



Switch position	Retraction pos.	DB10, DBB56	Protection level
Position 0	-	Bit 4	7
Position 1	0 or 1 Black key	Bit 5	6
Position 2	0 or 1 or 2 Green key	Bit 6	5
Position 3	0 or 1 or 2 or 3 Red key	Bit 7	4

Figure 2-12 Switch positions 0 to 3

Switch positions

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

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

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

References

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

Default settings via the PLC user program

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

2.3.5.3 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.

2.3.6 Switching over motor/drive data sets

2.3.6.1 General Information

Motor and drive data sets

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

Note

References

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

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

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

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

Axial NC/PLC interface

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

- Formatting interface (Page 75)
- Request interface (Page 76)
- Display interface (Page 76)

2.3.6.2 Formatting interfaces

Formatting

The formatting interface is used to set which bits of the request and display interface are used to address the motor data sets (MDS) and which are used to address the drive data sets (DDS):

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

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

Motor and drive data sets in the drive

Formatting depends on the number of motor data sets (MDS) and drive data sets (DDS) in the drive. The number can be determined using the following drive parameters:

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

Validity of the interfaces

After powering up, as soon as the control has received all the required information from the drive and this has been evaluated by the NC, the request (Page 76) and display interfaces (Page 76) are shown as valid:

DB31, ... DBX130.7 == 1 (request and display interfaces valid)

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

Note

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

See also

Example (Page 76)

Overview of the interfaces (Page 77)

2.3.6.3 Request interface

The switchover to a new motor (MDS) and/or drive data sets (DDS) is requested via the interface:

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

Value range

Addressing a motor or drive data set n, with n = 1, 2, 3, ..., is realized based on its index i, with i = n - 1 = 0, 1, 2, ...

- Motor data sets: MDS[0, 1, 2, ... 15]
- Drive data sets: DDS[0, 1, 2, ... 31]

Formatting interfaces

Formatting the request interface, i.e. which bits are used to address the motor data sets (MDS) – and which are used to address the drive data sets (DDS) is set via the formatting interface (Page 75).

Motor and drive data sets in the drive

The number of motor data sets (MDS) and drive data sets (DDS) in the drive can be determined using the following drive parameters:

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

Motor data set (MDS) for main spindle drives

For main spindle drives, the following association applies:

- MDS[0] → star operation
- MDS[1] → delta operation

2.3.6.4 Display interface

The active motor data set (MDS) and drive data set (DDS) are displayed via the interface:

DB31, ... DBX93.0 - .4 == <MDS / DDS index>

Value range and formatting are identical to the request interface (Page 76).

2.3.6.5 Example

Two motor data sets (MDS) and two drive data sets (DDS) are available in the drive for each motor data set. This corresponds to "No.": 9 of the possible data set combinations displayed in the Figure 2-13 Principle of the motor/drive data set switchover (Page 78).

Format

Bit positions for drive data set switchover (DDS):

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

Bit positions for motor data set switchover (MDS):

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

Invalid bit positions:

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

Interfaces of the drive data sets (DDS)

Relevant bit positions of the request and display interfaces:

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

Interfaces of the motor data sets (MDS)

Relevant bit positions of the request and display interfaces:

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

Invalid bit positions (MDS/DDS)

Invalid bit positions of the request and display interfaces:

- DB31, ... DBX21.1 / DBX93.2 == 0
- DB31, ... DBX21.1 / DBX93.3 == 0
- DB31, ... DBX21.1 / DBX93.4 == 0

See also

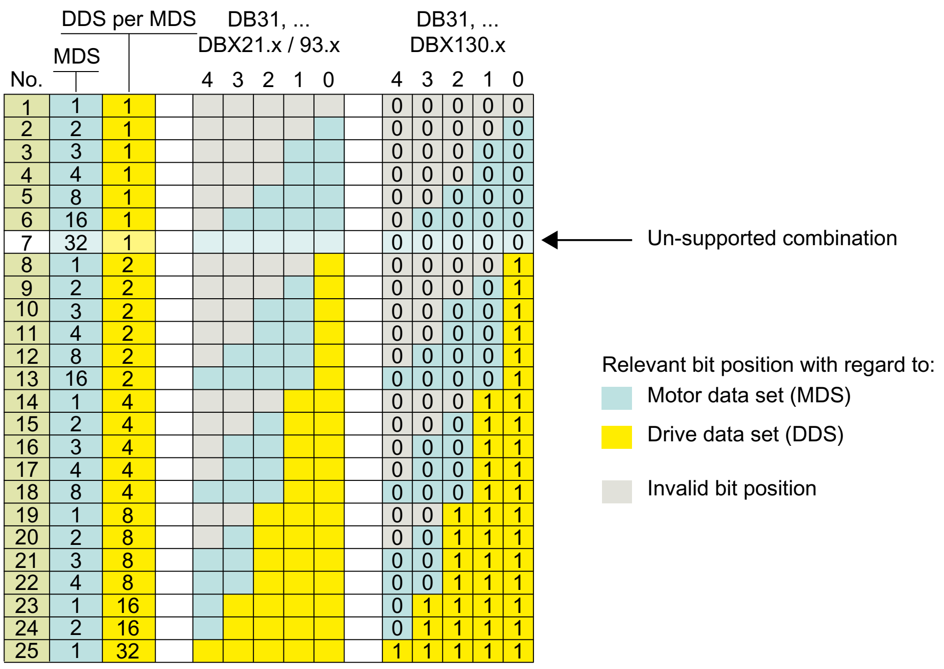
Overview of the interfaces (Page 77)

2.3.6.6 Overview of the interfaces

Table 2-1 Configurable MDS / DDS combinations

Number of MDS (motors)	Number of DDS (drives) per MDS
1	1 ... 32
2	1, 2, 4, 8, 16
3	1, 2, 4, 8

Number of MDS (motors)	Number of DDS (drives) per MDS
4	1, 2, 4, 8
5	1, 2, 4
6	1, 2, 4
7	1, 2, 4
8	1, 2, 4
9	1, 2
10	1, 2
11	1, 2
12	1, 2
13	1, 2
14	1, 2
15	1, 2
16	1, 2



		DDS per MDS	DB31, ... DBX21.x / 93.x					DB31, ... DBX130.x				
No.	MDS		4	3	2	1	0	4	3	2	1	0
1	1	1	⬜	⬜	⬜	⬜	⬜	0	0	0	0	0
2	2	1	⬜	⬜	⬜	⬜	🟦	0	0	0	0	0
3	3	1	⬜	⬜	⬜	🟦	🟦	0	0	0	0	0
4	4	1	⬜	⬜	⬜	🟦	🟦	0	0	0	0	0
5	8	1	⬜	⬜	🟦	🟦	🟦	0	0	0	0	0
6	16	1	⬜	🟦	🟦	🟦	🟦	0	0	0	0	0
7	32	1	🟦	🟦	🟦	🟦	🟦	0	0	0	0	0
8	1	2	⬜	⬜	⬜	⬜	🟨	0	0	0	0	1
9	2	2	⬜	⬜	⬜	🟦	🟨	0	0	0	0	1
10	3	2	⬜	⬜	🟦	🟦	🟨	0	0	0	0	1
11	4	2	⬜	⬜	🟦	🟦	🟨	0	0	0	0	1
12	8	2	⬜	🟦	🟦	🟦	🟨	0	0	0	0	1
13	16	2	🟦	🟦	🟦	🟦	🟨	0	0	0	0	1
14	1	4	⬜	⬜	⬜	🟨	🟨	0	0	0	1	1
15	2	4	⬜	⬜	🟦	🟨	🟨	0	0	0	1	1
16	3	4	⬜	🟦	🟦	🟨	🟨	0	0	0	1	1
17	4	4	⬜	🟦	🟦	🟨	🟨	0	0	0	1	1
18	8	4	🟦	🟦	🟦	🟨	🟨	0	0	0	1	1
19	1	8	⬜	⬜	🟨	🟨	🟨	0	0	1	1	1
20	2	8	⬜	🟦	🟨	🟨	🟨	0	0	1	1	1
21	3	8	🟦	🟦	🟨	🟨	🟨	0	0	1	1	1
22	4	8	🟦	🟦	🟨	🟨	🟨	0	0	1	1	1
23	1	16	🟦	🟨	🟨	🟨	🟨	0	1	1	1	1
24	2	16	🟦	🟨	🟨	🟨	🟨	0	1	1	1	1
25	1	32	🟨	🟨	🟨	🟨	🟨	1	1	1	1	1

Un-supported combination
Relevant bit position with regard to:
🟦 Motor data set (MDS)
🟨 Drive data set (DDS)
⬜ Invalid bit position

MDS	Number of motor data sets
DDS per MDS	Number of drive data sets per motor data set
DB31, ... DBX21.x	Request interface
DB31, ... DBX93.x	Display interface
DB31, ... DBX130.x	Formatting interface

Figure 2-13 Principle of the motor/drive data set switchover

2.3.6.7 Supplementary conditions

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

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

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

1 ≤ a ≤ (number of "DDS per MDS")

Example

4 motor data sets (MDS) and 8 drive data sets (DDS) per motor data set (DDS per MDS) are to be parameterized. This corresponds to "No.": 22 of the possible data set combination displayed in the Figure 2-13 Principle of the motor/drive data set switchover (Page 78):

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

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

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

Switchover instant: Drive parameter set

In principle it is possible to switch over drive parameter sets at any point in time. While an axis is traversing, especially when switching over speed controller parameters and the motor speed scaling, torque jumps/steps can occur. We therefore recommend that a drive parameter set is only switched over for stationary states, especially when the axis is at a standstill.

See also

Overview of the interfaces (Page 77)

2.4 Examples

2.4.1 Parameter set changeover

Parameter set changeover

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

Preconditions

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

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

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

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

Parameter-set-dependent machine data

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

Machine data	Comment
MD32200 $MA_POSCTRL_GAIN [0, AX1] = 4.0	K_V setting for parameter set 1
MD32200 $MA_POSCTRL_GAIN [1, AX1] = 2.0	K_V setting for parameter set 2
MD32200 $MA_POSCTRL_GAIN [2, AX1] = 1.0	K_V setting for parameter set 3
MD32200 $MA_POSCTRL_GAIN [3, AX1] = 0.5	K_V setting for parameter set 4
MD32200 $MA_POSCTRL_GAIN [4, AX1] = 0.25	K_V setting for parameter set 5
MD32200 $MA_POSCTRL_GAIN [5, AX1] = 0.125	K_V 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
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.	

Switchover

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

- Request by PLC user program:
  DB31, … DBX9.0 – DBX9.2 = 3 (parameter set servo)
  - A request to change over to the 4th parameter set is sent for machine axis AX1.
  - The parameter set is changed over once a delay has elapsed.
  - Parameter set 4 is now active, in accordance with machine data with index "3"
- Feedback by NC:
  DB31, … DBX69.0 – DBX69.2 = 3 (parameter set servo)
  - The NC confirms/acknowledges the parameter-set changeover.

2.5 Data lists

2.5.1 Machine data

2.5.1.1 Display machine data

Number	Identifier: $MM_	Description
SINUMERIK Operate		
9000	LCD_CONTRAST	Contrast
9001	DISPLAY_TYPE	Monitor type
9004	DISPLAY_RESOLUTION	Display resolution
9006	DISPLAY_SWITCH_OFF_INTERVAL	Time for screen darkening

2.5.1.2 NC-specific machine data

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

2.5.1.3 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

2.5.1.4 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

2.5.2 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)

2.5.3 Signals

2.5.3.1 Signals to NC

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

2.5.3.2 Signals from NC

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

2.5.3.3 Signals to operator panel front

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Brighten screen	DB19.DBX0.0	-
Darken screen	DB19.DBX0.1	DB1900.DBX5000.1 1)
Key disable	DB19.DBX0.2	DB1900.DBX5000.2
Delete cancel alarms	DB19.DBX0.3	-
Delete recall alarms	DB19.DBX0.4	-
Actual value in the WCS, (1) / MCS (0)	DB19.DBX0.7	DB1900.DBX5000.7
Part program: Unload	DB19.DBX13.5	-
Part program: Load	DB19.DBX13.6	-
Part program: Selection	DB19.DBX13.7	DB1700.DBX1000.7
File system active (0) / passive (1)	DB19.DBX14.7	-
Program selection from the PLC: index of the program list	DB19.DBB16 2)	DB1700.DBB1001 2)
Program selection from the PLC: program index in the program list	DB19.DBB17	DB1700.DBB1002
Mode change disable	DB19.DBX44.0	-

1) For SINUMERIK 828D, the bright/dark control is realized using DB1900.DBX5000.1:

DB1900.DBX5000.1=**0**: Screen bright

DB1900.DBX5000.1=**1**: Screen dark

DB1900.DBX5000.0 has no significance for the screensaver function.

2) Bit 7: Always 1

2.5.3.4 Signals from operator panel front

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Screen is dark	DB19.DBX20.1	-
User interface: Simulation active	DB19.DBX20.6	DB1900.DBX0.6
Switch over MCS/WCS	DB19.DBX20.7	DB1900.DBX0.7
Program selection of PLC status signals: Job completed	DB19.DBX26.1	DB1700.DBX2000.1
Program selection of PLC status signals: Error	DB19.DBX26.2	DB1700.DBX2000.2
Program selection of PLC status signals: Active	DB19.DBX26.3	DB1700.DBX2000.3
Program selection of PLC status signals: Unloading	DB19.DBX26.5	-
Program selection of PLC status signals: Loading	DB19.DBX26.6	-
Program selection of PLC status signals: Selection	DB19.DBX26.7	DB1700.DBX2000.7
FC9: Start "Measuring in JOG"	DB19.DBX42.0	-
FC9 Out: Active	DB19.DBX45.0	-
FC9 Out: Done	DB19.DBX45.1	-
FC9 Out: Error	DB19.DBX45.2	-
FC9 Out: StartErr	DB19.DBX45.3	-

2.5.3.5 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Delete distancetogo (channelspecific)	DB21, ...DBX6.2	DB320x.DBX6.2

2.5.3.6 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Channel-specific NC alarm pending	DB21, ...DBX36.6	DB330x.DBX4.6
NC alarm with machining stop is pending	DB21, … .DBX36.7	DB330x.DBX4.7
NC alarm with program stop	DB21, … .DBX39.1	DB330x.DBX7.1
Overstore active	DB21, ...DBX318.7	-

2.5.3.7 Signals to axis/spindle

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

2.5.3.8 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized, encoder 1/2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Traversing command minus/plus	DB31, ...DBX64.6/7	DB390x.DBX4.6/7
Follow up active	DB31, ...DBX61.3	DB390x.DBX1.3
Axis/spindle stationary ($n < n_{min}$)	DB31, ...DBX61.4	DB390x.DBX1.4
Position controller active	DB31, ...DBX61.5	DB390x.DBX1.5
Speed controller active	DB31, ...DBX61.6	DB390x.DBX1.6
Current controller active	DB31, ...DBX61.7	DB390x.DBX1.7
Lubrication pulse	DB31, ...DBX76.0	DB390x.DBX1002.0
Ramp-function generator disable active	DB31, ...DBX92.1	DB390x.DBX4000.1
Motor/drive data set: Display	DB31, ...DBX93.0 - 4	DB390x.DBX4001.0 - 4

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Drive ready	DB31, ...DBX93.5	DB390x.DBX4001.5
Integrator disable, speed controller	DB31, ...DBX93.6	DB390x.DBX4001.6
Pulses enabled	DB31, ...DBX93.7	DB390x.DBX4001.7
Motor temperature prewarning	DB31, ...DBX94.0	DB390x.DBX4002.0
Heat sink temperature prewarning	DB31, ...DBX94.1	DB390x.DBX4002.1
Run-up completed	DB31, ...DBX94.2	DB390x.DBX4002.2
$\|M_d\| < M_{dx}$	DB31, ...DBX94.3	DB390x.DBX4002.3
$\|n_{act}\| < n_{min}$	DB31, ...DBX94.4	DB390x.DBX4002.4
$\|n_{act}\| < n_x$	DB31, ...DBX94.5	DB390x.DBX4002.5
$n_{act} = n_{set}$	DB31, ...DBX94.6	DB390x.DBX4002.6
Alarm of alarm class C is active	DB31, ...DBX95.7	-
Motor/drive data set: Formatting	DB31, ...DBX130.0 - 4	DB390x.DBX4008.0 - 4
Motor/drive data set: Formatting is valid	DB31, ... DBX130.7	DB390x.DBX4008.7

A3: Axis monitoring functions 3

3.1 Contour monitoring

3.1.1 Contour error

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

Signal distortions can be linear or non-linear.

Linear signal distortions

Linear signal distortions are caused by:

- Speed and position controller not being set optimally
- Different servo gain factors (K_V) of the feed axes involved in creating the path
  With the same servo gain factor (K_V) for two linear-interpolated axes, the actual position follows the set position along the same path but with a time delay. With different servo gain factors (K_V), a parallel offset arises between the set and actual path.
- Unequal dynamic response of the feed drives
  Unequal drive dynamic responses lead to path deviations especially on contour changes. Circles are distorted into ellipses by unequal dynamic responses of the two feed drives.

Non-linear signal distortions

Non-linear signal distortions are caused by:

- Activation of the current limitation within the machining area
- Activation of the limitation of the speed setpoint
- Backlash within and/or outside the position control loop
  When traversing a circular path, contour errors occur primarily due to the reversal error and friction.
  During motion along straight lines, a contour error arises due to a reversal error outside the position control loop, e.g. due to a tilting milling spindle. This causes a parallel offset between the actual and the set contour. The shallower the gradient of the straight line, the larger the offset.
- Nonlinear friction behavior of slide guides

3.1.2 Following-error monitoring

Function

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

The following error that arises depends on:

- Position control loop gain
  MD32200 $MA_POSCTRL_GAIN (servo gain factor)
- Maximum acceleration
  MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)
- Maximum velocity
  MD32000 $MA_MAX_AX_VELO (maximum axis velocity)
- With activated feedforward control:
  Precision of the path model and the parameters:
  MD32610 $MA_VELO_FFW_WEIGHT (factor for the velocity feedforward control)
  MD32800 $MA_EQUIV_CURRCTRL_TIME (equivalent time constant current control loop for feedforward control)
  MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

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

MD36400 $MA_CONTOUR_TOL (Contour monitoring tolerance range)

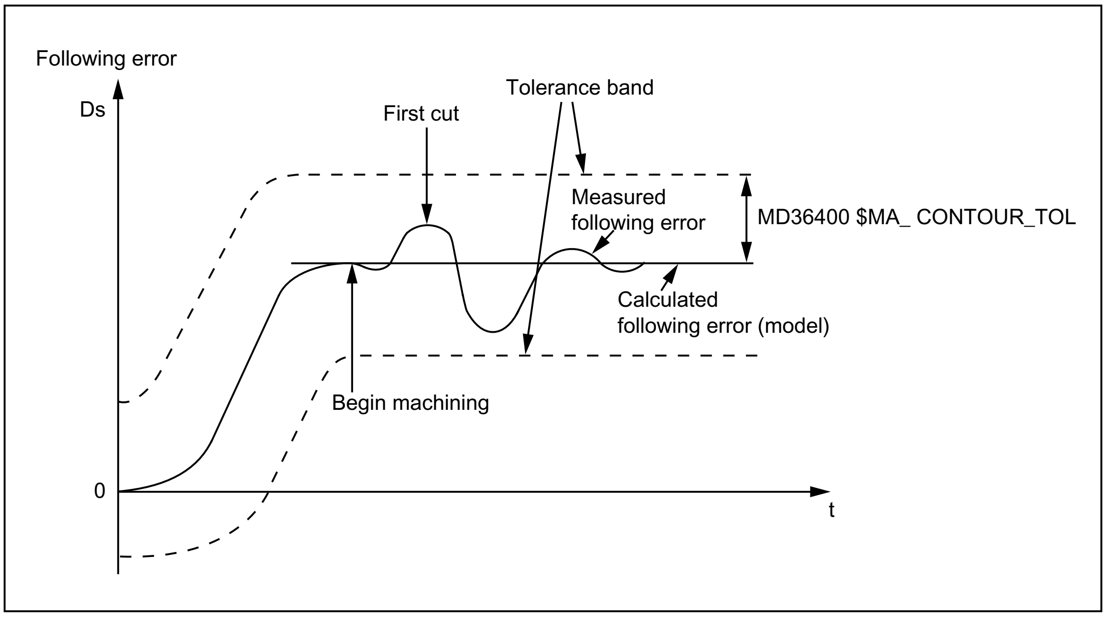


Figure 3-1 Following-error monitoring

Effectiveness

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

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

Fault

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

25050 "Axis <Axis name> Contour monitoring"

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

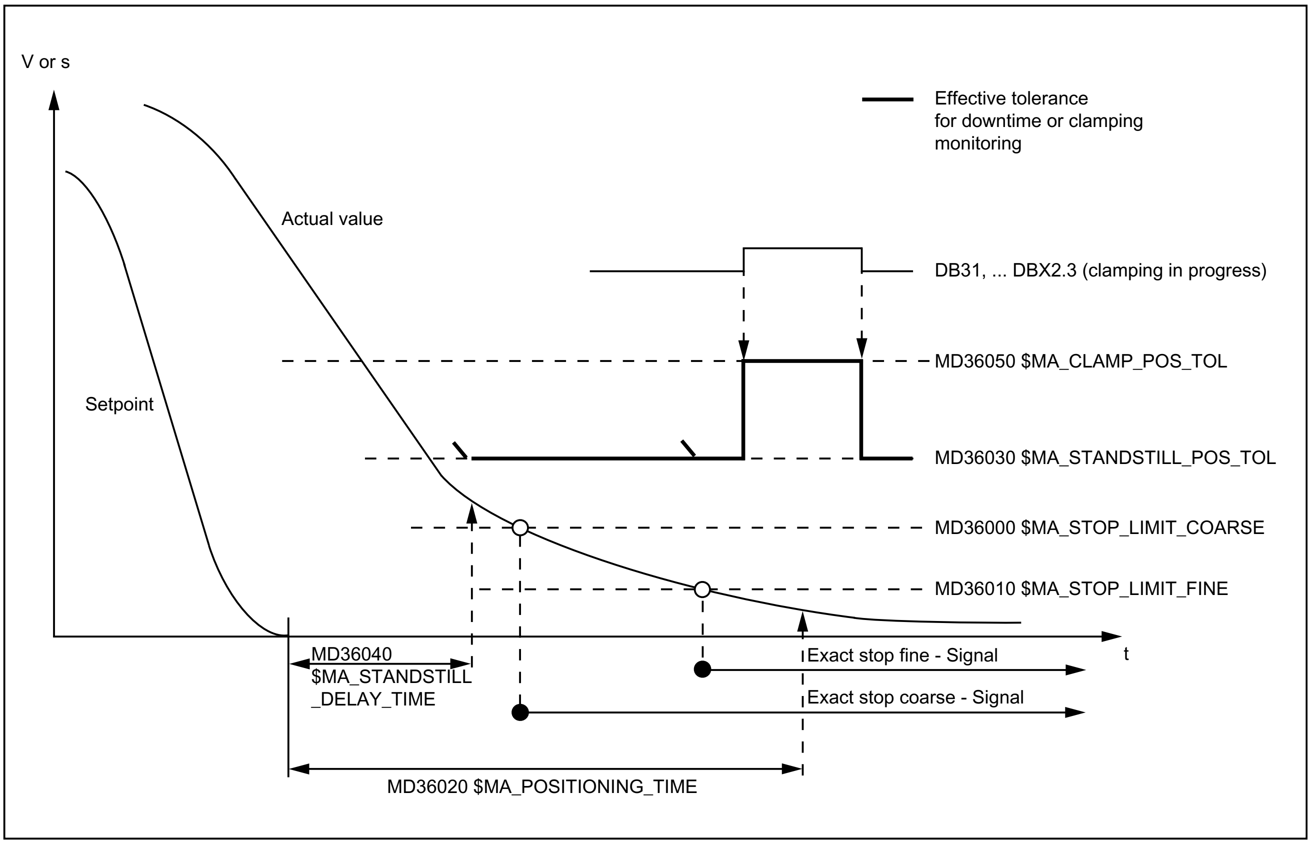
MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

3.2 Positioning, zero speed and clamping monitoring

3.2.1 Correlation between positioning, zero-speed and clamping monitoring

Overview

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


V or s
Effective tolerance for downtime or clamping monitoring
Actual value
DB31, ... DBX2.3 (clamping in progress)
MD36050 $MA_CLAMP_POS_TOL
Setpoint
MD36030 $MA_STANDSTILL_POS_TOL
MD36000 $MA_STOP_LIMIT_COARSE
MD36010 $MA_STOP_LIMIT_FINE
t
MD36040 $MA_STANDSTILL _DELAY_TIME
Exact stop fine - Signal
Exact stop coarse - Signal
MD36020 $MA_POSITIONING_TIME

3.2.2 Positioning monitoring

Function

At the end of a positioning operation:

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

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

MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)

MD36020 $MA_POSITIONING_TIME (delay time exact stop fine)

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

Note

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

Rules for MD setting

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

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

Effectiveness

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

- Linear axes
- Rotary axes
- Position-controlled spindles

Fault

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

25080 "Axis <Axis name> Position monitoring"

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

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

3.2.3 Zero-speed monitoring

Function

At the end of a positioning operation:

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

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

MD36040 $MA_STANDSTILL_DELAY_TIME (zero-speed monitoring delay time)

MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)

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

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

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

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

Effectiveness

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

- Linear axes
- Rotary axes
- Position-controlled spindles

Fault

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

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

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

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

3.2.4 Parameter set-dependent exact stop and standstill tolerance

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

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

the positioning tolerances:

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

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

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

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

3.2.5 Clamping monitoring

3.2.5.1 Function

For machine axes that are mechanically clamped upon completion of a positioning operation, a displacement of the axis from the position setpoint can result from the clamping process. Setting the NC/PLC interface signal DB31, ... DBX2.3 (clamping in progress) causes the clamping tolerance (MD36050 $MA_CLAMP_POS_TOL) rather than the standstill tolerance (MD36030 $MA_STANDSTILL_POS_TOL) to be monitored for the duration of the clamping process. Alarm 26000 "clamping monitoring" is issued if the clamping tolerance is overshot.

Alarm delay time

If a time-limited overshooting of the clamping tolerance is permitted, an alarm delay time must be specified via machine data MD36051 $MA_CLAMP_POS_TOL_TIME. The alarm is then output only after expiration of the parameterized time when the clamping tolerance is overshot. No alarm is output when the clamping tolerance is undershot again after expiration of the time. The time is restarted when the clamping tolerance is overshoot again.

To permit a response to overshooting the clamping tolerance prior to expiration of the alarm delay time, the axis-specific NC/PLC interface signal DB31, ... DBX102.3 (clamping tolerance overshot) is set. The signal is reset again when the clamping tolerance is undershot.

3.2.5.2 Machine data

Clamping tolerance

The clamping tolerance, which is higher than the standstill tolerance, is entered in machine data:

MD36050 $MA_CLAMP_POS_TOL[<axis>] = <clamping tolerance>

Alarm delay time

If a time-limited overshooting of the clamping tolerance is tolerated, the maximum permissible alarm delay time must be specified in the machine data.

MD36051 $MA_CLAMP_POS_TOL_TIME[<axis>] = <alarm delay time>

When the clamping tolerance is exceeded, alarm 26000 "Clamping monitoring" is only displayed after the alarm delay time has elapsed.

An alarm is not output, if the clamping tolerance is fallen below again before the alarm delay time lapses.

The alarm delay time is restarted when the clamping tolerance is exceeded again.

Special clamping functions

The special clamping functions that automate the release and set of the part program execution sequence are activated bit-by-bit using machine data:

MD36052 $MA_STOP_ON_CLAMPING[<Achse>], <bit> = <value>

<Bit>	<Val-ue>	Meaning
0	Automatic stop to release the clamping	
	0	Not active
	1	Active
1	Optimized clamping release	
	0	Not active
	1	Active, precondition: Bit 0 == 1
2	Automatic stop to set the clamping	
	0	Not active
	1	Active

3.2.5.3 NC/PLC interface signals

Activating the clamping monitoring

The clamping monitoring is activated by setting the NC/PLC interface signal:

DB31, ... DBX2.3 = 1 (clamping in progress)

Overshooting the clamping tolerance

Overshooting the clamping tolerance is indicated with the NC/PLC interface signal:

DB31, ... DBX102.3 == 1 (clamping tolerance overshot)

The signal is **set** when the clamping tolerance is overshot within the alarm delay time.

The signal is **reset** when the clamping tolerance is undershot within the alarm delay time or the follow-up mode is activated for the axis.

3.2.5.4 Fault responses

Fault responses when the clamping tolerance is exceeded:

- Alarm 26000 "Clamping monitoring" is indicated
- The axis is brought to a standstill with the parameterized maximum acceleration:
  MD32300 $MA_MAX_AX_ACCEL
  Whereby, the maximum duration of the braking ramp for error states is monitored:
  MD36610 $MA_AX_EMERGENCY_STOP_TIME
- Follow-up mode is activated for the axis:
  DB31, ... DBX61.3 == 1
- The "clamping tolerance exceeded" signal is reset:
  DB31, ... DBX102.3 == 0

3.2.5.5 "Automatic stop to release the clamping" clamping function

The "Automatic stop to release the clamping" clamping function adds an NC-internal stop before the traversing block of the clamping axis for continuous-path mode.

The stop is not effective or the continuous-path mode is not interrupted if the controller enable signal (DB31, ... DBX2.1) of the clamping axis is set prior to the block change.

If the controller enable signal of the clamping axis is **not** set prior to the block change, the stop acts.

Parameterization

MD36052 $MA_STOP_ON_CLAMPING[<clamping axis>] = 'H01'

Requirements/assumptions

- If, for the clamping axis there is a **traversing command** (DB31, ... DBX64.6 / .7), then the clamping is **released** by the PLC user program.
- The following relationship must exist between the controller enable signal (DB31, ... DBX2.1) and the clamping of the clamping axis:

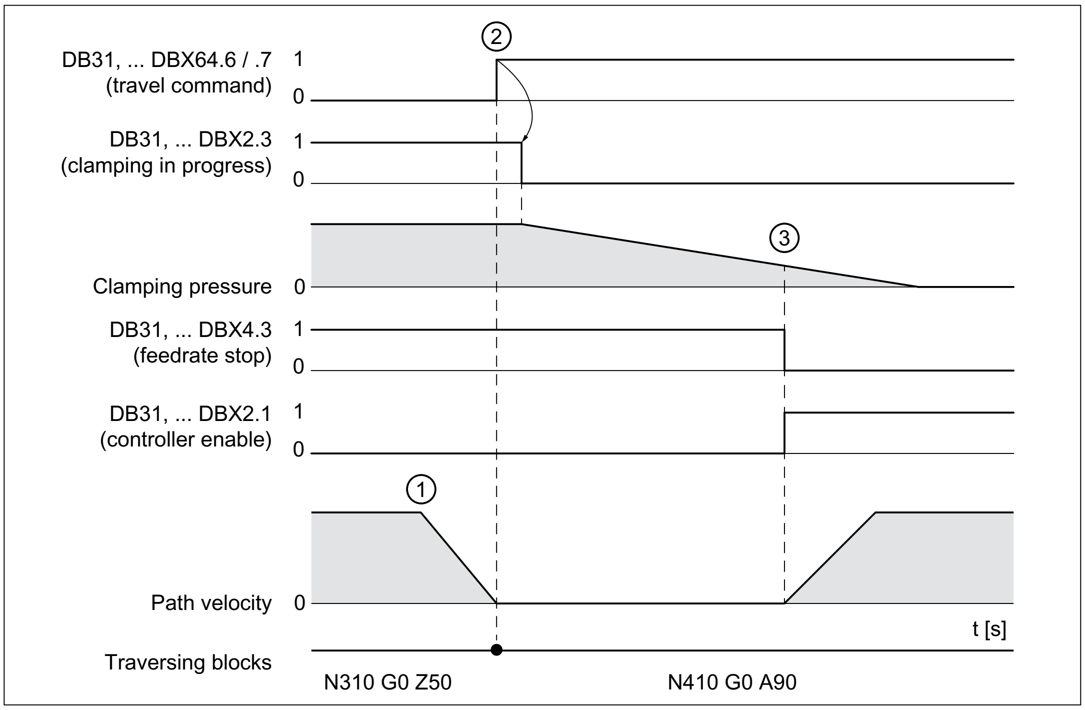
Controller enable	⇒	Clamping axis
not set	⇒	Clamped
Set	⇒	Not clamped

Example:

Program code	Comment
N100 G0 X0 Y0 Z0 A0 G90 G54 F500	; Approach initial state
N101 G641 ADIS=.1 ADISPOS=5	; Activate continuous-path mode
N210 G1 X10	; Traversing block
N220 G1 X5 Y20	; "
N310 G0 Z50	; Positioning block
N410 G0 A90	; " (**clamping axis**)
N510 G0 X100	; "
N520 G0 Z2	; "

Program code	Comment
`N610 G1 Z-4`	`; Traversing block`
`N620 G1 X0 Y-20`	`; "`

Schematic change of the NC/PLC interface signals and states for the N310 and N410 blocks:



① NC: The automatically inserted stop causes a stop at the N310 block end.

② NC → PLC: After the block change, the travel command for the clamping axis is set.
PLC: The clamping is released based on the travel command.

③ PLC → NC: The clamping pressure is removed appropriately. The clamping axis is enabled for traversal.

3.2.5.6 "Time-optimized release of the clamping" clamping function

The "Time-optimized release of the clamping" clamping function in conjunction with the "Automatic stop to release the clamping" clamping function" for continuous-path mode requests the release of the clamping NC-internal by the Look Ahead setting of the travel command for the clamping axis. The travel command is set only when until the traversal of the clamping axis, positioning (`G0` blocks) but no processing (`G1` blocks) is performed.

To obtain the reference to the traversing block of the clamping axis, the travel command prefixes a maximum of two rapid traverse blocks `(G0)`, including any internally generated intermediate blocks, to the traversing block.

Activation

MD36052 $MA_STOP_ON_CLAMPING[<clamping axis>] = 'H03'

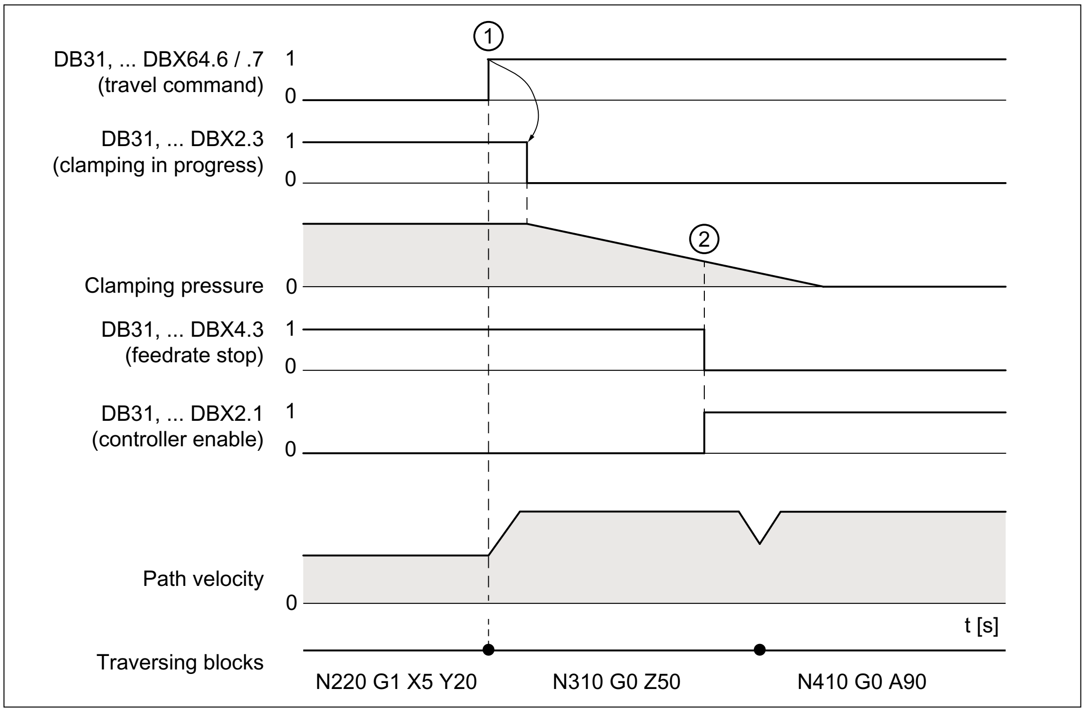
Requirements/assumptions

- If, for the clamping axis there is a **traversing command** (DB31, ... DBX64.6 / .7), then the clamping is **released** by the PLC user program.
- While other axes are traversing with rapid traverse (`G0`), the clamping axis must **not** be clamped.

Example:

Program code	Comment
`N100 G0 X0 Y0 Z0 A0 G90 G54 F500`	`; Approach initial state`
`N101 G641 ADIS=.1 ADISPOS=5`	`; Activate continuous-path mode`
`N210 G1 X10`	`; Machining block`
`N220 G1 X5 Y20`	`; "`
`N310 G0 Z50`	`; Positioning block`
`N410 G0 A90`	`; " (`**`clamping axis`**`)`
`N510 G0 X100`	`; "`
`N520 G0 Z2`	`; "`
`N610 G1 Z-4`	`; Machining block`
`N620 G1 X0 Y-20`	`; "`

Schematic change of the NC/PLC interface signals and states for the N220 to N410 blocks:



① NC → PLC: The travel command for the clamping axis is set because of the block change.
PLC: The clamping is released based on the travel command.

② PLC → NC: The clamping pressure is removed appropriately. The clamping axis is enabled for traversal.

3.2.5.7 "Automatic stop to set the clamping" clamping function

The clamping process takes a while. In continuous-path mode, an explicit stop of the traversing must be provided by programming, e.g. `G09`, `G60` or auxiliary function output, so that the clamping is reliably active before machining started.

The "Automatic stop to set the clamping" clamping function stops the traversing automatically in continuous-path mode. Clamping motion is stopped **before** or **in the** next machining block (traversing block without rapid traverse `G0`), if the clamping axis has not clamped up until then. The criterion that clamping has taken place and additional traversing motion has been enabled is the setting of the channel-specific feedrate override by the PLC user program not equal to 0% (DB21, ... DBB4 ≠ 0%)

Activation

MD36052 $MA_STOP_ON_CLAMPING[<clamping axis>] = 'H04'

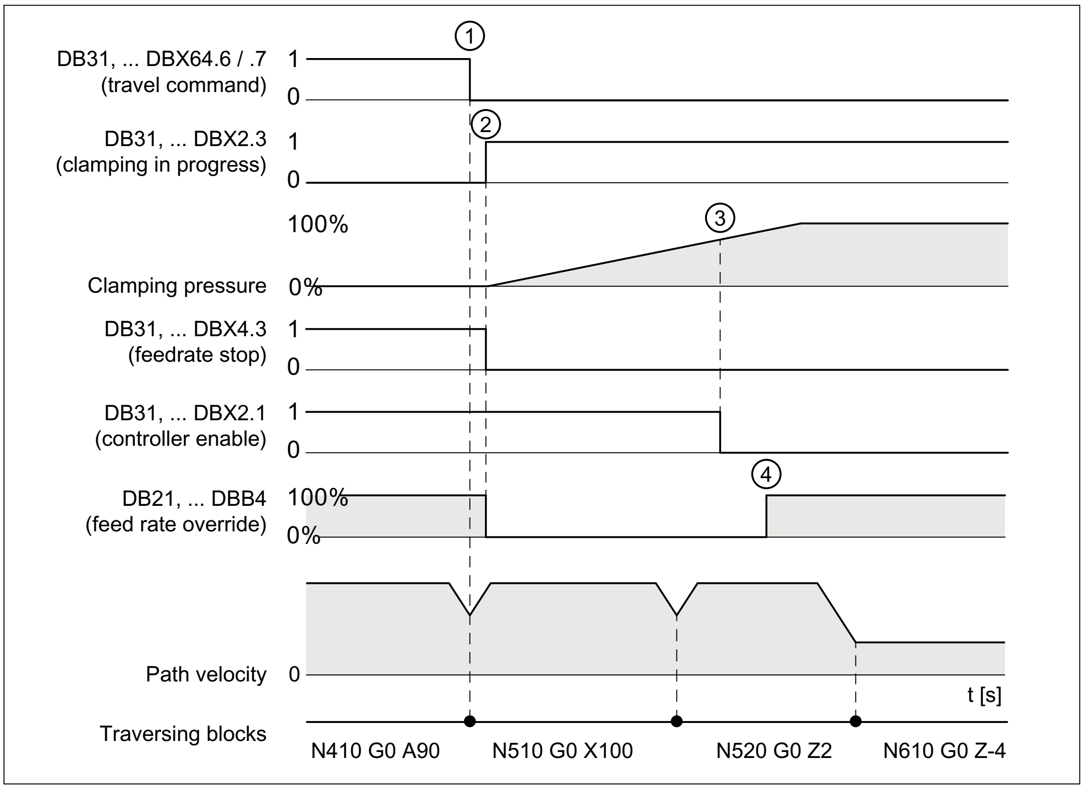
Requirements/assumptions

- If, for the clamping axis, there is **no** traversing command (DB31, ... DBX64.6 / .7), then the clamping is**closed** by the PLC user program.
- While other axes are traversing with rapid traverse (`G0`), the clamping axis must **not** be clamped.
- If the channel-specific feedrate override **is not equal to 0%** (DB21, ... DBB4 ≠ 0%), then the clamping axis is **clamped**.

Example

Program code	**Comment**
`N100 G0 X0 Y0 Z0 A0 G90 G54 F500`	`; Approach initial state`
`N101 G641 ADIS=.1 ADISPOS=5`	`; Activate continuous-path mode`
`N210 G1 X10`	`; Machining block`
`N220 G1 X5 Y20`	`; "`
`N310 G0 Z50`	`; Positioning block`
`N410 G0 A90`	`; "` (**`clamping axis`**)
`N510 G0 X100`	`; "`
`N520 G0 Z2`	`; "`
`N610 G1 Z-4`	`; Machining block`
`N620 G1 X0 Y-20`	`; "`

Schematic change of the NC/PLC interface signals and states for the N410 to N610 blocks:



① NC → PLC: The travel command for the clamping axis is reset because of the block change.
② PLC: The clamping is initiated
③ PLC → NC: The clamping pressure is sufficiently large to reset the controller enable
④ PLC → NC: Enable the N610 machining by setting the channel-specific feedrate override not 0%.

3.2.5.8 Supplementary conditions

Interrupted continuous-path mode

If during the above-described clamping functions, the continuous-path mode and thus also the "LookAhead" function is interrupted by blocks without traversing (e.g. output of an M function M82/M83), the functions behave as follows:

- **Clamping function: "Time-optimized release of the axis clamping"**
  (MD36052 $MA_STOP_ON_CLAMPING[<axis>] = 'B011')
  The function no longer acts because the travel command is set in Look Ahead mode only for blocks with active continuous-path mode. The output of the M function M82 in block N320 of the sample program below stops the traversing and so interrupts the continuous-path mode.
  The Look Ahead stopping on N410 by the function is not necessary because stopping occurs anyway by N320.
- **Clamping function: "Automatic stop to set the clamping":**
  (MD36052 $MA_STOP_ON_CLAMPING[<axis] = 'B100')
  The function generates a stop irrespective of `M83` that is executed as a function of "feedrate override 0%". The axis is thus stopped before the first machining block.

Note

Using clamping functions without clamping

The following clamping functions can also be used independent of clamping the axis:

- "Automatic stop to release the clamping":
  MD36052 $MA_STOP_ON_CLAMPING[<axis>] = 'B001'
  Behavior: A stop is made in the current block on the path when the controller enable (DB31, ... DBX2.1) for the parameterized <axis> is **not** set, but it is traversed in one of the following blocks.
- "Automatic stop to set the clamping":
  MD36052 $MA_STOP_ON_CLAMPING[<axis>] = 'B100'
  Behavior: A stop is made in the current block on the path when at the transition from rapid traverse blocks (G0) to traversing blocks (G1), the channel-specific feedrate override (DB21, ... DBB4) is 0%.

In both cases it is ensured that the path motion in continuous-path mode is already stopped before the start of the relevant part program block and not just within the block.

Table 3-1 Sample program: Interrupted continuous-path mode

Program code	Comment
`N100 G0 X0 Y0 Z0 A0 G90 G54 F500`	`; Approach initial state`
`N101 G641 ADIS=.1 ADISPOS=5`	`; Activate continuous-path mode`
`N210 G1 X10`	`; Traversing block`
`N220 G1 X5 Y20`	`; "`
`N310 G0 Z50`	`; Rapid traverse block`
`N320 M82`	; **Interrupt** continuous-path mode
`N410 G0 A90`	`; Rapid traverse block`
`N420 M83`	; **Interrupt** continuous-path mode
`N510 G0 X100`	`; Rapid traverse block`
`N520 G0 Z2`	`; "`

Program code	Comment
`N610 G1 Z-4`	`; Traversing block`
`N620 G1 X0 Y-20`	`; "`

Block change criterion: Clamping tolerance

After activation of clamping monitoring (DB31, ... DBX2.3), the block change criterion for traversing blocks in which the stop is made at the end of the block, the clamping tolerance rather than the exact stop condition acts for the clamping axis:

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

Behavior for releasing the clamping

If the clamping axis was moved by the clamping process from the position setpoint, it is returned by the NC to the position setpoint after releasing the clamping and setting the (DB31, ... DBX2.1) controller enable signal. Repositioning depends on whether "Follow-up mode" was activated for the axis during the clamping process:

- DB31, ... DBX1.4 == 0 (follow-up mode not active) ⇒ Abrupt by the position controller
- DB31, ... DBX1.4 == 1 (follow-up mode active) ⇒ interpolatory method

---

Note

The following data can be evaluated by the PLC user program as the criterion for activating the follow-up mode (DB31, ... DBX1.4):

- DB31, ... DBX60.6/.7 (position reached with coarse/fine exact stop)
- Actual position of the clamping axis

Follow-up mode

The clamping monitoring is not active in follow-up mode

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

3.3 Speed-setpoint monitoring

Function

The speed setpoint comprises:

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

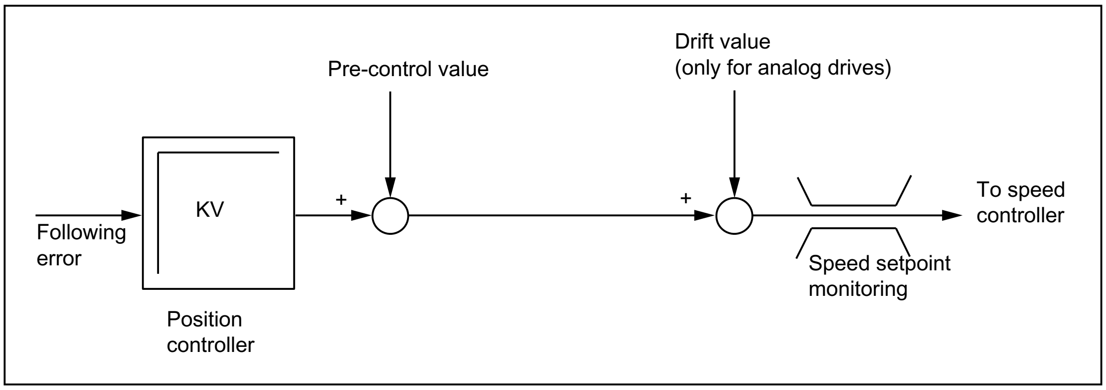


Figure 3-2 Speed setpoint calculation

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

MD36210 $MA_CTRLOUT_LIMIT (maximum speed setpoint)

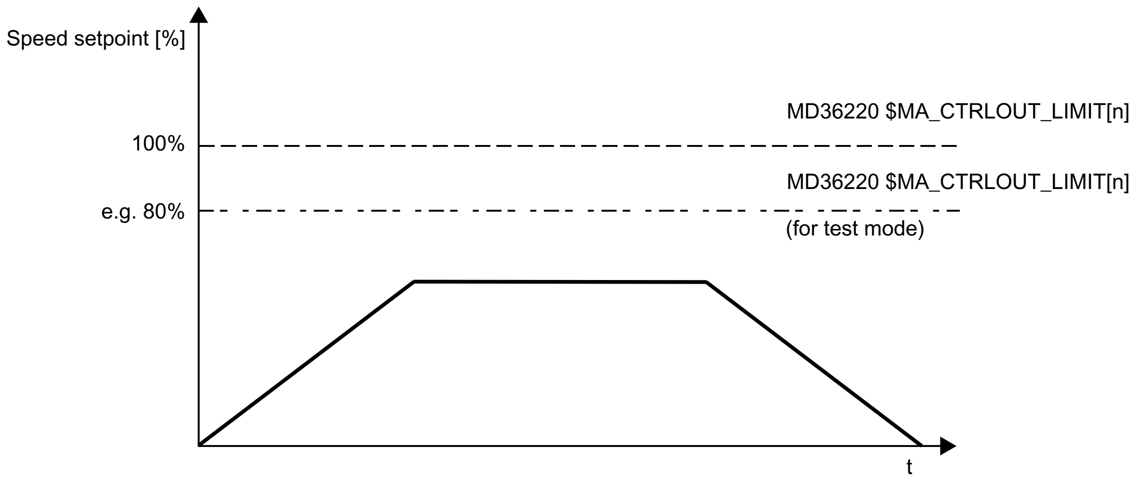


Figure 3-3 Speed setpoint limitation

Speed-setpoint monitoring delay

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

MD36220 $MA_CTRLOUT_LIMIT_TIME (speed-setpoint monitoring delay)

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

Effectiveness

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

Fault

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

25060 "Axis <Axis name> Speed-setpoint limitation"

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

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

Note

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

3.4 Actual-velocity monitoring

Function

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

MD36200 $MA_AX_VELO_LIMIT (velocity-monitoring threshold)

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

- For axes:
  MD32000 $MA_MAX_AX_VELO (maximum axis velocity)
- For spindles:
  MD35110 $MA_GEAR_STEP_MAX_VELO_LIMIT[n] (maximum speed of gear stage)

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

Activation

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

Effectiveness

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

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

Fault

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

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

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

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

3.5 Measuring system monitoring

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

Monitoring functions in the drive

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

References:
Drive Functions SINAMICS S120

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

MD36310 $MA_ENC_ZERO_MONITORING

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

For details on the alarms, see:

References:
Diagnostics Manual

Note

For hardware faults, the referencing status of the machine axis is reset:

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

Monitoring functions in the NC

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

3.5.1 Encoder-limit-frequency monitoring

Function

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

MD36300 $MA_ENC_FREQ_LIMIT (encoder limit frequency)

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

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

Effectiveness

The encoder limit frequency is operative for:

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

Fault

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

- Message to the PLC:
  DB31, ... DBX60.2 or 60.3 = 1 (encoder limit frequency exceeded 1 or 2)
- Spindles
  Spindles are not stopped but continue to turn with speed control.
  If the spindle speed is reduced so much that the encoder frequency passes below the encoder limit frequency, the actual value system of the spindle is automatically resynchronized.
- Axes
  The following alarm is displayed:
  21610 "Channel <Channel number> Axis <Axis name> Encoder <Encoder number > Frequency exceeded"
  The affected axis is stopped via the configured braking ramp in follow-up mode:
  MD36610 $MA_AX_EMERGENCY_STOP_TIME
  (maximum time for braking ramp when an error occurs)

Note

If the encoder limit frequency is exceeded, a position-controlled machine axis must be re-referenced (see Section "R1: Referencing (Page 1223)").

3.5.2 Plausibility check for absolute encoders

Function

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

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

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

Note

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

Note

Rotary absolute encoders

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

Note

Upgrading the NC software

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

Zero mark diagnostics

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

MD36312 $MA_ENC_ABS_ZEROMON_WARNING (zero-mark monitoring warning threshold)

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

Procedure when commissioning the system:

1. Deactivate zero-mark monitoring:
   MD36310 $MA_ENC_ZERO_MONITORING = 0
2. Activate zero-mark diagnostics:
   MD36312 $MA_ENC_ABS_ZEROMON_WARNING = 1
3. Move axis and monitor system variable $VA_ENC_ZERO_MON_ERR_CNT (number of detected limit value violations).
4. If $VA_ENC_ZERO_MON_ERR_CNT ≠ 0:
   Increase MD36312 value and repeat step 3.
5. If $VA_ENC_ZERO_MON_ERR_CNT = 0 (over a longer period of time!):
   The correct value for MD36310 is located! Apply the value from MD36312 to MD36310 and then set MD36312 to "0".

Note

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

Error case

Alarm 25020

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

"Axis <Axis name> Zero-mark monitoring active encoder"

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

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

Alarm 25021

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

"Axis <Axis name> Zero-mark monitoring passive encoder"

There is no further alarm response.

Note

In the event of a fault, the adjustment of the absolute encoder is lost and the axis is no longer referenced. The absolute encoder must be readjusted (see Section "Referencing with absolute encoders (Page 1253)").

Note

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.

3.5.3 Customized error reactions

Customized zero-mark monitoring

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

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

Example:

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

Effectiveness

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

MD36310 $MA_ENC_ZERO_MONITORING

Value	Meaning

0	If only user-specific monitoring is to be implemented, the default zero-mark monitoring must be deactivated:
	MD36310 = 0
	and
	MD36312 = 0

> 0	Customized monitoring and standard zero-mark monitoring operate in parallel.
100	All encoder monitoring functions are deactivated.

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

Example:

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

System variables

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

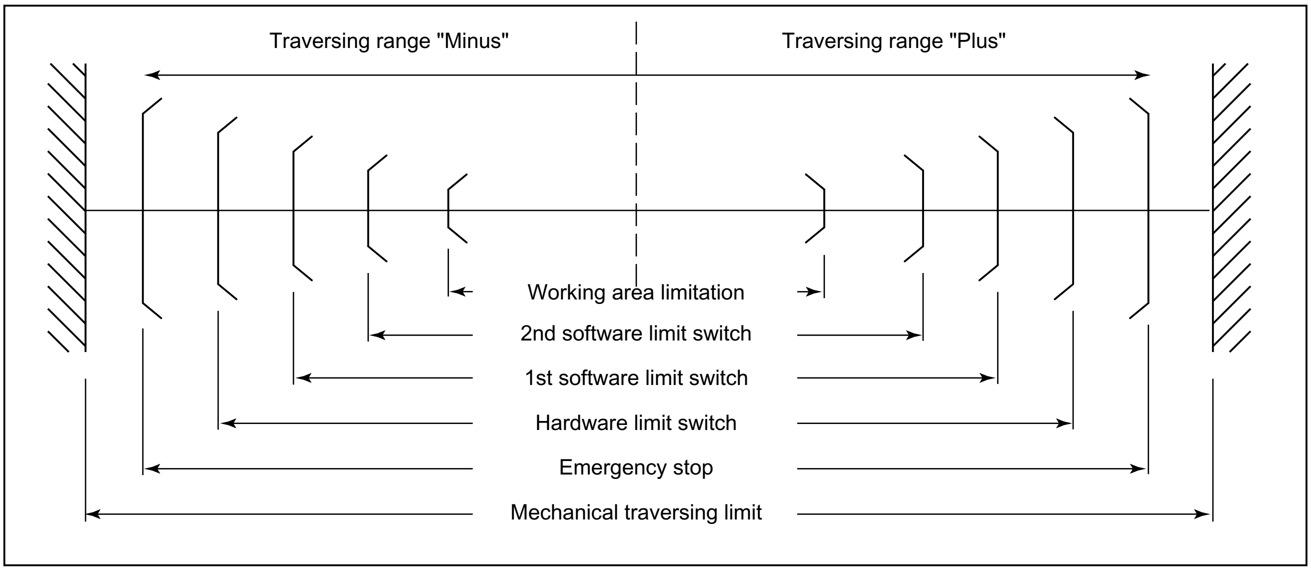
System variable	Meaning
$VA_ENC_ZERO_MON_ERR_CNT[<n>,<axis>]	Number of detected limit value violations. Contains the current number of detected limit value violations when comparing the absolute and the incremental encoder tracks. The value is reset to 0 at: • POWER ON • Selection/deselection of parking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

<n>: Encoder number

<axis>: Axis name

3.6 Limit-switch monitoring

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



3.6.1 Hardware limit switch

Function

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

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

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

Parameterization

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

MD36600 $MA_BRAKE_MODE_CHOICE (braking behavior on hardware limit switch)

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

Effectiveness

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

Effect

Upon reaching the hardware limit switch, the following occurs:

- Alarm 21614 "Channel <channel number> axis <axis name> 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.

3.6.2 Software limit switch

Function

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

MD36100 POS_LIMIT_MINUS (1st software limit switch minus)

MD36110 POS_LIMIT_PLUS (1st software limit switch plus)

MD36120 POS_LIMIT_MINUS2 (2nd software limit switch minus)

MD36130 POS_LIMIT_PLUS2 (2nd software limit switch plus)

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

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

Effectiveness

The software limit switches are active:

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

Supplementary conditions

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

- `PRESET`
  After use of the function `PRESET`, the software limit-switch monitoring is no longer active. The machine must first be re-referenced.
- Endlessly rotating rotary axes
  No software limit-switch monitoring takes place for endlessly rotating rotary axes:
  MD30310 $MA_ROT_IS_MODULO == 1 (modulo conversion for rotary axis and spindle)
  Exception: Setup-rotary axes

Effects

Automatic operating modes (AUTOMATIC, MDI)

- Without transformation, without overlaid motion, unchanged software limit switch:
  A part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is not started.
- With transformation:
  Different reactions occur depending on the transformation type:
  – Behavior as above.
    or
  – The part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is started. The affected machine axis stops at the active software limit switch. The other machine axes participating in the traversing motion are braked. The programmed contour is left during this process.
- With overlaid motion
  The part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is started. Machine axes that are traveling with overlaid motion or have traveled with overlaid motion stop at the active software limit switch in question. The other machine axes participating in the traversing motion are braked. The programmed contour is left during this process.

Manual operating modes

- JOG without transformation
  The machine axis stops at the software limit switch position.
- JOG with transformation
  The machine axis stops at the software limit switch position. Other machine axes participating in the traversing motion are braked. The preset path is left during this process.

General

- Changing of the software limit switch (1st ↔ 2nd software limit switch)
  If the actual position of the machine axis after changing lies behind the software limit switch, it is stopped with the maximum permissible acceleration.
- Overrunning the software limit switch in JOG mode
  If the position of the software limit switch is reached and renewed pressing of the traversing button should cause further travel in this direction, an alarm is displayed and the axis is not traversed farther:
  Alarm 10621 "Channel <channel number> axis <axis name> is at software limit switch <direction>"

3.7 Working area limitation monitoring

3.7.1 General

Function

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

The following versions are available:

- Working area limitation in the Basic Coordinate System (BCS)
  The traversing range limits are specified relative to the Basic Coordinate System.
- Working area limitation in the workpiece coordinate system (WCS) or adjustable zero system (AZS)
  The traversing range limits are specified relative to the workpiece coordinate system or to the adjustable zero system.

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

Reference point at the tool

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

- **Transformation inactive**
  Without transformations during traversing motion with an active tool the position of the tool tip P is monitored, i.e. during the monitoring the tool length is considered automatically. Consideration of the tool radius must be activated separately:
  MD21020 $MC_WORKAREA_WITH_TOOL_RADIUS (Consideration of the tool radius in the working area limitation)
- **Transformation active**
  In the case of certain transformations the monitoring of the working area limitation may differ from the behavior without transformation:
  - The tool length is a component of the transformation ($MC_TRAFO_INCLUDES_TOOL_X = TRUE):
    In this case the tool length is not considered, i.e. the monitoring refers to the tool carrier reference point.
  - Transformation with change in orientation:
    In the case of transformations with changes in orientation, monitoring is always based on the tool center point. MD21020 has no influence.

Note

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

Response

Automatic operating modes

- With / without transformation
  The parts program block with a programmed traversing motion that would lead to overrunning of the working area limits is not executed.
- With superimposed motion
  The axis, which would violate the working area limitation due to a superimposed motion, is braked with maximum acceleration and without jerk limits (`BRISK`), and will come to a stop in the position of the working area limitation. Other axes involved in the movement are braked according to current acceleration behavior (e.g. `SOFT`). The path correlation may be lost due to different braking accelerations (contour violation).

Manual operating modes

- JOG with / without transformation
  The axis is positioned at the working area limitation and then stopped.

Powerup response

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

Overrunning of the working area limitation in JOG mode

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

Geo-axis replacement

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

MD10604 $MN_WALIM_GEOAX_CHANGE_MODE = <value>

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

3.7.2 Working area limitation in BCS

Application

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

Working area limits

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

Working area limitation through setting data

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

SD43420 $SA_WORKAREA_LIMIT_PLUS (working area limitation plus)

SD43430 $SA_WORKAREA_LIMIT_MINUS (working area limitation minus)

Programmed working area limitation

The programming is done using the G commands:

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

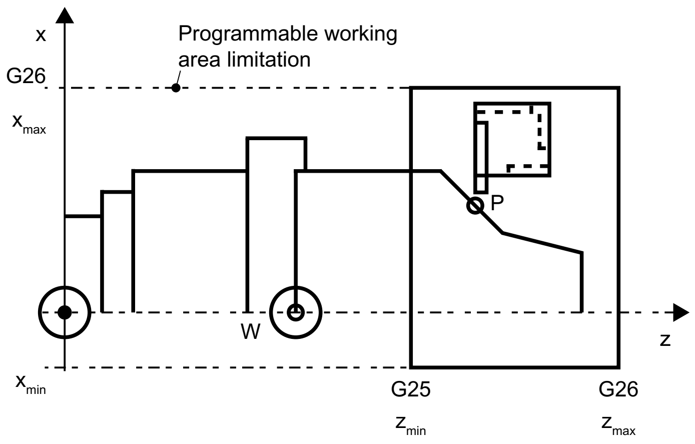


Figure 3-4 Programmed working area limitation

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

Activation/Deactivation

Working area limitation through setting data

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

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

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

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

Programmed working area limitation

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

`WALIMON`	Working area limitation ON
or	
`WALIMOF`	Working area limitation OFF

Changing the working area limitation

Working area limitation through setting data

HMI user interface: Operating area "Parameter"

- Automatic modes:
  - Changes: Possible only in the RESET state
  - Effectiveness: Immediately
- Manual operating modes:
  - Changes: Always possible
  - Effectiveness: At the start of the next traversing motion

Programmed working area limitation

The working area limitation can be changed in the part program via `G25` or `G26` <Axis name> <value>. The change takes effect immediately.

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

MD10710 $MN_PROG_SD_RESET_SAVE_TAB[0] = 43420

MD10710 $MN_PROG_SD_RESET_SAVE_TAB[1] = 43430

Reset position

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

MD20150 $MC_GCODE_RESET_VALUES (reset setting of the G groups)

3.7.3 Working area limitation in WCS/SZS

Application

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

Requirement

The channel axes must be referenced.

Working area limitation group

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

A working area limitation group comprises the following data:

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

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

MD28600 $MC_MM_NUM_WORKAREA_CS_GROUPS

Maximum 10 working area limitation groups are possible per channel.

Set working area limits

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

- $P_WORKAREA_CS_LIMIT_PLUS[<WALimNo>, <Ax>]
- $P_WORKAREA_CS_LIMIT_MINUS[<WALimNo>, <Ax>]

with <WALimNo>:	=	Working area limitation group	
		Value range:	0 (group 1) ... 9 (group 10)
<Ax>	=	Channel axis name	

Enable working area limits

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

- $P_WORKAREA_CS_PLUS_ENABLE[<WALimNo>, <Ax>]
- $P_WORKAREA_CS_MINUS_ENABLE[<WALimNo>, <Ax>]

with <WALimNo>:	=	Working area limitation group	
		Value range:	0 (group 1) ... 9 (group 10)
<Ax>	=	Channel axis name	

Using the direction-specific enable, it is possible to limit the working range for an axis in just one direction.

There is no activation through the enable.

Select reference system

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

$P_WORKAREA_CS_COORD_SYSTEM[<WALimNo>] = *<value>*

with <WALimNo>:	=	Working area limitation group	
		Value range:	0 (group 1) ... 9 (group 10)

<value>	Meaning
1	The reference system is the WCS.
3	The reference system is the SZS.

Activation of working area limits

The working area limits of a working area limitation group are activated in the part program with the G command `WALCS<n>`.

with `<n>`:	=	Number of the working area limitation group
		Value range: 1 ... 10

Deactivation of working area limits

The working area limits of a working area limitation group active in the channel are deactivated in the part program with the G command `WALCS0`.

Change working area limits

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

Data storage

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

Note

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

Data backup

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

- _N_CHx_WAL
  To save the system variable values for channel x.
- _N_COMPLETE_WAL
  To save the system variable values for all channels.

Note

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

Behavior in JOG mode

Initial situation:

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

Behavior when a working area limitation responds:

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

Setting the initial setting

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

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

with <n>:	=	Number of the working area limitation group
		Value range: 1 ... 10

The following setting determines whether the pre-selected working area limitation acts for ramp up and reset or part program end:

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

<value>	Meaning
0	The working area group takes effect in accordance with MD20150 (default setting).
1	The last active working area group remains active.

3.7.4 Example: Working area limitation in WCS/SZS

Assumption

Channel axes

Four axes are defined in the channel:

- Linear axes: X, Y, Z
- Rotary axis: A (not modulo)

Requirements

Channel axes

Four axes are defined in the channel:

- Linear axes: X, Y, Z
- Rotary axis: A (not modulo)

Working area limitation groups

Three working area limitation groups should be available in the channel:

MD28600 $MC_MM_NUM_WORKAREA_CS_GROUP = 3

From these three working area limitation groups, two groups are defined in the following.

Coordinate systems

- Working area limitation group 1: Working area limitation in the adjustable zero system (**AZS**).
- Working area limitation group 2: Working area limitation in the workpiece coordinate system (**WCS**).

Working area limitation group 1

- X axis in the plus direction: 10 mm
- X axis in the minus direction: No limitation
- Y axis in the plus direction: No limitation
- Y axis in the minus direction: 25 mm
- Z axis in the plus direction: No limitation
- Z axis in the minus direction: No limitation
- A axis in the plus direction: 10 degrees
- A axis in the minus direction: -40 degrees

Definitions via system variables in the NC program.

Program code

```
; Working area limitation group 1
$P_WORKAREA_CS_COORD_SYSTEM[1] = 3        ; working area limitation in the
SZS
$P_WORKAREA_CS_PLUS_ENABLE[1,X] = TRUE
$P_WORKAREA_CS_LIMIT_PLUS[1,X] = 10
$P_WORKAREA_CS_MINUS_ENABLE[1,X] = FALSE
$P_WORKAREA_CS_PLUS_ENABLE[1,Y] = FALSE
$P_WORKAREA_CS_MINUS_ENABLE[1,Y] = TRUE
$P_WORKAREA_CS_LIMIT_MINUS[1,Y] = 25
$P_WORKAREA_CS_PLUS_ENABLE[1,Z] = FALSE
$P_WORKAREA_CS_MINUS_ENABLE[1,Z] = FALSE
```

Program code

```
$P_WORKAREA_CS_PLUS_ENABLE[1,A] = TRUE
$P_WORKAREA_CS_LIMIT_PLUS[1,A] = 10
$P_WORKAREA_CS_MINUS_ENABLE[1,A] = TRUE
$P_WORKAREA_CS_LIMIT_MINUS[1,A] = -40
```

Working area limitation group 2

- X axis in the plus direction: 10 mm
- X axis in the minus direction: No limitation
- Y axis in the plus direction: 34 mm
- Y axis in the minus direction: -25 mm
- Z axis in the plus direction: No limitation
- Z axis in the minus direction: -600 mm
- A axis in the plus direction: No limitation
- A axis in the minus direction: No limitation

Definitions via system variables in the NC program.

Program code

```
; Working area limitation group 2
$P_WORKAREA_CS_COORD_SYSTEM[2] = 1        ; working area limitation in the
WCS
$P_WORKAREA_CS_PLUS_ENABLE[2,X] = TRUE
$P_WORKAREA_CS_LIMIT_PLUS[2,X] = 10
$P_WORKAREA_CS_MINUS_ENABLE[2,X] = FALSE
$P_WORKAREA_CS_PLUS_ENABLE[2,Y] = TRUE
$P_WORKAREA_CS_LIMIT_PLUS[2,Y] = 34
$P_WORKAREA_CS_MINUS_ENABLE[2,Y] = TRUE
$P_WORKAREA_CS_LIMIT_MINUS[2,Y] = –25
$P_WORKAREA_CS_PLUS_ENABLE[2,Z] = FALSE
$P_WORKAREA_CS_MINUS_ENABLE[2,Z] = TRUE
$P_WORKAREA_CS_LIMIT_PLUS[2,Z] = –600
$P_WORKAREA_CS_PLUS_ENABLE[2,A] = FALSE
$P_WORKAREA_CS_MINUS_ENABLE[2,A] = FALSE
```

Activation

The working area limitation groups are activated in the NC program using command `WALCS<x>`, with x: Number of the working area limitation group

3.8 Parking a machine axis

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

Activation / deactivation

Activate parking

The "Parking" function is **activated** for a machine axis by **resetting** the axis-specific NC/PLC interface signals for the position measuring systems and the controller enable:

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

The encoder status of the position measuring system of the axis is then set to "Not referenced":

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

The following further NC/PLC interface signals are also reset:

- DB31, ... DBX61.5 = 0 (position controller active)
- DB31, ... DBX61.6 = 0 (speed controller active)
- DB31, ... DBX61.7 = 0 (current controller active)
- DB31, ... DBX93.7 = 0 (pulses enabled)
- DB31, ... DBX102.5 = 0 (position measuring system 1 activated)
- DB31, ... DBX102.6 = 0 (position measuring system 2 activated)

Deactivate parking

The "Parking" function is **deactivated** for a machine axis by **setting** the axis-specific NC/PLC interface signals for the position measuring system to be activated and the controller enable:

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

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

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

- Incremental position measuring system ⇒ "not referenced" state
  - DB31, ... DBX60.4 = 0 (referenced / synchronized, position measuring system 1)
  - DB31, ... DBX60.5 = 0 (referenced/synchronized, position measuring system 2)
- Absolute position measuring system ⇒ "referenced/synchronized" state
  - DB31, ... DBX60.4 = 1 (referenced / synchronized, position measuring system 1)
  - DB31, ... DBX60.5 = 1 (referenced/synchronized, position measuring system 2)

The following NC/PLC interface signals are also set again:

- DB31, ... DBX61.5 = 1 (position controller active)
- DB31, ... DBX61.6 = 1 (speed controller active)
- DB31, ... DBX61.7 = 1 (current controller active)
- DB31, ... DBX93.7 = 1 (pulses enabled)
- DB31, ... DBX102.5 = 1 (position measuring system 1 activated)
- DB31, ... DBX102.6 = 1 (position measuring system 2 activated)

Incremental position measuring systems

After deactivation of the "parking" state, incremental position measuring systems have to be referenced before they have "referenced" encoder status.

WARNING

Incorrect synchronization of the position measuring system caused by offset of the actual machine axis position

If changes have been made to the position measuring system during "parking" that require a change to the parameterized machine data, for example, another encoder has been mounted, the position measuring system must be completely remeasured and referenced See Section "R1: Referencing (Page 1223)."

Machine axis without position measuring system

For a machine axis without a position measuring system (speed-controlled spindle), a status equivalent to "parking" is activated by canceling the controller enable:

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

3.9 Parking the passive position measuring system

3.9.1 Function

Contrary to the "Parking a machine axis (Page 125)" function, for which all position measuring systems of a machine axis are deactivated, using the "Park the passive position measuring system" function, users have the option, of only "parking" the passive position measuring system of a machine axis (i.e. to deactivate the encoder evaluation and monitoring in the drive and in the control), while the active position measuring system can remain operational.

Note

For an explanation of active/passive measuring system, refer to "Setpoint/actual-value system (Page 362)".

Application

The "Park passive measuring system" function can, for example, be used in the following cases:

- Changing attachment heads with and without integrated encoder
  Using the "Park passive position measuring system" function, it is possible to mount attachment heads alternating with and without integrated encoder for different machining tasks on the main spindle, without the missing encoder signals initiating drive and control faults.
  See also:
  – Example: Changing an attachment head for a direct position measuring system (Page 131)
  – Example: Changing an attachment head for two direct position measuring systems (Page 136)
- Using linear position measuring systems, which are not available over the complete traversing range of a machine axis
  Using the "Park passive position measuring system" function, it is possible to pass through the range outside the linear position measuring system, without the missing encoder signals initiating drive and control faults.
  See also:
  – "Example: Measuring system switchover when encoders are missing in certain parts of the range (Page 140)".

Activation / deactivation

Activation

The passive position measuring system of a machine axis is parked under the following conditions:

- "Park passive position measuring system" function is active for the measuring system:
  MD31046 $MA_ENC_PASSIVE_PARKING[<n>] = 1
  with <n> = 0 (position measuring system 1) or 1 (position measuring system 2)

  Note

  MD31046 is **not active**:

  - for axes with fewer than two encoders:
    MD30200 $MA_NUM_ENCS < 2
  - for simulated encoders:
    MD30240 $MA_ENC_TYPE = 0

  Note

  For position measuring systems, which are used as motor measuring systems, the "Park passive position measuring system" function should be deactivated (MD31046 = 0)!

  and

- The user sets the following NC/PLC interface signal to "0":
  DB31, ... DBX1.5 (position measuring system 1) = 0
  or
  DB31, ... DBX1.6 (position measuring system 2) = 0

The controller then sets the status for the activation state of the position measuring system to "0":

DB31, ... DBX102.5 (position measuring system 1 activated) == 0

or

DB31, ... DBX102.6 (position measuring system 2 activated) == 0

The position measuring system is now no longer monitored and updated.

Deactivation

"Park" is deactivated if the user activates the position measuring system:

DB31, ... DBX1.5 (position measuring system 1) = 1

or

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

The controller then sets the status for the activation state of the position measuring system back to "1":

DB31, ... DBX102.5 (position measuring system 1 activated) == 1

or

DB31, ... DBX102.6 (position measuring system 2 activated) == 1

Note

Switching over to a parked position measuring system takes longer than to a non-parked position measuring system. Because of the time taken, we recommend switching over while the axes are stationary.

Position of the position measuring system

Absolute position measuring systems

For absolute position measuring systems, the position after deactivating "park" corresponds to the actual absolute encoder position.

The position measuring system is referenced:

DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 1

or

DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 1

Incremental position measuring systems

For incremental position measuring systems, the position after deactivating "park" always corresponds to the last deactivation position of the position measuring system.

Switching over to the parked position measuring system is only realized if the parameterized permissible deviation between the actual values of the two position measuring systems (see MD36500 $MA_ENC_CHANGE_TOL) is not exceeded. Otherwise, users must apply the "Park a machine axis" function, in which case such a check is not made.

The position measuring system is **not** referenced:

DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 0

or

DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 0

Incremental position measuring systems with position transfer

Alternatively, for incremental position measuring systems, when the "Park passive position measuring system" is active (MD31046 $MA_ENC_PASSIVE_PARKING[<n>] = **1**) it is possible, after deactivating "Park" to transfer the position, and where relevant, also the "Referenced" status, from the previously active position measuring system.

For every position measuring system of a machine axis, this function can be activated using machine data:

MD34210 $MA_ENC_REFP_STATE[<n>]

with <n> = 0 (position measuring system 1) or 1 (position measuring system 2)

Value	Meaning
1	Only the position from the previous active position measuring system is transferred. The position measuring system is **not** referenced: DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 0 or DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 0
2	Position **and** "Referenced" status are transferred from the previously active position measuring system. The position measuring system is referenced: DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 1 or DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 1

Note

Position and "Referenced" status of the previously active position measuring system are only transferred for incremental position measuring systems, depending on MD34210 $MA_ENC_REFP_STATE[<n>] – and only when the "Park passive position measuring system" function is active (MD31046 $MA_ENC_PASSIVE_PARKING[<n>] = 1).

The transferred position has the accuracy of the previous active position measuring system. The position measuring system should be rereferenced if this accuracy is not adequate.

⚠ WARNING

Incorrect synchronization of the position measuring system caused by offset of the actual machine axis position

If changes have been made to the position measuring system during "parking" that require a change to the parameterized machine data, for example, another encoder has been mounted, the position measuring system must be completely remeasured and referenced See Chapter "R1: Referencing (Page 1223)".

3.9.2 Supplementary conditions

Interaction with "dual position feedback"

The "Park passive position measuring system" function **cannot** be used in conjunction with the "dual position feedback" function (MD32960 $MA_POSCTRL_DUAL_FEEDBACK_TIME > 0).

Interaction with "position difference input"

The "Park passive position measuring system" function **cannot** be used in conjunction with the "position difference input" function (MD32950 $MA_POSCTRL_DAMPING > 0).

Interaction with APC (option for SINUMERIK 840D sl)

The "Park passive measuring system" function **cannot**be used in conjunction with the "Advanced positioning control (APC)" drive function.

Interaction with encoder safety protection concept

Only the 1 encoder safety protection concept can be used in conjunction with the "Park passive position measuring system" function.

Interaction when connecting and disconnecting at the DRIVE-CLiQ

If, instead of the encoder cable, the DRIVE-CLiQ cables, e.g. between the SMC and Motor Module are unplugged and plugged, such encoders can only be unparked without fault via the "Parking a machine axis (Page 125)" function.

3.9.3 Example: Changing an attachment head for a direct position measuring system

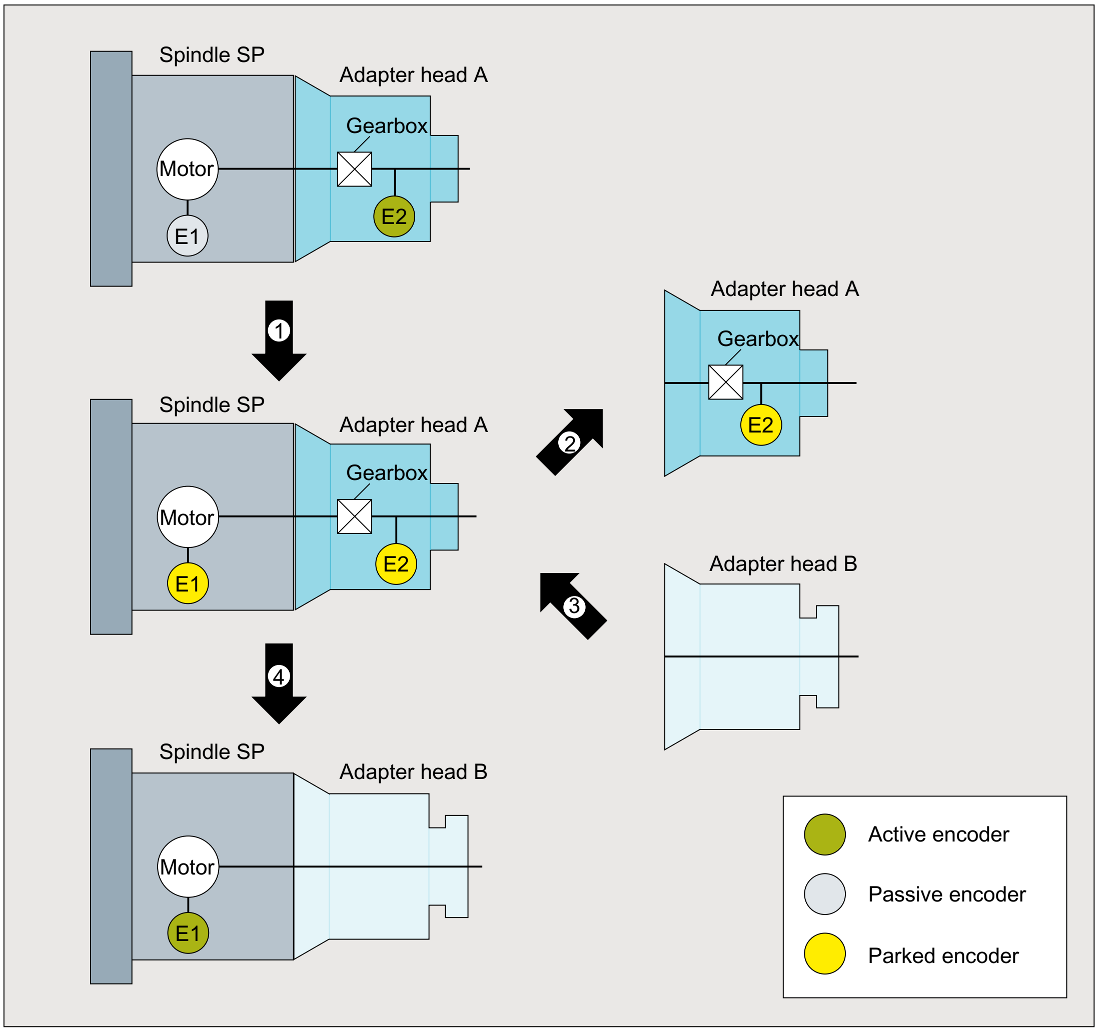
Initial situation

- Attachment head "A" has an encoder E2.
- Attachment head "B" does not have an encoder.
- Spindle "SP" has an encoder E1.
- One of the following two telegram types is configured in MD13060 $MN_DRIVE_TELEGRAM_TYPE (standard telegram type for PROFIdrive):
  - Telegram 116 (motor encoder + an external encoder)
    or
  - Telegram 136 (motor encoder + an external encoder, and torque precontrol)
- The following position measuring systems are configured in the spindle "SP":
  - Motor encoder E1 as position measuring system 1
  - Direct encoder E2 as position measuring system 2
- Attachment head "A" with encoder E2 is currently mounted on the spindle.
- Position measuring system 2 is the active measuring system:
  DB31, ... DBX1.6 = 1
  Position measuring system 1 is passive.
- The "Park passive position measuring system" function is:
  - Not active for position measuring system 1:
    MD31046 $MA_ENC_PASSIVE_PARKING [0] = 0
  - Active for position measuring system 2:
    MD31046 $MA_ENC_PASSIVE_PARKING [1] = 1

Objective

The user would like to change from attachment head "A" to attachment head "B."

Execution



① Before changing an attachment head, the user must deactivate all position measuring systems of the machine axis using the "Parking a machine axis (Page 125)" function:

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

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

The controller then resets the status signals for the position measuring systems:

DB31, ... DBX102.5 (position measuring system 1 activated) == 0

DB31, ... DBX102.6 (position measuring system 2 activated) == 0

② The user waits for the status signals and only now removes the attachment head "A" from the spindle. This electrically disconnects the encoder cable between the attachment head "A" and coupling. The absence of encoder E2 does not generate any NC or drive faults.

③ Attachment head "B" is now mounted on the spindle.

④ The user only activates position measuring system 1:

DB31, ... DBX1.5 (position measuring system 1) = 1

The control then sets the status signal:

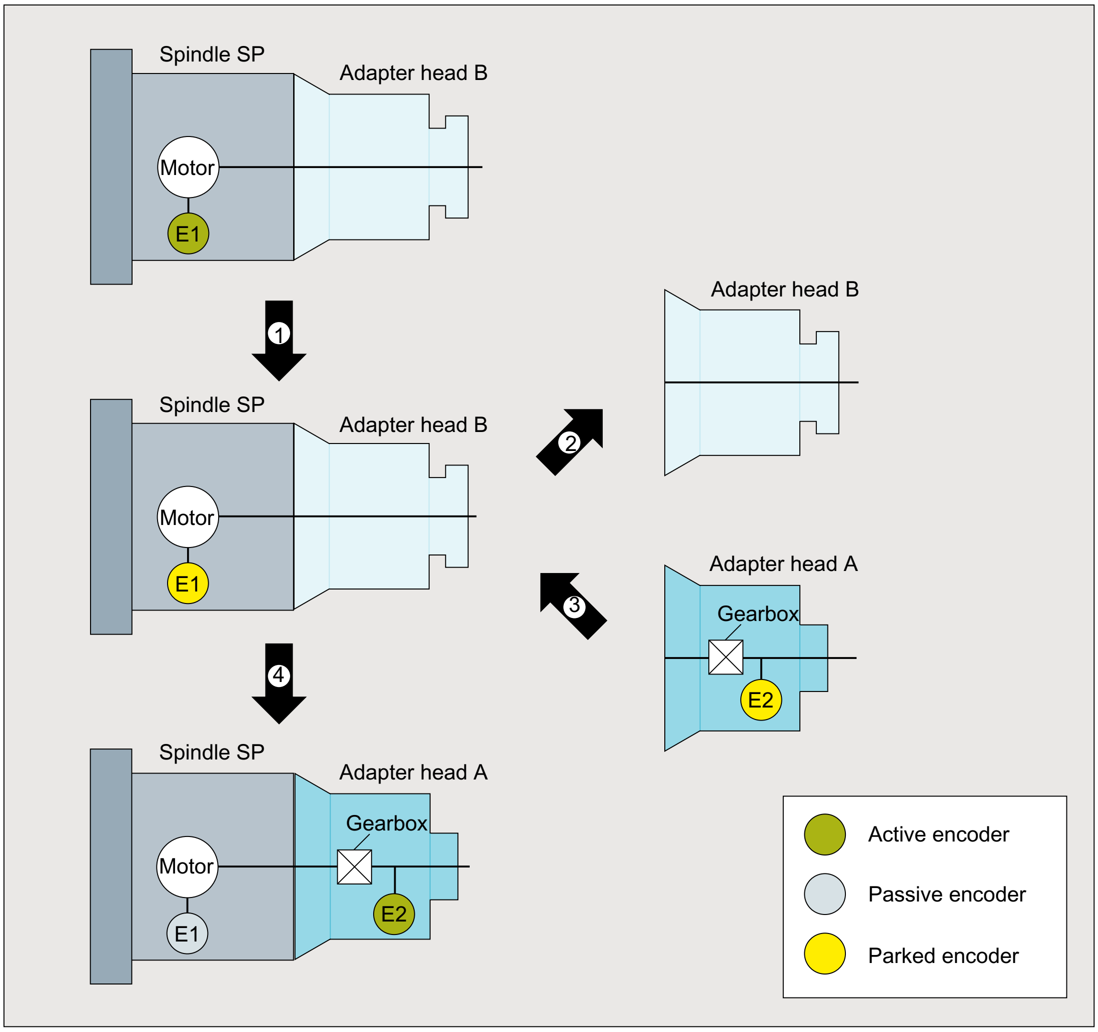
DB31, ... DBX102.5 (position measuring system 1 activated) == 1

The "Park passive position measuring system" function is active for position measuring system 2. This means that position measuring system 2 does not become passive, but remains in the "Park" state.

Objective

The user now wishes to remount attachment head "A" again.

Execution



① Using the "Park machine axis" function, the user deactivates position measuring system 1:
DB31, ... DBX1.5 (position measuring system 1) = 0
The controller then resets the status signal for the position measuring system:
DB31, ... DBX102.5 (position measuring system 1 activated) == 0

② The user waits for the status signal and only now removes the attachment head "B" from the spindle.

③ Attachment head "A" is now mounted on the spindle.

④ The user activates position measuring system 2:
DB31, ... DBX1.6 (position measuring system 2) = 1
As a consequence, position measuring system 1 is also simultaneously activated; This is because the "Park passive position measuring system" function is not active for position measuring system 1 (motor measuring system!). Position measuring system 1 becomes a passive position measuring system.
The controller then sets the status signals for the position measuring systems:
DB31, ... DBX102.5 (position measuring system 1 activated) == 1
DB31, ... DBX102.6 (position measuring system 2 activated) == 1

3.9.4 Example: Changing an attachment head for two direct position measuring systems

Initial situation

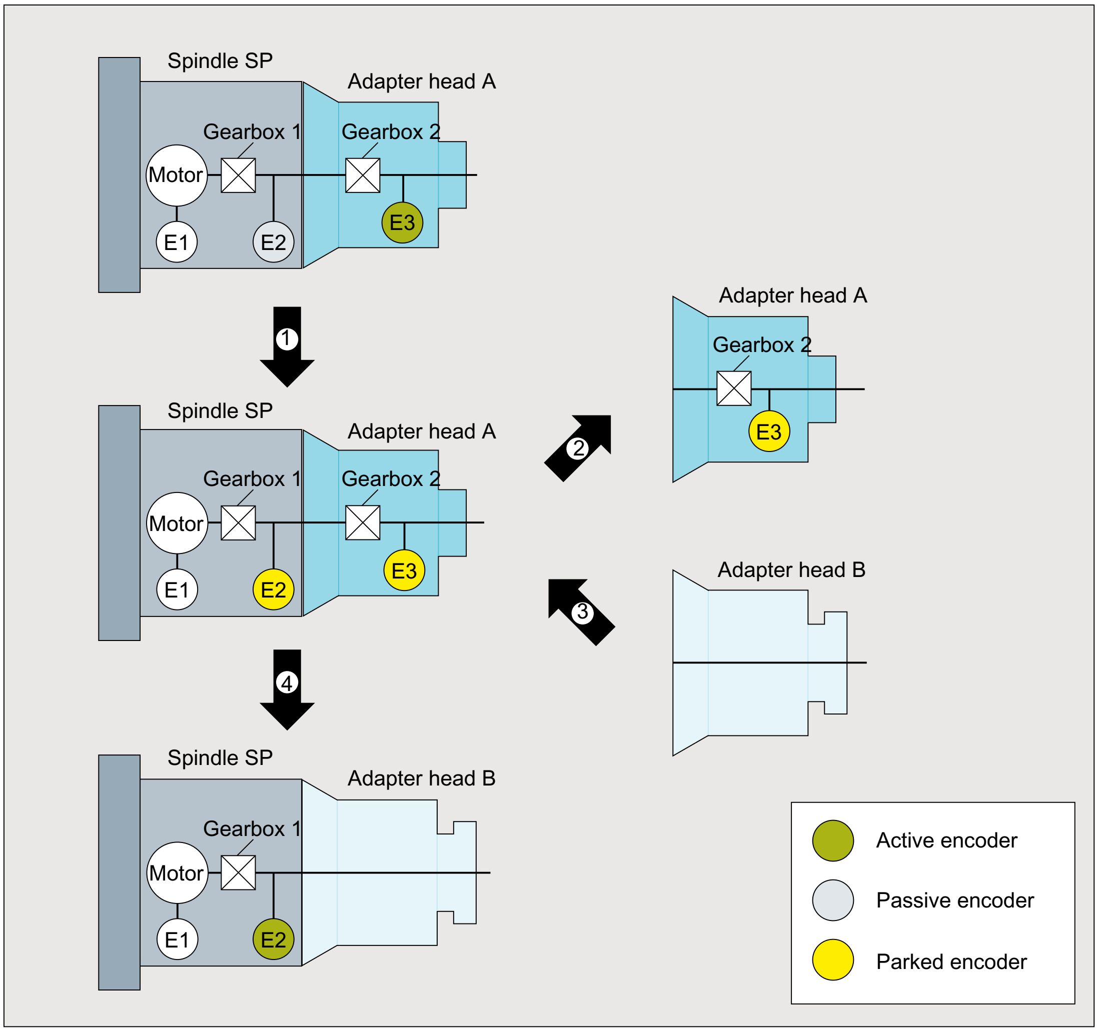
- Attachment head "A" has an encoder E3.
- Attachment head "B" does not have an encoder.
- Spindle "SP" has two encoders E1 and E2.
- One of the following two telegram types is configured in MD13060 $MN_DRIVE_TELEGRAM_TYPE (standard telegram type for PROFIdrive):
  - Telegram 118 (two external encoders)
    or
  - Telegram 138 (two external encoders, plus torque precontrol)

  References:
  For information with regard to the encoder assignment, see:
  Commissioning Manual, CNC Commissioning: NC, PLC, Drive;
  Section: "Communication between the NC and the drive" > "Drives: Assign axis"
- The following position measuring systems are configured in the spindle "SP":
  - Direct encoder E2 as position measuring system 1
  - Direct encoder E3 as position measuring system 2
- Attachment head "A" with encoder E3 is currently mounted on the spindle.
- Position measuring system 2 is the active measuring system:
  DB31, ... DBX1.6 = 1
  Position measuring system 1 is passive.
- The "Park passive position measuring system" function is:
  - Not active for position measuring system 1:
    MD31046 $MA_ENC_PASSIVE_PARKING [0] = 0
  - Active for position measuring system 2:
    MD31046 $MA_ENC_PASSIVE_PARKING [1] = 1

Objective

The user would like to change from attachment head "A" to attachment head "B."

Execution



① Before changing an attachment head, the user must deactivate all position measuring systems of the machine axis using the "Parking a machine axis (Page 125)" function:

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

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

The controller then resets the status signals for the position measuring systems:

DB31, ... DBX102.5 (position measuring system 1 activated) == 0

DB31, ... DBX102.6 (position measuring system 2 activated) == 0

② The user waits for the status signals and only now removes the attachment head "A" from the spindle. This electrically disconnects the encoder cable between the attachment head "A" and coupling. The absence of encoder E3 does not generate any NC or drive faults.

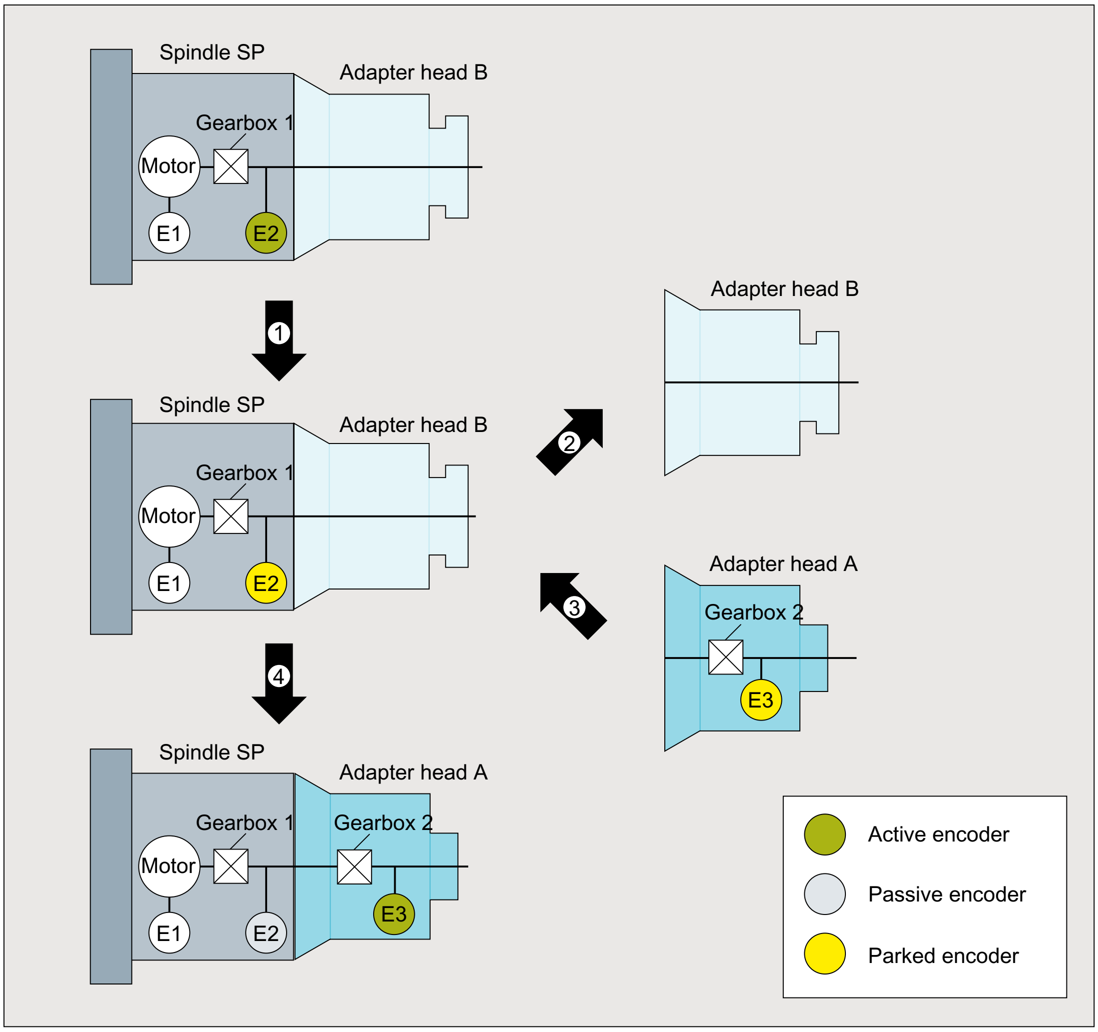
③ Attachment head "B" is now mounted on the spindle.

④ The user only activates position measuring system 1:
DB31, ... DBX1.5 (position measuring system 1) = 1
The control sets the status signal:
DB31, ... DBX102.5 (position measuring system 1 activated) == 1
The "Park passive position measuring system" function is active for position measuring system 2. This means that position measuring system 2 does not become passive, but remains in the "Park" state.

Objective

The user now wishes to remount attachment head "A" again.

Execution



① Using the "Park machine axis" function, the user deactivates position measuring system 1:
DB31, ... DBX1.5 (position measuring system 1) = 0
The controller then resets the status signal for the position measuring system:
DB31, ... DBX102.5 (position measuring system 1 activated) == 0

② The user waits for the status signal and only now removes the attachment head "B" from the spindle.

③ Attachment head "A" is now mounted on the spindle.

④ The user activates position measuring system 2:
DB31, ... DBX1.6 (position measuring system 2) = 1
As a consequence, position measuring system 1 is also simultaneously activated; This is because the "Park passive position measuring system" function is not active for position measuring system 1. Position measuring system 1 becomes a passive position measuring system.
The controller then sets the status signals for the position measuring systems:
DB31, ... DBX102.5 (position measuring system 1 activated) == 1
DB31, ... DBX102.6 (position measuring system 2 activated) == 1

3.9.5 Example: Measuring system switchover when encoders are missing in certain parts of the range

In the following example, the direct linear position measuring system is only available in the machining zones, while only the motor measuring system is available outside the machining zones.

Initial situation

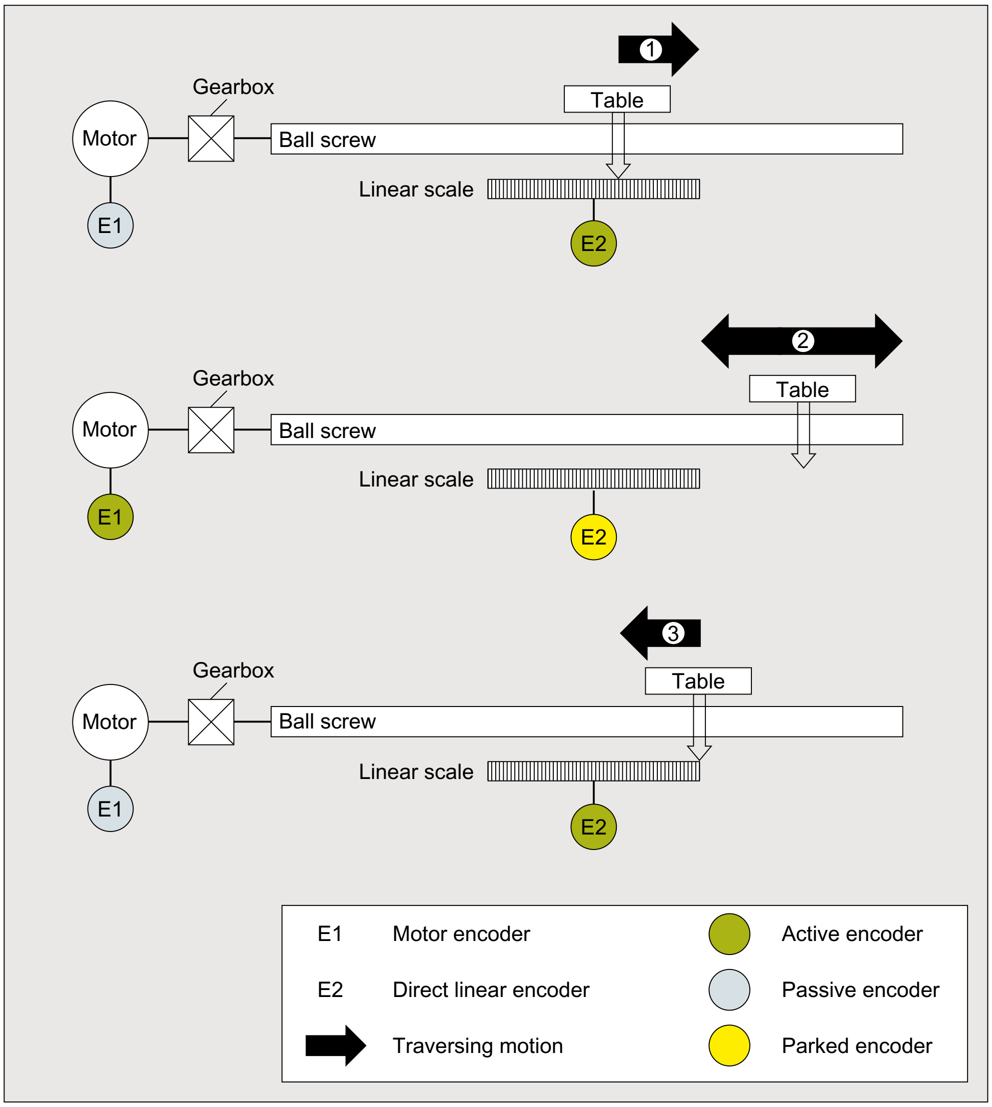
- Linear axis "X" has two incremental encoders:
  - Motor encoder E1
  - Direct linear encoder E2
- Direct linear encoder E2 is only available in the machining range.
- One of the following two telegram types is configured in MD13060 $MN_DRIVE_TELEGRAM_TYPE (standard telegram type for PROFIdrive):
  - Telegram 116 (motor encoder + an external encoder)
    or
  - Telegram 136 (motor encoder + an external encoder, and torque precontrol)
- The following position measuring systems are configured for the machine axis:
  - Motor encoder E1 as position measuring system 1
  - Direct linear encoder E2 as position measuring system 2
- When the machine is switched on, the user activates both position measuring systems:
  - DB31, ... DBX1.5 (position measuring system 1) = 1
  - DB31, ... DBX1.6 (position measuring system 2) = 1

  When both position measuring systems are simultaneously activated, then the control selects position measuring system 1 as active position measuring system.
- Position measuring system 2 is selected for machining:
  - DB31, ... DBX1.5 (position measuring system 1) = 0
  - DB31, ... DBX1.6 (position measuring system 2) = 1
- The "Park passive position measuring system" function is:
  - Not active for position measuring system 1:
    MD31046 $MA_ENC_PASSIVE_PARKING [0] = 0
  - Active for position measuring system 2:
    MD31046 $MA_ENC_PASSIVE_PARKING [1] = 1
- Transfer of the position and the "referenced" status is activated for position measuring system 2:
  MD34210 $MA_ENC_REFP_STATE[1] = 2

Objective

When passing through the range outside linear position measuring system E2, the missing encoder signals should not initiate any faults in the drive and in the control.

Execution



① Before the table reaches the end of the linear position measuring system, a switchover must be made to the motor measuring system. The user does this by activating both position measuring systems:

DB31, ... DBX1.5 (position measuring system 1) = 1

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

Using the "Park passive position measuring system" function, the linear position measuring system, which is passive after the measuring system switchover, is parked by the control.

The control resets the status signal:

DB31, ... DBX102.6 (position measuring system 2 activated) == 0

The user waits for the status signal before continuing to traverse in the range outside the linear position measuring system.

② Traversing through the range outside the linear position measuring system with the motor measuring system.

③ If the table returns to the linear position measuring system range, at standstill, the user switches from the motor measuring system to the linear position measuring system:

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

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

The control sets the status signal:

DB31, ... DBX102.6 (position measuring system 2 activated) == 1

The motor measuring system becomes a passive position measuring system.

Result

Both position measuring systems are referenced. The position of the linear position measuring system corresponds to the position of the motor measuring system at the switchover instant. The linear position measuring system must be rereferenced if the position accuracy is not sufficient.

3.10 Switching over encoder data sets

Application

Different attachment heads may be used in succession on one and the same power unit in order to perform different machining tasks.

One motor data set (MDS) and one encoder data set (EDS) must be programmed for each attachment head that is equipped with a motor or an encoder. These data sets must be switched over in the PLC program whenever the attachment head is changed. An MDS/EDS switchover can only be implemented indirectly in this regard via switchover of the drive data set (DDS).

Function

NOTICE

Machine damage

If the following drive parameters and machine data have different settings, the axis might not traverse at the programmed speed or to the programmed position.

The machine data must be changed simultaneously and consistently with encoder data switchover in the parked state.

Encoder data switchover in the control is limited to the following SINAMICS drive parameters.

- p0408 (rotary encoder pulse number)
- p0418 (fine resolution of encoder emulation Gx_XIST1 (in bits))
- p0419 (fine resolution absolute value Gx_XIST2 (in bits))

With these parameters, it is possible to switch over encoder data sets with the same encoder type but a different number of encoder pulses.

Switchover in the control is activated in the following machine data:

MD31700 $MA_ENC_EDS_ACTIVE (activate EDS use)

Value	Meaning
0	Encoder data set switchover EDS is not used.
1.	Encoder data set switchover EDS is used.

A machine data item is provided for each of the drive parameters p0408, p0418, and p0419, which must be parameterized in accordance with the active encoder data set:

- MD31710 $MA_ENC_RESOL_EDS (encoder pulses per revolution for EDS use)
- MD31720 $MA_ENC_PULSE_MULT_EDS (encoder multiplication (high resolution) for EDS use)
- MD31730 $MA_ABS_INC_RATIO_EDS (absolute encoder: ratio between absolute resolution and incremental resolution for EDS use)

Effectiveness

If MD31700 $MA_ENC_EDS_ACTIVE = 1, the following machine data no longer take effect:

- MD30260 $MA_ABS_INC_RATIO (absolute encoder: ratio between absolute resolution and incremental resolution)
- MD31020 $MA_ENC_RESOL (encoder pulses per revolution)
- MD31025 $MA_ENC_PULSE_MULT (encoder multiplication (high resolution))

The drive data switchover in DB3x.DBX21.0 - 4 also switches over the encoder data sets. Switchover is performed in the parked state (see Chapter "Parking a machine axis (Page 125)" and "Parking the passive position measuring system (Page 127)").

Note

If MD31700 $MA_ENC_EDS_ACTIVE = 1, no plausibility check between the values set in the drive and control is performed.

NC/PLC interface signal:

- DB31, ... DBX21.7 (pulse enable)
- DB31, ... DBX21.6 (integrator inhibit, speed controller)
- DB31, ... DBX21.5 (motor selection)
- DB31, ... DBX21.0 - 4 (request for switchover of a motor and/or drive data set)
  The interface can be flexibly parameterized using: DB31, ...DBX130.0 - 4

Supplementary conditions

SINUMERIK supplementary conditions

- Only machine data for rotary encoders are available.
- Traversing range extension for absolute-value encoders
  MD30270 $MA_ENC_ABS_BUFFERING = 0 is not permissible.

SINAMICS supplementary conditions are described in the following documentation.

References

SINAMICS S120 Function Manual Drive Functions;
"Basic information about the drive system" > "Data sets"

- Chapter "DDS: Drive Data Set"
- Chapter "EDS: Encoder Data Set"

Other machine data

In addition to the machine data of the axis monitoring function, the following additional machine data should be set or checked:

All machine axes

- 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: see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 343)".

Machine axes with analog speed setpoint interface

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

3.11 Data lists

3.11.1 Machine data

3.11.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10604	WALIM_GEOAX_CHANGE_MODE	Working area limitation during switchover of geometry axes
10710	PROG_SD_RESET_SAVE_TAB	Setting data to be updated

3.11.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of the G groups
21020	WORKAREA_WITH_TOOL_RADIUS	Allowance for tool radius with working area limitation
24130	TRAFO_INCLUDES_TOOL_1	Tool handling with active 1st transformation
24230	TRAFO_INCLUDES_TOOL_2	Tool handling with active 2nd transformation
24330	TRAFO_INCLUDES_TOOL_3	Tool handling with active 3rd transformation
24426	TRAFO_INCLUDES_TOOL_4	Tool handling with active 4th transformation
24436	TRAFO_INCLUDES_TOOL_5	Tool handling with active 5th transformation
24446	TRAFO_INCLUDES_TOOL_6	Tool handling with active 6th transformation
24456	TRAFO_INCLUDES_TOOL_7	Tool handling with active 7th transformation
24466	TRAFO_INCLUDES_TOOL_8	Tool handling with active 8th transformation
24476	TRAFO_INCLUDES_TOOL_9	Tool handling with active 9th transformation
24486	TRAFO_INCLUDES_TOOL_10	Tool handling with active 10th transformation
25106	TRAFO_INCLUDES_TOOL_11	Tool handling with active 11th transformation
25116	TRAFO_INCLUDES_TOOL_12	Tool handling with active 12th transformation
25126	TRAFO_INCLUDES_TOOL_13	Tool handling with active 13th transformation
25136	TRAFO_INCLUDES_TOOL_14	Tool handling with active 14th transformation
25146	TRAFO_INCLUDES_TOOL_15	Tool handling with active 15th transformation
25156	TRAFO_INCLUDES_TOOL_16	Tool handling with active 16th transformation
25166	TRAFO_INCLUDES_TOOL_17	Tool handling with active 17th transformation
25176	TRAFO_INCLUDES_TOOL_18	Tool handling with active 18th transformation
25186	TRAFO_INCLUDES_TOOL_19	Tool handling with active 19th transformation
25196	TRAFO_INCLUDES_TOOL_20	Tool handling with active 20th transformation
28600	MM_NUM_WORKAREA_CS_GROUPS	Number of coordinate system-specific working area limitations

3.11.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30200	NUM_ENCS	Number of encoders
30240	ENC_TYPE	Encoder type of the actual value acquisition (actual position value)
30260	ABS_INC_RATIO	Absolute encoder: ratio between absolute value and incremental value resolution
30270	ENC_ABS_BUFFERING	Absolute encoder: Traversing range extension
30310	ROT_IS_MODULO	Modulo conversion for rotary axis / spindle
30800	WORK_AREA_CHECK_TYPE	Type of checking of working area limits
31020	ENC_RESOL	Encoder pulses per revolution
31025	ENC_PULSE_MULT	Encoder multiplication (high resolution)
31046	ENC_PASSIVE_PARKING	Parking the passive position measuring system
31700	ENC_EDS_ACTIVE	Activate EDS use
31710	ENC_RESOL_EDS	Encoder pulses per revolution for EDS use
31720	ENC_PULSE_MULT_EDS	Encoder multiplication (high resolution) for EDS use
31730	ABS_INC_RATIO_EDS	Absolute encoder: Ratio between the absolute resolution and the incremental resolution for EDS use
32200	POSCTRL_GAIN [n]	K_V factor
32250	RATED_OUTVAL	Rated output voltage
32260	RATED_VELO	Rated motor speed
32300	MAX_AX_ACCEL	Maximum axis acceleration
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feedforward control
32910	DYN_MATCH_TIME [n]	Time constant for dynamic response adaptation
34210	ENC_REFP_STATE	Encoder status
35160	SPIND_EXTERN_VELO_LIMIT	Spindle speed limitation via PLCC
36000	STOP_LIMIT_COARSE	Exact stop coarse
36010	STOP_LIMIT_FINE	Exact stop fine
36020	POSITIONING_TIME	Delay time exact stop fine
36030	STANDSTILL_POS_TOL	Zero speed tolerance
36040	STANDSTILL_DELAY_TIME	Delay time zero-speed monitoring
36050	CLAMP_POS_TOL	Clamping tolerance with IS "Clamping active"
36052	STOP_ON_CLAMPING	Special functions for clamped axis
36060	STANDSTILL_VELO_TOL	Maximum velocity/speed "Axis/spindle stationary"
36100	POS_LIMIT_MINUS	1st software limit switch minus
36110	POS_LIMIT_PLUS	1st software limit switch plus
36120	POS_LIMIT_MINUS2	2nd software limit switch minus
36130	POS_LIMIT_PLUS2	2nd software limit switch plus
36200	AX_VELO_LIMIT	Threshold value for velocity monitoring
36210	CTRLOUT_LIMIT	Maximum speed setpoint
36220	CTRLOUT_LIMIT_TIME	Delay time for speed-setpoint monitoring
36300	ENC_FREQ_LIMIT	Encoder limit frequency

Number	Identifier: $MA_	Description
36302	ENC_FREQ_LIMIT_LOW	Encoder limit frequency for encoder resynchronization
36310	ENC_ZERO_MONITORING	Zero mark monitoring
36312	ENC_ABS_ZEROMON_WARNING	Zero-mark monitoring warning threshold
36400	CONTOUR_TOL	Tolerance band contour monitoring
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover
36510	ENC_DIFF_TOL	Measuring system synchronism tolerance
36600	BRAKE_MODE_CHOICE	Braking behavior at hardware limit switch
36610	AX_EMERGENCY_STOP_TIME	Maximum duration of the braking ramp for faults
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay controller enable

3.11.2 Setting data

3.11.2.1 Axis/spindlespecific setting data

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

3.11.3 Signals

3.11.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Follow-up mode	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1 / 2	DB31, ...DBX1.5/6	DB380x.DBX1.5/6
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Clamping in progress	DB31, ...DBX2.3	DB380x.DBX2.3
Velocity/spindle speed limitation	DB31, ...DBX3.6	DB380x.DBX3.6
Feed stop/spindle stop	DB31, ...DBX4.3	DB380x.DBX4.3
Hardware limit switch minus / hardware limit switch plus	DB31, ...DBX12.0/1	DB380x.DBX1000.0/1
Software limit switch minus / 2nd software limit switch plus	DB31, ...DBX12.2/3	DB380x.DBX1000.2/3
Motor/drive data set:	DB31, ...DBX21.0 - 4	DB380x.DBX4001.0 - 4
Motor being selected	DB31, ...DBX21.5	DB380x.DBX4001.5
Speed controller integrator disable	DB31, ...DBX21.6	DB380x.DBX4001.6
Pulse enable	DB31, ...DBX21.7	DB380x.DBX4001.7

3.11.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Encoder limit frequency exceeded 1 / 2	DB31, ...DBX60.2/3	DB390x.DBX0.2
Referenced/synchronized 1/2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Travel command minus/plus	DB31, ...DBX64.6/7	DB390x.DBX4.6/7
Position measuring system 1/2 activated	DB31, ... DBX102.5/6	DB390x.DBX5006.5/6
Motor/drive data set:	DB31, ...DBX130.0 - 4	DB390x.DBX4008.0 - 4

A5: Protection zones 4

4.1 Function

Protection zones are static or moveable in 2- or 3-dimensional ranges within a machine to protect machine elements against collisions - and must be defined by the user.

The following elements can be protected:

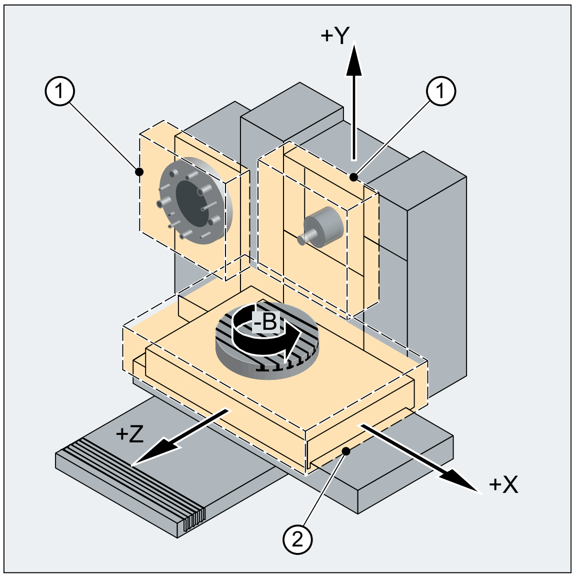
- Fixed machine elements (e.g. tool magazine, probe that can be swiveled).
- Moving machine elements that belong to the tool (e.g. tool, tool holder)
- Moving machine elements that belong to the workpiece (e.g. parts of the workpiece, clamping table, clamping shoe, spindle chuck, tailstock).

In order that it is guaranteed that the machine element is protected, the protection zones must be defined so that they completely envelope the element to be protected.

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

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

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



① **Tool**-related protection zone

② **Workpiece**-related protection zone

Figure 4-1 Example of protection zones on a milling machine

Defining protection zones

A protection zone can be 2 or 3 dimensional, and defined using polylines with a maximum of 10 corner points and arcs as contour elements. The definition can be made using commands in the part program (see "Defining protection zones (CPROTDEF, NPROTDEF) (Page 158)") or using system variables. All of the contour elements lie in the plane that can be selected with G17, G18 or G19.

3rd dimension

Extending the protection zone in the 3rd dimension can be limited between - ∞ and + ∞:

- -∞ to +∞
- -∞ up to the upper limit
- Lower limit to +∞
- Lower limit to upper limit

Coordinate system

The definition of a protection zone takes place with reference to the geometry axis of a channel and therefore in the basic coordinate system (BCS).

Reference

- Tool-related protection zones
  Coordinates for **tool**-related protection zones must be specified as absolute values referred to the **tool holder reference point F**.
- Workpiece-related protection zones
  Coordinates for **workpiece**-related protection zones must be specified as absolute values referred to the zero point of the **basic coordinate system (BCS)**.

Note

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

If no workpiece-oriented protection area is active, then there is no protection zone monitoring.

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 workpiece-related protection zones.

Machine/channel-specific protection zones

- Machine-specific protection zones
  Data for machine specific protection zones are defined once in the control. These protection zones can be activated by all channels.
- Channel-specific protection zones
  Data for channel-specific protection zones are defined in a channel. These protection zones can be activated only by this channel.

System variable

When the protection zones are defined using commands in the part program, the protection zone data is stored in system variables. The system variables can also be written directly so that the definition of protection areas can also be performed directly in the system variables. The same supplementary conditions apply for the definition of the contour of a protection zone as for a protection zone definition using commands in the part program (see "Defining protection zones (CPROTDEF, NPROTDEF) (Page 158)").

The protection zone definitions cover following system variables:

System variable	Type	Meaning	
$SN_PA_ACTIV_IMMED[<n>] $SC_PA_ACTIV_IMMED[<n>]	BOOL	Activation type The protection zone is active / not active immediately after the control system has been powered up and the axes referenced.	
		FALSE	Not immediately active
		TRUE	Immediately active
$SN_PA_T_W[<n>] $SC_PA_T_W[<n>]	INT	Protection zone type	
		0	Workpiece-related protection zone
		1	Reserved
		2	Reserved
		3	Tool-related protection zone
$SN_PA_ORI[<n>] $SC_PA_ORI[<n>]	INT	Orientation of the protection zone, i.e. polygon definition in the plane of:	
		0	1st and 2nd geometry axis
		1	3rd and 1st geometry axis
		2	2nd and 3rd geometry axis
$SN_PA_LIM_3DIM[<n>] $SC_PA_LIM_3DIM[<n>]	INT	Type of limitation in the 3rd dimension	
		0	No limitation
		1	Limitation in plus direction
		2	Limitation in minus direction
		3	Limitation in positive and negative directions
$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 minus direction in the 3rd dimension	
$SN_PA_CONT_NUM[<n>] $SC_PA_CONT_NUM[<n>]	INT	Number of valid contour elements	

System variable	Type	Meaning
$SN_PA_CONT_TYP[<n>, <i>] $SC_PA_CONT_TYP[<n>, <i>]	INT	Contour type[<i>], contour type (G1, G2, G3) of the <i>th 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
$SN_PA_CENT_ABS[<n>, <i>] $SC_PA_CENT_ABS[<n>, <i>]	REAL	Center point of the circular contour[<i>], absolute abscis-sa 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 = 1st protection zone The index "<i>" corresponds to the number of the contour element: 0 = 1st contour element The contour elements must be defined in ascending order.		

Note

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

Data of the protection zone definitions

Data storage

The protection zone definitions are stored in the following files:

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

Data backup

The protection zone definitions are saved in the following files:

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

Activating, preactivating and deactivating protection zones

Activation state

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

- Activated
- Preactivated

- Preactivated with conditional stop
- Deactivated

The activation state is always channel-specific even for machine specific protection zones.

Activating, preactivating and deactivating in the part program

The activation state of a protection zone can be changed in the part program at any time using commands (see "Activating/deactivating protection zones (CPROT, NPROT) (Page 161)").

Note

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

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

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

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

Preactivated with conditional stop

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

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

Activating using NC/PLC interface signals

Only protection zones that have been **preactivated** via the part program can be activated in the PLC user program via the NC/PLC interface signals:

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

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

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

Note

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

Deactivating using NC/PLC interface signals

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

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

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

Automatic deactivation for transformation change/geometry axis interchange

In the default setting, when changing a transformation, or for a geometry axis interchange, the active protection zones are automatically deactivated. On the other hand, if active protection zones should remain active, then the bit-coded machine data MD10618 $MN_PROTAREA_GEOAX_CHANGE_MODE must be correspondingly adapted (see "Machine data (Page 157)").

Activation state in special system states

Block search with calculation

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

Program test

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

NC RESET and end of program

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

Display of protection zone violations

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

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

Checking for protection zone violation

Function CALCPOSI can be used to check whether geometry axes can traverse a specified path without violating a protection zone (see "Checking for protection zone violation, working area limitation and software limit switches (CALCPOSI) (Page 165)").

4.2 Commissioning

4.2.1 Machine data

Memory requirements

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

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

Response for a transformation change/geometry axis interchange

The following machine data can be used to define as to whether, when changing a transformation or for a geometry axis interchange, active protection zones should be kept or deactivated:

MD10618 $MN_PROTAREA_GEOAX_CHANGE_MODE

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

4.3 Programming

4.3.1 Defining protection zones (CPROTDEF, NPROTDEF)

Protection zones, which protect machine elements against collisions, are defined in the part program in blocks. These contain the following elements:

1. Definition of the machining plane
   Before the actual protection zone definition, the machining plane must be selected, to which the contour description of the protection zone refers.
2. Start of the definition
   Depending on the particular NC command, either a channel-specific or machine-specific protection zone is created.
3. Contour description of the protection zone
   The contour of a protection zone is defined with traversing motion. These are not executed and have no connection to previous or subsequent geometry descriptions. They only define the protection zone.
4. End of definition

Syntax

```
DEF INT <Var>
G17/G18/G19
CPROTDEF/NPROTDEF(<n>,<t>,<AppLim>,<AppPlus>,<AppMinus>)
G0/G1/... X/Y/Z...
...
EXECUTE(<Var>)
```

Meaning

`DEF INT <Var>`:	Definition of a local help variable, of the INTEGER data type	
`<Var>`:	Name of the Help variable	
`G17/G18/G19`:	Machining plane **Note:** It is not permissible to change the machining plane before the end of the definition. Programming the applicate is not permitted between start and end of the definition.	
`CPROTDEF()`:	Predefined procedure to define a **channel**-specific protection zone	
`NPROTDEF()`:	Predefined procedure to define a **machine**-specific protection zone	
`<n>`:	Number of defined protection zone	
	Data type:	INT

`<t>`:	Type of protection zone		
	Data type:	BOOL	
	Value:	`TRUE`	**Tool**-related protection zone
		`FALSE`	**Workpiece**-related protection zone
`<AppLim>`:	Type of limitation in the third dimension		
	Data type:	INT	
	Value:	`0`	No limitation
		`1`	Limit in plus direction
		`2`	Limit in minus direction
		`3`	Limit in positive and negative direction
`<AppPlus>`:	Value of the limit in the positive direction in the 3rd dimension		
	Data type:	REAL	
`<AppMinus>`:	Value of the limit in the negative direction in the 3rd dimension		
	Data type:	REAL	
`G0/G1/... X/Y/Z... ...`:	The contour of a protection zone is specified with up to 11 traversing movements in the selected machining 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. 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 for an external protection zone must be described in the clockwise direction. 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 counter-clockwise direction **Note:** A protection zone cannot be described by a complete circle. A complete circle must be divided into two semicircles. **Note:** The sequence `G2` → `G3` or `G3` → `G2` is not permissible! A short `G1` block must be inserted between the two circular blocks.		
`EXECUTE(<Var>)`:	Predefined procedure that marks the end of the definition A switch is made back to normal program processing with `EXECUTE`.		

Example

See example under "Activating/deactivating protection zones (CPROT, NPROT) (Page 161)".

Additional information

Machine-specific protection zones

A machine-specific protection zone or its contour is defined using the geometry axis, i.e. referenced 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-specific protection zone is active, the basic coordinate system (BCS) of all of the channels involved must be identical:

- position of the coordinate origin referred to the machine zero
- Orientation of the coordinate axes

Reference point for contour description

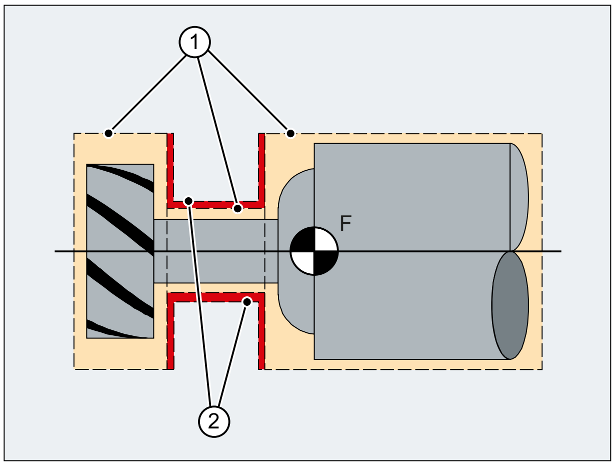
- Tool-related protection zones
  Coordinates for **tool**-related protection zones must be specified as absolute values referred to the **tool holder reference point F**.
- Workpiece-related protection zones
  Coordinates for **workpiece**-related protection zones must be specified as absolute values referred to the zero point of the **basic coordinate system (BCS)**.

Protection zones symmetrical around the center of rotation

For protection zones symmetrical around the axis or rotation (e.g. spindle chuck), you must describe the complete contour and not only the contour up to the center of rotation.

Tool-related protection zones

Tool-related protection zones must always be convex. If a concave protected zone is desired, this should be subdivided into several convex protection zones.



①	Convex protection zones
②	Concave protection zones (**not permissible!**)
F	Toolholder reference point

General conditions

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`

4.3.2 Activating/deactivating protection zones (CPROT, NPROT)

Protection zones previously defined in the part program can be activated at any time – or can be preactivated for subsequent activation by the PLC user program. Active protection zones can be deactivated at any time.

When activating or preactivating, it is also possible to relatively shift the reference point of the protection zone.

Note

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

Note

Monitoring protection zones

If a tool-related protection area is not active, the tool path is checked against the workpiece-related protection zones.

If no workpiece-oriented protection zone is active, then there is no protection zone monitoring.

Syntax

```
CPROT(<n>,<Status>,<XMov>,<YMov>,<ZMov>)
NPROT(<n>,<Status>,<XMov>,<YMov>,<ZMov>)
```

Meaning

`CPROT`:	Predefined procedure to activate a **channel**-specific protection zone
`NPROT`:	Predefined procedure to activate a **machine**-specific protection zone

`<n>`:	Number of the protection zone		
	Data type:	INT	
`<Status>`:	The channel-specific activation status is set using this parameter		
	Data type:	INT	
	Value:	`0`	Deactivate protection zone
		`1`	Preactivate protection zone
		`2`	Activate protection zone
		`3`	Preactivate protection zone with conditional stop
`<XMov>,<YMov>,<ZMov>`:	Additive offset values in the X/Y/Z direction The offset can take place in 1, 2, or 3 dimensions. The offset values refer to: ● The machine zero for a **workpiece**-related protection zone ● The tool carrier reference point F for a **tool**-specific protection zone		
	Data type:	REAL	

Example

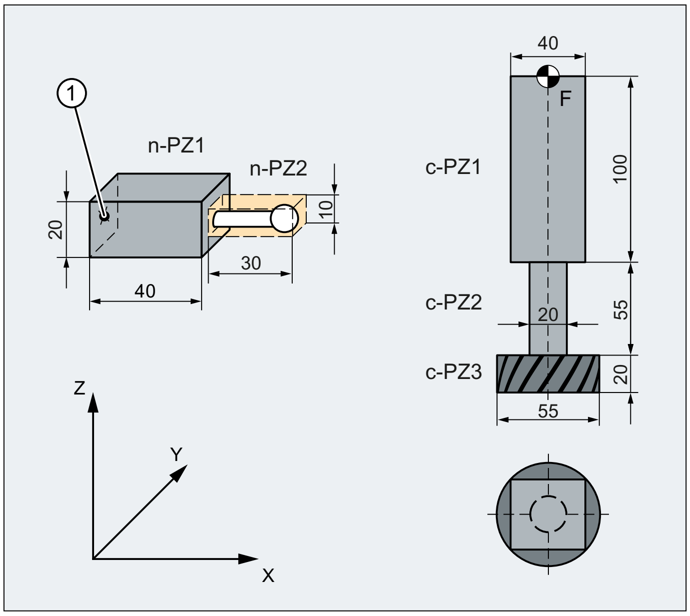
Possible collision of a milling cutter with the measuring probe is to be monitored on a milling machine. The position of the measuring probe is to be defined by an offset when the function is activated.

The following protection zones are defined for this:

- A machine-specific and a workpiece-related protection zone for both the measuring probe holder (n-PZ1) and the measuring probe itself (n-PZ2).
- A channel-specific and a tool-related protection zone for the milling cutter holder (c-PZ1), the cutter shank (c-PZ2) and the milling cutter itself (c-PZ3).

The orientation of all protection zones is in the Z direction.

The position of the reference point of the measuring probe on activation of the function must be X = -120, Y = 60 and Z = 80.



①	Name for the protection zone of the probe
F	Toolholder reference point

Program code	Comment
`DEF INT PROTZONE`	`; Definition of a Help variable`
`G17`	`; machining plane XY`
`; defining protection zones:`	
`NPROTDEF(1,FALSE,3,10,-10)`	`; protection zone n-PZ1`
`G01 X0 Y-10`	
`X40`	
`Y10`	
`X0`	
`Y-10`	
`EXECUTE(PROTZONE)`	
`NPROTDEF(2,FALSE,3,5,-5)`	`; protection zone n-PZ2`
`G01 X40 Y-5`	
`X70`	
`Y5`	
`X40`	
`Y-5`	
`EXECUTE(PROTZONE)`	
`CPROTDEF(1,TRUE,3,0,-100)`	`; protection zone c-PZ1`
`G01 X-20 Y-20`	
`X20`	
`Y20`	
`X-20`	
`Y-20`	
`EXECUTE(PROTZONE)`	

Program code	Comment
`CPROTDEF(2,TRUE,3,-100,-150)`	`; protection zone c-PZ2`
`G01 X0 Y-10`	
`G03 X0 Y10 J10`	
`X0 Y-10 J-10`	
`EXECUTE(PROTZONE)`	
`CPROTDEF(3,TRUE,3,-150,-170)`	`; protection zone c-PZ3`
`G01 X0 Y-27.5`	
`G03 X0 Y27.5 J27.5`	
`X0 Y27.5 J-27.5`	
`EXECUTE(PROTZONE)`	
`; activating protection zones:`	
`NPROT(1,2,-120,60,80)`	`; activate protection zone n-PZ1 with offset`
`NPROT(2.2,-120,60,80)`	`; activate protection zone n-PZ2 with offset`
`CPROT(1,2,0,0,0)`	`; activate protection zone c-PZ1`
`CPROT(2,2,0,0,0)`	`; activate protection zone c-PZ2`
`CPROT(3,2,0,0,0)`	`; activate protection zone c-PZ3`

Further information

Activation status after the control powers up

A protection zone can already be active after the control system powers up and the axes have been referenced. This is the case if, for the protection zone, the following system variable is set to TRUE:

- $SN_PA_ACTIV_IMMED[<n>] (for machine-specific protection zone) or
- $SC_PA_ACTIV_IMMED[<n>] (for channel-specific protection zone)
  Index "<n>" corresponds to the number of the protection zone: 0 = 1. Protection zone

The protection zone is activated with status = 2 – and without offset.

Multiple activation of a protection zone

A machine-specific protection zone can be active simultaneously in several channels (e.g. protection zone of a tailstock where there are two opposite sides). The protection zones are only monitored if all geometry axes have been referenced.

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

Protection zone monitoring for active tool radius compensation

For active tool radius compensation, a functioning protection zone monitoring is only possible if the plane of the tool radius compensation is identical to the plane of the protection zone definitions.

4.3.3 Checking for protection zone violation, working area limitation and software limit switches (CALCPOSI)

Function

In the workpiece coordinate system (WCS), the CALCPOSI function checks whether, starting from the starting position, the **geometry axes** can be traversed a specified distance without violating active limits. For the case that the distance cannot be fully traversed because of limits, a positive, decimal-coded status value and the maximum possible traversing distance are returned.

Definition

```
INT CALCPOSI(VAR REAL[3] <Start>, VAR REAL[3] <Dist>, VAR REAL[5]
<Limit>, VAR REAL[3] <MaxDist>, BOOL <MeasSys>, INT <TestLim>)
```

Syntax

```
<Status> = CALCPOSI(VAR <Start>, VAR <Dist>, VAR <Limit>, VAR
<MaxDist>, <MeasSys>, <TestLim>)
```

Meaning

`CALCPOSI(...):`	Predefined function for testing limit violations regarding the geometry axes	
	Preprocessing stop:	No
	Alone in the block:	Yes

`<status>`: (Part 1)	Function return value. Negative values indicate error states.		
	Data type:		INT
	Value range:		-8 ≤ x ≤ 100000
	Value:	0	The distance can be traversed completely.
		-1	At least one component is negative in <Limit>.
		-2	Error in a transformation calculation. Example: The traversing distance passes through a singularity so that the axis positions cannot be defined.
		-3	The specified traversing distance <Dist> and the maximum possible traversing distance <MaxDist> are linearly dependent. **Note** Can only occur in conjunction with <TestLim>, bit 4 == 1.
		-4	The projection of the traversing direction contained in <Dist> on to the limitation surface is the zero vector, or the traversing direction is perpendicular to the violated limitation surface. **Note** Can only occur in conjunction with <TestLim>, bit 5 == 1.
		-5	In <TestLim>, bit 4 == 1 AND bit 5 == 1
		-6	At least one machine axis that has to be considered for checking the traversing limits has not been referenced.
		-7	Collision avoidance function: Invalid definition of the kinematic chain or the protection zones.
		-8	Collision avoidance function: This command cannot be executed because of insufficient memory.
`<status>`: (Part 2)	Units digit **Note** If several limits are violated simultaneously, the limit with the greatest restriction on the specified traversing distance is signaled.		
	Value:	1	Software limit switches are limiting the traversing distance
		2	Working area limits are limiting the traversing distance
		3	Protection zones are limiting the traversing distance
		4	Collision avoidance function: Protection zones are limiting the traversing path
	Tens digit		
	Value:	1x	The initial value violates the limit
		2x	The specified straight line violates the limit. This value is returned even if the end point does not violate any limit itself, but the path from the starting point to the end point would cause a limit value to be violated (e.g. by passing through a protection zone, curved software limit switches in the WCS for non-linear transformations, e.g. transmit).

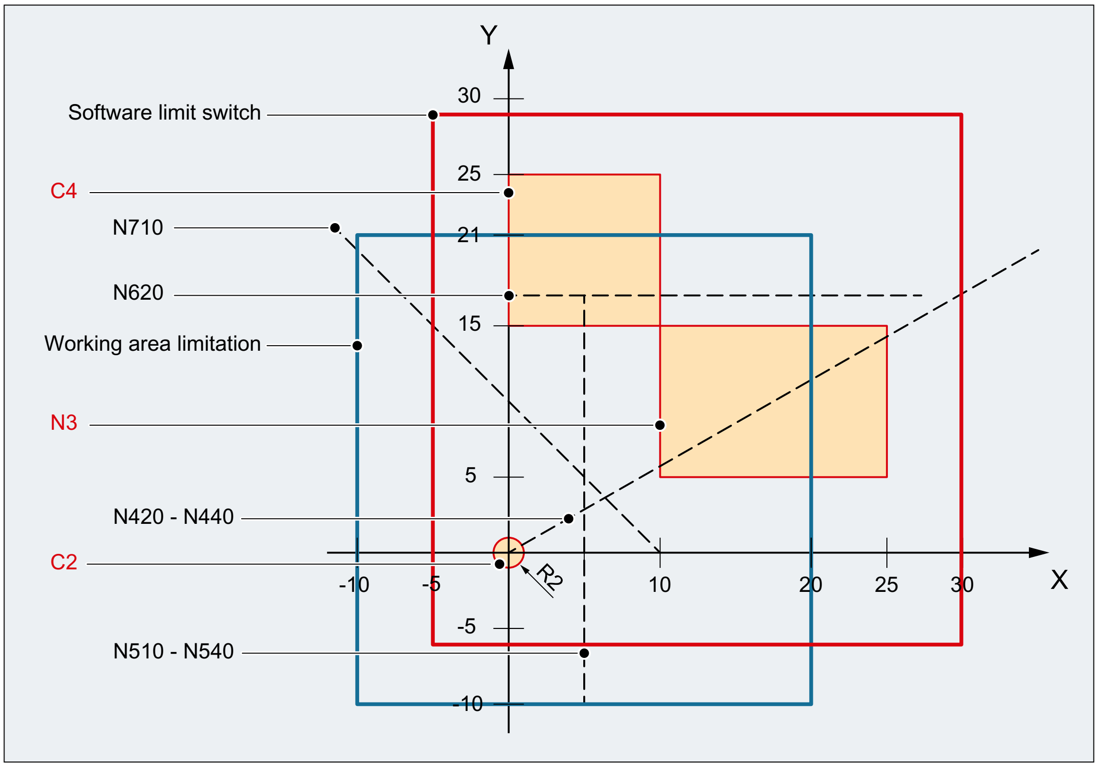
`<status>`: (Part 3)	Hundreds digit		
	Value:	1xx	AND units digit == 1 or 2: The positive limit value has been violated.
			AND units digit == 3 [1)]: An NC-specific protection zone has been violated.
		2xx	AND units digit == 1 or 2: The negative limit value has been violated.
			AND units digit == 3 [1)]: A channel-specific protection zone is violated.
`<status>`: (Part 4)	Thousands digit		
	Value:	1xxx	AND units digit == 1 or 2: Factor with which the axis number is multiplied that violates the limit. Numbering of the axes begins with 1. Reference: • Software limit switches: Machine axes • Working area limitation: Geometry axes AND units digit == 3 [1)]: Factor with which the number of the violated protection zone is multiplied.
`<status>`: (Part 5)	Hundred thousands digit		
	Value:	0xxxxx	Hundred thousands digit == 0: <Dist> remains unchanged
		1xxxxx	A direction vector is returned in <Dist>, which defines the further motion direction on the limitation surface. Can only occur with the following supplementary conditions: • Software limit switch or working area limit violated (not in the starting point) • A transformation is **not** active • <TestID>, bit 4 or bit 5 == 1
`<Start>`:	Reference to a vector with the start positions: • <Start> [0]: 1st geometry axis • <Start> [1]: 2nd geometry axis • <Start> [2]: 3rd geometry axis		
	Parameter type:	Input	
	Data type:	VAR REAL [3]	
	Value range:	-max. REAL value ≤ x[<n>] ≤ +max. REAL value	

`<Dist>`:	Reference to a vector.	
	Input: Incremental traversing distance ● <Dist> [0]: 1st geometry axis ● <Dist> [1]: 2nd geometry axis ● <Dist> [2]: 3rd geometry axis	
	Output (only for set hundred thousands digit in <Status>):	
	<Dist> contains a unit vector **v** as output value which defines the further traversing direction in the WCS.	
	Case 1: Formation of vector **v** for <TestID>, bit 4 == 1 The input vectors <Dist> and <MaxDist> span the motion plane. This plane is cut by the violated limitation surface. The intersecting line of the two planes defines the direction of vector **v**.The orientation (sign) is selected so that the angle between the input vector <MaxDist> and **v** is not greater than 90 degrees.	
	Case 2: Formation of vector **v** for <TestID>, bit 5 == 1 Vector **v** is the unit vector in the projection direction of the traversing vector contained in <Dist> on the limitation surface. If the projection of the traversing vector on the limitation surface is the zero vector, an error is returned.	
	Parameter type:	Input/output
	Data type:	VAR REAL [3]
	Value range:	-max. REAL value ≤ x[<n>] ≤ +max. REAL value
`<Limit>`:	Reference to an array of length 5. ● <Limit> [0 - 2]: Minimum clearance of the geometry axes to the limits: – <Limit> [0]: 1st geometry axis – <Limit> [1]: 2nd geometry axis – <Limit> [2]: 3rd geometry axis The minimum clearances are observed with: – Working area limitation: No restrictions – Software limit switches: If no transformation is active, or a transformation is active in which a clear assignment of the geometry axes to the linear machine axes is possible, e.g. 5-axis transformations. ● <Limit> [3]: Contains the minimum clearance for linear machine axes which, for example, cannot be assigned a geometry axis because of a non-linear transformation. This value is also used as limit value for the monitoring of the conventional protection zones and the collision avoidance protection zones. ● <Limit> [4]: Contains the minimum clearance for rotary machine axes which, for example, cannot be assigned a geometry axis because of a non-linear transformation. **Note** This value is only active for the monitoring of the software limit switches for special transformations.	
	Parameter type:	Input
	Data type:	VAR REAL [5]
	Value range:	-max. REAL value ≤ x[n] ≤ +max. REAL value

`<MaxDist>`:	Reference to a vector with the incremental traversing distance in which the specified minimum clearance of an axis limit is not violated by any of the relevant machine axes: ● <Dist> [0]: 1st geometry axis ● <Dist> [1]: 2nd geometry axis ● <Dist> [2]: 3rd geometry axis If the traversing distance is not restricted, the contents of this return parameter are the same as the contents of <Dist>.		
	For <TestID>, bit 4 == 1: <Dist> and <MaxDist> <MaxDist> and <Dist> must contain vectors as input values that span a motion plane. The two vectors must be mutually linearly independent. The absolute value of <MaxDist> is arbitrary. For the calculation of the motion direction, see the description for <Dist>.		
	Parameter type:		Output
	Data type:		VAR REAL [3]
	Value range:		-max. REAL value ≤ x[<n>] ≤ +max. REAL value
`<MeasSys>`:	Measuring system (inch/metric) for position and distance specifications (**optional**)		
	Data type:		BOOL
	Value:	FALSE (Default)	System of units corresponding to the currently active G command from the G group 13 (G70, G71, G700, G710). **Note** If G70 is active and the basic system is metric (or G71 is active and the basic system is inch), the system variables $AA_IW and $AA_MW are provided in the basic system and, if used, must be converted for CALCPOSI.
		TRUE	System of units according to the set basic system: MD52806 $MN_ISO_SCALING_SYSTEM
`<TestLim>`:	Bit-coded selection of the limits to be monitored (**optional**)		
	Data type:		INT
	Default value:		Bits 0, 1, 2, 3, 6, 7 == 1 (207)
	Bit	**Decimal**	**Meaning**
	0	1	Software limit switch
	1	2	Working area limitation
	2	4	Activated conventional protection zones
	3	8	Preactivated conventional protection zones
	4	16	With violated software limit switches or working area limits in <Dist>, return the traversing direction as in **Case 1** (see above).
	5	32	With violated software limit switches or working area limits in <Dist>, return the traversing direction as in **Case 2** (see above).
	6	64	Activated collision avoidance protection zones
	7	128	Preactivated collision avoidance protection zones
	8	256	Pairs of activated and preactivated collision avoidance protection zones
[1] If several protection zones are violated, the protection zone with the greatest restriction on the specified traversing distance is returned.			

Example

Limitations



In the example, the active software limit switches and working area limits in the X-Y plane and the following three protection zones are displayed:

- C2: Tool-related, channel-specific protection zone, active, circular, radius = 2 mm
- C4: Workpiece-related, channel-specific protection zone, preactivated, square, side length = 10 mm
- N3: Machine-specific protection zone, active, rectangular, side length = 10 mm x 15 mm

NC program

The protection zones and working area limits are defined first in the NC program. The `CALCPOSI()` function is then called with different parameter assignments.

Program code

```
N10 DEF REAL _START[3]
N20 DEF REAL _DIST[3]
N30 DEF REAL _LIMIT[5]
N40 DEF REAL _MAXDIST[3]
N50 DEF INT _PA
N60 DEF INT _STATUS
```

Program code

```
: toolrelated protection zone C2
N70 CPROTDEF(2, TRUE, 0)
N80 G17 G1 X-2 Y0
N90 G3 I2 X2
N100 I-2 X-2
N110 EXECUTE(_PA)
; workpiece-related protection zone C4
N120 CPROTDEF(4, FALSE, 0)
N130 G17 G1 X0 Y15
N140 X10
N150 Y25
N160 X0
N170 Y15
N180 EXECUTE(_PA)
; machine-specific protection zone N3
N190 NPROTDEF(3, FALSE, 0)
N200 G17 G1 X10 Y5
N210 X25
N220 Y15
N230 X10
N240 Y5
N250 EXECUTE(_PA)
; activate or preactivate protection zones
N260 CPROT(2, 2, 0, 0, 0)
N270 CPROT(4, 1, 0, 0, 0)
N280 NPROT(3, 2, 0, 0, 0)
; define working area limits
N290 G25 XX=-10 YY=-10
N300 G26 XX=20 YY=21
N310 _START[0] = 0.
N320 _START[1] = 0.
N330 _START[2] = 0.
N340 _DIST[0] = 35.
N350 _DIST[1] = 20.
N360 _DIST[2] =  0.
N370 _LIMIT[0] = 0.
N380 _LIMIT[1] = 0.
N390 _LIMIT[2] = 0.
N400 _LIMIT[3] = 0.
N410 _LIMIT[4] = 0.
N420 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST)
N430 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,,3)
N440 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,,1)
N450 _START[0] =  5.
N460 _START[1] = 17.
N470 _START[2] =  0.
N480 _DIST[0] =  0.
N490 _DIST[1] =-27.
N500 _DIST[2] =  0.
N510 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,,14)
N520 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,, 6)
N530 _LIMIT[1] = 2.
```

Program code

```
N540 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,, 6)
N550 _START[0] = 27.
N560 _START[1] = 17.1
N570 _START[2] =  0.
N580 _DIST[0] =-27.
N590 _DIST[1] =  0.
N600 _DIST[2] =  0.
N610 _LIMIT[3] = 2.
N620 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,,12)
N630 _START[0] = 0.
N640 _START[1] = 0.
N650 _START[2] = 0.
N660 _DIST[0] =  0.
N670 _DIST[1] = 30.
N680 _DIST[2] =  0.
N690 TRANS X10
N700 AROT Z45
N710 _STATUS = CALCPOSI(_START,_DIST, _LIMIT, _MAXDIST)
; delete frames from N690 and N700 again
N720 TRANS
N730 _START[0] =  0.
N740 _START[1] = 10.
N750 _START[2] =  0.
; vectors_DIST and _MAXDIST define the motion plane
N760 _DIST[0] = 30.
N770 _DIST[1] = 30.
N780 _DIST[2] =  0.
N790 _MAXDIST[0] = 1.
N800 _MAXDIST[1] = 0.
N810 _MAXDIST[2] = 1.
N820 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,,17)
N830 M30
```

Results of CALCPOSI()

N...	<status>	<MaxDist>[0] ≙ X	<MaxDist>[1] ≙ Y	Remarks
420	3123	8.040	4.594	N3 is violated.
430	1122	20.000	11.429	No protection zone monitoring, working area limitation is violated.
440	1121	30.000	17.143	Only software limit monitoring is still active.
510	4213	0.000	0.000	Starting point violates C4
520	0000	0.000	-27.000	Preactivated C4 is not monitored. The specified distance can be traversed completely.
540	2222	0.000	-25.000	Because _LIMIT[1] = 2, the traversing distance is restricted by the working area limitation.

N...	<status>	<MaxDist>[0] ≙ X	<MaxDist>[1] ≙ Y	Remarks
620	4223	-13.000	0.000	Clearance to C4 is a total of 4 mm due to C2 and _LIMIT[3]. Clearance C2 → N3 of 0.1 mm does not result in limitation of the traversing distance.
710	1221	0.000	21.213	Frame with translation and rotation active. The permissible traversing distance in _DIST applies in the shifted and rotated WCS.
820	102121	18.000	18.000	The software limit switch of the Y axis is violated. The calculation of a further traversing direction is requested with <_TESTLIM> = 17. This direction is in _DIST (0.707, 0.0, 0.707). It is valid because the hundred thousands digit is set in <_STATUS>.

Additional information

"Referenced" axis status

All machine axes considered by `CALCPOSI()` must be homed.

Circle-related distance specifications

All circle-related distance specifications are **always** interpreted as radius specifications. This must be taken into account particularly for transverse axes with activated diameter programming (`DIAMON`/`DIAM90`).

Traversing distance reduction

If the specified traversing distance of an axis is limited, the traversing distance of the other axes is also reduced proportionally in the `<MaxDist>` return value. The resulting end point is therefore still on the specified path.

Rotary axes

Rotary axes are only monitored when they are not modulo rotary axes.

It is permissible that no software limit switches, working area limits or protection zones are defined for one or more of the relevant axes.

Software limit switch and working area limitation status

Software limit switches and working area limits are only taken into account if they are active during the execution of `CALCPOSI()`. The status can be influenced, for example, via:

- Machine data: MD21020 $MC_WORKAREA_WITH_TOOL_RADIUS
- Setting data: $AC_WORKAREA_CS_...
- NC/PLC interface signals DB31, ... DBX12.2 / 3
- Commands: `WALIMON` / `WALIMOF`

Software limit switches and transformations

With CALCPOSI(), the positions of the machine axes (MCS) cannot always be uniquely determined from the positions of the geometry axes (WCS) during various kinematic transformations (e.g. TRANSMIT) because of ambiguities at certain positions of the traversing distance. In normal traversing operation, the uniqueness generally results from the history and the condition that a continuous motion in the WCS must correspond to a continuous motion in the MCS. Therefore, when monitoring the software limit switches, the machine position at the time when CALCPOSI() is executed is used to resolve the ambiguity in such cases.

Note

Preprocessing stop

When using CALCPOSI() in conjunction with transformations, it is the sole responsibility of the user to program a preprocessing stop (STOPRE) with the preprocessing before CALCPOSI() for the synchronization of the machine axis positions.

Protection zone clearance and conventional protection zones

With conventional protection zones, there is **no** guarantee that the safety clearance set in parameter <Limit>[3] is maintained for all protection zones during a traversing movement on the specified path. It is only guaranteed that no protection zone will be violated when the end point returned in <Dist> is extended by the safety clearance in the traversing direction. However, the straight line can pass very close to a protection zone.

Protection zone clearance and collision avoidance protection zones

With collision avoidance protection zones, there is a guarantee that the safety clearance set in parameter <Limit>[3] is maintained for all protection zones during a traversing movement on the specified traversing path.

The safety clearance specified in parameter <Limit>[3] only takes effect when the following applies:

<Limit>[3] > (MD10619 $MN_COLLISION_TOLERANCE)

If bit 4 is set in parameter <TestLim> (calculation of the ongoing traversing direction), then the direction vector received in <DIST> is only valid when the hundred thousands digit is set in the function return value (<status>). If a direction such as this cannot be determined, either because protection zones were violated, or because a transformation is active, then the input value in <DIST> remains unchanged. An additional error message is not output.

4.4 Special situations

4.4.1 Temporary enabling of protection zones

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

See also:

- Behavior in the AUTOMATIC and MDA modes (Page 175)
- Behavior in JOG mode (Page 176)

Note

Temporarily enabling protection zones is **only** possible for **workpiece**-related protection zones. **Tool**-related protection zones must either be deactivated in the part program or via the NC/PLC interface in the "Preactivated" state.

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

- When the control system powers up
- AUTOMATIC or MDA mode: The end of the block is outside the protection zone
- JOG mode: The end of the traversing motion is outside the protection zone
- The protection zone is activated

4.4.2 Behavior in the AUTOMATIC and MDA modes

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

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

Temporary enabling of protection zones

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

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

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

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

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

Continuation of a traversing motion without temporary enabling

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

Increased protection against the enabling of protection zones

If the enabling of a protection zone is to be protected better than just by using an NC start, then this must be realized by the machine manufacturer or user in the PLC user program.

4.4.3 Behavior in JOG mode

Simultaneous traversing of several geometry axes

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

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

After the end of the traversing motion, the alarm is cleared automatically.

If the current position is within an activated or preactivated protection zone, then the following actions are triggered:

- Alarm message 10702 "NC protection zone violated during manual mode" or 10703 "Channel-specific protection zone violated during manual mode" is output – specifying the violated protection zone and the traversing axis.
- Further traversing motion is disabled.
- The following NC/PLC interface signal is set for the protection zone involved:
  DB21, … DBX276.0 – 277.1 (machine-specific protection zone 1 - 10 violated) == 1
  or
  DB21, ... DBX278.0 – 279.1 (channel-specific protection zone 1 - 10 violated) == 1

To continue, see paragraph "Temporary enabling of protection zones".

Example:

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

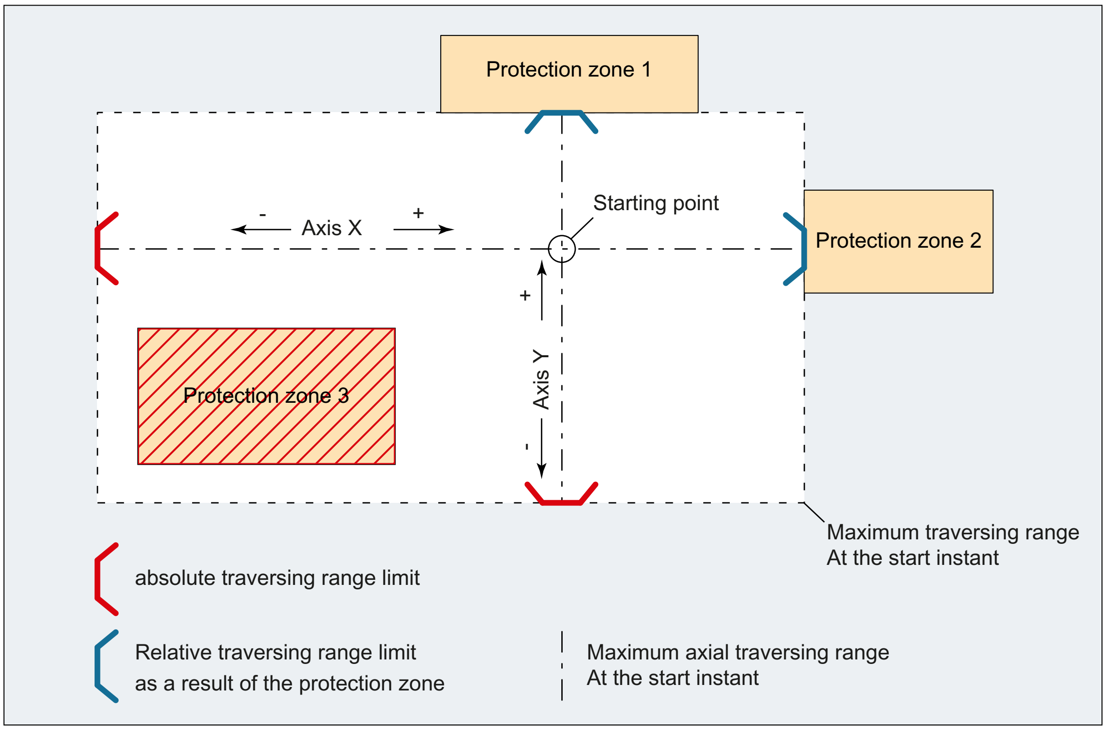


Figure 4-2 Motion range of the geometry axes at the start time

At the start time of traversing motion of axes X and Y, the axis-specific traversing range limits are determined from the start time:

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

The resulting maximum traversing range at the start time does not take protection zone 3 into account. As a consequence, protection zone 3 could be violated.

Note

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

Limiting the traversing motion of an axis

If the traversing motion of an axis is limited as a protection zone has been reached, then alarm 10706 "NC protection zone reached during manual mode" or 10707 "Channel-specific protection zone reached during manual mode" is output, specifying the protection zone reached and the traversing axis. It is assured that no protection zone will be violated when an axis is traversing in JOG. (This response is analogous to approaching software limit switches or a working area limitation.)

The alarm is reset:

- When an axis is traversed that does not lead into the protection zone.
- When the protection zone is enabled.
- When the control powers up.

If an axis starts to move towards a protection zone when it is at a protection zone limit, then alarm 10706 or 10707 is output, and motion is not started.

Temporary enabling of protection zones

If traversing motion is started within an activated protection zone, or at the limit of an activated protection zone, then alarm 10702 "NC protection zone violated during manual mode" or 10703 "Channel-specific protection zone violated during manual mode" is output – specifying the violated protection zone and the traversing axis – and motion is not started. The traversing motion can be performed when the relevant protection zone is temporarily enabled. The following actions must be performed for this:

- Generate a positive edge at the NC/PLC interface signal:
  DB21, ... DBX1.1 (enable protection zone)
- Start the same traversing motion again

Note

The NC/PLC interface signal "Protection zone violated" is set when the temporarily enabled protection zone is traversed:

DB21, … DBX276.0 – 277.1 (machine-specific protection zone 1 - 10 violated) == 1

or

DB21, ... DBX278.0 – 279.1 (channel-specific protection zone 1 - 10 violated) == 1

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

If further protection zones are affected by the traversing motion, additional alarms 10702 or 10703 are displayed for each protection zone. The protection zones displayed in the alarms can be enabled by creating further positive edges on the NC/PLC interface signal DB21, ... DBX1.1:

Behavior at change of operating mode

The protection zones temporarily enabled in JOG mode are retained after a change to AUTOMATIC or MDI mode. The temporary enable signals set the in the AUTOMATIC or MDI mode are also retained after a change to JOG mode.

Reset of an enable

The enable signal is reset internally and on the NC/PLC interface at the next standstill of a geometry axis for which the temporarily enabled protection zone has been completely exited:

DB21, … DBX276.0 – 277.1 (machine-specific protection zone 1 - 10 violated) == 0

or

DB21, ... DBX278.0 – 279.1 (channel-specific protection zone 1 - 10 violated) == 0

4.5 Boundary conditions

Restrictions in protection-zone monitoring

In the following cases, protection area monitoring is **not**possible:

- Traversing motion of orientation axes
- **Fixed machine**-specific protection zones for face transformation (TRANSMIT) or cylindrical surface transformation (TRACYL).
  Exception: Protection areas that are symmetrical around the axis of rotation of the spindle axis. Here, no DRF offset must be active.
- Mutual monitoring of **tool**-related protection areas

Positioning axes

For positioning axes, only the programmed block end point is monitored. Alarm 10704 "Protection zone monitoring is not guaranteed" is output while a positioning axis traverses:

Axis interchange

With regard to the protection zones after an axis replacement, the starting position is the last position approached in the relinquishing channel. Traversing motion in the channel taking over is not taken into account. For this reason, you must ensure that an axis replacement is not performed from a position with protection zone violation.

If an axis intended for the axis interchange is not active in a channel, the position of the axis last approached in the channel is taken as the current position. If this axis has not yet been traversed in the channel, 0.0 is taken as the position.

Behavior for superimposed motions

Superimposed motions that are included in the main run, cannot be taken into account by the block preparation with regard to the active protection zones.

This results in the following reactions:

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

4.6 Example

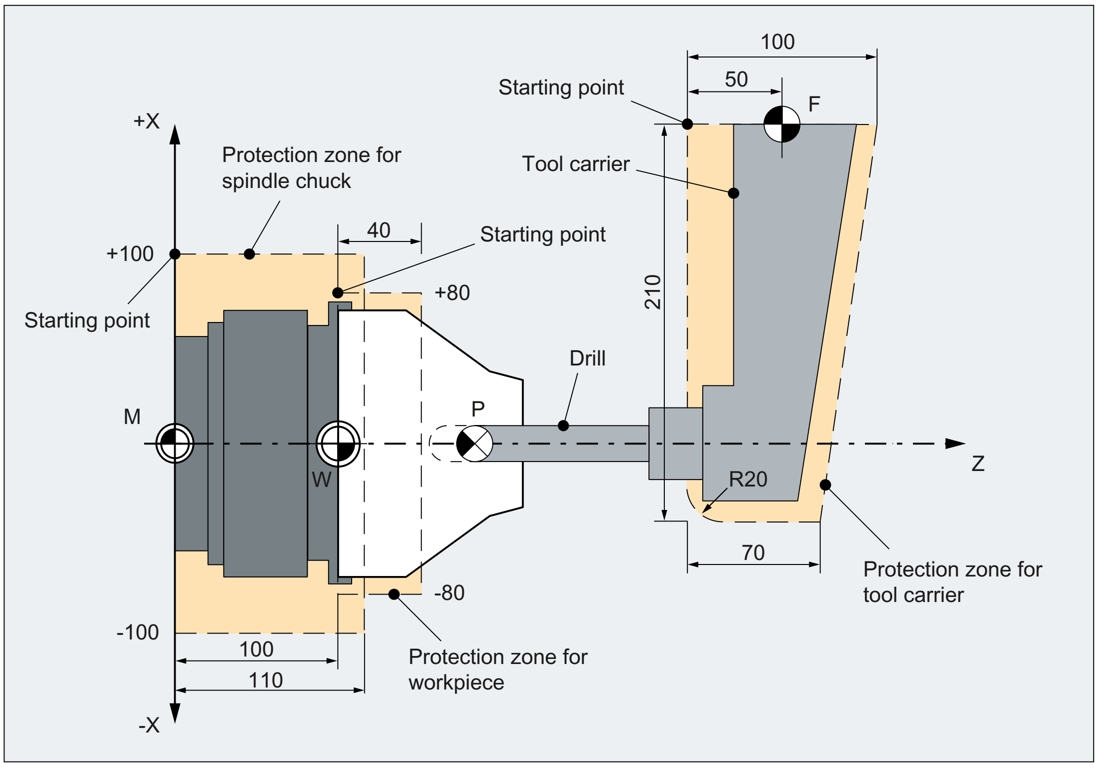
4.6.1 Protection zone on a lathe

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

- one machine-specific and workpiece-related protection zone for the spindle chuck, without limitation in the 3rd dimension
- one channel-specific protection zone for the workpiece, without limitation in the 3rd dimension
- one channel-specific, tool-related protection zone for the tool holder, without limitation in the 3rd dimension

The workpiece zero is placed at machine zero to define the protection zone for the workpiece.

When activated, the protection zone is then offset by 100 mm in the Z axis in the positive direction.



4.6.2 Protection zone definition in the part program

Part program excerpt for protection zone definition:

Program code	Comment
`DEF INT AB`	
`G18`	`; Working plane ZX`
`NPROTDEF(1,FALSE,0,0,0)`	`; Start of definition: Protection zone for spindle chuck`
`G01 X100 Z0`	`; Contour description: Traversing motion along the contour`
`G01 X-100 Z0`	`; Contour description: 1st Contour element`
`G01 X-100 Z110`	`; Contour description: 2nd Contour element`
`G01 X100 Z110`	`; Contour description: 3rd Contour element`
`G01 X100 Z0`	`; Contour description: 4th Contour element`
`EXECUTE(AB)`	`; End of definition: Protection zone for spindle chuck`
`CPROTDEF(1,FALSE,0,0,0)`	`; Start of definition: Protection zone for work-piece`
`G01 X80 Z0`	`; Contour description: Traversing motion along the contour`
`G01 X-80 Z0`	`; Contour description: 1st Contour element`
`G01 X-80 Z40`	`; Contour description: 2nd Contour element`
`G01 X80 Z40`	`; Contour description: 3rd Contour element`
`G01 X80 Z0`	`; Contour description: 4th Contour element`
`EXECUTE(AB)`	`; End of definition: Protection zone for workpiece`
`CPROTDEF(2,TRUE,0,0,0)`	`; Start of definition: Protection zone for tool-holder`
`G01 X0 Z-50`	`; Contour description: Traversing motion along the contour`
`G01 X-190 Z-50`	`; Contour description: 1st Contour element`
`G03 X-210 Z-30 I-20`	`; Contour description: 2nd Contour element`
`G01 X-210 Z20`	`; Contour description: 3rd Contour element`
`G01 X0 Z50`	`; Contour description: 4th Contour element`
`G01 X0 Z-50`	`; Contour description: 5th Contour element`
`EXECUTE(AB)`	`; End of definition: Protection zone for toolholder`

4.6.3 Protection zone definition with system variables

Machine-specific protection zone for the spindle chuck

System variable	Val-ue	Description
$SN_PA_ACTIV_IMMED[0]	0	Protection zone for spindle chuck not immediately active
$SN_PA_T_W[0]	0	The protection zone for the spindle chuck is workpiece-related

System variable	Val-ue	Description
$SN_PA_ORI[0]	1	Orientation of the protection zone: 1 = 3rd and 1st geometry axis
$SN_PA_LIM_3DIM[0]	0	Type of limitation in the 3rd 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 limit in the minus 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 straight line, protection zone for spindle chuck, contour element 1
$SN_PA_CONT_TYP[0,1]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone for spindle chuck, contour element 2
$SN_PA_CONT_TYP[0,2]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone for spindle chuck, contour element 3
$SN_PA_CONT_TYP[0,3]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone for spindle chuck, contour element 4
$SN_PA_CONT_TYP[0,4]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone for spindle chuck, contour element 5
$SN_PA_CONT_TYP[0,5]	0	Contour type[<i>] : 0 = not defined, protection zone for spindle chuck, contour element 6
$SN_PA_CONT_TYP[0,6]	0	Contour type[<i>] : 0 = not defined, protection zone for spindle chuck, contour element 7
$SN_PA_CONT_TYP[0,7]	0	Contour type[<i>] : 0 = not defined, protection zone for spindle chuck, contour element 8
$SN_PA_CONT_TYP[0,8]	0	Contour type[<i>] : 0 = not defined, protection zone for spindle chuck, contour element 9
$SN_PA_CONT_TYP[0,9]	0	Contour type[<i>] : 0 = not defined, protection zone for spindle chuck, contour element 10
$SN_PA_CONT_ORD[0,0]	-100	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 1
$SN_PA_CONT_ORD[0,1]	-100	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 2
$SN_PA_CONT_ORD[0,2]	100	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 3
$SN_PA_CONT_ORD[0,3]	100	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 4
$SN_PA_CONT_ORD[0,4]	0	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 5
$SN_PA_CONT_ORD[0,5]	0	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 6
$SN_PA_CONT_ORD[0,6]	0	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 7
$SN_PA_CONT_ORD[0,7]	0	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 8
$SN_PA_CONT_ORD[0,8]	0	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 9
$SN_PA_CONT_ORD[0,9]	0	End point of contour[<i>], ordinate value protection zone for spindle chuck, contour element 10

System variable	Val-ue	Description
$SN_PA_CONT_ABS[0,0]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 1
$SN_PA_CONT_ABS[0,1]	110	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 2
$SN_PA_CONT_ABS[0,2]	110	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 3
$SN_PA_CONT_ABS[0,3]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 4
$SN_PA_CONT_ABS[0,4]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 5
$SN_PA_CONT_ABS[0,5]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 6
$SN_PA_CONT_ABS[0,6]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 7
$SN_PA_CONT_ABS[0,7]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 8
$SN_PA_CONT_ABS[0,8]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 9
$SN_PA_CONT_ABS[0,9]	0	End point of contour[<i>], abscissa value protection zone for spindle chuck, contour element 10
$SN_PA_CENT_ORD[0,0]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 1
$SN_PA_CENT_ORD[0.1]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 2
$SN_PA_CENT_ORD[0,2]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 3
$SN_PA_CENT_ORD[0,3]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 4
$SN_PA_CENT_ORD[0,4]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 5
$SN_PA_CENT_ORD[0,5]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 6
$SN_PA_CENT_ORD[0,6]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 7
$SN_PA_CENT_ORD[0,7]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 8
$SN_PA_CENT_ORD[0,8]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 9
$SN_PA_CENT_ORD[0,9]	0	Center point of circular contour[<i>], ordinate value protection zone for spindle chuck, contour element 10
$SN_PA_CENT_ABS[0,0]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 1
$SN_PA_CENT_ABS[0.1]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 2
$SN_PA_CENT_ABS[0,2]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 3

System variable	Val-ue	Description
$SN_PA_CENT_ABS[0,3]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 4
$SN_PA_CENT_ABS[0,4]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 5
$SN_PA_CENT_ABS[0,5]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 6
$SN_PA_CENT_ABS[0,6]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 7
$SN_PA_CENT_ABS[0,7]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 8
$SN_PA_CENT_ABS[0,8]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 9
$SN_PA_CENT_ABS[0,9]	0	Center point of circular contour[<i>], abscissa value protection zone for spindle chuck, contour element 10

Channel-specific protection zone for the workpiece

System variable	Val-ue	Remark
$SC_PA_ACTIV_IMMED[0]	0	Protection zone for workpiece not immediately active
$SC_PA_TW[0]	0	Protection zone for workpiece is workpiece-related
$SC_PA_ORI[0]	1	Orientation of the protection zone: 1 = 3rd and 1st geometry axis
$SC_PA_LIM_3DIM[0]	0	Type of limitation in the 3rd dimension: 0 = no limitation
$SC_PA_PLUS_LIM[0]	0	Value of the limit in the positive direction in the 3rd dimension
$SC_PA_MINUS_LIM[0]	0	Value of the limit in the minus direction in the 3rd dimension
$SC_PA_CONT_NUM[0]	4	Number of valid contour elements
$SC_PA_CONT_TYP[0,0]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone workpiece, contour element 1
$SC_PA_CONT_TYP[0,1]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone workpiece, contour element 2
$SC_PA_CONT_TYP[0,2]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone workpiece, contour element 3
$SC_PA_CONT_TYP[0,3]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone workpiece, contour element 4
$SC_PA_CONT_TYP[0,4]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone workpiece, contour element 5
$SC_PA_CONT_TYP[0,5]	0	Contour type[<i>] : 0 = not defined, protection zone for workpiece, contour element 6
$SC_PA_CONT_TYP[0,6]	0	Contour type[<i>] : 0 = not defined, protection zone for workpiece, contour element 7
$SC_PA_CONT_TYP[0,7]	0	Contour type[<i>] : 0 = not defined, protection zone for workpiece, contour element 8
$SC_PA_CONT_TYP[0,8]	0	Contour type[<i>] : 0 = not defined, protection zone for workpiece, contour element 9
$SC_PA_CONT_TYP[0,9]	0	Contour type[<i>] : 0 = not defined, protection zone for workpiece, contour element 10

System variable	Val- ue	Remark
$SC_PA_CONT_ORD[0,0]	-80	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 1
$SC_PA_CONT_ORD[0,1]	-80	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 2
$SC_PA_CONT_ORD[0,2]	80	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 3
$SC_PA_CONT_ORD[0,3]	80	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 4
$SC_PA_CONT_ORD[0,4]	0	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 5
$SC_PA_CONT_ORD[0,5]	0	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 6
$SC_PA_CONT_ORD[0,6]	0	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 7
$SC_PA_CONT_ORD[0,7]	0	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 8
$SC_PA_CONT_ORD[0,8]	0	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 9
$SC_PA_CONT_ORD[0,9]	0	End point of contour[<i>], ordinate value protection zone for workpiece, contour element 10
$SC_PA_CONT_ABS[0,0]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 1
$SC_PA_CONT_ABS[0,1]	40	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 2
$SC_PA_CONT_ABS[0,2]	40	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 3
$SC_PA_CONT_ABS[0,3]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 4
$SC_PA_CONT_ABS[0,4]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 5
$SC_PA_CONT_ABS[0,5]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 6
$SC_PA_CONT_ABS[0,6]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 7
$SC_PA_CONT_ABS[0,7]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 8
$SC_PA_CONT_ABS[0,8]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 9
$SC_PA_CONT_ABS[0,9]	0	End point of contour[<i>], abscissa value protection zone for workpiece, contour element 10
$SC_PA_CENT_ORD[0,0]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 1
$SC_PA_CENT_ORD[0,1]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 2
$SC_PA_CENT_ORD[0,2]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 3

System variable	Val-ue	Remark
$SC_PA_CENT_ORD[0,3]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 4
$SC_PA_CENT_ORD[0,4]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 5
$SC_PA_CENT_ORD[0,5]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 6
$SC_PA_CENT_ORD[0,6]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 7
$SC_PA_CENT_ORD[0,7]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 8
$SC_PA_CENT_ORD[0,8]	0	Center point of circular contour[<i>], ordinate value protection zone for workpiece, contour element 9
$SC_PA_CENT_ORD[0,9]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 10
$SC_PA_CENT_ABS[0,0]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 1
$SC_PA_CENT_ABS[0,1]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 2
$SC_PA_CENT_ABS[0,2]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 3
$SC_PA_CENT_ABS[0,3]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 4
$SC_PA_CENT_ABS[0,4]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 5
$SC_PA_CENT_ABS[0,5]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 6
$SC_PA_CENT_ABS[0,6]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 7
$SC_PA_CENT_ABS[0,7]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 8
$SC_PA_CENT_ABS[0,8]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 9
$SC_PA_CENT_ABS[0,9]	0	Center point of circular contour[<i>], abscissa value protection zone for workpiece, contour element 10

Channel-specific protection zone for the tool holder

System variable	Val-ue	Remark
$SC_PA_ACTIV_IMMED[1]	0	Protection zone for tool holder not immediately active
$SC_PA_TW[1]	3	Protection zone for the tool holder is tool-related
$SC_PA_ORI[1]	1	Orientation of the protection zone: 1 = 3rd and 1st geometry axis
$SC_PA_LIM_3DIM[1]	0	Type of limitation in the 3rd dimension: 0 = no limitation
$SC_PA_PLUS_LIM[1]	0	Value of the limit in the positive direction in the 3rd dimension
$SC_PA_MINUS_LIM[1]	0	Value of the limit in the minus direction in the 3rd dimension
$SC_PA_CONT_NUM[1]	5	Number of valid contour elements

System variable	Val-ue	Remark
$SC_PA_CONT_TYP[1,0]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone tool holder, contour element 1
$SC_PA_CONT_TYP[1,1]	3	Contour type[<i>] : 3 = G3 for circle element, counter clockwise, protection zone for tool holder, contour element 2
$SC_PA_CONT_TYP[1,2]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone for tool holder, contour element 3
$SC_PA_CONT_TYP[1,3]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone for tool holder, contour element 4
$SC_PA_CONT_TYP[1,4]	1	Contour type[<i>] : 1 = G1 for straight line, protection zone for tool holder, contour element 5
$SC_PA_CONT_TYP[1,5]	0	Contour type[<i>] : 0 = not defined, protection zone for tool holder, contour element 6
$SC_PA_CONT_TYP[1,6]	0	Contour type[<i>] : 0 = not defined, protection zone for tool holder, contour element 7
$SC_PA_CONT_TYP[1,7]	0	Contour type[<i>] : 0 = not defined, protection zone for tool holder, contour element 8
$SC_PA_CONT_TYP[1,8]	0	Contour type[<i>] : 0 = not defined, protection zone for tool holder, contour element 9
$SC_PA_CONT_TYP[1,9]	0	Contour type[<i>] : 0 = not defined, protection zone for tool holder, contour element 10
$SC_PA_CONT_ORD[1,0]	-190	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 1
$SC_PA_CONT_ORD[1,1]	-210	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 2
$SC_PA_CONT_ORD[1,2]	-210	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 3
$SC_PA_CONT_ORD[1,3]	0	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 4
$SC_PA_CONT_ORD[1,4]	0	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 5
$SC_PA_CONT_ORD[1,5]	0	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 6
$SC_PA_CONT_ORD[1,6]	0	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 7
$SC_PA_CONT_ORD[1,7]	0	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 8
$SC_PA_CONT_ORD[1,8]	0	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 9
$SC_PA_CONT_ORD[1,9]	0	End point of contour[<i>], ordinate value protection zone for tool holder, contour element 10
$SC_PA_CONT_ABS[1,0]	-50	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 1
$SC_PA_CONT_ABS[1,1]	-30	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 2
$SC_PA_CONT_ABS[1,2]	20	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 3

System variable	Value	Remark
$SC_PA_CONT_ABS[1,3]	50	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 4
$SC_PA_CONT_ABS[1,4]	-50	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 5
$SC_PA_CONT_ABS[1,5]	0	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 6
$SC_PA_CONT_ABS[1,6]	0	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 7
$SC_PA_CONT_ABS[1,7]	0	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 8
$SC_PA_CONT_ABS[1,8]	0	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 9
$SC_PA_CONT_ABS[1,9]	0	End point of contour[<i>], abscissa value protection zone for tool holder, contour element 10
$SC_PA_CENT_ORD[1,0]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 1
$SC_PA_CENT_ORD[1,1]	-190	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 2
$SC_PA_CENT_ORD[1,2]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 3
$SC_PA_CENT_ORD[1,3]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 4
$SC_PA_CENT_ORD[1,4]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 5
$SC_PA_CENT_ORD[1,5]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 6
$SC_PA_CENT_ORD[1,6]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 7
$SC_PA_CENT_ORD[1,7]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 8
$SC_PA_CENT_ORD[1,8]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 9
$SC_PA_CENT_ORD[1,9]	0	Center point of circular contour[<i>], ordinate value protection zone for tool holder, contour element 10
$SC_PA_CENT_ABS[1,0]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 1
$SC_PA_CENT_ABS[1,1]	-30	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 2
$SC_PA_CENT_ABS[1,2]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 3
$SC_PA_CENT_ABS[1,3]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 4
$SC_PA_CENT_ABS[1,4]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 5
$SC_PA_CENT_ABS[1,5]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 6

System variable	Val-ue	Remark
$SC_PA_CENT_ABS[1,6]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 7
$SC_PA_CENT_ABS[1,7]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 8
$SC_PA_CENT_ABS[1,8]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 9
$SC_PA_CENT_ABS[1,9]	0	Center point of circular contour[<i>], abscissa value protection zone for tool holder, contour element 10

4.6.4 Activating protection zones

Part program excerpt for activating protection zones for spindle chuck, workpiece, and tool holder:

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

4.7 Data lists

4.7.1 Machine data

4.7.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10618	PROTAREA_GEOAX_CHANGE_MODE	Response for a transformation change and geometry axis interchange
18190	MM_NUM_PROTECT_AREA_NCK	Number of available machine-specific protection zones

4.7.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
28200	MM_NUM_PROTECT_AREA_CHAN (SRAM)	Number of available channel-specific protection zones
28210	MM_NUM_PROTECT_AREA_ACTIVE	Maximum number of protection zones that can be activated simultaneously in the channel
28212	MM_NUM_PROTECT_AREA_CONTUR	Maximum number of definable contour elements in the channel

4.7.2 Signals

4.7.2.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Enable protection zones	DB21, ...DBX1.1	DB320x.DBX1.1
Feed disable	DB21, ...DBX6.0	DB320x.DBX6.0
Activate machine-specific protection zones 1 ... 8	DB21, ...DBX8.0 ... 7	DB320x.DBX8.0 ... 7
Activate machine-specific protection zone 9	DB21, ...DBX9.0	DB320x.DBX9.0
Activate machine-specific protection zone 10	DB21, ...DBX9.1	DB320x.DBX9.1
Activate channel-specific protection zones 1 ... 8	DB21, ...DBX10.0 ... 7	DB320x.DBX10.0 ... 7
Activate channelspecific protection zone 9	DB21, ...DBX11.0	DB320x.DBX11.0
Activate channelspecific protection zone 10	DB21, ...DBX11.1	DB320x.DBX11.1

4.7.2.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Machine-specific protection zones 1 ... 8 preactivated	DB21, ...DBX272.0 ... 7	DB330x.DBX8.0 ... 7
Machine-specific protection zone 9 preactivated	DB21, ...DBX273.0	DB330x.DBX9.0
Machine-specific protection zone 10 preactivated	DB21, ...DBX273.1	DB330x.DBX9.1
Channel-specific protection zones 1 ... 8 preactivated	DB21, ...DBX274.0 ... 7	DB330x.DBX10.0 ... 7
Channelspecific protection zone 9 preactivated	DB21, ...DBX275.0	DB330x.DBX11.0
Channelspecific protection zone 10 preactivated	DB21, ...DBX275.1	DB330x.DBX11.1
Machine-specific protection zones 1 ... 8 preactivated	DB21, ...DBX276.0 ... 7	DB330x.DBX12.0 ... 7
Machine-specific protection zone 9 violated	DB21, ...DBX277.0	DB330x.DBX13.0
Machine-specific protection zone 10 violated	DB21, ...DBX277.1	DB330x.DBX13.1
Channel-specific protection zone 1 ... 8 violated	DB21, ...DBX278.0 ... 7	DB330x.DBX14.0 ... 7
Channelspecific protection zone 9 violated	DB21, ...DBX279.0	DB330x.DBX15.0
Channelspecific protection zone 10 violated	DB21, ...DBX279.1	DB330x.DBX15.1

B1: Continuous-path mode, Exact stop, Look Ahead 5

5.1 Brief Description

Exact stop or exact stop mode

In exact stop traversing mode, all axes involved in the traversing motion (except axes of modal traversing modes) are decelerated at the end of each block until they come to a standstill. The transition to the next block occurs only when all axes involved in the traversing motion have reached their programmed target position depending on the selected exact stop criterion.

Continuous-path mode

In the continuous-path mode, the NC attempts to keep the programmed path velocity as constant as possible. In particular, deceleration of the path axes at the block limits of the part program is to be avoided.

LookAhead

LookAhead is a function for optimizing the continuous path mode.

Smooth and uniform machining of workpieces is necessary to ensure a high-quality surface finish. For this reason, path velocity variations should be avoided during machining whenever possible. Without LookAhead, the NC only takes the traversing block immediately following the current traversing block into consideration when determining the possible path velocity. If the following block contains only a short path, the NC must reduce the path velocity (decelerate in the current block) to be able to stop in time at the end of the next block, if necessary.

When the NC "looks ahead" over a configurable number of traversing blocks following the current traversing block, a much higher path velocity can be attained under certain circumstances because the NC now has considerably more traversing blocks and more path available for calculation.

This has the following advantages:

- Machining with higher path velocities on average
- Improved surface quality by avoiding deceleration and acceleration

Smoothed path velocity

"Smoothing the path velocity" is a function especially for applications (such as high speed milling in mold and die making) that require an extremely steady and consistent path velocity. Deceleration and acceleration processes that would cause high-frequency excitation of machine resonances are avoided with the "Smoothing the path velocity" function.

This has the following advantages:

- Improved surface quality and shorter machining time by avoiding excitation of machine resonances.
- Constant profile of path velocity and cutting rates by avoiding "unnecessary" acceleration processes, i.e. acceleration processes that do not greatly improve the program run time.

Adaptation of the dynamic path response

In addition to "smoothing the path velocity", "dynamic path response adaptation" is another function for avoiding high-frequency excitations of machine resonances while optimizing the dynamic path response at the same time. To achieve this, highly frequency changes in path velocity are automatically executed with lower jerk or acceleration values than the dynamic response limit value parameterized in the machine data.

Thus, with low-frequency changes in the path velocity, the full dynamic response limit values apply, whereas with high-frequency changes, only the reduced dynamic response limit values act due to the automatic dynamic response adaptation.

Dynamic response mode for path interpolation

Optimizing the path dynamic response also includes the technology-specific dynamic response settings which are preset for different machining technologies (including tapping, roughing, smoothing) and can be activated in the part program by calling the respective dynamic response mode.

Free-form surface mode

Any fluctuation in curvature or torsion leads to a change in path velocity. This generally results in unnecessary decelerating and accelerating while machining free-form surface workpieces, which may adversely affect the quality of the surfaces of the workpieces.

The following functions are available for machining free-form surfaces.

- "Free-form surface mode: Basic functions"
  This makes the definition of the path velocity profile "less sensitive" to fluctuations in curvature and torsion.
- "Free-form surface mode: Extension function"
  This extension in standard LookAhead functionality is used to calculate the path velocity profile while machining free-form surfaces.

The advantages of free-form surface mode lie in a more homogeneous workpiece surface and lower machine stress levels.

Compressing linear blocks

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 "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.

Advantages:

- Reduction of the number of traversing blocks
- Increased path velocity
- Increased surface quality
- Continuous block transitions

Compressing short spline blocks

A spline defines a curve, which comprises 2nd or 3rd degree polynomials With spline interpolation, the control system can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

The advantages of the spline interpolation as compared to the linear interpolation are:

- Reduction of the number of required part program blocks for describing a curved contour.
- Soft characteristic that reduces the stress on the mechanical system, even between part program blocks.

The disadvantages of spline interpolation as compared to linear interpolation are:

- For a spline curve no exact curve characteristic, but only a tolerance band can be specified, within which the spline curve should lie.

As with linear interpolation, the processing of splines can produce such short blocks that the path velocity must be reduced to enable interpolation of the spline blocks. This is also the case, when the spline has a long, smooth curve. The "Compression of short spline blocks" function allows you to combine these spline blocks such that the resulting block length is sufficient and does not reduce the path velocity.

5.2 Exact stop mode

Exact stop or exact stop mode

In the exact stop traversing mode, all path axes and special axes involved in the traversing motion that are not traversed modally, are decelerated at the end of each block until they come to a standstill. The transition to the next block occurs only when all axes involved in the traversing motion have reached their programmed target position depending on the selected exact stop condition.

This results in the following response are obtained:

- 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.

Exact stop condition

The following exact stop conditions can be set:

- "Exact stop coarse"
- "Exact stop fine"
- "Interpolator end"

Application

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

Activation

In a program, exact stop operation can be specified using the following commands, either modal or for specific blocks:

Command	Meaning
`G60`	Exact stop operation is effective from the current **modal** block.
`G9`	Exact stop operation is effective in the current block.

Exact stop conditions - "Exact stop coarse" and "Exact stop fine"

The exact stop condition "coarse" or "fine" is fulfilled by a machine axis if its current following error is less than or equal to the tolerance range around the setpoint position, parameterized in the machine data:

- MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)
- MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)

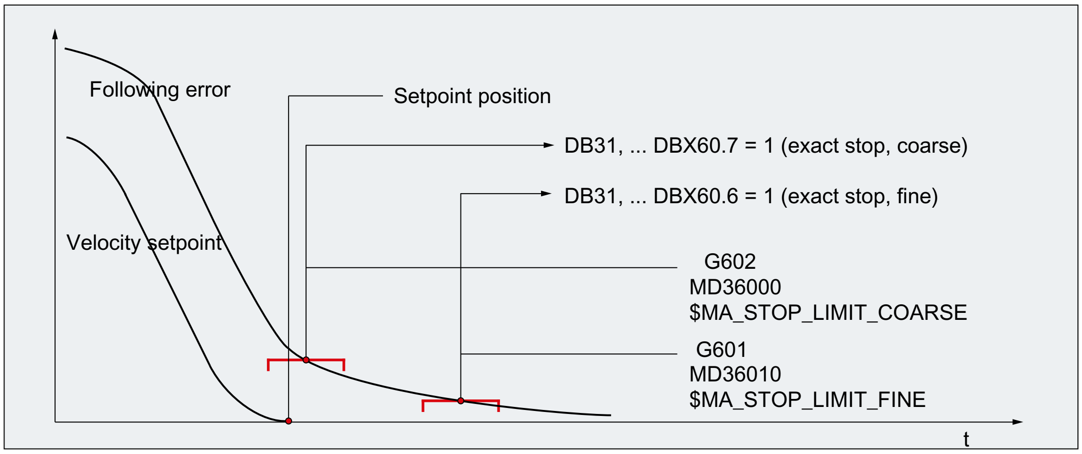


Figure 5-1 Tolerance window of exact stop conditions

Note

The tolerance windows of the exact stop conditions "Exact stop coarse" and "Exact stop fine" should be assigned in such a way that the following requirement is fulfilled:

"Exact stop coarse" > "Exact stop fine"

Exact stop condition, "Interpolator end"

With exact stop condition "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 condition "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 conditions "Exact stop coarse" and "Exact stop fine".

Activating the programmable exact stop conditions

The programmable exact stop conditions are activated using the following commands:

Command	Exact stop condition
`G601`	Exact stop fine
`G602`	Exact stop coarse
`G603`	Interpolator end

Block change depending on exact stop condition

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

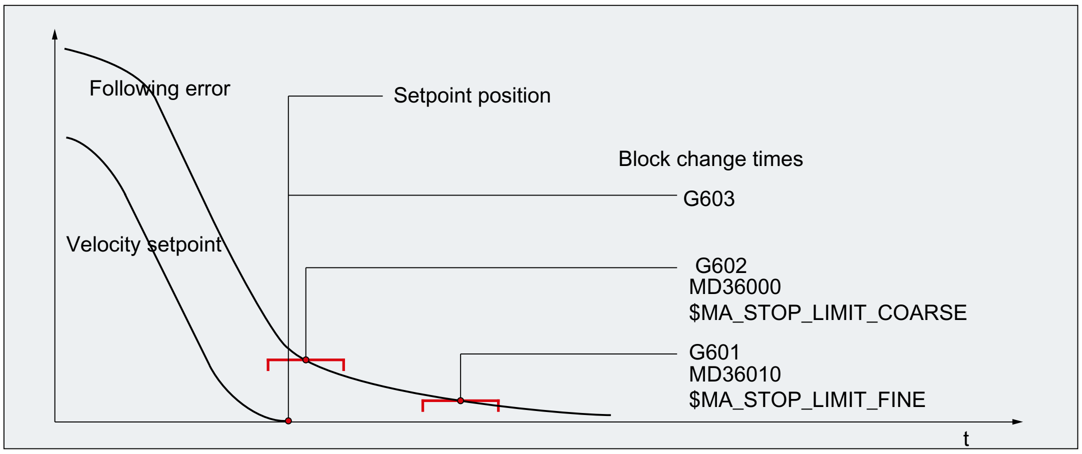


Figure 5-2 Block change depending on exact stop condition

Weighting factor for exact stop conditions

A parameter set-dependent evaluation of the exact stop conditions can be specified via the following axis-specific machine data:

MD36012 $MA_STOP_LIMIT_FACTOR[<Parameter set>] = <Value>

The evaluation factor is applied to the values of the following machine data:

- MD36000 $MA_STOP_LIMIT_COARSE
- MD36010 $MA_STOP_LIMIT_FINE
- MD36030 $MA_STANDSTILL_POS_TOL

Application examples

- Adaptation of the positioning response to different mass ratios, such as after a new gearbox stage has been selected.
- Reduction in positioning time, depending on various machining states, such as roughing and finishing

Parameterizable input of the effective exact stop conditions

The exact stop conditions for the commands of the 1st G group be permanently is specified. As a consequence, programmed exact stop conditions are no longer effective.

Exact stop conditions can be set **independently of one another** for the following commands:

- Rapid traverse `G0`
- All other commands of the 1st G group

The exact stop condition is set on a channel-for-channel basis using the following decimal-coded machine data:

MD20550 $MC_EXACT_POS_MODE = <Z><E>

Z	E	Effective exact stop condition
0	0	Programmed exact stop condition
1	1	G601 (Exact stop window fine)
2	2	G602 (Exact stop window coarse)
3	3	G603 (Interpolator end)
E (ones position): Setting the exact stop condition für rapid traverse. Z (tens position): Setting the exact stop condition for all other commands of the 1st G group.		

Example

MD20550 $MC_EXACT_POS_MODE = 02

- <E> = 2: With rapid traverse, exact stop condition G602 (exact stop window coarse) is always active, irrespective of any programming.
- <Z> = 0: When traversing with all other commands of the 1st G group, the programmed exact stop condition is effective.

Block change behavior for block transitions from G0 ↔ non-G0 to continuous path mode

For the continuous path mode, the block change behavior between rapid traverse and non rapid traverse blocks (G0 ↔ non-G0) can be set using the following machine data:

MD20552 $MC_EXACT_POS_MODE_G0_TO_G1 = <Value>

<Value>	Meaning
0	No additional stop at the block transition.
1	Stop at the block transition. The behavior corresponds to G601 (exact stop window, fine)
2	Stop at the block transition. The behavior corresponds to G602 (exact stop window, coarse)
3	Stop at the block transition. The behavior corresponds to G603 (interpolator end)
4	**No** stop at the block transition. For the continuous path mode, for **block changes** from **G0 → non-G0**, in the G0 block, the actual value of the **feedrate correction** of the subsequent non-G0 block is predicatively taken into account. Depending on the axis dynamic performance and the path length of the current block, the block change is executed with the exact or the best possible adapted velocity of the following block.
5	**No** stop at the block transition. For the continuous path mode, for **block changes** from **G0 → non-G0** and **non-G0 → G0**, the current value of the **feedrate correction** (G0 → non-G0) or the **rapid traverse contour** (non-G0 → G0) of the following block is predicatively taken into account. Depending on the axis dynamic performance and the path length of the current block, the block change is executed with the exact or the best possible adapted velocity of the following block.

5.3 Continuous-path mode

5.3.1 General functionality

Continuous-path mode

In the continuous-path mode, the path velocity is **not** decelerated for the block change in order to permit the fulfillment of an exact stop criterion. The objective of this mode is to avoid rapid deceleration of the path axes at the block-change point so that the axis velocity remains as constant as possible when the program moves to the next block. To achieve this objective, the "LookAhead" function is also activated when the continuous-path mode is selected.

Continuous-path mode causes the smoothing and tangential shaping of angular block transitions by local changes in the programmed contour. The extent of the change relative to the programmed contour can be limited by specifying the overload factor or rounding criteria.

Continuous-path mode:

- Contour rounding.
- Reduces machining times by eliminating braking and acceleration processes that are required to fulfill the exact-stop criterion.
- Improves cutting conditions because of the more constant velocity.

Continuous-path mode is suitable if:

- A contour must be traversed as quickly as possible (e.g. with rapid traverse).
- The exact contour may deviate from the programmed contour within a specific tolerance for the purpose of obtaining a continuous contour

Continuous-path mode is not suitable if:

- A contour is to be traversed precisely.
- An absolutely constant velocity is required.

Implicit exact stop

In some cases, an exact stop needs to be generated in continuous-path mode to allow the execution of subsequent actions. In such situations, the path velocity is reduced to zero.

- If auxiliary functions are output before the traverse motion, the previous block is only terminated when the selected exact-stop criterion is fulfilled.
- If auxiliary functions are to be output after the traverse motion, they are output after the interpolator end of the block.
- If an executable block (e.g. starting of a positioning axis) contains no travel information for the path axes, the previous block is terminated on reaching the selected exact-stop criterion.
- If a positioning axis is declared to be the geometry axis, the previous block is terminated at the interpolator end when the geometry axis is programmed.
- If a synchronized axis is programmed that was last programmed as a positioning axis or spindle (initial setting of the special axis is positioning axis), the previous block is ended at the interpolator end.
- If the transformation is changed, the block previously processed is terminated with the active exact-stop criterion.
- A block is terminated on interpolator end if the following block contains the changeover of the acceleration profile BRISK/SOFT (see Section "B2: Acceleration (Page 267)").
- If the "empty buffer" function is programmed, the previous block is terminated when the selected exact-stop criterion is reached.

Velocity = 0 in continuous-path mode

Regardless of the implicit exact stop response, the path motion is braked down to zero velocity at the end of the block in cases where:

- Positioning axes are programmed with the instruction `POS`, and their traversing time exceeds that of the path axes. The block change occurs when the "exact stop fine" of the positioning axes is reached.
- The time taken to position a spindle programmed with the instruction `SPOS` is longer than the traversing time of the path axes. The block change is carried out when the "exact stop fine" of the positioning spindle is reached.
- The current block contains traversing commands for geometry axes and the following block traversing commands for synchronized axes or, alternatively, the current block contains traversing commands for synchronized axes and the subsequent block traversing commands for geometry axes.
- Synchronization is required.

5.3.2 Velocity reduction according to overload factor

Function

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

With the reduced velocity, axial jumps in velocity are produced with a non-tangential contour at the block transition. These jumps in velocity are also performed by the coupled motion synchronized axes. The jump in velocity prevents the path velocity dropping to zero. This jump is performed if the axial velocity was reduced with the axial acceleration to a velocity from which the new setpoint can be reached with the jump. The magnitude of the setpoint jump can be limited using an overload factor. Because the magnitude of the jump is axial, the minimum jump of the path axes which are active during the block change is considered during block transition.

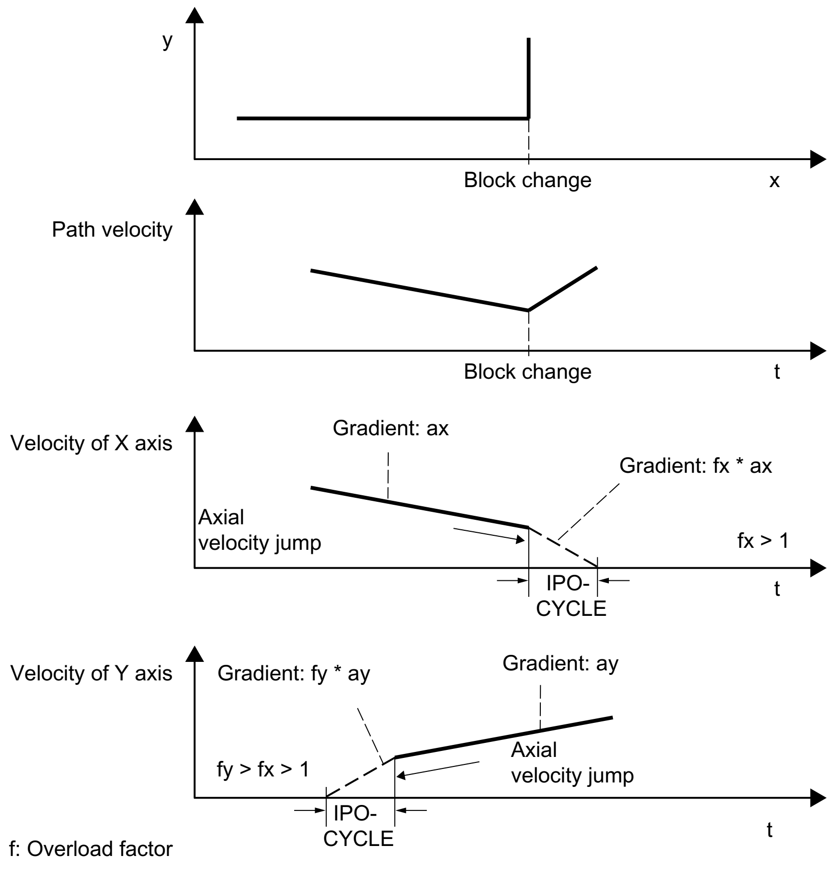


Figure 5-3 Axial velocity change on block transition

With a practically tangential block transition, the path velocity is not reduced if the permissible axial accelerations are not exceeded. This means that very small bends in the contour (e.g. 0.5°) are overtraveled directly.

Overload factor

The overload factor restricts step changes in the machine axis velocity at block ends. To ensure that the velocity jump does not exceed the maximum load on the axis, the jump is derived from the acceleration of the axis.

The overload factor indicates the extent by which the acceleration of the machine axis (MD32300 $MA_MAX_AX_ACCEL) may be exceeded for an IPO-cycle.

The velocity jump results as follows:

Velocity jump = axis acceleration * (overload factor-1) * interpolator cycle.

The overload factor is saved in the machine data:

MD32310 $MA_MAX_ACCEL_OVL_FACTOR (overload factor for axial velocity jumps)

Factor 1.0 means that only tangential transitions with finite velocity can be traversed. For all other transitions, the velocity is reduced to zero by changing the setpoint. This behavior is equivalent to the function "Exact stop with interpolator end". This is undesirable for continuous-path mode, so the factor must be set to greater than 1.0.

Note

For startup and installation, please note that the factor must be reduced if the machine is likely to be subject to vibrations during angular block transitions and rounding is not to be used.

By setting the following machine data, the block transitions are rounded independent of the set overload factor with G641/G642:

MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS

Activation/deactivation

Continuous-path mode with a reduction in speed according to the overload factor can be activated in any NC part program block by the modal command `G64`.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode G64 can be deactivated by selecting:

- Modal exact stop G60
- Rounding G641, G642, G643, G644 or G645

Implicit continuous-path mode

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

5.3.3 Rounding

Function

The "Rounding" function adds intermediate blocks (positioning blocks) along a programmed contour (path axes) at non-continuous (angular) block transitions so that the resulting new block transition is continuous (tangential).

Synchronized axes

The rounding considers not only the geometry axes but also all synchronous axes. Although a continuous block transition cannot be created for both axis types concurrently for the parallel travel of path and synchronous axes. In this case, to favor path axes that always travel exactly, only an approximately continuous block transition is created for synchronous axes.

Rounding for G64

Rounding is performed even when to observe the dynamic-response limits at the block transition, a speed is required that for G64 exceeds the permitted speed at the block transition (see Section "Velocity reduction according to overload factor (Page 199)" "Overload factor").

Impact on synchronization conditions

The use of rounding shortens the programmed blocks between which the rounding block(s) are added. The programmed block boundary disappears and are then no longer available as criterion for any synchronization conditions (e.g. auxiliary function output parallel to motion, stop at block end).

Note

We recommend that when the "Rounding" function is used, synchronization conditions apply to the end of the block before the rounding location rather than the end of the inserted rounding block. The following block would then not be started and with a stop at the block end, the contour of the following block can still be changed manually.

Exceptions

In the following cases, **no** rounding occurs at the block transition, for example, between the N10 blocks after N20, i.e. no rounding block is added:

Implicit stopping of the traversing motion

Possible causes:

- Auxiliary function output active before the traversing motion for N20
- N20 does not contain any traversing motion for path axes
- In N20, an axis that was previously a positioning axis traverses as a path axis for the first time
- In N20, an axis that was previously a path axis traverses as a positioning axis for the first time
- Geometry axes traverse in N10 but not in N20
- Geometry axes traverse in N20 but not in N10
- Activation of thread-cutting `G33` in N20
- Change from BRISK and SOFT
- Transformation-relevant axes are not completely assigned to the path motion (e.g. for oscillation axes, positioning axes).

The insertion of the rounding block would slow part program machining overproportionally

Possible causes:

- A program or program section consists of a number of very short traversal blocks (≈ 1 interpolator cycle / traversal block; because each traversal block requires at least one interpolator cycle, the inserted intermediate block would almost double the machining time)
- G64 (path control operation without rounding) without speed reduction active for block change
- The parameterized overload factor (MD32310 $MA_MAX_ACCEL_OVL_FACTOR) permits the traversal of the programmed contour without the path speed needing to be reduced. See also: MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS

Path parameters inhibit the rounding

Possible causes:

- `G641` (path control operation with rounding in accordance with the path criterion) is active but rapid traverse is active (G0) AND ADISPOS == 0 (rounding separation for G0)
- `G641` (path control operation with rounding in accordance with the path criterion) is active but rapid traverse is **not** active AND ADIS == 0 (rounding separation for path functions G1, G2, G3, ...)
- `G642` or `G643` (path control operation with rounding while observing defined tolerances) is active but all tolerances == zero

N10 or N20 does not contain traversing motion (zero block).

Normally no zero blocks are created. Exceptions:

- Active synchronous action
- Program jumps

Impact on synchronization conditions

The programmed blocks between which the rounding contour is added are shortened during rounding. The original programmed block boundary disappears and is then no longer available for any synchronization conditions (e.g. auxiliary function output parallel to motion, stop at block end).

Note

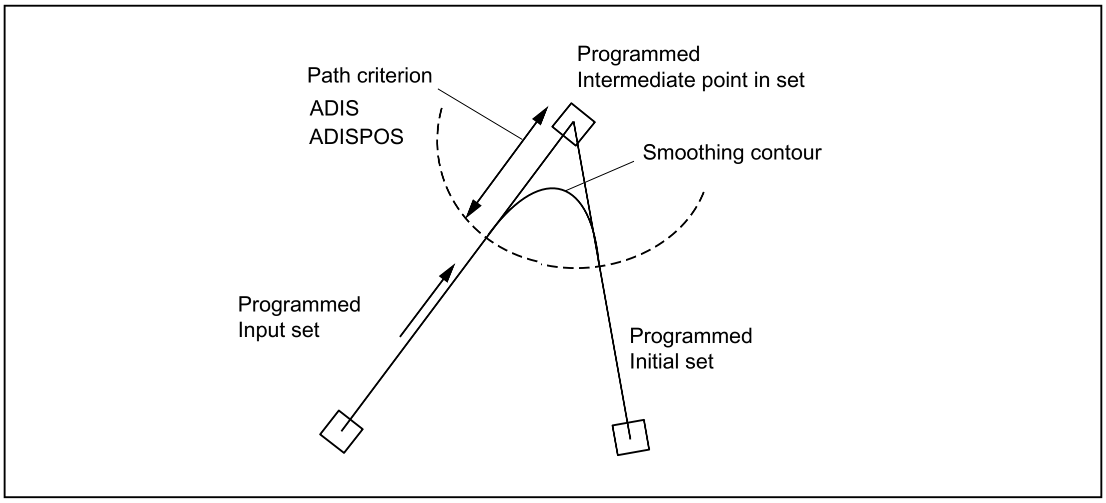
We recommend that when the "Rounding" function is used, synchronization conditions apply to the end of the block before the rounding location rather than the end of the inserted rounding block. The following block would then not be started and with a stop at the block end, the contour of the following block can still be changed.

5.3.3.1 Rounding according to a path criterion (G641)

Function

In continuous-path mode with rounding according to a path criterion, the size of the rounding area is influenced by the path criteria ADIS and ADISPOS.

The path criteria ADIS and ADISPOS describe the maximum distances which a rounding block can occupy before and after a block.



Note

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

Note

ADISPOS is programmed in the same way as ADIS, but must be used specifically for movements in rapid traverse mode G0.

Scope of the path criterion

- ADIS or ADISPOS must be programmed. If the default is "zero", G641 behaves like G64.
- If only one of the blocks involved is rapid traverse G0, the smaller rounding distance applies.

- If a very small value is used for ADIS, the controller must make sure that every interpolated block, even an intermediate rounding block, contains at least one interpolation point. The maximum path velocity is thereby limited to ADIS / interpolator cycle.
- Irrespective of ADIS and ADISPOS, the rounding area is limited by the block length.
  In blocks with short distances (distance < 4* ADIS and < 4 * ADISPOS respectively), the rounding distance is reduced so that a traversable part of the original block is retained. The remaining length depends on the axis path and is approximately 60% of the distance still to be traversed in the block. ADIS or ADISPOS is therefore reduced to the remaining 40% of the distance to be traversed. This algorithm prevents a rounding block being inserted for a very small change in contour. In this case, switchover to continuous-path mode G64 is automatic until rounding blocks can be inserted again.

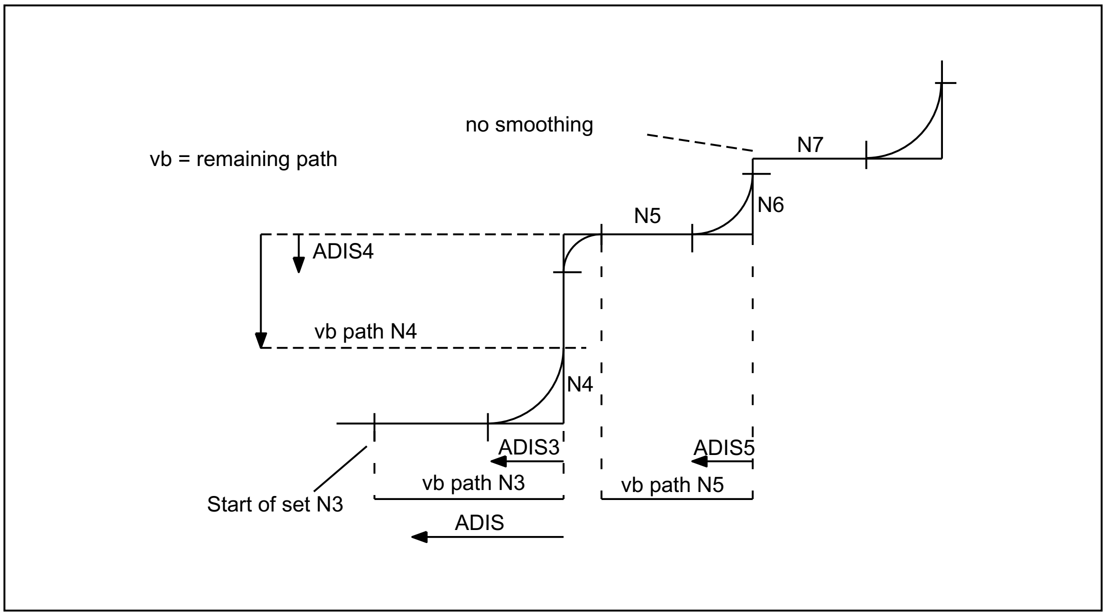


Figure 5-4 Path with limitation of ADIS

Activation/deactivation

Continuous-path mode with rounding based on a path criterion can be activated in any NC part program block by the modal command `G641`. Before or on selection, the path criteria `ADIS/ADISPOS` must be specified.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding based on a path criterion (G641) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G642, G643, G644 or G645

Program example

Program code	Comment
`N1 G641 Y50 F10 ADIS=0.5`	`; Continuous-path mode with rounding based on a path criterion (rounding clearance: 0.5 mm)`
`N2 X50`	
`N3 X50.7`	
`N4 Y50.7`	
`N5 Y51.4`	
`N6 Y51.0`	
`N7 X52.1`	

5.3.3.2 Rounding in compliance with defined tolerances (G642/G643)

Function

In continuous-path mode involving rounding in compliance with defined tolerances, the rounding normally takes place while adhering to the maximum permissible path deviation. Instead of these axis-specific tolerances, the maintenance of the maximum contour deviation (contour tolerance) or the maximum angular deviation of the tool orientation (orientation tolerance) can be configured.

Activation

Continuous-path mode with rounding in compliance with defined tolerances can be activated in any NC part program block by the modal command `G642` or `G643`.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding in compliance with defined tolerances (G642/G643) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G644 or G645

Differences between G642 - G643

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

G642	G643
With G642, the rounding path is determined on the basis of the shortest distance for rounding all axes. This value is taken into account when generating a rounding block.	In the case of G643, each axis may have a different rounding path. The rounding travels are taken into account axis-specifically and block-internally (⇒ no separate rounding block).
With G642, the rounding area results from the smallest tolerance setting.	Very different specifications for the contour tolerance and the tolerance of the tool orientation can only have effect with G643.

Parameterization

Maximum path deviation

The maximum path deviation permitted with G642/G643 is set for each axis in the machine data:

MD33100 $MA_COMPRESS_POS_TOL

Contour tolerance and orientation tolerance

The contour tolerance and orientation tolerance are set in the channel-specific setting data:

SD42465 $SC_SMOOTH_CONTUR_TOL (maximum contour deviation)

SD42466 $SC_SMOOTH_ORI_TOL (maximum angular deviation of the tool orientation)

The settings data can be programmed in the NC program and can in this way be specified differently for each block transition.

Note

The setting data SD42466 $SC_SMOOTH_ORI_TOL is effective only in active orientation transformation.

Rounding behavior

Rounding behavior with G642 and G643 is configured via the machine data:

MD20480 $MC_SMOOTHING_MODE (rounding behavior with G64x)

The units positions (**E**) define the behavior for G643, the tens positions (**Z**) the behavior for G642:

Value E or Z	Meaning
0	**All axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL
1	**Geometry axes:** Rounding by maintaining the contour tolerance: SD42465 $SC_SMOOTH_CONTUR_TOL **Remaining axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL
2	**Geometry axes:** Rounding by maintaining the orientation tolerance: SD42466 $SC_SMOOTH_ORI_TOL **Remaining axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL

Value E or Z	Meaning
3	**Geometry axes:** Rounding by maintaining the contour tolerance and the orientation tolerance: SD42465 $SC_SMOOTH_CONTUR_TOL SD42466 $SC_SMOOTH_ORI_TOL **Remaining axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL
4	**All axes:** The rounding length programmed with `ADIS` or with `ADISPOS` is used (as in case of G641). Any axis-specific tolerance or contour and orientation tolerance specifications are ignored.

Profile for limit velocity

The use of a velocity profile for rounding in compliance with defined tolerances is controlled via the hundreds position in MD20480:

Value	Meaning	
< 100:	A profile of the limit velocity is calculated within the rounding area, based on the defined maximum values for acceleration and jerk on the participating axes or path. This can lead to an increase in the path velocity in the rounding area and therefore to the acceleration of the participating axes.	
≥100:	A profile of the limit velocity is not calculated for rounding blocks with G641/G642. A constant velocity limit is specified instead. This prevents the participating axes being accelerated into the rounding area during rounding with G641/G642. However, in certain cases, this setting can cause the rounding blocks to be traversed too slowly, especially in large rounding areas.	
	1xx:	No velocity profile for G641
	2xx:	No velocity profile for G642

Note

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

Supplementary conditions

Restriction for protection zones with active radius compensation and tool orientation:

Although tool radius compensation is applied for a tool orientation, which is not perpendicular to one of the three datum planes of the basic coordinate system, the protection zones are not rotated onto the corresponding plane.

For G643 the following must apply:

MD28530 $MC_MM_PATH_VELO_SEGMENTS > 0 (number of memory elements for limiting the path velocity)

If this condition is met, then it must be applicable for all axes:

MD35240 $MC_ACCEL_TYPE_DRIVE = FALSE (acceleration characteristic DRIVE for axes on/off)

5.3.3.3 Rounding with maximum possible axial dynamic response (G644)

Function

Maximizing the dynamic response of the axes is key to this type of continuous-path mode with rounding.

Note

Rounding with G644 is only possible if:

- All the axes involved contain only a linear motion in both the observed blocks.
- **No** kinematic transformation is active

In case an involved axis contains a polynomial (polynomial programmed, spline active, compressor active) or a kinematic transformation is active, the block transition is rounded with G642.

Activation

Continuous-path mode with rounding with the maximum possible axial dynamic response can be activated in any NC part program block by the modal command `G644`.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding with the maximum possible axial dynamic response (G644) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G642, G643 or G645

Parameterization

Rounding behavior with G644 is configured via the thousands and tens of thousands places in the machine data:

MD20480 $MC_SMOOTHING_MODE (rounding behavior with G64x)

Value	Meaning
Thousand's place:	
0xxx:	When rounding with G644, the maximum deviations for each axis specified by the following machine data are respected: MD33100 $MA_COMPRESS_POS_TOL If the dynamics of the axis permit, then any specified tolerance is not utilized.
1xxx:	Input the maximum rounding path by programming `ADIS=`... or `ADISPOS=`...(as for G641)

Value	Meaning
2xxx:	Input the maximum possible frequencies of each axis in the rounding area using the machine data: MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead) The rounding area is defined so that no frequencies in excess of the specified maximum can occur while the rounding motion is in progress.
3xxx:	Any axis that has a velocity jump at a corner traverses around the corner with the maximum possible dynamic response (maximum acceleration and maximum jerk). SOFT: If SOFT is active, the maximum acceleration **and** the maximum jerk of each axis is maintained. BRISK: If BRISK is active, only the maximum acceleration and **not** the maximum jerk of each axis is maintained. With this setting, neither the maximum deviations nor the rounding distance are checked. The resulting deviations or rounding distances are determined exclusively by the dynamic limits of the respective axis and the current path velocity.
4xxx:	As in case of 0xxx, the maximum deviations of each axis specified with the following machine data are used: MD33100 $MA_COMPRESS_POS_TOL Contrary to 0xxx, the specified tolerance is also utilized, if possible. Therefore, the axis does not attain its maximum possible dynamics.
5xxx:	As in case of 1xxx, the maximum possible rounding path is specified through programming of `ADIS=`... or `ADISPOS=` respectively. Contrary to 1xxx, the specified rounding path is also utilized, if possible. Therefore, the axes involved do not attain their maximum possible dynamics.
Ten thousands digit	
0xxxx	The velocity profiles of the axes in the rounding area are determined without jerk limiting for BRISK and with jerk limiting for SOFT.
1xxxx	The velocity profiles of the axes in the rounding area are always determined with jerk limiting, regardless of whether BRISK or SOFT is active.

When specifying the maximum axial deviations (MD33100 $MA_COMPRESS_POS_TOL) or the maximum rounding distance (ADIS / ADISPOS) the available rounding path is normally not used, if permitted by the dynamics of the axes involved. Through this, the length of the rounding path depends on the active path feedrate. In case of lower path speeds, one gets lower deviations from the programmed contours. However, it can be set that in these cases the specified maximum axial deviation or the specified rounding distance is utilized, if possible. In this case the deviations depend on the programmed contour independent of the programmed path feedrate.

Note

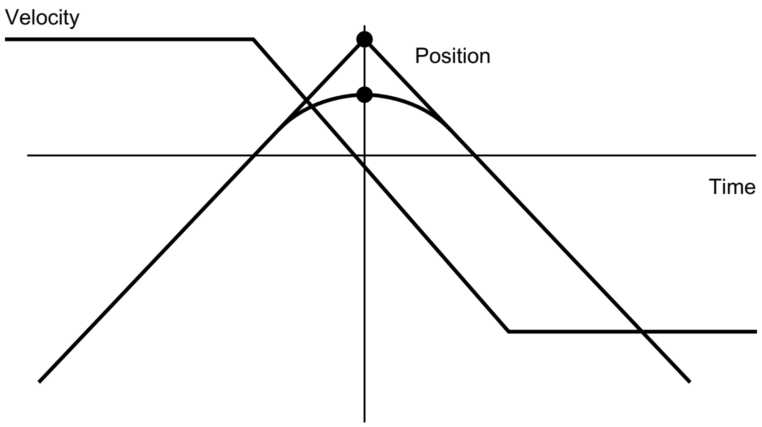
Apart from the ones mentioned, the following limitation can also become active additionally:

The rounding distance cannot exceed half the length of the original participating blocks.

Jerk limitation

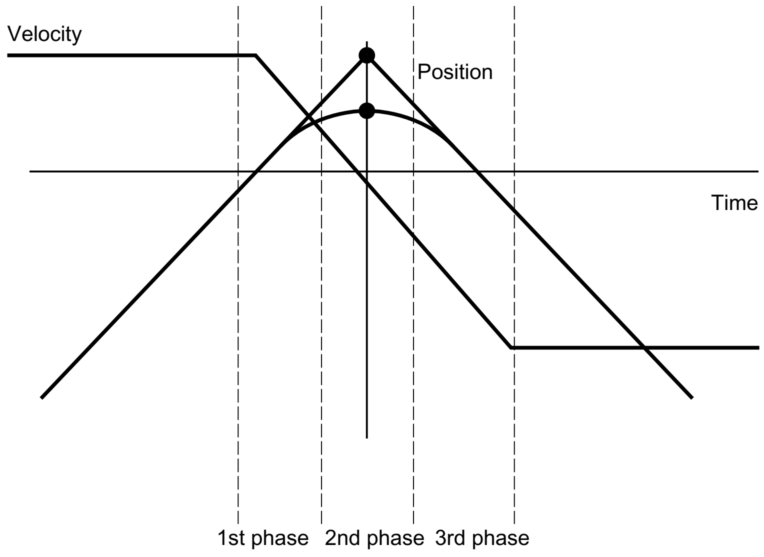
The smoothing of the velocity jump on each axis and thus the shape of the rounding path depends on whether an interpolation is performed with or without jerk limitation.

Without jerk limitation the acceleration of each axis reaches its maximum value in the entire rounding area.



With jerk limitation, the jerk of each axis is limited to its maximum value within the rounding area. The rounding motion thus generally consists of three phases:

- **Phase 1**
  During phase 1, each axis builds up its maximum acceleration. The jerk is constant and equal to the maximum possible jerk on the respective axis.
- **Phase 2**
  During phase 2, the maximum permissible acceleration is applied.
- **Phase 3**
  During phase 3, which is the last phase, the acceleration of each axis is reduced back to zero with the maximum permissible jerk.



5.3.3.4 Rounding of tangential block transitions (G645)

Function

In continuous-path mode with rounding, rounding blocks are also only generated on tangential block transitions if the curvature of the original contour exhibits a jump in at least one axis.

The rounding motion is defined here so that the acceleration of all axes involved remains smooth (no jumps) and the parameterized maximum deviations from the original contour (MD33120 $MA_PATH_TRANS_POS_TOL) are not exceeded.

In the case of angular, non-tangential block transitions, the rounding behavior is the same as with G642 (see Section "Rounding in compliance with defined tolerances (G642/G643) (Page 206)").

Activation/deactivation

Continuous-path mode with rounding of tangential block transitions can be activated in any NC part program block by the modal command `G645`.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding of tangential block transitions (G645) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G642, G643 or G644

Comparison between G642 and G645

When rounding with G642, the only block transitions rounded are those which form a corner, i.e. the velocity of at least one axis jumps. However, if a block transition is tangential, but there is a jump in the curvature, no rounding block is inserted with G642. If this block transition is traversed with finite velocity, the axes experience some degree of jump in acceleration which (with the jerk limit activated!) may not exceed the parameterized limit (MD32432 $MA_PATH_TRANS_JERK_LIM). Depending on the level of the limit, the path velocity at the block transition may be greatly reduced as a result. This constraint is avoided by using G645 because the rounding motion is defined here in such a way that no jumps occur in acceleration.

Parameterization

The following machine data indicates the maximum permissible path deviation for each axis during rounding with G645:

MD33120 $MA_PATH_TRANS_POS_TOL

This value is only of relevance to tangential block transitions with variable acceleration. When angular, non-tangential block transitions are rounded, (as with G642) the tolerance from MD33100 $MA_COMPRESS_POS_TOL becomes effective.

See also

Free-form surface mode: Basic functions (Page 237)

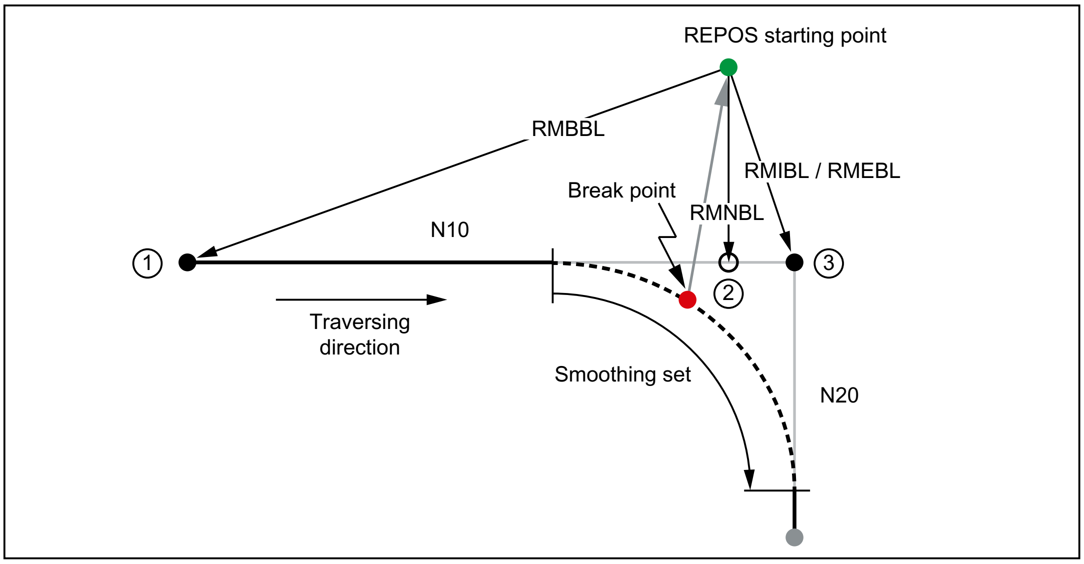
5.3.3.5 Rounding and repositioning (REPOS)

If the machining in the area of the rounding contour is interrupted, a REPOS operation **cannot** be used to position again directly on the rounding contour. In this case, positioning can be made only on the **programmed** contour.

Example

Programmed: Two traversing blocks N10 and N20 with programmed rounding G641.

The traversing motion is interrupted in the rounding area. The axes, e.g. manually, are then traversed to the REPOS start point. Depending on the selected REPOS mode, the repositioning on the contour is made at the points ①, ② or ③.



RMBBL	Repositioning at the start of the interrupted traversal block
RMIBL	Repositioning at the interruption location
RMEBL	Repositioning at the end of the interrupted traversal block
RMNBL	Repositioning at the next contour point
①	Block start N10
②	To the REPOS start point of the next contour point
③	Block end N10 / block start N20

5.3.4 LookAhead

5.3.4.1 Standard functionality

Function

LookAhead is a function which is active in continuous-path mode (G64, G64x) and determines a foreseeable velocity control for multiple NC part program blocks over and beyond the current block.

Note

LookAhead is only available for path axes, **not** for spindles and positioning axes.

If a part program contains consecutive blocks with very small paths, only one velocity is reached per block without LookAhead, enabling deceleration of the axes at the end of the block while maintaining acceleration limits. This means that the programmed velocity is not reached at all. With LookAhead, however, it is possible to implement the acceleration and deceleration phase over multiple blocks with approximately tangential block transitions, thereby achieving a higher feedrate with shorter distances.

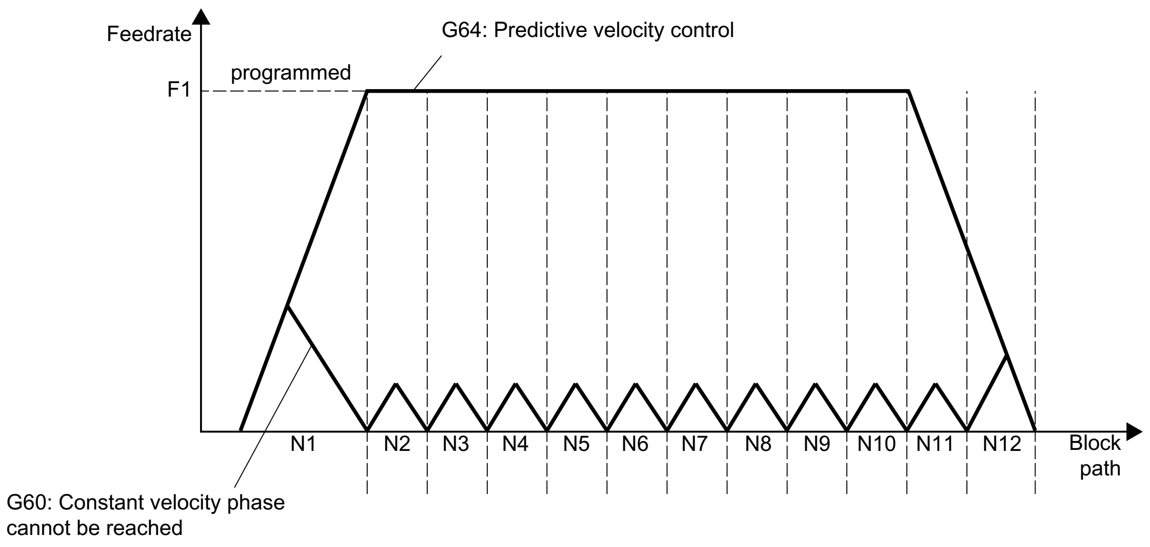


Figure 5-5 Velocity control with short distances and exact stop G60 or continuous-path mode G64 with LookAhead

Deceleration to velocity limits is possible with LookAhead such that violation of the acceleration and velocity limit is prevented.

LookAhead takes plannable velocity limits into consideration such as:

- Exact stop at block end
- Velocity limit in the block
- Acceleration limit in the block

- Velocity limit on block transition
- Synchronization with block change at block transition.

Mode of operation

LookAhead carries out a block-specific analysis of velocity limits and specifies the required brake ramp profile based on this information. LookAhead is adapted automatically to block length, braking capacity and permissible path velocity.

For safety reasons, the velocity at the end of the last prepared block must initially be assumed to be zero because the next block might be very small or be an exact-stop block, and the axes must have been stopped by the end of the block.

With a series of blocks with high set velocity and very short paths, the speed can be increased in each block depending on the velocity value currently calculated by the LookAhead function in order to achieve the required set velocity. After this it can be reduced so that the velocity at the end of the last block considered by the LookAhead function can be zero. This results in a serrated velocity profile (see the following fig.) which can be avoided by reducing the set velocity or increasing the number of blocks considered by the LookAhead function.

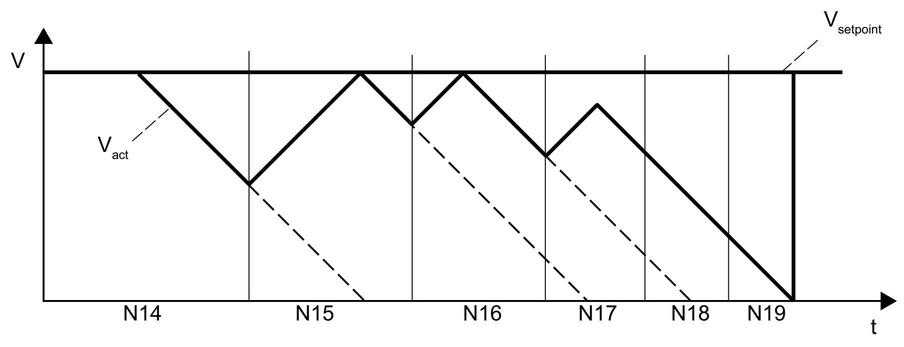


Figure 5-6 Example for modal velocity control (number of blocks considered by the LookAhead function = 2)

Activation/deactivation

LookAhead is activated by selecting continuous-path mode G64, G641, G642, G643, G644 or G645.

Selecting the exact stop which works on a non-modal basis enables rounding to be interrupted (G09).

LookAhead is deactivated by selecting the modal exact stop (G60).

Parameterization

Number of blocks

To achieve reliable axis traversal in continuous-path mode, the feedrate must be adapted over several blocks. The number of blocks considered by the LookAhead function is calculated automatically and can, if required, be limited by a machine data. The default setting is "1", which means that LookAhead only considers the following block for velocity control.

Because LookAhead is especially important for short blocks (relative to the deceleration path), the number of blocks required is of interest for LookAhead braking. It is sufficient to consider the path length to be equal to the deceleration path that is required to brake from maximum velocity to standstill.

For a machine with a low axial acceleration of a = 1 m/s^2 and a high feedrate of v_{path} = 10 m/min, the following number of $n_{LookAhead}$ blocks are allocated to the controller where it has has an attainable block cycle time of TB = 10 ms:

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

Considering these conditions, it is advisable to adapt the feedrate over 10 blocks. The number of blocks entered for the LookAhead function forecast does not change the LookAhead algorithm and memory requirement.

Since the machining velocity is very often set to a lower value than the maximum velocity in a program, more blocks than are required would be predicted, overloading the processor unnecessarily. For this reason, the required number of blocks is derived from the velocity which is calculated from the following multiplication:

- Programmed velocity * MD12100 $MN_OVR_FACTOR_LIMIT_BIN
  (when using a **binary**-coded feedrate override switch)
- Programmed velocity * MD12030 $MN_OVR_FACTOR_FEEDRATE[**30**]
  (when using a **Gray**-coded feedrate override switch)

The value for MD12100 or the 31st override value for MD12030 defines the dynamic response reserves which the velocity control provides for when the path feedrate is overshot.

Note

The 31st override value for MD12030 should correspond to the highest override factor which is actually used.

Note

The number of blocks considered by the LookAhead function is limited by the possible number of NC blocks in the IPO buffer.

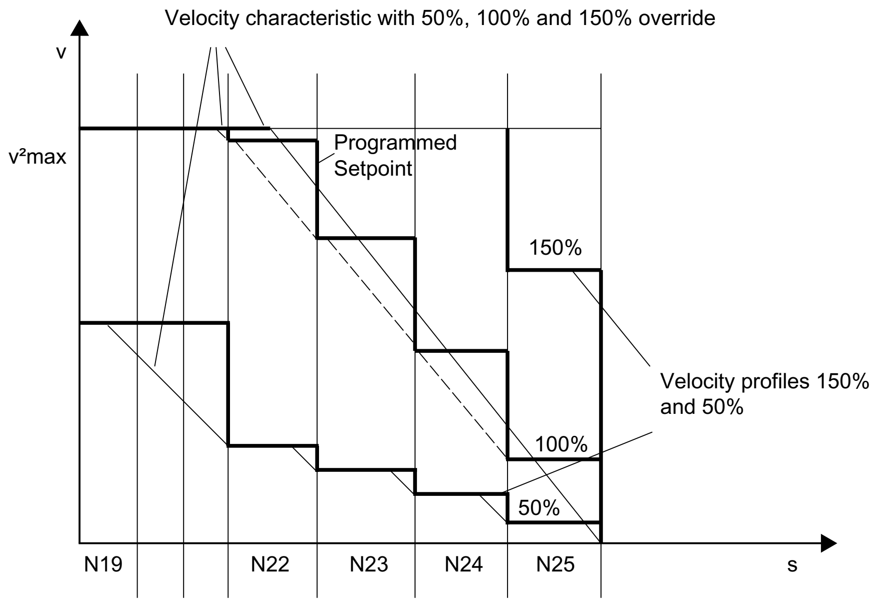
Velocity profiles

In addition to the fixed, plannable velocity limitations, LookAhead can also take account of the programmed velocity. This makes it possible to achieve a lower velocity by applying LookAhead beyond the current block.

- **Determination of the following block velocity**
  One possible velocity profile contains the determination of the following block velocity. Using information from the current and the following NC block, a velocity profile is calculated from which, in turn, the required velocity reduction for the current override is derived.
  The calculated maximum value of the velocity profile is limited by the maximum path velocity. With this function it is possible to initiate a speed reduction in the current block taking override into account such that the lower velocity of the following block can be achieved. If the reduction in velocity takes longer than the travel time of the current block, the velocity is further reduced in the following block. Velocity control is only ever considered for the following block.
  The function is activated via the machine data:
  MD20400 $MC_LOOKAH_USE_VELO_NEXT_BLOCK

Value	Meaning
TRUE	Function active
FALSE	Function **not** active

- **Definition of override points**
  If the velocity profile of the following block velocity is not sufficient because, for example, very high override values (e.g. 200%) are used or a constant cutting rate G96/G961 is active, with the result that the velocity must be further reduced in the following block, LookAhead provides a way of reducing the programmed velocity over several NC blocks. By defining override points, LookAhead calculates a limiting velocity profile for each value. The required velocity reductions for the current override are derived from these profiles. The calculated maximum value of the velocity profile is limited by the maximum path velocity. The upper point should cover the velocity range that will be reached by the maximum value set in the machine data:
  MD12030 $MN_OVR_FACTOR_FEEDRATE[n] (evaluation of the path feedrate override switch)
  It can also be reached via the value of the machine data:
  MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch)
  In this way, a reduction of the velocity continuing into the block in which it is programmed can be avoided.
  If velocity reductions across block boundaries are required at a 100% override, a point should be set in the lower override range as well.
  The number of override points used per channel is specified in the machine data:
  MD20430 $MC_LOOKAH_NUM_OVR_POINTS (number of override switch points for LookAhead)
  The associated points are stored in the machine data:
  MD20440 $MC_LOOKAH_OVR_POINTS (override switch points for LookAhead)
  Example:
  Limiting velocity characteristics, whereby:
  - Override = 50%, 100% or 150%
  - Number of LookAhead blocks = 4
  - MD20430 $MC_LOOKAH_NUM_OVR_POINTS = 2
  - MD20440 $MC_LOOKAH_OVR_POINTS = 1.5, 0.5
  - MD20400 $MC_LOOKAH_USE_VELO_NEXT_BLOCK = 1



A combination of both procedures (determination of following block velocity and determination of override points) can be used to calculate the velocity profiles and is generally advisable because the preset machine data for these functions already takes the widest range of override-dependent velocity limits into account.

Note

If neither of the procedures has been activated, the setpoint velocity is always applied in the current block.

Note

Plannable velocity limits restrict override-specific velocity limits.

Relief factor with block cycle problems

Block cycle problems are encountered in cases where the traversing distances of the NC blocks to be processed are so short that the LookAhead function has to reduce the machine velocity to provide enough time for block processing. In this situation, constant braking and acceleration of path motion may occur.

Velocity fluctuations of this type can be dampened by specifying a relief factor:

MD20450 $MC_LOOKAH_RELIEVE_BLOCK_CYCLE (relieving factor for the block cycle time)

Supplementary conditions

Axis-specific feed stop/axis disable

Axis-specific feed stop and axis-specific axis disable are ignored by LookAhead.

If an axis is to be interpolated that should on the other hand be made stationary by axis-specific feed stop or axis disable, LookAhead does not stop path motion before the block in question but decelerates **in** the block itself.

If this response is not wanted, an **axis**-specific feed stop can be transferred to a **channel**-specific feed stop via the PLC to stop the path immediately (see also Section "Function (Page 93)").

5.3.4.2 Free-form surface mode: Extension function

Function

The "Free-form surface mode: Extension function" is an extension of the Look Ahead standard functionality and is used to calculate the path velocity profile during free-form surface machining (see also Section "Free-form surface mode: Basic functions (Page 237)").

Its use optimizes the continuous-path mode as follows:

- Symmetry between the acceleration and deceleration profiles
- Uniform acceleration process, even with changing jerk or acceleration limits

- Uniform acceleration process of target velocity profiles, irrespective of the degree to which they can or cannot be started with the specified dynamic response limit
- Look Ahead braking to lower setpoint velocities

Uniformity and compliance with the dynamic response limit guarantee that the setpoint velocity profiles are smoothed to a homogeneous velocity profile on the part. This serves to minimize the effect of following errors on the quality of the surface.

Advantages

- Greater uniformity in the surface of the workpiece
- Lower machine load

Applications

The "Free-form surface mode: Extension function" is used to machine workpieces which primarily comprise free-form surfaces.

Note

As better results are not achieved for standard machining applications, standard Look Ahead functionality should be used in these cases.

Activation

The function is only effective:

- In AUTOMATIC
- In the "Acceleration with jerk limit (SOFT)" mode

Parameterization

Working memory

The memory for the "Free-form surface mode: Extended function" is configured via the machine data:

MD28533 $MC_MM_LOOKAH_FFORM_UNITS = <value>

The required memory depends on the part program, the block lengths, the axis dynamic response, as well as on an active kinematic transformation.

The following setting applies as a guideline for machining free-form surfaces: MD28533 = 18

Note

Due to the additional storage requirements, MD28533 should only be set for the channels in which free-form surfaces are being machined.

Number of NC blocks in the IPO buffer

It is generally advisable to significantly increase the configured number of NC blocks in the interpolation buffer when using the "Free-form surface mode: Extension function":

MD28060 $MC_MM_IPO_BUFFER_SIZE > 100

If the block memory capacity is too low, this may diminish the uniformity of the path-velocity profile.

Activation/deactivation

The function can be switched on or off independently for every dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 235)"):

MD20443 $MC_LOOKAH_FFORM[<n>]= <value>

Index <n>	Dynamic response mode	<value>	Free-form surface mode: Extension function
0	Standard dynamic response settings (DYNNORM)	0	Off
		1	On
1	Positioning mode, tapping (DYNPOS)	0	Off
		1	On
2	Roughing (DYNROUGH)	0	Off
		1	On
3	Finishing (DYNSEMIFIN)	0	Off
		1	On
4	Smooth finishing (DYNFINISH)	0	Off
		1	On

The "Free-form surface mode: Extension function" is typically only active if the "Free-form surface mode: Basic functions" are also active. Therefore, the settings in MD20443 $MC_LOOKAH_FFORM[<n>] should correspond to the settings in MD20606 $MC_PREPDYN_SMOOTHING_ON[<n>].

The standard Look Ahead functionality is active in the dynamic response modes in which the "Free-form surface mode: Extension function" is switched off.

Programming

Generally speaking, the "Free-form surface mode: Extension function" becomes effective as a result of a change in the dynamic response mode in the part program.

Example

The following parameters are assumed:

MD20443 $MC_LOOKAH_FFORM[0] = 0

MD20443 $MC_LOOKAH_FFORM[1] = 0

MD20443 $MC_LOOKAH_FFORM[2] = 1

MD20443 $MC_LOOKAH_FFORM[3] = 1

MD20443 $MC_LOOKAH_FFORM[4] = 1

Change of the dynamic response mode in the part program:

Program code	Comment
`N10 DYNPOS`	`; Activate DYNPOS dynamic response mode. Standard Look Ahead functionality is active in the DYNPOS dynamic response mode.`
`...`	
`N100 G17 G54 F10000`	
`N101 DYNFINISH`	`; Activate DYNFINISH dynamic response mode. The "Free-form surface mode: Extension functions" are active in the DYNFINISH dynamic response mode.`
`N102 SOFT G642`	
`N103 X-0.274 Y149.679 Z100.000 G0`	
`N104 COMPCAD`	
`...`	
`N1009 Z4.994 G01`	
`N10010 X.520 Y149.679 Z5.000`	
`N10011 X10.841 Y149.679 Z5.000`	
`N10012 X11.635 Y149.679 Z5.010`	
`N10013 X12.032 Y149.679 Z5.031`	
`M30`	

Note

When switching between the standard Look Ahead functionality and the "Free-form surface mode: Extension function" or vice versa, continuous-path mode is interrupted by an interpolator stop.

Supplementary conditions

Automatic function switchover

Use of the following functions results in an automatic switchover to standard Look Ahead functionality:

- Thread cutting/tapping (G33, G34, G35, G331, G332, G63)
- Path master-value coupling
- Punching, nibbling
- Cartesian PTP travel

The "Free-form surface mode: Extension function" is then switched on again automatically.

Using the commands of G group group 15 (feed types)

The following feed types are not recommended in conjunction with the "Free-form surface mode: extension function" function:

- Feedrate per revolution (G95, G96, G97, etc.)
- Inverse-time feedrate (G93)

Use of FLIN

The FLIN feedrate profile cannot be used in conjunction with the "Free-form surface mode: extension function" function:

Influence of feedrate overrides

Feedrate overrides (via a machine control panel, $AC_OVR, ...) can extend the traverse time over standard Look Ahead functionality considerably.

Interaction with rapid traverse motion (G0)

G0 blocks which are interspersed during free-form surface machining do not switch the Look Ahead functionality over (from the "Free-form surface mode: Extended function" to the standard Look Ahead functionality or vice versa). This means that even though the standard dynamic response setting (DYNNORM) is effective with G0, the standard Look Ahead functionality which is preset for DYNNORM (→ MD20443 $MC_LOOKAH_FFORM[0]) does not automatically become effective as well. By retaining the Look Ahead functionality which is currently active, a more homogeneous velocity profile is achieved, particularly since G0 and polynomial blocks are usually smoothed and connected by rounding.

5.4 Dynamic adaptations

5.4.1 Smoothing of the path velocity

Introduction

The velocity control function utilizes the specified axial dynamic response. If the programmed feedrate cannot be achieved, the path velocity is brought to the parameterized axial limit values and the limit values of the path (velocity, acceleration, jerk). This can lead to repeated braking and acceleration on the path.

If a short acceleration takes place during a machining function with high path velocity, and is thus followed almost immediately by braking, the reduction in the machining time is only minimal. Acceleration of this kind can, however, have undesirable effects if, for example, it results in machine resonance.

In some applications in mold making, especially in the case of high-speed cutting, it is desirable to achieve a constant path velocity. In these cases, it can therefore be reasonable to sacrifice transient acceleration processes in favor of a smoother tool path velocity.

Function

If the "smoothing the path velocity" function is active, a smoothing factor, which determines the maximum permissible productivity loss, takes effect with a view to achieving smoother path velocity control: Acceleration processes which contribute less than this factor to a shorter program runtime are not performed. Account is only taken of acceleration processes whose frequencies lie above the configurable limit frequencies of of the axes involved.

Benefits:

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

Note

The smoothing of the path velocity does not lead to contour errors.

Variations in axis velocity due to curvatures in the contour at constant path velocity may continue to occur and are not reduced with this function.

Variations in path velocity due to the input of a new feedrate are not changed either. This remains the responsibility of the programmer of the subprogram.

Requirements

- The smoothing of the path velocity is only effective in continuous-path mode with LookAhead over multiple blocks with SOFT and BRISK. Smoothing is **not** effective with G0.
- The controller's cycle times must be configured in such a way that preprocessing can prepare sufficient blocks to enable an acceleration process to be analyzed.

Activation/deactivation

The "smoothing of the path velocity" function is activated/deactivated with the machine data:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

Value	Meaning
0.0	Smoothing of the path velocity **not** active (default)
> 0	Smoothing of the path velocity active

A change in the MD setting is only made effective through NEW CONF.

Parameterization

Smoothing factor

The smoothing factor is set via the channel-specific machine data:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

The percentage value defines how much longer a processing step without accelerations/ decelerations may be than the corresponding step with accelerations/decelerations.

This would be a "worst-case" value, if all accelerations within the part program, except the initial approach motion, were smoothed. The actual extension will always be smaller, and may even be 0, if the criterion is not met by any of the accelerations. Values between 50 and 100% may also be entered without significantly increasing the machining time.

Consideration of the programmed feed

The path velocity can be smoothed with or without taking the programmed feedrate into consideration. The selection is made via the machine data:

MD20462 $MC_LOOKAH_SMOOTH_WITH_FEED (path smoothing with programmed feedrate)

Value	Meaning
0	Programmed feedrate is **not** taken into consideration.
1	Programmed feedrate is considered (default setting).

When considering the programmed feedrate, the specified smoothing factor (see MD20460) is maintained better when the override is 100%.

Axis-specific limit frequencies

The axis-specific limit frequencies are defined via the machine data:

MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead)

Acceleration and deceleration processes, which run with a high frequency, are smoothed depending upon the parameterization of the following machine data or else are reduced in dynamics:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

MD20465 $MC_ADAPT_PATH_DYNAMIC (adaptation of the dynamic path response)

For further information on MD20465, see Section "Adaptation of the dynamic path response (Page 227)".

Note

If vibrations are generated in the mechanical system of an axis and if the corresponding frequency is known, MD32440 should be set to a value smaller than this frequency.

The needed resonance frequencies can be calculated using the built-in measuring functions.

Mode of operation

The minimum value for MD32440 is calculated as f_{path} on the basis of the axes involved in the path. For the smoothing only those acceleration processes are taken into consideration, in which the start and the end velocity of this motion are reached within the time given below:

$t = t_2 - t_1 = 2 / f_{path}$

These acceleration processes are dispensed with if the resulting extension in the processing time does not exceed the limit specified in excess of the smoothing factor (MD20460).

Example

The following parameters are assumed:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 10%

MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 Hz

MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20 Hz

MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 10 Hz

The path involves the three axes X = AX1, Y = AX2, Z = AX3.

The minimum value of MD32440 for these three axes is thus 10 Hz. This means that any acceleration, which is completed within a period of $t_2 - t_1$ = 2/10 Hz = 200 ms, is examined. The time t_2 is the time it takes to reach velocity v_1 again following an acceleration process starting from velocity v_1. The extending of the execution time is also only considered within this range.

If the time $t_2 - t_1$ is greater than 200 ms or if the additional program execution time $t_3 - t_2$ is more than 10% (= MD20460) of $t_2 - t_1$, the following time characteristic applies:

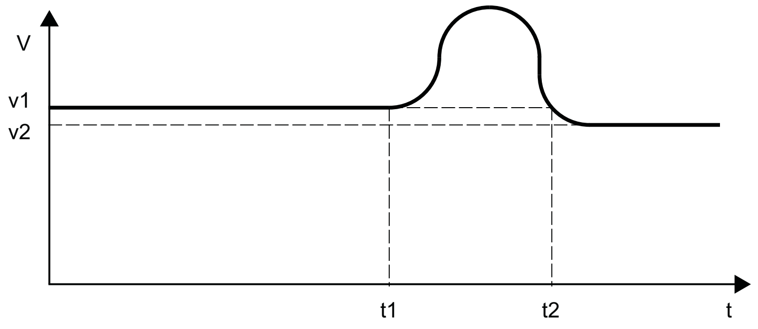


Figure 5-7 Characteristic of time-optimum path velocity (without smoothing)

If, however, the time $t_2 - t_1$ is less than 200 ms or if the additional program execution time $t_3 - t_2$ is no more than 10% of $t_2 - t_1$, the following time characteristic applies:

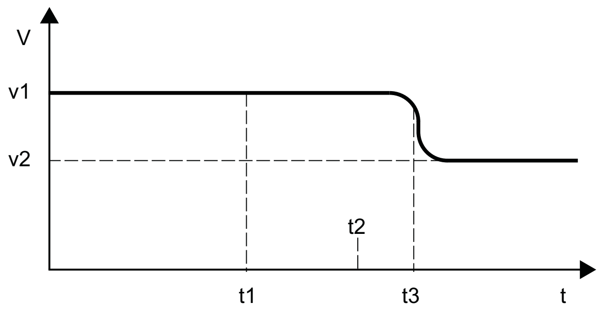


Figure 5-8 Characteristic of the smoothed path velocity

5.4.2 Adaptation of the dynamic path response

Function

Highly dynamic acceleration and deceleration processes during machining can cause excitation of mechanical vibrations of machine elements and consequently a reduction of the surface quality of the workpiece.

The dynamic response of the acceleration and deceleration processes can be adapted to the machine conditions using the "adaptation of the dynamic path response" function.

Note

The "adaptation of the dynamic path response" function only concerns the resulting path and not the deceleration and acceleration processes of the individual axes involved in the path. For this reason, critical deceleration and acceleration processes of the axes with respect to the excitation of mechanical vibrations can occur due to discontinuous contour profiles or kinematic transformations, even with a constant path velocity profile.

Effectiveness

The "adaptation of the dynamic path response" function is only effective during path motions:

- Continuous-path mode (G64, G64x)
  In continuous-path mode, the optimal effect of the dynamic response adaptation is attained with an active 100% override. Considerable deviations from this value or functions that cause the path axes to decelerate (e.g. auxiliary function outputs to the PLC) greatly reduce the desired action.
- Exact stop (G60)

In addition, the "adaptation of the dynamic path response" function is **not** active during path motions:

- Programmed rapid traverse (G0)
- Changes in the override value
- Stop requests during motion (e.g. NC Stop, NC Reset)
- "Velocity-dependent path acceleration" function (DRIVE) is active

Activation/deactivation

The function is activated/deactivated with the machine data:

MD20465 $MC_ADAPT_PATH_DYNAMIC (adaptation of the dynamic path response)

Value	Meaning
= 1.0	Dynamic adaptation **not** active (default setting)
> 1.0	Dynamic adaptation active

When activation takes place, the **"smoothing the path velocity" function is always activated** internally in continuous-path mode as well (see Section "Smoothing of the path velocity (Page 223)").

If the smoothing factor (MD20460 $MC_LOOKAH_SMOOTH_FACTOR) is set to 0% (= function deactivated; default!), a smoothing factor of 100% is used as a substitute. For a smoothing factor other than 0%, the set value takes effect.

Parameterization

Adaptation factor of the dynamic path response

Via the adaptation factor of the dynamic path response, temporary changes in the path velocity are executed with smaller dynamic response limit values.

The adaptation factor is to be set on a channel-specific basis:

- For traversing motions with acceleration without jerk limitation (**BRISK**) via:
  MD20465 $MC_ADAPT_PATH_DYNAMIC[**0**]
  → The adaptation factor acts on the acceleration.
- For traversing motions with acceleration with jerk limitation (**SOFT**) via:
  MD20465 $MC_ADAPT_PATH_DYNAMIC[**1**]
  → The adaptation factor acts on the jerk.

Axis-specific limit frequencies

The dynamic response limiting should only be active during deceleration and acceleration processes that trigger mechanical vibrations larger than a specific limiting frequency, thus causing excitation of machine resonances.

This limit frequency from which the dynamic response limiting activates, is specified on an axis-specific basis via the machine data:

MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead)

For further information, see Section "Smoothing of the path velocity (Page 223)".

Mode of operation

During processing and via all the axes involved in the path, the controller cyclically establishes the minimum of all the limit frequencies to be the limit frequency (f) for the adaptation of the dynamic response and calculates the relevant time window (t_{adapt}) from this:

$t_{adapt} = 1 / f$

The size of the relevant time window t_{adapt} determines the further behavior:

1. The time needed to change the velocity is less than t_{adapt}:
   The acceleration rates are reduced by a factor > 1 and ≤ the value written in machine data:
   MD20465 ADAPT_PATH_DYNAMIC (adaptation of the path dynamics)
   The reduction in acceleration rate increases the time taken to change the velocity.
   The following cases are different:
   - The acceleration rate is reduced with a value less than MD20465 so that the process lasts for t_{adapt} [s]. The permitted reduction does not need to be fully utilized.
   - The acceleration time is reduced with the value written in MD20465. The process lasts less than t_{adapt} despite the reduced acceleration. The permitted reduction was fully utilized.
2. The time needed to change the velocity is greater than t_{adapt}:
   No dynamic response adaptation is required.

Example

The following example is intended to show the effect of the "adaptation of the dynamic path response" function on traversing motions with acceleration and without jerk limitation (BRISK).

The following parameters are assumed:

MD20465 $MC_ADAPT_PATH_DYNAMIC[0] = 1.5	
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 1.0	
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 Hz	T_{AX1} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 10 Hz	T_{AX2} = 1/10 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20 Hz	T_{AX3} = 1/20 Hz = 50 ms

Note

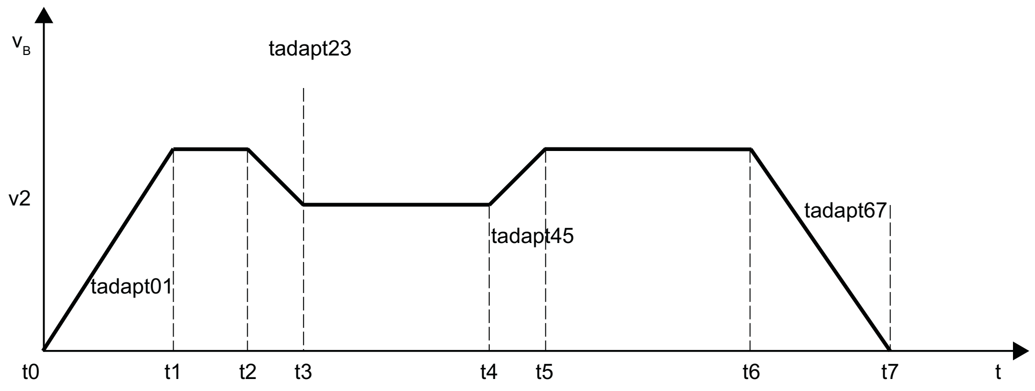
To illustrate the effect of dynamic response adaptation, the value for the smoothing factor (MD20460) is set to "1", whereby the "smoothing of the path velocity" function is practically deactivated.

The path involves the three axes X = AX1, Y = AX2, Z = AX3.

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

If only axes AX1 and/or AX3 are involved in path motions, all deceleration and acceleration processes that would last less than $T_{AX1} = T_{AX3}$ are adapted.

The relevant time window is marked $t_{adapt...}$ in the figures below.



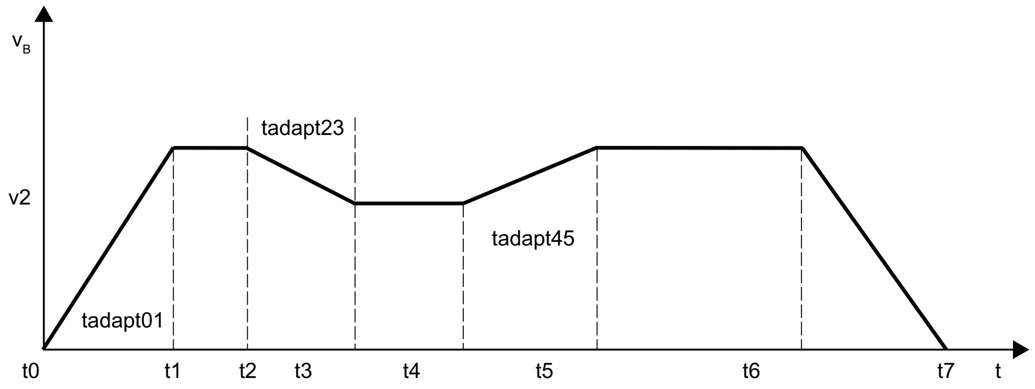


Figure 5-9 Path velocity profile optimized for time without smoothing or dynamic adaptation response

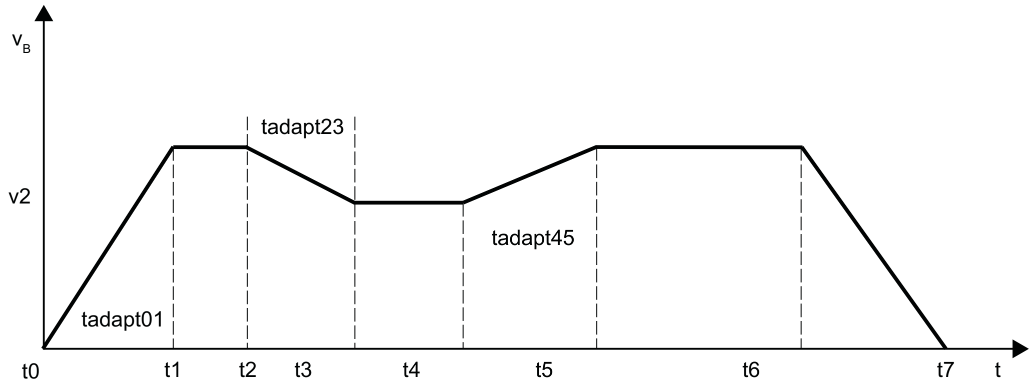


Figure 5-10 Path velocity profile with adaptation of dynamic path response

Intervals t_0 - t_1 and t_2 - t_3:	The acceleration process between t_0 - t_1 and the deceleration process between t_2 - t_3 are extended in terms of time to $t_{adapt01}$ or $t_{adapt23}$ as a result of the acceleration being adapted.
Interval t_4 - t_5:	The acceleration process between t_4 - t_5 is executed with an acceleration reduced by the maximum adaptation factor of 1.5. However, the acceleration process is completed before time $t_{adapt45}$.
Interval t_6 - t_7:	The deceleration process between t_6 - t_7 remains unchanged as it lasts longer than $t_{adapt67}$.

5.4.3 Determination of the dynamic response limiting values

In addition to determining the natural frequency of the path axes for assigning parameters to the axis-specific limit frequencies (MD32440 $MA_LOOKAH_FREQUENCY), the implementation of the "adaptation of the dynamic path response" function also requires dynamic response limits to be determined for velocity, acceleration and jerk.

Procedure

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

1. Deactivate the "adaptation of the dynamic path response" function:
   MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 1
2. Observe the positioning behavior of each path axis at different traversing velocities. When doing so, set the jerk such that the desired positioning tolerance is maintained.

   Note

   The higher the traversing velocity from which the positioning process is started, the higher in general the jerk can be set.

3. Use the maximum permissible jerk determined for the least critical traversing velocity:
   MD32431 $MA_MAX_AX_JERK (maximum jerk)
4. Determine the F_{APD} factor for all of the path axes using:
   F_{APD} = (largest determined jerk) / (smallest determined jerk)

   Note

   The smallest determined jerk is the value for the jerk during the most critical traversing velocity.

5. Enter the largest F_{APD} factor that was determined via all the path axes as the value for the adaptation factor for the path dynamic response:
   MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = F_{APD}

5.4.4 Interaction between the "smoothing of the path velocity" and "adaptation of the path dynamic response" functions

The following examples serve to illustrate the interaction between the "smoothing of the path velocity" and "adaptation of the path dynamic response" functions in continuous-path mode.

Example 1

Acceleration mode: BRISK

The path involves the 3 axes X = AX1, Y = AX2, Z = AX3.

The following parameters are assumed:

MD20465 $MC_ADAPT_PATH_DYNAMIC[0] = 3	
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 80.0	
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20	T_{AX1} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20	T_{AX2} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20	T_{AX3} = 1/20 Hz = 50 ms

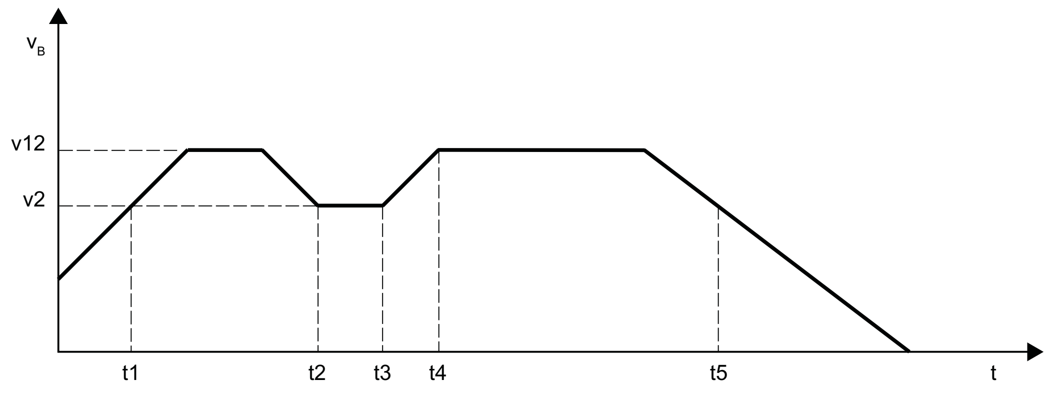


Figure 5-11 Path velocity profile optimized for time without smoothing or dynamic adaptation response

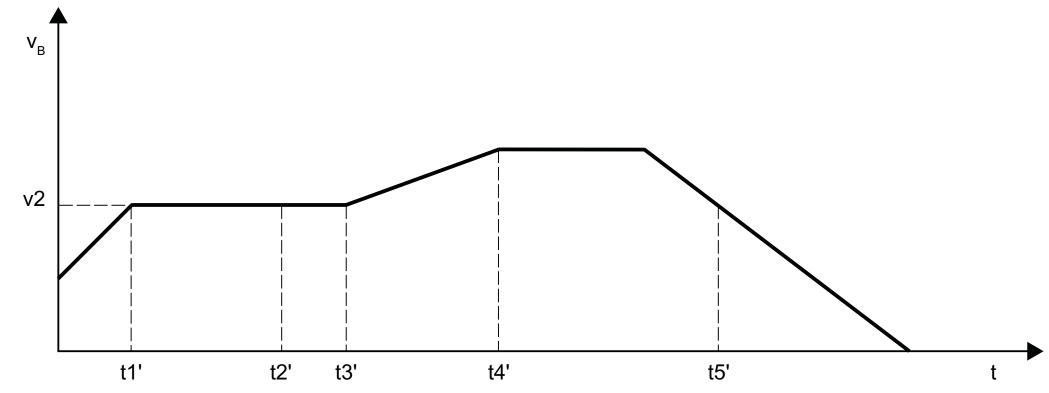


Figure 5-12 Path velocity profile with smoothing of the path velocity and adaptation of dynamic path response

Effects of smoothing on path velocity:

Interval t_1 - t_2:	The acceleration and deceleration process between t_1 and t_2 does not take place because the lengthening of the machining time without the acceleration process to v_{12} is less than the resulting time if a smoothing factor of 80 % is applied.
Interval t_3 - t_5:	The acceleration and braking profile between t_3 and t_5 does not fulfill this condition or takes longer than the parameterized smoothing time T_{Axn} = 2/20 Hz = 100 ms.

Effects of the dynamic response adaptation:

Interval t_3 - t_4:	The acceleration process between t_3 and t_4 is shorter than MIN(T_{AXn}) = 1/20 Hz = 50 ms and is, therefore, executed with an acceleration reduced by an adaptation factor of 3.
Interval up to t_1:	The acceleration up to t_1 left over after path smoothing is stretched to the time period up to t_1' by the dynamic response adaptation.

Note

The example shows that those acceleration or deceleration processes that are not eliminated by the smoothing of the path velocity can be subsequently optimized by adapting the dynamic path response. For this reason, both functions should always be activated, if possible.

Example 2

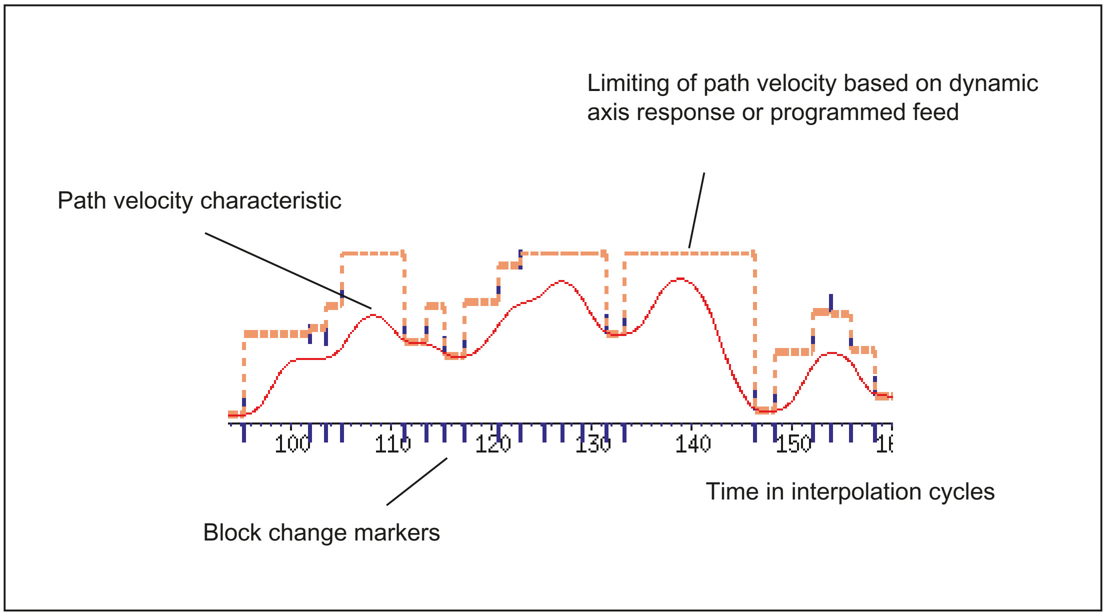
Acceleration mode: SOFT

The path involves the 3 axes X = AX1, Y = AX2, Z = AX3.

The following parameters are assumed:

MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 1	
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 0.0	
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 10	T_{AX1} = 1/20 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 10	T_{AX2} = 1/20 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20	T_{AX3} = 1/20 Hz = 50 ms

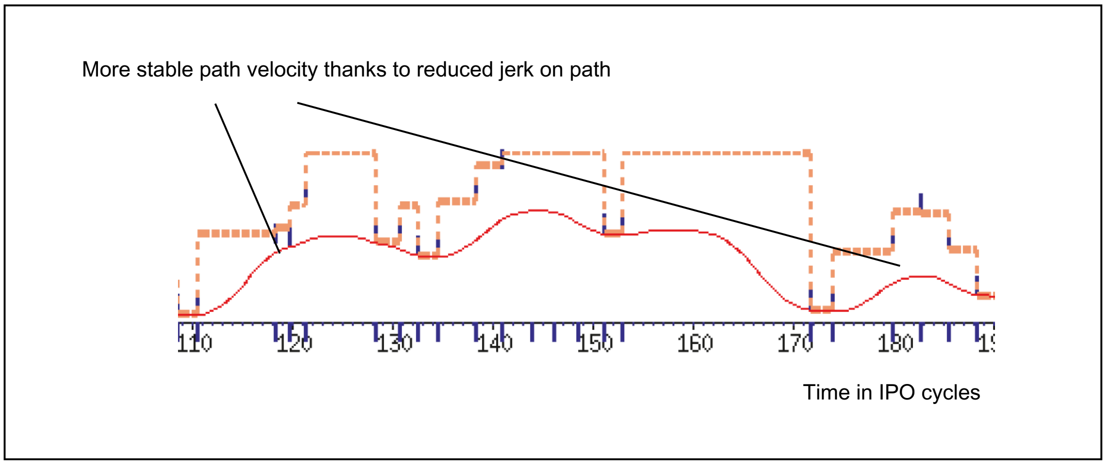
This leads to a path velocity profile which is optimized in terms of time without smoothing the path velocity or adapting the dynamic path response:



The parameter assignment is changed as follows:

MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 4
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 1.0

This results in a path velocity profile with adaptation of the dynamic path response and with minimum, and thus virtually deactivated, smoothing of the path velocity:

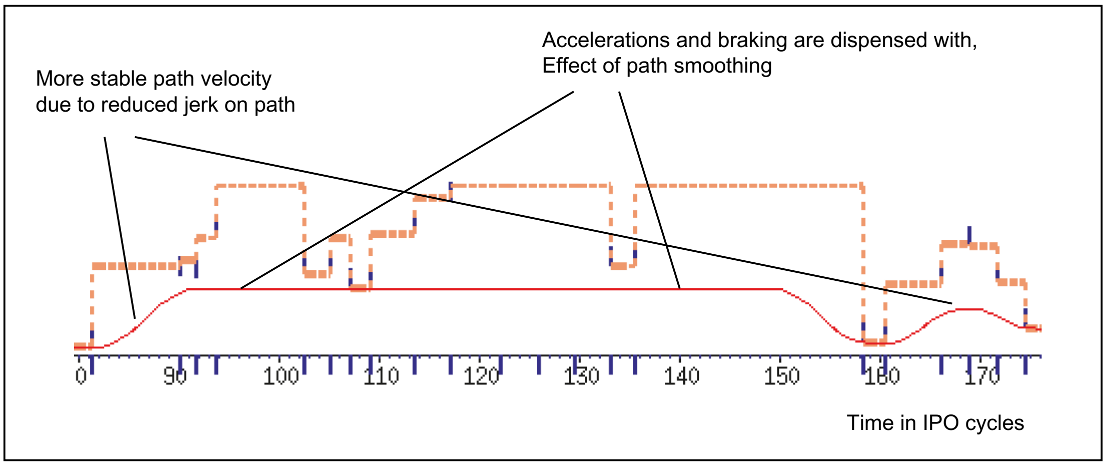


The smoothing factor is set to 0% instead of 1% (in accordance with the default!):

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 0.0

A smoothing factor of 100% comes into effect with this parameter assignment.

This gives rise to a path velocity profile with smoothing of the path velocity and adaptation of dynamic path response:



5.4.5 Dynamic response mode for path interpolation

Function

Technology-specific, dynamic response settings can be saved in machine data and can be activated in the part program via the commands from G group 59 (dynamic response mode for path interpolation).

Command	Activates the dynamic response settings for:
`DYNNORM`	Standard dynamic response settings
`DYNPOS`	Positioning mode, tapping
`DYNROUGH`	Roughing
`DYNSEMIFIN`	Finishing
`DYNFINISH`	Smooth finishing

Note

The dynamic response of the **path axes** alone is determined by the commands from G group 59 (dynamic response mode for path interpolation). They have no effect on:

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

The standard dynamic response setting (DYNNORM) always takes effect for these axis motions.

Application

By switching the dynamic response settings, roughing can be optimized in terms of time and smoothing can be optimized in terms of the surface, for example.

Parameterization

Axis-specific dynamic response settings:

- MD32300 $MA_MAX_AX_ACCEL[<n>] (axis acceleration)
- MD32310 $MA_MAX_ACCEL_OVL_FACTOR[<n>] (overload factor for axial jumps in velocity)
- MD32431 $MA_MAX_AX_JERK[<n>] (maximum axial jerk for path motion)
- MD32432 $MA_PATH_TRANS_JERK_LIM[<n>] (maximum axial jerk at the block transition in continuous-path mode)
- MD32433 $MA_SOFT_ACCEL_FACTOR[<n>] (scaling of the acceleration limitation with SOFT)

Channel-specific dynamic response settings:

- MD20600 $MC_MAX_PATH_JERK[<n>] (path-related maximum jerk)
- MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL[<n>] (influence of path curvature on path acceleration)
- MD20603 $MC_CURV_EFFECT_ON_PATH_JERK[<n>] (influence of path curvature on path jerk)

where index <n> =0	for DYNNORM
1	for DYNPOS

2	for DYNROUGH
3	for DYNSEMIFIN
4	for DYNFINISH

Note

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

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

Suppressing G commands

It is recommended that G commands from G group 59 (dynamic response mode for path interpolation) which are not intended for use should be suppressed via the following machine data:

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[<n>] (list of reconfigured NC commands)

If a suppressed G command is used, an alarm is displayed. This prevents machine data that has not been parameterized taking effect.

Example

The G commands `DYNPOS` and `DYNSEMIFIN` can be suppressed with the following settings:

- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0]="DYNPOS"
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1]=" "
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[2]="DYNSEMIFIN"
- MD10712 $MN_ NC_USER_CODE_CONF_NAME_TAB[3]=" "

References

You can find detailed information about programming the G commands from G group 59 (dynamic response mode for path interpolation) in:
References:
Programming Manual, Job Planning; Section: Path traversing behavior

5.4.6 Free-form surface mode: Basic functions

Introduction

In applications in tool and mold making, it is important that the surfaces on the workpiece are as uniform as possible. This requirement is generally more important than the precision of the surface of the workpiece.

Workpiece surfaces which lack uniformity can be attributable to the following causes, for example:

- The part program for manufacturing the workpiece contains a non-uniform geometry. This, most notably, affects the profile of the curvature and torsion.

  Note

  The curvature k of a contour is the inverse of radius r of the adapted circle in a contour point (k = 1/r). The torsion is the change in curvature (first derivative).

  As a result of the lack of uniformity in geometry, the machine's dynamic response limits are reached during processing of the part program, and needless deceleration and acceleration processes occur. Depending on the extent of the effective over-travel of the axes, this leads to different deviations in contours.
- Needless deceleration and acceleration processes can trigger machine vibrations which result in unwanted marks on the workpiece.

There are various options available for eliminating these causes:

- Use a CAD/CAM system
  The part programs generated by CAD/CAM systems contain a very uniform curvature and torsion profile, preventing needless reductions in path velocity.
- Specify the maximum path velocity in such a way that unwanted geometric fluctuations in the curvature and torsion profile have no effect.

Function

"Free-form surface mode: Basic functions" can be used to make the definition of path velocity limits insensitive to small geometric fluctuations in curvature and torsion without exceeding the machine's dynamic limits in terms of the acceleration and jerk of the axes.

This has the following advantages:

- Greater uniformity in the profile of the path velocity
- Greater uniformity in the surface of the workpiece
- Reduction in the processing time (if the dynamic response of the machine permits it)

Applications

The function is used to process workpieces which primarily comprise free-form surfaces.

Requirements

The function can only be activated if the requisite memory capacity is reserved during memory configuration:

MD28610 $MC_MM_PREPDYN_BLOCKS = **10**

The value entered prescribes the number of blocks which have to be taken into consideration in the determination of the path velocity (velocity preparation).
A sensible value is "10".

If MD28610 has a value of "0", only the motions of the axes in a particular block are taken into consideration when determining the maximum velocity of the path for that block. If the geometry of neighboring blocks is also taken into consideration when determining the velocity of the path (value > 0), a more uniform profile in path velocity is achieved.

Activation/deactivation

The function can be switched on or off independently for every dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 235)"):

MD20606 $MC_PREPDYN_SMOOTHING_ON[<n>] = <value>

Index <n>	Dynamic response mode	<value>	Free-form surface mode: Basic functions
0	Standard dynamic response settings (DYNNORM)	0	Off
		1	On
1	Positioning mode, tapping (DYNPOS)	0	Off
		1	On
2	Roughing (DYNROUGH)	0	Off
		1	On
3	Finishing (DYNSEMIFIN)	0	Off
		1	On
4	Smooth finishing (DYNFINISH)	0	Off
		1	On

Note

Due to the additional storage requirements, the function should only be activated in the relevant processing channels.

Parameterization

Change in the contour sampling factor

The secant error which occurs during the interpolation of curved contours is dependent on the following factors:

- Curvature
- Interpolator clock cycle (display in the MD10071 $MN_IPO_CYCLE_TIME)
- Velocity with which the relevant contour is traversed

The maximum possible secant error is defined for each axis in the machine data:

MD33100 $MA_COMPRESS_POS_TOL (maximum tolerance with compression)

If the set interpolator clock cycle is not sufficiently small, the max. path velocity may be reduced in the case of contours with greater curvature. This is necessary for ensuring that the surface of the workpiece is also produced with an adequate degree of precision in this case.

By changing the contour sampling factor, the time interval with which a curved contour is sampled in the interpolator (contour sampling time) can be set differently than the interpolator clock cycle. A contour sampling time which is shorter than the interpolator clock cycle can prevent a reduction in path velocity in the case of contours with greater curvature.

The contour sampling factor is set with the machine data:

MD10682 $MN_CONTOUR_SAMPLING_FACTOR

The effective contour sampling time is calculated as follows:

$T_s = f * T_1$

where:	T_s	= Effective contour sampling time
	T_1	= Interpolator clock cycle
	f	= Contour sampling factor (value from MD10682)

The default contour sampling factor is "1", i.e. the contour sampling time equals the interpolator clock cycle.

The contour sampling factor can be both greater or less than "1".

If a value less than "1" is set, monitoring of contour sampling precision is disabled.

The set sampling time must not be below the configured minimum contour sampling time:

MD10680 $MN_MIN_CONTOUR_SAMPLING_TIME

Note

MD10680 is specifically set for every controller model and cannot be changed.

Programming

Depending on the setting in machine data MD20606 $MC_PREPDYN_SMOOTHING_ON, "Free-form surface mode: Basic functions" can be switched on and off in the part program by changing the active dynamic response mode.

Example:

By assigning the parameters MD20606 $MC_PREPDYN_SMOOTHING_ON[2-4] = 1 and MD20606 $MC_PREPDYN_SMOOTHING_ON[0-1] = 0, the function can be switched on via the commands `DYNROUGH`, `DYNSEMIFIN`, and `DYNFINISH` and switched off via the commands `DYNNORM` and `DYNPOS`.

See also

Rounding of tangential block transitions (G645) (Page 212)

Velocity-dependent jerk adaptation (axis-specific) (Page 292)

Free-form surface mode: Extension function (Page 219)

5.5 Compressor functions

5.5.1 Compression of linear, circular and rapid traverse blocks

5.5.1.1 Function

CAD/CAM systems generate a large number of linear and circular blocks, some of which with very short path lengths, to describe complex contours. The maximum possible path velocity is frequently limited by the block change time. As of a certain path velocity, not enough new traversing blocks can be prepared in the preprocessing and switched to the main run.

Compressing linear blocks

Compressor functions replace consecutive linear blocks with polynomial blocks having the longest possible path lengths while maintaining a parameterizable contour precision. This has the following advantages:

- Reduction of the number of traversing blocks
- Increasing the path velocity
- Increasing the surface quality
- Continuous block transitions

The compressor functions with their most important properties/attributes that are available are listed in the following table:

Compressors	Function	Continuity at block transitions	Notes about use
COMPON	COMPON creates **from up to 10** consecutive linear blocks, type "G01 X... Y... Z... F..." a polynomial block.	Continuous velocity	
COMPCURV	the same as COMPON	Continuous velocity and acceleration	

Compressors	Function	Continuity at block transitions	Notes about use
COMPCAD	COMPCAD can create a polynomial block from **any number** of consecutive linear blocks.	Continuous velocity and acceleration	COMPCAD is very processor and memory-intensive. It is therefore recommended that COMPCAD only be used where measures to improve surface quality were not successful in the CAD/CAM program.
COMPSURF	the same as COMPCAD	Continuous velocity and acceleration	Using COMPSURF, even better results can be obtained than with COMPCAD. However, just the same as for COMPCAD, COMPSURF takes up a lot of computation time and memory space. By using COMPSURF, for example for inclined line-by-line finishing programs, poor data quality and/or irregular point distribution in the CAD/CAM program, significantly improved workpiece surfaces can be achieved. In addition, COMPSURF facilitates a direction-independent identical smoothing of the milling paths that can be deactivated; this significantly increases the surface quality for bidirectional milling tools.

If commands that are not traverse commands (e.g. auxiliary function output), are programmed in and between the traversing blocks to be compressed, compression is interrupted.

The maximum tolerable deviation of the calculated path to the programmed positions can be specified in machine data for all compressor functions (see "Parameterization (Page 243)"). Unlike COMPON and COMPCURV, the parameterized tolerances are not used in different directions in neighboring paths with COMPCAD and COMPSURF.

Compression of circular and rapid traverse blocks

In addition to the compression of linear blocks, all compressor functions also allow rapid traverse blocks (G0 blocks) to be compressed. On the other hand, circular blocks can only be compressed using the COMPCAD compressor function.

The compression of circular and/or rapid traverse blocks is set for specific channels using the **hundreds position** in machine data:

MD20482 $MC_COMPRESSOR_MODE = <value>

Value	Meaning
0xx	Circular blocks and G0 blocks are **not** compressed. This is compatible with earlier SW versions.
1xx	Circular blocks are linearized and compressed by COMPCAD. Advantage: The compressor function operates more precisely and therefore creates generally better surfaces. Disadvantage: The compressor function is more sensitive to defects in the NC programs. For reasons of compatibility it might therefore be necessary to keep the setting 0xx.

Value	Meaning
2xx	G0 blocks are compressed - it is possible that a different tolerance becomes effective (see "Tolerances for rapid traverse movements (Page 256)"). Advantage: As a result that the tolerance has been set higher and the compression of G0 infeed motion, these can be more quickly and more fluidly executed.
3xx	Combination of the two previous options: Circular blocks as well as G0 blocks are compressed.

Activation / deactivation

Compressor functions COMPON, COMPCURV, COMPCAD and COMPSURF are activated/ deactivated using the G commands of G group 30 (see "Programming (Page 245)").

5.5.1.2 Commissioning

Parameterization

Axis-specific machine data

Number	Identifier $MA_	Meaning
MD33100	COMPRESS_POS_TOL	Maximum permissible axis-specific path deviation with compression

Channel-specific machine data

Number	Identifier $MC_	Meaning
MD20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of an NC block that can be compressed
MD20171	SURF_BLOCK_PATH_LIMIT	Maximum traversing length of a compressible NC block for compression with COMPSURF
MD20172	COMPRESS_VELO_TOL	Maximum permissible deviation from feedrate for compression
MD20173	SURF_VELO_TOL	Maximum permissible deviation from feedrate for compression with COMPSURF
MD20482	COMPRESSOR_MODE	Principle of operation of the compressor
MD20484	COMPRESSOR_PERFORMANCE	Compressor performance
MD20485	COMPRESS_SMOOTH_FACTOR	Smoothing factor for compression with COMPCAD for the particular dynamic response mode
MD20486	COMPRESS_SPLINE_DEGREE	Spline degree for compression with COMPCAD for the particular dynamic response mode
MD20487	COMPRESS_SMOOTH_FACTOR_2	Smoothing factor for rotary axes for compression with COMPCAD for the particular dynamic response mode

Number	Identifier $MC_	Meaning
MD28071	MM_NUM_SURF_LEVELS	Dimensioning the COMPSURF function (DRAM)
MD28072	MM_MAXNUM_SURF_GROUPS	Dimensioning the COMPSURF function regarding axis groups (DRAM)

Channel-specific setting data

Number	Identifier $SC_	Meaning
SD42470	CRIT_SPLINE_ANGLE	Corner angle limit for compression with COMP-CAD
SD42471	MIN_CURV_RADIUS	Minimum radius of curvature for compression with COMPCAD
SD42472	MIN_SURF_RADIUS	Minimum radius of curvature for compression with COMPSURF
SD42473	ACTNUM_SURF_GROUPS	Number of axis groups for COMPSURF
SD42475	COMPRESS_CONTUR_TOL	Maximum permissible contour deviation with com-pression
SD42476	COMPRESS_ORI_TOL	Maximum deviation of the tool orientation for com-pression **Note:** Only for active orientation transformation!
SD42477	COMPRESS_ORI_ROT_TOL	Maximum deviation of the tool rotation for com-pression **Note:** Only for 6-axis machines with rotatable tool!

Note

Corner limit angle and compressor function COMPCAD

The corner limit angle for the compressor function COMPCAD set via the setting data SD42470 $SC_CRIT_SPLINE_ANGLE is only used as an approximate measure for corner detection. By evaluating the plausibility, the compressor can also identify flatter block transitions as corners and larger angles as outliers.

Recommended settings for tool and mold making with Advanced Surface / Top Surface

Compressor functions are very important in milling of free-form surfaces in tool and mold making. If they are used as part of the option "Advanced Surface" or "Top Surface" for which a license is required, please observe the setting recommendations!

Special test programs are available in the SIOS portal for checking the set machine and setting data:

- Test programs for Advanced Surface (https://support.industry.siemens.com/cs/ww/en/view/78956392)
- Test programs for Top Surface (https://support.industry.siemens.com/cs/ww/en/view/109738423)

5.5.1.3 Programming

Activating/deactivating NC block compression (COMPON, COMPCURV, COMPCAD, COMPSURF, COMPOF)

The functions to compress linear blocks (and dependent on the parameterization, also circular and/or rapid traverse blocks) are activated/deactivated using G commands of G group 30. The commands are modal.

Syntax

```
COMPON / COMPCURV / COMPCAD / COMPSURF
...
COMPOF
```

Meaning

`COMPON`:	Activating the compressor function COMPON
`COMPCURV`:	Activating the compressor function COMPCURV
`COMPCAD`:	Activating the compressor function COMPCAD
`COMPSURF`:	Activating the compressor function COMPSURF
`COMPOF` :	Deactivating the currently active compressor function

Note

The rounding function G642 and jerk limitation SOFT further improve the surface quality. These commands must be written at the beginning of the program.

Example: COMPCAD

```
Program code                  Comment
N10 G00 X30 Y6 Z40
N20 G1 F10000 G642            ; Activation: Rounding function G642
N30 SOFT                      ; Activation: Jerk limitation SOFT
N40 COMPCAD                   ; Activation: Compressor function COMPCAD
N50 STOPFIFO
N24050 Z32.499                ; 1st traversing block
N24051 X41.365 Z32.500        ; 2nd traversing block
...
N99999 X... Z...              ; last traversing block
COMPOF                        ; compressor function off.
...
```

5.5.1.4 Supplementary conditions

Orientation transformation (TRAORI)

When orientation transformation is active, the compressor functions can also compress motion blocks for tool orientation and tool rotation.

Reference:
Function Manual, Special Functions; Chapter "F2: Multi-axis transformation" > "Orientation" > "Compression of the orientation"

Block search with calculation

If the target block for block search **type 2** or **type 4** (block search **with calculation** to ...) is in a program section in which a compressor function is active, positions are approached on the path calculated by the compressor on repositioning. These positions must precisely match the positions on the path programmed in the part program.

Part program blocks, which are replaced by compression, cannot be found as target block in the block search. Alarm 15370 "Search target not found" is output.

5.5.2 Compression of short spline blocks

Function

When spline blocks are generated to describe complex contours using CAD/CAM systems, spline blocks with very short path lengths occur between spline blocks with long path lengths. These force the control to significantly reduce the path velocity. The functions for the compression of short spline blocks generate new spline blocks with the longest possible path lengths.

Availability

System	Availability
SINUMERIK 840D sl	Standard (basic scope)
SINUMERIK 828D	Option

Commissioning

Activation

The compression of short spline blocks can be activated for the following spline types:

- BSPLINE
- BSPLINE/ORICURVE
- CSPLINE

It is activated in the channel-specific machine data:

MD20488 $MC_SPLINE_MODE, bit <n> = <value>

Bit	<value>	Spline type	Compressing short spline blocks
0	0	BSPLINE	Not active
	1	BSPLINE	Active
1	0	BSPLINE/ORICURVE	Not active
	1	BSPLINE/ORICURVE	Active
2	0	CSPLINE	Not active
	1	CSPLINE	Active

Supplementary conditions

- If commands that are not traverse commands, e.g. auxiliary function outputs, are programmed in and between the traversing blocks to be compressed, the spline blocks cannot be combined.
- The maximum number of blocks that can be combined into a program section in succession, depends on the parameterized size of the block memory available in the block preparation. MD28070 $MC_MM_NUM_BLOCKS_IN_PREP (number of blocks for block preparation)

Example

To achieve a higher path velocity when executing the traversing blocks, compression for short spline blocks is activated for BSPLINE interpolation:

MD20488 $MC_SPLINE_MODE, Bit 0 = 1

```
Program code                              Comment
N10 G1 G64 X0 Y0 Z0 F1000                 ; Initial setting
N20 G91 F10000 BSPLINE                    ; Activation: B spline
N30 X0.001 Y0.001 Z0.001                  ; From here: Combine short spline blocks
N40 X0.001 Y0.001 Z0.001
...
```

5.6 Contour/Orientation tolerance

5.6.1 Commissioning

5.6.1.1 Parameter assignment

Machine data

Contour tolerance / orientation tolerance

MD33100 $MA_COMPRESS_POS_TOL[<axis>] = <value> (maximum tolerance with compression)

Via the axis-specific machine data, the maximum permitted contour deviation (contour tolerance) or angle deviation of the tool orientation (orientation tolerance) of each axis is set. The machine data is effective with the following functions:

- Rounding: G642, G643, G644, G645
- Compressor: COMPON, COMPCURV, COMPCAD, COMPSURF
- The higher the value, the more short blocks can be compressed into a long block.

The machine data is not effective for smoothing function G641. For G641, the path distance to the block transition, which can be programmed with ADIS or ADISPOS, is effective.

Smoothing mode

MD20480 $MC_SMOOTHING_MODE (rounding behavior with G64x)

Compressor mode

MD20482 $MC_COMPRESSOR_MODE (mode of compression)

Smoothing G645

MD33120 $MA_PATH_TRANS_POS_TOL (maximum contour deviation for smoothing G645)

Effective for rounding tangential, but not continuously curved block transitions (e.g. circle - straight line)

Setting data

Channel-specific contour tolerance

SD42465 $SC_SMOOTH_CONTUR_TOL (maximum contour deviation)

Channel-specific orientation tolerance

SD42466 $SC_SMOOTH_ORI_TOL (maximum angular deviation of the tool orientation)

Channel-specific orientation tolerance for smoothing with OST

SD42676 $SC_ORI_SMOOTH_TOL (tolerance for smoothing with orientation on rounding)

Channel-specific orientation tolerance for smoothing orientation using ORISON

SD42678 $SC_ORISON_TOL (tolerance for smoothing the orientation)

5.6.2 Programming

5.6.2.1 Programming contour/orientation tolerance (CTOL, OTOL, ATOL)

Addresses CTOL, OTOL and ATOL can be used to adapt the machining tolerances - parameterized using machine and setting data - for compressor functions, smoothing and orientation smoothing in the part program.

The programmed tolerance values are valid until they are reprogrammed or deleted by assigning a negative value. Further, they are deleted at the end of the program or a reset The parameterized tolerance values become effective again after deletion.

Syntax

```
CTOL=<Value>
OTOL=<Value>
ATOL[<Axis>]=<Value>
```

Meaning

`CTOL`:	Address to program the **contour tolerance**			
	Applications:	• All compressor functions • All rounding types except G641 and G644		
	Preprocessing stop:	No		
	Effective:	Modal		
	`<Value>`:	The value for the contour tolerance is specified as a length.		
		Type:	REAL	
		Unit:	inch/mm (dependent on the current dimensions setting)	
		Value range:	≥ 0:	Tolerance value
			< 0:	The programmed tolerance value is deleted ⇒ The tolerance value parameterized in the machine or setting data becomes effective again.

`OTOL`:	Address to program the **orientation tolerance**			
	Applications:	• All compressor functions • ORISON orientation smoothing • All smoothing types except G641, G644 and OSD		
	Preprocessing stop:	No		
	Effective:	Modal		
	`<Value>`:	The value for the orientation tolerance is specified as an angle.		
		Type:	REAL	
		Unit:	degrees	
		Value range:	≥ 0:	Tolerance value
			< 0:	The programmed tolerance value is deleted ⇒ The tolerance value parameterized in the machine or setting data becomes effective again.
`ATOL`:	Address for programming an **axis-specific tolerance**			
	Applications:	• All compressor functions • ORISON orientation smoothing • All smoothing types except G641, G644 and OSD		
	Preprocessing stop:	No		
	Effective:	Modal		
	`<Axis>`:	Name of the channel axis to which the programmed tolerance will apply		
	`<Value>`:	The value for the axis tolerance will be specified as a length or an angle dependent on the axis type (linear or rotary axis).		
		Type:	REAL	
		Unit:	For linear axes:	inch/mm (dependent on the current dimensions setting)
			For rotary axes:	degrees
		Value range:	≥ 0:	Tolerance value
			< 0:	The programmed tolerance value is deleted ⇒ The tolerance value parameterized in the machine or setting data becomes effective again.

Note

The channel-specific tolerance values programmed with CTOL and OTOL have higher priority than the axis-specific tolerance values programmed with ATOL.

Note

Scaling frames

Scaling frames affect programmed tolerances in the same way as axis positions; in other words, the relative tolerance remains the same.

Example

Program code	Comment
COMPCAD G645 G1 F10000	; Activate COMPCAD compressor function.
X... Y... Z...	; The machine and setting data is applied here.
X... Y... Z...	
X... Y... Z...	
CTOL=0.02	; A contour tolerance of 0.02 mm is applied starting from here.
X... Y... Z...	
X... Y... Z...	
X... Y... Z...	
ASCALE X0.25 Y0.25 Z0.25	; A contour tolerance of 0.005 mm is applied starting from here.
X... Y... Z...	
X... Y... Z...	
X... Y... Z...	
CTOL=-1	; The machine and setting data is applied again starting from here.
X... Y... Z...	
X... Y... Z...	
X... Y... Z...	

5.6.2.2 Programming contour/orientation tolerance (CTOL, OTOL, ATOL) Additional information

System variables

Reading with preprocessing stop

Using the following system variables, the currently active tolerances can be read in the part program and synchronized action:

- $AC_CTOL
  Channel-specific contour tolerance effective when the actual main run block was preprocessed.
  If no contour tolerance is effective, $AC_CTOL will return the root of the sum of the squares of the tolerances of the geometry axes.
- $AC_OTOL
  Channel-specific orientation tolerance effective when the actual main run block was preprocessed.
  If no orientation tolerance is effective, $AC_OTOL will return the root of the sum of the squares of the tolerances of the orientation axes during active orientation transformation. Otherwise, it will return the value "-1."
- $AA_ATOL[<axis>]
  Axis-specific contour tolerance effective when the actual main run block was preprocessed. If no contour tolerance is active, $AA_ATOL[<geometry axis>] returns the contour tolerance divided by the root of the number of geometry axes.
  If an orientation tolerance and an orientation transformation are active $AA_ATOL[<orientation axis>] will return the orientation tolerance divided by the root of the number of orientation axes.

Note

If now tolerance values have been programmed, the $A variables are not differentiated enough to distinguish the tolerance of the individual functions.

Circumstances like this can occur if the machine data and the setting data set different tolerances for compressor functions, smoothing and orientation smoothing. The system variables then return the greatest value occurring with the functions that are currently active. For example, if a compressor function is active with an orientation tolerance of 0.1° and ORISON orientation smoothing with 1°, the $AC_OTOL variable will return the value "1." If orientation smoothing is deactivated, $AC_OTOL returns a value value "0.1."

Reading without preprocessing stop

Using the following system variables, the currently active tolerances can be read in the part program:

- $P_CTOL
  Currently active channel-specific contour tolerance.
- $P_OTOL
  Currently active channel-specific orientation tolerance.
- $PA_ATOL
  Currently active axis-specific contour tolerance.

Supplementary conditions

The tolerances programmed with CTOL, OTOL and ATOL also affect functions that indirectly depend on these tolerances:

- Limiting the chord error in the setpoint value calculation
- The basic functions of the free-form surface mode

The following smoothing functions are **not** affected by the programming of CTOL, OTOL and ATOL:

- Smoothing the orientation with OSD
  OSD does not use a tolerance, it uses a distance from the block transition.
- Smoothing with G644
  G644 is not used for smoothing, it is used for optimizing tool changes and other motion not involving machining.
- Smoothing with G645
  G645 virtually always behaves like G642 and, thus, uses the programmed tolerances. The tolerance value from machine data MD33120 $MA_PATH_TRANS_POS_TOL is only used in uniformly tangential block transitions with a jump in curvature, e.g. a tangential circle/straight line transition. The rounding path at these points may also be located outside the programmed contour, where many applications are less tolerant. Furthermore, it generally takes a small, fixed tolerance to compensate for the sort of changes in curvature which need not concern the NC programmer.

5.7 Rapid traverse movements

5.7.1 Function

5.7.1.1 Rapid traverse

During rapid traversing, programmed tool movements are carried out at the fastest possible traversing velocity.

The velocity of the rapid traversing is defined separately for each axis (see "Parameter assignment (Page 257)").

Application

Rapid traversing movements are used for the following tasks, for example:

- Quickly positioning of the tool
- Passing around the workpiece
- Approaching tool change points
- Retracting the tool

Note

Rapid traversing movements are not suitable for workpiece machining!

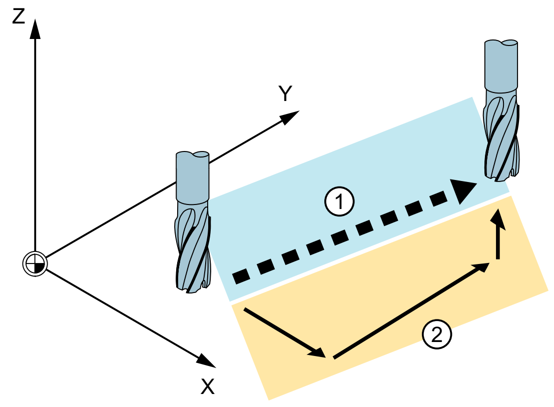
Activation

Rapid traverse is activated by programming of G0 in the part program (see "Programming (Page 257)").

5.7.1.2 Interpolation response of path axes for rapid traversing movements

Liner/non-linear interpolation

Path axes can be traversed in linear or non-linear interpolation mode in rapid traverse movement.



①	Path for rapid traverse with linear interpolation
②	Single axis movements for rapid traverse with non-linear interpolation

Linear interpolation

Properties:

- The path axes are interpolated together.
- The tool movement programmed with G0 is executed at the highest possible velocity (rapid traverse).
- The rapid traverse velocity is defined separately for each axis.
- If the rapid traverse movement is executed simultaneously on several axes, the rapid traverse speed is determined by the axis which requires the most time for its section of the path.

Linear interpolation is always performed in the following cases:

- For a G command combination with G0 that does **not** allow positioning axis motion, e.g.: G40, G41, G42, G96, G961 and MD20750 $MC_ALLOW_G0_IN_G96 == FALSE
- With a combination of G0 and G64.
- When a compressor or transformation is active.
- In point-to-point (PTP) travel mode.
- When a contour handwheel is selected (FD = 0).
- In case of an active frame with rotation of geometry axes.
- If nibbling is active for geometry axes.

Non-linear interpolation

Properties:

- Each path axis interpolates as a single axis (positioning axis) independently of the other axes at the axis-specific rapid traverse velocity.
- The channel-specific "Delete distance-to-go" command via the PLC and synchronized action is applied to all positioning axes that were programmed as path axes.

In non-linear interpolation, with reference to the axis-specific jerk, one of the following two settings applies alternatively:

- Positioning axis commands BRISKA, SOFTA, DRIVEA
- Machine data:
  - MD32420 $MA_JOG_AND_POS_JERK_ENABLE
  - MD32430 $MA_JOG_AND_POS_MAX_JERK

The existing system variables which refer to the distance-to-go ($AC_PATH, $AC_PLTBB and $AC_PLTEB) are supported.

⚠ CAUTION

Risk of collision

Since the tool movement for non-linear interpolation can differ from the tool movement for linear interpolation, synchronous actions relative to the coordinates of the path movement may not become active.

Selection of interpolation type

The type of interpolation which is to be in effect for rapid traverse movements is preset by machine data (see "Parameter assignment (Page 257)").

Independently of the default setting, the desired interpolation response can also be set in the part program (see "Switch on/off linear interpolation for rapid traverse movements (RTLION, RTLIOF) (Page 259)").

5.7.1.3 Tolerances for rapid traverse movements

The tolerances for rapid traverse movements can be set differently from the workpiece machining tolerances by specifying a G0 tolerance factor.

Advantage

Larger tolerances for rapid traverse movements (G0 tolerance factor > 1) allow G0 blocks to be traversed through faster.

Preconditions

The G0 tolerance factor is only effective, if the following conditions are fulfilled:

- One of the following functions is active:
  - Compressor functions COMPON, COMPCURV, COMPCAD or COMPSURF
  - Smoothing function G642 or G645
  - Orientation smoothing OST
  - Orientation smoothing ORISON
  - Smoothing for path-relevant orientation ORIPATH
- Several (≥ 2) consecutive G0 blocks in the part program.
  For a single G0 block, the G0 tolerance factor is not effective, as at the transition from a non G0 motion to a G0 motion (and vice versa), the "**lower tolerance**" always applies (workpiece machining tolerance)!

Definition of the G0 tolerance factor

The tolerance factor for rapid traverse movements is preset via machine data to be channel-specific (see "Parameter assignment (Page 257)").

The preset tolerance factor can be temporarily adapted by programming in the part program (see "Adapt tolerance factor for rapid traverse movements (STOLF) (Page 260)").

5.7.1.4 Rapid traverse override

With the rapid traverse override switch on the machine control panel, the operator can reduce the rapid traverse velocity by percentages on-site and with immediate effect.

An active rapid traverse correction has an effect on all path axes that are traversed in rapid traverse mode with linear or non-linear interpolation and are assigned to the current channel.

For further information, see "Feedrate override via machine control panel (Page 1423)".

5.7.2 Commissioning

5.7.2.1 Parameter assignment

Rapid traverse velocity

The rapid traverse velocity corresponds to the maximum permitted axis velocity, which is defined for each axis via the following machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity) will still apply even after the coupling is activated

Interpolation response for rapid traverse movements

The interpolation response for rapid traverse movements is channel-specifically preset for via the machine data:

MD20730 $MC_G0_LINEAR_MODE = <Value> (interpolation behavior with G0)

<value>	Meaning
0	In the rapid traversing mode (G0) the **non-linear** interpolation is active. Path axes are traversed as positioning axes.
1	In the rapid traversing mode (G0) the **linear** interpolation is active. The path axes are interpolated together.

G0 tolerance factor

The tolerance factor for rapid traverse movements is channel-specifically set via the machine data:

MD20560 $MC_G0_TOLERANCE_FACTOR (tolerance factor for G0)

The tolerance factor can be both greater or less than 1.0. If the factor is equal to 1.0 (default value), then the same tolerances are active for rapid traverse movements as for non-rapid traverse movements. Normally, the tolerance factor is set to > 1.0.

5.7.3 Programming

5.7.3.1 Activating rapid traverse (G0)

The traversing of the path axes at rapid traversing velocity is activated with the G command G0.

Syntax

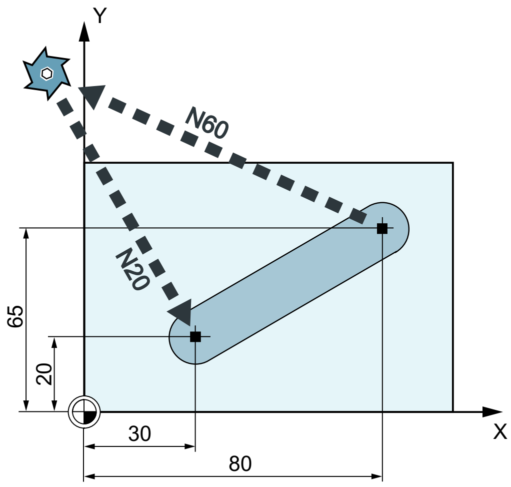
```
G0 X… Y… Z…
G0 RP=… AP=…
```

Meaning

`G0:`	Traversing the axis with rapid traverse velocity	
	Effective:	Modal
`X... Y... Z...:`	Specifying the end point in Cartesian coordinates	
`RP=... AP=... :`	Specifying the end point in polar coordinates	

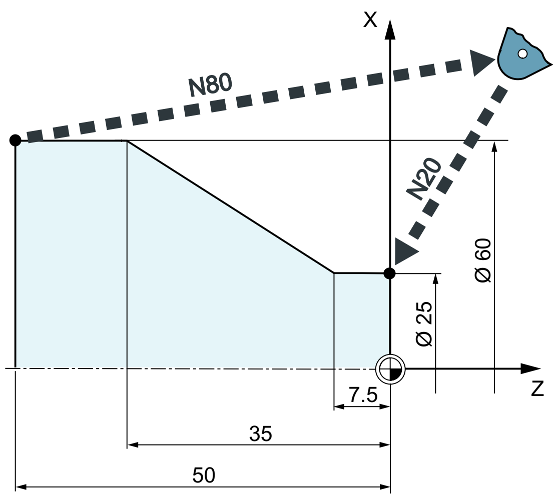
Examples

Example 1: Milling



Program code	Comment
`N10 G90 S400 M3`	`; Absolute dimension input, spindle clockwise`
`N20 G0 X30 Y20 Z2`	`; Approach the starting position`
`N30 G1 Z-5 F1000`	`; Tool infeed`
`N40 X80 Y65`	`; Traversing along a straight line`
`N50 G0 Z2`	
`N60 G0 X-20 Y100 Z100 M30`	`; Retract tool, end of program`

Example 2: Turning



Program code	Comment
`N10 G90 S400 M3`	`; Absolute dimension input, spindle clockwise`
`N20 G0 X25 Z5`	`; Approach the starting position`
`N30 G1 G94 Z0 F1000`	`; Tool infeed`
`N40 G95 Z-7.5 F0.2`	
`N50 X60 Z-35`	`; Traversing along a straight line`
`N60 Z-50`	
`N70 G0 X62`	
`N80 G0 X80 Z20 M30`	`; Retract tool, end of program`

5.7.3.2 Switch on/off linear interpolation for rapid traverse movements (RTLION, RTLIOF)

Independently of the default setting (MD20730 $MC_G0_LINEAR_MODE), the interpolation response for rapid traverse movements can also be set in the part program using the commands of the G group 55.

Syntax

```
RTLIOF
...
RTLION
```

Meaning

`RTLIOF`:	G command for switching off the linear interpolation ⇒ In the rapid traversing mode (G0), the **non-linear** interpolation is active. All of the path axes reach their end points independently of one another.	
	Effective:	Modal

`RTLION`:	G command for switching on the linear interpolation ⇒ In the rapid traversing mode (G0), the **linear** interpolation is active. All of the path axes reach their end points simultaneously.	
	Effective:	Modal

Note

Preconditions for RTLIOF

To ensure, with RTLIOF **non-linear** interpolation, the following conditions must be fulfilled:

- No transformation (TRAORI, TRANSMIT, etc.) active.
- G60 active (stop at the block end).
- No compressor active (COMPOF).
- No tool radius compensation active (G40).
- No contour handwheel selected.
- No nibbling active.

If one of these conditions is not met, linear interpolation is as with RTLION.

Example

Program code	**Comment**
	`; Linear interpolation is the default:`
	`; MD20730 $MC_GO_LINEAR_MODE == TRUE`
`...`	
`N30` **`RTLIOF`**	`; Switch off linear interpolation.`
`N40 G0 X0 Y10`	`; G0 blocks are traversed using non-linear inter-polation.`
`N50 G41 X20 Y20`	`; TRC active  G0 blocks are traversed using lin-ear interpolation.`
`N60 G40 X30 Y30`	`; TRC not active  G0 blocks are traversed using non-linear interpolation.`
`N70` **`RTLION`**	`; Switch on linear interpolation.`
`...`	

Further information

Reading the current interpolation behavior

The current interpolation behavior can be read via the system variables $AA_G0MODE.

5.7.3.3 Adapt tolerance factor for rapid traverse movements (STOLF)

The tolerance factor for rapid traverse movements (G0), which is preset using machine data (MD20560 $MC_G0_TOLERANCE_FACTOR), can be adapted in the part program by programming STOLF. In this case, the value in the machine data is not changed. After reset or end of part program, the tolerance factor set in the machine data becomes effective again.

Syntax

```
STOLF=<value>
```

Meaning

`STOLF`:	Address to program a tolerance factor for rapid traverse movements			
	Applications:	• Compressor functions COMPON, COMPCURV, COMPCAD and COMPSURF • Smoothing functions G642 and G645 • Orientation smoothing OST • Orientation smoothing ORISON • Smoothing for path-relevant orientation ORIPATH		
	Preprocessing stop:	No		
	Effective:	Modal		
	`<Value>`:	Tolerance factor The tolerance factor can be both greater or less than 1.0. If the factor is equal to 1.0 (default value), then the same tolerances are active for rapid traverse movements as for non-rapid traverse movements. Normally, the tolerance factor is set to > 1.0.		
		Type:	REAL	
		Range of values:	≥ 0:	Tolerance value
			< 0:	The programmed tolerance value is deleted ⇒ The tolerance value parameterized in the machine data becomes effective again.

Example

Program code	**Comment**
`COMPCAD G645 G1 F10000`	`; Compressor function COMPCAD`
`X... Y... Z...`	`; The machine and setting data apply here.`
`X... Y... Z...`	
`X... Y... Z...`	
`G0  X... Y... Z...`	

`G0  X... Y... Z...`	`; Machine data $MC_G0_TOLERANCE_FACTOR (e.g. =3) is effective here, i.e. a smoothing tolerance of:`
	`$MC_G0_TOLERANCE_FACTOR * $MA_COMPRESS_POS_TOL`

`CTOL=0.02`	
`STOLF=4`	
`G1 X... Y... Z...`	`; A contour tolerance of 0.02 mm is applied starting from here.`
`X... Y... Z...`	
`X... Y... Z...`	
`G0 X... Y... Z...`	

Program code	Comment
`X... Y... Z...`	`; From here, a G0 tolerance factor of 4 applies, i.e. a contour tolerance of 0.08 mm.`

Additional information

Reading of the tolerance factor currently in effect

The tolerance factor for rapid traverse movements effective in the part program or in the current IPO block can be read using system variables.

- In synchronized actions or with preprocessing stop in the part program via system variable:

$AC_STOLF	Active G0 tolerance factor G0 tolerance factor, which was effective when processing the actual main run block.

- Without preprocessing stop in the part program via system variable:

$P_STOLF	Programmed G0 tolerance factor

If no value with STOLF is programmed in the active part program, then these two system variables supply the value set using MD20560 $MC_G0_TOLERANCE_FACTOR.

If no rapid traverse (G0) is active in a block, then these system variables always supply a value of 1.

5.8 RESET behavior

MD20150

With the reset (channel or mode group reset), the initial setting parameterized channel-specifically becomes effective for all G groups:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

The following G groups are of relevance to "continuous-path mode, exact stop, LookAhead":

- Group 10: Exact stop - continuous-path mode
- Group 12: Block-change criterion for exact stop
- Group 21: Acceleration profile
- Group 30: NC block compression
- Group 59: Dynamic response mode for path interpolation

For detailed information on setting initial states, see Section "K1: Mode group, channel, program operation, reset response (Page 479)."

5.9 Supplementary conditions

5.9.1 Block change and positioning axes

If path axes are traversed in continuous path mode in a part program, traversing positioning axes can also simultaneously affect both the response of the path axes and the block change.

A detailed description of the positioning axes can be found in:
References:
Function Manual, Extended Functions; Positioning axes (P2)

5.9.2 Block change delay

Even if all path axes and special axes traversing in the part program block have satisfied their specific block transition criteria, the block change can still be delayed due to other unsatisfied conditions and/or active functions:

Examples:

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

Effects

If a block change cannot be executed in continuous path mode, all axes programmed in this part program block (except cross-block traversing special axes) are stopped. In this case, contour errors do not occur.

The stopping of path axes **during machining** can cause undercuts on the workpiece surface.

5.10 Data lists

5.10.1 Machine data

5.10.1.1 General machine data

Number	Identifier: $MN_	Description
10110	PLC_CYCLE_TIME_AVERAGE	Average PLC acknowledgment time
10680	MIN_CONTOUR_SAMPLING_TIME	Minimum contour sampling time
10682	CONTOUR_SAMPLING_FACTOR	Contour sampling factor
10712	NC_USER_CODE_CONF_NAME_TAB	List of reconfigured NC commands
12030	OVR_FACTOR_FEEDRATE	Evaluation of the path feedrate override switch

Number	Identifier: $MN_	Description
12100	OVR_FACTOR_LIMIT_BIN	Limit for binary-coded override switch
18360	MM_EXT_PROG_BUFFER_SIZE	FIFO buffer size for execution from external source (DRAM)

5.10.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Basic setting of the groups
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of an NC block that can be compressed
20171	SURF_BLOCK_PATH_LIMIT	Maximum traversing length of an NC block for compression with COMPSURF
20172	COMPRESS_VELO_TOL	Maximum permissible deviation from path feed for compression
20173	SURF_VELO_TOL	Maximum permissible deviation from path feed for compression with COMPSURF
20400	LOOKAH_USE_VELO_NEXT_BLOCK	LookAhead following block velocity
20430	LOOKAH_NUM_OVR_POINTS	Number of override switch points for LookAhead
20440	LOOKAH_OVR_POINTS	Override switch points for LookAhead
20443	LOOKAH_FFORM	Activating the extended LookAhead
20450	LOOKAH_RELIEVE_BLOCK_CYCLE	Relief factor for the block cycle time
20455	LOOKAH_FUNCTION_MASK	Special LookAhead functions
20460	LOOKAH_SMOOTH_FACTOR	Smoothing factor for LookAhead
20462	LOOKAH_SMOOTH_WITH_FEED	Smoothing with programmed feed
20465	ADAPT_PATH_DYNAMIC	Adaptation of path dynamic response
20480	SMOOTHING_MODE	Rounding behavior with G64x
20482	COMPRESSOR_MODE	Principle of operation of the compressor
20484	COMPRESSOR_PERFORMANCE	Compressor performance
20485	COMPRESS_SMOOTH_FACTOR	Smoothing factor for compression with COMPCAD for the particular dynamic mode
20486	COMPRESS_SPLINE_DEGREE	Spline degree for compression with COMPCAD for the particular dynamic mode
20487	COMPRESS_SMOOTH_FACTOR_2	Smoothing factor for rotary axes for compression with COMPCAD for the particular dynamic mode
20488	SPLINE_MODE	Setting for spline interpolation
20490	IGNORE_OVL_FACTOR_FOR_ADIS	G641/G642 independent of the overload factor
20550	EXACT_POS_MODE	Exact-stop conditions with G0/G1
20552	EXACT_POS_MODE_G0_TO_G1	Exact stop criterion for rapid traverse transitions in the continuous path mode
20560	G0_TOLERANCE_FACTOR	G0 tolerance factor
20600	MAX_PATH_JERK	Pathrelated maximum jerk
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

Number	Identifier: $MC_	Description
20606	PREPDYN_SMOOTHING_ON	Activation of the curvature smoothing
20730	G0_LINEAR_MODE	Interpolation behavior with G0
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)
28071	MM_NUM_SURF_LEVELS	Dimensioning the COMPSURF function (DRAM)
28072	MM_MAXNUM_SURF_GROUPS	Dimensioning the COMPSURF function regarding axis groups (DRAM)
28520	MM_MAX_AXISPOLY_PER_BLOCK	Maximum number of axis polynomials per block
28530	MM_PATH_VELO_SEGMENTS	Number of storage elements for limiting path velocity in block
28533	MM_LOOKAH_FFORM_UNITS	Memory for extended LookAhead
28540	MM_ARCLENGTH_SEGMENTS	Number of storage elements for arc length function representation per block
28610	MM_PREPDYN_BLOCKS	Number of blocks for velocity preparation

5.10.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32000	MAX_AX_VELO	Maximum axis velocity
32310	MAX_ACCEL_OVL_FACTOR	Overload factor for axial velocity jumps
32431	MAX_AX_JERK	Maximum axial jerk when traversing along the path
32432	PATH_TRANS_JERK_LIM	Maximum axial jerk at the block transition in continuous-path mode
32433	SOFT_ACCEL_FACTOR	Scaling of acceleration limitation for SOFT
32434	G00_ACCEL_FACTOR	Scaling of acceleration limitation for G00
32435	G00_JERK_FACTOR	Scaling of axial jerk limitation for G00
32440	LOOKAH_FREQUENCY	Smoothing limit frequency for LookAhead
33100	COMPRESS_POS_TOL	Maximum deviation with compensation
33120	PATH_TRANS_POS_TOL	Maximum deviation when rounding with G645
35240	ACCEL_TYPE_DRIVE	DRIVE acceleration characteristic for axes on/off
36000	STOP_LIMIT_COARSE	Exact stop coarse
36010	STOP_LIMIT_FINE	Exact stop fine
36012	STOP_LIMIT_FACTOR	Exact stop coarse/fine factor and zero speed monitoring
36020	POSITIONING_TIME	Delay time exact stop fine

5.10.2 Setting data

5.10.2.1 Channelspecific setting data

Number	Identifier $SC_	Description
42465	SMOOTH_CONTUR_TOL	Max. contour deviation during rounding
42466	SMOOTH_ORI_TOL	Max. deviation of the tool orientation during rounding
42470	CRIT_SPLINE_ANGLE	Corner angle limit for compression with COMPCAD
42471	MIN_CURV_RADIUS	Minimum radius of curvature for compression with COMPCAD
42472	MIN_SURF_RADIUS	Minimum radius of curvature for compression with COMPSURF
42473	ACTNUM_SURF_GROUPS	Number of axis groups for COMPSURF
42475	COMPRESS_CONTUR_TOL	Maximum permissible contour deviation with compression
42476	COMPRESS_ORI_TOL	Maximum deviation of the tool orientation for compression **Note:** Only for active orientation transformation!
42477	COMPRESS_ORI_ROT_TOL	Maximum deviation of the tool rotation for compression **Note:** Only for 6-axis machines with rotatable tool!
42676	ORI_SMOOTH_TOL	Tolerance for smoothing of the orientation when smoothing
42678	ORISON_TOL	Smoothing tolerance of the orientation

5.10.3 Signals

5.10.3.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
All axes stationary	DB21, ...DBX36.3	DB330x.DBX4.3

5.10.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Position reached with exact stop coarse	DB31, ...DBX60.6	DB390x.DBX0.6
Position reached with exact stop fine	DB31, … .DBX60.7	DB390x.DBX0.7

B2: Acceleration 6

6.1 Brief description

6.1.1 General information

Scope of functions

The Description of Functions covers the following sub-functions:

- Acceleration
- Jerk
- Kneeshaped acceleration characteristic

Acceleration and jerk

The effective acceleration and jerk can be optimally matched to the machine and machining situation concerned using axis- and channel-specific programmable maximum values, programmable acceleration profiles in part programs and synchronized actions, and dynamic adaptations and limitations.

Kneeshaped acceleration characteristic

The knee-shaped acceleration characteristic means that, in the case of machine axes featuring a motor (in particular stepper motors) with a torque characteristic that is highly dependent upon speed, acceleration can be set at the level required to ensure optimum utilization of the motor whilst at the same time protecting it against overload.

6.1.2 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)

6.2 Functions

6.2.1 Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific)

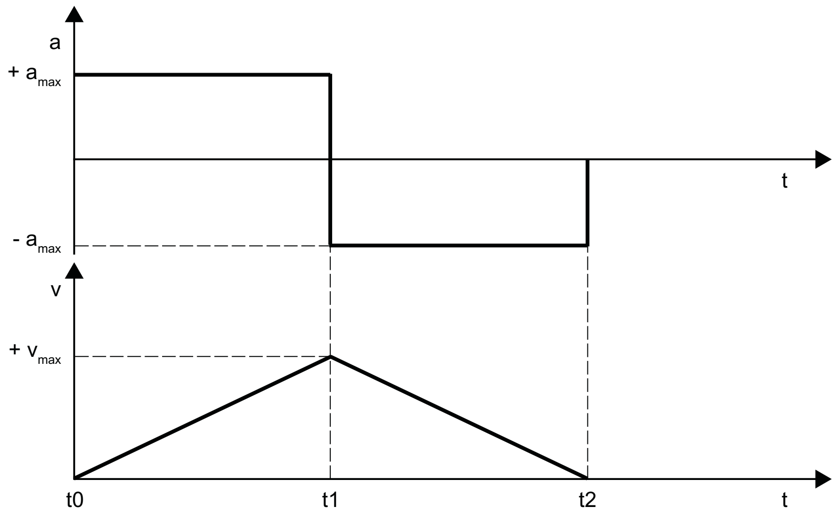
6.2.1.1 General Information

General Information

In the case of acceleration without jerk limitation (jerk = infinite) the maximum value is applied for acceleration immediately. As regard to acceleration with jerk limitation, it differs in the following respects:

- **Advantages**
  Shorter processing times with the same maximum values for velocity and acceleration.
- **Disadvantages**
  Increased load on the machine's mechanical components and risk of inducing high-frequency and difficult-to-control mechanical vibrations.

Acceleration profile



a_{max}:	Maximum acceleration value
v_{max}:	Maximum velocity value
t:	Time

Figure 6-1 Velocity and acceleration schematic for stepped acceleration profile

The following features of the acceleration profile can be identified from the figure above:

- Time: t_0
  Sudden acceleration from 0 to $+a_{max}$
- Interval: t_0 - t_1
  Constant acceleration with $+a_{max}$; linear increase in velocity

- Time: t_1
  Sudden acceleration from 2 * a_{max} with immediate switchover from acceleration to braking

Note

The sudden acceleration can normally be avoided by specifying a constant velocity time (see Section "Constant travel time (channel-specific) (Page 272)").

- Interval: t_1 - t_2
  Constant acceleration with -a_{max}; linear decrease in velocity

6.2.1.2 Parameterization

Maximum axial acceleration for path motions

The maximum axial acceleration for path motions can be set for the specific technology for each machine axis via the following machine data:

MD32300 $MA_MAX_AX_ACCEL[<parameter set index>]

mit <parameter set index> = 0, 1, 2 ... (max. parameter set number - 1)

For the technology-specific parameter sets, see Chapter "Dynamic response mode for path interpolation (Page 235)".

The path parameters are calculated by the path planning of the preprocessing so that the parameterized maximum values of the machine axes involved in the path are not exceeded.

Note

It is possible for the maximum value to be exceeded in connection with specific machining situations (see Chapters "Acceleration matching (ACC) (axis-specific) (Page 274)" and "Path acceleration for real-time events (channel-specific) (Page 277)").

Maximum axial acceleration for positioning axis motions

With positioning axis motions, one of the two following maximum values is effective depending on the set positioning axis dynamic response mode:

- MD32300 $MA_MAX_AX_ACCEL [0] (maximum axial acceleration for path motions in the dynamic response mode DYNNORM)
- MD32300 $MA_MAX_AX_ACCEL [1] (maximum axial acceleration for path motions in the dynamic response mode DYNPOS)

The positioning axis dynamic response mode is set in the NC-specific machine data:

MD18960 $MN_POS_DYN_MODE = <mode>

<mode>	Meaning
0	Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[0]
1	Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[1]

Maximum axial acceleration for JOG motions

See Chapter "Acceleration and jerk for JOG motions (Page 306)".

6.2.1.3 Programming

Path acceleration without jerk limitation (BRISK)

Syntax

```
BRISK
```

Functionality

The `BRISK` part-program instruction is used to select the "without jerk limitation" acceleration profile for the purpose of path acceleration.

G group: 21

Effective: Modal

Reset response

The channel-specific initial setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Supplementary conditions

If the acceleration profile is changed in a part program during machining (`BRISK`/`SOFT`) an exact stop is performed at the end of the block.

Single-axis acceleration without jerk limitation (BRISKA)

Syntax

`BRISKA (`*axis{,axis}*`)`

Function

The `BRISKA` part-program command is used to select the "without jerk limitation" acceleration profile for single-axis movements (JOG, JOG/INC, positioning axis, reciprocating axis, etc.).

G group: -

Effectiveness: Modal

Axis:

- Value range: Axis name of the 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 behavior

The axis-specific initial setting is activated via a reset:

MD32420 $MA_JOG_AND_POS_ENABLE

6.2.2 Constant travel time (channel-specific)

6.2.2.1 General Information

Overview

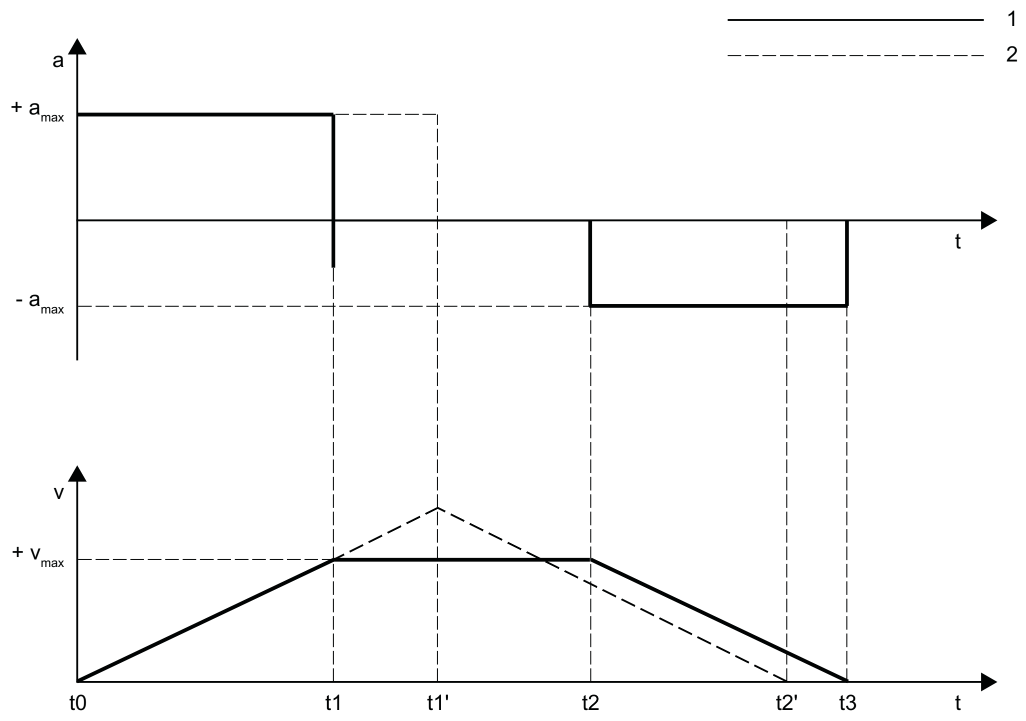
For acceleration without jerk limitation, an acceleration jump of 2 * a_{max} occurs when changing from acceleration and braking. To prevent this acceleration jump, a channel-specific constant traversing time can be parameterized. The constant traversing time specifies the time for constant velocity traversing between the acceleration and the braking phase:

MD20500 $MC_CONST_VELO_MIN_TIME (minimum time with constant velocity)

Note

Constant traversing time does not act for:

- Active function: LookAhead
- In traversal blocks with a traversal time less than or equal to the interpolation cycle.



1:	Curve with constant traversing time
2:	Curve without constant traversing time
a_{max}:	Maximum acceleration value
v_{max}:	Maximum velocity value
t:	Time

Figure 6-2 Principle characteristic for an abrupt acceleration

The above figure shows the effect of the constant traversing time:

- Instant: t_1
  End of the acceleration phase with acceleration jump 1 * a_{max}
- Interval: t_1 - t_2
  Acceleration 0; constant velocity over the parameterized constant traversing time
- Instant: t_2
  Start of the braking phase with acceleration jump 1 * a_{max}

The instants t_0, t_1' and t_2' indicate the associated curve that would have resulted without constant traversing time.

6.2.2.2 Parameterization

The constant travel time is parameterized for specific channels using machine data:

MD20500 $MC_CONST_VELO_MIN_TIME
(minimum time with constant velocity)

6.2.3 Acceleration matching (ACC) (axis-specific)

6.2.3.1 General Information

Function

Using the `ACC` command, the currently effective maximum axis acceleration parameterized in the acceleration-specific machine data can be reduced for a specific axis. The reduction is in the form of a percentage factor, which is specified when programming the command.

For instance, reducing the maximum possible axis acceleration in a machining segment can be used to prevent mechanical vibration as a result of high machining forces.

Effectiveness

Reducing the acceleration with `ACC` is **effective** for all interpolation types in the AUTOMATIC and MDI operating modes – and for the dry run feedrate.

Acceleration reduction is **not effective**:

- in the JOG mode
- During the machine function REF (reference point approach).
- If, as a result of a fault that has been detected, the axis is stopped with a fast stop (setpoint = 0).

6.2.3.2 Programming

Syntax

```
ACC[<axis>]=<reduction factor>

ACC[SPI(<spindle number>)]=<reduction factor>

ACC(S<spindle number>)=<reduction factor>
```

Meaning

`<ACC>`:	Command for the axis-specific reduction of the currently maximum possible acceleration, derived from the machine data	
	Effectiveness:	Modal
`<axis>`:	Channel axis name of path axis	
	Data type:	AXIS
	Value range:	Channel axis names
`SPI(<spindle number>)`:	The `SPI(...)` function converts the spindle number into the corresponding channel axis name.	
`S<spindle number>`:	Spindle name in the channel	

Further information

System variable

The acceleration reduction (set using ACC), currently active in the channel, can be read on an axis-for-axis basis using:

```
$AA_ACC[<axis>]
```

Reset response

The acceleration reduction set using ACC can be kept after a channel reset or after the end of the program. The parameterization is performed via:

MD32320 $MA_DYN_LIMIT_RESET_MASK

6.2.4 Acceleration margin (channel-specific)

6.2.4.1 General Information

General information

Under normal circumstances, preprocessing makes maximum use of the parameterized maximum values of the machine axes for the purpose of path acceleration. In order that an acceleration margin may be set aside for overlaid movements, e.g., within the context of the "Rapid lift away from the contour" function, path acceleration can be reduced by a programmable factor. When, for example, a factor of 0.2 is applied, preprocessing will only use 80% of the maximum possible path acceleration. 20% is set aside as an acceleration margin for overlaid movements.

6.2.4.2 Parameterization

Parameters for the acceleration margin are assigned for each channel by means of machine datum:
MD20610 $MC_ADD_MOVE_ACCEL_RESERVE
(acceleration margin for overlaid motions)

6.2.5 Path-acceleration limitation (channel-specific)

6.2.5.1 General Information

General Information

To enable a flexible response to the machining situations concerned, setting data can be used to limit the path acceleration calculated during preprocessing for specific channels:

SD42500 $SC_SD_MAX_PATH_ACCEL (maximum path acceleration)

The value specified in the setting data is only taken into account if it is smaller than the path acceleration calculated during preprocessing.

The limitation must be activated for specific channels using setting data:

SD42502 $SC_IS_SD_MAX_PATH_ACCEL = TRUE

6.2.5.2 Parameterization

Parameterization is carried out for specific channels using setting data:

SD42500 $SC_SD_MAX_PATH_ACCEL (maximum path acceleration)

SD42502 $SC_IS_SD_MAX_PATH_ACCEL (activation of path-acceleration limitation)

6.2.5.3 Programming

Limit value

Syntax

`$SC_SD_MAX_PATH_ACCEL =` *limit value*

Functionality

The path-acceleration limitation can be adjusted for the situation by programming the setting data.

Limit value:

- Value range: ≥ 0
- Unit: m/s^2

Application:

- Part program
- Static synchronized action

Switch ON/OFF

Syntax

`$SC_IS_SD_MAX_PATH_ACCEL =` *value*

Functionality

The path-acceleration limitation can be activated/deactivated by programming the setting data.

Parameter: *Value*

- Value range: TRUE, FALSE

Application:

- Part program
- Static synchronized action

6.2.6 Path acceleration for real-time events (channel-specific)

6.2.6.1 General Information

General Information

So that no compromise has to be made between machining-optimized acceleration on the one hand and time-optimized acceleration in connection with the following real-time events on the other:

- NC Stop / NC Start
- Changing the feedrate override
- Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function

For the real-time events mentioned above, the path acceleration can be specified using a channel-specific system variable:

`$AC_PATHACC =` *path acceleration*

Real-time event acceleration will only be active for the duration of the change in velocity in respect of one of the real-time events specified above.

Limitation

If the specified path acceleration exceeds the capabilities of the machine axes that are of relevance for the path, a limit will be imposed on the path acceleration within the controller so that the resulting axial acceleration (a_{res}) is restricted to less than 2x the parameterized maximum axial value (a_{max}).

$a_{res} = 2 * a_{max,}$ with a_{max} = MD32300 $MA_MAX_AX_ACCEL

Note

Path acceleration for real-time events is enabled, irrespective of the radial acceleration.

Effectiveness

Effective	Real-time event acceleration is only enabled in AUTOMATIC and MDA operating modes in conjunction with the following real-time events: • NC Stop / NC Start • Override changes • Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function
Not effective	Path acceleration for real-time events is ineffective for changes in path velocity that are attributable to path planning during preprocessing for the channel, such as contour curvatures, corners, kinematic transformation limitations, etc. Real-time-event path acceleration is ineffective if the programmed value is smaller than the path acceleration calculated during preprocessing for the path section concerned.

Programming

For information about programming system variables in the part program or synchronized actions, see Section "Programming (Page 278)".

6.2.6.2 Programming

Syntax

`$AC_PATHACC =` *path acceleration*

Functionality

Real-time-event path acceleration is set via the channel-specific system variables.

Parameter: *Path acceleration*

- Value range: Path acceleration ≥ 0
- Unit: m/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`).

6.2.7 Acceleration with programmed rapid traverse (G00) (axis-specific)

6.2.7.1 General Information

Frequently, the acceleration for the machine axes involved in the machining process must be set lower than the machine's performance capability officially allows because of the supplementary conditions associated with the specific process concerned.

For time-optimized traversing of the machine axes with programmed rapid traverse (part-program instruction `G00`), a specific maximum value can be programmed for the axis-specific acceleration.

JOG setup mode

This function does not affect acceleration in respect of a rapid traverse override in JOG setup mode.

6.2.7.2 Parameterization

The maximum value for axis-specific acceleration with programmed rapid traverse is parameterized (`G00`) using the axis-specific machine data:

MD32434 $MA_G00_ACCEL_FACTOR
(scaling of the acceleration limitation with G00)

This is used to generate the maximum value for axis-specific acceleration with programmed rapid traverse (`G00`) that is taken into account by the path planning component during preprocessing:

Acceleration[axis] =
MD32300 $MA_MAX_AX_ACCEL * MD32434 $MA_G00_ACCEL_FACTOR

6.2.8 Acceleration with active jerk limitation (SOFT/SOFTA) (axis-specific)

6.2.8.1 General Information

Function

Compared with acceleration without jerk limitation, acceleration with jerk limitation results in a certain degree of time loss, even when the same maximum acceleration value is used. To compensate for this time loss, a specific maximum value can be programmed for the axis-specific acceleration as far as traversing of the machine axes with active jerk limitation (`SOFT`/`SOFTA`) is concerned.

The maximum value for acceleration with active jerk limitation is parameterized using a factor calculated in relation to the axis-specific maximum value. This is used to generate the maximum value for axis-specific acceleration with active jerk limitation that is taken into account by the path planning component during preprocessing:

Acceleration[axis] =
MD32300 $MA_MAX_AX_ACCEL * MD32433 $MA_SOFT_ACCEL_FACTOR

6.2.8.2 Parameterization

The maximum value for acceleration with active jerk limitation (`SOFT`/`SOFTA`) is parameterized using the axis-specific machine data:

MD32434 $MA_SOFT_ACCEL_FACTOR
(scaling of the acceleration limitation with SOFT)

6.2.9 Excessive acceleration for non-tangential block transitions (axis-specific)

6.2.9.1 General Information

Function

In the case of non-tangential block transitions (corners), the programmable controller may have to decelerate the geometry axes significantly in order to ensure compliance with the parameterized axis dynamics. For the purpose of reducing/avoiding deceleration in connection with non-tangential block transitions, a higher level of axis-specific acceleration can be enabled.

Excessive acceleration is parameterized using a factor calculated in relation to the axis-specific maximum value. This is used to generate the maximum value for axis-specific acceleration with non-tangential block transitions that is taken into account by the path planning component during preprocessing:

Acceleration[axis] =
MD32300 $MA_MAX_AX_ACCEL * MD32310 $MA_MAX_ACCEL_OVL_FACTOR

6.2.9.2 Parameterization

Excessive acceleration for non-tangential block transitions is parameterized using the axis-specific machine data:
MD32310 $MA_MAX_ACCEL_OVL_FACTOR
(overload factor for velocity jumps)

6.2.10 Acceleration margin for radial acceleration (channel-specific)

6.2.10.1 General Information

Overview

In addition to the path acceleration (tangential acceleration), radial acceleration also has an effect on curved contours. If this is not taken into account during parameterization of the path parameters, the effective axial acceleration during acceleration and deceleration on the curved contour can, for a short time, reach 2x the maximum value.

Effective axial acceleration =

Path acceleration + radial acceleration =

2 * (MD32300 $MA_MAX_AX_ACCEL)

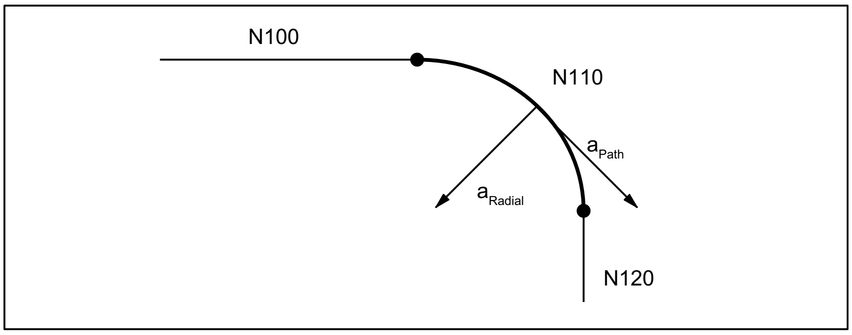


Figure 6-3 Radial and path acceleration on curved contours

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:

$$\boxed{a_{\text{Radial}} = \frac{v^2_{\text{Path}}\ [\text{m/min}]}{r\ [\text{mm}] * 3.6\ [\text{m/s}^2]} = \frac{5^2}{10 * 3.6} = 0.694\ \text{m/s}^2}$$

The acceleration margin is set as follows:

$$\boxed{\begin{array}{l}\text{MD20602 \$MC\_CURV\_EFFECT\_ON\_PATH\_ACCEL} = \\ \dfrac{a_{\text{Radial}}\ [\text{m/s}^2]}{\text{MD32300 \$MA\_MAX\_AX\_ACCEL}\ [\text{m/s}^2]} = \dfrac{0.694}{3} \approx 0.23\end{array}}$$

Linear motions

The acceleration margin referred to above is ineffective in the case of linear motions (linear interpolation) without active kinematic transformation.

6.2.10.2 Parameterization

The proportion of maximum available axis acceleration to be taken into account as an acceleration margin for radial acceleration on curved contours is parameterized using the channel-specific machine data:

MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL
(influence of path curvature on dynamic path response)

6.2.11 Jerk limitation with path interpolation (SOFT) (channel-specific)

6.2.11.1 General Information

Overview

As far as the functionality described in the rest of this document is concerned, constant acceleration, i.e., acceleration with jerk limitation (jerk = infinite value), is the assumed acceleration profile. In the case of acceleration with jerk limitation, linear interpolation is applied in respect of acceleration from 0 to the maximum value.

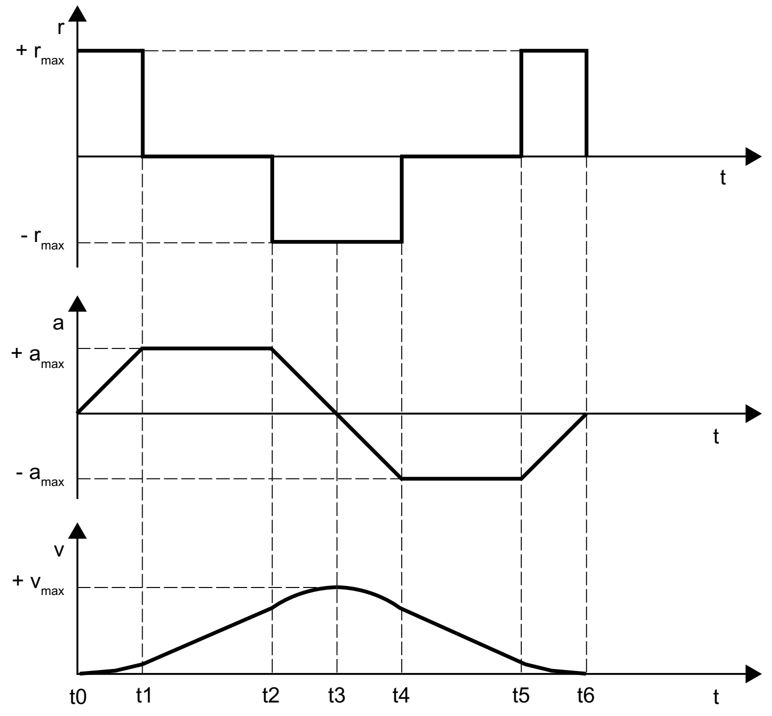
Advantages

Minimal load on the machine's mechanical components and low risk of high-frequency and difficult-to-control mechanical vibrations thanks to constant excessive acceleration.

Disadvantages

Longer machining times compared with stepped acceleration profile when the same maximum velocity and acceleration values are used.

Acceleration profile



r_{max}:	Maximum jerk value
a_{max}:	Maximum acceleration value
v_{max}:	Maximum velocity value
t:	Time

Figure 6-4 Jerk, acceleration and velocity schematic with jerk limitation acceleration profile

The following features of the acceleration profile can be identified from the figure above:

- Interval: t_0 - t_1
  Constant jerk with $+r_{max}$; linear increase in acceleration; quadratic increase in velocity
- Interval: t_1 - t_2
  Constant acceleration with $+a_{max}$; linear increase in velocity
- Interval: t_2 - t_3
  Constant jerk with $-r_{max}$; linear decrease in acceleration; quadratic decrease in excessive velocity until maximum value $+v_{max}$ is reached
- Interval: t_3 - t_4
  Constant jerk with $+r_{max}$; linear increase in braking acceleration; quadratic decrease in velocity
- Interval: t_4 - t_5
  Constant braking acceleration with $-a_{max}$; linear decrease in velocity
- Interval: t_5 - t_6
  Constant jerk with $-r_{max}$; linear decrease in braking acceleration; quadratic decrease in velocity reduction until zero velocity is reached v = 0

6.2.11.2 Parameterization

Maximum jerk value for path motions (axis-specific)

The maximum axial jerk for path motions can be set for the specific technology for each machine axis via the following machine data:

MD32431 $MA_MAX_AX_JERK[<parameter set index>]

With <parameter set index> = 0, 1, 2 ... (max. parameter set number - 1)

For the technology-specific parameter sets, see Section "Dynamic response mode for path interpolation (Page 235)".

The path parameters are calculated by the path planning of the preprocessing so that the parameterized maximum values of the machine axes involved in the path are not exceeded.

Note

It is possible for the maximum value to be exceeded in connection with specific machining situations (see Section "Path jerk for real-time events (channel-specific) (Page 289)").

Maximum jerk value for path motions (channel-specific)

In addition to the axis-specific setting, the maximum jerk value can also be specified as channel-specific path parameter via the following machine data:

MD20600 $MC_MAX_PATH_JERK (path-related maximum jerk)

In order to exclude the mutual influencing of axis and channel-specific maximum jerk values, the channel-specific maximum value must be set to a value greater than the axial maximum values.

6.2.11.3 Programming

Syntax

```
SOFT
```

Functionality

The `SOFT` part-program instruction is used to select the acceleration profile with jerk limitation for the traversing operations of geometry axes in the channel.

G group: 21

Effective: Modal

Reset response

The channel-specific initial setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Boundary conditions

If the acceleration mode is changed in a part program during machining (`BRISK` ↔ `SOFT`), a block change is performed at the point of transition with an exact stop at the end of the block, even in continuous-path mode.

6.2.12 Jerk limitation with single-axis interpolation (SOFTA) (axis-specific)

6.2.12.1 Parameterization

Basic setting of axis-specific jerk limitation

Acceleration with jerk limitation can be set as the axis-specific initial setting:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE== TRUE

Maximum axis-specific jerk for positioning axis movements

When traversing positioning axes with active jerk limitation, the value from one of the following machine data takes effect as the maximum axis-specific jerk:

- MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axis-specific jerk)
- MD32431 $MA_MAX_AX_JERK [0] (maximum axis-specific jerk for path motions in the dynamic response mode DYNNORM)
- MD32431 $MA_MAX_AX_JERK [1] (maximum axis-specific jerk for path motions in dynamic response mode DYNPOS)

The machine data to be used is determined by the set positioning axis dynamic response mode:

MD18960 $MN_POS_DYN_MODE = <mode>

<mode>	Meaning
	The following applies as the maximum axis-specific jerk for positioning axis motions:
0	MD32430 $MA_JOG_AND_POS_MAX_JERK With active G75 (fixed-point approach): MD32431 $MA_MAX_AX_JERK[**0**]
1	MD32431 $MA_MAX_AX_JERK[**1**]

Maximum axial jerk for JOG motions

See Chapter "Acceleration and jerk for JOG motions (Page 306)".

6.2.12.2 Programming

Syntax

`SOFTA` (*Axis* {*Axis*})

Functionality

The `SOFTA` part-program command is used to select acceleration with jerk limitation for single-axis movements (positioning axis, reciprocating axis, etc.)

G group: -

Effectiveness: Modal

Axis:

- Value range: Axis name of the 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 behavior

The axis-specific initial setting is activated via a reset:

MD32420 $MA_JOG_AND_POS_ENABLE

6.2.13 Path-jerk limitation (channel-specific)

6.2.13.1 General Information

Overview

To enable a flexible response to the machining situations concerned, setting data can be used to limit the path jerk calculated during preprocessing for specific channels:

SD42510 $SC_SD_MAX_PATH_JERK (maximum path jerk)

The value specified in the setting data is only taken into account in the channel if it is smaller than the path jerk calculated during preprocessing.

The limitation must be activated for specific channels using setting data:

SD42512 $SC_IS_SD_MAX_PATH_JERK = TRUE

6.2.13.2 Parameterization

Parameterization is carried out for specific channels using setting data:

SD42510 $SC_SD_MAX_PATH_JERK (maximum path jerk)

SD42512 $SC_IS_SD_MAX_PATH_JERK
(activation of path-jerk limitation)

6.2.13.3 Programming

Maximum path jerk

Syntax

$SC_SD_MAX_PATH_JERK = *jerk value*

Functionality

The path-jerk limitation can be adjusted for the situation by programming the setting data.

Jerk value:

- Value range: ≥ 0
- Unit: m/s^3

Application:

- Part program
- Static synchronized action

Switch ON/OFF

Syntax

$SC_IS_SD_MAX_PATH_JERK = *value*

Functionality

The path-jerk limitation can be activated/deactivated by programming the setting data.

Parameter: *Value*

- Value range: TRUE, FALSE

Application:

- Part program
- Static synchronized action

6.2.14 Path jerk for real-time events (channel-specific)

6.2.14.1 General Information

Overview

So that no compromise has to be made between machining-optimized jerk on the one hand and time-optimized jerk in connection with the following real-time events on the other:

- NC Stop / NC Start
- Changing the feedrate override
- Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function

for the real-time events mentioned, the path jerk can be specified using a channel-specific system variable:

$AC_PATHJERK = *path jerk*

Path jerk for real-time events will only be active for the duration of the change in velocity in respect of one of the real-time events specified above.

Limitation

As the jerk is not a physical variable of any relevance to the drive, **no** limit is imposed on the jerk set.

Effectiveness

Effective	Path jerk for real-time events is only enabled in AUTOMATIC and MDA operating modes in conjunction with the following real-time events: • NC Stop / NC Start • Override changes • Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function
Not effective	Path jerk for real-time events is ineffective for changes in the path velocity that are attributable to path planning during preprocessing for the channel, such as contour curvatures, corners, kinematic transformation limitations, etc. Path jerk for real-time events is ineffective if the programmed value is smaller than the path jerk calculated during preprocessing for the path section concerned.

Programming

For the purpose of setting the jerk for real-time events in accordance with the acceleration, the system variables can be set as follows:

$AC_PATHJERK = $AC_PATHACC/smoothing time

- $AC_PATHACC: Path acceleration [m/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 Section "Programming (Page 290)".

6.2.14.2 Programming

Syntax

$AC_PATHJERK = *path jerk*

Functionality

The path jerk for real-time events is set via the channel-specific system variables.

Jerk value:

- Value range: Path jerk ≥ 0
- Unit: m/s^3

Application:

- Part program
- Static synchronized action

Reset behavior

The function is deactivated on reset.

Boundary conditions

Programming $AC_PATHJERK in the part program automatically triggers a preprocessing stop with REORG (`STOPRE`).

6.2.15 Jerk with programmed rapid traverse (G00) (axis-specific)

6.2.15.1 General Information

Overview

Frequently, the maximum jerk for the machine axes involved in the machining process must be set lower than the machine's performance capability officially allows because of the supplementary conditions associated with the specific process concerned.

For time-optimized traversing of the machine axes with programmed rapid traverse (part-program instruction G00), a specific maximum value can be programmed for the axis-specific jerk.

JOG setup mode

This function does not affect jerk in respect of a rapid traverse override in JOG setup mode.

6.2.15.2 Parameterization

The maximum value for axis-specific jerk with programmed rapid traverse is parameterized (`G00`) using the axis-specific machine data:

MD32434 $MA_G00_ACCEL_FACTOR
(scaling of the acceleration limitation with G00)

This is used to generate the maximum value for axis-specific jerk with programmed rapid traverse (`G00`) that is taken into account by the path planning component during preprocessing:

Jerk[axis] =
MD32431 $MA_MAX_AX_JERK * MD32435 $MA_G00_JERK_FACTOR

6.2.16 Excessive jerk for block transitions without constant curvature (axis-specific)

6.2.16.1 General Information

Overview

In the case of block transitions without constant curvature (e.g. straight line > circle), the programmable controller has to decelerate movement of the geometry axes significantly in order to ensure compliance with the parameterized axis dynamics. For the purpose of reducing/avoiding deceleration in connection with block transitions without constant curvature, a higher level of axis-specific jerk can be enabled.

The excessive jerk is parameterized using a dedicated axis-specific maximum value.

6.2.16.2 Parameterization

The excessive jerk for block transitions without constant curvature is parameterized using the axis-specific machine data:

MD32432 $MA_PATH_TRANS_JERK_LIM
(excessive jerk for block transitions without constant curvature)

6.2.17 Velocity-dependent jerk adaptation (axis-specific)

Function

The dynamic path response results from the parameterized, constant axial maximum values for velocity, acceleration and jerk of the axes involved in the path:

- MD32000 $MA_MAX_AX_VELO (maximum axis velocity)
- MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)
- MD32431 $MA_MAX_AX_JERK (max. axial jerk for path motion)

For **contours with non-constant curvature** (torsion), as for example in connection with free-form surfaces, fluctuations in the path velocity, particularly in the upper velocity range, can result mainly due to the axial jerk. The fluctuations of the path velocity lead to adverse affects in the surface quality.

The influence of the axial jerk on the path velocity is decreased for contours with non-constant curvature through a velocity-dependent increase of the permissible axial jerk. Fluctuations in the path velocity can be avoided with the appropriate parameterization.

The velocity-dependent increase of the permissible axial jerk has no effect on the maximum possible path acceleration and path jerk. These result from the constant axial maximum values parameterized in the in the machine data even when jerk adaptation is active.

As both curvature and torsion are zero in the case of linear motion, the velocity-dependent jerk adaptation has no effect with linear motions.

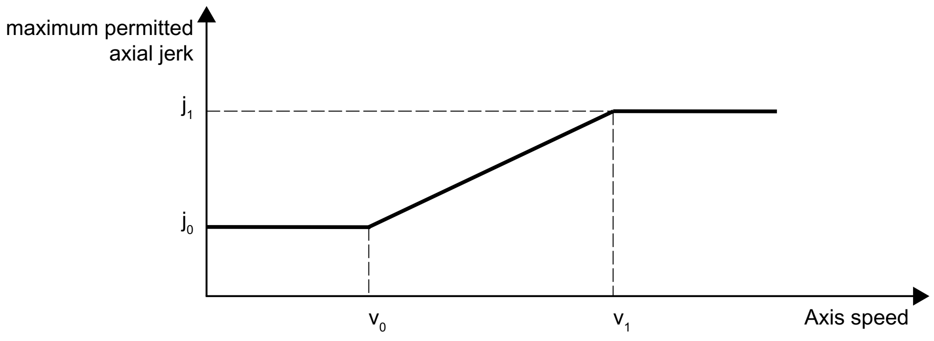
Availability

The "velocity-dependent jerk adaptation" function is available independent of the function "Free-form surface mode: Basic functions (Page 237)".

Parameterization

The "Velocity-dependent jerk adaptation" function is parameterized with the following machine data:

- MD32437 $MA_AX_JERK_VEL0[<n>] = <threshold value$_{lower}$>
  Lower velocity threshold of the jerk adaptation. Velocity-dependent jerk adaptation takes effective as of this velocity.
  The lower velocity threshold can be set separately via index n for each dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 235)"):
- MD32438 $MA_AX_JERK_VEL1[<n>] = <threshold value$_{upper}$>
  Upper velocity threshold of the jerk adaptation. The velocity-dependent jerk reaches its maximum value j_{max} parameterized with MD32439 $MA_MAX_AX_JERK_FACTOR at this velocity.
  The upper velocity threshold can be set separately via index n for each dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 235)"):
- MD32439 $MA_MAX_AX_JERK_FACTOR = <factor>
  Factor for the parameterization of the maximum velocity-dependent jerk j_{max} on reaching the upper velocity threshold MD32438 $MA_AX_JERK_VEL1[<n>]:
  j_{max} = (MD32431 $MA_MAX_AX_JERK) * (MD32439 $MA_MAX_AX_JERK_FACTOR)
  The velocity-dependent jerk adaptation is active at a value > 1.0.
  The velocity-dependent jerk adaptation is inactive at a value = 1.0.



v_0:	MD32437 $MA_AX_JERK_VEL0
v_1:	MD32438 $MA_AX_JERK_VEL1
j_0:	MD32431 $MA_MAX_AX_JERK
j_1:	MD32439 $MA_MAX_AX_JERK_FACTOR * MD32431 $MA_MAX_AX_JERK

Figure 6-5 Axial jerk as a function of the axis velocity

Note

The velocity-dependent jerk adaptation is only active, if:

MD32439 $MA_MAX_AX_JERK_FACTOR > 1.0

Example

Example of parameter assignment:

- MD32437 $MA_AX_JERK_VEL0 = 3000 mm/min
- MD32438 $MA_AX_JERK_VEL1 = 6000 mm/min
- MD32439 $MA_MAX_AX_JERK_FACTOR[AX**1**] = 2.0
- MD32439 $MA_MAX_AX_JERK_FACTOR[AX**2**] = 3.0
- MD32439 $MA_MAX_AX_JERK_FACTOR[AX**3**] = 1.0

Effect

- The velocity-dependent jerk adaptation becomes active for the 1st and 2nd axis, while the function for the 3rd axis is not active.
- The parameterized jerk is effective at axis velocities in the range of 0 to 3000 mm/min.
- The maximum jerk is linearly increased for axis velocities in the range 3000 mm/min to 6000 mm/min.
- The maximum permitted jerk of the 1st axis is, for axis velocities greater than 6000 mm/min, increased by a factor of 2 - for the 2nd axis, by factor of 3.
- The parameterized values apply in each dynamic response mode.

6.2.18 Jerk filter (axis-specific)

6.2.18.1 General Information

Overview

In certain application scenarios, e.g. when milling free-form surfaces, it may be beneficial to smooth the position setpoint characteristics of the machine axes. This enables surface quality to be improved by reducing the mechanical vibrations generated in the machine.

For the purpose of smoothing the position setpoint characteristic of a machine axis, a jerk filter can be activated at position controller level, independently of the channel- and axis-specific jerk limitations taken into account at interpolator level.

The effect of the jerk filter must be as strong as possible without having an unacceptable impact on contour accuracy. The filter should also have as "balanced" a smoothing effect as possible, i.e. if the same contour is traversed forwards and backwards, the contour smoothed by the filter should be as similar as possible in both directions.

To enable the jerk filter to be optimally matched to the machine conditions, various filter modes are available:

- 2nd-order filter (PT2)
- Sliding mean value generation
- Bandstop filter

Mode: 2nd-order filter

Owing to the fact that it is a simple low-pass filter, "2nd-order filter" mode can only meet the requirements specified above where relatively small filter time constants (around 10 ms) are concerned. When used in conjunction with larger time constants, impermissible contour deviations are soon manifest. The effect of the filter is relatively limited.

This filter mode offers advantages if very large filter time constants are needed and contour accuracy is only of secondary importance (e.g. positioning axes).

For historical reasons, this filter mode is set as the default.

Mode: Sliding mean value generation

Where minimal contour deviations are required, filter time constants within the range of 20-40 ms can be set using the "sliding mean value generation" filter mode. The smoothing effect is largely symmetrical.

The display of the calculated servo gain factor (K_V) in the user interface, shows smaller values than would normally be appropriate for the filter. The contour accuracy is in fact higher than the displayed servo gain factor (K_V) appears to suggest.

When changing from "2nd-order filter" to "sliding mean value generation" filter mode, the displayed servo gain factor (K_V) 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:

$$H(s) = \frac{\dfrac{s^2}{(2 * \pi * f_Z)^2} + \dfrac{2 * s * D_Z}{(2 * \pi * f_Z)}}{\dfrac{s^2}{(2 * \pi * f_N)^2} + \dfrac{2 * s * D_N}{2 * \pi * f_N}}$$

where:

f_Z:	Numerator natural freq.
f_N:	Denominator natural freq.
D_Z:	Numerator damping
D_N:	Denominator damping

Since a vibration-capable filter setting is not expected to yield useful results in any case, as with the jerk filter's "2nd-order filter" (PT2) low-pass filter (PT2) mode there is no setting option for the denominator damping D_N. The denominator damping D_N is permanently set to 1.

The bandstop filter can be parameterized in two different ways:

- Real bandstop filter
- Bandstop filter with additional amplitude response increase/decrease at high frequencies

Real bandstop filter

The real bandstop filter is applied when identical numerator and denominator natural frequencies are selected:

$f_Z = f_N = f_{block}$ (blocking frequency)

If numerator damping setting = 0 is selected, the blocking frequency is equivalent to complete attenuation. In this case the 3 dB bandwidth is calculated as follows:

$f_{3\ dB\ bandwidth} = 2 * f_{block}$

If instead of complete attenuation, a reduction by a factor of k is all that is required, then numerator damping should be selected in accordance with k. In this case the above formula for calculating the 3 dB bandwidth no longer applies.

Bandstop filter with additional amplitude response increase/decrease at high frequencies

In this case, the numerator and denominator natural frequencies are set to different values. The numerator natural frequency determines the blocking frequency.

By selecting a lower/higher denominator natural frequency than the numerator natural frequency, you can increase/decrease the amplitude response at high frequencies. An amplitude response increase at high frequencies can be justified in most cases, as the controlled system generally possesses a lowpass characteristic itself, i.e. the amplitude response drops at high frequencies anyway.

Supplementary conditions

If too high a numerator natural frequency is selected, the filter is disabled. In this case the limiting frequency f_{Zmax} depends on the position-control cycle:

$$f_{Zmax} = \frac{1}{2 * \pi * T_{Zmin}} = \frac{1}{2 * \pi * T_{Position\ control\ cycle\ clock}} \text{ (Shannon-Theorem)}$$

6.2.18.2 Parameterization

Activation

The jerk filter is activated using the machine data:

MD32400 $MA_AX_JERK_ENABLE (axial jerk limitation)

The jerk filter is active in all operating modes and with all types of interpolation.

Filter mode

The filter mode is selected via the machine data:

MD32402 $MA_AX_JERK_MODE

Value	Filter mode
1	2nd-order filter
2	Sliding mean value generation
3	Bandstop filter

Time constant

The time constant for the axial jerk filter is set with the machine data:

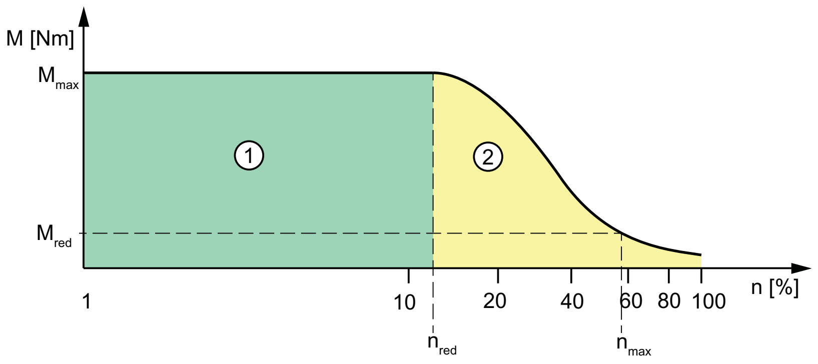
MD32410 $MA_AX_JERK_TIME

The jerk filter is only effective when the time constant is greater than a position-control cycle.

6.2.19 Kneeshaped acceleration characteristic curve

6.2.19.1 Function: Adaptation to the motor characteristic curve

Various motor types, particularly stepper motors, have a torque characteristic that is highly dependent on the speed with a steep decline in the torque in the upper speed range. For optimal utilization of the motor characteristic, the acceleration of the associated NC axis must be reduced as of a specific speed.



①	Normal range
②	Reducing range
n_{red}	Speed as of which traversing is done with reduced torque
n_{max}	Maximum speed
M_{max}	Maximum torque
M_{red}	Torque at n_{max} (corresponds to acceleration reduction)

Characteristic curve types

As a controller-internal emulation of the torque characteristic curve of the motor, the following characteristic curve types can be selected for the reducing range (see chapter "Parameterization (Page 301)"):

- Constant torque characteristic
- Hyperbolic torque characteristic
- Linear torque characteristic

6.2.19.2 Function: Effects on the path acceleration

The path acceleration characteristic curve results from the types of characteristic curve 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 where 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

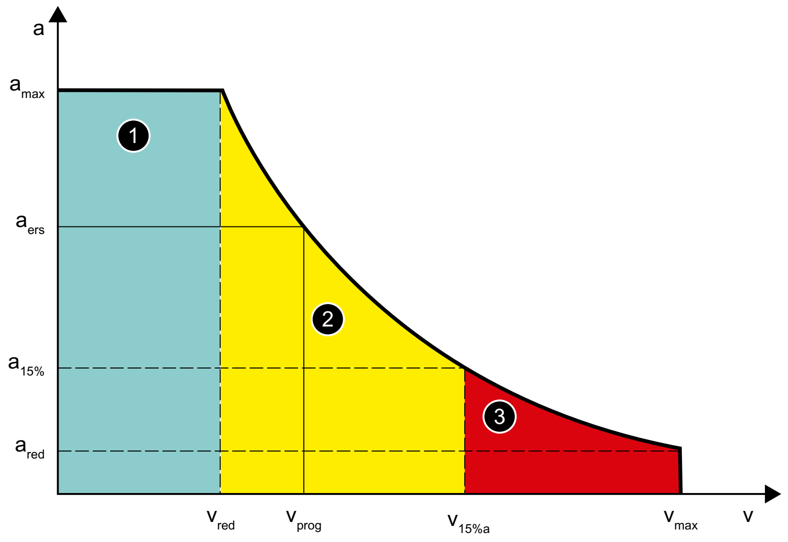
Machine axes featuring stepper motor and DC drive can be interpolated together.

6.2.19.3 Function: Substitute curve

If the programmed path cannot be traversed with the acceleration curve parameterized in the machine data of the participating axes, e.g. if kinematic transformation and acceleration types `BRISK` or `SOFT` are active, a substitute curve is generated by reducing the dynamic limit values. The reduced 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.

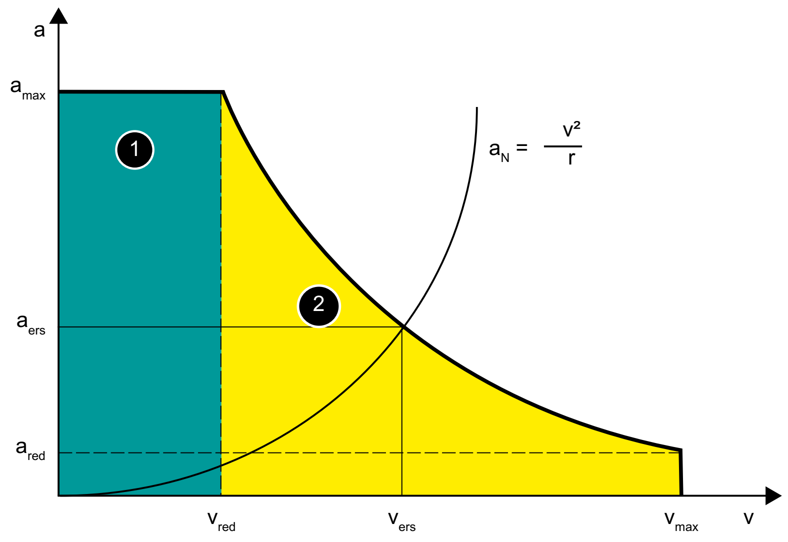


①	Normal range
②	Reducing range
③	Locked area
a_{ers}:	Substitute characteristic curve constant acceleration
$a_{15\%}$	Minimal constant acceleration
	$a_{15\%} = 0.15 * (a_{max} - a_{red}) + a_{red}$
v_{ers}	Substitute characteristic curve velocity
v_{prog}	Programmed velocity
$v_{15\%a}$	Velocity at $a_{15\%}$

Figure 6-6 Substitute path characteristic curve: Linear path

Substitute characteristic curve with curved path sections

In the case of curved path sections, normal and tangential acceleration are considered together. The path velocity is reduced so that only up to 25 % of the speed-dependent acceleration capacity of the axes is required for normal acceleration. The remaining 75 % of the acceleration capacity is set aside for the tangential acceleration, i.e., deceleration/ acceleration on the path.



①	Normal range
②	Reducing range
a_N	Normal acceleration
a_{ers}:	Substitute characteristic curve constant acceleration
v_{ers}	Substitute characteristic curve velocity
r	Path radius

Figure 6-7 Substitute path characteristic curve: Curved path

Block transitions with continuous-path mode

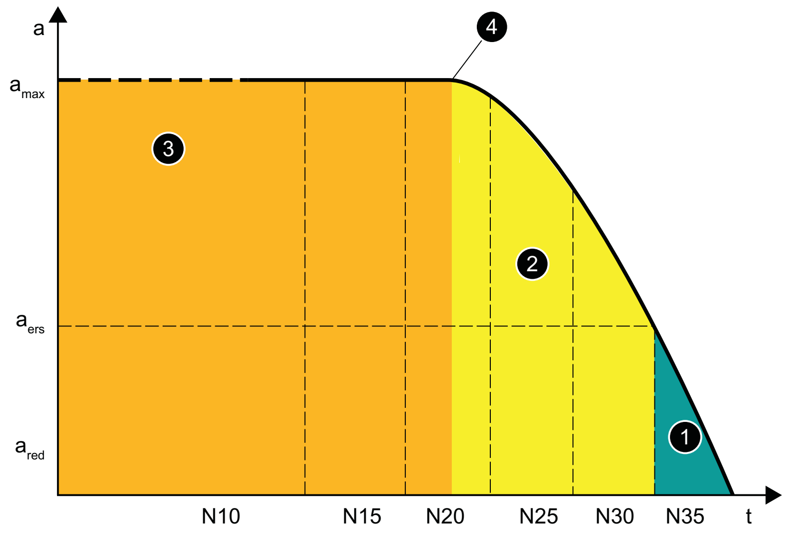
If continuous-path mode is active, non-tangential block transitions result in axial velocity jumps when the programmed path velocity is used for traversing.

As a result, the path velocity is controlled in such a way that prevents any axial velocity proportion from exceeding the creep velocity v_{red} at the time of the block transition.

Deceleration ramp with continuous-path mode and LookAhead

In the case of consecutive traversing blocks with short paths, an acceleration or deceleration operation may be spread over several part program blocks.

In such a situation the "LookAhead" function also takes into account the parameterized speed-dependent acceleration characteristic.



①	Normal range ⇒ a = a_{max}
②	Reducing range ⇒ a < a_{max}
③	Constant travel range ⇒ a = 0 m/s^2
④	Brake application point
v_{red}:	Creep velocity
v_{max}	Maximum velocity
Nx	Traversing block with block number Nx

Figure 6-8 Deceleration with LookAhead

6.2.19.4 Parameterization

Path axis

Parameter assignment

The parameterization of the axis-specific velocity limit, above which the reduced acceleration acts as an axis for traversing movements, is done via the machine data:

MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT = <speed limit>

Single axis

Activation

The activation of the axis-specific velocity limit, above which the reduced acceleration acts as a single axis for traversing movements, is done via machine data:

MD35240 $MA_ACCEL_TYPE_DRIVE = TRUE

Parameter assignment

The following machine data is relevant for parameterizing the axis-specific acceleration characteristic curve above the configured velocity limit:

- MD32000 $MA_MAX_AX_VELO (maximum axis velocity)
- MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT = <speed limit>
  From the parameterized velocity, the reduced acceleration is in effect>
- MD35230 $MA_ACCEL_REDUCTION_FACTOR = <reduction factor>
  Beyond the reduction speed, interpolation between the maximum and minimum acceleration occurs up to the maximum speed. The minimum acceleration is calculated as: minimum acceleration = maximum acceleration * (1 - reduction factor)
- MD35242 $MA_ACCEL_REDUCTION_TYPE (see paragraph below: "Selection of the torque characteristic curve")
- MD32300 $MA_MAX_AX_ACCEL (Maximum axis acceleration)

Selection of the torque characteristic curve

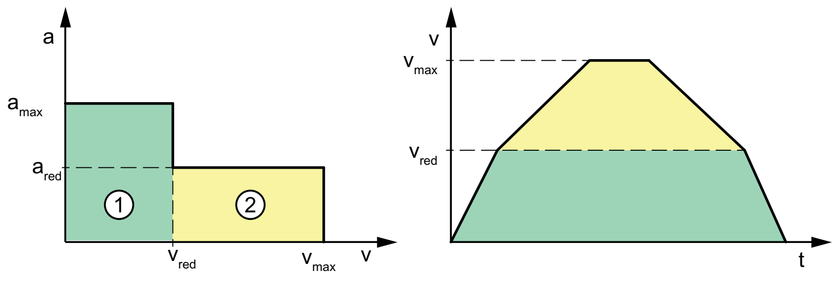
The following characteristic curve types can be selected for the reducing range via the machine data as a controller-internal emulation of the torque characteristic curve of the motor:

MD35242 $MA_ACCEL_REDUCTION_TYPE = <value>

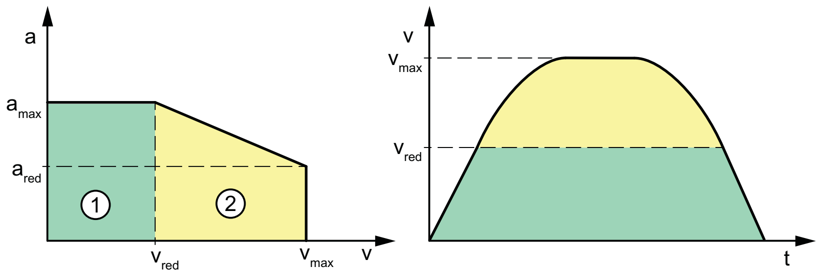
Value	Meaning
0	Constant torque characteristic
1	Hyperbolic torque characteristic
2	Linear torque characteristic

The following figures show the principal velocity and acceleration characteristic curves for the respective types of characteristic curve:

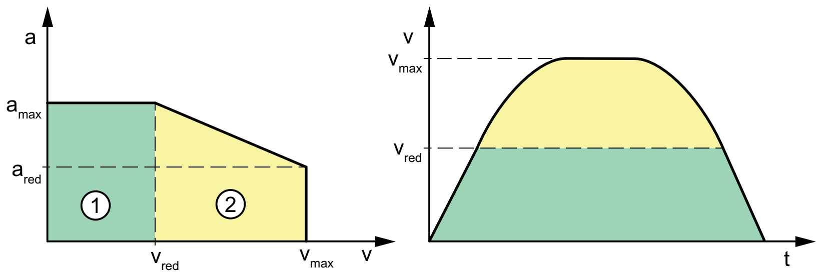
- Constant torque characteristic (value = 0)



- Hyperbolic torque characteristic (value = 1)



- Linear torque characteristic (value = 2)



Characteristic parameters

The characteristic curve parameters result from the following machine data:

①	Normal range
②	Reducing range
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

6.2.19.5 Programming: Channel-specific activation (DRIVE)

Functionality

The axis-specific velocity limit, above which the reduced acceleration acts as an axis for traversing movements, is activated with the command `DRIVE`.

Syntax

```
DRIVE
```

Meaning

`DRIVE`:	Channel-specific switch-on of the reduced acceleration above the parameterized velocity limit	
	G group:	21
	Basic position:	MD20150 $MC_GCODE_RESET_VALUES[20]
	Effectiveness:	Modal

Supplementary conditions

Characteristic changeover

If the acceleration reduction is active due to the programming of `DRIVE` and the acceleration profile is changed over by `SOFT` or `BRISK`, traversing is then done with a substitute characteristic with reduced dynamic limit values.

With the re-programming of `DRIVE`, the reduction in acceleration can be re-activated beyond the parameterized velocity limit.

See also

Parameterization (Page 301)

6.2.19.6 Programming: Axis-specific activation (DRIVEA)

Functionality

The axis-specific velocity limit, above which the reduced acceleration acts as a single axis for traversing movements, is activated for the programmed axes with the pre-defined procedure `DRIVEA()`.

Syntax

```
DRIVEA(<Axis_1>, <Axis_2>, ...)
```

Meaning

`DRIVEA`:	Axis-specific activation of the parameterized knee-shaped acceleration characteristic curve.	
	Effectiveness:	Modal
`<axis_x>`:	Axis for which the parameterized knee-shaped acceleration characteristic curve is to be activated.	
	Data type:	AXIS
	Value range:	Channel axis names

Supplementary conditions

If the knee-shaped acceleration characteristic curve is parameterized for an axis, then this becomes the default acceleration profile for traversing operations.

If the effective acceleration profile is changed for a specific axis using the `SOFTA` or `BRISKA` part-program commands, 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`.

6.2.19.7 Boundary conditions

Single axis interpolation

After activating the knee-shaped acceleration characteristic curve in case of single-axis interpolations (positioning axis, oscillating axis, manual traversing, etc.), the machine axis is traversed exclusively in the mode `DRIVEA` .

It is not possible to switch over the acceleration profile via the following part program instructions:

- Abrupt acceleration changes (`BRISKA`)
- Acceleration with jerk limitation (`SOFTA`)

Path interpolation

If for a machine axis involved in a programmed path, the knee-shaped acceleration characteristic curve parameterized without the part program instruction `DRIVE` is active, then a substitute characteristic curve with reduced dynamic limit values is determined for the path.

Kinematic transformation

The knee-shaped acceleration characteristic curve is not considered in an active kinematic transformation. With internal control, a switchover is done to acceleration without jerk limitation (BRISK) and a substitute characteristic curve becomes active for the path acceleration.

6.2.20 Acceleration and jerk for JOG motions

The axis-specific acceleration and jerk limitation values also take effect in JOG mode.

It is also possible to limit acceleration and jerk channel-specifically for manual traversing of geometry and orientation axes. This enables better handling of the kinematics that generate Cartesian motions entirely via rotary axes (robots).

6.2.20.1 Parameterization

Axis-specific limitation of acceleration and jerk

Maximum axis-specific acceleration

The maximum acceleration for manual traversing of an axis is defined in machine data:

MD32300 $MA_MAX_AX_ACCEL [0] (maximum axis-specific acceleration for path motions in dynamic response mode DYNNORM)

Note

Only the dynamic response mode DYNNORM is always effective for JOG mode.

See also Chapter "Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific) (Page 269)".

Basic setting of axis-specific jerk limitation

Acceleration with jerk limitation can be specified as the axis-specific basic setting for JOG mode:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE== TRUE

Maximum axis-specific jerk

For JOG motions with active jerk limitation, the value from the following machine data takes effect as the maximum axis-specific jerk:

MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axis-specific jerk)

See also Chapter "Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) (Page 286)".

Channel-specific limitation of acceleration and jerk

Maximum acceleration when manually traversing geometry axes

The maximum acceleration when manually traversing geometry axes can be specified for each channel via the machine data:

MD21166 $MC_JOG_ACCEL_GEO [<geometry axis>]

With <geometry axis> = 0, 1, 2

Note

With MD21166 $MC_JOG_ACCEL_GEO [<geometry axis>], there is no direct limitation to MD32300 $MA_MAX_AX_ACCEL.

Note

When a transformation is active, MD32300 $MA_MAX_AX_ACCEL determines the maximum possible axis-specific acceleration.

Maximum jerk when manually traversing geometry axes

The maximum jerk when manually traversing geometry axes in the SOFT acceleration mode (acceleration with jerk limitation) can be specified for each channel via the machine data:

MD21168 $MC_JOG_JERK_GEO [<geometry axis>]

With <geometry axis> = 0, 1, 2

Note

MD21168 $MC_JOG_JERK_GEO acts only when the axis-specific jerk limitation in JOG mode has been enabled for the base machine axes:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE

With MD21168 = 0, instead of the channel-specific jerk limitation, the axis-specific limit value from MD32430 $MA_JOG_AND_POS_MAX_JERK is effective.

Maximum jerk when manually traversing orientation axes

The maximum jerk when manually traversing orientation axes can be specified for each channel via the machine data:

MD21158 $MC_JOG_JERK_ORI [<orientation axis>]

For MD21158 to take effect, the channel-specific jerk limitation for the manual traversing of orientation axes must be enabled via the following machine data:

MD21159 $MC_JOG_JERK_ORI_ENABLE == TRUE

6.2.20.2 Supplementary conditions

Behavior for the manual traversing of geometry axes with active rotation

When manually traversing geometry axes in the SOFT acceleration mode (acceleration with jerk limitation), the value from MD32430 $MA_JOG_AND_POS_MAX_JERK is also used with active rotation or active orientable tool carrier.

Part program instruction SOFTA / BRISKA / DRIVEA

The part program instruction `SOFTA(<axis1>,<axis2>, ...)` is also effective in JOG mode, i.e. the maximum axis-specific jerk from MD32430 $MA_JOG_AND_POS_MAX_JERK is effective for the specified axes when traversing in JOG mode (exactly as when setting MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE).

Note

On the other hand, the SOFT part program instruction has no effect on JOG mode.

As with SOFTA, the part program instructions BRISKA and DRIVEA are also effective in JOG mode, i.e. the acceleration is without jerk limitation, even when MD32420 $MA_JOG_AND_POS_JERK_ENABLE is set to "TRUE" for the relevant machine axes.

Note

Manual traversing of orientation axes is not affected by BRISKA/SOFTA/DRIVEA.

See also

Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific) (Page 269)

Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) (Page 286)

6.3 Examples

6.3.1 Acceleration

6.3.1.1 Path velocity characteristic

The following example shows the path velocity characteristic, based on programmed traversing motion and the actions initiated in a part program segment.

Part program extract

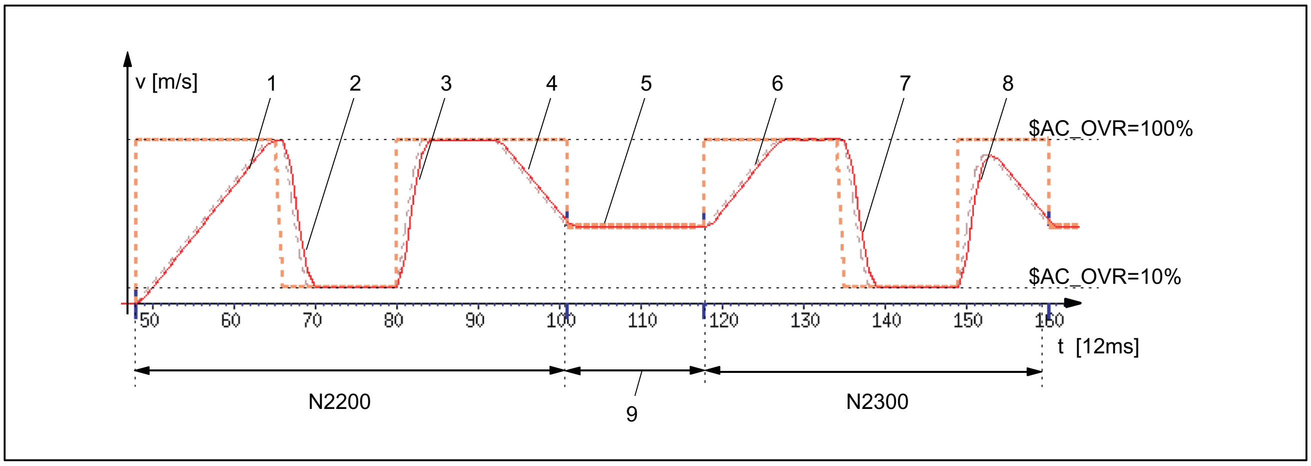
Program code

```
; Synchronized action: Acceleration switchover depending on fast input 1
($A_IN[1]):
N53 ID=1 WHEN $A_IN[1] == 1 DO $AC_PATHACC = 2.*$MA_MAX_AX_ACCEL[X]
; Synchronized action: Test override profile (simulation of external
interventions):
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
```

Program code

```
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
```

Path velocity characteristic



Acceleration profile: BRISK

1: Accelerate to 100% of path velocity (F10000) in accordance with acceleration default: ACC (N2200...)

2: Brake to 10% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N54...)

3: Accelerate to 100% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N55...)

4: Brake to block end velocity for intermediate smoothing block in accordance with acceleration default: ACC (N2200...)

5: Speed limitation as a result of smoothing (see 9)

6: Accelerate to 100% of path velocity ($AC_OVR) in accordance with acceleration default: ACC (N2300...)

7: Decelerate as a result of override modification at a rate of acceleration that is in accordance with real-time acceleration $AC_PATHACC (N53/N54...)

8: Accelerate to 100% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N55...)

9: Intermediate block inserted within the control as a result of the programmed smoothing (RNDM) (N2200...)

Figure 6-9 Switching between path acceleration specified during preprocessing and real-time acceleration

6.3.2 Jerk

6.3.2.1 Path velocity characteristic

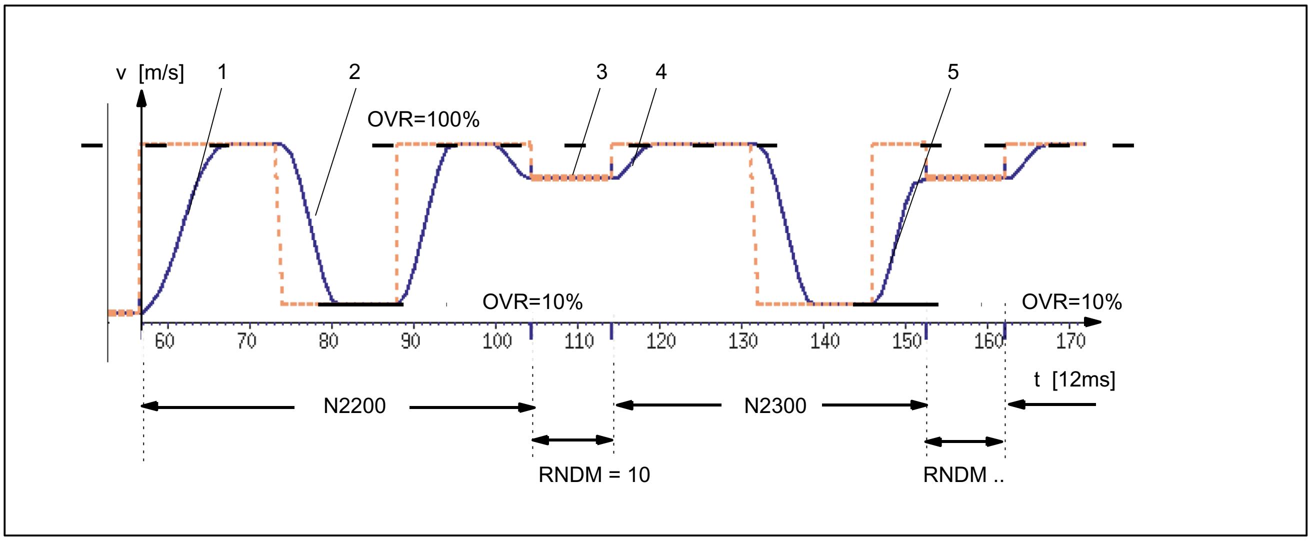
The following example shows the characteristic of path velocity and jerk based on programmed traversing motion and the actions initiated in a part program segment.

Part program extract

Program code

```
; Setting of path acceleration and path jerk in the event of external
intervention:
N0100 $AC_PATHACC = 0.0
N0200 $AC_PATHJERK = 4.0 * ($MA_MAX_AX_JERK[X] + $MA_MAX_AX_JERK[Y]) / 2.0
; Synchronized action:  varying the override (simulation of external
interventions)
N0530 ID=1 WHENEVER ($AC_TIMEC > 16) DO $AC_OVR=10
N0540 ID=2 WHENEVER ($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
```

Characteristic of path velocity and jerk



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 limiting using arcs
5:	Jerk according to $AC_PATHJERK

Figure 6-10 Switching between path jerk specified during preprocessing and $AC_PATHJERK

6.3.3 Acceleration and jerk

The following example shows the characteristic of velocity and acceleration of the X axis based on the programmed traversing motion of the part program extract. Further, which of the velocity and acceleration-relevant machine data are decisive for which section of the contour.

Part program extract

Program code	Comment
`N90 F5000 SOFT G64`	`; Continuous-path mode, jerk-limited acceleration`
`N100 G0 X0 Y0 Z0`	`; Rapid traverse`
`N110 G1 X10`	`; Straight line`
`N120 G3 CR=5 X15 Y5`	`; Circular arc, radius 5 mm, block transition: Tangential`
`N130 G3 CR=10 X5 Y15`	`; Circular arc, radius 10 mm, block transition: Tangential`
`N140 G1 X-5 Y17.679`	`; Straight line, 15° kink`

Contour

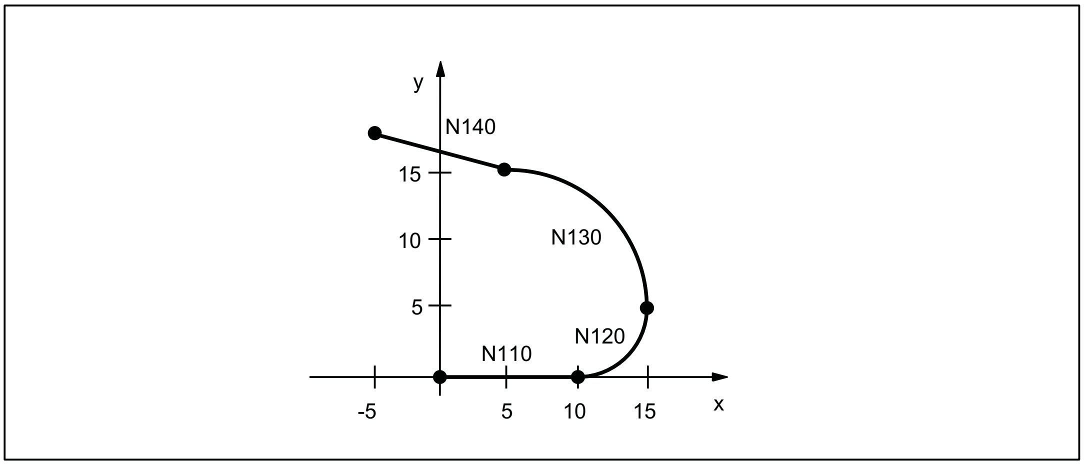


Figure 6-11 Contour of the part program extract

Velocity and acceleration characteristic

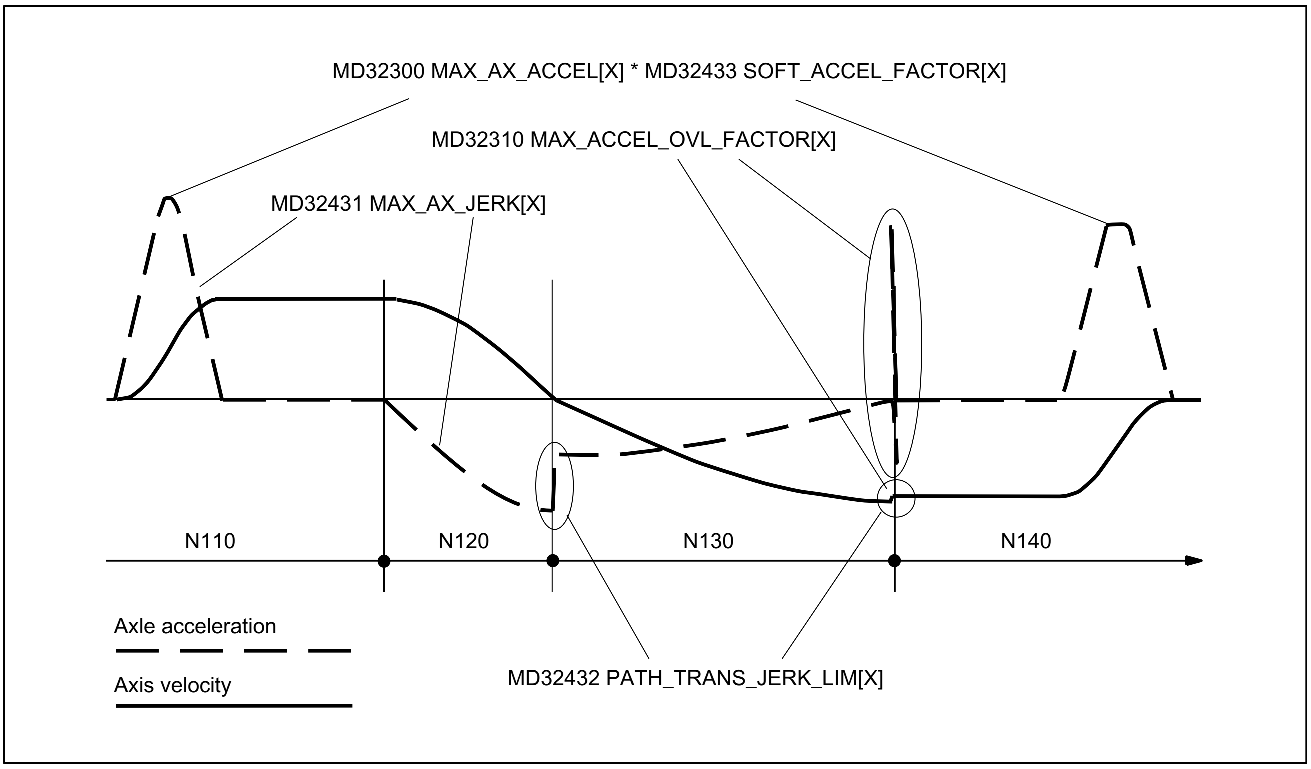


Figure 6-12 Velocity and acceleration characteristic curves X axis

6.3.4 Knee-shaped acceleration characteristic curve

6.3.4.1 Activation

The example illustrates how the knee-shaped acceleration characteristic curve is activated on the basis of the machine data and a part program extract.

Machine data

Machine data	**Value**
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
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
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 entering as channel-specific initial setting:

MC_GCODE_RESET_VALUE[20] = 3 (DRIVE)

Part program extract

Program code	**Comment**
N10 G1 X100 Y50 Z50 F700	; Path motion (X,Y, Z) with DRIVE
N15 Z20	; Path motion (Z) with DRIVE
N20 BRISK	; Switchover to BRISK
N25 G1 X120 Y70	; Path motion (Y, Z) with substitute characteristic curve
N30 Z100	; Path motion (Z) with BRISK
N35 POS[X] = 200 FA[X] = 500	; Positioning motion (X) with DRIVEA
N40 BRISKA(Z)	; Activate BRISKA for Z
N40 POS[Z] = 50 FA[Z] = 200	; Positioning motion (Z) with BRISKA
N45 DRIVEA(Z)	; Activate DRIVEA for Z
N50 POS[Z] = 100	; Positioning motion (Z) with DRIVE
N55 BRISKA(X)	; results in error message

6.4 Data lists

6.4.1 Machine data

6.4.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
18960	POS_DYN_MODE	Type of positioning axis dynamic response

6.4.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of the G groups
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity
20600	MAX_PATH_JERK	Path-related maximum jerk
20602	CURV_EFFECT_ON_PATH_ACCEL	Influence of path curvature on path dynamic response
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid motions
21158	JOG_JERK_ORI	Maximum jerk of the orientation axes when traversing in JOG
21159	JOG_JERK_ORI_ENABLE	Initial setting of the channel-specific jerk limitation of orientation axes for traversing in JOG mode
21166	JOG_ACCEL_GEO	Maximum acceleration rate of the geometry axes when traversing in JOG
21168	JOG_JERK_GEO	Maximum jerk of the geometry axes when traversing in JOG

6.4.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32000	MAX_AX_VELO	Maximum axis velocity
32300	MAX_AX_ACCEL	Maximum axis acceleration
32310	MAX_ACCEL_OVL_FACTOR	Overload factor for velocity jump
32320	DYN_LIMIT_RESET_MASK	Reset behavior of dynamic limits
32400	AX_JERK_ENABLE	Axial jerk limitation
32402	AX_JERK_MODE	Filter type for axial jerk limitation
32410	AX_JERK_TIME	Time constant for axial jerk filter
32420	JOG_AND_POS_JERK_ENABLE	Basic setting for axial jerk limitation
32430	JOG_AND_POS_MAX_JERK	Axial jerk for single axis motion
32431	MAX_AX_JERK	Maximum axial jerk at the block change in continuous-path mode

Number	Identifier: $MA_	Description
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 G0
32435	G00_JERK_FACTOR	Scaling of axial jerk limitation for G0
32437	AX_JERK_VEL0	First velocity threshold for velocity-dependent jerk adaptation
32438	AX_JERK_VEL1	Second velocity threshold for the velocity-dependent jerk adaptation
32439	MAX_AX_JERK_FACTOR	Factor to set the maximum jerk for higher velocities (velocity-dependent jerk adaptation)
35220	ACCEL_REDUCTION_SPEED_POINT	Speed for reduced acceleration
35230	ACCEL_REDUCTION_FACTOR	Reduced acceleration
35240	ACCEL_TYPE_DRIVE	DRIVE acceleration characteristic for axes on/off
35242	ACCEL_REDUCTION_TYPE	Type of acceleration reduction

6.4.2 Setting data

6.4.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42500	SD_MAX_PATH_ACCEL	Max. path acceleration
42502	IS_SD_MAX_PATH_ACCEL	Analysis of SD 42500: ON/OFF
42510	SD_MAX_PATH_JERK	Max. path-related jerk
42512	IS_SD_MAX_PATH_JERK	Analysis of SD 42510: ON/OFF

6.4.3 System variables

Identifier	Description
$AC_PATHACC	Path acceleration for real-time events
$AC_PATHJERK	Path jerk for real-time events

F1: Travel to fixed stop

7

7.1 Brief description

Function

With the "Travel to fixed stop" function, moving machine parts, e.g. tailstock or sleeve, can be traversed so that they can apply a defined torque or force with respect to other machine parts over any time period.

Characteristics

- Programmability using commands in the part program or synchronized action:
  - Activating and deactivating the "Travel to fixed stop" function
  - Setting the clamping torque
  - Setting the monitoring window
- Manual intervention option using setting data:
  - Activating and deactivating the "Travel to fixed stop" function
  - Setting the clamping torque
  - Setting the monitoring window
- Presettings using machine data
  - Clamping torque
  - Monitoring window
- Identifying the function status via NC/PLC interface signals
- Enable or acknowledgment options via NC/PLC interface signals
- Reading the reference and actual status of the function via system variable
- "Travel to fixed stop" is possible for axes and spindles.
- "Travel to fixed stop" is possible simultaneously for several axes and parallel to the traversing of other axes.
- Multi-channel block search can be performed with calculation of all the required additional data (SERUPRO).
- Simulated axis traversal in conjunction with "travel to fixed stop" and "torque reduction" possible.
- "Vertical" axes can also be moved with FXS to a fixed stop.

7.2 Detailed description

7.2.1 Programming

Function

Travel to fixed stop

The "Travel to fixed stop" function is controlled via the `FXS`, `FXST` and `FXSW` commands.

The activation can also be performed without traversing motion of the relevant axis. The torque is immediately limited. The fixed stop is monitored as soon as the axis is traversed.

Note

Synchronized actions

The "Travel to fixed stop" function can also be controlled via synchronized actions.

References:
Function Manual, Synchronized Actions

Travel with limited torque/force

Travel with limited torque/force can be controlled via the `FOCON`, `FOCOF` and `FOC` commands (see Section "Travel with limited torque/force FOC (Page 335)").

Syntax

```
FXS[<axis>]=<request>
FXST[<axis>]=<clamping torque>
FXSW[<axis>] = <window width>
```

Meaning

Parameter	Meaning
`FXS`:	"Travel to fixed stop" function, effectiveness: Modal
`<Request>`:	0 = switch off through 1 = switch on
`FXST`:	Set clamping torque
`<Clamping torque>`:	Clamping torque in % of the maximum drive torque. SINAMICS S120: p2003
`FXSW`:	Set monitoring window
`<Window width>`:	Width of the tolerance window around the fixed stop Unit: mm, inch or degrees
`<axis>`:	Name of the channel axis, type: AXIS

Changes to the torque limiting (FXST)

The torque limit value can be changed in each block. The change becomes effective before executing the traversing motion programmed in the block. The torque limitation acts in addition to the acceleration limitation (`ACC`).

Ramp-shaped change

A time can be set using the machine data, within which the torque limit value is linearly changed.

MD37012 $MA_FIXED_STOP_TORQUE_RAMP_TIME (time until the new torque limit is reached)

Changes to the monitoring window (FXSW)

The monitoring window can be changed in each block. The change becomes effective before executing the traversing motion programmed in the block.

When the monitoring window is changed, not only does the window width change, but also the reference point of the window to the actual axis position.

"Travel to fixed stop" when the continuous-path mode is active (G64)

The following machine data can be used to set that with the selection of the (`FXS`) function during active continuous-path mode (`G64`), no exact stop is triggered at the block change (`G60`):

MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgements for travel to fixed stop)

Bit	Value	Meaning
0	0	Start of the traversing motion without acknowledgement by the PLC
	1	Start of the traversing motion after acknowledgement by the PLC

Supplementary conditions

- The traversing motion to the fixed stop can be programmed as a path- or block-related or modal positioning axis motion.
- Travel to fixed stop can be be selected for several machine axes simultaneously.
- For a machine axis, which is traversed to a fixed stop, it is not permissible that transformation, coupling and frame functions are active:
- The travel path and the activation of the function must be programmed in one block in the part program.
- If "Travel to fixed stop" is activated via synchronized actions, the travel path and the activation of the function can be programmed in separate blocks.

7.2.2 Functional sequence

7.2.2.1 Selection

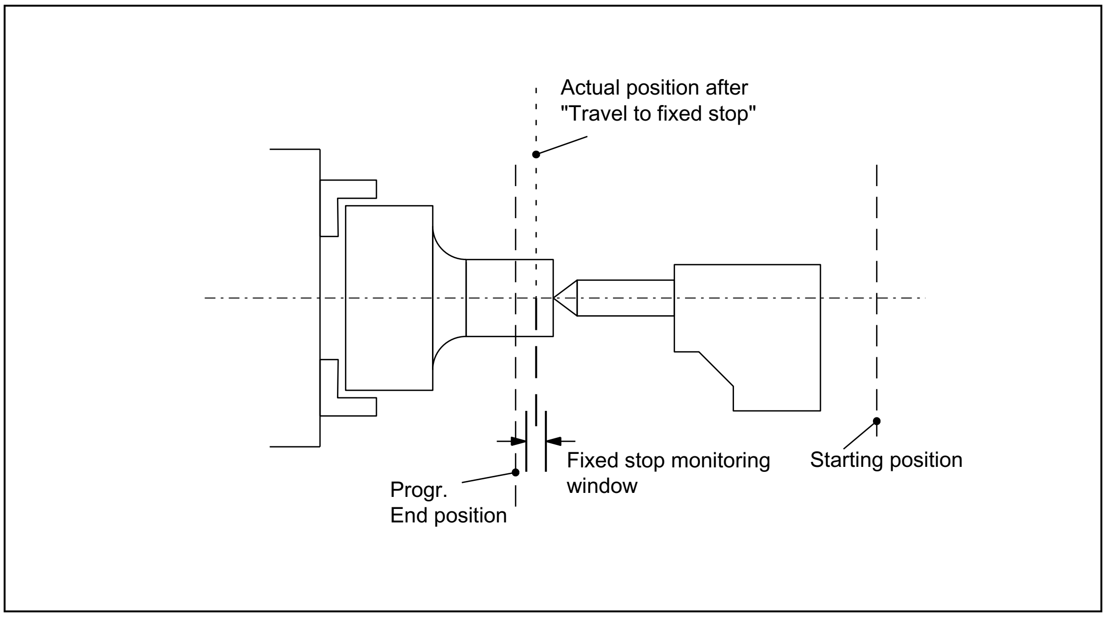

Actual position after "Travel to fixed stop"
Fixed stop monitoring window
Starting position
Progr. End position

Figure 7-1 Example of travel to fixed stop

Procedure

The NC detects that the function "Travel to fixed stop" is selected via the command `FXS[x]=1` and signals the PLC using the IS DB31, ... DBX62.4 ("Activate travel to fixed stop") that the function has been selected.

If the machine data:

MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgements for travel to fixed stop)

is set correspondingly, the system waits for the acknowledgement of the PLC using the IS DB31, ... DBX3.1 ("Enable travel to fixed stop").

The programmed target position is then approached from the start position at the programmed velocity. The fixed stop must be located between the start and target positions of the axis/spindle. A programmed torque limit (clamping torque specified via `FXST[<axis>]`) is effective from the start of the block, i.e. the fixed stop is also approached with reduced torque. Allowance for this limitation is made in the NC through an automatic reduction in the acceleration rate.

If no torque has been programmed in the block or since the start of the program, then the value is valid in the axis-specific machine data:

MD37010 $MA_FIXED_STOP_TORQUE_DEF (default for fixed stop clamping torque)

is entered.

7.2.2.2 Fixed stop is reached

Detecting the fixed stop

Detecting the fixed stop or identifying that the machine axis has reached the fixed stop can be set using the following machine data:

MD37040 $MA_FIXED_STOP_BY_SENSOR = <value> (fixed stop detection via sensor)

<value>	Meaning
0	The fixed stop has been reached when the contour deviation of the machine axis has exceeded the value set in the machine data: MD37030 $MA_FIXED_STOP_THRESHOLD (threshold for fixed stop detection)
1	An external sensor detects when the fixed stop has been reached and informs the control via the following axial NC/PLC interface signal: DB31, ... DBX1.2 == 1 (sensor for fixed stop)
2	The fixed stop has been reached, if one of the conditions, specified under <value> == 0 OR <value> == 1 applies.

Ineffective NC/PLC interface signals

When the axis is in the "Fixed stop reached" state, the following NC/PLC interface signals have no effect:

- DB31, ... DBX1.3 (axis/spindle disable)
- DB31, ... DBX2.1 (controller enable)

Actions when the fixed stop is reached

The following actions are executed when the fixed stop is reached:

- The torque in the drive is increased up to the programmed clamping torque (`FXST`)
- The remaining distance to go is deleted
- The position setpoint is tracked
- The NC/PLC interface signal is set: DB31, ... DBX62.5 = 1 ("fixed stop reached")
- Executing a block change:
  - Executing a block change when the fixed stop is reached: MD37060 $MA_FIXED_STOP_ACKN_MASK, Bit 1 = 0
    The block change is immediately executed after reaching the fixed stop.
  - Block change is only executed after the acknowledgment by the PLC user program: MD37060 $MA_FIXED_STOP_ACKN_MASK, Bit 1 = 1
    After reaching the fixed stop, the block is not changed until acknowledgment by the PLC user program:
    DB31, ... DBX1.1 == 1 (acknowledge fixed stop reached)

  In order to also maintain the clamping torque after the block change, the NC still outputs a setpoint for the machine axis.
- Activating the fixed stop monitoring or the monitoring window

Monitoring window

If, in the traversing block to the fixed stop or since the beginning of the program, no specific value for the monitoring window is programmed with `FXSW` then the value set in the machine data is active:

MD37020 $MA_FIXED_STOP_WINDOW_DEF (default for fixed stop monitoring window)

If the axis leaves the position it was in when the fixed stop was detected by more than the specified window, then alarm 20093 "Fixed stop monitoring has responded" is displayed and the "Travel to fixed stop" function is deselected.

The monitoring window must be selected by the user such that the alarm is activated only when the axis leaves the fixed stop position.

NOTICE

"Fixed stop reached" and when the fixed stop breaks

As soon as the "fixed stop reached" state is identified, then a speed setpoint derived from the axis-specific machine data servo gain factor (K_V) (MD32200) and the threshold for the fixed stop detection (MD37030) is entered for the drive. If the fixed stop breaks in this state, then the axis accelerates until it reaches the monitoring window limit. The velocity that is then reached is proportional to the values set in the specified machine data. For appropriately high values, it is possible that the drive accelerates up to the maximum motor speed.

Overview

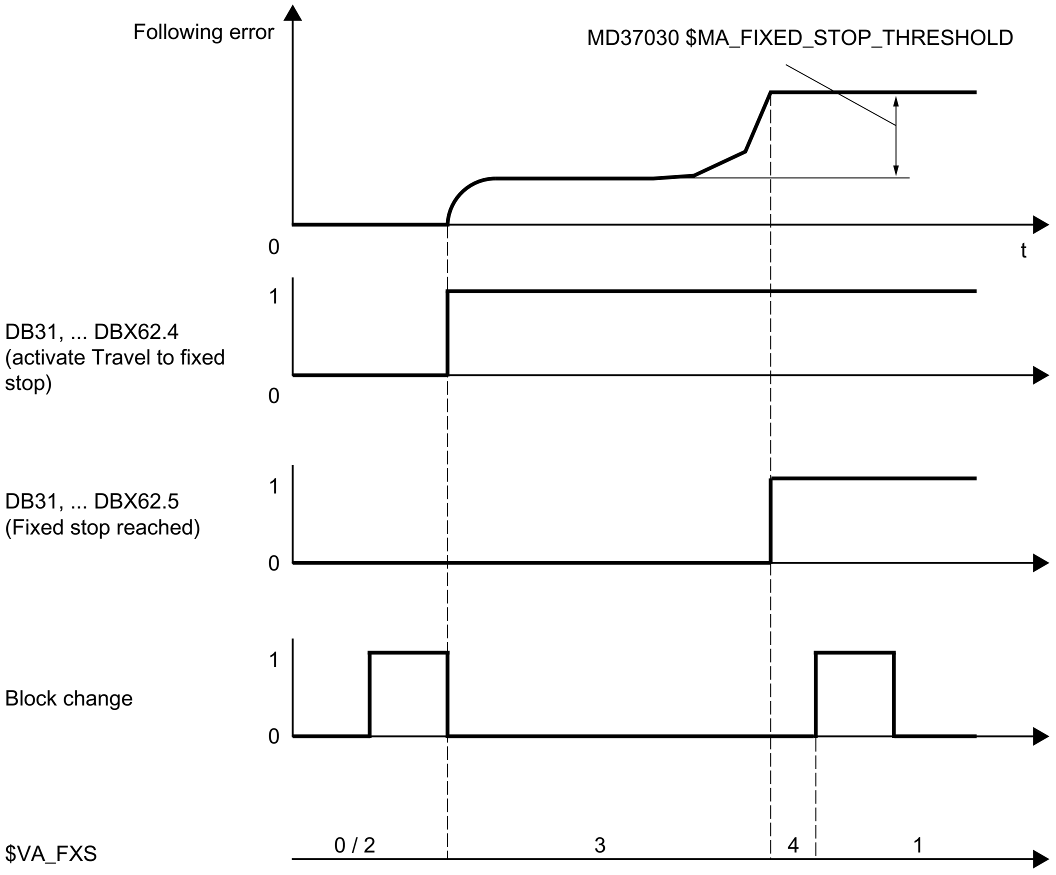

Following error
MD37030 $MA_FIXED_STOP_THRESHOLD
0
t
1
DB31, ... DBX62.4 (activate Travel to fixed stop)
0
1
DB31, ... DBX62.5 (Fixed stop reached)
0
1
Block change
0
$VA_FXS
0 / 2
3
4
1

Figure 7-2 Fixed stop is reached

7.2.2.3 Fixed stop is not reached

Alarm suppression

Alarms for various causes of breakage can be suppressed using the machine data:

MD37050 $MA_FIXED_STOP_ALARM_MASK = <value>

Value	Description: Suppressed alarms
0	Alarm 20091 "Fixed stop not reached"
2	Alarm 20091 "Fixed stop not reached" Alarm 20094 "Fixed stop aborted"
3	Alarm 20094 "Fixed stop aborted"

Actions in the case of a fault or breakage

The following actions are executed when a fault occurs or for a breakage:

- The NC/PLC interface signal is reset: DB31, ... DBX62.4 = 0 (activate travel to fixed stop)
- Depending on the setting in the machine data, the system waits for acknowledgment:
  - MD37060 $MA_FIXED_STOP_ACKN_MASK
  - DB31, ... DBX3.1 == 0 (enable travel to fixed stop)
- Withdrawing torque limits
- Executing a block change

Overview

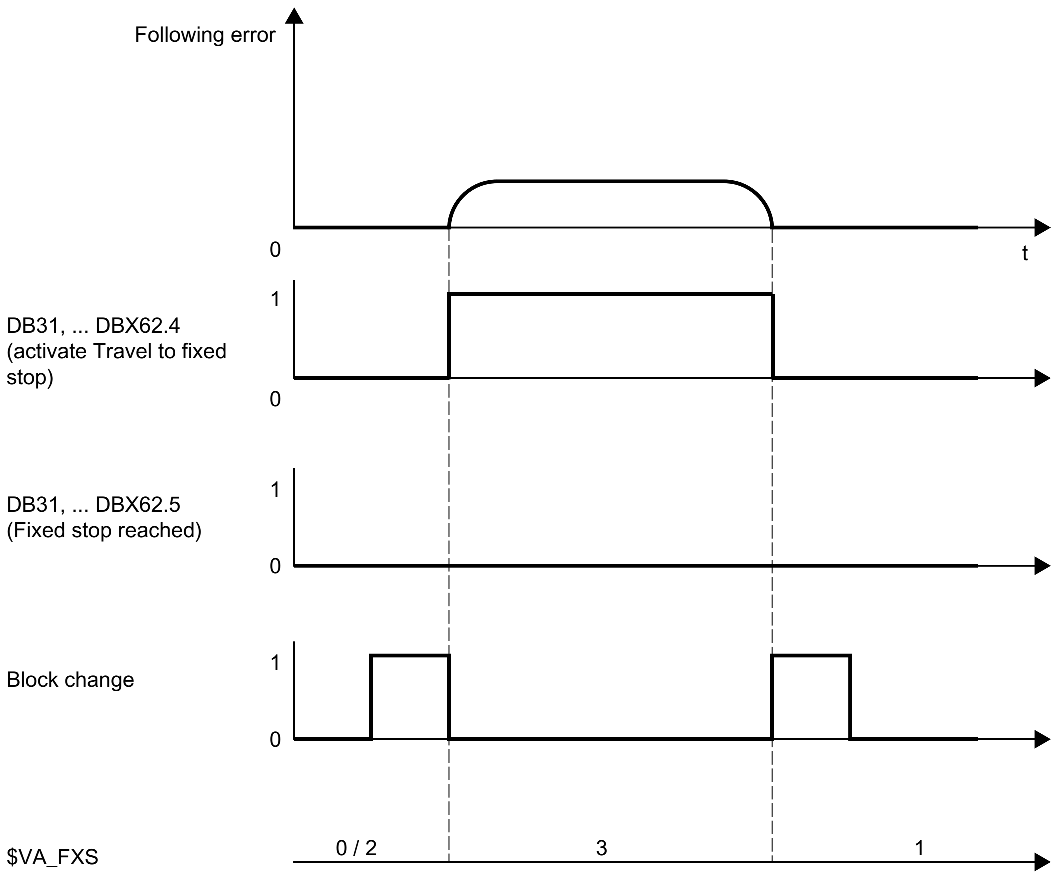

Following error
0
t
DB31, ... DBX62.4 (activate Travel to fixed stop)
1
0
DB31, ... DBX62.5 (Fixed stop reached)
1
0
Block change
1
0
$VA_FXS
0 / 2
3
1

Figure 7-3 Fixed stop is not reached

7.2.2.4 Deselection

The "travel to fixed stop" function is deselected using the command `FXS[<axis>] = 0` in a block of an NC program.

Actions when deselecting the function

When deselecting the function, the following actions are executed:

- A preprocessing stop is initiated (`STOPRE`)
- The NC/PLC interface signals are reset
  - DB31, ... DBX62.4 = 0 (activate travel to fixed stop)
  - DB31, ... DBX62.5 = 0 (fixed stop reached)
- Depending on the machine data, the system waits for acknowledgment from the PLC user program:
  - MD37060 $MA_FIXED_STOP_ACKN_MASK
  - DB31, ... DBX3.1 == 0 (enable travel to fixed stop)
  - DB31, ... DBX1.1 == 0 (acknowledge fixed stop reached)
- Exiting the follow-up mode
- Axis resumes closed-loop position control

Pulse enable

The pulse enable or pulse inhibit can be canceled via:

- Drive: Via terminal EP (enable pulses)
- NC/PLC interface signal: DB31, ... DBX21.7 ("pulse enable")

The behavior at the fixed stop can be set via the following machine data:

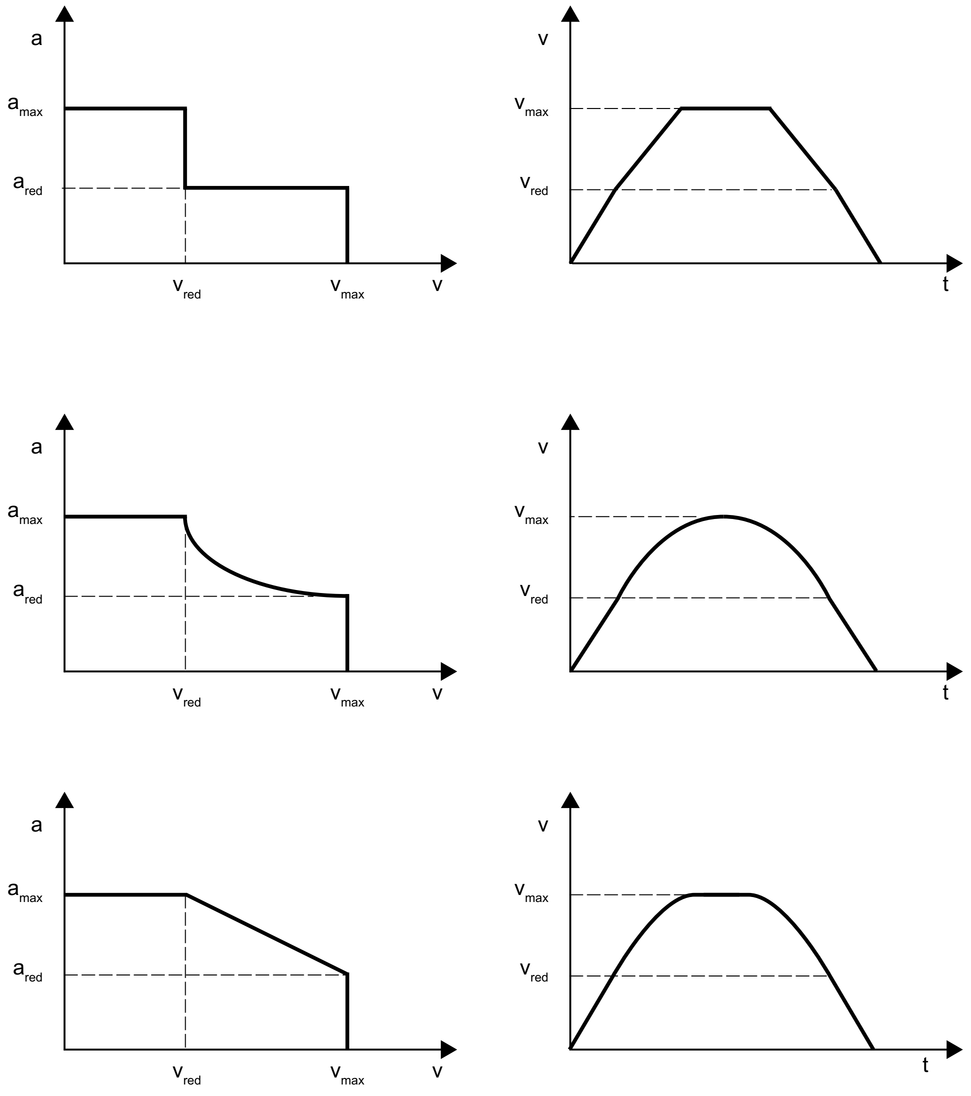
MD37002 $MA_FIXED_STOP_CONTROL, bit 0 and bit 1 (sequence control for traversing to fix stop)

Bit	Value	Meaning
0	Behavior for pulse**inhibit** at the fixed stop	
	0	"Travel to fixed stop" is aborted
	1	"Travel to fixed stop" is interrupted, i.e. the drive is without power.
1	Behavior for pulse**enable** at the fixed stop	
	0	The torque is injected suddenly.
	1	The torque is linearly ramped-up over the time specified in the machine data: MD37012 $MA_FIXED_STOP_TORQUE_RAMP_TIME

Special case: Deleting the pulse enable during deselection

If, during deselection, the function in state: "Waiting for PLC acknowledgements" the pulse enable is deleted, the torque limit is reduced to 0. In this phase, if pulse enable is set again torque is no longer established in the drive. Once the deselection has been completed the axis can be traversed normally again.

Overview



① Traversing block with deselection `FXS[<axis>]=0`

Figure 7-4 Fixed stop deselection

7.2.3 Behavior during block search

Function

Block search with calculation

- If the target block is located in a program section in which the axis must stop at a fixed limit, then the fixed stop is approached if it has not yet been reached.
- If the target block is located in the program section in which the axis must not stop at a fixed limit, then the axis leaves the fixed stop if it is still positioned there.
- If the axis is in the "Fixed stop reached" state, message 10208 "Press NC start to continue the program" is displayed. The program can be continued with NC start.
- Clamping torque `FXST` and monitoring window `FXSW` have the value that they have for normal program processing at the start of the target block.

Block search without calculation

The `FXS`, `FXST` and `FXSW` commands are ignored during the block search.

Effectiveness of FOCON/FOCOF

The state of the modal-acting torque/force reduction `FOCON`/`FOCOF` is maintained during the block search and is effective in the approach block.

Block search with FXS or FOC

The user selects `FXS` or `FOC` in a program area of the target block in order to acquire all states and functions of the machining last valid. The NC starts the selected program in Program test mode automatically. After the target block has been found, the NC stops at the beginning of the target block, deselects Program test internally again and displays the Stop condition "Search target found" in its block display.

CAUTION
SERUPRO approach does not really take the FXS command into account.
The approach to the programmed end position of the `FXS` block is only simulated **without torque limitation**.

If `FXS` is located between the beginning of the program and the search target, the command is not executed by the NC. The motion is only simulated up to the programmed end point.

The user can log the turning on and turning off of `FXS` in the part program. If necessary, the user can start an ASUP in order to activate or deactivate `FXS` in this SERUPRO-ASUP.

System variable

The reference and actual state of the "travel to fixed stop" function can be read using the following system variable:

- $AA_FXS (reference state)
- $VA_FXS (actual state)

SERUPRO: $AA_FXS (reference state)

During SERUPRO, $AA_FXS supplies the following values depending on the activation status of the "Travel to fixed stop" function:

Activation status of the "travel to fixed stop" function	System variable $AA_FXS ==
"Deactivated"	0 (axis not at fixed stop)
"Activated"	3 ("Travel to fixed stop" selection is active)

Note

During SERUPRO, the system variable $AA_FXS only supplies the values 0 and 3. As a result, based on $AA_FXS, the program sequence can be changed with SERUPRO compared to the normal program execution for program branches.

SERUPRO: $VA_FXS (actual state)

During SERUPRO, the variable $VA_FXS always supplies the real state of axis on the machine.

Example

The current state of the "Travel to fixed stop" function can be determined in the SERUPRO ASUP via the system variables $AA_FXS and $VA_FXS, and the appropriate response initiated:

Program code: FXS_SERUPRO_ASUP.MPF	Comment
N100 WHEN ($AA_FXS[X]==3) AND ($VA_FXS[X]==0) DO FXS[X]=1	; Reference=="Selection not active" AND ; Actual=="Axis not at fixed stop" ; => "Switch on"
N200 WHEN ($AA_FXS[X]==0) AND ($VA_FXS[X]==1) DO FXS[X]=0	; Reference=="Axis not at fixed stop" AND ; Actual=="Fixed stop successfully approached" ; => "Switch on"
N1020 REPOSA	; Reapproaching the contour linearly with all axes

Displaying the REPOS offset

Once the search target has been found, for each axis, the actual state regarding the "Travel to fixed stop" is displayed using the following NC/PLC interface signals:

- DB31, ... DBX62.4 (activate travel to fixed stop)
- DB31, ... DBX62.5 (fixed stop reached)

Example:

If the axis is at the fixed stop and the target block is available after deselection of `FXS`, the new target position is displayed via DB31, ... DBX62.5 (fixed stop reached) as the `REPOS` offset.

REPOS and FXS

With `REPOS`, the functionality of `FXS` is repeated automatically and called FXS-REPOS in the following. This sequence is comparable to the FXS_SERUPRO_ASUP.MPF program. 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 executed in the SERUPRO ASUP automatically takes care of

$AA_FXS[<axis>] = $VA_FXS[<axis>].

This deactivates FXS-REPOS for axis X.

Deactivating FXS-REPOS

FXS-REPOS is deactivated by:

- An `FXS` synchronized action which refers to `REPOSA`
- $AA_FXS[X] = $VA_FXS[X] in the SERUPRO_ASUP

Note

A SERUPRO ASUP **without** `FXS` treatment or no SERUPRO ASUP results automatically in FXS-REPOS.

⚠ CAUTION

Speed too high for FXS-REPOS

`FXS-REPOS` traverse all path axes together to the target position. Axes with and without `FXS` treatment thus traverse together with the G command and feedrate valid in the target block. As a result, the fixed stop may be approached in rapid traverse(`G0`) or higher velocity.

FOC-REPOS

FOC-REPOS behaves in the same way as FXS-REPOS.

A changing torque characteristic during the program preprocessing cannot be implemented with `FOC-REPOS`.

Example

Axis X is traversed from position 0 to 100. `FOC` is switched on every 20 millimeters for 10 millimeters. The resulting torque characteristic is generated with non-modal `FOC` and cannot be traced by FOC-REPOS. Axis X is traversed by FOC-REPOS with or without `FOC` in accordance with the last programming before the target block.

For programming examples of `FXS` "Travel to fixed stop", see Section "Program test (Page 495)".

7.2.4 Behavior for reset and function abort

NC reset

As long as the function is still not in the "Successful travel to fixed stop" state, the travel to fixed stop can be aborted with NC reset.

Even when the fixed stop has already has been approached, but the specified stop torque not yet fully reached, then the function can still be aborted with NC reset. The position setpoint of the axis is synchronized with the current actual position.

As soon as the function is in the "Successful travel to fixed stop" state, the function also remains active after the NC reset.

Function abort

A function abort can be triggered by the following events:

- Emergency stop

> **CAUTION**
>
> **Dangerous machine situations possible for travel to limit stop**
>
> It must be ensured that no dangerous machine situations occur while travel to fixed stop is active when an "Emergency stop" is triggered or reset.
>
> For example, behavior when setting and canceling the pulse enable:
> MD37002 $MA_FIXED_STOP_CONTROL, bit 0 (behavior for pulse disable at the stop)
>
> - Bit 0 = 0: Travel to fixed stop is aborted
> - Bit 0 = 1: Travel to fixed stop is interrupted, i.e. the drive is without power
>   Once the pulse disable is canceled again, the drive presses with the stop torque again.

> **Note**
>
> NC and drive have no power during "Emergency stop", i.e. the PLC must react.

- Functional state: "Fixed stop not reached"
- Functional state: "Fixed stop aborted"
- Aborted by the PLC user program:
  DB31, ... DBX62.4 = 0 ("Activate travel to fixed stop")
- Cancellation of the pulse enable and machine data parameterization:
  MD37002 $MA_FIXED_STOP_CONTROL, bit 0 = 0 (see above)

7.2.5 Behavior with regard to other functions

Measurement with delete distance-to-go

"Travel to fixed stop" (`FXS`) cannot be programmed in a block with "Measurement with delete distance-to-go" (`MEAS`). Except when on function acts on a path axis while the other acts on a positioning axis or both functions act on positioning axes.

Contour monitoring

The axis contour monitoring function is inoperative while "Travel to fixed stop" is active.

Positioning axes

For "Travel to fixed stop" with positioning axes `POSA`, the block change is executed even when the positioning axis has not yet reached the fixed stop by this time.

Vertical axes

The "Travel to fixed stop" function can be used for vertical axes even when alarms are active.

If a function-specific alarm occurs for a vertical axis when traveling to fixed stop, the NC/PLC interface signal DB11, DBX6.3 (mode group ready) is not reset: This means that the corresponding drive is not de-energized.

This corresponds to an electronic weight compensation for the vertical axis and can be configured via the following machine data:

MD37052 $MA_FIXED_STOP_ALARM_REACTION

References
Further information on vertical axes can be found in:

- SINAMICS S120 Function Manual
- Function Manual, Extended Functions; Compensation (K3),
  Section: Electronic counterweight

7.2.6 Setting data

The values programmed via the function-specific `FXS`, `FXST` and `FXSW` commands are written block-synchronously to the following, immediately effective, axis-specific setting data:

Switching the function on/off

SD43500 $SA_FIXED_STOP_SWITCH (selection/deselection of travel to fixed stop)

Clamping torque

SD43510 $SA_FIXED_STOP_TORQUE (clamping torque)

Note

Clamping torque greater than 100%

A value for the clamping torque in SD43510 greater than 100% of the maximum motor torque is only advisable for a short time. In addition, the maximum motor torque is limited by the drive. For example, the following drive parameters have a limiting effect:

- p1520/p1521 upper torque limit/force limit / lower torque limit/force limit
- p1522/p1523 upper torque limit/force limit / lower torque limit/force limit
- p1530/p1531 power limit, motoring / power limit, regenerating
- p0640 current limit
- p0326 motor stall force correction factor

Detailed information on the drive parameters and the functions can be found in:

References

- SINAMICS S120/S150 List Manual
- SINAMICS S120 Function Manual

Monitoring window

SD43520 $SA_FIXED_STOP_WINDOW (monitoring window)

Default setting

The defaults for the setting data are set via the following machine data:

- Clamping torque:
  MD37010 $MA_FIXED_STOP_TORQUE_DEF (default clamping torque)
- Monitoring window:
  MD37020 $MA_FIXED_STOP_WINDOW_DEF (default monitoring window)

Effectiveness

The setting data for the clamping torque and monitoring window takes effect immediately. In this way, the clamping state can be adapted to the machining situation at any time by the operator or via the PLC user program.

References

Further detailed information on the machine and setting data can be found in:

List Manual, Detailed Machine Data Description

7.2.7 System variables

Reference/actual state

The reference and actual state of the "travel to fixed stop" function can be read using the following system variables:

- $AA_FXS = <value> (status, "Travel to fixed stop" reference state)
- $VA_FXS = <value> (status, "Travel to fixed stop" actual state)

<value>	Description
0	Axis is not at fixed stop
1	Fixed stop was successfully approached
2	Approach to fixed stop failed
3	Selection "Travel to fixed stop" is active.
4	Fixed stop was detected.
5	Travel to fixed stop deselection active

Additional information

If errors occurred when traversing to the fixed stop ($VA_FXS == 2), then additional information is displayed in the following system variables:

- $VA_FXS_INFO = <value> (additional information when "traveling to fixed stop")

<value>	Description
0	No additional information available
1	No approach motion programmed
2	Programmed end position reached, motion stopped
3	Cancellation using a reset, DB21, ... DBX7.7 == 1
4	Monitoring window was exited
5	Torque reduction rejected by drive
6	PLC has withdrawn the enable, DB31, ... DBX3.1 == 0 (travel to fixed stop enabled)

Application example for $AA_FXS

In order that a block change is executed, no alarm should be triggered when a fault occurs. The cause is then determined using the system variable $VA_FXS_INFO and a specific response initiated.

Requirement: MD37050 $MA_FIXED_STOP_ALARM_MASK = 0 (when traveling to fixed stop, no alarm initiated)

```
Program code                                                    Comment
X300 Y500 F200 FXS[X1]=1 FXST[X1]=25 FXSW[X1]=5                 ; Travel to fixed stop
IF $AA_FXS[X1] == 2 GOTOF FXS_ERROR                             ; IF fixed stop == reached
```

Program code	Comment
`   R100=$AA_IM[X1]`	; THEN (normal case)
`   IF R100 ...`	; Evaluation of the
`   ...`	; actual position
`   GOTOF PROG_END`	
`FXS_ERROR:`	; ELSE (error case)
`   CASE($VA_FXS_INFO[X1]) OF 0 GOTOF LABEL_0 OF 1 GOTOF LABEL_1 ...`	; Error handling
`LABEL_0: ...`	
`      GOTOF CASE_END`	
`LABEL_1: ...`	
`      GOTOF CASE_END`	
`...`	
`CASE_END:`	
`...`	
`PROG_END: M30`	; ENDIF end of program

7.2.8 Alarms

Alarm 20091 "Fixed stop not reached"

If the fixed stop position is not reached during travel to fixed stop, alarm 20091 "Fixed stop not reached" is displayed and a block change executed.

Alarm 20092 "Travel to fixed stop is still active"

If there is a travel request or renewed function selection for the axis after the fixed stop has been reached, alarm 20092 "Travel to fixed stop is still active" is displayed.

Alarm 20093 "Standstill monitoring at fixed stop has triggered"

If an axis has reached the fixed stop and is then moved out of this position by more than the value specified in the setting data

SD43520 FIXED_STOP_WINDOW (fixed stop monitoring window)

alarm 20093 "Standstill monitoring at fixed stop has triggered" is displayed, travel to fixed stop for this axis is deselected and the following system variable set:
$AA_FXS[x] = 2

Alarm 20094 "Function has been aborted"

The travel to fixed stop is aborted if the clamping torque can no longer be applied due to the cancellation of the pulse enable, or the requested acknowledgement signal at the NC/PLC interface has been reset:

- Acknowledgement signal required: MD37060 $MA_FIXED _STOP_ACKN_MASK, bit 0 = 1
- Acknowledgement signal: DB31, ... DBX3.1 == 0 (enable travel to fixed stop)

Enabling the fixed stop alarms

The following machine data can be use to set whether the fixed stop alarms

- Alarm 20091 "Fixed stop not reached",
- Alarm 20094 "Fixed stop aborted"

are displayed:

MD37050 $MA_FIXED_STOP_ALARM_MASK (enable of the fixed stop alarms)

Settable functional behavior for fixed stop alarms

The following machine data can be used to set that the function is not aborted when function-specific alarms occur:

- Alarm 20090 Travel to fixed stop not possible
- Alarm 20091 Fixed stop not reached
- Alarm 20092 Travel to fixed stop is still active
- Alarm 20093 Standstill monitoring at fixed stop has triggered
- Alarm 20094 Travel to fixed stop aborted

MD37052 $MA_FIXED_STOP_ALARM_REACTION (reaction to fixed stop alarms)

Alarm suppression after new programming

Travel to fixed stop can be used for simple measuring processes.

For example, it is possible to carry out a check for tool breakage by measuring the tool length by traversing onto a defined obstacle. To do so, the fixed stop alarm must be suppressed. When the function for clamping workpieces is then used "normally," the alarm can be activated using part program commands.

7.2.9 Travel with limited torque/force FOC

Function

Using the "Force Control" function to traverse with limited torque/force, when traversing, the maximum permissible torque/force can be limited to a percentage of the maximum possible axis torque. The limit value can be changed at any time, also while the axis is traversing. The changes can be realized in the interpolator clock cycle, dependent on distance, time or on any other variable. The "Force Control" function can also be programmed in synchronized actions. The function can be permanently activated, or block related.

Programming

Syntax

```
FOCON[<axis>]
FOCOF[<axis>]
```

```
FOC[<axis>]
```

Meaning

Parameter	Meaning
`FOCON`:	Activate torque/force limiting
`FOCOF`:	Deactivate torque/force limiting
`FOC`:	Activate non-modal torque/force limiting
`<axis>`:	Channel axis name, type: AXIS

Example

```
Program code                   Comment
N10 FOCON[X]                   ; Modal activation of the torque limit
N20 X100 Y200 FXST[X]=15       ; X travels with reduced torque (15%)
N30 FXST[X]=75 X20             ; Changing the torque to 75%,
N40 FOCOF[X]                   ; Disable torque limit
```

Parameterization

Machine data

- MD37010 $MA_FIXED_STOP_TORQUE_DEF (default for fixed stop clamping torque)
  The value specified in the machine data is effective after activating the function, as long as no explicit value is programmed using `FXST`.
- MD36042 $MA_FOC_STANDSTILL_DELAY_TIME (delay time standstill monitoring for active torque/force limiting)

Permanent activation (FOCON/FOCOF)

Permanent activation of the "Force Control" function after POWER ON or RESET can be set using the machine data:

MD37080 $MA_FOC_ACTIVATION_MODE, bit 0 and bit 1 (basic setting of the modal force/ torque limiting)

Bit	Value	Meaning
0	0	The "Force Control" function is **not** active after POWER ON.
	1	The "Force Control" function is active after POWER ON.
1	0	The "Force Control" function is **not** active after RESET.
	1	The "Force Control" function is active after RESET.

If a limiting torque or force is programmed using `FXST = <value>` before activation, then this is effective from activation onwards.

If a limiting torque or force was not programmed using `FXST = <value>` before activation, then the value specified in the machine data is effective as standard.

MD37010 $MA_FIXED_STOP_TORQUE_DEF (default for fixed stop clamping torque)

Block-related activation (FOC)

The function for the actual block is activated using the FOC command.

Activating the function using a synchronized action takes effect up to the end of the current part program block.

Priority of FXS before FOC

Activating the "Travel to fixed stop" function FXS has a higher priority than the "Force Control" function. If "Travel to fixed stop" is active at the same time as "Force Control", then the first is executed.

Deselecting the "Travel to fixed stop" function FXS cancels the clamping. A "Force Control" function that is at same time permanently active, remains active.

System variable

Status of the "Force Control" function

The status of the "Force Control" function can be read using system variable $AA_FOC.

Changing the status of the "Travel to fixed stop" function FXS does not change the status of the "Force Control" function.

$AA_FOC	
Value	**Meaning**
0	FOC not active
1	FOC modal active
2	FOC non-modal active

Status of torque limiting

The actual status of the torque limiting can be read using system variable $VA_TORQUE_AT_LIMIT.

$VA_TORQUE_AT_LIMIT	
Value	**Meaning**
0	actually effective torque < torque limit value
1	actually effective torque == torque limit value

Restrictions

the "Force Control" function has 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 NC/PLC interface to check that the active torque limit has been reached.

- If the acceleration limitation is not adapted accordingly, an increase in the following error occurs during the traversing motion.
- If the acceleration limitation is not adapted accordingly, the end-of-block position is possibly reached later than specified in:
  MD36040 $MA_STANDSTILL_DELAY_TIME.
  The machine data:
  MD36042 $MA_FOC_STANDSTILL_DELAY_TIME
  is introduced for this and monitored in this status.

Possible application for link and container axes

All axes that can be traversed in a channel, i.e. also link axes and container axes, can be traversed to fixed stop.

References:
Function Manual, Extended Functions; Several Operator Panels on multiple NCUs, Distributed Systems (B3)

The status of the machine axis is kept in the case of a container rotation, i.e. a clamped machine axis remains at the stop.

If a modal torque limitation has been activated with FOCON, this is kept for the machine axis even after a container rotation.

7.3 Examples

Example 1: Travel to fixed stop with static synchronized actions

Travel to fixed stop (FXS) is initiated when requested via R parameter ($R1) in a static synchronized action.

Program code	Comment
N10 IDS=1 WHENEVER	; Static synchronized action 1:
(($R1==1) AND	; R1==1 (FXS for Y requested) AND
($AA_FXS[Y]==0)) DO	; Avoidance of multiple selection
	; $AA_FXS[Y]==0 (axis not at limit): =>
$R1=0 FXS[Y]=1	; reset $R1, activate FXS for Y
FXST[Y]=10	; Limit torque: 10%
FA[Y]=200	; Axial feedrate Y: 200
POS[Y]=150	; Positioning movement Y
N11 IDS=2 WHENEVER	; Static synchronized action 2:
($AA_FXS[Y]==4) DO	; $AA_FXS[Y]==4 (limit detected): =>
FXST[Y]=30	; Limit torque: 30%
N12 IDS=3 WHENEVER	; Static synchronized action 3:
($AA_FXS[Y]==1) DO	; $AA_FXS[Y]==1 (limit reached successfully): =>

Program code	Comment
FXST[Y]=$R0	; Limit torque: Value from R parameter $R0
N13 IDS=4 WHENEVER	; Static synchronized action 4:
(($R3==1) AND	; R3==1: Deselection of FXS for Y requested
($AA_FXS[Y]==1)) DO	; $AA_FXS[Y]==1 (limit reached successfully): =>
FXS[Y]=0	; Deselect FXS
FA[Y]=1000 POS[Y]=0	; Positioning movement Y
N20 FXS[Y]=0 G0 G90 X0 Y0	; Set initial settings: FXS deselected, X and Y at initial position
N30 RELEASE(Y)	; Release Y for movements in synchronized actions
...	
N60 GET(Y)	Include axis Y back into the path group
...	

Note

Avoidance of multiple selection for FXS

To avoid a multiple selection, we recommend that prior to activating FXS, query either the $AA_FXS==0 system variable or a user-specific flag. See above, program example N10

Example 2: Traveling to fixed stop with block-related synchronized actions

"Travel to fixed stop" is activated from a specific position of the traversing motion of the following block

Program code	Comment
N10 G0 G90 X0	; Starting position
N20 WHEN $AA_IW[X]>17 DO FXS[X]=1	; Synchronized action: Actual position of X axis > 17
	; => activate FXS for axis X
N30 G1 F200 X100	; Traversing motion for axis X

Note

A block-related synchronized action is processed in the following main program block.

7.4 Data lists

7.4.1 Machine data

7.4.1.1 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
36042	FOC_STANDSTILL_DELAY_TIME	Delay time, standstill monitoring for `active torque/ force limiting`
37000	FIXED_STOP_MODE	Travel to fixed stop mode
37002	FIXED_STOP_CONTROL	Sequence monitoring for travel to fixed stop
37010	FIXED_STOP_TORQUE_DEF	Fixed stop clamping torque default setting
37012	FIXED_STOP_TORQUE_RAMP_TIME	Time until the modified torque limit is reached
37020	FIXED_STOP_WINDOW_DEF	Default for fixed stop monitoring window
37030	FIXED_STOP_THRESHOLD	Threshold for fixed stop detection
37040	FIXED_STOP_BY_SENSOR	Fixed stop detection via sensor
37050	FIXED_STOP_ALARM_MASK	Enabling the fixed stop alarms
37052	FIXED_STOP_ALARM_REACTION	Reaction to fixed stop alarms
37060	FIXED_STOP_ACKN_MASK	Monitoring of PLC acknowledgments for travel to fixed stop
37070	FIXED_STOP_ANA_TORQUE	Torque limit on fixed stop approach for analog drives
37080	FOC_ACTIVATION_MODE.	Initial setting of the modal torque/force limiting

7.4.2 Setting data

7.4.2.1 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43500	FIXED_STOP_SWITCH	Selection of travel to fixed stop
43510	FIXED_STOP_WINDOW	Fixed stop clamping torque
43520	FIXED_STOP_TORQUE	Fixed stop monitoring window

7.4.3 Signals

7.4.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Acknowledge fixed stop reached	DB31, ...DBX1.1	DB380x.DBX1.1
Sensor for fixed stop	DB31, ...DBX1.2	DB380x.DBX1.2
Axis/spindle disable	DB31, ...DBX1.3	DB380x.DBX1.3
Controller enable	DB31, … .DBX2.1	DB380x.DBX2.1
Travel to fixed stop enabled	DB31, … .DBX3.1	DB380x.DBX3.1

7.4.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate travel to fixed stop	DB31, ...DBX62.4	DB390x.DBX2.4
Fixed stop reached	DB31, ...DBX62.5	DB390x.DBX2.5

G2: Velocities, setpoint / actual value systems, closed-loop control 8

8.1 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

8.2 Velocities, traversing ranges, accuracies

8.2.1 Velocities

Maximum path and axis velocities and spindle speed

The maximum path and axis velocities and spindle speed are influenced by the machine design, the dynamic response of the drive and the limit frequency of the actual-value acquisition (encoder limit frequency).

The maximum axis velocity is defined in machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

The maximum permissible spindle speed is specified using machine data:

MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

For explanations, see Section "S1: Spindles (Page 1273)".

With a higher feedrate (resulting from programmed feedrates and feedrate override), the velocity is limited to V_{max}.

This automatic feedrate limiting can lead to a drop in velocity over several blocks with programs generated by CAD systems with extremely short blocks.

Example:

Interpolation cycle = 12 ms

```
N10 G0 X0 Y0; [mm]
```

```
N20 G0 X100 Y100; [mm]
```

⇒ Path length programmed in block = 141.42 mm

⇒ V_{max} = (141.42 mm/12 ms) * 0.9 = 10606.6 mm/s = 636.39 m/min

Minimum path, axis velocity

The following restriction applies to the minimum path or axis velocity:

$$V_{min} \geq \frac{10^{-3}}{\text{Computational resolution} \left[\frac{\text{Incr.}}{\text{mm or degrees}}\right] * \text{IPO cycle [s]}}$$

The computational resolution is defined using machine data:
MD10200 $MN_INT_INCR_PER_MM (computational resolution for linear positions)

or
MD10210 $MN_INT_INCR_PER_DEG (computational resolution for angular positions)

If V_{min} is not reached, no traversing is carried out.

Example:

MD10200 $MN_INT_INCR_PER_MM = 1000 [incr/mm] ;

Interpolation cycle = 12 ms;

⇒ $V_{min} = 10^{-3}/(1000\ ^{incr}/_{mm} \times 12\ \text{ms}) = 0.005\ ^{mm}/_{min}$;

The value range of the feedrates depends on the computational resolution selected.

For the standard assignment of machine data:

MD10200 $MN_INT_INCR_PER_MM
(computational resolution for linear positions) (1000 incr./mm)

or

MD10210 $MN_INT_INCR_PER_DEG

(computational resolution for angular positions) (1000 incr./deg.)

The following value range can be programmed with the specified resolution:

Range of values for path feed F and geometry axes:	
Metric system:	Inch system:
0.001 ≤ F ≤ 999,999.999 [mm/min, mm/rev, degrees/min, degrees/rev]	0.001 ≤ F ≤ 399,999.999 [inch/min, inch/rev]

Range of values for feedrate for positioning axes:	
Metric system:	Inch system:
0.001 ≤ FA ≤ 999,999.999 [mm/min, mm/rev, degrees/min, degrees/rev]	0.001 ≤ FA ≤ 399,999.999 [inch/min, inch/rev]

Range of values for spindle speed S:	0.001 ≤ S ≤ 999,999.999 [rpm]

If the computational resolution is increased/decreased by a factor, then the value ranges change accordingly.

8.2.2 Traversing ranges

Range of values of the traversing ranges

The range of values of the traversing range depends on the computational resolution selected.

For the standard assignment of machine data:

MD10200 $MN_INT_INCR_PER_MM

(computational resolution for linear positions) (1000 incr./mm)

or

MD10210 $MN_INT_INCR_PER_DEG

(computational resolution for angular positions) (1000 incr./deg.)

The following value range can be programmed with the specified resolution:

Table 8-1 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 restricted by software limit switches and working areas (see Section "A3: Axis monitoring functions (Page 87)").

For special features for a large traversing range for linear and rotary axes, see Section "R1: Referencing (Page 1223)".

The traversing range for rotary axes can be limited via machine data.

References:

Function Manual, Extended Functions; Rotary Axes (R2)

8.2.3 Positioning accuracy of the control system

Actual-value resolution and computational resolution

The positioning accuracy of the control depends on the actual-value resolution (=encoder increments/(mm or degrees)) and the computational resolution (=internal increments/(mm or degrees)).

The coarse resolution of these two values determines the positioning accuracy of the control.

The input resolution, interpolator and position-control cycle selections have no effect on this accuracy.

As well as limiting using MD32000, the control limits the maximum path velocity in relation to the situation and according to the following formula:

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \frac{\text{DRIVE\_AX\_RATIO\_DENOM [n]}}{\text{DRIVE\_AX\_RATIO\_NUMERA [n]}} * \text{LEADSCREW\_PITCH} * \text{INT\_INCR\_PER\_MM}$$

8.2.4 Input/display resolution, computational resolution

Resolutions: Differences

Resolutions, e.g. resolutions of linear and angular positions, velocities, accelerations and jerk, must be differentiated as follows:

- Input resolution
  Data is input via the control panel or part programs.
- Display resolution
  Data is displayed via the control panel.
- Computational resolution
  Data input via the control panel or part program is displayed internally.

The input and display resolution is determined by the specified operator panel front used, whereby the display resolution for position values with the machine data:

MD9004 $MM_DISPLAY_RESOLUTION (display resolution)

can be changed.

The machine data:

MD9011 $MM_DISPLAY_RESOLUTION_INCH (display resolution for INCH measuring system)

can be used to configure the display resolution for position values with inch setting.

This allows you to display up to six decimal places with the inch setting.

For the programming of part programs, the input resolutions listed in the Programming Guide apply.

The desired computational resolution is defined using the machine data:

MD10200 $MN_INT_INCR_PER_MM (computational resolution for linear positions)

and

MD10210 $MN_INT_INCR_PER_ DEG (computational resolution for angular positions).

It is independent of the input/display resolution but should have at least the same resolution.

The maximum number of places after the decimal point for position values, velocities, etc., in the part program and the number of places after the decimal point for tool offsets, zero offsets, etc. (and therefore also for the maximum possible accuracy) is defined by the computational resolution.

The accuracy of angle and linear positions is limited to the computational resolution by rounding the product of the programmed value with the computational resolution to an integer number.

To make the rounding clear, powers of 10 should be used for the calculation resolution.

Example of rounding:

Computational resolution: 1000 incr./mm

Programmed path: 97.3786 mm

Effective value: 97.379 mm

Example of programming in the $^{1}/_{10}$ µm range:

All the linear axes of a machine are to be programmed and traversed within the range of values 0.1 to 1000 µm.

⇒ In order to position accurately to 0.1 µm, the computational resolution must be set to ≥ 10^4 incr./mm.

⇒ MD10200 $MN_INT_INCR_PER_MM = 10000 [incr./mm]:

⇒ Example of related part program:

Program code	Comment
`N20 G0 X 1.0000 Y 1.0000`	`; Axes travel to the position X=1.0000 mm, Y=1.0000 mm;`
`N25 G0 X 5.0002 Y 2.0003`	`; Axes travel to the position X=5.0002 mm, Y=2.0003 mm`

8.2.5 Scaling of physical quantities of machine and setting data

Input/output units

Machine and setting data with a physical unit are interpreted in the input/output units stated below depending on whether the metric or inch system is selected:

Physical unit	Input/output units for standard basic system	
	Metric	Inch
Linear position	1 mm	1 inch
Angular position	1 degree	1 degree
Linear velocity	1 mm/min	1 inch/min
Angular velocity	1 rpm	1 rpm
Linear acceleration	1 m/s^2	1 inch/s^2
Angular acceleration	1 rev./^{s2}	1 rev./s^2
Linear jerk	1 m/s^3	1 inch/s^3
Angular jerk	1 rev./s^3	1 rev./s^3
Time	1 s	1 s
Position controller servo gain	1/s	1/s
Revolutional feedrate	1 mm/rev	inch/rev
Compensation value linear position	1 mm	1 inch
Compensation value angular position	1 degree	1 degree

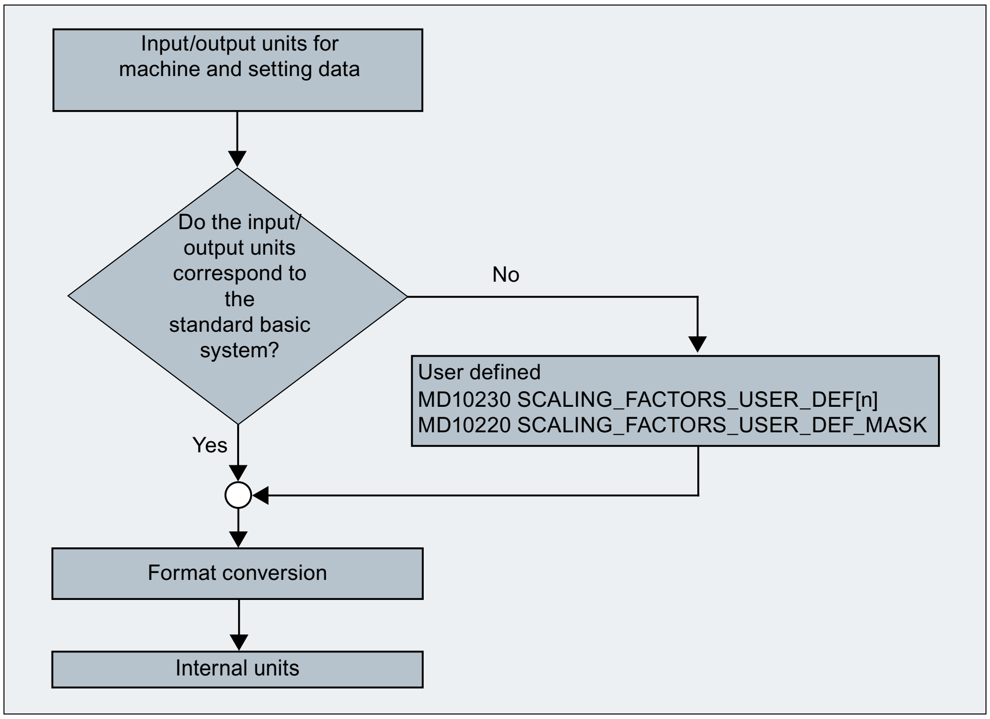
Internally in the control, the following units are used, whatever basic system is selected:

Physical unit	Unit
Linear position	1 mm
Angular position	1 degree
Linear velocity	1 mm/s
Angular velocity	1 deg./s
Linear acceleration	1 mm/s^2
Angular acceleration	1 degree/s^2
Linear jerk	1 mm/s^3
Angular jerk	1 degree/s^3
Time	1 s
Position controller servo gain	1/s

Physical unit	Unit
Revolutional 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. This also requires an adjustment between the newly selected input/output units and the internal units via the following machine data:

- MD10220 $MN_SCALING_USER_DEF_MASK
- MD10230 $MN_SCALING_FACTORS_USER_DEF[n]


Input/output units for machine and setting data
Do the input/output units correspond to the standard basic system?
No
Yes
User defined
MD10230 SCALING_FACTORS_USER_DEF[n]
MD10220 SCALING_FACTORS_USER_DEF_MASK
Format conversion
Internal units

The following applies:

Selected I/O unit = (MD10230 $MN_SCALING_FACTORS_USER_DEF[n]) * internal unit

The selected I/O unit, expressed in the internal units 1 mm, 1 degree, and 1 s must therefore be entered in machine data MD10230 $MN_SCALING_FACTORS_USER_DEF[n].

Example 1:

Machine data input/output of the linear velocities is to be in m/min instead of mm/min (initial state).

(The internal unit is mm/s)

⇒ The scaling factor for the linear velocities is to differ from the standard setting. For this, in machine data:
MD10220 $MN_SCALING_USER_DEF_MASK
bit number 2 must be set.
⇒ MD10220 $MN_SCALING_USER_DEF_MASK = 'H4'; (bit no. 2 as hex value)

⇒ The scaling factor for the linear velocities is to differ from the standard setting. For this, in machine data:
MD10220 $MN_SCALING_USER_DEF_MASK
bit number 2 must be set.
⇒ MD10220 $MN_SCALING_USER_DEF_MASK = 'H4'; (bit no. 2 as hex value)

⇒ The scaling factor is calculated using the following formula:

$$\text{MD10230 SCALING\_FACTORS\_USER\_DEF[n]} = \frac{\text{Selected input/output unit}}{\text{Internal unit}}$$

$$\text{MD10230 SCALING\_FACTORS\_USER\_DEF[n]} = \frac{1\ \frac{\text{m}}{\text{min}}}{1\ \frac{\text{mm}}{\text{s}}} = \frac{\frac{1000}{60}\ \frac{\text{mm}}{\text{s}}}{1\ \frac{\text{mm}}{\text{s}}} = \frac{1000}{60} = 16{,}667;$$

⇒ MD10230 SCALING_FACTORS_USER_DEF[n] = 16,667

Index n defines the "linear velocity" in the "Scaling factors of physical quantities" list.

Example 2:

In addition to the change in Example 1, the machine data input/output of linear accelerations must be in ft/s^2 instead of m/s^2 (initial state).
(The internal unit is mm/s^2.)

⇒ MD10220 SCALING_USER_DEF_MASK = H14 ; (Bit No. 4 and Bit No. 2 of example 1 as a Hex value)

$$\Rightarrow \text{MD10220 SCALING\_FACTORS\_USER\_DEF[4]} = \frac{1\ \frac{\text{ft}}{\text{s}^2}}{1\ \frac{\text{mm}}{\text{s}^2}} = \frac{12*25{,}4\ \frac{\text{mm}}{\text{s}^2}}{1\ \frac{\text{mm}}{\text{s}^2}} = 304{,}8\ ;$$

⇒ MD10220 SCALING_FACTORS_USER_DEF[4] = 304,8

Index 4 defines the "linear acceleration" in the "Scaling factors of physical quantities" list.

8.3 System of units, metric/inch

8.3.1 Function

8.3.1.1 Parameterized and programmed system of units

SINUMERIK control systems can operate with a metric system of units as well as an inch system of units.

Parameterized system of units (basic system)

The basic setting of the system of units (basic system) is set in machine data MD10240 $MN_SCALING_SYSTEM_IS_METRIC (see "Commissioning (Page 354)").

Depending on the basic system, all length-related data is interpreted either as metric or inch system of units.

Metric system of units:	mm, mm/min, m/s2, m/s3, mm/rev.
Inch system of units:	inch, inch/min, inch/s2, inch/s3, inch/rev.

The basic system also defines the interpretation of the programmed F value for linear axes:

Feed type	Metric system of units	Inch system of units
G94	mm/min	inch/min
G95	mm/rev.	inch/rev.

Programmed system of units

Using the commands of G group 13 (system of units, inch/metric) within the part program, you can toggle between the metric and inch system of units (see "Programming (Page 358)").

The programmed system of units and the basic system may be identical or different at any time. Switchover of the system of units within a particular section of the part program could, for example, be used to machine an inch thread on a workpiece within a metric basic system.

The reset setting of the G group is defined in machine data MD20150 $MC_GCODE_RESET_VALUES (see "Commissioning (Page 354)").

HMI display

Data relating to length is displayed as follows on the user interface:

	Display	
	in the basic system	in the programmed system of units
Machine data	x	
Data in the machine coordinate system	x	

	Display	
	in the basic system	in the programmed system of units
Tool data	x	
Work offsets	x	
Data in the workpiece coordinate system		x

NC/PLC interface

In the case of NC/PLC interface signals containing dimension information, e.g. feedrate for path and positioning axes, data exchange is carried out with the PLC in the configured basic system.

Reading in external part programs

If part programs, including data sets (work offsets, tool offsets, etc.), programmed in a system of units other than the basic system are read in from an external source, then the basic system must first be changed.

8.3.1.2 Extended system of units functionality

From SW 5 and higher, the system of units functionality has been extended, which significantly simplifies toggling between systems of units.

The functions include:

- System of units switchover at the user interface (Page 352)
- New G commands G700/G710 (see "Programming (Page 358)")
- Data backup with system of units identifier INCH/METRIC (see "Commissioning (Page 354)")
- Automatic data conversions when the system of units is changed, e.g. for
  - Work offsets
  - Compensation data (EEC, QEC)
  - Tool offsets
  - ...

For compatibility reasons, the extended system of units functionality must be activated using a machine data (see "Commissioning (Page 354)").

8.3.1.3 System of units switchover at the user interface

Requirement

Extended system of units functionality is active (MD10260 $MN_CONVERT_SCALING_SYSTEM = 1; see "Commissioning (Page 354)")

Function

The relevant softkey on the HMI in the "Machine" operating area is used to change the system of units of the control system.

The system of units is only switched over under the following conditions:

- MD20110 $MC_RESET_MODE_MASK, bit 0 is set in every channel.
- All channels are in the Reset state.
- Axes do not currently 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 while the system of units is being switched over.

The actual change in the system of units is made by writing all the necessary machine data and subsequently activating them.

The following machine data are automatically switched over consistently for all configured channels:

- MD10240 $MN_SCALING_SYSTEM_IS_METRIC
- MD20150 $MN_GCODE_RESET_VALUES

Reset position

When the system of units is switched over from the user interface, the reset position of the G group 13 is automatically adapted to the actual system of units.

MD20150 $MC_GCODE_RESET_VALUES[12] = <actual system of units>

NCU link

Note

If several NCUs are linked by NCU link, the switchover has the same effect on all linked NCUs. If the requirements for a switchover are not fulfilled on one of the connected NCUs, no switchover will take place on any of the NCUs. It is assumed that when there is a NCU link, interpolations between several NCUs will take place on the existing NCUs, whereby the interpolations can provide correct results only if the same systems of units are used.

References:

Function Manual, Extended Functions; Section "B3: distributed systems"

System data

When the system of units is switched over, from the user's viewpoint, all length-related specifications are converted to the new system of units automatically.

This includes:

- Positions
- Feedrates
- Accelerations

- Jerk
- Tool offsets
- Programmable, adjustable and external work offsets and DRF offsets
- Compensation values
- Protection zones
- Machine data
- JOG and handwheel factors

After the switchover, all of the above mentioned data is available in physical quantities.

Data, for which no unique physical units are defined, is **not** converted automatically.

This includes:

- R parameters
- GUDs (**G**lobal **U**ser **D**ata)
- LUDs (**L**ocal **U**ser **D**ata)
- PUDs (**P**rogram global **U**ser **D**ata)
- Analog inputs/outputs
- Data exchange via FC21

The user is prompted to take the currently valid system of units MD10240 $MN_SCALING_SYSTEM_IS_METRIC into consideration.

NC/PLC interface

The current system of units setting can be read at the NC/PLC interface using the following signal:

DB10 DBX107.7 (inch system of units)

The number of times that the system of units has been switched over since the last time the control powered up can be read out using the following signal byte:

DB10 DBB71 (change counter, system of units inch/metric)

Starting value after the control powered up: 1

8.3.2 Commissioning

NC-specific machine data

Basic system

Das von der open loop control zu verwendende Basic system for the scaling lengthndependinger physical Größen bei der data-input/output if predefined Monitor das machine data:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC

Value	Meaning
0 (= FALSE)	Data relating to length is interpreted as **inch** data.
1 (= TRUE)	Data relating to length is interpreted as **metric** data.

The system must be powered up again after changing this machine data as otherwise associated machine data, which have physical units, will be incorrectly scaled.

The following procedure must be complied with:

- Changing the MD using manual entry:
  - 1. Power up the system.
  - 2. Machine data with physical units: Enter a value.
- The MD is changed using machine data.
  - 1. Power up the system.
  - 2. Load the machine data file once again so that the new physical units are taken into account.

When changing MD10240, Alarm 4070 "Normalizing machine data has been changed" is output.

Note

When changing over the system of units at a control system, all length-related data must be converted consistently and completely into the other measuring system.

Conversion factor (NC-specific)

The factor for converting from metric into the inch system of units is set in machine data:

MD10250 $MN_SCALING_VALUE_INCH (conversion factor for inch)

Default value: 25.4

The conversion factor becomes active when selecting the non-metric basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC = 0).

The following data are multiplied by the conversion factor:

- Length-related data for input/output (e.g. when uploading machine data, work offsets)
- Programmed F values for linear axes
- Programmed geometry of an axis (position, polynomial coefficients, radius for circle programming, ...), if the system of units programmed with G70/G71 differs from the basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC).

Note

By changing the conversion factor, the control system can be adapted to customer-specific systems of units.

Extended system of units functionality

The extended system of units functionality can be activated using the following compatibility machine data:

MD10260 $MN_CONVERT_SCALING_SYSTEM

Value	Meaning
0 (= FALSE)	Extended system of units functionality **not**active (default setting; compatibility to previous software releases)
1 (= TRUE)	Extended system of units functionality active

When changing MD10240, Alarm 4070 "Normalizing machine data has been changed" is output.

Data backup

Individual data sets that are read from the control system and contain data affected by the system of units, are assigned an identifier during reading that indicates the current setting for the system of units:

- MD10260 $MN_CONVERT_SCALING_SYSTEM
- MD10240 $MN_SCALING_SYSTEM_IS_METRIC

The identifier records in which system of units the data have been read out. This ensures that no data sets are read into the control with a system of units other than that which is currently set.

Since the identifier is also evaluated in part programs, these can also be "protected" against operator errors as described above. You can therefore prevent part programs containing, e.g. only metric data, from running on an inch system of units.

Archive and machine data sets are downward-compatible with the following setting:

MD11220 $MN_INI_FILE_MODE = 2

Note

The INCH/METRIC identifier is only generated if the compatibility machine data:

MD10260 $MN_CONVERT_SCALING_SYSTEM = TRUE

Note

Rounding machine data

All length-related machine data is rounded to the nearest 1 pm when writing in the inch system of units (MD10240 $MN_SCALING_SYSTEM_IS_METRIC=0 and MD10260 $MN_CONVERT_SCALING_SYSTEM=1), to avoid rounding problems.

This corrects the loss of accuracy resulting from conversion to ASCII when reading out a data backup in the inch system of units when the data is read back into the system.

Input resolution and computational resolution

The input and calculation resolution is defined as an internal number of increments per millimeter.

MD10200 $MN_INT_INCR_PER_MM

The precision of entry of linear positions is limited to the calculation resolution. The product of the programmed position value and the calculation resolution is rounded up to the next integer. To make the rounding clear and understandable, powers of 10 should be used for the calculation resolution.

Example:

1 inch = 25.4 mm ⇒ 0.0001 inch = 0.00254 mm = 2.54 µm = 2540 nm

To be able to program and display the last 40 nm, a value of 100,000 must be parameterized for the input and calculation resolution.

Only with this identical setting for both systems of units is it possible to change the system of units 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.

System of units for positioning tables

The system of units for positional data of the indexing axis tables and switching points for software cams is configured in machine data:

MD10270 $MN_POS_TAB_SCALING_SYSTEM

Reference:
Function Manual, Extended Functions; Chapter "N3: software cams, position switching signals" and "T1: indexing axes"

User tool data

The physical units for user-defined tool and tool cutting edge data can be set in the following machine data:

- MD10290 $MN_CC_TDA_PARAM_UNIT
- MD10292 $MN_CC_TOA_PARAM_UNIT

Note

When the system of units is switched over, all length-related tool data is converted to the new system of units.

Channel-specific machine data

Reset position

For each channel, the reset position of G group 13 (system of units, inch/metric) is set in the machine data:

MD20150 $MC_GCODE_RESET_VALUES[12] (reset position of G group 13)

Axis/spindle-specific machine data

Conversion factor (axis-specific)

To make pure positioning axes independent of G70/G71, a factor for converting from metric into the inch system of units can be set in machine data:

MD31200 $MA_SCALING_FACTOR_G70_G71 (conversion factor when G70/G71 is active)

Default value: 25.4

Note

The conversion factor should be identical for all three geometry axes.

JOG and handwheel factor

The following machine data comprises two values containing axis-specific increment weighting factors for each of the two system of units.

MD31090 $MA_JOG_INCR_WEIGHT

Depending on the actual setting in MD10240 SCALING_SYSTEM_IS_METRIC, the control system automatically selects the appropriate value.

The user defines the two increment factors when the system is commissioned.

Example: Increment factors for the 1st axis

- 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 at every inch/metric switchover.

Remaining distances are not accumulated during incremental traversing with JOG when the system of units is changed, since all internal positions always refer to mm.

System of units for sag compensation

The system of units for sag compensation is configured using machine data:

MD32711 $MA_CEC_SCALING_SYSTEM_METRIC

Reference:
Function Manual, Extended Functions; Chapter "K3: compensations:

8.3.3 Programming

8.3.3.1 Switching over the system of units (G70/G71/G700/G710)

Using the commands of G group 13 (inch/metric system of units) within a part program, you can switch over between the metric and inch system of units.

Activation

In order that commands G700 and G710 are available, the extended system of units functionality must be switched on (MD10260 $MN_CONVERT_SCALING_SYSTEM = 1).

Syntax

```
G70
G71
G700
G710
```

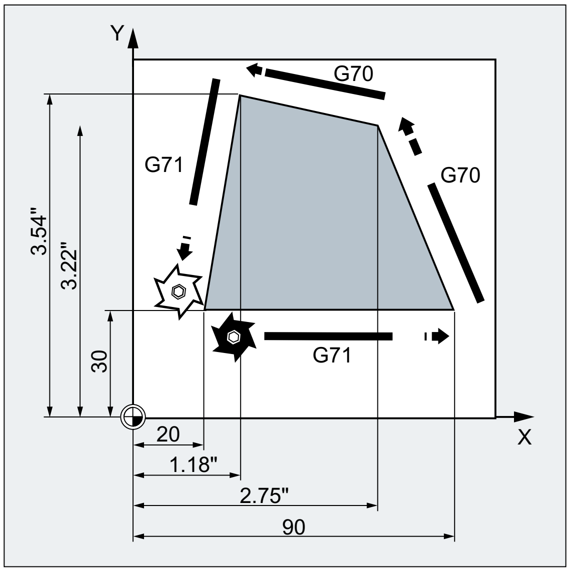
Meaning

`G70`:	Activating the inch system of units The **inch system of units** is used to read and write **geometrical data in units of length**. **Technological data in units of length** (e.g. feedrates, tool offsets, adjustable work offsets, machine data and system variables) is read and written using the **parameterized basic system**.	
	G group:	13
	Initial setting:	Settable via MD20150 $MC_GCODE_RESET_VALUES
	Effectiveness:	Modal
`G71`:	Activating the metric system of units The **metric system of units** is used to read and write **geometrical data in units of length**. **Technological data in units of length** (e.g. feedrates, tool offsets, adjustable work offsets, machine data and system variables) is read and written using the **parameterized basic system**.	
	G group:	13
	Initial setting:	Settable via MD20150 $MC_GCODE_RESET_VALUES
	Effectiveness:	Modal
`G700`:	Activating the inch system of units All geometrical and technological data in units of length is read and written using the inch system of units.	
	G group:	13
	Initial setting:	Settable via MD20150 $MC_GCODE_RESET_VALUES
	Effectiveness:	Modal
`G710`:	Activating the metric system of units All geometrical and technological data in units of length is read and written using the metric system of units.	
	G group:	13
	Initial setting:	Settable via MD20150 $MC_GCODE_RESET_VALUES
	Effectiveness:	Modal

NOTICE
Axis-specific data of rotary axes
Axis-specific data of rotary axes is read and written using the parameterized basic system.

Example

The basic system is metric (MD10240 $MN_SCALING_SYSTEM_IS_METRIC = 1). However, the workpiece drawing has dimensions shown in inches. This is the reason why within the part program, the inch system of units is selected. After the inch dimensions have been processed, the metric system of units is again selected.



Program code	Comment
N10 G0 G90 X20 Y30 Z2 S2000 M3 T1	; X=20 mm, Y=30 mm, Z=2 mm, F=rapid traverse mm/min
N20 G1 Z-5 F500	; Z=-5 mm, F=500 mm/min
N30 X90	; X=90 mm
N40 **G70** X2.75 Y3.22	; programmed system of units: inch
	; X=2.75 inch, Y=3.22 inch, F=500 mm/min
N50 X1.18 Y3.54	; X=1.18 inch, Y=3.54 inch, F=500 mm/min
N60 **G71** X20 Y30	; programmed system of units: Metric
	; X=20 mm, Y=30 mm, F=500 mm/min
N70 G0 Z2	; Z=2 mm, F=rapid traverse mm/min
N80 M30	; end of program

Additional information

Reading and writing data in the case of G70/G71 and G700/G710

Data area	G70 / G71		G700 / G710	
	Read	Write	Read	Write
Display, decimal places (WCS)	P	P	P	P
Display, decimal places (MCS)	G	G	G	G
Feedrates	G	G	P	P
Position data X, Y, Z	P	P	P	P

Data area	G70 / G71		G700 / G710	
	Read	Write	Read	Write
Interpolation parameters I, J, K	P	P	P	P
Circle radius (CR)	P	P	P	P
Polar radius (RP)	P	P	P	P
Thread pitch	P	P	P	P
Programmable FRAME	P	P	P	P
Settable FRAMES	G	G	P	P
Basic frames	G	G	P	P
External work offsets	G	G	P	P
Axial preset offset	G	G	P	P
Working area limits (G25/G26)	G	G	P	P
Protection zones	P	P	P	P
Tool offsets	G	G	P	P
Length-related machine data	G	G	P	P
Length-related setting data	G	G	P	P
Length-related system variables	G	G	P	P
GUDs	G	G	G	G
LUDs	G	G	G	G
PUDs	G	G	G	G
R parameters	G	G	G	G
Siemens cycles	P	P	P	P
Jog/handwheel increment factor	G	G	G	G
P: Writing/reading is performed in the programmed system of units. G: Writing/reading is performed in the configured basic system				

Synchronized actions

Note

Reading position data in synchronized actions

If a system of units has not been explicitly programmed in the synchronized action (condition component and/or action component) **length-related position data** in the synchronized action will always be read in the parameterized **basic system**.

References: Function Manual, Synchronized Actions

8.4 Setpoint/actual-value system

8.4.1 General information

Control loop

A control loop with the following structure can be configured for every closed-loop controlled axis/spindle:

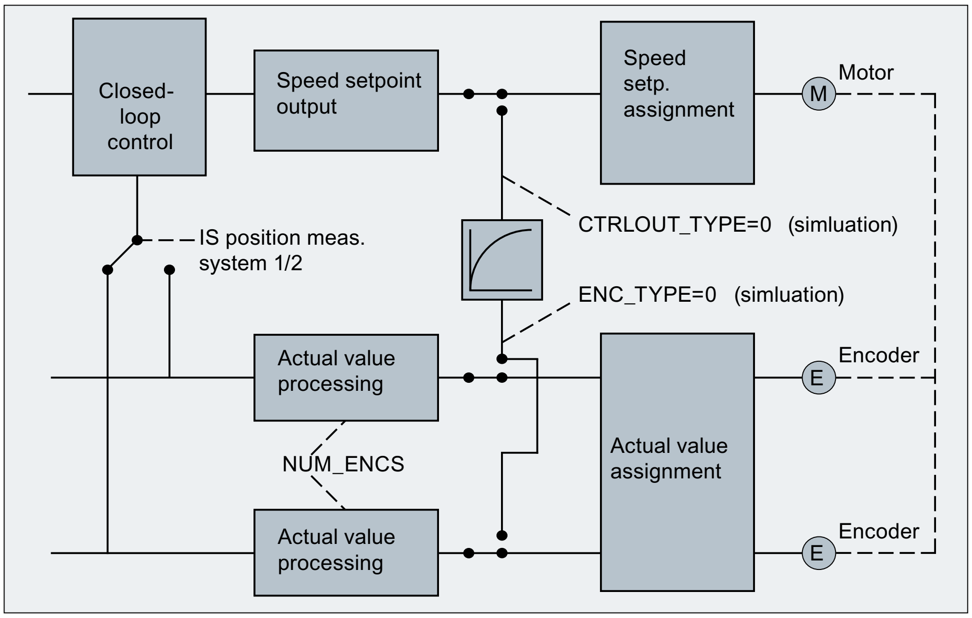


Figure 8-1 Block diagram of a control loop

Setpoint output

A setpoint telegram can be output for each axis/spindle. The setpoint output to the actuator is realized from the SINUMERIK 840D sl.

Actual-value acquisition

A maximum of two measuring systems can be connected for each axis/spindle, e.g. a direct measuring system for machining processes with high accuracy requirements and an indirect measuring system for high-speed positioning tasks.

The number of encoders used is recorded in the machine data:

MD30200 $MA_NUM_ENCS (number of encoders)

In the case of two actual-value branches, the actual value is acquired for both branches.

The active measuring system is always used for position control, absolute value calculation and display. If both measuring systems are activated at the same time by the PLC interface, positioning measuring system 1 is chosen internally by the controller.

Reference point approach is executed by the selected measuring system.

Every position measuring system must be referenced separately.

For explanations on encoder monitoring functions, see Section "A3: Axis monitoring functions (Page 87)".

For an explanation of actual-value acquisition compensation functions, see:
References:
Function Manual, Extended Functions; Compensations (K3)

Switching between measuring systems

One can switch between the two measuring systems through the following NC/PLC interface signals:

DB31, ... DBX1.5 (position measuring system 1)

DB31, ... DBX1.6 (position measuring system 2)

For further information, see Section "A2: Various NC/PLC interface signals and functions (Page 41)".

It is possible to switch over measuring systems at any time, the axes do not have to be stationary to do this. Switchover only takes place if a permissible deviation between the actual values and the two measuring systems has not been violated.

The associated tolerance is entered in the machine data:

MD36500 $MA_ENC_CHANGE_TOL (max. tolerance on position actual value switchover)

On switchover, the current difference between position measuring system 1 and 2 is traversed immediately.

Monitoring

The permissible deviation between the actual values of the two measuring systems is to be entered in the machine data:

MD36510 $MA_ENC_DIFF_TOL

For the cyclic comparison of the two measuring systems used, this difference must not be exceeded, as otherwise Alarm 25105 "Measuring systems deviate" is generated.

If the axis is not referenced (at least in the current control measuring system), then the related monitoring is not active if MD36510 = 0 or if neither of the two measuring systems in the axis is active/available.

Types of actual-value acquisition

The used encoder type must be defined through the following machine data:

MD30240 $MA_ENC_TYPE (type of actual-value acquisition (actual position value))

Simulation axes

The speed control loop of an axis can be simulated for test purposes.

The axis "traverses" with a following error, similar to a real axis.

A simulation axis is defined by setting the two following machine data to "0":

MD30130 $MA_CTRLOUT_TYPE[n] (output value of setpoint)

MD30240 $MA_ENC_TYPE[n] (type of actual-value acquisition)

As soon as the standard machine data has been loaded, the axes become simulation axes.

The setpoint and actual value can be set to the reference point value with reference point approach.

The machine data:

MD30350 $MA_SIMU_AX_VDI_OUTPUT (output of axis signals with simulation axes)

can be used to define whether the axis-specific interface signals are to be output on the PLC during the simulation.

Actual-value correction

If actual-value corrections performed by the NC on the encoder selected for position control do not influence the actual value of another encoder defined in the same axis, then this encoder is to be declared as "independent" via the following machine data:

MD30242 $MA_ENC_IS_INDEPENDENT

Actual-value corrections include the following:

- Modulo treatment
- Reference point approach
- Measuring system comparison
- PRESET

8.4.2 Setpoint and encoder assignment

Setpoint marshalling

The following machine data is relevant for the setpoint assignment of a machine axis.

MD30100		**$MA_CTRLOUT_SEGMENT_NR**
Setpoint assignment: Bus segment		
System	**Value**	**Meaning**
840D sl	5	PROFIBUS DP / PROFINET (default)

MD30110		$MA_CTRLOUT_MODULE_NR
Setpoint assignment: Drive number / module number		
System	**Value**	**Meaning**
840D sl	x	The logical I/O address of the drive is assigned from MD13050 $MN_DRIVE_LOGIC_ADDRESS[n] via the drive number. The drive number (x) results from the index (n) of MD13050: x = n + 1 **Note** The machine data is of no significance if the drive is simulated (MD30130 $MA_CTRLOUT_TYPE = 0).

MD30120		$MA_CTRLOUT_NR
Setpoint assignment: Setpoint output on drive module/module		
System	**Value**	**Meaning**
840D sl	1	Modular drive at PROFIBUS / PROFINET with PROFIdrive profile (default)

MD30130		$MA_CTRLOUT_TYPE
Setpoint output type		
System	**Value**	**Meaning**
840D sl	0	Simulation (operation without drive)
	1	Setpoint output active

Encoder assignment

The following machine data is relevant for assigning the encoder information of the drive - transferred in the PROFIdrive telegram - to the encoder inputs of the machine axis:

MD30210		$MA_ENC_SEGMENT_NR[n]
Actual value assignment, bus segment		
System	**Value**	**Meaning**
840D sl	5	PROFIBUS DP / PROFINET

MD30220		$MA_ENC_MODULE_NR[n]
Actual value assignment: Drive number/measuring circuit number		
System	**Value**	**Meaning**
840D sl	x	The logical I/O address of the drive is assigned from MD13050 $MN_DRIVE_LOGIC_ADDRESS[n] via the drive number. The drive number (x) results from the index (n) of MD13050: x = n + 1

MD30230		$MA_ENC_INPUT_NR[n]
Actual value assignment: Input on drive module/measuring circuit module		
System	**Value**	**Meaning**
840D sl	x	Number of the encoder interface within the PROFIdrive telegram Examples PROFIdrive telegram 103 x = 1 → 1st encoder interface (G1_ZSW, G1_XIST1, G1_XIST2) x = 2 → 2nd encoder interface (G2_ZSW, G2_XIST1, G2_XIST2) PROFIdrive telegram 118 x = 1 → 1st encoder interface (G2_ZSW, G2_XIST1, G2_XIST2) x = 2 → 2nd encoder interface (G3_ZSW, G3_XIST1, G3_XIST2) **Note:** For SINAMICS S120: - Encoder 1 (G1_...): Motor encoder - Encoder 2 (G2_...): Direct measuring system - Encoder 3 (G3_...): Additional measuring system

MD30240		$MA_ENC_TYPE[n]
Encoder type of the actual value acquisition (actual position value)		
System	**Value**	**Meaning**
840D sl	0	Simulation (operation without encoder)
	1	Incremental encoder
	4	Absolute encoder
Note: Corresponds to PROFIdrive parameter p979		

MD30242		$MA_ENC_IS_INDEPENDENT[n, axis]
Encoder is independent		
System	**Value**	**Meaning**
840D sl	0	The encoder is not independent.
	1	The encoder is independent. If the actual-value corrections, which are made for the encoder selected for the position control, are not to influence the actual value of the second encoder defined in the same axis, then this should be declared as independent. Actual value corrections are: ● - Modulo treatment ● - Reference point approach ● - Measuring system alignment ● - PRESET Example: One axis, two encoders, the 2nd encoder is independent MD30200 $MA_NUM_ENCS[AX1] = 2 MD30242 $MA_ENC_IS_INDEPENDENT[0, AX1] = 0 MD30242 $MA_ENC_IS_INDEPENDENT[1, AX1] = 1 Selection, position measuring system 1 / 2: DB31.DBX1.5 / 1.6 If encoder 1 is selected for closed-loop position control, then the actual value corrections are only performed on this encoder, as encoder 2 is independent. If encoder 2 is selected for position control, then the actual value corrections are performed on both encoders, as encoder 1 is not independent. This means that the machine data only has an effect on the passive encoder of a machine axis.
	2	The passive encoder is dependent. The actual encoder value is changed by the active encoder. In combination with MD35102 $MA_REFP_SYNC_ENCS = 1, for reference point approach, the passive encoder is aligned to the active encoder - but is NOT referenced. In the referencing mode MD34200 $MA_ENC_REFP_MODE = 3 (distance-coded reference marks) the passive encoder is automatically referenced with the next traversing motion after passing the zero mark distance. This is done independent of the actual operating mode setting.
	3	The encoder is independent. For modulo rotary axes, modulo actual value corrections are also performed in the passive encoder.

Note

Machine data index [n]

The machine data index [n] for encoder assignment has the following meaning:

- n = 0: First encoder assigned to the machine axis
- n = 1: Second encoder assigned to the machine axis

The assignment is made using machine data:

- MD30220 $MA_ENC_MODULE_NR[n]
- MD30230 $MA_ENC_INPUT_NR[n]

References

Commissioning Manual, CNC Commissioning: NC, PLC, Drive;
Section: "Communication between the NC and the drive" > "Drives: Assign axis"

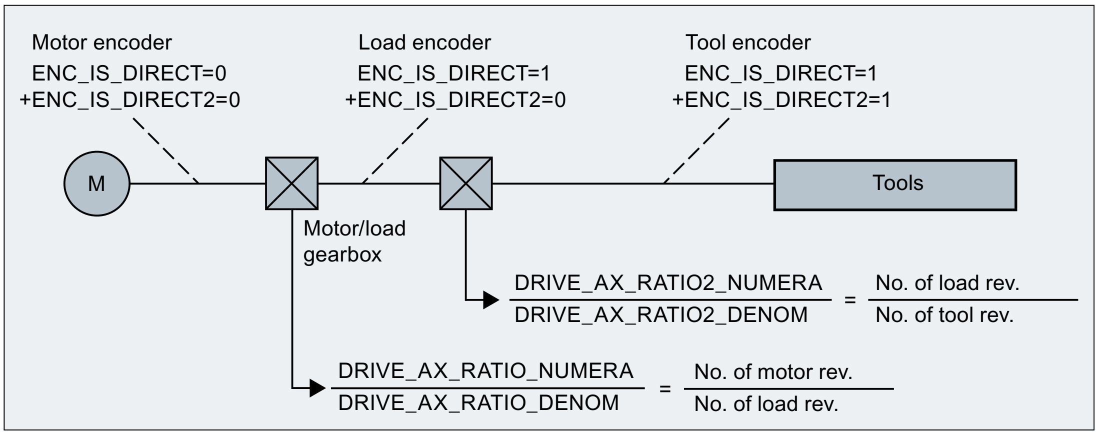
8.4.3 Adapting the motor/load ratios

Gear types

The following gear types are available for adapting the mechanical ratios:

Gear type	Activation	Adaptation	Installation location
Motor/load gear	Parameter set	Fixed configuration	Gear unit
Measuring gear encoder	Power On	Sensor-dependent	Sensor-side
Load intermediate gear unit	Warm restart	Load-dependent	Tool-side

Local position of gear unit / encoder



$$\frac{\text{DRIVE\_AX\_RATIO2\_NUMERA}}{\text{DRIVE\_AX\_RATIO2\_DENOM}} = \frac{\text{No. of load rev.}}{\text{No. of tool rev.}}$$

$$\frac{\text{DRIVE\_AX\_RATIO\_NUMERA}}{\text{DRIVE\_AX\_RATIO\_DENOM}} = \frac{\text{No. of motor rev.}}{\text{No. of load rev.}}$$

Figure 8-2 Gear unit types and encoder locations

Motor/load gear

The motor/load gear supported by SINUMERIK is configured via the following machine data:

- MD31060 $MA_DRIVE_AX_RATIO_NUMERA (numerator load gear)
- MD31050 $MA_DRIVE_AX_RATIO_DENOM (denominator load gear)

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 system to synchronize the position controller with the relevant transmission ratios.

Since a gear stage change is not always carried out automatically, and there are also several ways to request a gear stage change, the position controller is not always incorporated via parameter sets.

Note

For further information about the parameter sets for gear stage change, see Section "S1: Spindles (Page 1273)".

Intermediate gear

Additional, configurable load intermediate gears are also supported by the control system:

- 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 gear and the motor/load gear.

⚠ CAUTION

Different gear transmission ratios for switching

Unlike the motor/load gear, there is no parameter set for the intermediate gear and, therefore, no way of controlling the time-synchronized switchover to the part program or PLC (NC/PLC interface). Part machining during gear change is, therefore, ruled out. It remains the task of the user to match the synchronization of the relevant changed machine data to the corresponding mechanical switchover and activate it. On switchover during a motion, compensations **cannot** be ruled out due to jumps in the scaling factors. These are not monitored for violation of the maximum acceleration.

Encoder directly at the tool

Another connection option is possible for a "tool-side encoder" on the intermediate gear by configuring the following machine data:

MD31044 $MA_ENC_IS_DIRECT2

Encoder not directly at the tool

The following supplementary conditions apply to a gear change of the intermediate gear in position-control mode: The gear ratio to be changed is incorporated in a re-scaling of the encoder information in this case.

In this case, the following applies to axes/spindles in positioning mode:

- A non-abrupt gear change is **only possible at zero speed**.
  To do this, the tool-side position before and after a gear change are set equal for a change in the ratio, since the mechanical position does not (or hardly) change 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.

Boundary 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 machine reference

⚠ CAUTION

Loss of machine reference

The control cannot detect all situations that can lead to loss of the machine reference. Therefore, it is the general responsibility of the commissioning engineer or user to initiate explicit referencing of zero marker synchronization in such cases.

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 machine reference has been lost, it must first be restored through an adjustment or referencing of the lost machine reference.

See also

R1: Referencing (Page 1223)

8.4.4 Speed setpoint output

Control direction and travel direction of the feed axes

You must determine the travel direction of the feed axis before starting work.

Control direction

Before the position control is started up, the speed controller and current controller of the drive must be started up and optimized.

Travel direction

With the machine data:

MD32100 $MA_AX_MOTION_DIR (travel direction),

the direction of motion of the axis can be reversed,

without affecting the control direction of the position control.

Speed setpoint adjustment

SINUMERIK 840D sl

In the case of speed setpoint comparison, the NC is informed which speed setpoint corresponds to which motor speed in the drive, for parameterizing the axial control and monitoring. This comparison is carried out automatically.

For PROFIBUS DP drives, alternatively, the manual speed setpoint comparison is also possible.

- **Manual comparison**
  In the machine data:
  MD32250 $MA_RATED_OUTVAL
  a value not equal to zero is entered.

Note

Velocity adjustment and maximum speed setpoint

Owing to the automatic speed setpoint comparison a velocity adjustment is not necessary for SINUMERIK 840D sl!

Maximum speed setpoint

For SINUMERIK 840D sl, the maximum speed setpoint is defined as a percentage. 100% means maximum speed setpoint or maximum speed for PROFIdrive drives (manufacturer-specific setting parameters in the drive, e.g. p1082 for SINAMICS).

The output of the spindle speed is implemented in the NC for SINUMERIK 840D sl.

Data for five gear stages are realized in the controller.

These stages are defined by a minimum and maximum speed for the stage itself and by a minimum and maximum speed for the automatic gear stage changeover. A new set gear stage is output only if the new programmed speed cannot be traversed in the current gear stage.

With the machine data:

MD36210 $MA_CTRLOUT_LIMIT[n] (maximum speed setpoint)

the speed setpoint is restricted percentage-wise

Values up to 200% are possible.

When the speed is exceeded, an alarm is generated.

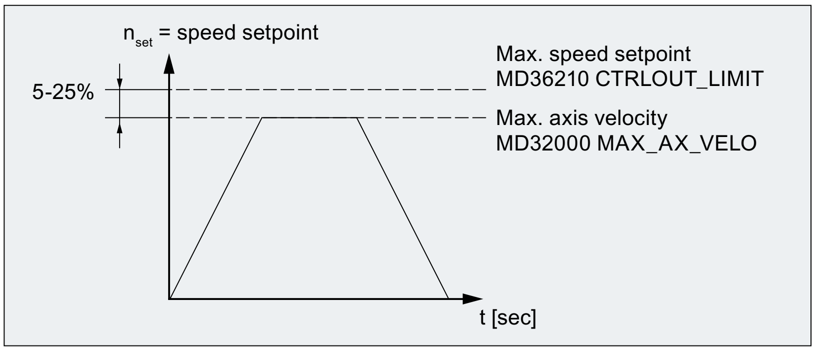


Figure 8-3 Maximum speed setpoint

However, due to control processes, the axes should not reach their maximum velocity (MD32000 $MA_MAX_AX_VELO) at 100% of the speed setpoint, but at 80% to 95%.

In case of axes, whose maximum speed is attained at around 80% of the speed setpoint range, the default value (80%) of the machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

can be taken over.

8.4.5 Machine data of the actual value system

Axis-specific machine data

To parameterize the actual value system, the following axis-specific machine data should be set:

Encoder and parameter set-independent machine data: $MA_	Meaning
MD30200 NUM_ENCS	Number of encoders
MD30300 IS_ROT_AX	Rotary axis / spindle
MD30310 ROT_IS_MODULO	Modulo conversion for rotary axis / spindle
MD30320 DISPLAY_IS_MODULO	360-degree modulo display for rotary axis or spindle
MD30330 MODULO_RANGE	Size of the modulo range
MD30340 MODULO_RANGE_START	Start position of the modulo range
MD31030 $MA_LEADSCREW_PITCH	Leadscrew pitch
MD31064 $MA_DRIVE_AX_RATIO2_DENOM	Intermediate gear denominator
MD31066 $MA_DRIVE_AX_RATIO2_NUMERA	Intermediate gear numerator
MD32000 $MA_MAX_AX_VELO	Maximum axis velocity

Encoder-dependent machine data: $MA_	Meaning
MD30210 ENC_SEGMENT_NR[n]	Actual value assignment: Number of bus segments
MD30220 ENC_MODULE_NR[n]	Actual value assignment: Drive number/ measuring circuit number
MD30230 ENC_INPUT_NR[n]	Actual value assignment: Input on drive module/measuring circuit module
MD30240 ENC_TYPE[n]	Encoder type of the actual value acquisition (actual position value)
MD30242 ENC_IS_INDEPENDENT[n]	Encoder is independent
MD30244 ENC_MEAS_TYPE[n]	Encoder measuring type
MD30250 ACT_POS_ABS[n]	Internal encoder position
MD30260 ABS_INC_RATIO[n]	Absolute encoder: Ratio between the absolute resolution and the incremental resolution
MD30270 ENC_ABS_BUFFERING[n]	Absolute encoder: Traversing range extension
MD34090 $MA_REFP_MOVE_DIST_CORR[n]	Reference point offset
MD34320 $MA_ENC_INVERS[n]	Length measuring system is in the opposite sense
n: Encoder index, with n = 0, 1, ... (1st encoder, 2nd encoder, ...)	

Note

The "Activate machine data" can be activated either in the part program with the command `NEWCONF` or via the user interface by pressing a softkey.

8.4.6 Actual-value resolution

8.4.6.1 Machine data of the actual value resolution

Depending where the measuring system (encoder) is mounted, the following measuring system types should be taken into account:

- Load-side encoder: Direct measuring system (DM)
- Motor-side encoder: Indirect measuring system (IM)

Parameterizing the actual value resolution depending on the axis type (linear/rotary axis)

The control system calculates the actual value resolution based on the following machine data.

Machine data for calculating the actual value resolution	Linear axis	Linear axis		Rotary axis	
	Linear scale / direct measuring system	Indirect measuring system	Direct measuring system: Machine/ tool	Indirect measuring system	Direct measuring system: Machine/tool
MD30300 $MA_IS_ROT_AX	0	0	0	1	1
MD31000 $MA_ENC_IS_LINEAR[n] MD31010 $MA_ENC_GRID_POINT_DIST[n] MD34320 $MA_ENC_INVERS[n]	1 Spacing 1)	0 - -	0 - -	0 - -	0 - -
MD31020 $MA_ENC_RESOL[n]	-	Pulses/ rev	Pulses/ rev	Pulses/ rev	Pulses/ rev
MD31025 $MA_ENC_PULSE_MULT[n]	Encoder multiplication				
MD31030 $MA_LEADSCREW_PITCH	-	mm/rev.	mm/rev.	-	-
MD31040 $MA_ENC_IS_DIRECT[n]	- / 1	0	1	0	1
MD31044 $MA_ENC_IS_DIRECT2[n]	- / 1	0	1	0	1
MD31050 $MA_DRIVE_AX_RATIO_DENOM[n]	-	Load rev.	- / 1	Load rev.	2)
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[n]	-	Motor rev. if infeed gear available	- / 1	Motor rev.	2)
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 3)	Motor rev.	Motor rev.	Load rev.

- Not relevant

1) For distance-coded measuring systems

2) These machine data are not required for encoder matching (path evaluation). However, they must be entered correctly for the setpoint calculation! Otherwise the required servo gain factor (K_V) will not be set. The load revolutions are entered into machine data MD31050 $MA_DRIVE_AX_RATIO_DENOM and the motor revolutions in machine data MD31060 $MA_DRIVE_AX_RATIO_NUMERA.

3) 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.

Machine data of the actual value resolution

The actual-value resolution results from the design of the machine, whether gearboxes are available and their gear ratio, the leadscrew pitch for linear axes and the resolution of the encoder being used. The following machine data must be set for this on the control system:

Encoder and parameter set-independent machine data: $MA_	Meaning
MD30300 IS_ROT_AX	Axis is a rotary axis / spindle
MD31010 ENC_GRID_POINT_DIST	Distance between reference marks of the linear scale
MD31030 LEADSCREW_PITCH	Leadscrew pitch
MD31064 DRIVE_AX_RATIO2_DENOM	Denominator of attached gearbox
MD31066 DRIVE_AX_RATIO2_NUMERA	Numerator of attached gearbox

Encoder-dependent machine data: $MA_	Meaning
MD31000 ENC_IS_LINEAR[n]	Measuring system is a linear scale
MD31020 ENC_RESOL[n]	Encoder pulses per revolution
MD31025 ENC_PULSE_MULT[n]	Encoder multiplication (high resolution)
MD31040 ENC_IS_DIRECT[n]	Direct or indirect measuring system
MD31044 ENC_IS_DIRECT2[n]	Encoders installed at the attached gearbox
MD31070 DRIVE_ENC_RATIO_DENOM[n]	Measuring gearbox denominator
MD31080 DRIVE_ENC_RATIO_NUMERA[n]	Measuring gearbox numerator
n: Encoder index, with n = 0, 1, ... (1st encoder, 2nd encoder, etc.)	

Parameter-set-dependent machine data: $MA_	Meaning
MD31050 $MA_DRIVE_AX_RATIO_DENOM[m]	Denominator load gearbox
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[m]	Numerator load gearbox
m: Parameter set index, with m = 0, 1, ... (1st parameter set, 2nd parameter set, etc.)	

Parameterizing the computational resolution

The ratio of control-internal computational resolution to the actual-value resolution is an indication of how exactly the values calculated by the control system can be implemented on the machine.

Computational resolution: Linear axes

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} = \frac{\text{Internal increments / (mm)}}{\text{Encoder increments / (mm)}}$$

Computational resolution: Rotary axes

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} = \frac{\text{Internal increments / (degree)}}{\text{Encoder increments / (degree)}}$$

Machine data

General machine data: $MN_	Meaning
MD10200 INT_INCR_PER_MM	Computational resolution for linear positions
MD10210 INT_INCR_PER_DEG	Computational resolution for angular positions

Recommended setting

The above components and settings that are responsible for the actual-value resolution, should be selected so that the actual-value resolution is higher than the parameterized computational resolution.

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} \leq 1$$

8.4.6.2 Example: Linear axis with linear scale

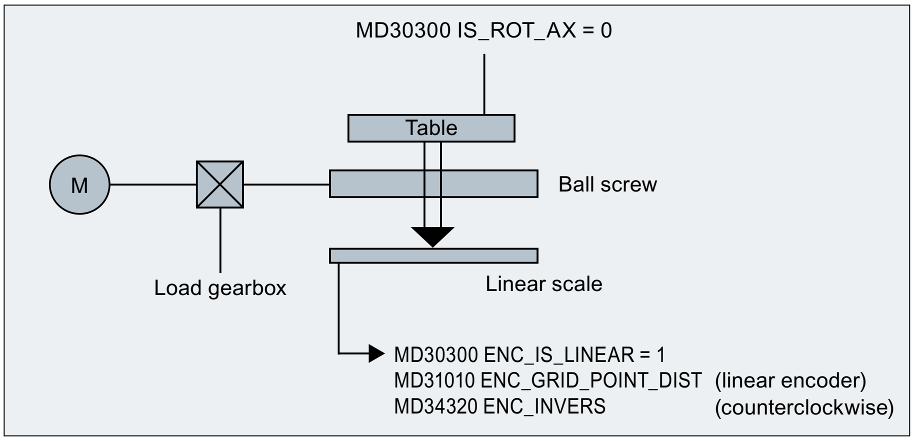

MD30300 IS_ROT_AX = 0
Table
M
Ball screw
Load gearbox
Linear scale
MD30300 ENC_IS_LINEAR = 1
MD31010 ENC_GRID_POINT_DIST (linear encoder)
MD34320 ENC_INVERS (counterclockwise)

Figure 8-4 Linear axis with linear scale

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{\text{ENC\_GRID\_POINT\_DIST [n]} \; * \; \text{INT\_INCR\_PER\_MM}}{\text{ENC\_PULSE\_MULT[n]}}$$

8.4.6.3 Example: Linear axis with rotary encoder on motor

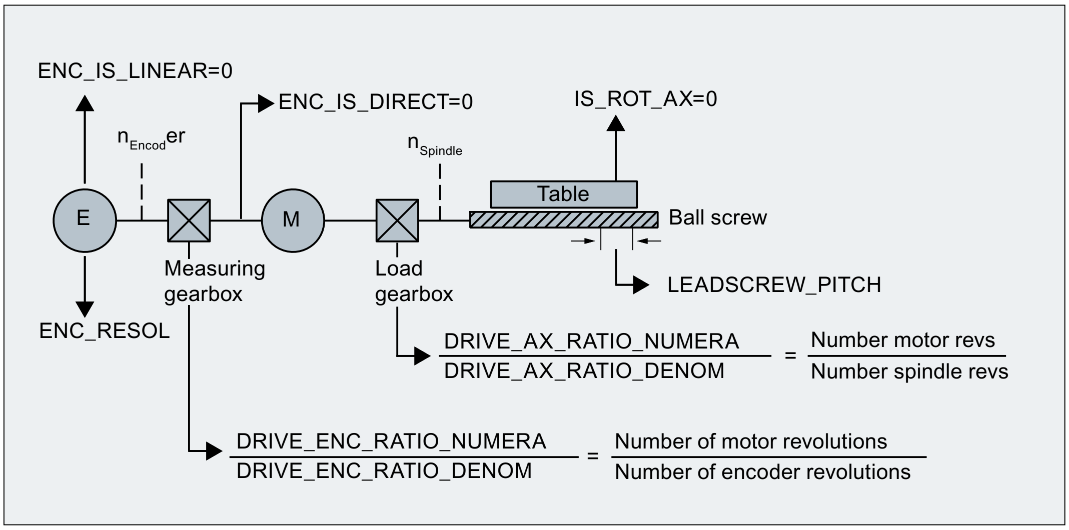

ENC_IS_LINEAR=0
ENC_IS_DIRECT=0
IS_ROT_AX=0
nEncoder
nSpindle
E
M
Table
Ball screw
Measuring gearbox
Load gearbox
LEADSCREW_PITCH
ENC_RESOL
DRIVE_AX_RATIO_NUMERA / DRIVE_AX_RATIO_DENOM = Number motor revs / Number spindle revs
DRIVE_ENC_RATIO_NUMERA / DRIVE_ENC_RATIO_DENOM = Number of motor revolutions / Number of encoder revolutions

Figure 8-5 Linear axis with rotary encoder on motor

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \frac{\text{DRIVE\_AX\_RATIO\_DENOM [n]}}{\text{DRIVE\_AX\_RATIO\_NUMERA [n]}} * \text{LEADSCREW\_PITCH} * \text{INT\_INCR\_PER\_MM}$$

Example

Assumptions:

- Rotary encoder on the motor: 2048 pulses/revolution
- Internal pulse multiplication: 2048
- Gearbox, motor / ball screw: 5:1
- Leadscrew pitch: 10 mm/revolution
- Computational resolution: 10000 increments per mm

Machine data	Value
MD30300 $MA_IS_ROT_AX	0
MD31000 $MA_ENC_IS_LINEAR[0]	0
MD31040 $MA_ENC_IS_DIRECT[0]	0
MD31020 $MA_ENC_RESOL[0]	2048
MD31025 $MA_ENC_PULSE_MULT	2048
MD31030 $MA_LEADSCREW_PITCH	10
MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[0]	1
MD31070 $MA_DRIVE_ENC_RATIO_DENOM[0]	1
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[0]	5
MD31050 $MA_DRIVE_AX_RATIO_DENOM[0]	1
MD10210 $MN_INT_INCR_PER_DEG	10000

$$\Rightarrow \frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{2048 * 2048} * \frac{1}{1} * \frac{1}{5} * 10\ \text{mm}$$
$$*10000\ \text{incr./mm} = 0.004768$$

An encoder increment corresponds to 0.004768 internal increments or 209.731543 encoder increments correspond to an internal increment.

8.4.6.4 Example: Linear axis with rotary encoder on the machine

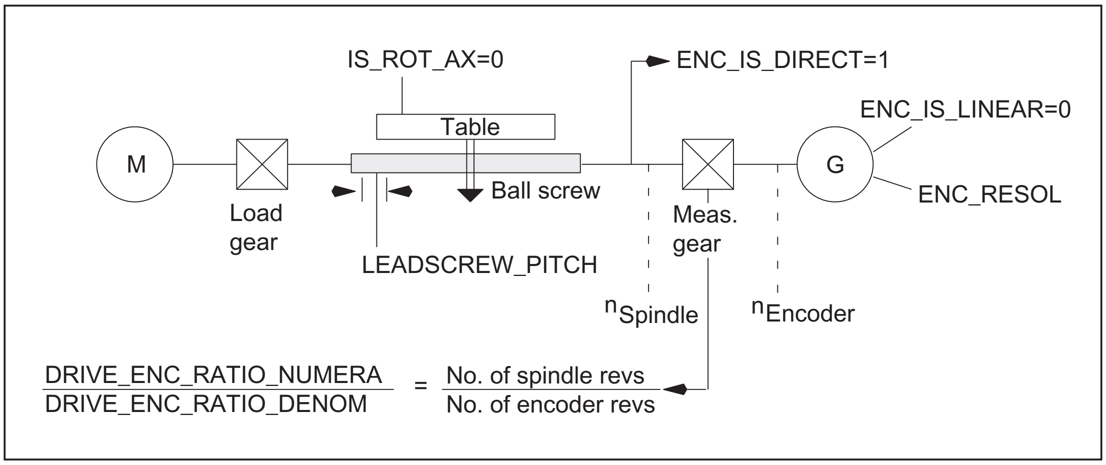


Figure 8-6 Linear axis with rotary encoder on the machine

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

$$\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \text{LEADSCREW\_PITCH} * \text{INT\_INCR\_PER\_MM}$$

8.4.6.5 Example: Rotary axis with rotary encoder on motor

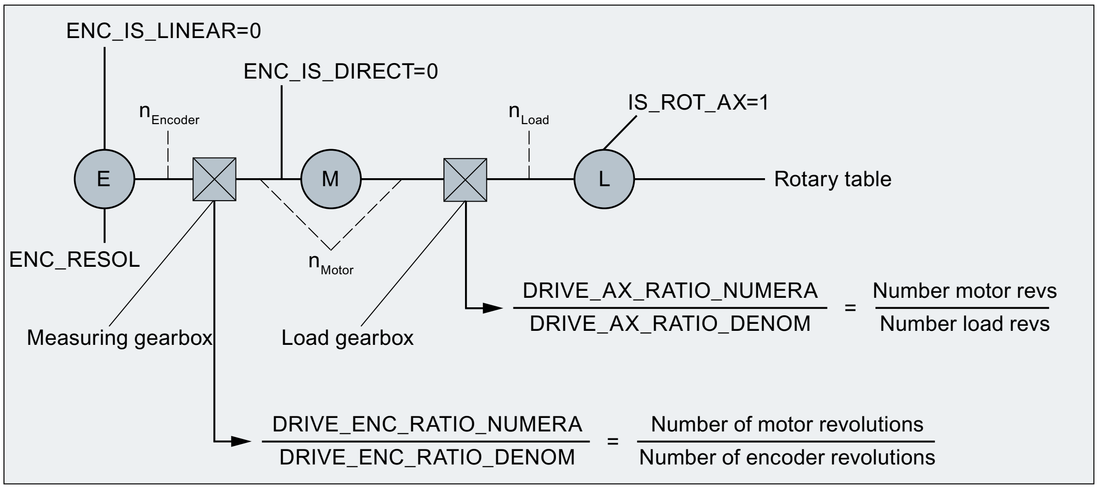

ENC_IS_LINEAR=0
ENC_IS_DIRECT=0
n_Encoder
n_Load
IS_ROT_AX=1
E
M
L
Rotary table
ENC_RESOL
n_Motor
DRIVE_AX_RATIO_NUMERA / DRIVE_AX_RATIO_DENOM = Number motor revs / Number load revs
Measuring gearbox
Load gearbox
DRIVE_ENC_RATIO_NUMERA / DRIVE_ENC_RATIO_DENOM = Number of motor revolutions / Number of encoder revolutions

Figure 8-7 Rotary axis with rotary encoder on motor

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

$$\frac{\text{Internal increments/degree}}{\text{Encoder increments/degree}} = \frac{\text{360 degrees}}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \frac{\text{DRIVE\_AX\_RATIO\_DENOM [n]}}{\text{DRIVE\_AX\_RATIO\_NUMERA [n]}} * \text{INT\_INCR\_PER\_DEG}$$

Example

Assumptions:

- Rotary encoder on the motor: 2048 pulses/revolution
- Internal pulse multiplication: 2048
- Gearbox, motor / rotary axis: 5:1
- Computational resolution: 1000 increments per degree

Machine data	Value
MD30300 $MA_IS_ROT_AX	1
MD31000 $MA_ENC_IS_LINEAR[0]	0
MD31040 $MA_ENC_IS_DIRECT[0]	0
MD31020 $MA_ENC_RESOL[0]	2048
MD31025 $MA_ENC_PULSE_MULT	2048
MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[0]	1
MD31070 $MA_DRIVE_ENC_RATIO_DENOM[0]	1
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[0]	5
MD31050 $MA_DRIVE_AX_RATIO_DENOM[0]	1
MD10210 $MN_INT_INCR_PER_DEG	1000

$$\Rightarrow \frac{\text{Internal increments/degree}}{\text{Encoder increments/degree}} = \frac{\text{360 degrees}}{2048 * 2048} * \frac{1}{1} * \frac{1}{5} * 1000 \text{ inc./degree} = 0{,}017166$$

An encoder increment corresponds to 0.017166 internal increments or 58.254689 encoder increments correspond to an internal increment.

8.4.6.6 Example: Rotary axis with rotary encoder on the machine

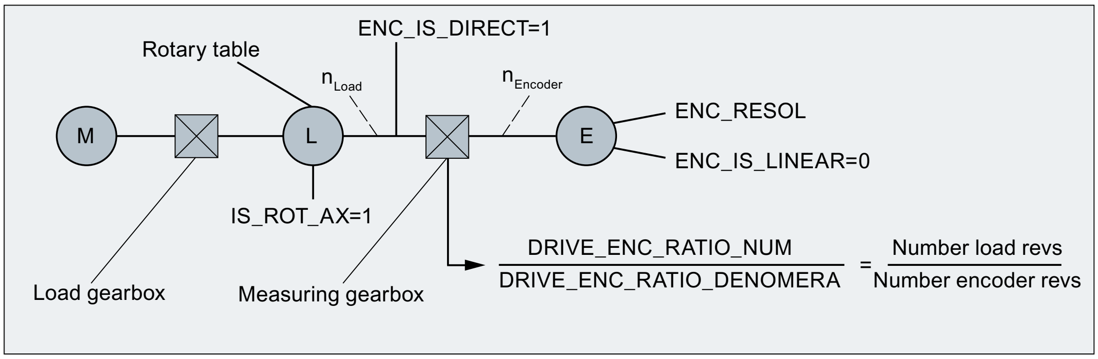


Figure 8-8 Rotary axis with rotary encoder on the machine

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

$$\frac{\text{Internal increments/degree}}{\text{Encoder increments/degree}} = \frac{\text{360 degrees}}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \text{INT\_INCR\_PER\_DEG}$$

8.4.6.7 Example: Intermediate gear with encoder on the tool

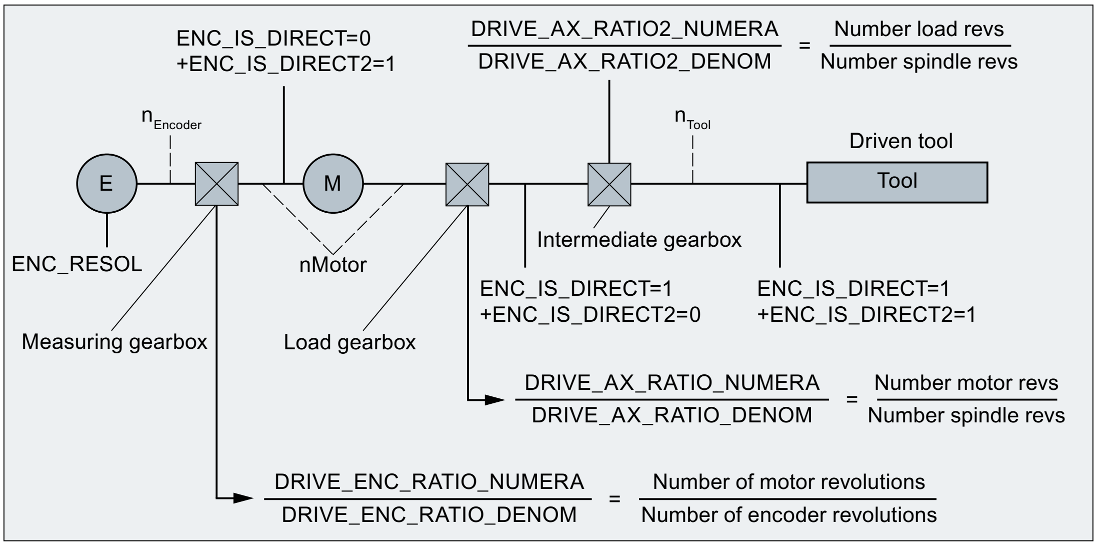

ENC_IS_DIRECT=0
+ENC_IS_DIRECT2=1
DRIVE_AX_RATIO2_NUMERA / DRIVE_AX_RATIO2_DENOM = Number load revs / Number spindle revs
n_Encoder
n_Tool
Driven tool
E
M
Tool
ENC_RESOL
nMotor
Intermediate gearbox
ENC_IS_DIRECT=1
+ENC_IS_DIRECT2=0
ENC_IS_DIRECT=1
+ENC_IS_DIRECT2=1
Measuring gearbox
Load gearbox
DRIVE_AX_RATIO_NUMERA / DRIVE_AX_RATIO_DENOM = Number motor revs / Number spindle revs
DRIVE_ENC_RATIO_NUMERA / DRIVE_ENC_RATIO_DENOM = Number of motor revolutions / Number of encoder revolutions

Figure 8-9 Intermediate gear with encoder directly on the rotating tool

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

$$\frac{\text{Internal increments/degree}}{\text{Encoder increments/degree}} = \frac{\text{360 degrees}}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \text{INT\_INCR\_PER\_DEG}$$

8.5 Closed-loop control

8.5.1 General information

Position control of an axis/spindle

The closed-loop control of an axis consists of the current and speed control loop of the drive plus a higher-level position control loop in the NC.

The basic structure of an axis/spindle position control is illustrated below:

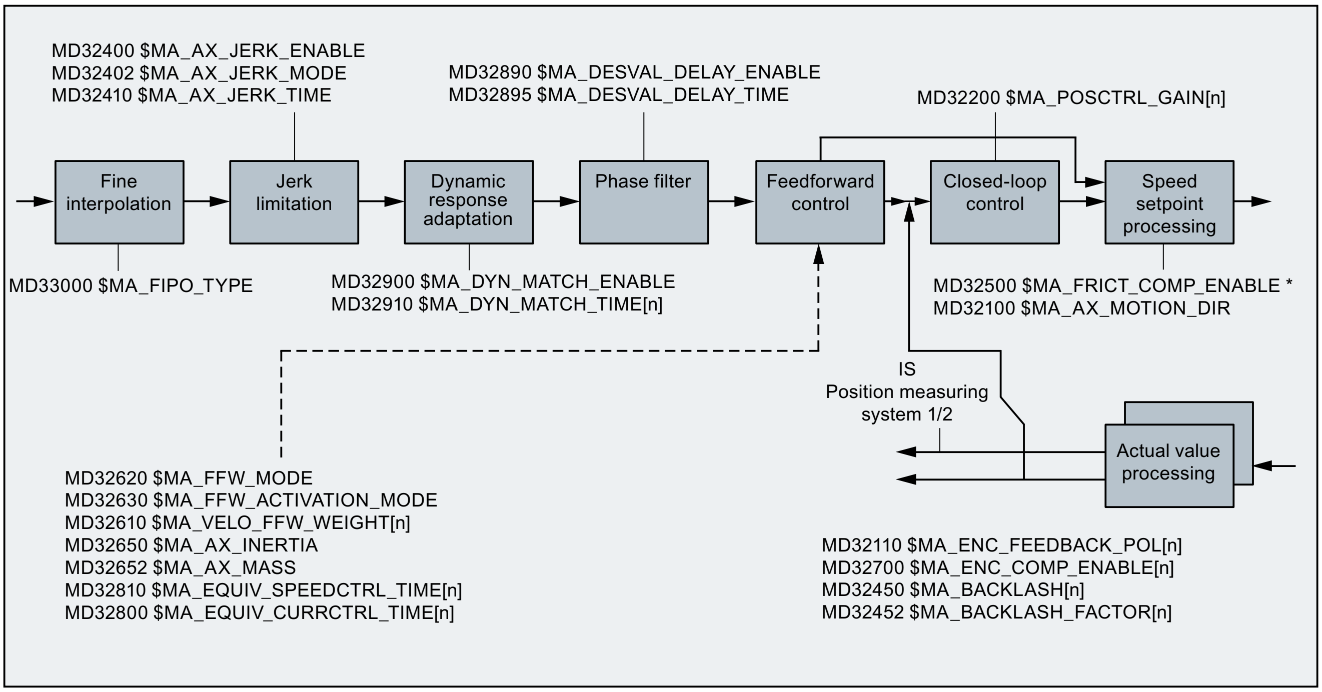

MD32400 $MA_AX_JERK_ENABLE
MD32402 $MA_AX_JERK_MODE
MD32410 $MA_AX_JERK_TIME
MD32890 $MA_DESVAL_DELAY_ENABLE
MD32895 $MA_DESVAL_DELAY_TIME
MD32200 $MA_POSCTRL_GAIN[n]
Fine interpolation
Jerk limitation
Dynamic response adaptation
Phase filter
Feedforward control
Closed-loop control
Speed setpoint processing
MD33000 $MA_FIPO_TYPE
MD32900 $MA_DYN_MATCH_ENABLE
MD32910 $MA_DYN_MATCH_TIME[n]
MD32500 $MA_FRICT_COMP_ENABLE *
MD32100 $MA_AX_MOTION_DIR
IS
Position measuring system 1/2
Actual value processing
MD32620 $MA_FFW_MODE
MD32630 $MA_FFW_ACTIVATION_MODE
MD32610 $MA_VELO_FFW_WEIGHT[n]
MD32650 $MA_AX_INERTIA
MD32652 $MA_AX_MASS
MD32810 $MA_EQUIV_SPEEDCTRL_TIME[n]
MD32800 $MA_EQUIV_CURRCTRL_TIME[n]
MD32110 $MA_ENC_FEEDBACK_POL[n]
MD32700 $MA_ENC_COMP_ENABLE[n]
MD32450 $MA_BACKLASH[n]
MD32452 $MA_BACKLASH_FACTOR[n]

Figure 8-10 Principle representation of the setpoint processing and closed-loop control

For information on the jerk limitation, see Section "B2: Acceleration (Page 267)".

For a description of the feedforward control, backlash, friction compensation, and leadscrew error compensation.

References:
Function Manual Extended Functions; Compensations (K3)

Fine interpolation

Using the fine interpolator (FIPO), the contour precision can be further increased by reducing the staircase effect in the speed setpoint. You can set three different types of fine interpolation:

MD33000 $MA_FIPO_TYPE = <FIPO mode>

<FIPO mode>	Meaning
1	Differential fine interpolation with mean value generation (smoothing) over an IPO cycle
2	Cubic fine interpolation
3	Cubic fine interpolation optimized for use with the pre-control for the highest contour precision

K_V factor

In order that few contour deviations occur in the continuous-path mode, a high servo gain factor (K_V) is necessary:

MD32200 $MA_POSCTRL_GAIN[n]

However, if the servo gain factor (K_V) is too high, instability, overshoot and possibly impermissible high loads on the machine will result.

The maximum permissible servo gain factor (K_V) depends on the following:

- Design and dynamics of the drive
  (rise time, acceleration and braking capacity)
- Machine quality
  (elasticity, oscillation damping)
- Position control cycle or speed control cycle for active DSC

The servo gain factor (K_V) is defined as:

$$K_v = \frac{\text{Velocity}}{\text{Following error}} \quad ; \quad \frac{[\text{m/min}]}{[\text{mm}]}$$

Dynamic response adaptation

Axes that interpolate with one another, but with different servo gain factors (K_V) can be set to the same following error using the dynamic adaptation function. This allows an optimum contour accuracy to be achieved without loss of control quality by reducing the servo gain factors (K_V) to the dynamically weakest axis.

The function is activated via:

MD32900 $MA_DYN_MATCH_ENABLE = 1 (dynamic response adaptation)

The dynamic response adaptation is realized by entering a new equivalent time constant. It is calculated from the difference in the equivalent time constant of the dynamically weakest axis and the axis to be adapted:

MD32910 $MA_DYN_MATCH_TIME [n] = <difference in the equivalent time constant>

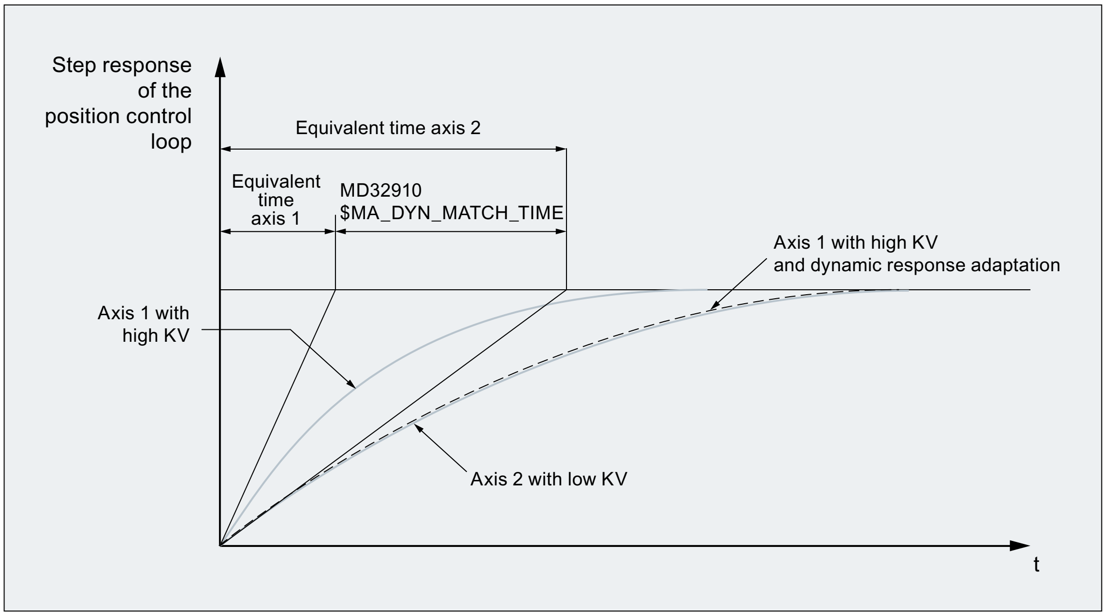


Figure 8-11 Dynamic response adaptation

Example of a dynamic response adaptation of three axes without speed feedforward control

The equivalent time constant of the position control loop is:	
Axis 1:	30 ms
Axis 2:	20 ms
Axis 3:	24 ms

With an equivalent time constant of 30 ms, axis 1 is the dynamically weakest axis.

This results in the following new equivalent time constants for the axes:

Axis 1:	MD32910 $MA_DYN_MATCH_TIME = 0 ms
Axis 2:	MD32910 $MA_DYN_MATCH_TIME = 30 ms - 20 ms = 10 ms
Axis 3:	MD32910 $MA_DYN_MATCH_TIME = 30 ms - 24 ms = 6 ms

Approximation formulas for the equivalent time constant of the position control loop of an axis

The equivalent time constant T_{equiv} of the position control loop of an axis is approximately calculated depending on the type of feedforward control:

- Without feedforward control:

$$T_{Equiv} \approx \frac{1}{\text{MD32200 POSCTRL\_GAIN [1/s]}}$$

- With speed feedforward control:

  T_{Equiv} ≈ MD32810 EQUIV_SPEEDCTRL_TIME

- For combined torque/speed feedforward control

  T_{Equiv} ≈ MD32800 EQUIV_CURRCTRL_TIME

Note

If dynamic response adaptation is realized for a geometry axis, then all other geometry axes must be set to the same dynamic response.

References:
Commissioning Manual CNC: NC, PLC, Drives

8.5.2 Parameter sets of the position controller

Six parameter sets per machine axis are available to quickly adapt the position control to the changed machine properties during operation, e.g. a gear change of the spindle, or to adjust the dynamic response to another axis, e.g. during tapping.

Machine data

A parameter set comprises the following machine data:

Number	Identifier $MA_	Meaning
31050	DRIVE_AX_RATIO_DENOM	Denominator load gearbox
31060	DRIVE_AX_RATIO_NUMERA	Numerator load gearbox
32200	POSCTRL_GAIN	Servo gain factor (K_V)
32452	BACKLASH_FACTOR	Backlash compensation
32610	VELO_FFW_WEIGHT	Feedforward control factor
36012	STOP_LIMIT_FACTOR	Exact stop coarse/fine factor and zero speed
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant, current control loop for feed-forward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant, speed control loop for feed forward control
32910	DYN_MATCH_TIME	Time constant for dynamic response adaptation
36200	AX_VELO_LIMIT	Threshold value for velocity monitoring

Tapping, thread cutting

For tapping or thread cutting, the following applies with regard to the parameter sets of axes:

- For machine axes that are **not** involved in tapping or thread cutting, parameter set 1 (index = 0) is active. The other parameter sets do not have to be taken into account.
- For machine axes that are involved in tapping or thread cutting, the same parameter set number as that of the current gear stage of the spindle is active.
  All parameter sets correspond to the gear stages and must therefore be parameterized.

The current parameter set is displayed on the user interface at:

Operating Area "Diagnostics" > "Service axis"

Parameter sets during gear stage change

Each gear stage of a spindle is assigned a separate parameter set. The gear stage is selected via the following NC/PLC interface signal:

DB31, ... DBX16.0 - 2 = <actual gear stage>

Actual gear stage	DB31, ... DBX16.0 - 2	Parameter set	
1st gear stage	000	2	(Index=1)
1st gear stage	001	2	(Index=1)
2nd gear stage	010	3	(Index=2)
3rd gear stage	011	4	(Index=3)
4th gear stage	100	5	(Index=4)
5th gear stage	101 110 111	6	(Index=5)

For further information on gear stages for spindles, see Section "S1: Spindles (Page 1273)".

8.6 Optimization of the control

8.6.1 Position controller, position setpoint filter: Balancing filter

Function

With feedforward control active, the position setpoint is sent through a so-called balancing filter before it reaches the controller itself. It is thus possible to control the speed setpoint to 100% in advance, without resulting in overshoots when positioning.

Activation

The feedforward control variant is selected and so also the filter activated using the axis-specific machine data:

MD32620 $MA_FFW_MODE (feedforward control mode)

Value	Meaning
3	Speed precontrol
4	Combined torque/speed precontrol

Activating and deactivating via the part program

Part programs can be used to activate and deactivate the feedforward control for all axes, using commands `FFWON` and `FFWOF`.

If the feedforward control of the individual axes should not be influenced by `FFWON`/`FFWOF`, the setting in the following machine data must be changed for these axes:

MD32630 $MA_FFW_ACTIVATION_MODE (activate feedforward control from program)

Parameterization

Recommended setting for recommissioning

If recommissioning, or if default values have been loaded (switch position 1 on commissioning switch and POWER ON), the following machine data default values apply:

- MD32620 $MA_FFW_MODE = 3
- MD32610 $MA_VELO_FFW_WEIGHT (feedforward control factor for the velocity feedforward control) = 1

The balancing time for the speed feedforward control then just has to be adjusted in the following machine data:

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

Setting the equivalent time constant of the speed control loop (MD32810)

We recommend that the axis be allowed to move in and out in "AUTOMATIC" mode with a part program and that travel-in to the target position, i.e. the actual position value of the active measuring system, be monitored with the servo trace.

The initial value for setting is the time constant of the speed control loop. This can be read from the reference frequency characteristic of the speed control loop. In the frequent case of a PI controller with speed setpoint smoothing, an approximate equivalent time can be read from drive machine data p1414, p1415, p1416 and p1421.

This start value (e.g. 1.5 ms) is now entered:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME = 0.0015

The axis then travels forward and backward and the operator monitors a greatly-magnified characteristic of the actual position value at the target position.

The following rules apply to making manual fine adjustments:

Monitoring	Measure	Effect
Overshoot	Increase MD32810	The position controller responds more slowly. ⇒ The tendency to overshoot decreases. Problem: The contour error (deviation from the programmed contour) increases at curves.
Excessively slow approach	Reduce MD32810	The position controller responds faster. ⇒ The contour error at curves decreases. Problem: The tendency to overshoot increases.

Note

The effects are similar to modification of the position controller gain (MD32200 $MA_POSCTRL_GAIN). They can be observed on the user interface in the operating section "Diagnostics" > "Axis service" using the computed K_V value.

MD32810 fine adjustment

Therefore, MD32810 should be assigned as small a value as possible, with the overshoot setting the limit during positioning. Furthermore, the initial value is only modified slightly during fine adjustment, typically by adding or deducting 0.25 ms.

For example, if the initial value is 1.5 ms, the optimum value calculated manually is usually within the range 1.25 ms to 1.75 ms.

In the case of axes equipped with direct measuring systems (load encoders) and strong elasticity, you may possibly accept small overshoots of some micrometers. These can be reduced with the help of the position setpoint filter for dynamic response adaptation (MD32910 $MA_DYN_MATCH_TIME) and for jerk (MD32410 $MA_AX_JERK_TIME), which also reduces the axis speed.

Identical axis data within an interpolation group

All the axes within an interpolation group should have **identical settings** in the following data:

- MD32200 $MA_POSCTRL_GAIN (adapted using MD32910)
- MD32620 $MA_FFW_MODE
- MD32610 $MA_VELO_FFW_WEIGHT
- MD32810 $MA_EQUIV_SPEEDCTRL_TIME (or MD32800 $MA_EQUIV_CURRCTRL_TIME) (dependent on the mechanical system and drive)
- MD32400 $MA_AX_JERK_ENABLE
- MD32402 $MA_AX_JERK_MODE
- MD32410 $MA_AX_JERK_TIME

The servo gain display (K_V) of the user interface in operating section "Diagnostics" > "Axis service" is used for checking.

Non-identical axis data within an interpolation group

If identical values are not possible for the above data, the following machine data can be used to make an adjustment:

MD32910 $MA_DYN_MATCH_TIME (time constant of dynamic response adaptation)

This allows the same servo gain value (K_V) to be displayed.

Different servo gain display values (K_V) usually point to the following:

- The gear ratios do not match in one or several axes.
- The feedforward control setting data does not match.

Setting the equivalent time constant of the current control loop (MD32800)

The activation of the torque feedforward control filter is performed with:

MD32620 $MA_FFW_MODE = 4

The same rules and recommendations apply to setting the time constant of the current control loop MD32800 $MA_EQUIV_CURRCTRL_TIME as to the speed feedforward control.

Limitation to stiff machines

Experience has shown that this expenditure is only worthwhile in the case of very stiff machines, and requires appropriate experience. The elasticities of the machine are often excited due to the injection of the torque so strongly that the existing vibrations neutralize the gain in contour accuracy.

In this case, it would be worth trying the function "Dynamic Stiffness Control" (DSC) as an alternative:

MD32640 $MA_STIFFNESS_CONTROL_ENABLE = 1

Note

The following conditions apply to use of DSC:

- In the NC, no actual value inversion (MD32110 $MA_ENC_FEEDBACK_POL = -1) must be parameterized. Actual-value inversion in the drive (SINAMICS parameter p0410) is permissible.
- The function must be activated for all axes that are in an interpolation relationship (path motion or coupling) (MD32640 = 1).
- If one of the axes is parameterized as a simulated axis (MD30130 $MA_CTRLOUT_TYPE = 0, MD30240 $MA_ENC_TYPE = 0), DSC cannot be applied. In this case, the function must be deactivated for all axes that are in an interpolation relationship (MD32640 = 0).

⚠ CAUTION

Exception error condition

An exception error condition may occur if p1192 is set for a non-existing measuring system and DSC is activated!

For service situations in which an additional measuring system has to be disabled and DSC is active, it must be assured that p1192 is also adapted.

Control response with POWER ON, RESET, REPOS, etc.

In the case of POWER ON and RESET, as well as with "Enable machine data", the setting data of the feedforward control is read in again (see the appropriate values of the relevant machine data).

Mode change, block search and REPOS have **no** influence on the feedforward control.

8.6.2 Position controller, position setpoint filter: Jerk filter

Function

In some applications, such as when milling sculptured surfaces, it can be advantageous to smooth the position setpoint curves using the jerk filter. The objective to optimize the surface quality by minimizing the excitations of machine vibrations. The filter effect of the position setpoints must be as strong as possible without impermissibly affecting contour accuracy. The smoothing behavior of the filter must also be as "symmetrical" as possible, i.e. if the same contour was to be traversed both forward and backward, the characteristic rounded by the filter should be as similar as possible in both directions.

The effect of the filter can be monitored by means of the effective servo gain factor (K_V), which is displayed on the "Axis service" screen form. The filtering effect rounds the position setpoints slightly, thus reducing the path accuracy so that with increasing filter time a smaller effective servo gain factor (K_V) is displayed.

Note

The jerk filter creates a dependent phase offset for each amplitude setting. Only the additional use of the phase filter (see "Position controller, position setpoint filter: Phase filter (Page 392)") permits a transparent setting of the axis dynamic response.

Activation

The function of the axis-specific jerk filter setpoint must be activated with the following machine data:

MD32400 $MA_AX_JERK_ENABLE[<axis>] = "TRUE"

Parameterization

Filter type for axis-specific jerk limitation

The "floating averaging" filter type is normally used for the axis-specific jerk limitation:

MD32402 $MA_AX_JERK_MODE[<axis>] = **2**

Note

If filter type MD32402 $MA_AX_JERK_MODE = 2 was not activated previously, "Power On" must be initiated once. Otherwise, "Enable machine data" or "Reset" at the machine control panel are sufficient.

Time constant for the axis-specific jerk filter

The time constant of the axis-specific jerk filter [s] is set in the machine data:

MD32410 $MA_AX_JERK_TIME[<axis>]

Fine adjustment

The fine adjustment of the jerk filter is carried out as follows:

1. Assess the traversing response of the axis (e.g. based on positioning processes with servo trace).
2. Modify the filter time in MD32410 $MA_AX_JERK_TIME.
3. Activate the modified time via "Enable machine data" or RESET on the machine control panel.

Deactivation

Disabling the jerk filter:

1. Block the filter calculation:
   MD32400 $MA_AX_JERK_ENABLE = 0
2. Activate the interlock via "Enable machine data" or RESET on the machine control panel.

Boundary conditions

Filter times

The jerk filter is only effective when the time constant (MD32410) is greater than one position control cycle.

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 = 1 to MD32400 = 2, the displayed servo gain (K_V) can be reduced while retaining the same filter time, although the path accuracy improves.

Axes that are interpolating axes with one another

- 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.

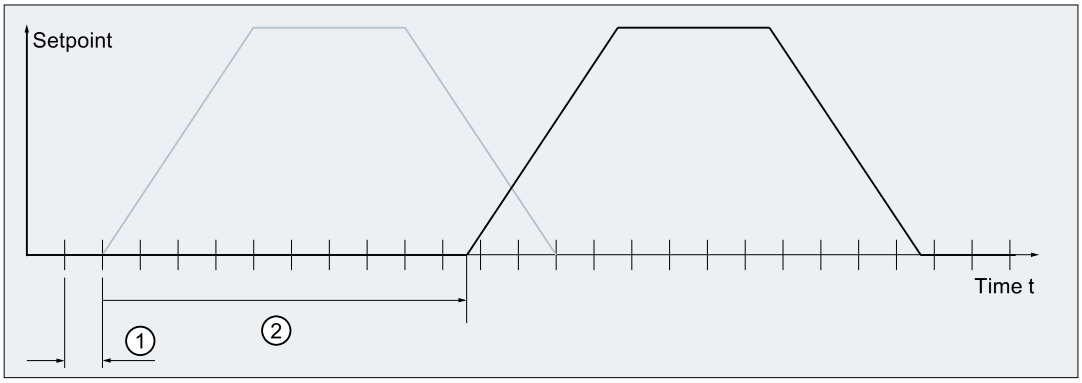
References

For further information on jerk limitation at the interpolator level, see Sections "Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) (Page 286)" and "Axis/spindlespecific machine data (Page 314)".

8.6.3 Position controller, position setpoint filter: Phase filter

Function

The axis-specific phase filter setpoint (dead time / delay) implements a pure phase shifter with which the setpoint phase response can be influenced. Together with the axis-specific jerk filter setpoint (MD32402_$MA_AX_JERK_MODE[<axis>] = 2; see Section "Position controller, position setpoint filter: Jerk filter (Page 390)") it is possible to adapt the amplitude response and the phase response independently of one another to the dynamically weakest of several axes, which should proceed along a programmed path together.



① Position controller cycle clock: 2 ms

② Time constant of the axis-specific phase filter setpoint: 19.2 ms

Figure 8-12 Effect of the axis-specific phase filter setpoint

Parameter assignment: Time constant

The time constant for the axis-specific phase filter setpoint can be set in the range:

(0 to 64) * position controller cycle clock (MD10061 $MN_POSCTRL_CYCLE_TIME)

The value of the time constants must be entered in seconds [s] in the machine data:

MD32895 $MA_DESVAL_DELAY_TIME[<Axis>]

Example: Position controller cycle clock: 2 ms ⇒ adjustable time constant: 0.0 to 0.128 s

Note

The time constant of the phase filter setpoint delays the axis' response characteristics for tapping, retractions, and exact stops / block changes, etc. It is therefore recommended to set the time constant as low as possible.

Limiting to a maximum value

A greater time constant than the permitted maximum value (64 * position controller cycle clock) is limited internally to the maximum value. No message / alarm is displayed.

Parameter assignment: Activation

The function of the axis-specific phase filter setpoint must be activated with the following machine data:

MD32890 $MA_DESVAL_DELAY_ENABLE[<Axis>] = "TRUE"

Examples

Assumption: Position controller cycle clock = 2 ms

1. MD32890 $MA_DESVAL_DELAY_ENABLE[<Axis>] = "FALSE"
   MD32895 $MA_DESVAL_DELAY_TIME[<Axis>] = <Time constant>
   - Phase filter setpoint: Not active
   - Time constant: Irrelevant
2. MD32890 $MA_DESVAL_DELAY_ENABLE[<Axis>] = TRUE
   MD32895 $MA_DESVAL_DELAY_TIME[<axis>] = 0.002
   - Phase filter setpoint: Active
   - Time constant: 2 ms ⇒ the setpoint output is delayed by a position controller cycle clock.
3. MD32890 $MA_DESVAL_DELAY_ENABLE[<Axis>] = TRUE
   MD32895 $MA_DESVAL_DELAY_TIME[<axis>] = 0.256
   - Phase filter setpoint: Active
   - Set time constant: 256 ms;
     maximum possible time constant: 64 * 2 ms = 128 ms
     ⇒ internally effective time constant: 128 ms

Supplementary conditions

SINUMERIK Safety Integrated

The phase filter setpoint delays the output of axis-specific setpoints; e.g. for retractions (ESR for Stop E). However, the phase filter setpoint has no influence over shutdown processes; e.g. the SBH activation time.

8.6.4 Position controller: injection of positional deviation

Preconditions

- 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:
  - Direct measuring system:
    MD31040 $MA_ENC_IS_DIRECT[1]=1
    The encoder for position actual-value acquisition is connected directly to 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).
- Telegram type 136 or 138 must be configured as standard telegram type for PROFIdrive both in the drive and also in the NC (MD13060 $MN_DRIVE_TELEGRAM_TYPE).

Function

For active injected positional deviation, the difference position between the direct and the indirect measuring system of an axis is determined and in accordance with the weighting-factor setting is applied as additional current setpoint for the feedforward control in the position control cycle. The resulting oscillation damping improves the stability and positioning behavior of the axis.

Application

The function is used for axes with strong tendency to vibrate.

Effectiveness

The function acts only for axes with small natural frequency (to approximately 20 Hz).

Activation/parameterization

The function is activated by specifying the weighting factor:

MD32950 $MA_POSCTRL_DAMPING (damping of the speed control loop) = <value>

Value range: -100% ... +100%

An input value "100%" means: A supplementary torque in accordance with SINAMICS parameter p2003 is applied when the determined position difference between the two measuring systems reaches the following value:

- With linear motors: 1 mm
- With linear axis with rotary motor: MD31030 $MA_LEADSCREW_PITCH (leadscrew pitch)
- For rotary axis/spindle: 360 degrees

Standard setting is 0. In this case, the injection of positional deviation is inactive.

Note

The weighting factor 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), the parameter is too large or the value has the incorrect sign.

8.6.5 Position control with proportional-plus-integral-action controller

Function

As standard, the core of the position controller is a P controller. It is possible to switch-in an integral component for special applications (such as an electronic gear). The resulting proportional-plus-integral-action controller then corrects the error between setpoint and actual positions down to zero in a finite, settable time period when the appropriate machine data is set accordingly.

⚠ CAUTION

Overshootings of the actual position for activated PI controller

In this instance, you must decide whether this effect is admissible for the application in question. Knowledge of the control technology and measurements with servo trace are an absolute prerequisite for using the function. If the appropriate machine data is incorrectly set, then machines could be damaged due to instability.

Procedure

1. First optimize the position control loop as a proportional-action controller 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.
   - $T_n \rightarrow \infty$:
     Effectiveness of the integral component is almost 0. Behavior of the controller like a pure proportional controller.
4. Find the right compromise for T_n between these two extreme cases for the application.

   Note

   T_n must not be chosen too near the stability limit because the occurrence of an instability can cause machine damage.

   It is therefore recommended to set T_n to no less than 1 s.

   Use servo trace to trace the travel-in of an automatic program traveling to and from a target position.
5. Set the servo trace to display the following:
   - Following error
   - Actual velocity
   - Actual position
   - Reference position
6. 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

Supplementary conditions

DSC

If the integrator function is used, DSC (Dynamic Stiffness Control) must be switched off.

Example

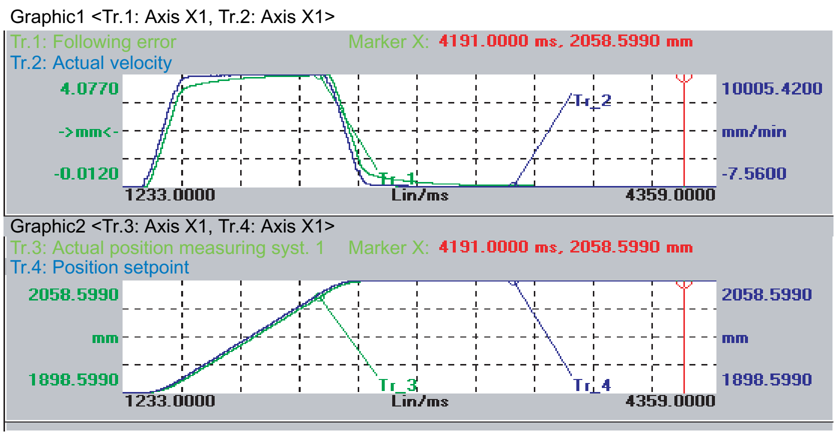
Setting result after several iterative processes for $_K$R and T$_n$.

Machine data settings:

- MD32220 $MA_POSCTRL_INTEGR_ENABLE = 1
- MD32210 $MA_POSCTRL_INTEGR_TIME = 0.003
- MD32200 $MA_POSCTRL_GAIN[1] = 5.0

Parameter set selection 0

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.



8.7 Data lists

8.7.1 Machine data

8.7.1.1 Displaying machine data

Number	Identifier: $MM_	Description
9004	DISPLAY_RESOLUTION	Display resolution
9010	SPIND_DISPLAY_RESOLUTION	Display resolution for spindles
9011	DISPLAY_RESOLUTION_INCH	Display resolution for INCH system of measurement

8.7.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB	Machine axis name
10050	SYSCLOCK_CYCLE_TIME	System basic cycle
10070	IPO_SYSCLOCK_TIME_RATIO	Factor for interpolator cycle
10060	POSCTRL_SYSCLOCK_TIME_RATIO	Factor for position-control cycle
10200	INT_INCR_PER_MM	Computational resolution for linear positions
10210	INT_INCR_PER_DEG	Computational resolution for angular positions
10220	SCALING_USER_DEF_MASK	Activation of scaling factors
10230	SCALING_FACTORS_USER_DEF	Scaling factors of physical quantities
10240	SCALING_SYSTEM_IS_METRIC	Basic system metric
10250	SCALING_VALUE_INCH	Conversion factor for switchover to inch system
10260	CONVERT_SCALING_SYSTEM	Basic system switchover active
10270	POS_TAB_SCALING_SYSTEM	Measuring system of position tables
10290	CC_TDA_PARAM_UNIT	Physical units of the tool data for CC
10292	CC_TOA_PARAM_UNIT	Physical units of the tool edge data for CC
13050	DRIVE_LOGIC_ADDRESS	Logical drive addresses
13060	DRIVE_TELEGRAM_TYPE	Standard message frame type for PROFIBUS DP
13070	DRIVE_FUNCTION_MASK	DP function used
13080	DRIVE_TYPE_DP	Drive type PROFIBUS DP

8.7.1.3 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of the G groups

8.7.1.4 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30110	CTRLOUT_MODULE_NR	Setpoint assignment: Drive number
30120	CTRLOUT_NR	Setpoint assignment: Setpoint output on drive module
30130	CTRLOUT_TYPE	Output type of setpoint
30200	NUM_ENCS	Number of encoders
30220	ENC_MODULE_NR	Actual value assignment: Drive module number
30230	ENC_INPUT_NR	Actual value assignment: Input on the drive module
30240	ENC_TYPE	Type of actual value acquisition (actual position value)
30242	ENC_IS_INDEPENDENT	Encoder is independent
30300	IS_ROT_AX	Rotary axis
31000	ENC_IS_LINEAR	Direct measuring system (linear scale)
31010	ENC_GRID_POINT_DIST	Distance between reference marks on linear scales
31020	ENC_RESOL	Encoder pulses per revolution
31030	LEADSCREW_PITCH	Leadscrew pitch
31040	ENC_IS_DIRECT	Encoder is connected directly to the machine
31044	ENC_IS_DIRECT2	Encoder on intermediate gear
31050	DRIVE_AX_RATIO_DENOM	Denominator load gear
31060	DRIVE_AX_RATIO_NUMERA	Numerator load gear
31064	DRIVE_AX_RATIO2_DENOM	Intermediate gear denominator
31066	DRIVE_AX_RATIO2_NUMERA	Intermediate gear numerator
31070	DRIVE_ENC_RATIO_DENOM	Measuring gear denominator
31080	DRIVE_ENC_RATIO_NUMERA	Measuring gear numerator
31090	JOG_INCR_WEIGHT	Weighting of increment for INC/handwheel
31200	SCALING_FACTOR_G70_G71	Factor for converting values when `G70`/`G71` is active
32000	MAX_AX_VELO	Maximum axis velocity
32100	AX_MOTION_DIR	Travel direction
32110	ENC_FEEDBACK_POL	Sign actual value (feedback polarity)
32200	POSCTRL_GAIN	Servo gain factor (K_V)
32210	POSCTRL_INTEGR_TIME	Integrator time position controller
32220	POSCTRL_INTEGR_ENABLE	Activation of integral component of position controller
32250	RATED_OUTVAL	Rated output voltage
32260	RATED_VELO	Rated motor speed
32450	BACKLASH	Backlash
32500	FRICT_COMP_ENABLE	Friction compensation active
32610	VELO_FFW_WEIGHT	Feedforward control factor for speed feedforward control
32620	FFW_MODE	Feedforward control mode
32630	FFW_ACTIVATION_MODE	Activate feedforward control from program
32650	AX_INERTIA	Moment of inertia for torque feedforward control
32652	AX_MASS	Axis mass for torque precontrol

Number	Identifier: $MA_	Description
32711	CEC_SCALING_SYSTEM_METRIC	System of measurement of sag compensation
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant current control loop for feed-forward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feed-forward control
32890	DESVAL_DELAY_ENABLE	Axis-specific phase filter setpoint
32895	DESVAL_DELAY_TIME	Time constant for the axis-specific phase filter setpoint
32900	DYN_MATCH_ENABLE	Dynamics matching
32910	DYN_MATCH_TIME [n]	Time constant for dynamic response adaptation
32930	POSCTRL_OUT_FILTER_ENABLE	Activation of low-pass filter at position controller output
32950	POSCTRL_DAMPING	Damping of the speed control loop
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	Axis-specific jerk limitation
36410	AX_JERK_TIME	Time constant for the axis-specific jerk filter
36500	ENC_CHANGE_TOL	Max. tolerance for actual position value switching
36510	ENC_DIFF_TOL	Measuring system synchronism tolerance
36700	ENC_COMP_ENABLE[n]	Interpolatory compensation

8.7.2 Signals

8.7.2.1 Signals from the NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Change counter, inch/metric system of units	DB10.DBB71	DB2700.DBB0015
Inch system of units	DB10.DBX107.7	DB2700.DBX1.7

8.7.2.2 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Position measuring system 1	DB31, ...DBX1.5	DB38xx.DBX1.5
Position measuring system 2	DB31, ...DBX1.6	DB38xx.DBX1.6

H2: Auxiliary function outputs to PLC 9

9.1 Brief description

9.1.1 Function

Auxiliary functions permit activation of the system functions of the NC and PLC user functions. Auxiliary functions can be programmed in:

- Part programs
- Synchronized actions
- User cycles

For detailed information on the use of auxiliary function outputs in synchronized actions, see:

References:
Function Manual, Synchronized Actions

Predefined auxiliary functions

Predefined auxiliary functions activate system functions. The auxiliary function is also output to the NC/PLC interface.

The following auxiliary functions are predefined:

Type	Function	Example	Meaning
M	Additional function	M30	End of program
S	Spindle function	S100	Spindle speed 100 (e.g. rpm)
T	Tool number	T3	Tool number 3
D, DL	Tool offset	D1	Tool cutting edge number 1
F	Feedrate	F1000	Feedrate 1000 (e.g. mm/min)

Userdefined auxiliary functions

User-defined auxiliary functions are either extended predefined auxiliary functions or user-specific auxiliary functions.

Extension of predefined auxiliary functions

Extension of predefined auxiliary functions refers to the "address extensions" parameter. The address extension defines the number of the spindle to which the auxiliary function applies. The spindle function M3 (spindle right) is predefined for the master spindle of a channel. If a

2nd spindle is assigned to a channel, a corresponding user-defined auxiliary function must be defined that extends the predefined auxiliary function.

Type	Function	Example	Meaning
M	Additional function	M2=3	2nd spindle: Spindle right
S	Spindle function	S2=100	2nd spindle: Spindle speed = 100 (e.g. rpm)
T	Tool number	T2=3	

User-specific auxiliary functions

User-specific auxiliary functions do not activate system functions. User-specific auxiliary functions are output to the NC/PLC interface only. The functionality of the auxiliary functions must be implemented by the machine manufacturer / user in the PLC user program.

Type	Function	Example	Meaning
H[1)]	Auxiliary function	H2=5	User-specific function

1) Recommendation

9.1.2 Definition of an auxiliary function

An auxiliary function is defined by the following parameters:

- **Type, address extension and value**
  The three parameters are output to the NC/PLC interface.
- **Output behavior**
  The auxiliary function-specific output behavior defines for how long an auxiliary function is output to the NC/PLC interface and when it is output relative to the traversing motion programmed in the same part program block.
- **Group assignment**
  An auxiliary function can be assigned to a particular auxiliary function group. The output behavior can be defined separately for each auxiliary function group. This becomes active if no auxiliary function-specific output behavior has been defined. Group membership also affects output of an auxiliary function after block search.

For more detailed information on auxiliary function output to the NC/PLC interface, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)".

9.1.3 Overview of the auxiliary functions

M functions

M (special function)					
Address extension		**Value**			
Range of values	**Meaning**	**Value range** 10)	**Type**	**Meaning**	**Number** 8)
0 (implicit)	---	0 ... 99	INT	Function	5
Range of values	**Meaning**	**Value range** 11)	**Type**	**Meaning**	**Number** 8)

M (special function)					
1 ... 20	Spindle number	1 ... 99	INT	Function	5
Range of values	**Meaning**	**Value range** [12]	**Type**	**Meaning**	**Number** [8]
0 ... 99	Any	100 ... 2147483647	INT	Function	5

[8] See "Meaning of footnotes" at the end of the overview.

[10] For the value range 0 ... 99, the address extension is 0. Mandatory without address extension: M0, M1, M2, M17, M30

[11] M3, M4, M5, M19, M70: The address extension is the spindle number, e.g. M2=5 ⇒ spindle stop (M5) for spindle 2. Without address extension, the M function acts on the master spindle.

[12] User-specific M functions.

Use

Controlling machine functions in synchronism with the part program.

Additional information

- The following M functions have a predefined meaning: M0, M1, M2, M17, M30, M3, M4, M5, M6, M19, M70, M40, M41, M42, M43, M44, M45.
- A dynamic NC/PLC interface signal for displaying the validity is assigned to the M functions (M0 - M99). In addition, 64 additional signals can be assigned for user M functions (see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)").
- For subprograms, machine data can be used to set whether an output of the M function should be realized for the end of the part program M17, M2 or M30 to the PLC:
  MD20800 $MC_SPF_END_TO_VDI (subprogram end to PLC)
- For the predefined M functions 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[10]		Value			
Range of values	Meaning	Range of values	Type	Meaning	Number [8]
0 ... 20	Spindle number [5]	0 ... ± 3.4028 exp38 [3]	REAL	Spindle speed	3

[3] [5] [8] See "Meaning of footnotes" at the end of the overview

[10] The master spindle of the channel is addressed if no address extension is specified.

Use

Spindle speed.

Additional information

- The S functions are assigned to the 3rd auxiliary function group as default setting.
- Without an address extension, the S functions refer to the master spindle of the channel.
- S function-specific machine data:
  MD22210 $MC_AUXFU_S_SYNC_TYPE (output time of the S functions)

H functions

The functionality of an H function must be implemented in the PLC user program.

H (auxiliary function) [10]					
Address extension		Value			
Range of values	Meaning	Range of values	Type	Meaning	Number [8]
0 ... 99	Any	- 2147483648 ... + 2147483647	INT	Any	3
		0 ... ± 3.4028 exp38 [2] [3] [4]	REAL		

[2] [3] [4] [8] See "Meaning of footnotes" at the end of the overview.

Use

User-specific auxiliary functions.

Additional information

H function-specific machine data:

- MD22110 $MC_AUXFU_H_TYPE_INT (type of H-auxiliary function is an integer)
- MD22230 $MC_AUXFU_H_SYNC_TYPE (output time of the H functions)

T functions

Tool names are not output at the NC/PLC interface.

T (tool number) [1] [5] [6]					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
1 ... 12	Spindle number (with active tool management)	0 ... 32000 (also symbolic names for active tool management)	INT	Selection of the tool	1

[1] [5] [6] [8] See "Meaning of footnotes" at the end of the overview.

Use

Tool selection.

Additional information

- Identification of the tools, optionally via tool number or location number (see Section "W1: Tool offset (Page 1451)").
  References:
  Function Manual Tool Management
- When T0 is selected, the current tool is removed from the toolholder but not replaced by a new tool (default setting).
- T function-specific machine data:
  MD22220 $MC_AUXFU_T_SYNC_TYPE (output time of the T functions)

D functions

The tool offset is deselected using D0. Default setting is D1.

D (tool offset)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
---	---	0 ... 9	INT	Selection of the tool offset	1

[8] See "Meaning of footnotes" at the end of the overview.

Use

Selection of the tool offset.

Additional information

- Initial setting: D1
- After a tool change, the default tool cutting edge can be parameterized via:
  MD20270 $MC_CUTTING_EDGE_DEFAULT (basic position of the tool cutting edge without programming)

- Deselection of the tool offset: D0
- D function-specific machine data:
  MD22250 $MC_AUXFU_D_SYNC_TYPE (output time of the D functions)

DL functions

The additive tool offset selected with DL refers to the active D number.

DL (additive tool offset)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8)]
---	---	0 ... 6	INT	Selection of the summed tool offset	1

8) See "Meaning of footnotes" at the end of the overview.

Use

Selection of the additive tool offset with reference to an active tool offset.

Additional information

- Initial setting: DL = 0
- DL values cannot be output to the PLC via synchronized actions.
- Default setting of the additive tool offset without an active DL function:
  MD20272 $MC_SUMCORR_DEFAULT (basic setting of the additive offset without a program)
- Deselection of the additive tool offset: DL = 0
- DL function-specific machine data:
  MD22252 $MC_AUXFU_DL_SYNC_TYPE (output time DL functions)

F functions

F (feedrate)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8)]
---	---	0.001 ... 999 999.999	REAL	Path feedrate	6

8) See "Meaning of footnotes" at the end of the overview.

Use

Path velocity.

Additional information

F function-specific machine data:

- MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

FA functions

FA (axes-specific feedrate)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** 8)
1 - 31	Axis number	0.001 ... 999 999.999	REAL	Axial feedrate	6

8) See "Meaning of footnotes" at the end of the overview.

Use

Axis-specific velocity.

Additional information

F function-specific machine data:

- MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

Meaning of footnotes

1) If tool management is active, neither a T change signal nor a T word is output at the channel-specific NC/PLC interface.

2) The type for the values can be selected by the user via MD22110 $MC_AUX-FU_H_TYPE_INT.

3) Because of the limited display options on the operator panel screens, the REAL type values displayed are restricted to:

–999 999 999.9999 to 999 999 999.9999

The NC calculates internally but with complete accuracy.

4) The REAL values are rounded and output to the PLC when setting the machine data:

MD22110 $MC_AUXFU_H_TYPE_INT = 1 (type of H-auxiliary functions is an integer)

The PLC user program must interpret the value transferred according to the machine data setting.

5) If the tool management is active, the meaning of the address extension can be parameterized. Address extension = 0 means the value must be replaced by that of the master spindle number, i.e. it is equivalent to not programming the address extension.

Auxiliary functions M19 "Position spindle" collected during a block search are not output to the PLC.

6) M6: Range of values of the address extension:

- without tool management: 0 ... 99

- with tool management: 0 ... maximum spindle number

0: To be replaced by the value of the master spindle number or master 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.

9.2 Predefined auxiliary functions

Function

Every pre-defined auxiliary function is assigned to a system function and cannot be changed. If a pre-defined auxiliary function is programmed in a part program/cycle, then this is output to the PLC via the NC/PLC interface and the corresponding system function is executed in the NC.

Definition of a predefined auxiliary function

The parameters of the predefined auxiliary function are stored in machine data and can be changed in some cases. All machine data, which are assigned to an auxiliary function, have the same index <n>.

- MD22040 $MC_AUXFU_PREDEF_GROUP[<n>] (group assignment of predefined auxiliary functions)
- MD22050 $MC_AUXFU_PREDEF_TYPE[<n>] (type of predefined auxiliary functions)
- MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>] (address extension for predefined auxiliary functions)
- MD22070 $MC_AUXFU_PREDEF_VALUE[<n>] (value of predefined auxiliary functions)
- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>] (output behavior of predefined auxiliary functions)

9.2.1 Overview: Predefined auxiliary functions

Significance of the parameters listed in the following tables:

Parameter	Meaning
Index <n>	Machine data index of the parameters of an auxiliary function
Type	MD22050 $MC_AUXFU_PREDEF_TYPE[<n>]
Address extension	MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>]
Value	MD22070 $MC_AUXFU_PREDEF_VALUE[<n>]
Group	MD22040 $MC_AUXFU_PREDEF_GROUP[<n>]

Predefined auxiliary functions

General auxiliary functions, Part 1					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Stop	0	M	0	0	1
Conditional stop	1	M	0	1	1

General auxiliary functions, Part 1					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
End of subprogram	2	M	0	2	1
	3	M	0	17	1
	4	M	0	30	1
Tool change	5	M	(0)	6 1)	(1)

Spindle-specific auxiliary functions, spindle 1					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	6	M	1	3	(2)
Spindle left	7	M	1	4	(2)
Spindle stop	8	M	1	5	(2)
Position spindle	9	M	1	19	(2)
Axis mode	10	M	1	70 2)	(2)
Automatic gear stage	11	M	1	40	(4)
Gear stage 1	12	M	1	41	(4)
Gear stage 2	13	M	1	42	(4)
Gear stage 3	14	M	1	43	(4)
Gear stage 4	15	M	1	44	(4)
Gear stage 5	16	M	1	45	(4)
Spindle speed	17	S	1	-1	(3)

General auxiliary functions, Part 2					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Feedrate	18	F	0	-1	(1)
Cutting edge selection	19	D	0	-1	(1)
DL	20	L	0	-1	(1)
Tool selection	21	T	(0)	-1	(1)
Stop (associated)	22	M	0	-1 3)	1
Conditional stop (associated)	23	M	0	-1 4)	1
End of subprogram	24	M	0	-1 5)	1
Nibbling	25	M	0	20 6)	(10)
Nibbling	26	M	0	23 6)	(10)
Nibbling	27	M	0	22 6)	(11)
Nibbling	28	M	0	25 6)	(11)
Nibbling	29	M	0	26 6)	(12)
Nibbling	30	M	0	122 6)	(11)
Nibbling	31	M	0	125 6)	(11)
Nibbling	32	M	0	27 6)	(12)

Spindle-specific auxiliary functions, spindle 2					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	33	M	2	3	(72)
Spindle left	34	M	2	4	(72)
Spindle stop	35	M	2	5	(72)
Position spindle	36	M	2	19	(72)
Axis mode	37	M	2	70 [2)]	(72)
Automatic gear stage	38	M	2	40	(74)
Gear stage 1	39	M	2	41	(74)
Gear stage 2	40	M	2	42	(74)
Gear stage 3	41	M	2	43	(74)
Gear stage 4	42	M	2	44	(74)
Gear stage 5	43	M	2	45	(74)
Spindle speed	44	S	2	-1	(73)

Spindle-specific auxiliary functions, spindle 3					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	45	M	3	3	(75)
Spindle left	46	M	3	4	(75)
Spindle stop	47	M	3	5	(75)
Position spindle	48	M	3	19	(75)
Axis mode	49	M	3	70 [2)]	(75)
Automatic gear stage	50	M	3	40	(77)
Gear stage 1	51	M	3	41	(77)
Gear stage 2	52	M	3	42	(77)
Gear stage 3	53	M	3	43	(77)
Gear stage 4	54	M	3	44	(77)
Gear stage 5	55	M	3	45	(77)
Spindle speed	56	S	3	-1	(76)

Spindle-specific auxiliary functions, spindle 4					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	57	M	4	3	(78)
Spindle left	58	M	4	4	(78)
Spindle stop	59	M	4	5	(78)
Position spindle	60	M	4	19	(78)
Axis mode	61	M	4	70 [2)]	(78)

Spindle-specific auxiliary functions, spindle 4					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Automatic gear stage	62	M	4	40	(80)
Gear stage 1	63	M	4	41	(80)
Gear stage 2	64	M	4	42	(80)
Gear stage 3	65	M	4	43	(80)
Gear stage 4	66	M	4	44	(80)
Gear stage 5	67	M	4	45	(80)
Spindle speed	68	S	4	-1	(79)

Spindle-specific auxiliary functions, spindle 5					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	69	M	5	3	(81)
Spindle left	70	M	5	4	(81)
Spindle stop	71	M	5	5	(81)
Position spindle	72	M	5	19	(81)
Axis mode	73	M	5	70 2)	(81)
Automatic gear stage	74	M	5	40	(83)
Gear stage 1	75	M	5	41	(83)
Gear stage 2	76	M	5	42	(83)
Gear stage 3	77	M	5	43	(83)
Gear stage 4	78	M	5	44	(83)
Gear stage 5	79	M	5	45	(83)
Spindle speed	80	S	5	-1	(82)

Spindle-specific auxiliary functions, spindle 6					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	81	M	6	3	(84)
Spindle left	82	M	6	4	(84)
Spindle stop	83	M	6	5	(84)
Position spindle	84	M	6	19	(84)
Axis mode	85	M	6	70 2)	(84)
Automatic gear stage	86	M	6	40	(86)
Gear stage 1	87	M	6	41	(86)
Gear stage 2	88	M	6	42	(86)
Gear stage 3	89	M	6	43	(86)
Gear stage 4	90	M	6	44	(86)
Gear stage 5	91	M	6	45	(86)
Spindle speed	92	S	6	-1	(85)

Spindle-specific auxiliary functions, spindle 7					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	93	M	7	3	(87)
Spindle left	94	M	7	4	(87)
Spindle stop	95	M	7	5	(87)
Position spindle	96	M	7	19	(87)
Axis mode	97	M	7	70 2)	(87)
Automatic gear stage	98	M	7	40	(89)
Gear stage 1	99	M	7	41	(89)
Gear stage 2	100	M	7	42	(89)
Gear stage 3	101	M	7	43	(89)
Gear stage 4	102	M	7	44	(89)
Gear stage 5	103	M	7	45	(89)
Spindle speed	104	S	7	-1	(88)

Spindle-specific auxiliary functions, spindle 8					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	105	M	8	3	(90)
Spindle left	106	M	8	4	(90)
Spindle stop	107	M	8	5	(90)
Position spindle	108	M	8	19	(90)
Axis mode	109	M	8	70 2)	(90)
Automatic gear stage	110	M	8	40	(92)
Gear stage 1	111	M	8	41	(92)
Gear stage 2	112	M	8	42	(92)
Gear stage 3	113	M	8	43	(92)
Gear stage 4	114	M	8	44	(92)
Gear stage 5	115	M	8	45	(92)
Spindle speed	116	S	8	-1	(91)

Spindle-specific auxiliary functions, spindle 9					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	117	M	9	3	(93)
Spindle left	118	M	9	4	(93)
Spindle stop	119	M	9	5	(93)
Position spindle	120	M	9	19	(93)
Axis mode	121	M	9	70 2)	(93)

Spindle-specific auxiliary functions, spindle 9					
System function	Index <n>	Type	Address extension	Value	Group
Automatic gear stage	122	M	9	40	(95)
Gear stage 1	123	M	9	41	(95)
Gear stage 2	124	M	9	42	(95)
Gear stage 3	125	M	9	43	(95)
Gear stage 4	126	M	9	44	(95)
Gear stage 5	127	M	9	45	(95)
Spindle speed	128	S	9	-1	(94)

Spindle-specific auxiliary functions, spindle 10					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	129	M	10	3	(96)
Spindle left	130	M	10	4	(96)
Spindle stop	131	M	10	5	(96)
Position spindle	132	M	10	19	(96)
Axis mode	133	M	10	70 2)	(96)
Automatic gear stage	134	M	10	40	(98)
Gear stage 1	135	M	10	41	(98)
Gear stage 2	136	M	10	42	(98)
Gear stage 3	137	M	10	43	(98)
Gear stage 4	138	M	10	44	(98)
Gear stage 5	139	M	10	45	(98)
Spindle speed	140	S	10	-1	(97)

Spindle-specific auxiliary functions, spindle 11					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	141	M	11	3	(99)
Spindle left	142	M	11	4	(99)
Spindle stop	143	M	11	5	(99)
Position spindle	144	M	11	19	(99)
Axis mode	145	M	11	70 2)	(99)
Automatic gear stage	146	M	11	40	(101)
Gear stage 1	147	M	11	41	(101)
Gear stage 2	148	M	11	42	(101)
Gear stage 3	149	M	11	43	(101)
Gear stage 4	150	M	11	44	(101)
Gear stage 5	151	M	11	45	(101)
Spindle speed	152	S	11	-1	(100)

Spindle-specific auxiliary functions, spindle 12					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	153	M	11	3	(102)
Spindle left	154	M	12	4	(102)
Spindle stop	155	M	12	5	(102)
Position spindle	156	M	12	19	(102)
Axis mode	157	M	12	70 2)	(102)
Automatic gear stage	158	M	12	40	(104)
Gear stage 1	159	M	12	41	(104)
Gear stage 2	160	M	12	42	(104)
Gear stage 3	161	M	12	43	(104)
Gear stage 4	162	M	12	44	(104)
Gear stage 5	163	M	12	45	(104)
Spindle speed	164	S	12	-1	(103)

Spindle-specific auxiliary functions, spindle 13					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	165	M	13	3	(105)
Spindle left	166	M	13	4	(105)
Spindle stop	167	M	13	5	(105)
Position spindle	168	M	13	19	(105)
Axis mode	169	M	13	70 2)	(105)
Automatic gear stage	170	M	13	40	(107)
Gear stage 1	171	M	13	41	(107)
Gear stage 2	172	M	13	42	(107)
Gear stage 3	173	M	13	43	(107)
Gear stage 4	174	M	13	44	(107)
Gear stage 5	175	M	13	45	(107)
Spindle speed	176	S	13	-1	(106)

Spindle-specific auxiliary functions, spindle 14					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	177	M	14	3	(108)
Spindle left	178	M	14	4	(108)
Spindle stop	179	M	14	5	(108)
Position spindle	180	M	14	19	(108)
Axis mode	181	M	14	70 2)	(108)

Spindle-specific auxiliary functions, spindle 14					
System function	Index <n>	Type	Address extension	Value	Group
Automatic gear stage	182	M	14	40	(110)
Gear stage 1	183	M	14	41	(110)
Gear stage 2	184	M	14	42	(110)
Gear stage 3	185	M	14	43	(110)
Gear stage 4	186	M	14	44	(110)
Gear stage 5	187	M	14	45	(110)
Spindle speed	188	S	14	-1	(109)

Spindle-specific auxiliary functions, spindle 15					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	189	M	15	3	(111)
Spindle left	190	M	15	4	(111)
Spindle stop	191	M	15	5	(111)
Position spindle	192	M	15	19	(111)
Axis mode	193	M	15	70 2)	(111)
Automatic gear stage	194	M	15	40	(113)
Gear stage 1	195	M	15	41	(113)
Gear stage 2	196	M	15	42	(113)
Gear stage 3	197	M	15	43	(113)
Gear stage 4	198	M	15	44	(113)
Gear stage 5	199	M	15	45	(113)
Spindle speed	200	S	15	-1	(112)

Spindle-specific auxiliary functions, spindle 16					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	201	M	16	3	(114)
Spindle left	202	M	16	4	(114)
Spindle stop	203	M	16	5	(114)
Position spindle	204	M	16	19	(114)
Axis mode	205	M	16	70 2)	(114)
Automatic gear stage	206	M	16	40	(116)
Gear stage 1	207	M	16	41	(116)
Gear stage 2	208	M	16	42	(116)
Gear stage 3	209	M	16	43	(116)
Gear stage 4	210	M	16	44	(116)
Gear stage 5	211	M	16	45	(116)
Spindle speed	212	S	16	-1	(115)

Spindle-specific auxiliary functions, spindle 17					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	213	M	17	3	(117)
Spindle left	214	M	17	4	(117)
Spindle stop	215	M	17	5	(117)
Position spindle	216	M	17	19	(117)
Axis mode	217	M	17	70 2)	(117)
Automatic gear stage	218	M	17	40	(119)
Gear stage 1	219	M	17	41	(119)
Gear stage 2	220	M	17	42	(119)
Gear stage 3	221	M	17	43	(119)
Gear stage 4	222	M	17	44	(119)
Gear stage 5	223	M	17	45	(119)
Spindle speed	224	S	17	-1	(118)

Spindle-specific auxiliary functions, spindle 18					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	225	M	18	3	(120)
Spindle left	226	M	18	4	(120)
Spindle stop	227	M	18	5	(120)
Position spindle	228	M	18	19	(120)
Axis mode	229	M	18	70 2)	(120)
Automatic gear stage	230	M	18	40	(122)
Gear stage 1	231	M	18	41	(122)
Gear stage 2	232	M	18	42	(122)
Gear stage 3	233	M	18	43	(122)
Gear stage 4	234	M	18	44	(122)
Gear stage 5	235	M	18	45	(122)
Spindle speed	236	S	18	-1	(121)

Spindle-specific auxiliary functions, spindle 19					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	237	M	19	3	(123)
Spindle left	238	M	19	4	(123)
Spindle stop	239	M	19	5	(123)
Position spindle	240	M	19	19	(123)
Axis mode	241	M	19	70 2)	(123)

Spindle-specific auxiliary functions, spindle 19					
System function	Index <n>	Type	Address extension	Value	Group
Automatic gear stage	242	M	19	40	(125)
Gear stage 1	243	M	19	41	(125)
Gear stage 2	244	M	19	42	(125)
Gear stage 3	245	M	19	43	(125)
Gear stage 4	246	M	19	44	(125)
Gear stage 5	247	M	19	45	(125)
Spindle speed	248	S	19	-1	(124)

Spindle-specific auxiliary functions, spindle 20					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	249	M	20	3	(126)
Spindle left	250	M	20	4	(126)
Spindle stop	251	M	20	5	(126)
Position spindle	252	M	20	19	(126)
Axis mode	253	M	20	70 2)	(126)
Automatic gear stage	254	M	20	40	(128)
Gear stage 1	255	M	20	41	(128)
Gear stage 2	256	M	20	42	(128)
Gear stage 3	257	M	20	43	(128)
Gear stage 4	258	M	20	44	(128)
Gear stage 5	259	M	20	45	(128)
Spindle speed	260	S	20	-1	(127)

Toolholder-specific auxiliary functions, T auxiliary functions					
System function	Index <n>	Type	Address extension	Value	Group
Tool selection	261	T	1	-1	129
Tool selection	262	T	2	-1	130
Tool selection	263	T	3	-1	131
Tool selection	264	T	4	-1	132
Tool selection	265	T	5	-1	133
Tool selection	266	T	6	-1	134
Tool selection	267	T	7	-1	135
Tool selection	268	T	8	-1	136
Tool selection	269	T	9	-1	137
Tool selection	270	T	10	-1	138
Tool selection	271	T	11	-1	139
Tool selection	272	T	12	-1	140

Toolholder-specific auxiliary functions, T auxiliary functions					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Tool selection	273	T	13	-1	141
Tool selection	274	T	14	-1	142
Tool selection	275	T	15	-1	143
Tool selection	276	T	16	-1	144
Tool selection	277	T	17	-1	145
Tool selection	278	T	18	-1	146
Tool selection	279	T	19	-1	147
Tool selection	280	T	20	-1	148

Toolholder-specific auxiliary functions, M6 auxiliary functions					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Tool change	281	M	1	6 1)	149
Tool change	282	M	2	6 1)	150
Tool change	283	M	3	6 1)	151
Tool change	284	M	4	6 1)	152
Tool change	285	M	5	6 1)	153
Tool change	286	M	6	6 1)	154
Tool change	287	M	7	6 1)	155
Tool change	288	M	8	6 1)	156
Tool change	289	M	9	6 1)	157
Tool change	290	M	10	6 1)	158
Tool change	291	M	11	6 1)	159
Tool change	292	M	12	6 1)	160
Tool change	293	M	13	6 1)	161
Tool change	294	M	14	6 1)	162
Tool change	295	M	15	6 1)	163
Tool change	296	M	16	6 1)	164
Tool change	297	M	17	6 1)	165
Tool change	298	M	18	6 1)	166
Tool change	299	M	19	6 1)	167
Tool change	300	M	20	6 1)	168

Legend:

( ) The value can be changed.

1) The value is depends on the machine data:
MD22560 $MC_TOOL_CHANGE_M_MODE (M function for tool change)

2) The value can be preset with a different value using the following machine data:
MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR (M function for switching over to controlled axis mode (ext. mode))
MD20094 $MC_SPIND_RIGID_TAPPING_M_NR (M function for switching over to controlled axis mode)
Note
The value 70 is always output at the PLC.

3) The value is set using machine data:
MD22254 $MC_AUXFU_ASSOC_M0_VALUE (additional M function for program stop)

4) The value is set using machine data:
MD22256 $MC_AUXFU_ASSOC_M1_VALUE (additional M function for conditional stop)

5) The value is set using machine data:
MD10714 $MN_M_NO_FCT_EOP (M function for spindle active after reset)

6) The value is set using machine data:
MD26008 $MC_NIBBLE_PUNCH_CODE (definition of M functions)

9.2.2 Overview: Output behavior

Significance of the parameters listed in the following table:

Parameter	Meaning
Index <n>	Machine data index of the parameters of an auxiliary function
Output behavior	MD22080 $MC_AUXFU_PREDEF_SPEC[<n>], Bits 0 ... 18 Bits 19 ... 31: Reserved

Output behavior of the predefined auxiliary functions

System function	Index <n>																		
		Output behavior, bit																	
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Stop	0	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Conditional stop	1	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
End of subroutine	2	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
	3	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
	4	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Tool change	5	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle right	6	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle left	7	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle stop	8	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle positioning	9	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Axis mode	10	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Automatic gear stage	11	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 1	12	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)

System function	**Index <n>**																			
		Output behavior, bit																		
Gear stage 2	13	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 3	14	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 4	15	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 5	16	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Spindle speed	17	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(1)	(0)	(0)	(0)	0	(0)	(1)
Feed	18	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(1)	(0)	0	(1)	0	(0)	(1)
Cutting edge selection	19	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
DL	20	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
Tool selection	21	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
Stop (associated)	22	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Conditional stop (asso-ciated)	23	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
End of subroutine	24	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Nibbling	25	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	26	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	27	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	28	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	29	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	30	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	31	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	32	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)

( ) The value can be changed.

Significance of the bits

Bit	**Meaning**
0	Acknowledgement "normal" after an OB1 cycle
	An auxiliary function with normal acknowledgment is output to the NC/PLC interface at the beginning of the OB1 cycle. The auxiliary function-specific change signal indicates to the PLC user program that the auxiliary function is valid. The auxiliary function is acknowledged as soon as organization block OB1 has run once. This corresponds to a complete PLC user cycle. The auxiliary function with normal acknowledgment is output in synchronism with the part program block in which it is programmed. If execution of the parts program block, e.g. path and/or positioning axis movements, is completed before acknowledgment of the auxiliary function, the block change is delayed until after acknowledgment by the PLC. In continuous-path mode, a constant path velocity can be maintained in conjunction with an auxiliary function with normal acknowledgment, if the auxiliary function is output by the PLC during the traversing motion and before reaching the end of the block.

Bit	Meaning
1	Acknowledgement "quick" with OB40
	An auxiliary function with quick acknowledgment is output to the NC/PLC interface before the next OB1 cycle. The auxiliary function-specific change signal indicates to the PLC user program that the auxiliary function is valid. The auxiliary function is acknowledged immediately by the PLC basic program in the next OB40 cycle. Acknowledgment of the auxiliary function is not confirmation that the corresponding PLC user function has been executed. The auxiliary function is still executed in the OB1 cycle. Next output of the auxiliary functions to the PLC is therefore not possible until after this OB1 cycle has run completely. This is noticeable in continuous-path mode (drop in path velocity) especially if auxiliary functions with quick acknowledgment are output in several consecutive part program blocks. With auxiliary functions with quick acknowledgment, it cannot be guaranteed that the PLC user program will respond in synchronism with the block. Note Parameterization of the output behavior of auxiliary functions as "quick auxiliary functions" is only possible in conjunction with user-defined auxiliary functions.
2	No predefined auxiliary function
	A predefined auxiliary function is treated like a user-defined auxiliary function with this setting. The auxiliary function then no longer triggers the corresponding system function but is only output to the PLC. Example: Reconfiguration of the "Position spindle" auxiliary function (index 9) to a user-defined auxiliary function with normal acknowledgment and output prior to the traversing motion. MD22080 $MC_AUXFU_PREDEF_SPEC [9] = 'H25' (100101B)
3	No output to the PLC
	The auxiliary function is not output to the PLC.
4	Spindle response after acknowledgement by the PLC
	The associated system function is only executed after acknowledgment by the PLC.
5	Output prior to motion
	The auxiliary function is output to the PLC before the traversing motions programmed in the part program block (path and/or block-related positioning axis movements).
6	Output during motion
	The auxiliary function is output to the PLC during the traversing motions programmed in the part program block (path and/or block-related positioning axis movements).
7	Output at block end
	The auxiliary function is output to the PLC after the traversing motions programmed in the part program block have been completed (path and/or block-related positioning axis movements).
8	Not output after block search, types 1, 2, 4
	Block search, types 1, 2, 4: The auxiliary function collected during the block search is not output.
9	Collection during block search with program test (type 5, SERUPRO)
	For a block search with program test, the auxiliary function is collected group-specific in the following system variables: • $AC_AUXFU_M_VALUE[<n>] • $AC_AUXFU_M_EXT[<n>] • $AC_AUXFU_M_STATE[<n>]
10	No output during block search with program test (type 5, SERUPRO)
	For block search with program test, the auxiliary function is not output to the PLC.

Bit	Meaning
11	Cross-channel auxiliary function (SERUPRO)
	For block search with program test (SERUPRO), the help function is collected on a cross-channel basis in the global list of the auxiliary functions. Note For each auxiliary function group, only the last auxiliary function of the group is always collected.
12	Output performed via synchronized action (read only)
	The bit is set if the auxiliary function was output to the PLC via a synchronized action.
13	Implicit auxiliary function (read-only)
	The bit is set if the auxiliary function was implicitly output to the PLC.
14	Active M01 (read only)
	The bit is set if the auxiliary function, for active M01, was output to the PLC.
15	No output during positioning test run
	During the run-in test, the auxiliary function is not output to the PLC.
16	Nibbling off
17	Nibbling on
18	Nibbling

Note

In the case of auxiliary functions for which no output behavior has been defined, the following default output behavior is active:

- Bit 0 = 1: Output duration one OB1 cycle
- Bit 7 = 1: Output at block end

9.2.3 Parameterization

9.2.3.1 Group assignment

The handling of the auxiliary functions for a block search is defined using the group assignment of an auxiliary function. The 168 auxiliary function groups available are subdivided into predefined and user-definable groups:

Predefined groups:	1 ... 4		10 ... 12		72 ... 168
User-defined groups:		5 ... 9		13 ... 71	

Each predefined auxiliary function is assigned, as standard, to an auxiliary function group. For most pre-defined auxiliary functions, this assignment can be changed using the following machine data:

MD22040 $MC_AUXFU_PREDEF_GROUP[<n>] (group assignment of predefined auxiliary functions)

If an auxiliary function is not assigned to any group, then a value of "0" should be entered into the machine data.

For the pre-defined auxiliary functions with the following indices <n>, the group assignment cannot be changed: 0, 1, 2, 3, 4, 22, 23, 24

Note

1. Auxiliary function group and block search

Auxiliary functions of the 1st auxiliary function group are, for a block search, only collected, but not output.

9.2.3.2 Type, address extension and value

An auxiliary function is programmed via the type, address extension and value parameters (see Section "Programming an auxiliary function (Page 437)").

Type

The identifier of an auxiliary function is defined via the "type," e.g.:

"M"	For additional function
"S"	For spindle function
"F"	For feed

The setting is made via the following machine data:

MD22050 $MC_AUXFU_PREDEF_TYPE[<n>] (type of predefined auxiliary functions)

Note

The "type" cannot be changed for predefined auxiliary functions.

Address extension

The "address extension" of an auxiliary function is for addressing different components of the same type. In the case of predefined auxiliary functions, the value of the "address extension" is the spindle number to which the auxiliary function applies.

The setting is made via the following machine data:

MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>] (address extension for predefined auxiliary functions)

Grouping together auxiliary functions

To assign an auxiliary function for all spindles of a channel to the same auxiliary function group, the value "-1" is entered for the "address extension" parameter.

Example:

The auxiliary function M3 (machine data index = 6) is assigned to the second auxiliary function group for all the channel's spindles.

MD22040 $MC_AUXFU_PREDEF_GROUP[6]	= 2
MD22050 $MC_AUXFU_PREDEF_TYPE[6]	= "M"
MD22060 $MC_AUXFU_PREDEF_EXTENSION[6]	= -1
MD22070 $MC_AUXFU_PREDEF_VALUE[6]	= 3

Value

The parameters "value" and "type" define the meaning of an auxiliary function, i.e. the system function that is activated on the basis of this auxiliary function.

The "value" of an auxiliary function is defined in the machine data:

MD22070 $MC_AUXFU_PREDEF_VALUE[<n>] (value of predefined auxiliary functions)

Note

The "value" cannot be changed for a predefined auxiliary function. For some predefined auxiliary functions, the "value" can be reconfigured via additional machine data (see Section "Associated auxiliary functions (Page 432)").

9.2.3.3 Output behavior

Parameter "Output behavior" defines when the predefined auxiliary function is output to the NC/PLC interface and when it is acknowledged by the PLC.

The setting is done via the following machine data:

MD22080 $MC_AUXFU_PREDEF_SPEC[<n>] (output behavior of predefined auxiliary functions)

Output behavior relative to motion

Output prior to motion

- The traversing motions (path and/or block-related positioning axis movements) of the previous part program block end with an exact stop.
- The auxiliary functions are output at the beginning of the current parts program block.
- The traversing motion of the actual part program block (path and/or positioning axis motion) is only started after acknowledgment of the auxiliary functions by the PLC:
  - Output duration one OB1 cycle (normal acknowledgment): after one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): after one OB40 cycle

Output during motion

- The auxiliary functions are output at the beginning of the traversing motions (path and/or positioning axis movements).
- The path velocity of the current parts program block is reduced so that the time to the end of the block is greater than the time to acknowledgment of the auxiliary functions by the PLC.
  - Output duration one OB1 cycle (normal acknowledgment): one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): one OB40 cycle

Output after motion

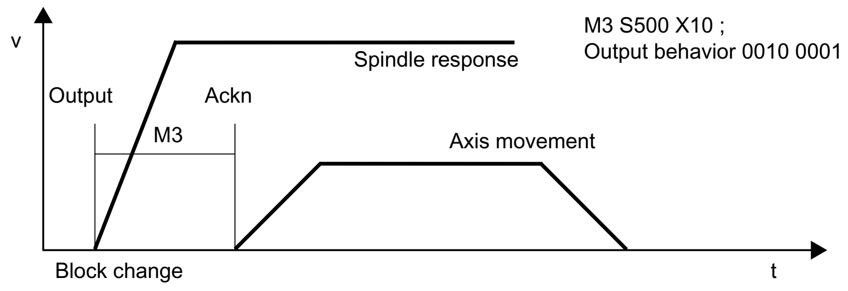
- The traversing motions (path and/or block-related positioning axis movements) of the current part program block end with an exact stop.
- The auxiliary functions are output after completion of the traversing motions.
- The block change is performed after acknowledgment of the auxiliary functions by the PLC:
  - Output duration one OB1 cycle (normal acknowledgment): after one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): after one OB40 cycle

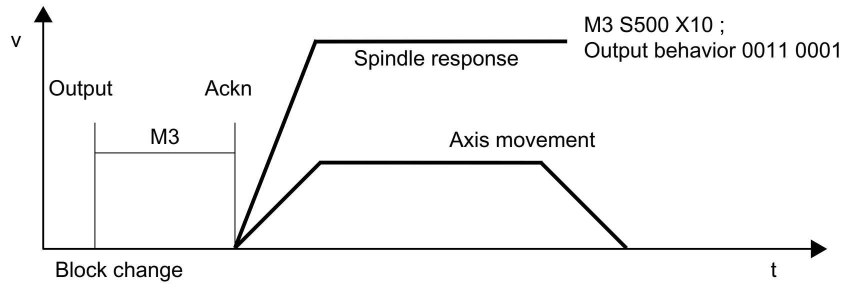
Examples of different output behavior

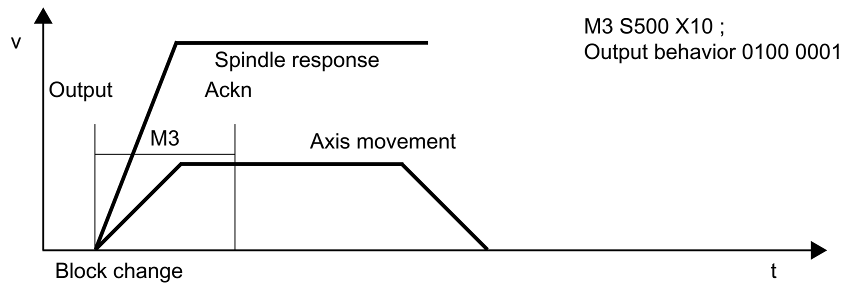
The following figures illustrate the differing behavior regarding:

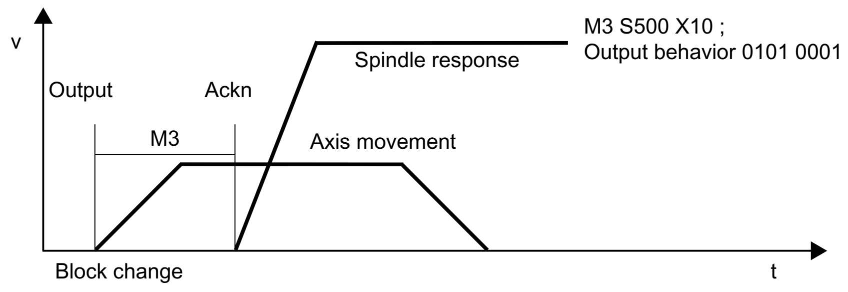
- Output and acknowledgment of the auxiliary function
- Spindle response (speed change)
- Traverse movement (velocity change)

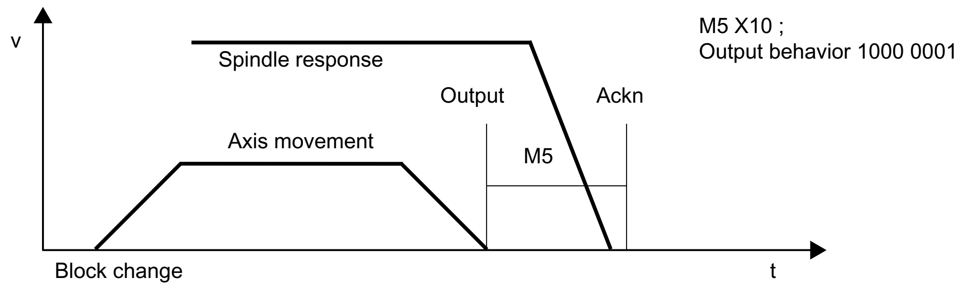
The binary values specified in the diagrams under "Output behavior" refer to the parameterized output behavior (MD22080).

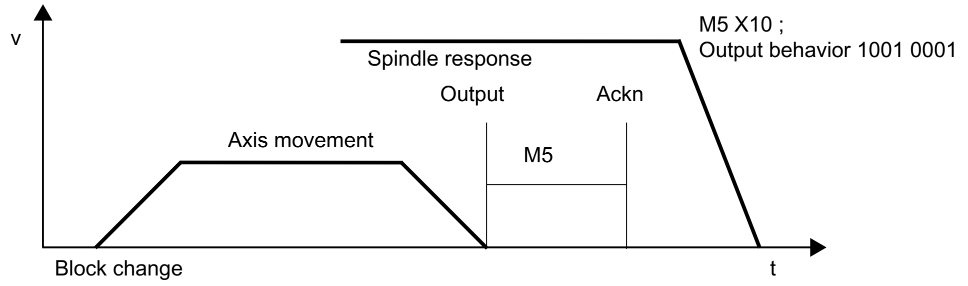

v
Output
Ackn
M3
Spindle response
Axis movement
Block change
t
M3 S500 X10 ;
Output behavior 0010 0001

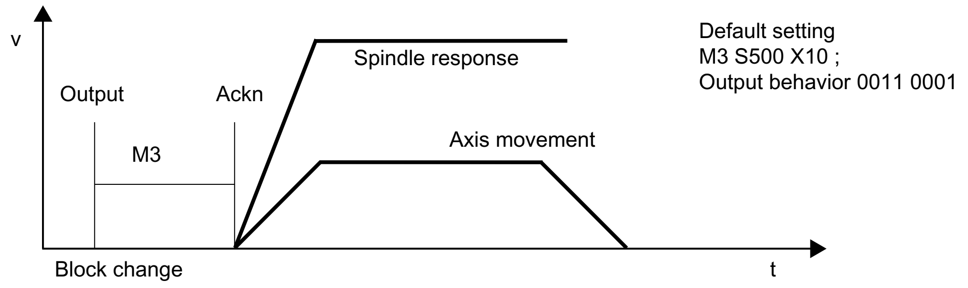

v
Output
Ackn
M3
Spindle response
Axis movement
Block change
t
M3 S500 X10 ;
Output behavior 0011 0001

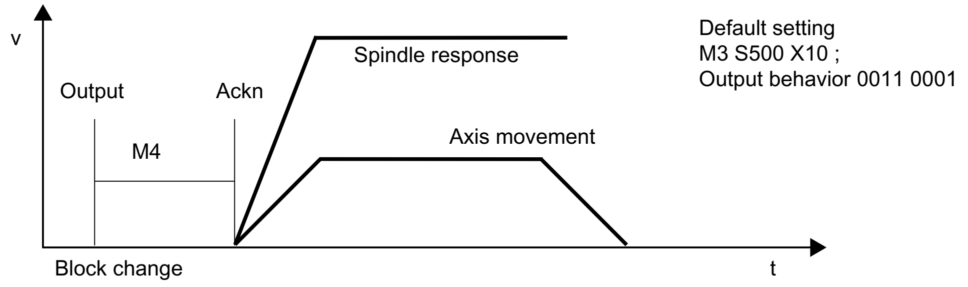

v
Output
Ackn
M3
Spindle response
Axis movement
Block change
t
M3 S500 X10 ;
Output behavior 0100 0001

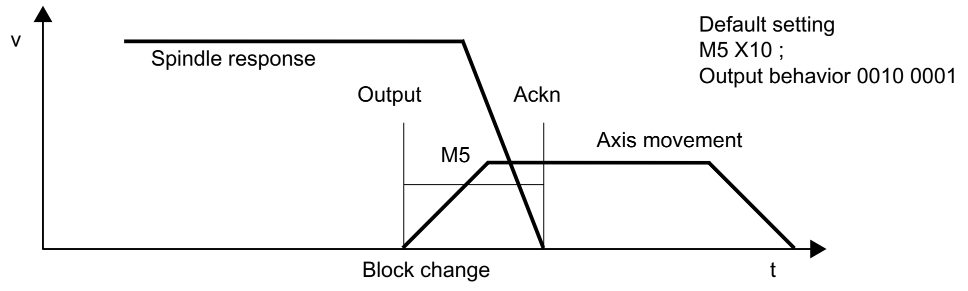

v
Output
Ackn
M3
Spindle response
Axis movement
Block change
t
M3 S500 X10 ;
Output behavior 0101 0001

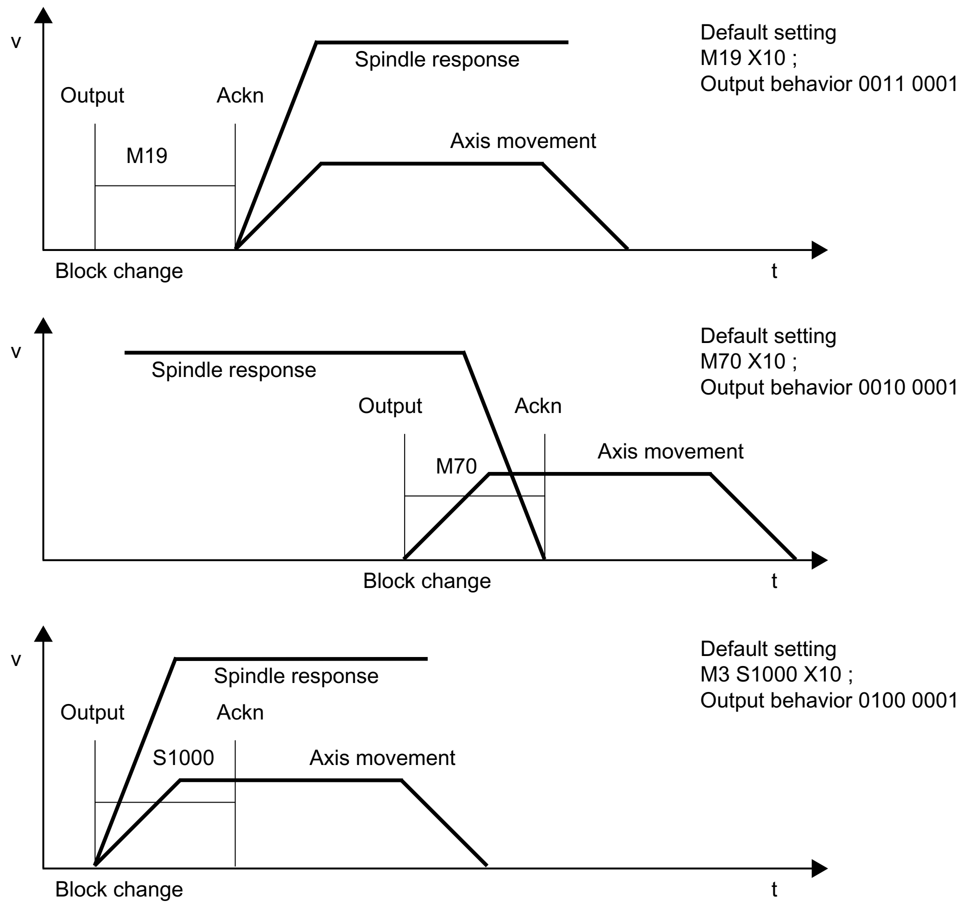

v
Spindle response
Output
Ackn
Axis movement
M5
Block change
t
M5 X10 ;
Output behavior 1000 0001


v
Spindle response
Output
Ackn
Axis movement
M5
Block change
t
M5 X10 ;
Output behavior 1001 0001


v
Spindle response
Output
Ackn
M3
Axis movement
Block change
t
Default setting
M3 S500 X10 ;
Output behavior 0011 0001


v
Spindle response
Output
Ackn
M4
Axis movement
Block change
t
Default setting
M3 S500 X10 ;
Output behavior 0011 0001


v
Spindle response
Output
Ackn
M5
Axis movement
Block change
t
Default setting
M5 X10 ;
Output behavior 0010 0001



9.3 Userdefined auxiliary functions

There are two uses for user-defined auxiliary functions:

- Extension of predefined auxiliary functions
- User-specific auxiliary functions

Extension of predefined auxiliary functions

Because there is only one set of machine data for the predefined auxiliary functions, they can only ever be used to address one spindle of the channel. To address further spindles, user-defined auxiliary functions must be parameterized to supplement the predefined auxiliary functions.

Extension of predefined auxiliary functions refers to the "address extensions" parameter. The number of the spindle that the auxiliary function refers to is entered in the "address extension" parameter.

The relevant predefined auxiliary functions can be extended for the following system functions:

System function	**Type**		
		Address extension [1)]	
			Value
Tool change	M	1	6
Spindle right	M	1	3
Spindle left	M	1	4
Spindle stop	M	1	5
Position spindle	M	1	19
Axis mode	M	1	70
Automatic gear stage	M	1	40
Gear stage 1	M	1	41
Gear stage 2	M	1	42
Gear stage 3	M	1	43
Gear stage 4	M	1	44
Gear stage 5	M	1	45
Spindle speed	S	1	-1
Tool selection	T	1	-1

1) Address extension = 1 is the default value used in the auxiliary functions predefined in the machine data

Example:

Extension of the predefined auxiliary function for the system function "spindle right" for the second and third spindle of the channel.

Auxiliary function "spindle right" for the second spindle of the channel:

MD22010 $MC_AUXFU_ASSIGN_TYPE[n]	= "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[n]	**= 2**
MD22030 $MC_AUXFU_ ASSIGN_VALUE[n]	= 3

Auxiliary function "spindle right" for the third spindle of the channel:

MD22010 $MC_AUXFU_ ASSIGN_TYPE[m]	= "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[m]	**= 3**
MD22030 $MC_AUXFU_ ASSIGN_VALUE[m]	= 3

User-specific auxiliary functions

User-specific auxiliary functions have the following characteristics:

- User-specific auxiliary functions only activate user functions.
- No system functions can be activated by user-specific auxiliary functions.

- A user-specific auxiliary function is output to the PLC according to the parameterized output behavior.
- The functionality of a user-specific auxiliary function is implemented by the machine manufacturer/user in the PLC user program.

9.3.1 Parameterization

9.3.1.1 Maximum number of user-defined auxiliary functions

The maximum number of user-defined auxiliary function per channel can be parameterized via the machine data:

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN (maximum number of user-defined auxiliary functions)

9.3.1.2 Group assignment

The handling of the auxiliary functions for a block search is defined using the group assignment of an auxiliary function. The 168 auxiliary function groups available are subdivided into predefined and user-definable groups:

Predefined groups:	1 ... 4		10 ... 12		72 ... 168
User-defined groups:		5 ... 9		13 ... 71	

Every user-defined auxiliary function is assigned as standard to the 1st auxiliary function group. The assignment can be changed using the following machine data:

MD22000 $MC_AUXFU_ASSIGN_GROUP[<n>] (group assignment of user-defined auxiliary functions)

If an auxiliary function is not assigned to any group, then a value of "0" should be entered into the machine data.

Note

1. Auxiliary function group and block search

Auxiliary functions of the 1st auxiliary function group are, for a block search, only collected, but not output.

9.3.1.3 Type, address extension and value

An auxiliary function is programmed via the type, address extension and value parameters (see Section "Programming an auxiliary function (Page 437)").

Type

The name of an auxiliary function is defined via the "type".

The identifiers for user-defined auxiliary functions are:

Type	Identifier	Meaning
"H"	Auxiliary function	User-specific auxiliary functions
"M"	Special function	Extension of predefined auxiliary functions
"S"	Spindle function	
"T"	Tool number	

The setting is made via the following machine data:

MD22010 $MC_AUXFU_ASSIGN_TYPE[<n>] (type of user-defined auxiliary functions)

Address extension

MD22020 $MC_AUXFU_ASSIGN_EXTENSION[<n>] (address extension user-defined auxiliary functions)

The functionality of the address extension is not defined in user-specific auxiliary functions. It is generally used to distinguish between auxiliary functions with the same "value".

Grouping together auxiliary functions

If all the auxiliary functions of the same type and value are assigned to the same auxiliary function group, a value of "-1" must be entered for the "address extension" parameter.

Example:

All user-specific auxiliary functions with the value "= 8" are assigned to the tenth auxiliary function group.

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 1 ]              = 10
MD22010 $MC_AUXFU_ ASSIGN_TYPE [ 1 ]              = "H"
MD22020 $MC_AUXFU_ ASSIGN_EXTENSION [ 1 ]         = -1
MD22030 $MC_AUXFU_ ASSIGN_VALUE [ 1 ]             = 8
```

Value

MD22030 $MC_AUXFU_ASSIGN_VALUE[<n>] (value of user-defined auxiliary functions)

The functionality of the "value" parameter is not defined in user-specific auxiliary functions. The value is generally used to activate the corresponding PLC user function.

Grouping together auxiliary functions

If all the auxiliary functions of the same type and address extension are assigned to the same auxiliary function group, a value of "-1" must be entered for the "value" parameter.

Example:

All user-specific auxiliary functions with the address extension "= 2" are assigned to the eleventh auxiliary function group.

MD22000 $MC_AUXFU_ASSIGN_GROUP [2]	= 11
MD22010 $MC_AUXFU_ ASSIGN_TYPE [2]	= "H"
MD22020 $MC_AUXFU_ ASSIGN_EXTENSION [2]	= 2
MD22030 $MC_AUXFU_ ASSIGN_VALUE [2]	= -1

9.3.1.4 Output behavior

The "output behavior" of user-defined auxiliary functions can be parameterized via the machine data:

MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>] (output behavior of user-defined auxiliary functions)

For a description of the individual output parameters, see the "Output behavior (Page 424)" section of the predefined auxiliary functions. The information given there can be applied analogously to the output behavior of user-defined auxiliary functions.

9.4 Associated auxiliary functions

Function

Associated help functions are user-defined help functions with the same properties as the corresponding predefined help functions. User-defined auxiliary functions can be associated for the following predefined auxiliary functions:

- M0 (programmed stop)
- M1 (optional stop)

G group

An associated help function is assigned to the G group of the corresponding predefined help function.

Parameterization

Association of a user-defined auxiliary function with one of the predefined auxiliary functions mentioned is set in the machine data:

- MD22254 $MC_AUXFU_ASSOC_M0_VALUE (associated M function for "Programmed stop")
- MD22256 $MC_AUXFU_ASSOC_M1_VALUE (associated M function for "Optional stop")

Selection

Selection of "Associated help function" (M-1) is made via the SINUMERIK Operate user interface in the operating area "Automatic" > "Program control" by setting the HMI/PLC interface signal DB21, ... DBX24.4.

Depending on the value of the FB1 parameter `MMCToIf`, the interface signal is transmitted from the basic PLC program to the NC/PLC interface signal DB21, ... DBX30.5:

- "TRUE": Transmission
- "FALSE": No transmission

By default, the value of the parameter is "TRUE".

Note

Selection option via user interface SINUMERIK Operate

The selection of an associated help function is only displayed in the "Automatic" > "Program control" operating area if it is parameterized in the machine data.

Application

Associated auxiliary functions can be used in:

- Main program
- Subprogram
- Cycle

Note

Associated auxiliary functions may **not** be used in synchronized actions.

NC/PLC interface signals

In the case of an associated user-defined auxiliary function, the same signals are output to the NC/PLC interface as for the corresponding predefined auxiliary function. To distinguish which auxiliary function has actually been programmed, the value of the user-defined auxiliary function ("value" parameter) is output as the value of the auxiliary function. This means it is possible to distinguish between predefined and user-defined auxiliary functions in the PLC user program.

NC/PLC interface signals for associated help functions:

- DB21, ... DBX24.4 (associated help function selected)
- DB21, ... DBX30.5 (activate associated help function)
- DB21, ... DBX318.5 (associated help function active)

Boundary conditions

Please note the following boundary conditions:

- A user-defined auxiliary function may not be associated a multiple number of times.
- It is not permissible to associate predefined auxiliary functions (e.g. M3, M4, M5 etc.).

Examples

1. Associating the user-defined auxiliary function M111 for M0:
   MD22254 $MC_AUXFU_ASSOC_M0_VALUE = 111
   The user-defined auxiliary function M111 therefore has the same functionality as M0.
2. Associating the user-defined auxiliary function M222 for M1:
   MD22256 $MC_AUXFU_ASSOC_M1_VALUE = 222
   The user-defined auxiliary function M222 therefore has the same functionality as M1.

9.5 Type-specific output behavior

Function

The output behavior of auxiliary functions relative to a traversing motions programmed in the parts program block can be defined type-specifically.

Parameter assignment

Parameters are assigned to type-specific output behavior via the machine data:

MD22200 $MC_AUXFU_**M**_SYNC_TYPE (output time for M functions)

MD22210 $MC_AUXFU_**S**_SYNC_TYPE (output time for S functions)

MD22220 $MC_AUXFU_**T**_SYNC_TYPE (output time for T functions)

MD22230 $MC_AUXFU_**H**_SYNC_TYPE (output time for H functions)

MD22240 $MC_AUXFU_**F**_SYNC_TYPE (output time for F functions)

MD22250 $MC_AUXFU_**D**_SYNC_TYPE (output time for D functions)

MD22252 $MC_AUXFU_**DL**_SYNC_TYPE (output time for DL functions)

The following output behaviors can be parameterized:

MD $MC_AUXFU_**xx**_SYNC_TYPE = <value>

Value	Output behavior
0	Output prior to motion
1	Output during motion
2	Output at block end
3	No output to the PLC
4	Output according to the output behavior defined with MD22080

For a description of the various output behaviors, see the section titled "Output behavior (Page 424)".

Note

For the output behavior that can be set for each type of auxiliary function, please refer to the "Detailed Description of Machine Data" Parameter Manual.

Example

Output of auxiliary functions with different output behaviors in a part program block with traverse movement.

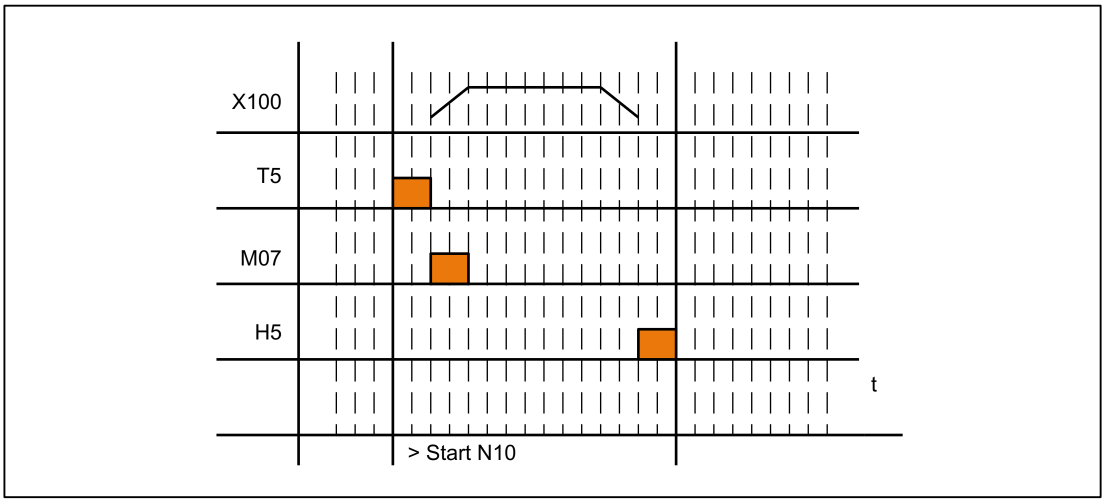
Output behavior for which parameters have been assigned:

MD22200 $MC_AUXFU_M_SYNC_TYPE = 1	⇒	M function: Output **during** motion
MD22220 $MC_AUXFU_T_SYNC_TYPE = 0	⇒	T function: Output **prior to** motion
MD22230 $MC_AUXFU_H_SYNC_TYPE = 2	⇒	H function: Output **at the end of the block**

Parts program block:

```
Program code
...
N10 G01 X100 M07 H5 T5
...
```

Time sequence for auxiliary function output:



9.6 Priorities of the output behavior for which parameters have been assigned

The following priorities must be observed for the following areas in connection with the parameterized output behavior of an auxiliary function:

- Output duration (normal / quick acknowledgement)
- Output relative to motion (prior to / during / after the motion)

As a general rule, the parameterized output behavior with lower priority becomes active if no output behavior with higher priority has been parameterized.

Output duration

The following priorities apply to the output duration:

Priority	Output behavior	Defined via:
Highest	Auxiliary function-specific	Part program instruction: QU(…) (see Section "Programmable output duration (Page 438)")
↓	Auxiliary function-specific	MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>] MD22080 $MC_AUXFU_PREDEF_SYNC[<n>]
↓	Group-specific	MD11110 $MC_AUXFU_GROUP_SPEC[<n>]
Lowest	Not defined	Default output behavior: Output duration one OB1 cycle

Output relative to motion

The following rules apply to output relative to motion:

Priority	Output behavior	Defined via:
Highest	Auxiliary function-specific	MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>] MD22080 $MC_AUXFU_PREDEF_SYNC[<n>]
↓	Group-specific	MD11110 $MC_AUXFU_GROUP_SPEC[<n>]
↓	Type-specific	MD22200 $MC_AUXFU_M_SYNC_TYPE MD22210 $MC_AUXFU_S_SYNC_TYPE MD22220 $MC_AUXFU_T_SYNC_TYPE MD22230 $MC_AUXFU_H_SYNC_TYPE MD22240 $MC_AUXFU_F_SYNC_TYPE MD22250 $MC_AUXFU_D_SYNC_TYPE MD22252 $MC_AUXFU_DL_SYNC_TYPE
Lowest	Not defined	Default output behavior: Output at block end

Note

Part program blocks without path motion

In a part program block without a path motion (even those with positioning axes and spindles), the auxiliary functions are all output immediately in a block.

9.7 Programming an auxiliary function

Syntax

An auxiliary function is programmed in a part program block with the following syntax:
`<Type>[<Address extension>=]<Value>`

Note

If no address extension is programmed, the address extension is implicitly set = 0.

Predefined auxiliary functions with the address extension = 0 always refer to the master spindle of the channel.

Symbolic addressing

The values for the "address extension" and "value" parameters can also be specified symbolically. The symbolic name for the address extension must then be stated in brackets.

Example:

Symbolic programming of the auxiliary function M3 (spindle right) for the 1st spindle:

Program code	Comment
DEF SPINDEL_NR=1	; 1st 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 numerical value is transferred instead.

Examples

Example 1: Programming of predefined auxiliary functions

Program code	Comment
N10 M3	; "Spindle clockwise" for the master spindle of the channel.
N20 M0=3	; "Spindle clockwise" for the master spindle of the channel.
N30 M1=3	; "Spindle clockwise" for the 1st spindle of the channel.
N40 M2=3	; "Spindle clockwise" for the 2nd spindle of the channel.

Example 2: Programming examples of auxiliary functions with the corresponding values for output to the PLC

Program code	Comment
`DEF Coolant=12`	`; Output to the PLC: - - -`
`DEF Lubricant=130`	`; Output to the PLC: - - -`
`H[coolant]=lubricant`	`; Output to the PLC: H12=130`
`H=coolant`	`; Output to the PLC: H0=12`
`H5`	`; Output to the PLC: H0=5`
`H=5.379`	`; Output to the PLC: H0=5.379`
`H17=3.5`	`; Output to the PLC: H17=3.5`
`H[coolant]=13.8`	`; Output to the PLC: H12=13.8`
`H='HFF13'`	`; Output to the PLC: H0=65299`
`H='B1110'`	`; Output to the PLC: H0=14`
`H5.3=21`	`; Error`

9.8 Programmable output duration

Function

User-specific auxiliary functions, for which the output behavior "Output duration of an OB1 cycle (slow acknowledgment)" was parameterized, can be defined for individual outputs via the part program guide `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

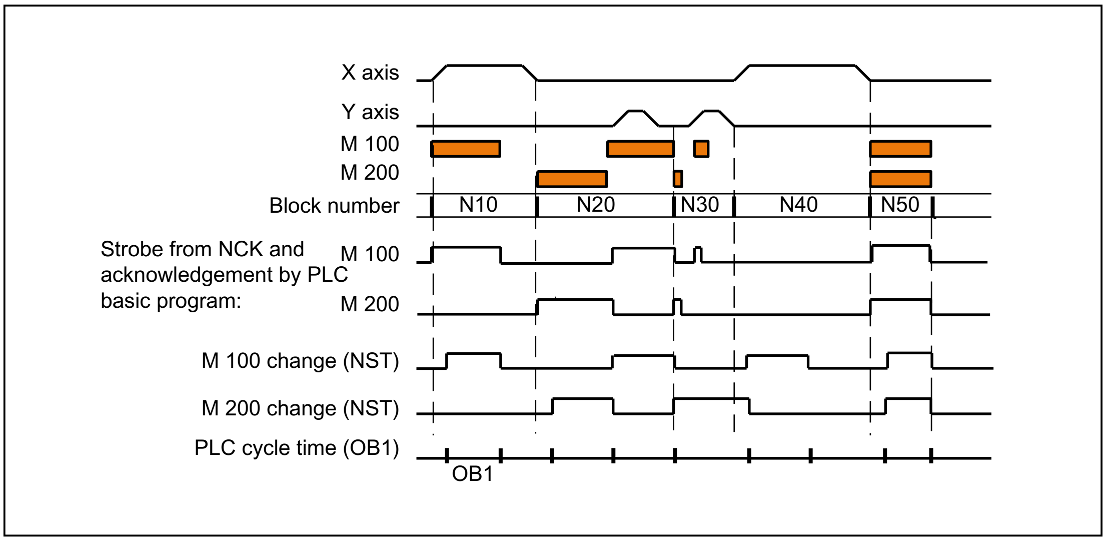
Different behavior for the output of the auxiliary functions M100 and M200 in a part program. The output behavior of the auxiliary functions is parameterized as follows:

- M100
  - Output duration one OB1 cycle (slow acknowledgment)
  - Output during motion
- M200
  - Output duration one OB1 cycle (slow acknowledgment)
  - Output prior to motion

Program code	Comment
`N10 G94 G01 X50 M100`	`; Output of M100: During motion`
	`; Acknowledgment: Slow`
`N20 Y5 M100 M200`	`; Output of M200: Piror to motion`
	`; Output of M100: During motion`
	`; Acknowledgment: Slow`
`N30 Y0 M=QU(100) M=QU(200)`	`; Output of M200: Piror to motion`
	`; Output of M100: During motion`
	`; Acknowledgment: Fast`
`N40 X0`	
`N50 M100 M200`	`; Output of M200: Immediately 1)`
	`; Output of M100: Immediately 1)`
	`; Acknowledgment: Slow`
`M17`	

1) Without a traversing motion, auxiliary functions are always output to the PLC immediately.

The following figure shows the time sequence of the part program. Please note the time difference during the processing of part program blocks N20 and N30.



9.9 Auxiliary function output to the PLC

Function

On output of an auxiliary function to the PLC, the following signals and values are transferred to the NC/PLC interface:

- Change signals
- "Address extension" parameter
- "Value" parameter

Data areas in the NC/PLC interface

The change signals and values of the auxiliary functions are within the following data areas in the NC/PLC interface:

- Change signals for auxiliary function transfer from NC channel:
  DB21, ... DBB58 - DBB67
- Transferred M and S functions:
  DB21, ... DBB68 - DBB112
- Transferred T, D and DL functions:
  DB21, ... DBB116 - DBB136
- Transferred H and F functions:
  DB21, ... DBB140 - DBB190
- Decoded M signals (M0 - M99):
  DB21, ... DBB194 - DBB206 (dynamic M functions)

For information on the access procedure to the NC/PLC interface, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)".

A detailed description of the above data areas in the NC/PLC interface can be found in:

References:
List Manual, Lists, Book 2; PLC User Interfaces,
Section: Channel-specific signals (DB 21 - DB 30)"

9.10 Auxiliary functions without block change delay

Function

For auxiliary functions with parameterized and/or programmed output behavior, too:

- "Output duration one OB40 cycle (quick acknowledgment)"
- "Output before the motion" or "Output during the motion"

there may be drops in velocity in continuos-path mode (short traverse paths and high velocities). This the system has to wait for acknowledgment of the auxiliary function by the

PLC toward the end of the block. To avoid these velocity drops, the block change can be made irrespective of whether such auxiliary functions have been acknowledged:

Parameter assignment

Suppression of the block change delay with quick auxiliary functions is set via the machine data:

MD22100 $MC_AUXFU_QUICK_BLOCKCHANGE (block change delay with quick auxiliary functions)

Value	Meaning
0	In the case of quick auxiliary function output to the PLC, the block change is delayed until acknowledgment by the PLC (OB40).
1	In the case of quick auxiliary function output to the PLC, the block change is not delayed.

Boundary conditions

Synchronism of auxiliary functions that are output without a block change delay is no longer ensured for the part program block in which they are programmed. In the worst case scenario, acknowledgment comes one OB40 cycle and execution of the auxiliary function comes one OB1 cycle after the change to the next part program block.

9.11 M function with an implicit preprocessing stop

Function

Triggering a preprocessing stop in conjunction with an auxiliary function can be programmed explicitly via the `STOPRE` part program command. Always triggering a preprocessing stop in M function programming can be parameterized for each M function via the following machine data:

MD10713 $MN_M_NO_FCT_STOPRE[<n>] (M function with preprocessing stop)

Example

The user-defined M function M88 is intended to trigger a preprocessing stop.

Parameterization:

MD10713 $MN_M_NO_FCT_STOPRE [0] = 88

Application:

Part program (extract)

```
Program code             Comment
...
N100 G0 X10 M88          ; Traversing motion and implicit preprocessing stop via M88.
```

Program code	Comment
`N110 Y=R1`	`; N110 is only interpreted after the traversing motion has been completed and the M function has been acknowledged.`
`...`	

Supplementary conditions

If a subprogram is 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 subprogram)
- M98 (ISO dialect T / ISO dialect M)

9.12 Response to overstore

Overstore

On the SINUMERIK operator interface, before starting the following functions:

- NC START of a part program
- NC START to resume an interrupted part program

the auxiliary functions that are output at the start can be changed by the "Overstore" function.

Possible applications include:

- Addition of auxiliary functions after block search
- Restoring the initial state to position a part program

Types of auxiliary functions that can be overstored

The following types of auxiliary functions can be overstored:

- M (special function)
- S (spindle speed)
- T (tool number)
- H (aux. function)
- D (tool offset number)
- DL (additive tool offset)
- F (feed)

Duration of validity

An overstored auxiliary function, e.g. M3 (spindle right), is valid until it is overwritten by another auxiliary function from the same auxiliary function group, by additional overstoring or by programming in a part program block.

9.13 Behavior during block search

9.13.1 Auxiliary function output during type 1, 2, and 4 block searches

Output behavior

In the case of type 1, 2, and 4 block searches, the auxiliary functions are collected on the basis of specific groups. The last auxiliary function in each auxiliary function group is output after NC-START in a separate part program block before the actual reentry block, and has the following output behavior:

- Output duration of one OB1 cycle (normal acknowledgement)
- Output prior to motion

Output control

Whether or not the auxiliary function is output to the PLC after a block search can be configured via bit 8 of the machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]
  (output behavior of predefined auxiliary functions)
  where <n> = system function index (0 ... 32)
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]
  (output behavior of user-defined auxiliary functions)
  where <n> = auxiliary function index (0 ... 254)
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]
  (output behavior of the auxiliary functions in a group)
  where <n> = group index (0 ... 63)

Bit	Value	Meaning
10	0	Output during type 1, 2, and 4 block searches
	1	No output during type 1, 2, and 4 block searches

This behavior does not affect the display and does not affect variables $AC_AUXFU_STATE[<n>], $AC_AUXFU_VALUE[<n>], and $AC_AUXFU_EXT[<n>].
The auxiliary functions are always regarded as collected after a block search, even though they are not output to the PLC.

During collection, an auxiliary function that is not output after a block search also overwrites an auxiliary function whose bit 8 is not set.

The user can scan the collected auxiliary functions after a block search and, under certain circumstances, output them again by means of the subprogram or synchronized actions.

Note

The following auxiliary functions are not collected:

- Auxiliary functions which are not assigned to any auxiliary function group.
- Auxiliary functions which are assigned to the first auxiliary function group.

Overstorage of auxiliary functions

After completion of a block search, the collected auxiliary functions are ouput on the next NC-START. If it is necessary to output additional auxiliary functions, they can be added via the "Overstore" function (see Section "Response to overstore (Page 442)").

M19 behavior (position spindle)

After a block search, the last spindle positioning command programmed with M19 is always carried out, even if other spindle-specific auxiliary functions are programmed between the part program with M19 and the target block. Setting the necessary spindle enables must therefore be derived from the interface signals of the traverse commands in the PLC user program:

DB31, ... DBX64.6/64.7 (traversing command minus/plus)

In this case, the spindle-specific auxiliary functions M3, M4, and M5 are not suitable because they might not be output to the PLC until after the spindle positioning.

For detailed information on the block search, see Section "K1: Mode group, channel, program operation, reset response (Page 479)".

9.13.2 Assignment of an auxiliary function to a number of groups

Function

User-defined auxiliary functions can also be assigned to multiple groups via the group assignment (MD22000 $MC_AUXFU_ASSIGN_GROUP). During the block search these auxiliary functions are collected for all the configured groups.

Note

Predefined auxiliary functions can only be assigned to one group.

Example

The DIN includes the following M-commands for coolant output:

- M7: Coolant 2 ON
- M8: Coolant 1 ON
- M9: Coolants 1 and 2 OFF

Consequently, both coolants can also be active together:

- If M7 and M8 are collected in two separate groups (e.g. groups 5 and 6)
- If M9 has to be assigned to these two groups, e.g.
  - Group 5: M7, M9
  - Group 6: M8, M9

Parameterization:

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN = 4

MD22000 $MC_AUXFU_ASSIGN_GROUP [0] = 5

MD22000 $MC_AUXFU_ASSIGN_GROUP [1] = 5

MD22000 $MC_AUXFU_ASSIGN_GROUP [2] = 6

MD22000 $MC_AUXFU_ASSIGN_GROUP [3] = 6

MD22010 $MC_AUXFU_ASSIGN_TYPE [0] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [1] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [2] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [3] = M

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [0] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [1] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [2] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [3] = 0

MD22030 $MC_AUXFU_ASSIGN_VALUE [0] = 7

MD22030 $MC_AUXFU_ASSIGN_VALUE [1] = 9

MD22030 $MC_AUXFU_ASSIGN_VALUE [2] = 8

MD22030 $MC_AUXFU_ASSIGN_VALUE [3] = 9

MD22035 $MC_AUXFU_ASSIGN_SPEC [0] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [1] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [2] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [3] = 'H121'

Part program (section):

Program code
...

Program code
N10 ... M8
N20 ... M9
N30 ... M7
...

During the block search, the auxiliary function M9 is collected for groups 5 and 6.

Scan of the collected M auxiliary functions:

M function of the fifth group: $AC_AUXFU_M_VALUE [4] = 7

M function of the sixth group: $AC_AUXFU_M_VALUE [5] = 9

9.13.3 Time stamp of the active M auxiliary function

When outputting collected auxiliary functions following a block search, attention must be paid to the sequence during collecting. For this reason, each group is assigned a time stamp which can be queried on a group-specific basis by way of the system variable below:

$AC_AUXFU_M_TICK[<n>] (time stamp of the active M auxiliary function)

9.13.4 Determining the output sequence

Function

The following predefined procedure is provided to simplify the process of determining the output sequence of M auxiliary functions for the programmer:

```
AUXFUMSEQ(VAR INT _NUM_IN, VAR INT _M_IN[], VAR INT _EXT_IN[], VAR
INT _NUM_OUT, VAR INT _M_OUT[], VAR INT _EXT_OUT[])
```

Input parameters:

`VAR INT _NUM_IN`:	Number of relevant M commands
`VAR INT _M_IN[]`:	Field of relevant M codes
`VAR INT _EXT_IN[]`:	Field of relevant M address extensions

Output parameters:

`VAR INT _NUM_OUT`:	Number of determined M codes
`VAR INT _M_OUT[]`:	Field of determined M codes
`VAR INT _EXT_OUT[]`:	Field of determined M address extensions

The function determines the sequence in which the M auxiliary functions, which have been collected on a group-specific basis, are output for the predefined M codes. The sequence is determined from the collection times $AC_AUXFU_M_TICK[<n>] (see Section "Time stamp of the active M auxiliary function (Page 446)").

A particular M code is only taken into account once, even if it belongs to more than one group. If the number of relevant M commands is less than or equal to 0, all the collected M codes are output. The number of relevant M commands is limited to 64.

Example

M commands for coolant output:

- M7: Coolant 2 ON
- M8: Coolant 1 ON
- M9: Coolants 1 and 2 OFF

Group assignment:

- Group 5: M7, M9
- Group 6: M8, M9

Part program (section):

Program code

```
...
N10 ... M8
N20 ... M9
N30 ... M7
...
```

During block searches, the auxiliary functions are collected on the basis of specific groups. The last auxiliary function in an auxiliary function group is output to the PLC following a block search:

- Group 5: M7
- Group 6: M9

If they are output in the sequence M7 → M9, no coolant is then active. However, coolant 2 would be active during the execution of the program. Therefore, the correct output sequence for the M auxiliary functions is determined with an ASUP which contains the predefined procedure `AUXFUMSEQ(…)`:

Program code

```
DEF INT _I, _M_IN[3], _EXT_IN[3], _NUM_OUT, _M_OUT[2], _EXT_OUT[2]
_M_IN[0]=7 _EXT_IN[0]=0
_M_IN[1]=8 _EXT_IN[1]=0
_M_IN[2]=9 _EXT_IN[2]=0
AUXFUMSEQ(3,_M_IN,_EXT_IN,_NUM_OUT,_M_OUT,_EXT_OUT)
FOR _I = 0 TO _NUM_OUT-1
   M[_EXT_OUT[_I]]=_M_OUT[_I]
ENDFOR
```

9.13.5 Output suppression of spindle-specific auxiliary functions

Function

In certain situations, such as a tool change, it may be necessary not to output the spindle-specific auxiliary functions collected during the block search in action blocks, but to delay output, for example, until after a tool change. The automatic output of the spindle-specific auxiliary functions after a block search may be suppressed for this purpose. Output can then be performed manually later by overstoring or by an ASUP.

Parameterization

Suppression of the automatic output of the spindle-specific auxiliary functions after a block search is set via machine data:

MD11450 $MN_SEARCH_RUN_MODE (behavior after a block search)

Bit	Value	Meaning
2	0	The output of the spindle-specific auxiliary functions is performed in the action blocks.
	1	Output of the auxiliary functions is suppressed in the action blocks.

System variables

The spindle-specific auxiliary functions are always stored in the following system variables during block searches, irrespective of the parameter assignment described above:

System variable	Description	
$P_SEARCH_S [<n>]	Accumulated spindle speed	
	Value range:	0 ... Smax
$P_SEARCH_SDIR [<n>]	Accumulated spindle direction of rotation	
	Value range:	3, 4, 5, -5, -19, 70
$P_SEARCH_SGEAR [<n>]	Accumulated spindle gear stage M function	
	Value range:	40 ... 45
$P_SEARCH_SPOS [<n>]	Accumulated spindle position	
	Value range:	0 ... MD30330 $MA_MODULO_RANGE (size of the module range)
	or	
	Accumulated traversing path	
	Value range:	-100,000,000 ... 100,000,000
$P_SEARCH_SPOSMODE [<n>]	Accumulated position approach mode	
	Value range:	0 ... 5

For later output of the spindle-specific auxiliary functions, the system variables can be read in an ASUP, for example, and output after the action blocks are output:

DB21, ... DBX32.6 = 1 (last action block active)

Note

The contents of the system variables $P_S, $P_DIR and $P_SGEAR may be lost after block search due to synchronization operations.

For more detailed information on ASUP, block search and action blocks, see Section "K1: Mode group, channel, program operation, reset response (Page 479)".

Example

Block search for contour with suppression of output of the spindle-specific auxiliary functions and start of an ASUP after output of action blocks.

Parameterization: MD11450 $MN_SEARCH_RUN_MODE, bit 2 = 1

After the block search on N55, the ASUP is started.

Part program:

Program code	Comment
N05 M3 S200	; Spindle 1:
N10 G4 F3	
N15 SPOS=111	; Spindle 1 is positioned to 111 degrees in the ASUP
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	

ASUP:

Program code	Comment
PROC ASUP_SAVE	
MSG ("Output of the spindle functions")	
DEF INT SNR=1	
AUSG_SPI:	
M[SNR]=$P_SEARCH_SGEAR[SNR]	; Output gear stage.
S[SNR]=$P_SEARCH_S[SNR]	; Output speed (for M40, a suitable gear stage is determined).
M[SNR]=$P_SEARCH_SDIR[SNR]	; Output direction of rotation, positioning or axis mode.
SNR=SNR+1	; Next spindle.
REPEAT AUSG_SPI P=$P_NUM_SPINDLES-1	; For all spindles.
MSG("")	
REPOSA	
RET	

Explanation of example

If the number of spindles is known, outputs of the same type can be written in one part program block to reduce program runtime.

Output of $P_SEARCH_SDIR should be made in a separate part program block because spindle positioning or switchover to axis mode in conjunction with the gear change can cause an alarm.

If the ASUP which has been started is ended with `REPOSA`, spindle 1 remains at position 111 degrees, while spindle 2 is repositioned at position 77 degrees.

If a different response is required, the program sequence for block search (for example) "`N05 M3 S`..." and "`N30 SPOS[2] = IC(...)`" requires special treatment.

Whether block search is active can be ascertained in the ASUP via the system variable $P_SEARCH.

```
$P_SEARCH==1 ; Block search active
```

In the case of incremental positioning after speed control operation, the path to be traversed is defined but, in some cases, the final position reached only becomes known during positioning. This is the case, for example, during position calibration on crossing the zero mark when switching on position control. For this reason, the distance programmed after the zero position is accepted as the REPOS position (`REPOSA` in the ASUP).

Supplementary conditions

Collected S values

The meaning of an S value in the part program depends on the feed type that is currently active:

`G93`, `G94`, `G95`, `G97`, `G971`:	The S value is interpreted as the speed
`G96`, `G961`:	The S value is interpreted as a constant cutting rate

If the feed operation is changed (e.g. for a tool change) before output of the system variable $P_SEARCH_S, the original setting from the target block in the part program must be restored to avoid use of the wrong type of feed.

Collected direction of rotation

For output of the direction of rotation, the system variable $P_SEARCH_SDIR is assigned default value "-5" at the start of the block search. This value has no effect on output.

This ensures that the last spindle operating mode is retained for a block search across program sections in which spindles are not programmed with a direction of rotation, positioning or axis mode.

The programming of M19, SPOS, and SPOSA is collected as "M-19" (internal M19) in the system variables $P_SEARCH_SDIR as an alternative to M3, M4, M5, and M70.

For the output of "M-19", the positioning data is read internally from the system variables $P_SEARCH_SPOS and $P_SEARCH_SPOSMODE. Both system variables can also be written to, in order, for example, to make corrections.

Note

Because of the assignments described above (e.g. M[<n>] = $P_SEARCH_SDIR[<n>]), the values "-5" and "19" generally remain hidden from the user and only have to be observed in the case of special evaluation of the system variables in the ASUP.

9.13.6 Auxiliary function output with a type 5 block search (SERUPRO)

Output behavior

In the case of type 5 block searches (SERUPRO), an auxiliary function can be output to the PLC during the block search and/or collected on a group-specific basis in the following system variables:

- $AC_AUXFU_PREDEF_INDEX[<n>] (index of a predefined auxiliary function)
- $AC_AUXFU_TYPE[<n>] (type of auxiliary function)
- $AC_AUXFU_STATE[<n>] (output state of the auxiliary function)
- $AC_AUXFU_EXT[<n>] (address extension of the auxiliary function)
- $AC_AUXFU_VALUE[<n>] (value of the auxiliary function)

For a description of the system variables, see Section "Querying system variables (Page 464)".

Output control

Whether an auxiliary function is output to the PLC during a type 5 block search (SERUPRO) and/or collected on a group-specific basis in the following system variables can be configured via bits 9 and 10 of the machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]
  (output behavior of predefined auxiliary functions)
  where <n> = system function index (0 ... 32)
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]
  (output behavior of user-defined auxiliary functions)
  where <n> = auxiliary function index (0 ... 254)
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]
  (output behavior of the auxiliary functions in a group)
  where <n> = group index (0 ... 63)

Bit	Value	Meaning
9	0	No collection during type 5 block searches (SERUPRO)
	1	Collection during type 5 block searches (SERUPRO)
10	0	Output during type 5 block searches (SERUPRO)
	1	No output during type 5 block searches (SERUPRO)

Output counter

The user can output the collected auxiliary functions to the PLC on a channel-by-channel basis in the block search ASUP. For the purposes of serialized output via multiple channels, the three output counters are changed across all the channels each time an auxiliary function is output:

System variable	Meaning		
$AC_AUXFU_TICK[<n>,<m>]	Output counter of the active auxiliary function		
	Index	**Meaning**	
	<n>	Group index (0 … 63)	
	<m>	Output counter (0 ... 2)	
		Val-ue	**Meaning**
		0	Output sequence counter (all outputs within an IPO cycle)
		1	Package counter within an output sequence in the IPO cycle
		2	Auxiliary function counter within a package

Explanation

- An auxiliary function package comprises a maximum of ten auxiliary functions.
- Two packages can be processed per IPO cycle in each channel during SERUPRO because synchronized actions are processed in this cycle.
- An output sequence of up to a maximum of 20 packages (2 packages per channel * 10 channels) can be processed within an IPO cycle across all channels.

The encoding indicates how many auxiliary function packages and, within these, how many auxiliary functions have been processed during the same IPO cycle:

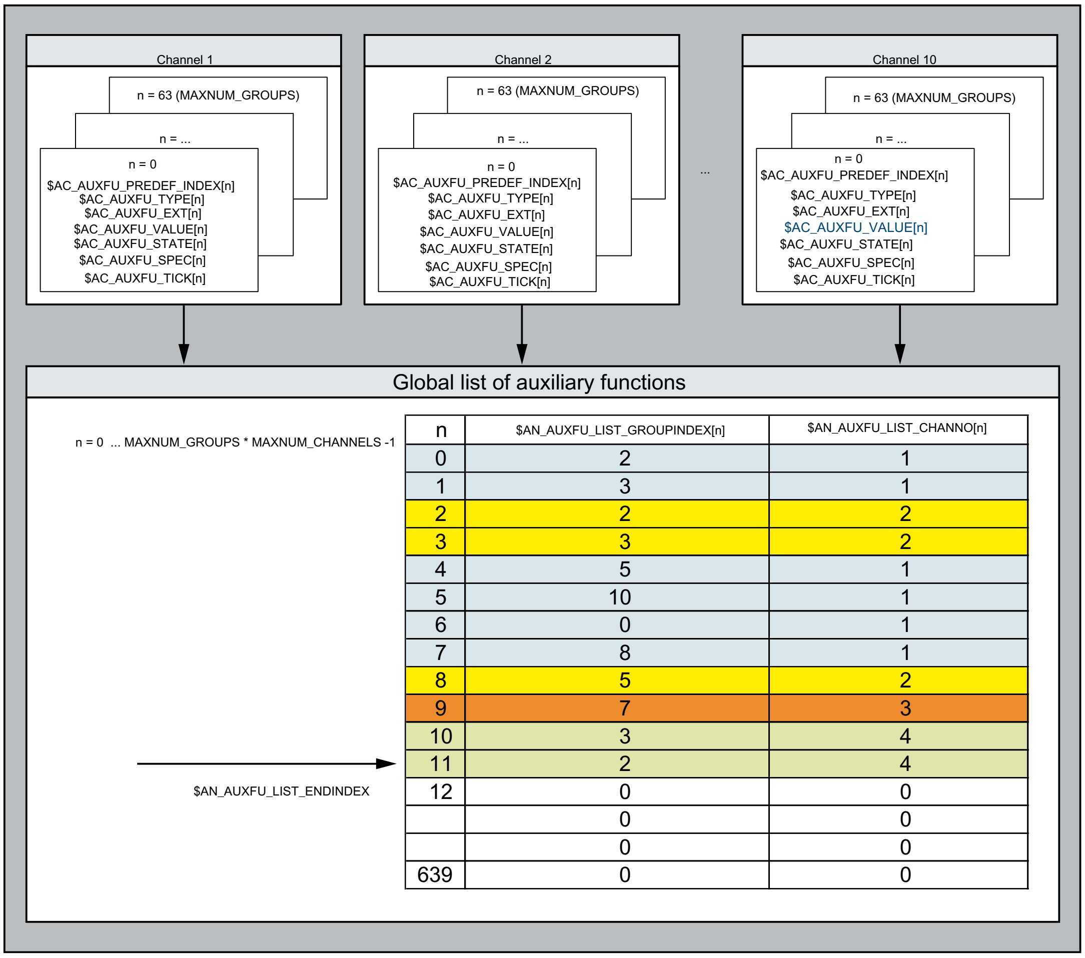
- Auxiliary functions which have been collected in one IPO cycle have the same sequence counter.
- Auxiliary functions which have been collected in one package (block or synchronized action) all have the same package counter.

The total on the auxiliary function counter increases every time an auxiliary function is collected.

Global list of auxiliary functions

At the end of SERUPRO, the auxiliary functions collected on a group-specific basis in the individual channels are entered in a cross-channel list with the channel number and group index according to their counter state.

System variable 1)	Meaning
$AC_AUXFU_TICK[<n>,<m>]	Counter value
$AN_AUXFU_LIST_CHANNO[<n>] 2)	Channel number
$AN_AUXFU_LIST_GROUPINDEX[<n>] 2)	Group index
1) Value range index <n>: 0 ... MAXNUM_GROUPS * MAXNUM_CHANNELS - 1	
2) The system variables can be read and written.	



n	$AN_AUXFU_LIST_GROUPINDEX[n]	$AN_AUXFU_LIST_CHANNO[n]
0	2	1
1	3	1
2	2	2
3	3	2
4	5	1
5	10	1
6	0	1
7	8	1
8	5	2
9	7	3
10	3	4
11	2	4
12	0	0
	0	0
	0	0
639	0	0

The global list is structured on the basis of the sequence in which the search target was found. It is intended to be used as a system proposal for auxiliary functions to be output in the following ASUP at the end of SERUPRO. If an auxiliary function is not to be output, the corresponding group index is to be set to "0".

Behavior regarding spindle auxiliary functions

Following the start of the block search, all the channels collect the auxiliary functions in the channel variables on a group-specific basis. In order to perform a far-reaching restoration of the spindle state in the SERUPRO target block using the collected auxiliary functions, the last active auxiliary function in any group of spindle auxiliary functions must characterize the state of the spindle in the target block. In the case of transitions in spindle states, obsolete auxiliary functions are deleted or, if necessary, implicit auxiliary functions are entered.

All the spindle auxiliary functions from the global auxiliary function list must correspond to the spindle states achieved in the target block to enable the auxiliary functions to be processed when the list is output and to ensure that no alarms or unintended spindle states are requested which could prevent the continuation of the part program.

This affects the groups of auxiliary functions for any spindle configured in the system, whereby the spindle number corresponds to an auxiliary function's address extension.

Group a:	M3, M4, M5, M19, M70
Group b:	M40, M41, M42, M43, M44, M45
Group c:	S

Deleting obsolete auxiliary functions

In the functions below, the auxiliary functions from group a are deleted for the spindle concerned:

- For the following spindle when a generic coupling, such as COUPON, TRAILON, EGON, etc. is switched on

Generating implicit auxiliary functions from group a

In the functions below, the auxiliary functions from group a are generated implicitly for the spindle concerned:

- For the following spindle when the synchronous spindle coupling is switched off
  - COUPOF generates M3, M4 and S or M5 in the main run depending on the coupling situation.
  - COUPOF(S<n>, S<m>, POS) and COUPOFS(S<n>, S<m>, POS, POS) generate M3, M4 and S.
  - COUPOFS generates M5 in the main run.
  - COUPOFS(S<n>, S<m>, POS) generates M19 in the main run.
    The implicit M19 ("SPOS[<address extension>] = IC(0)" in the ASUP) activates the positioning mode without a traversing motion.
- M70 is generated during a traversing motion as an axis or during the transition to axis mode by selecting a transformation during which the spindle enters as an axis.
- M5 is generated during SPCOF.

Note

Within the context of the "axis interchange" and "axis container rotation" functions, the auxiliary functions for programming the spindle must always be specified in a way which ensures compatibility with the actual (motor) state during interchange/rotation. A distinction is made here between the axis interchange and axis container mechanisms.

Example of axis container rotation:

An axis container has four spindles, each assigned to a separate channel (1 - 4). M3 S1000 is always programmed in channel 1, and an axis container rotation is then executed. The other channels do not perform any spindle programming. After the 3rd axis container rotation and the 4th spindle programming, M3, all four spindles rotate clockwise at a speed of 1000 rpm. If the end of the SERUPRO now lies within this range, every ASUP for a channel is expected to contain an M3 S1000 for the local spindle.

During interchange however, the collected auxiliary functions may only be assigned to the channel where the spindle is currently located.

Cross-channel auxiliary function

An auxiliary function can also be collected on a cross-channel basis in the global auxiliary function list in the case of type 5 block searches (SERUPRO). Only the last auxiliary function collected from this group (highest counter state) is entered in the global list.

The appropriate configuring is performed with the following machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>], Bit 11
  (output behavior of predefined auxiliary functions)
  where <n> = system function index
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>], Bit 11
  (output behavior of user-defined auxiliary functions)
  where <n> = auxiliary function index
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>], Bit 11
  (output behavior of the auxiliary functions in a group)
  where <n> = group index

Bit	Value	Meaning
11	0	Channel-specific collection
	1	Cross-channel collection

The spindle auxiliary functions are filtered out beforehand at the end of the block search depending on the spindle state. The channel data is updated accordingly. The global auxiliary function list can be processed sequentially in the ASUPs at the end of the SERUPRO, and the sorted auxiliary functions can be output with channel synchronization.

Querying the last auxiliary function collected

The index of the last auxiliary function collected in the global list can be queried using the system variable $AN_AUXFU_LIST_ENDINDEX.

9.13.7 ASUB at the end of the SERUPRO

Function

After completing the block search with the program test (SERUPRO), before starting the subsequent processing, the auxiliary functions collected during the block search must be output. For this purpose, during the block search, the auxiliary functions are collected in a global list. The SERUPRO end ASUPs generate the corresponding part program blocks channel-specific from this list. This ensures that the collected auxiliary functions can be output both channel-specific as well as cross-channel in the correct sequence. A fully functional SERUPRO end ASUP is a component of the NC software.

Users/machinery construction OEMs can change the SERUPRO end ASUP. The subsequently described functions support processing the global list of auxiliary functions and generating the part program blocks required for synchronized auxiliary function output.

Function AUXFUSYNC(...)

Function:

The function `AUXFUSYNC` generates a complete part program block as string from the global list of auxiliary functions at each call. The part program block either contains auxiliary functions or commands to synchronize auxiliary function outputs (`WAITM`, `G4`, etc.).

The function triggers a preprocessing stop.

Syntax:

```
PROC AUXFUSYNC(VAR INT <NUM>, VAR INT <GROUPINDEX>[10], VAR
STRING[400] <ASSEMBLED>)
```

Parameters:

`<NUM>`:	Contains information about the part program block, supplied in parameter `<ASSEMBLED>` or the auxiliary functions contained in it. Value range: -1, 0, 1 ... 10

Value	Meaning
≥1	Number of auxiliary functions contained in the part program block
0	Part program block without auxiliary functions, e.g. `WAITM`, `G4`
-1	End identifier. The global list of auxiliary functions has been completely processed for the actual channel.

`<GROUP INDEX>`:	Contains the indices of the auxiliary function groups contained in the part program block. With index = number of the auxiliary function group - 1
`<ASSEMBLED>`:	Contains the complete part program block for the channel-specific SERUPRO end ASUP as string.

Further information:

If auxiliary functions were collected via a synchronized action, two NC blocks are generated. One NC block to output the auxiliary functions. An executable NC block via which the NC block is transported to the main run to output the auxiliary functions:

1. Output of the auxiliary functions via synchronized action, e.g.: `WHEN TRUE DO M100 M102`
2. Executable NC block, e.g.: `G4 F0.001`

Function AUXFUDEL(...)

Function:

The function AUXFUDEL deletes the specified auxiliary function from the global list of auxiliary functions channel-specific for the calling channel. Deletion is realized by setting the corresponding group index ...`GROUPINDEX[n]` to 0.

The function must be called before calling `AUXFUSYNC`.

The function triggers a preprocessing stop.

Syntax:

```
PROC AUXFUDEL(CHAR <TYPE>, INT <EXTENSION>, REAL <VALUE>, INT
<GROUP>)
```

Parameters:

`<TYPE>`:	Type of auxiliary function to be deleted
`<EXTENSION>`:	Address extension of the auxiliary function to be deleted
`<VALUE>`:	Value of auxiliary function to be deleted
`<GROUP>`:	Number of the auxiliary function group

Function AUXFUDELG(...)

Function:

The function AUXFUDELG deletes all auxiliary functions of the specified auxiliary function group from the global list of auxiliary functions channel-specific for the calling channel. Deletion is realized by setting the corresponding group index ...`GROUPINDEX[n]` to 0.

The function must be called before calling `AUXFUSYNC`.

The function triggers a preprocessing stop.

Syntax:
```
PROC AUXFUDELG(INT <GROUP>)
```

Parameters:

`<GROUP>`:	`Number of the auxiliary function group`

Multi-channel block search

NOTICE

Multi-channel block search and AUXFUDEL / AUXFUDELG

If, for a multi-channel block search in the SERUPRO end ASUPs, auxiliary functions with AUXFUDEL / AUXFUDELG are deleted from the global list of auxiliary functions, before calling the AUXFUSYNC function, the channels involved must be synchronized. The synchronization ensures that before calling the AUXFUSYNC, all delete requests are processed and a consistent list is available.

Examples

Two examples for configuring a user-specific SERUPRO end ASUP.

Example 1: Deleting auxiliary functions and generating the auxiliary function output with AUXFUSYNC(...)

Program code	Comment
`N10 DEF STRING[400] ASSEMBLED=""`	
`N20 DEF STRING[31] FILENAME="/_N_CST_DIR/_N_AUXFU_SPF"`	
`N30 DEF INT GROUPINDEX[10]`	
`N40 DEF INT NUM`	
`N60 DEF INT ERROR`	

```
Program code                                         Comment
N90
N140 AUXFUDEL("M",2,3,5)                             ; M2=3 (5th auxiliary function group) delete
N150
N170 AUXFUDELG(6)                                    ; Delete the collected auxiliary function of the
                                                     ; 6th group.
N180
N190 IF ISFILE(FILENAME)
N210    DELETE(ERROR,FILENAME)                       ; Delete the FILENAME file
N220    IF (ERROR<>0)                                ; Error evaluation
N230       SETAL(61000+ERROR)
N240    ENDIF
N250 ENDIF
; CAUTION!
; If, for a multi-channel block search, auxiliary functions with AUXFUDEL/AUXFUDELG
; are deleted from the global list of auxiliary functions, before the loop to
; generate the subprogram FILENAME with AUXFUSYNC, the channels must be synchronized.
   The synchronization ensures that all delete requests were processed
; in all channels and a consistent list is available.
; Example: WAITM(99,1,2,3)
N270 LOOP
N300    AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)          ; Generate a part program block
N310
N320    IF (NUM==-1)                                 ; All auxiliary functions of the channel
                                                     ; have been executed.
N340       GOTOF LABEL1
N350    ENDIF
N380    WRITE(ERROR,FILENAME,ASSEMBLED)              ; Write a part program block to file FILENAME.
N390    IF (ERROR<>0)                                ; Error evaluation
N400       SETAL(61000+ERROR)
N410    ENDIF
N430 ENDLOOP
N440
N450 LABEL1:
N460
N480 CALL FILENAME                                   ; Execute a generated subprogram.
N490
N510 DELETE(ERROR,FILENAME)                          ; Delete the file again after execution.
N520 IF (ERROR<>0)
N530    SETAL(61000+ERROR)
N540 ENDIF
N550
N560 M17
```

Example 2: Deleting auxiliary functions and generating the auxiliary function output without AUXFUSYNC(...)

```
Program code                                         Comment
N0610 DEF STRING[400] ASSEMBLED=""
N0620 DEF STRING[31] FILENAME="/_N_CST_DIR/_N_AUXFU_SPF"
N0630 DEF INT GROUPINDEX[10]
N0640 DEF INT NUM
N0650 DEF INT LAUF
N0660 DEF INT ERROR
N0670 DEF BOOL ISQUICK
N0680 DEF BOOL ISSYNACT
N0690 DEF BOOL ISIMPL
...
N0760 AUXFUDEL("M",2,3,5)                            ; M2=3 (5th auxiliary function group) delete
N0770
N0790 AUXFUDELG(6)                                   ; Delete the collected auxiliary function of the
                                                     ; 6th group.
N0800
N0810 IF ISFILE(FILENAME)
N0830    DELETE(ERROR,FILENAME)                      ; File already exists and must be
                                                     ; deleted.
N0840    IF (ERROR<>0)
N0850       SETAL(61000+ERROR)
N0860    ENDIF
N0870 ENDIF
N0880
; CAUTION!
; If, for a multi-channel block search, auxiliary functions with AUXFUDEL/AUXFUDELG
; are deleted from the global list of auxiliary functions, before the loop to
; generate the subprogram FILENAME with AUXFUSYNC, the channels must be synchronized.
   The synchronization ensures that all delete requests were processed
; in all channels and a consistent list is available.
; Example: WAITM(99,1,2,3)
N0890 LOOP
N0920   AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)          ; Procedure to generate
                                                     ; auxiliary function blocks from the global
                                                     ; auxiliary function list.
N0930
N0940   IF (NUM==-1)                                 ; All auxiliary functions of the channel
                                                     ; have been executed.
N0960     GOTOF LABEL1
N0970   ENDIF
N0980
N1000   IF (NUM>0)                                   ; If auxiliary functions are output,
                                                     ; the block is generated.
```

```
Program code                                         Comment
N1010
N1020     ASSEMBLED=""
N1030
N1050     FOR LAUF=0 TO NUM-1                        ; Collected auxiliary functions for a
                                                     ; block.
N1060
N1080       IF GROUPINDEX[LAUF]<>0                   ; Auxiliary functions deleted from the
                                                     ; global list have the group index 0.
N1090
N1100         ISQUICK=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2'
N1110
N1120         ISSYNACT=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H1000'
N1130
N1140         ISIMPL=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2000'
N1150
N1180         IF ISSYNACT                            ; Assemble a block for the M auxiliary
                                                     ; function output
N1190           ASSEMBLED= ASSEMBLED << "WHEN TRUE DO "
N1200         ENDIF
N1210 ; Implicitly generated M19 is mapped to SPOS[SPI(<Spindle no.>)] = IC(0)
N1230         IF (ISIMPL AND ($AC_AUXFU_VALUE[GROUPINDEX[LAUF]==19))
N1240           ASSEMBLED= ASSEMBLED << "SPOS[SPI(" <<
                  $AC_AUXFU_EXT[GROUPINDEX[LAUF]] << ")=IC(0)"
N1260         ELSE
N1270           ASSEMBLED= ASSEMBLED << "M[" << $AC_AUXFU_EXT[GROUPINDEX[LAUF]] << "]="
N1280
N1290           IF ISQUICK
N1300             ASSEMBLED= ASSEMBLED << "QU("
N1310           ENDIF
N1320
N1330           ASSEMBLED= ASSEMBLED << $AC_AUXFU_VALUE[GROUPINDEX[LAUF]]
N1340
N1350           IF ISQUICK
N1360             ASSEMBLED= ASSEMBLED << ")"
N1370           ENDIF
N1380         ENDIF
N1400       ENDIF
N1420     ENDFOR
N1430
N1450     WRITE(ERROR,FILENAME,ASSEMBLED)            ; Write an auxiliary function block to a file.
N1460
N1470     IF ISSYNACT
N1480       ASSEMBLED="G4 F0.001"
N1490       WRITE(ERROR,FILENAME,ASSEMBLED)
```

Program code	Comment
`N1500     ENDIF`	
`N1510`	
`N1520   ELSE`	
`N1540     WRITE(ERROR,FILENAME,ASSEMBLED)`	; Write an auxiliary function block to a file.
`N1550   ENDIF`	
`N1560`	
`N1570 ENDLOOP`	
`N1580`	
`N1590 LABEL1:`	
`N1600`	
`N1620 CALL FILENAME`	; Execute a generated subprogram.
`N1630`	
`N1650 DELETE(ERROR,FILENAME)`	; Delete the file again after execution.
`N1660 IF (ERROR<>0)`	
`N1670   SETAL(61000+ERROR)`	
`N1680 ENDIF`	
`N1690`	
`N1700 M17`	

9.14 Implicitly output auxiliary functions

Function

Implicitly output auxiliary functions are auxiliary functions which have not been programmed explicitly and which are also output by other system functions (e.g. transformation selection, tool selection, etc.). These implicit auxiliary functions do not lead to any system function; instead, the M codes are collected according to output behavior parameters assigned to them and/or are output to the PLC.

Parameterization

The M codes for auxiliary functions to be output implicitly are defined with the machine data:

- MD22530 $MC_TOCARR_CHANGE_M_CODE (M code at toolholder change)
  This machine data value indicates the number of the M code which is output when a toolholder is activated at the NC/PLC interface.
  If the value is positive, the unchanged M code is always output.
  If the value is negative, the number of the toolholder is added to the machine data value, and this number is output.
- MD22532 $MC_GEOAX_CHANGE_M_CODE (M code when switching the geometry axes)
  Number of the M code which is output when the geometry axes on the NC/PLC interface are switched.
- MD22534 $MC_TRAFO_CHANGE_M_CODE (M code in the case of transformation changes)
  Number of the M code which is output during a transformation switch of the geometry axes at the NC/PLC interface.

Note

No M code is output if the number of the M code being output or the MD22530/MD22532/MD22534 value is between 0 and 6, or is either 17 or 30. Whether or not an M code which is generated in this manner leads to conflicts with other functions is not monitored.

Output behavior

In the case of implicitly output auxiliary functions, bit 13 is set in machine data MD22080 or MD22035 (output behavior of predefined or user-defined auxiliary functions).

This bit can be queried via the system variable $AC_AUXFU_SPEC[<n>].

Implicitly output auxiliary function M19

To achieve uniformity in terms of how M19 and SPOS or SPOSA behave at the NC/PLC interface, auxiliary function M19 can be output to the NC/PLC interface in the event of SPOS and SPOSA (see Section "General functionality (Page 1279)").

The implicitly output auxiliary function M19 is collected during the block search.

9.15 Information options

Information about auxiliary functions (e.g. about the output status) is possible via:

- The group-specific modal M auxiliary function display on the user interface
- Querying system variables in part programs and synchronized actions

9.15.1 Group-specific modal M auxiliary function display

Function

The output status and acknowledgment status of M auxiliary functions can be displayed on the user interface on a group-specific basis.

Requirements

To implement function-oriented acknowledgment and display of M auxiliary functions, the auxiliary functions must be managed in the PLC and, thus, in the user program itself. Therefore, it is up to the PLC programmer to program the acknowledgment of these auxiliary functions. The programmer must know which auxiliary functions in which group have to be acknowledged.

Standard

M auxiliary functions that are not managed by means of the PLC are identified by the NC as "transferred" and output to the PLC. There is no functional acknowledgment for these auxiliary functions. All M-auxiliary functions collected after a block search are also displayed so that the operator knows which auxiliary functions will be output after a start following a block search.

PLC activities

In the case of auxiliary function groups that are managed by the PLC itself, the PLC user program must acknowledge all auxiliary functions of this group at **Transfer** and **End of function**. The PLC programmer must know all the auxiliary functions of these groups.

Miscellaneous

Only the **group-specific** M auxiliary functions are displayed. The block-by-block display is also retained. Up to 15 groups can be displayed, whereby **only the last M function of a group** that was either collected or output to the PLC is displayed for each group. The M functions are presented in various display modes depending on their status:

Status	Display mode
Auxiliary function is collected	Inverted with yellow font
Auxiliary function is output from NC to PLC	Inverted
Auxiliary function has been transferred from NC to PLC and transport acknowledgment has taken place	Black font on gray background
Auxiliary function is managed by the PLC and has been taken over directly by the PLC	Black font on gray background
Auxiliary function is managed by the PLC and the function acknowledgment has taken place	Black font on gray background

Display update

The display is organized in such a way that the collected auxiliary functions are always output first, before those managed by the PLC and before those managed by the NC. A collected auxiliary function is marked as collected until it has been output from the NC to the PLC. PLC-managed auxiliary functions are retained until they are substituted by another auxiliary function. At a reset, only the collected auxiliary functions and the NC-managed auxiliary functions are deleted.

9.15.2 Querying system variables

Function

Auxiliary functions can be queried on a group-specific basis via system variables in the part program and via synchronized actions:

$AC_AUXFU_... [<n>] = <Value>

System variable	Meaning		
$AC_AUXFU_PREDEF_INDEX[<n>]	<Value>:	Index of the last auxiliary function collected for an auxiliary function group (block search) or the last predefined auxiliary function to be output	
		Type:	INT
		If no auxiliary function has been output yet for the specified group or if the auxiliary function is a user-defined auxiliary function, the variable supplies the value "-1".	
	<n>:	Group index (0 … 63)	
	Note: A predefined auxiliary function can be uniquely identified via this variable.		
$AC_AUXFU_TYPE[<n>]	<Value>:	Type of the last auxiliary function collected for an auxiliary function group (block search) or the last auxiliary function to be output	
		Type:	CHAR
	<n>:	Group index (0 … 63)	
$AC_AUXFU_EXT[<n>] or or M function-specific: $AC_AUXFU_M_EXT[<n>]	<Value>:	Address extension of the last auxiliary function collected for an auxiliary function group (block search) or the last auxiliary function to be output	
		Type:	INT
	<n>:	Group index (0 … 63)	

System variable	Meaning		
$AC_AUXFU_VALUE[<n>] or or M function-specific: $AC_AUXFU_M_VALUE[<n>]	<Value>:	Value of the last auxiliary function collected for an auxiliary function group (block search) or the last auxiliary function to be output	
		Type :	REAL
	<n>:	Group index (0 … 63)	
$AC_AUXFU_SPEC[<n>]	Value:	Bit-encoded output behavior according to MD22080/MD22035 (or `QU` programming) of the last auxiliary function collected for an auxiliary function group (block search) or the last auxiliary function to be output	
		Type :	INT
	<n>:	Group index (0 … 63)	
	Note: This variable can be used to determine whether the auxiliary function should be output with a fast acknowledgment.		
$AC_AUXFU_STATE[<n>] or or M function-specific: $AC_AUXFU_M_STATE[<n>]	<Value>:	Output state of the last auxiliary function collec-ted for an auxiliary function group (block search) or the last auxiliary function to be output	
		Type:	INT
		Value range:	0 ... 5
		0:	Auxiliary function does not exist
		1:	M-auxiliary function was collected via a block search
		2:	M-auxiliary function has been output to the PLC
		3:	M-auxiliary function has been output to the PLC and the transport acknowledgment has taken place
		4:	M-auxiliary function is managed by the PLC and has been taken over by the PLC
		5:	M-auxiliary function is managed by the PLC, and the function acknowledgment has taken place
	<n>:	Group index (0 … 63)	

Example

All M-auxiliary functions of the 1st group are to be stored in the order 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 the system variables, see:

List Manual, System Variables

9.16 Supplementary conditions

9.16.1 General constraints

Spindle replacement

Because the auxiliary functions are parameterized channel-specifically, if function: "spindle replacement" is used, the spindle-specific auxiliary function must be parameterized immediately in all channels that use the spindles.

Tool management

If tool management is active, the following constraints apply:

- T and M<k> functions are not output to the PLC.
  Note
  k is the parameterized value of the auxiliary function for the tool change (default: 6):
  MD22560 $MC_TOOL_CHANGE_M_CODE (auxiliary function for tool change)
- If no address extension is programmed, the auxiliary function refers to the master spindle or the master tool holder of the channel.
  Definition of the master spindle:
  - MD20090 $MC_SPIND_DEF_MASTER_SPIND
  - Part program instruction: SETMS

  Definition of the master tool holder
  - MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER
  - Part program instruction: SETMTH

Maximum number of auxiliary functions per part program block

A maximum of 10 auxiliary functions may be programmed in one part program block.

DL (additive tool offset)

The following restrictions apply to the DL function:

- Only one DL function can be programmed per part program block.
- If DL functions are used in synchronous actions, parameter: "Value" is not output to the PLC.

9.16.2 Output behavior

Thread cutting

During active thread cutting `G33`, `G34` and `G35`, the following output behavior is always active for the spindle-specific auxiliary functions:

- `M3` (spindle right)
- `M4` (spindle left)

- Output duration of one OB40 cycle (quick acknowledgement)
- Output during motion

The spindle-specific auxiliary function `M5` (spindle stop) is always output at the end of the block. The part program block that contains `M5` is always ended with exact stop, i.e. even during active continuous-path mode.

Synchronized actions

With output auxiliary functions from synchronized actions, the parameterized output behavior is ignored except for the following parameters:

- Bit0: Output duration of one OB1 cycle (normal acknowledgement)
- Bit1: Output duration of one OB40 cycle (quick acknowledgement)

Auxiliary functions: M17 or M2/M30 (end of subprogram)

In its own part program block

If one of the auxiliary functions `M17`, `M2` or `M30` is programmed as the only auxiliary function in a part program block and an axis is still in motion, the auxiliary function is not output to the PLC until after the axis has stopped.

Overriding the parameterized output behavior

The parameterized output behavior of the auxiliary functions `M17` or `M2`/`M30` is overridden by the output behavior that is determined in the following machine data:

MD20800 $MC_SPF_END_TO_VDI, bit 0 (subprogram end / stop to PLC)

Bit	Value	Meaning
0	0	The auxiliary functions `M17` or `M2`/`M30` (subprogram end) are not output to the PLC. Continuous-path mode is not interrupted at the end of the subprogram
	1	The auxiliary functions `M17` or `M2`/`M30` (subprogram end) are output to the PLC.

Auxiliary function: M1 (conditional stop)

Overriding the parameterized output behavior

The parameterized output behavior of the auxiliary function `M1` is overridden by the output behavior defined in the following machine data:

MD20800 $MC_SPF_END_TO_VDI, bit 1 (subprogram end / stop to PLC)

Bit	Value	Meaning
1	0	The auxiliary function `M01` (conditional stop) is always output to the PLC. A quick acknowledgement is ineffective, because `M01` is permanently assigned to the first auxiliary function group and is therefore always output at the end of the block.
	1	The auxiliary function `M01` (conditional stop) is only output to the PLC, if the function: "Programmed stop" is activated via the HMI user interface. In the case of a quick acknowledgement, the `M1` is output to the PLC during the motion. While the function is not active, this does not interrupt continuous-path mode.

Part program blocks without traversing motion

In a part program block without a traversing motion, all auxiliary functions are output in a block immediately, irrespective of their parameterized output behavior.

Spindle-specific auxiliary function output only as information for the PLC user program

In certain controller situations, e.g. at the end of a block search, the collected spindle-specific auxiliary functions (e.g. `M3`, `M4`, `M5`, `M19`, `M40`...`M45`, `M70`) is output to the NC/PLC interface only for information purposes for the PLC user program. The controller generates a part program block (action block) in which the collected auxiliary functions are entered with a **negative** address extension. The corresponding system functions are then **not** executed.

Example: M(-2) = 41 request gear stage change for the 2nd spindle

9.17 Examples

9.17.1 Extension of predefined auxiliary functions

Task

Parameter assignment of auxiliary functions M3, M4, and M5 for the second spindle of the channel

Parameter assignment: M3

Requirements:

- Machine data index: 0 (first user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M3 (spindle right)

- Address extension: 2 as appropriate for the 2nd spindle of the channel
- Output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Output prior to motion

Parameter assignment:

```
MD22000 $MC_AUXFU_ASSIGN_GROUP[ 0 ]          = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 0 ]          = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 0 ]     = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 0 ]         = 3
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 0 ]          = 'H21'
```

Parameter assignment: M4

Requirements:

- Machine data index: 1 (second user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M4 (spindle left)
- Address extension: 2 as appropriate for the 2nd spindle of the channel
- Output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Spindle response following acknowledgment
  - Output during motion

Parameter assignment:

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 1 ]         = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 1 ]          = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 1 ]     = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 1 ]         = 4
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 1 ]          = 'H51'
```

Parameter assignment: M5

Requirements:

- Machine data index: 2 (third user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M5 (spindle stop)

- Address extension: 2 as appropriate for the 2nd spindle of the channel
- Output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Spindle response following acknowledgment
  - Output at block end

Parameter assignment:

MD22000 $MC_AUXFU_ASSIGN_GROUP [2]	= 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [2]	= "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [2]	= 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [2]	= 5
MD22035 $MC_AUXFU_ASSIGN_SPEC [2]	= 'H91'

9.17.2 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 as standard to the 2nd auxiliary function group.
- All S and S1 values of spindle 1 (master spindle) are assigned as standard to the 3rd auxiliary function group.

- The gear stage last programmed is to be output after block search. The following auxiliary functions are assigned for this reason to the 9th auxiliary function group:
  - M40, M41, M42, M43, M44, M45
  - M1=40, M1=41, M1=42, M1=43, M1=44, M1=45
- The auxiliary functions M3, M4, M5, M70 and M1=3, M1=4, M1=5, M1=70 (2nd auxiliary function group) and S and S1 values (3rd auxiliary function group) should possess the following output behavior:
  - Output duration one OB40 cycle (quick acknowledgment)
  - Output prior to motion
- The auxiliary functions for gear changeover M40 to M45 and M1=40 to M1=45 (9th auxiliary function group) should have the following output behavior:
  - Output duration of one OB1 cycle (normal acknowledgment)
  - Output prior to motion

Spindle 2

- Only one M function for directional reversal may be programmed in one block. The direction of rotation last programmed is to be output after block search. The following auxiliary functions are assigned for this reason to the 10th auxiliary function group:
  - M2=3, M2=4, M2=5, M2=70
- All S2 values are assigned to the 11th auxiliary function group.
- The auxiliary functions M2=3, M2=4, M2=5, M2=70 (10th auxiliary function group) and S2 values (11th auxiliary function group) should have the following output behavior:
  - Output duration one OB40 cycle (quick acknowledgment)
  - Output prior to motion

Cooling water

- It is not permissible to switch the cooling water on and off in one part program block. After a block search, the cooling water will be switched on or off. The following auxiliary functions are assigned for this reason, e.g. to the 12th or 13th auxiliary function group:
  – 12th auxiliary function group: M50, M51
  – 13th auxiliary function group: M52, M53
- The auxiliary functions M50, M51 (12th auxiliary function group) and M52, M53 (13th auxiliary function group) should have the following output behavior:
  – Output duration of one OB1 cycle (normal acknowledgment)
  – Output prior to motion

Parameterization of the machine data

The machine data is parameterized by appropriate programming within a part program.

Program code	**Comment**
`$MN_AUXFU_MAXNUM_GROUP_ASSIGN=21`	`; Number of user-defined auxiliary functions per channel`
`$MN_AUXFU_GROUP_SPEC[1]='H22'`	`; Output behavior of the 2nd auxiliary function group`
`$MN_AUXFU_GROUP_SPEC[2]='H22'`	`; Output behavior of the 3rd auxiliary function group`
`$MN_AUXFU_GROUP_SPEC[8]='H21'`	`; Output behavior of the 9th auxiliary function group`
`$MC_AUXFU_ASSIGN_TYPE[0]="M"`	`; Description of the 1st auxiliary function: M40`
`$MC_AUXFU_ASSIGN_EXTENSION[0]=0`	
`$MC_AUXFU_ASSIGN_VALUE[0]=40`	
`$MC_AUXFU_ASSIGN_GROUP[0]=9`	
	`; ... (and analogously for the 2nd - 5th auxiliary functions)`
`$MC_AUXFU_ASSIGN_TYPE[5]="M"`	`; Description of the 6th auxiliary function: M45`
`$MC_AUXFU_ASSIGN_EXTENSION[5]=0`	
`$MC_AUXFU_ASSIGN_VALUE[5]=45`	
`$MC_AUXFU_ASSIGN_GROUP[5]=9`	
`$MC_AUXFU_ASSIGN_TYPE[6]="M"`	`; Description of the 7th auxiliary function: M1=40`
`$MC_AUXFU_ASSIGN_EXTENSION[6]=1`	
`$MC_AUXFU_ASSIGN_VALUE[6]=40`	
`$MC_AUXFU_ASSIGN_GROUP[6]=9`	
	`; . . . (and analogously for the 8th - 11th auxiliary functions)`

Program code	Comment
`$MC_AUXFU_ASSIGN_TYPE[11]="M"`	`; Description of the 12th auxiliary function: M1=45`
`$MC_AUXFU_ASSIGN_EXTENSION[11]=1`	
`$MC_AUXFU_ASSIGN_VALUE[11]=45`	
`$MC_AUXFU_ASSIGN_GROUP[11]=9`	
`$MN_AUXFU_GROUP_SPEC[9] = 'H22'`	`; Output behavior of the 10th auxiliary function group`
`$MC_AUXFU_ASSIGN_TYPE[12]="M"`	`; Description of the 13th auxiliary function: 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 the 14th auxiliary function: M2=4`
`$MC_AUXFU_ASSIGN_EXTENSION[13]=2`	
`$MC_AUXFU_ASSIGN_VALUE[13]=4`	
`$MC_AUXFU_ASSIGN_GROUP[13]=10`	
`$MC_AUXFU_ASSIGN_TYPE[14]="M"`	`; Description of the 15th auxiliary function: 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 the 16th auxiliary function: 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 the 11th auxiliary function group`
`$MC_AUXFU_ASSIGN_TYPE[16]="S"`	`; Description of the 17th auxiliary function: S2=<all values>`
`$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 the 12th auxiliary function group`

Program code	Comment
`$MC_AUXFU_ASSIGN_TYPE[17]="M"`	`; Description of the 18th auxiliary function: M50`
`$MC_AUXFU_ASSIGN_EXTENSION[17]=0`	
`$MC_AUXFU_ASSIGN_VALUE[17]=50`	
`$MC_AUXFU_ASSIGN_GROUP[17]=12`	
`$MC_AUXFU_ASSIGN_TYPE[18]="M"`	`; Description of the 19th auxiliary function: M51`
`$MC_AUXFU_ASSIGN_EXTENSION[18]=0`	
`$MC_AUXFU_ASSIGN_VALUE[18]=51`	
`$MC_AUXFU_ASSIGN_GROUP[18]=12`	
`$MN_AUXFU_GROUP_SPEC[12]='H21'`	`; Specification of the 13th auxiliary function group`
`$MC_AUXFU_ASSIGN_TYPE[19]="M"`	`; Description of the 20th auxiliary function: 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 the 21st auxiliary function: M53`
`$MC_AUXFU_ASSIGN_EXTENSION[20]=0`	
`$MC_AUXFU_ASSIGN_VALUE[20]=53`	
`$MC_AUXFU_ASSIGN_GROUP[20]=13`	

9.18 Data lists

9.18.1 Machine data

9.18.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10713	M_NO_FCT_STOPRE	M function with preprocessing stop
10714	M_NO_FCT_EOP	M function for spindle active after NC RESET
10715	M_NO_FCT_CYCLE	M function to be replaced by subroutine
11100	AUXFU_MAXNUM_GROUP_ASSIGN	Maximum number of user-defined auxiliary functions per channel

Number	Identifier: $MN_	Description
11110	AUXFU_GROUP_SPEC	Group-specific output behavior
11450	SEARCH_RUN_MODE	Behavior after a block search

9.18.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control initial setting after part program start.
20112	START_MODE_MASK	Definition of control initial setting after powerup and on RESET or at end of part program
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20800	SPF_END_TO_VDI	Subroutine end / Stop at PLC
22000	AUXFU_ASSIGN_GROUP	Group assignment of user-defined auxiliary functions
22010	AUXFU_ASSIGN_TYPE	Type of user-defined auxiliary functions
22020	AUXFU_ASSIGN_EXTENSION	Address extension for user-defined auxiliary functions
22030	AUXFU_ASSIGN_VALUE	Value of user-defined auxiliary functions
22035	AUXFU_ASSIGN_SPEC	Output behavior of user-defined auxiliary functions
22040	AUXFU_PREDEF_GROUP	Group assignment of predefined auxiliary functions
22050	AUXFU_PREDEF_TYPE	Type of predefined auxiliary functions
22060	AUXFU_PREDEF_EXTENSION	Address extension for predefined auxiliary functions
22070	AUXFU_PREDEF_VALUE	Value of predefined auxiliary functions
22080	AUXFU_PREDEF_SPEC	Output behavior of predefined auxiliary functions
22100	AUXFU_QUICK_BLOCKCHANGE	Block change delay with quick auxiliary functions
22110	AUXFU_H_TYPE_INT	Type of H auxiliary functions
22200	AUXFU_M_SYNC_TYPE	M functions output time
22210	AUXFU_S_SYNC_TYPE	S functions output time
22220	AUXFU_T_SYNC_TYPE	T functions output time
22230	AUXFU_H_SYNC_TYPE	H functions output time
22240	AUXFU_F_SYNC_TYPE	F functions output time
22250	AUXFU_D_SYNC_TYPE	D functions output time
22252	AUXFU_DL_SYNC_TYPE	DL functions output time
22254	AUXFU_ASSOC_M0_VALUE	Additional M function for program stop
22256	AUXFU_ASSOC_M1_VALUE	Additional M function for conditional stop
22530	TOCARR_CHANGE_M_CODE	M code for change of tool holder
22532	GEOAX_CHANGE_M_CODE	M code for replacement of geometry axes
22534	TRAFO_CHANGE_M_CODE	M code for change of tool holder
22560	TOOL_CHANGE_M_CODE	Auxiliary function for tool change

9.18.2 Signals

9.18.2.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate associated M01	DB21, ...DBX30.5	DB320x.DBX14.5

9.18.2.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
M function 1 - 5 change	DB21, ...DBX58.0-4	DB250x.DBX4.0-4
M function 1 - 5 not decoded	DB21, ...DBX59.0-4	-
S function 1 - 3 change	DB21, ...DBX60.0-2	DB250x.DBX6.0
S function 1 - 3 quick	DB21, ...DBX60.4-6	-
T function 1 - 3 change	DB21, ...DBX61.0-2	-
T function 1 - 3 quick	DB21, ...DBX61.4-6	-
D function 1 - 3 change	DB21, ...DBX62.0-2	DB250x.DBX10.0
D function 1 - 3 quick	DB21, ...DBX62.4-6	-
DL function change	DB21, ...DBX63.0	-
DL function quick	DB21, ...DBX63.4	-
H function 1 - 3 change	DB21, ...DBX64.0-2	DB250x.DBX12.0-2
H function 1 - 3 quick	DB21, ...DBX64.4-6	-
F function 1 - 6 change	DB21, ...DBX65.0-5	-
M function 1 - 5 quick	DB21, ...DBX66.0-4	-
F function 1 - 6 quick	DB21, ...DBX67.0-5	-
Extended address M function 1 (16 bit int)	DB21, ...DBB68-69	DB250x.DBB3004
M function 1 (DInt)	DB21, ...DBB70-73	DB250x.DBD3000
Extended address M function 2 (16 bit int)	DB21, ...DBB74-75	DB250x.DBB3012
M function 2 (DInt)	DB21, ...DBB76-79	DB250x.DBD3008
Extended address M function 3 (16 bit int)	DB21, ...DBB80-81	DB250x.DBB3020
M function 3 (DInt)	DB21, ...DBB82-85	DB250x.DBD3016
Extended address M function 4 (16 bit int)	DB21, ...DBB86-87	DB250x.DBB3028
M function 4 (DInt)	DB21, ...DBB88-91	DB250x.DBD3024
Extended address M function 5 (16 bit int)	DB21, ...DBB92-93	DB250x.DBB3036
M function 5 (DInt)	DB21, ...DBB94-97	DB250x.DBD3032
Extended address S function 1 (16 bit int)	DB21, ...DBB98-99	DB250x.DBB4004
S function 1 (real)	DB21, ...DBB100-103	DB250x.DBD4000
Extended address S function 2 (16 bit int)	DB21, ...DBB104-105	DB250x.DBB4012
S function 2 (real)	DB21, ...DBB106-109	DB250x.DBD4008
Extended address S function 3 (16 bit int)	DB21, ...DBB110-111	DB250x.DBB4020
S function 3 (real)	DB21, ...DBB112-115	DB250x.DBD4016

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Extended address T function 1 (16 bit int)	DB21, ...DBB116-117	-
T function 1 (integer)	DB21, ...DBB118-119	DB250x.DBD2000
Extended address T function 2 (16 bit int)	DB21, ...DBB120-121	-
T function 2 (integer)	DB21, ...DBB122-123	-
Extended address T function 3 (16 bit int)	DB21, ...DBB124-125	-
T function 3 (integer)	DB21, ...DBB126-127	-
Extended address D function 1 (8 bit int)	DB21, ...DBB128	-
D function 1 (8 bit int)	DB21, ...DBB129	DB250x.DBD5000
Extended address D function 2 (8 bit int)	DB21, ...DBB130	-
D function 2 (8 bit int)	DB21, ...DBB131	-
Extended address D function 3 (8 bit int)	DB21, ...DBB132	-
D function 3 (8 bit int)	DB21, ...DBB133	-
Extended address DL function (8 bit int)	DB21, ...DBB134	-
DL function (real)	DB21, ...DBB136	-
Extended address H function 1 (16 bit int)	DB21, ...DBB140-141	DB250x.DBB6004
H function 1 (real or DInt)	DB21, ...DBB142-145	DB250x.DBD6000
Extended address H function 2 (16 bit int)	DB21, ...DBB146-147	DB250x.DBB6012
H function 2 (REAL or DInt)	DB21, ...DBB148-151	DB250x.DBD6008
Extended address H function 3 (16 bit int)	DB21, ...DBB152-153	DB250x.DBB6020
H function 3 (real or DInt)	DB21, ...DBB154-157	DB250x.DBD6016
Extended address F function 1 (16 bit int)	DB21, ...DBB158-159	-
F function 1 (real)	DB21, ...DBB160-163	-
Extended address F function 2 (16 bit int)	DB21, ...DBB164-165	-
F function 2 (real)	DB21, ...DBB166-169	-
Extended address F function 3 (16 bit int)	DB21, ...DBB170-171	-
F function 3 (real)	DB21, ...DBB172-175	-
Extended address F function 4 (16 bit int)	DB21, ...DBB176-177	-
F function 4 (real)	DB21, ...DBB178-181	-
Extended address F function 5 (16 bit int)	DB21, ...DBB182-183	-
F function 5 (real)	DB21, ...DBB184-187	-
Extended address F function 6 (16 bit int)	DB21, ...DBB188-189	-
F function 6 (real)	DB21, ...DBB190-193	-
Dynamic M function: M00 - M07	DB21, ...DBB194	DB250x.DBB1000
Dynamic M function: M08 - M15	DB21, ...DBB195	DB250x.DBB1001
Dynamic M function: M16 - M23	DB21, ...DBB196	DB250x.DBB1002
Dynamic M function: M24 - M31	DB21, ...DBB197	DB250x.DBB1003
Dynamic M function: M32 - M39	DB21, ...DBB198	DB250x.DBB1004
Dynamic M function: M40 - M47	DB21, ...DBB199	DB250x.DBB1005
Dynamic M function: M48 - M55	DB21, ...DBB200	DB250x.DBB1006
Dynamic M function: M56 - M63	DB21, ...DBB201	DB250x.DBB1007
Dynamic M function: M64 - M71	DB21, ...DBB202	DB250x.DBB1008

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Dynamic M function: M72 - M79	DB21, ...DBB203	DB250x.DBB1009
Dynamic M function: M80 - M87	DB21, ...DBB204	DB250x.DBB1010
Dynamic M function: M88 - M95	DB21, ...DBB205	DB250x.DBB1011
Dynamic M function: M96 - M99	DB21, ...DBX206.0-3	DB250x.DBB1012.0-3
Associated M00/M01 active	DB21, ...DBX318.5	DB330x.DBX4002.5

9.18.2.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
F function for positioning axis (real)	DB31, ...DBB78-81	-

9.18.2.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
M function for spindle (Int)	DB21, ...DBB86-87	DB370x.DBD0000
S function for spindle (real)	DB21, ...DBB88-91	DB370x.DBD0004

K1: Mode group, channel, program operation, reset response 10

10.1 Product brief

Channel

An NC channel represents the smallest unit for manual traversing of axes and automatic processing of part programs. At any one time, a channel will always be in a particular mode, e.g. AUTOMATIC, MDI, or JOG. A channel can be regarded as an independent NC.

Mode group

A channel always belongs to a mode group. A mode group can also consist of several channels.

A mode group can be identified by the fact that all channels of the mode group are always in the same mode at a particular time, e.g. AUTOMATIC, MDI, or JOG. This is ensured through the NC internal mode logic.

A mode group can be regarded as an independent multi-channel NC.

Channel gaps

When channels are configured, placeholder channels can be provided in order to create as uniform a configuration as possible over machines in a series. Only the channels that are actually used are then activated.

Program test

The following options are available for testing or moving in position a new part program.

- Program execution without setpoint outputs
- Program execution in singleblock mode
- Program execution with dry run feedrate
- Skip part program blocks
- Block search with or without calculation.

Block search

The block search function enables the following program simulations for locating specific program points:

- Type 1 without calculation at contour
- Type 2 with calculation at contour

- Type 4 with calculation at block end point
- Type 5 automatic start of the selected program point with calculation of all required data from history
- Automatic start of an ASUP after a block search
- Cascaded block search
- Cross-channel block search in "Program test" mode

Program operation

The execution of part programs or part program blocks in AUTOMATIC or MDI modes is referred to as program operation. During execution, the program sequence can be controlled by PLC interface signals and commands.

For each channel, basic settings or channel-specific machine data can be specified. These basic 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 subprograms (ASUPs), interrupt routines

Interrupt inputs allow the NC to interrupt the the current part program execution so that it can react to more urgent events in interrupt routines or ASUPs.

Single block

With the single-block function, the user can execute a part program block-by-block.

There are 3 types of setting for the single-block function:

- SLB1: = IPO single block
- SLB2: = decode single block
- SLB3: = stop in cycle

Basic block display

A second basic block display can be used with the existing block display to display all blocks that produce an action on the machine.

The actually approached end positions are shown as an absolute position. The position values refer either to the workpiece coordinate system (WCS) or the settable zero system (SZS).

Program execution from external source

When complex workpieces are machined, the NC may not have enough memory for the programs. Using the function "Execution from external source" or "EES (execution from external storage) enables part programs to be called and executed from an external memory (e.g. from the hard disk).

Behavior after POWER ON, Reset, ...

The control-system response after:

- Power up (POWER ON)
- Reset/part program end
- Part program start

can be modified for functions, such as G commands, tool length compensation, transformation, coupled axis groupings, tangential follow-up, programmable synchronous spindle for certain system settings through machine data.

Subprogram call with M, T and D functions

For certain applications, it may be advantageous to replace M, T or D functions as well as a few NC language commands`SPOS`, `SPOSA`, by a subprogram call. This can be used, for example, to call the tool change routine.

Relevant machine data can be used to define and control subprograms having M, T or D functions. For example, for a gear stage change.

Program runtime/part counter

Information on the program runtime and the part count is provided to assist the machine tool operator.

The functions defined for this purpose are **not identical to the functions of tool management** and are intended primarily for systems without tool management.

10.2 Mode group

Mode group

In a mode group, multiple channels if an NC are grouped together to form one processing unit. In principle, it is an independent "NC" within an NC.

A mode group is essentially characterized by the fact that all channels assigned to it are any instant are always in the same mode (AUTOMATIC, JOG, MDI).

Note

This description continues on the assumption that there is one mode group and one channel.

Functions that need several channels, e.g. "Axis interchange" function, are described in:
References:
Function Manual, Extended Functions; "K5: Cross-channel program coordination" and "K10: Cross-channel axis interchange"

Parameterization

A channel is assigned to a mode group in the machine data:

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[<channel index>] = mode group number

All channels with the same mode group number form one mode group.

Note

A NC always comprises one mode group with one channel by default. It is not possible to parameterize an NC without a channel.

Mode-group-specific NC/PLC interface signals

The mode-group-specific NC/PLC interface signals are in data block DB11.

The mode-group-specific NC/PLC interface essentially comprises the following interface signals:

- Request signals PLC → NC
  - Mode group reset
  - Mode group stop axes plus spindles
  - Mode group stop
  - Mode changeover inhibit
  - Mode: JOG, MDI, AUTOM.
  - Single block: Type A, Type B
  - Machine function REF, REPOS, TEACH IN, INC x
- NC → PLC status signals
  - Mode strobe: JOG, MDI, AUTOMATIC
  - Machine function strobe: REF, REPOS, TEACH IN
  - All channels in the reset state
  - Mode group has been reset
  - Mode group Ready
  - Active mode: JOG, MDI, AUTOMATIC
  - Active machine function: REF, REPOS, TEACH, INC x

Activated and non-activated channels, channel gaps

Activated channel

A channel with a mode group number ≠ 0 is an active channel.

Non-activated channel

A channel with a mode group number 0 is a non-activated channel. It does not take up any memory internally in the control.

Channel gaps

A channel with a mode group number 0 is not only a non-activated channel. It is also a so-called channel gap in the sequence of the channels.

The advantage of channel gaps is that the configuration data can be kept largely identical in a series of similar machines. For the specific commissioning, only those channels are activated that are required for the machine in question. The unoccupied memory can be freely used as additional user memory.

Table 10-1 Example

Machine data	Meaning
MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[0] = 1	Channel 1, mode group 1
MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[1] = 2	Channel 2, mode group 2
MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[2] = 0	Channel 3, not used
MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[3] = 1	Channel 4, mode group 1
MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[4] = 2	Channel 5, mode group 2

See also

K2: Axis Types, Coordinate Systems, Frames (Page 705)

S1: Spindles (Page 1273)

10.2.1 Mode group stop

Function

The following NC/PLC interface signals are used to stop the traversing motions of the axes or of the axes and spindles in all mode group channels and to interrupt part program execution:

DB11 DBX0.5 (mode group stop)

DB11 DBX0.6 (mode group stop, axes plus spindles)

10.2.2 Mode group reset

Function

A mode group reset is requested via a mode group-specific NC/PLC interface signal:

DB11 DBX0.7 = 1 (mode group reset)

Effect

Effect on the channels of mode group:

- Part program preparation (preprocessing) is stopped.
- All axes and spindles are decelerated to zero speed according to their acceleration curves without contour violation.
- Any auxiliary functions not yet output to the PLC are no longer output.
- The preprocessing pointers are set to the interruption point, and the block pointers are set to the beginning of the appropriate part programs.
- All initial settings (e.g. the G commands) are set to the parameterized values.
- All alarms with "Channel reset" criterion are canceled.

If all the channels of the mode group are in reset state, then:

- All alarms with "Mode group reset" cancel criterion are canceled.
- The NC/PLC interface indicates completion of the mode group reset and the mode group's readiness to operate:
  DB11 DBX6.7 (all channels in the reset state)
  DB11 DBX6.3 = 1 (mode group ready)

10.3 Mode types and mode type change

Unique mode

All channels of a mode group are always in the same mode:

- AUTOMATIC
- JOG
- MDI

If individual channels are assigned to different mode groups, a channel switchover activates the corresponding mode group. This allows mode changes to be initiated via a channel switchover.

Modes

The following modes are available:

- **AUTOMATIC**
  Automatic execution of part programs:
  - Part program test
  - All channels of the mode group can be active at the same time.
- **JOG in Automatic**
  JOG in AUTOMATIC is an extension of AUTOMATIC mode intended to simplify use. JOG can be executed without leaving AUTOMATIC mode if boundary conditions so permit.

- **JOG**
  Manual traversing of axes via traversing keys of the machine control panel or via a handwheel connected to the machine control panel:
  - Channel-specific signals and interlocks are taken into account for motions executed by means of an ASUP or via static synchronized actions.
  - Couplings are taken into account.
  - Every channel in the mode group can be active.
- **MDI**
  Manual Data Automatic (the blocks are entered via the user interface)
  - Restricted execution of part programs and part program sections.
  - Part program test
  - A maximum of one channel per mode group can be active (applies only to TEACH IN).
  - Axes can only be traversed manually in subordinate machine functions such as JOG, REPOS or TEACH IN.

Applies to all modes

Cross-mode synchronized actions

Modal synchronized actions can be executed by means of IDS in all modes for the following functions parallel to the channel:

- Command axis functions
- Spindle functions
- Technology cycles

Selection

The user can select the desired mode by means of softkeys on the user interface.

This selection (AUTOMATIC, MDI, or JOG) is forwarded to the PLC on the NC/PLC interface, but is not activated:
DB11 DBX4.0, 0.1, 0.2 (strobe mode)

Activation and priorities

The mode of the mode group is activated via the NC/PLC interface:

DB11 DBX0.0, 0.1, 0.2 (mode)

If several modes are selected at the same time, the following priority applies:

Priority	Mode	Mode group signal (NC → PLC)
1st priority, high	JOG	DB11 DBX0.2
2nd priority, medium	MDI	DB11 DBX0.1
3rd priority, low	AUTOMATIC	DB11 DBX0.0

Display

The current mode of the mode group is displayed via the NC/PLC interface:

DB11 DBX6.0, 0.1, 0.2 (active mode)

Mode group signal (NC → PLC)	Active operating mode
DB11 DBX6.2	JOG
DB11 DBX6.1	MDI
DB11 DBX6.0	AUTOMATIC

Machine functions

Machine functions can be selected within a mode that also apply within the mode group:

- Machine functions within the **JOG** mode
  - REF (reference point approach)
  - REPOS (repositioning)
  - JOG retract (retraction motion in the tool direction)
- Machine functions within the **MDI** mode
  - REF (reference point approach)
  - REPOS (repositioning)
  - TEACHIN (teach-in of axis positions)

NC/PLC interface signals

- DB11 DBX5.0, 0.1, 0.2 (machine function strobe): Requirement
- DB11 DBX1.0, 0.1, 0.2 (machine function): Activation
- DB11 DBX7.0-2 (active machine function): Feedback signal

Channel states

- **Channel reset**
  The machine is in its initial state. This is defined by the machine manufacturer's PLC program, e.g. after POWER ON or at the end of the program.
- **Channel active**
  A program has been started, and the program is being executed or a reference point approach is in progress.
- **Channel interrupted**
  The running program or reference point approach has been interrupted.

Functions within modes

Modes are supplemented through user-specific functions. The individual functions are technology- and machine-independent and can be started and/or executed from the three channel states "Channel reset", "Channel active" or "Channel interrupted".

Supplementary condition for the TEACH IN machine function

TEACH IN is not permissible for leading or following axes of an active axis grouping, e.g. for:

- Gantry-axis grouping or a gantry axis pair
- Coupled-axis grouping of master and slave axis

JOG in Automatic

JOG in AUTOMATIC mode is permitted if the mode group is in "RESET" state and the axis is jog-capable. "RESET" state for the mode group means:

- All channels are in the "RESET" state
- All programs are canceled
- DRF is not active in any channel

An axis is **JOG-capable** if it is not in any of the following states:

- PLC axis as concurrent positioning axis (request of the axis from the PLC)
- Command axis (the axis has been programmed by a synchronized action and the motion has not been completed yet)
- Rotating spindle (spindle rotating despite RESET)
- An asynchronous reciprocating axis

Note: The "jog-capable" property is independent of the "JOG in AUTOMATIC" function.

Activation

The function "JOG in AUTOMATIC" can be activated with the machine data:

MD10735 $MN_JOG_MODE_MASK

- Before POWER ON, the following machine data must be set:
  MD10735 $MN_JOG_MODE_MASK, bit 0 = 1
- The user switches to AUTO (PLC user interface DB11 DBX0.0 = 0→1 edge). "JOG in AUTOMATIC" is then active if the NC previously had channel state "RESET" and program state "Aborted" in all mode group channels. The axis in question must also be "jog-capable". DRF must be deactivated (if not already deactivated).
- RESET is initiated or the running program is finished with `M30`/`M2` in all mode group channels that do not have channel state "Reset" and program state "Aborted".
- The relevant axis is automatically made "JOG-capable" (e.g. axis interchange: PLC → NC).

Note: In most applications, the axes to be traversed are "JOG-capable" and with the switchover to AUTOMATIC, "JOG in AUTOMATIC" is also active.

Characteristics

- The +/– keys cause a JOG motion, and the mode group is switched **internally** to JOG. (i.e. "Internal JOG").
- Moving the handwheels causes a JOG motion, and the mode group is switched **internally** to JOG, unless DRF is active.

- An ongoing JOG motion is not complete until the end position of the increment has been reached (if this has been set) or the motion 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 NC 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 motion, "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 motion is active, the NC is internally in JOG mode, and, thus, a block search request is refused and a Start cannot start the part program. Start starts any remaining increment or has no effect.
- While a mode group axis is being traversed in JOG mode, the mode group remains internally in JOG mode.
  Comment: This phase can begin with the JOG motion of an axis and end with the end of the JOG motion of *another* axis.
- Axis interchange is not possible for an axis with active JOG movement (the axis might change the mode group). The NC blocks any axis interchange attempt.
- The PLC user interface indicates "Automatic" mode:
  - DB11 DBX6.0, 6.1, 6.2 = 1
  - DB11 DBX7.0, 7.1, 7.2 = 0
- In "JOG in AUTOMATIC", the NC/PLC interface displays whether the mode group is in "Mode group RESET".
  - DB11 DBX6.4 (mode group has been reset, mode group 1)
  - DB11 DBX26.4 (mode group has been reset, mode group 2)
  - DB11 DBX46.4 (mode group has been reset, mode group 3)
- In "JOG in AUTOMATIC", the NC/PLC interface displays whether the NC has automatically switched to "Internal JOG".
  - DB11 DBX6.5 (NC internal JOG active, mode group 1)
  - DB11 DBX26.5 (NC internal JOG active, mode group 2)
  - DB11 DBX46.5 (NC internal JOG active, mode group 3)

Supplementary conditions

"JOG in AUTOMATIC" can only switch internally to JOG if the mode group is in the "mode group reset" status, 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 NC switches to "Internal JOG", the selected increment is retained.

If the user attempts to jog the geometry or orientation axes, the NC switches to "Internal JOG" and the movement executed. Several axes can be physically moved in this way; they must all be "JOG-capable".

Following the JOG motion, the NC 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 NC will then switch to "Internal JOG" even if the axis is not enabled.

See also

R1: Referencing (Page 1223)

10.3.1 Monitoring functions and interlocks of the individual modes

Channel status determines monitoring functions

Monitoring functions in operating modes

Different monitoring functions are active in individual operating modes. These monitoring functions are not related to any particular technology or machine.

In a particular mode only some of the monitoring functions are active depending on the operating status. The channel status determines which monitoring functions are active in which mode and and in which operating state.

Interlocking functions in operating modes

Different interlocks can be active in the different operating modes. These interlocking functions are not related to any particular technology or machine.

Almost all the interlocks can be activated in every mode, depending on the operating status.

10.3.2 Mode change

Introduction

A mode change is requested and activated via the mode group interface (DB11). A mode group will either be in AUTOMATIC, JOG, or MDI mode, i.e. it is not possible for several channels of a mode group to take on different modes at the same time.

What mode transitions are possible and how these are executed can be configured in the PLC program on a machine-specific basis.

Note

The mode is not changed internally until the signal "Channel status active" is no longer pending. For error-free mode change however, all channels must assume a permissible operating mode.

Possible mode changes

The following table shows possible mode changes for a channel:

	AUTOMATIC		JOG				MDI		
				AUTO	MDI		Manual traversing		AUTO
	Reset	Inter-rupted	Reset	Inter-rupted	Inter-rupted	Reset	Inter-rupted	Active	Inter-rupted
AUTOMATIC			X	X		X			
JOG	X	X				X	X		X
MDI	X	X	X		X				

Possible mode changes are indicated by an "X".

Special cases

- **Error during mode change**
  If a mode change request is rejected by the system, the error message "Operating mode cannot be changed until after NC Stop" is output. This error message can be cleared without changing the channel status.
- **Mode change disable**
  A mode change can be prevented by means of interface signal:
  DB11, DBX0.4 (mode change disable).

  This suppresses the mode change request.
  The user must configure a message to the operator indicating that mode change is disabled. No signal is issued for this by the system.
- **Mode change from MDI to JOG**
  If all channels of the mode group are in Reset state after a mode change from MDI 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.

10.4 Channel

In a channel of the NC, the part programs defined by the user are executed.

The following properties characterize a channel:

- A channel is always assigned to a mode group (Page 481) (BAG).
- A channel can only execute one part program at any point in time.
- Machine, geometry, and special axes and spindles are assigned to a channel. Only these can be traversed via the part program executed in the channel.
- A channel consists of the internal units
  - Preprocessing (program decoding and block preparation)
  - Main run (path and axis interpolation).

- A channel has an interface with the PLC. Via this NC/PLC interface, the PLC user program can read various channel-specific status data and write requests to the channel.
- Channel-specific tool offsets (Page 1451) are active in a channel.
- Channel-specific coordinate systems (Page 724) are active in a channel.
- Each channel has a unique channel name by default. The defined channel name can be changed in machine data:
  MD20000 $MC_CHAN_NAME = "<channel name>"

Multiple channels can be combined to form a mode group. The channels of a mode group are always in the same mode (AUTOMATIC, JOG, MDI).

Channel configuration

Channels can be filled with their own channel name via the following machine data:

MD20000 $MC_CHAN_NAME (channel name)

The various axes are then assigned to the available channels via machine data. There can be only one setpoint-issuing channel at a time for an axis/spindle. The axis/spindle actual value can be read by several channels at the same time. The axis/spindle must be registered with the relevant channel.

The following channel-specific settings can also be made using machine data:

- Position of deletions or the basic program settings of G groups via the machine data:
  MD20150 $MC_GCODE_RESET_VALUES (reset setting of the G groups)
- Auxiliary function groups regarding the combination and the output time.
- Transformation conditions between machine axes and geometry axes.
- Other settings for the execution of a part program.

Change in the channel assignment

An online change in the channel configuration cannot be programmed in a part program or PLC user program. Changes in the configuration must be made via the machine data. The changes become effective only after a new POWER ON.

Container axes and link axes

An axis container combines a group of axes in a container. These axes are referred to as container axes. This involves assigning a pointer to a container slot (ring buffer location within the relevant container) to a channel axis. One of the axes in the container is located temporarily in this slot.

Each machine axis in the axis container must be assigned at all times to exactly **one** channel axis.

Link axes can be assigned permanently to one channel or dynamically (by means of an axis container switch) to several channels of the local NCU or the other NCU. A link axis is a non-local axis from the perspective of one of the channels belonging to the NCU to which the axis is not physically connected.

The assignment between the link axes and a channel is implemented as follows:

- For permanent assignment using machine data:
  Allow the direct logic machine axis image to show link axes.
- For dynamic assignment:
  Allow the axis container slot machine data to show link axes.

Further information on link axes and container axes can be found in:
References:
Function Manual, Extended Functions; Several Operator Panel Fronts and Multiple NCUs, Distributed Systems (B3)

Interface signals

The signals of the 1st channel are located in the NC/PLC interface in DB21, the signals from channel 2 are located in DB22. The channel or channels can be monitored and controlled from the PLC or NC.

Channel-specific technology specification

The technology used can be specified for each channel:

MD27800 $MC_TECHNOLOGY_MODE

In the delivery state, machine data is active for milling as standard.

Spindle functions using a PLC

In addition to function block FC18, spindle functions can also be started and stopped via the axial NC/PLC interface signals in parallel to part programs that are running.

Requirements:

- Channel status: "Interrupted" or "RESET"
- Program status: "Interrupted" or "canceled"

The following functions can be controlled from the PLC via interface signals:

- Stop (corresponds to `M5`)
- Start with clockwise direction of rotation (corresponds to `M3`)
- Start with counter-clockwise direction of rotation (corresponds to `M4`)
- Select gear stage
- Positioning (corresponds to `M19`)

For several channels, the spindle started by the PLC is active in the channel to which it is assigned at the start.

For more information on the special spindle interface, see Section "S1: Spindles (Page 1273)."

PLC-controlled single-axis operations

An axis can also be controlled from the PLC instead of from a channel. For this purpose, the PLC requests the axis from the NC via the NC/PLC interface:

DB31, ... DBX28.7 = 1 (PLC controls axis)

The following functions can be controlled from the PLC:

- Cancel axis/spindle sequence (equivalent to delete distance-to-go)
- Stop or interrupt axis/spindle
- Resume axis/spindle operation (continue the motion sequence)
- Reset axis/spindle to the initial state

For more information on the channel-specific signal exchange (PLC → NC), see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)."

The exact functionality of independent single-axis operations is described in:

References:
Function Manual, Extended Functions; Positioning Axes (P2)

10.4.1 Start inhibit, global and channel-specific

Function

In ShopMill/ShopTurn, a program can only be started by default in the machine area. Starting a program in any of the other areas (e.g. tools) is prevented by a global start lock (PI service).

The automated start lock sequences can be deactivated using a NC/PLC interface signal.

NC/PLC interface signals

Deactivate start lock

An active start lock is deactivated with the interface signal:

DB21, ... DBX7.5 (deactivate start lock)

Machine data

The machine data is used to set an alarm if a start is requested with the start lock set:

MD11411 $MN_ENABLE_ALARM_MASK, bit 6

OPI variables

Variable	Description
startLockState	Status of the global start lock
chanStartLockState	Status of the channel-specific start lock

Variable	Description
startRejectCounter	Start counter for a global start lock
startLockCounter	Start counter for a channel-specific start lock

PI services

Setting the start lock (_N_STRTLK)

The start lock can be set either globally or on a channel-for-channel basis using the PI service _N_STRTLK.

- Global start lock
  If the global start lock is set, then the start of an NC program is locked in all channels. An already running NC program is not influenced. Only the next start of an NC program is locked.
- Channel-specific start lock
  As a result of the channel-specific start lock, the start of an NC program is only locked in the AUTOMATIC mode and in the "Reset" channel state.

Reset of the global or channel-specific start lock (_N_STRTUL)

With the PI service _N_STRTUL, the start lock can be reset, either globally or for specific channels.

10.5 Program test

Several control functions are available for testing a new part program. These functions are provided to reduce danger at the machine and time required for the test phase. Several program functions can be activated at the same time to achieve a better result.

Test options

The following test options are described below:

- Program execution without setpoint outputs
- Program execution in singleblock mode
- Program execution with dry run feedrate
- Skip part program blocks
- Block search with or without calculation.

10.5.1 Program execution without setpoint outputs

Function

In the "Program test" status, a part program is executed without the output of axis or spindle setpoints.

The user can use this to check the programmed axis positions and auxiliary function outputs of a part program. This program simulation can also be used as an extended syntax check.

Selection

This function is selected via the operator interface in the "Program control" menu.

The selection sets the following interface signal:

DB21, ... DBX25.7 (program test selected)

This does not activate the function.

Activation

The function is activated via interface signal:

DB21, ... DBX1.7 (activate program test)

Display

The corresponding field on the operator interface is reversed and the interface signal in the PLC as a checkback of the active program test:

DB21, ... DBX33.7 (program test active)

Program start and program run

When the program test function is active, the part program can be started and executed (incl. auxiliary function outputs, wait times, G command outputs, etc.) via the interface signal:

DB21, ... DBX7.1 (NC-Start)

The safety functions such as software limit switches, working area limits continue to be valid.

The only difference compared to normal program operation is that an internal **axis disable** is set for all axes (including spindles). The machine axes do not move, the actual values are generated internally from the setpoints that are not output. The programmed velocities remain unchanged. This means that the position and velocity information on the operator interface is exactly the same as that output during normal part program execution. The position control is not interrupted, so the axes do not have to be referenced when the function is switched off.

Note

The signals for exact stop DB31, ... DBX60.6/60.7 (exact stop coarse/fine) mirror the actual status on the machine.

They are only canceled during program testing if the axis is pushed out of its set position (the set position remains constant during program testing).

With signal DB21, ... DBX33.7 (program test active) both the PLC program and the part program can use variable $P_ISTEST to decide how to react or branch in response to these signals during testing.

Note

Dry run feedrate

"Program execution without axis motion" can also be activated with the function "Dry run feedrate". With this function, part program sections with a small programmed feedrate can be processed in a shorter time.

Note

Tool management

Because of the axis disable, the assignment of a tool magazine is not changed during program testing. A PLC application must be used to ensure that the integrity of the data in the tool management system and the magazine is not corrupted. The toolbox diskettes contain an example of the basic PLC program.

10.5.2 Program execution in single-block mode

Function

In case of "Program execution in single-block mode" the part program execution stops after every program block. If tool cutter radius compensation or a tool nose radius correction is selected, processing stops after every intermediate block inserted by the controller.

The program status switches to "Program status stopped".

The channel status remains active.

The next part program block is processed on NC Start.

Application

The user can execute a part program block-by-block to check the individual machining steps. Once the user decides that an executed part program block is functioning correctly, the next block can be called.

Single-block types

The following single block types are differentiated:

- Decoding single block
  With this type of single block, all blocks of the part program (even the pure computation blocks without traversing motions) are processed sequentially by "NC Start".
- Action single block (initial setting)
  With this type of single block, the blocks that initiate actions (traversing motions, auxiliary function outputs, etc.) are processed individually.
  Blocks that were generated additionally during decoding (e.g. for cutter radius compensation at acute angles) are also processed individually in single-block mode.
  Processing is however not stopped at calculation blocks as these do not trigger actions.

The single-block types are determined via the user interface in the menu "Program controls".

⚠ CAUTION

Function feature for single-block type series

In a series of `G33`/`G34`/`G35` blocks, a single block is only operative if "dry run feed" is selected.

Calculation blocks are not processed in single-step mode (only if single decoding block is active).

SBL2 is also ineffective with `G33`/`G34`/`G35`.

Selection

It is possible to select the single-block mode:

- Via the machine control panel (key "Single Block")
- Via the user interface
  For an exact procedure, see:
  References:
  Operating Manual of the installed HMI application

Activation

The function is activated through the PLC basic program via the interface signal:

DB21, ... DBX0.4 (activate single block)

Display

Active single-block mode is indicated by a reversal in the relevant field in the status line on the user interface.

Because of the single-block mode, as soon as the part program processing has processed a part program block, the following interface signal is set:

DB21, ... DBX35.3 (program status interrupted)

Processing without single-block stop

Despite the selected single-block mode, a processing without the single-block stop can be set for specific program runs, e.g. for:

- Internal ASUBs
- User ASUBs
- Intermediate blocks
- Block search group blocks (action blocks)
- Init blocks
- Subprograms with `DISPLOF`
- Non-reorganizable blocks
- Non-repositionable blocks
- Reposition block without travel information
- Tool approach block.

The setting is made via the following machine data:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop)

References:
List Manual, Detailed Description of the Machine Data

10.5.3 Program execution with dry run feedrate

Function

The "Program processing with drive run feedrate" function is used to test the coordination of the NC program, selected in the channel, with the PLC user program, external signals and/or other channels of the NC, for example. The NC program is executed precisely corresponding to the programmed commands, functions and traversing velocity. To reduce program processing time during the test, traversing motion can be executed faster by activating the drive run feedrate. For example, for G01, G02, G03, G33, G34, G35, G95 the test feedrate applies instead of the programmed feedrate.

The dry run feedrate is selected at the user interface in the AUTOMATIC and MDA modes, and can be activated when automatic operation is interrupted or at the end of the block. The activation is realized from the PLC user program via the NC/PLC interface.

For further information on influencing the feedrate, see Chapter "V1: Feedrates (Page 1389)".

NOTICE

Destroyed tool or workpiece as a result of excessively high cutting velocities

When a workpiece is machined with the "Program processing with dry run feedrate" function active, it is possible that cutting velocities are obtained that lie outside the permissible range. These high velocities can destroy the tool and/or workpiece.

Parameterization

Dry run feedrate

The dry run feedrate effective in the channel is set using:

SD42100 $SC_DRY_RUN_FEED = <dry run feedrate>

Note

Revolutional feedrate

The dry run feedrate also replaces the programmed revolutional feedrate in program blocks with G95.

Dry run settings

The selection criterion for the feedrate effective that is effective for "Program processing with dry run feed rate" is set using:

SD42101 $SC_DRY_RUN_FEED_MODE = <value>

<Value>	Meaning
0	The maximum from SD42100 and the programmed feedrate are effective as dry run feedrate.
1	The minimum from SD42100 and the programmed feedrate are effective as dry run feedrate.
2	SD42100 is effective as dry run feedrate.
10	As for a value of 0, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). For these functions, the programmed feedrate applies.
11	As for a value of 1, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). For these functions, the programmed feedrate applies.
12	As for a value of 2, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). For these functions, the programmed feedrate applies.

Activation

Selection

The function is selected at the user interface via the operating area "Machine" > Operating mode "AUTOMATIC" or "MDA" > "Program control " > "Dry run feedrate".

The following NC/PLC interface signal is set to request that the PLC user program activates the function:

DB21, ... DBX24.6 = 1 (dry run feedrate selected)

Activation

To request that the NC activates the function, the PLC user program must set the following NC/PLC interface signal:

DB21, ... DBX0.6 = 1 (activate dry run feedrate)

Feedback signal

DRY is displayed on the user interface in the status display of the "Machine" operating area as feedback signal for the machine operator that the function is active in the NC.

References

Detailed information on using the "Program processing with dry run feedrate" function is available in:

- Operating Manual, Turning
- Operating Manual, Milling

Bullet points: Program control, dry run feedrate, DRY

10.5.4 Skip part-program blocks

Function

When testing or breaking in new programs, it is useful to be able to disable or skip certain part program blocks during program execution. For this, the respective records must be marked with a slash.

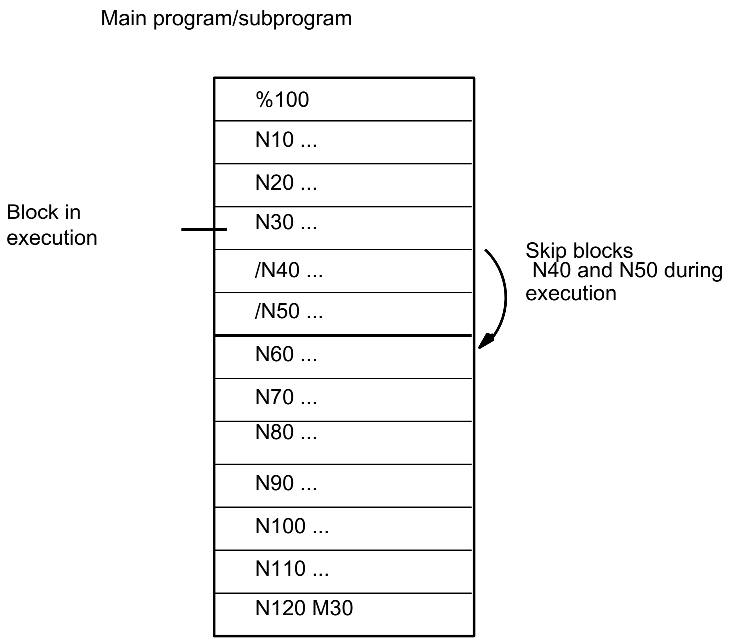


Figure 10-1 Skipping part program blocks

Selection

This function is selected via the operator interface in the "Program control" menu.

The selection sets the following interface signal:

DB21, ... DBX26.0 (skip block selected)

This does not activate the function.

Activation

The function is activated via the interface signal:

DB21, ... DBX2.0 (activate skip block)

Note

The "Skip part programs" function remains active during block searches.

Display

Activated "Skip block" function is indicated by a reversal of the relevant field on the operator interface.

10.6 Workpiece simulation

Function

The actual part program is completely calculated in the tool simulation and the result is graphically displayed in the user interface. The result of programming is verified without traversing the machine axes. Incorrectly programmed machining steps are detected at an early stage and incorrect machining on the workpiece prevented.

Simulation NC

The simulation uses its own NC instance (simulation NC). Therefore, before a simulation is started, the real NC must be aligned to the simulation NC. With this alignment, all active machine data is read out of the NC and read into the simulation NC. The NC and cycle machine data is included in the active machine data.

Compile cycles in simulation (only 840D sl)

Up to SW 4.4, no compile cycles are supported, as of SW 4.4 and higher only selected compile cycles (CC) are supported for the workpiece simulation. The machine data of the supported compile cycles is aligned once after the control has powered-up. An alignment with "simulation start" does not take place!

Note

In part programs, CC-specific language commands and machine data of unsupported CCs **cannot** be used (see also paragraph "CC-commands in the part program").

Special motion of supported CCs (OEM transformations) are - under certain circumstances - incorrectly displayed.

CC-commands in the part program

Language commands in the part program of compile cycles that are not supported (`OMA1` ... `OMA5`, `OEMIPO1/2`, `G810` ... `G829`, own procedures and functions) therefore result in an alarm message and cancellation of the simulation without any individual handling.

Solution: Individually handle the missing CC-specific language elements in the part program ($P_SIM query). Example:

Program code	Comment
N1 G01 X200 F500	
IF (1==$P_SIM)	
N5 X300	; CC not active for simulation.
ELSE	
N5 X300 OMA1=10	
ENDIF	

10.7 Block search, types 1, 2, and 4:

Function

Block search offers the possibility of starting part program execution from almost any part program block.

This involves the NC rapidly performing an internal run through the part program (without traversing motions) to the selected target block during block search. Here, every effort is made to achieve the exact same control status as would result at the target block during normal part program execution (e.g. with respect to axis positions, spindle speeds, loaded tools, NC/PLC interface signals, variable values) in order to be able to resume automatic part program execution from the target block with minimum manual intervention.

Block search types

- **Type 1: Block search without calculation**
  Block search without calculation is used to find a part program block in the quickest possible way. No calculation of any type is performed. The control status at the target block remains unchanged compared to the status before the start of the block search.
- **Type 2: Block search with calculation at contour**
  Block search with calculation at contour is used to enable the programmed contour to be approached in any situation. On NC Start, the start position of the target block or the end position of the block before the target block is approached. This is traversed up to the end position. Processing is true to contour.
- **Type 4: Block search with calculation at block end point**
  Block search with calculation at block end point is used to enable a target position (e.g. tool change position) to be approached in any situation. The end position of the target block or the next programmed position is approached using the type of interpolation valid in the target block. This is not true to contour.
  Only the axes programmed in the target block are moved. If necessary, a collision-free initial situation must be created manually on the machine in "JOG REPOS" mode before the start of further automatic part program execution.
- **Type 5: Block search with calculation in "Program test" (SERUPRO) mode**
  SERUPRO (**se**arch **ru**n by **pro**gramtest) is a cross-channel block search with calculation. Here, the NC starts the selected part program in "Program test" mode. On reaching the target block, the program test is automatically deselected. This type of block search also enables interactions between the channel in which the block search is being performed and synchronized actions as well as with other NC channels.
  See Section "Block search Type 5 (SERUPRO) (Page 517)".

Note

For further explanations regarding the block search, see Section "Behavior during block search (Page 443)."

Subsequent actions

After completion of a block search, the following subsequent actions may occur:

- Type 1 - Type 5: Automatic Start of an ASUP
  When the last action block is activated, a user program can be started as an ASUP.
- Type 1 - Type 4: Cascaded block search
  A further block search with a different target specification can be started from "Search target found".

10.7.1 Description of the function

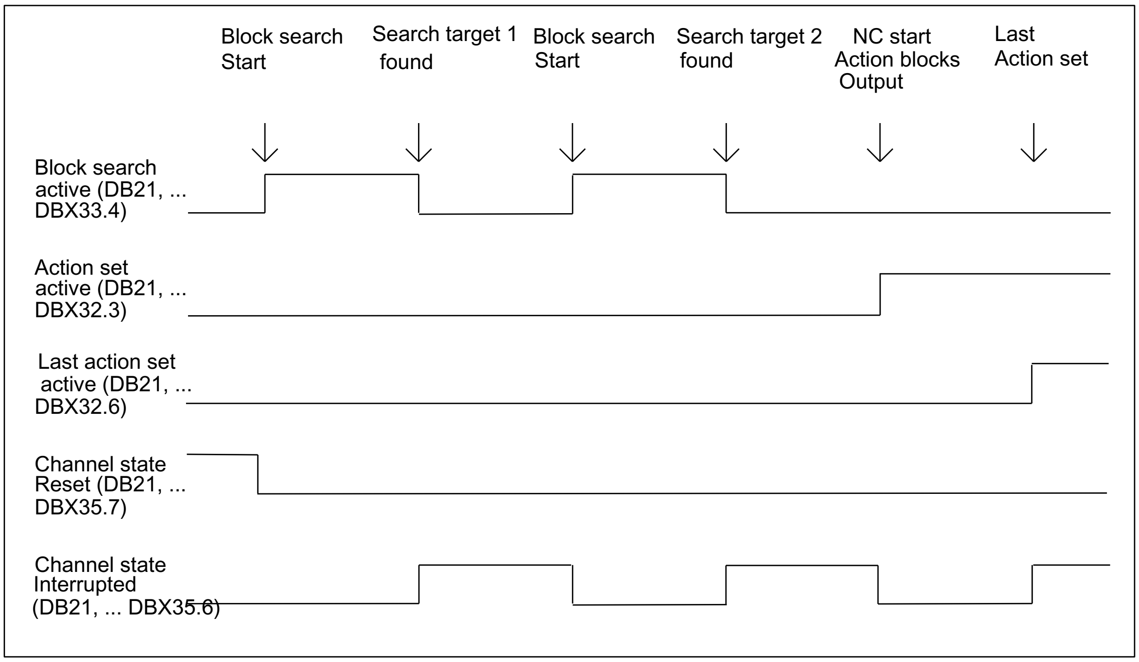
Basic sequence for type 2 or type 4

1. User: Activation of the block search **type 2** or **type 4** (block search **with calculation** to ...) via the operator interface
2. Search for the target block with collection of auxiliary functions
3. Stop after "Search target found ⇒ display of alarm 10208 "Continue program with NC start"
4. User: NC start to execute the action block s⇒ DB21, ... DBX7.1 = 1 (NC start)
5. Execution of the action blocks
6. Last action block is activated ⇒ Automatic start of /_N_CMA_DIR/_N_PROG_EVENT_SPF (default) as an ASUB.
7. Last ASUP block (`REPOSA`) is activated ⇒ DB21, ... DBX32.6 = 1 (last action block active)
8. **Optional**: Execution of user-specific requirements via PLC user program
9. Display of the alarm 10208 "Continue program with NC start"?
10. User: Continue program with NC start ⇒ DB21, ... DBX7.1 = 1 (NC start)

Search target not found

If the search target is not found, alarm 15370 "Search target not found during block search" is displayed and the block search is canceled.

Time sequence


Block search Start
Search target 1 found
Block search Start
Search target 2 found
NC start Action blocks Output
Last Action set
Block search active (DB21, ... DBX33.4)
Action set active (DB21, ... DBX32.3)
Last action set active (DB21, ... DBX32.6)
Channel state Reset (DB21, ... DBX35.7)
Channel state Interrupted (DB21, ... DBX35.6)

Action blocks

During a block search **type 2** or **type 4** (block search **with calculation** to ...), actions such as tool (T, D), spindle (S), feedrate programming or M function outputs are collected. With NC start to execute the action blocks, the collected actions are output to the PLC.

Note

With NC start for the action blocks, the spindle programming collected during block search **type 2** or **type 4** (block search **with calculation** to ...) (S value, `M3` / `M4` / `M5` / `M19`, `SPOS`) becomes active.

The user has to ensure in the PLC user program that the tool can be operated or the spindle programming is reset or not output:

DB31, ... DBX2.2 = 1 (spindle reset) .

Supplementary conditions

Continuation mode after block search type 4

If initial programming of an axis is performed incrementally to a block search type 4 (block search with calculation to block end point), the programmed incremental value can be added to the position value collected up to the search target or to the current actual value of the axis. The setting is made via:

SD42444 $SC_TARGET_BLOCK_INCR_PROG

The setting data is evaluated with NC start for outputting the action blocks.

Single block

If you do not wish to stop after each action block after the search target during block search type 2 or type 4 (block search **with calculation** to ...) is found and while function "Single block" (DB21, ... DBX0.4 == 1 (activate single block)) is active, this behavior can be deactivated in machine data:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, Bit 3 = 1 (ignore signal block for action blocks)

Interface signal "start block active"

The interface signal is **only** set during block search **type 2** (block search with **calculation to contour**):

- DB21, ... DBX32.4 = 1 (approach block active)

For block search **type 4** (block search with **calculation to block end point**), the interface signal is not set because no approach block is generated here (approach block equal to target block).

Type of interpolation of the target block

For a block search **type 4** (block search with **calculation to block end point**), the start movement is performed in the interpolation type valid in the target block. For interpolation types other than linear interpolation (`G0` or `G1`), the start movement can be canceled with an alarm (e.g. circle end point error in circular interpolation `G2` or `G3`).

10.7.2 Block search in connection with other NCK functions

10.7.2.1 ASUB after and during block search

Block search type 2 and type 4: Synchronization of the channel axes

If an ASUP is started after a block search type 2 or type 4 (block search with calculation to ...)", the actual positions of all channel axes are synchronized.

Effect

If the following system variables are read in the ASUP, they contain the following values:

- $P_EP: Current actual position of the channel axis in the WCS
- $AC_RETPOINT: Collected block search position of the channel axis in the WCS

Block search type 2: ASUP completion

For block search type 2 (block search with calculation at contour), the following command `REPOSA` (reapproach to the contour; linear; all channel axes) must be programmed by completion of the ASUP.

Effect

- All channel axes are moved to their search position that was collected during the block search.
- $P_EP == "collected block search position of the channel axis (WCS)"

Block search type 4: `REPOS` behavior

After block search type 4 (block search with calculation to block end point), no automatic repositioning is triggered during the period described by the **beginning** and **end** by the `REPOS` command:

- **Begin**: NC/PLC interface signals DB21, ... DBX32.6 == 1 (last action block active)
- **End**: Continuing program execution with NC start.

The start point of the approach movement are the current positions of the channel axes at the time of the NC start command. The end point results from the other transversing movements programmed in the part program.

For block search type 4, no approach movement is generated by the NC.

Effect:

- After exiting the ASUP, the system variable $P_EP (programmed end position) contains the actual position, at which the channel axes were positioned by the ASUP or manually (mode: JOG).
  $P_EP == "current actual position of the channel axis in the WCS

10.7.2.2 PLC actions after block search

If all action blocks have been executed by the NC and actions by the PLC, e.g. starting an ASUP to perform a tool change, or of the operator, e.g. restore, are possible, the following channel-specific NC/PLC interface signal is set:

- DB21, ... DB32.6 = 1 (last action block active)

Parameterizing the time of alarm output

To inform the operator that NC start is required in the channel to continue program execution, alarm 10208 "Continue program with NC start" is displayed.

When the alarm is displayed can be set via machine data:
MD11450 $MN_SEARCH_RUN_MODE, Bit 0 = <value>

<value>	Meaning
0	With the change of the last action block after a block search, the following takes place: • Execution of the part program is stopped • DB21, ... DBB32.6 = 1 (last action block active) • Display alarm 10208
1	With the change of the last action block after a block search, the following takes place: • Execution of the part program is stopped • DB21, ... DBB32.6 = 1 (last action block active) • Only display alarm 10208 if DB21, ... DBX1.6 == 1 (PLC action ended)

In combination with the alarm, the following interface signals are set:

- DB21, ... DBX36.7 = 1 (NC alarm with machining standstill is pending)
- DB21, ... DBX36.6 = 1 (NC alarm channel-specific is pending)

10.7.2.3 Spindle functions after block search

Control system response and output

Whether the spindle-specific auxiliary functions collected during block search are automatically output to the PLC in the action block or user-specifically at a later time can be set in

MD11450 $MN_SEARCH_RUN_MODE, Bit 2 = <value>

<value>	Meaning
0	Output of spindle-specific auxiliary functions collected during block search (`M3`, `M4`, `M5`, `M19`, `M70`) in action blocks.
1	The spindle-specific auxiliary functions collected during block search are not output in the action block. They can be output at a later time, for example, in an ASUP. The spindle-specific auxiliary functions collected are stored in system variables:

System variables

The spindle-specific auxiliary functions collected during block search are stored in the following system variables:

System variable	Description
$P_SEARCH_S [<n>]	The spindle speed or cutting speed last programmed
$P_SEARCH_SDIR [<n>]	Spindle direction last programmed
$P_SEARCH_SGEAR [<n>]	Gear stages last programmed M function
$P_SEARCH_SMODE [<n>]	Spindle mode last programmed
$P_SEARCH_SPOS [<n>]	Spindle position or traverse path last programmed by means of `M19`, `SPOS` or `SPOSA`
$P_SEARCH_SPOSMODE [<n>]	Position approach mode last programmed by means of `M19`, `SPOS`, or `SPOSA`
<n>: Spindle number	

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.

For more detailed information on ASUB, block search and action blocks, see Sections "Output suppression of spindle-specific auxiliary functions (Page 448)" and "Program test (Page 495)."

10.7.2.4 Reading system variables for a block search

In part programs, values can be read from the areas processing, main run, or servo/drive via system variables:

$P_...	Preprocessing-related system variables contain programmed values
$A_...	Main-run-related system variables contain current values
$V_...	Servo/drive-related system variables contain current values

Because no blocks enter the main run during a block search of type 2 and type 4 (block search **with calculation** to ...), main run and servo/drive-related system variables are not changed during the block search. Where necessary, for these system variables, the block search must specially handled in the NC program by querying whether a block search is active $P_SEARCH (block search active).

Preprocessing-related system variables provide correct values in all search types.

10.7.3 Automatic start of an ASUB after a block search

Parameter assignment

Making the function effective

The automatic ASUB start after a block search is activated by the following MD setting:

MD11450 $MN_SEARCH_RUN_MODE, bit 1 = 1

Program to be activated

In the default setting, the program **_N_PROG_EVENT_SPF** is activated from the directory _N_CMA_DIR as ASUB after the block search by changing the last action block. If another program is to be activated, then the name of this user program must be entered in the following machine data:

MD11620 $MN_PROG_EVENT_NAME

Behavior when the single-block processing is set

The following channel-specific machine data are used to set whether the activated ASUP will be processed without interruption although single-block processing is set or whether single-block processing will be activated:

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK, bit 4 = <value>

<value>	Meaning
0	Single-block processing is active.
1	Single-block processing is suppressed.

Behavior when the read-in disable is set

The channel-specific machine data is used to set whether the ASUP will be processed without interruption although read-in disable (DB21, ... DBX6.1 = 1) is set, or whether the read-in disable will be effective:

MD20107 $MC_PROG_EVENT_IGN_INHIBIT, Bit 4 = <value>

<value>	Meaning
0	Read-in disable is active.
1	Read-in disable is suppressed.

Note

For more information on parameterizing MD11620, MD20108, and MD20107, see Section "Parameterization (Page 574)."

Programming

The event by which the ASUP has been started is stored in the system variable $P_PROG_EVENT. On automatic activation after a block search, $P_PROG_EVENT returns the value "5."

Sequence

Sequence of automatic start of an ASUB after a block search

1. User: Activation of the block search **type 2** or **type 4** (block search **with calculation** to ...) via the operator interface
2. Search for the target block with collection of auxiliary functions
3. Stop after "Search target found ⇒ display of alarm 10208 "Continue program with NC start"
4. User: NC start to execute the action block s⇒ DB21, ... DBX7.1 = 1 (NC start)
5. Execution of the action blocks
6. Last action block is activated ⇒ Automatic start of /_N_CMA_DIR/_N_PROG_EVENT_SPF (default) as an ASUB.
7. Last ASUP block (`REPOSA`) is activated ⇒ DB21, ... DBX32.6 = 1 (last action block active)
8. **Optional**: Execution of user-specific requirements via PLC user program
9. Display of the alarm 10208 "Continue program with NC start"?

   Note

   With MD11450 $MN_SEARCH_RUN_MODE, Bit 0 == 1, alarm 10208 will only be output after enabling by the PLC user program (DB21, ... DBX1.6 = 1 (PLC action ended)).

10. User: Continue program with NC start ⇒ DB21, ... DBX7.1 = 1 (NC start)

10.7.4 Cascaded block search

Functionality

The "Cascaded block search" function can be used to start another block search from the status "Search target found". The cascading can be continued after each located search target as often as you want and is applicable to the following block search functions:

- Type 1 block search without calculation
- Type 2 block search with calculation at contour
- Type 3 block search with calculation at block end point

Note

Another "cascaded block search" can be started from the stopped program execution only if the search target has been found.

Activation

The "cascaded block search" is configured in the existing machine data:
MD11450 $MN_SEARCH_RUN_MODE

- The 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 block search

When the search target is reached, the program execution is stopped and the search target displayed as current block. After each located search target, a new block search can be repeated as often as you want.

Changing the search target specifications

You can change the search target specifications and block search function before every block search start.

Example: Execution 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

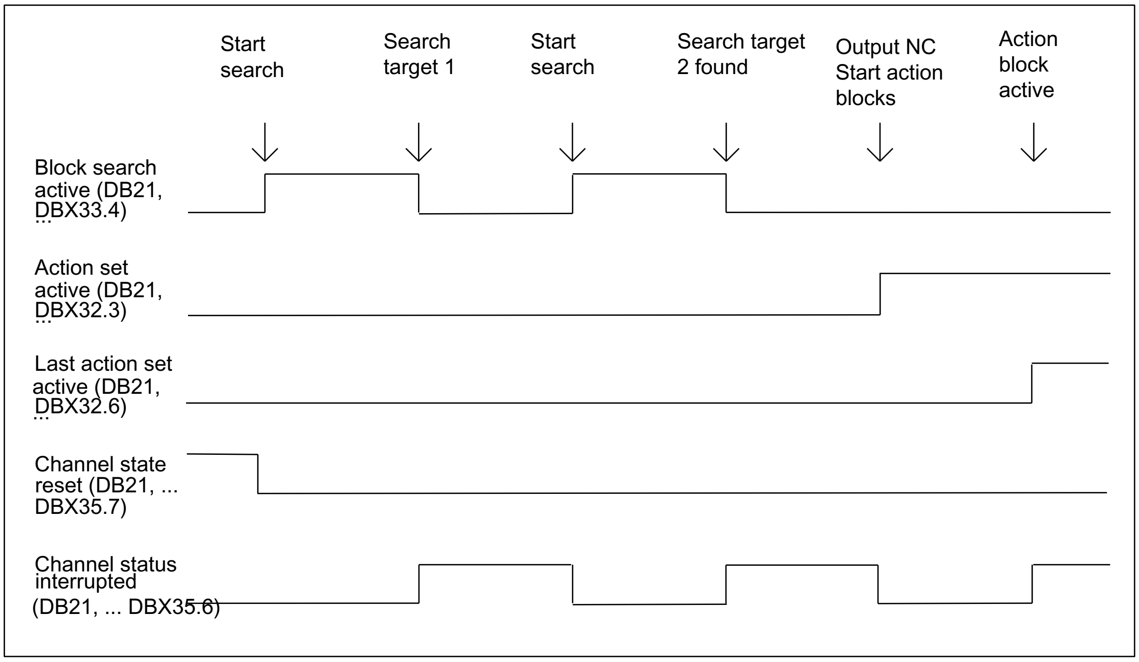

Start search
Search target 1
Start search
Search target 2 found
Output NC Start action blocks
Action block active
Block search active (DB21, ... DBX33.4)
Action set active (DB21, ... DBX32.3)
Last action set active (DB21, ... DBX32.6)
Channel state reset (DB21, ... DBX35.7)
Channel status interrupted (DB21, ... DBX35.6)

Figure 10-2 Chronological order of interface signals

10.7.5 Examples for block search with calculation

Selection

From the following examples, select the type of block search that corresponds to your task.

Type 4 block search with calculation at block end point

Example with automatic tool change after block search with active tool management:

1. Set machine data:
   MD11450 $MN_ SEARCH_RUN_MODE to 1
   MD11602 $MN_ASUP_START_MASK Bit 0 = 1 (ASUP Start from stopped state)
2. Select ASUP "BLOCK_SEARCH_END" from PLC via FB4 (see also Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)").
3. Load and select part program "WORKPIECE_1".
4. Block search to block end point, block number `N220`.
5. HMI signals "Search target found".
6. NC Start for output of action blocks.

7. With the PLC signal:
   DB21... DB32.6 (last action block active),
   the PLC starts ASUP "BLOCK_SEARCH_END" via FC9 (see also Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)").
8. After the end of the ASUP (can be evaluated, e.g. via M function `M90` to be defined, see example for block `N1110`), the PLC sets signal
   DB21, ... DBX1.6 (PLC action complete).
   Alternatively, NC/PLC interface signal:
   DB21-DB30 DBB318 bit 0 (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, JOG-REPOS, overstoring)
10. Continue part program with NC Start.

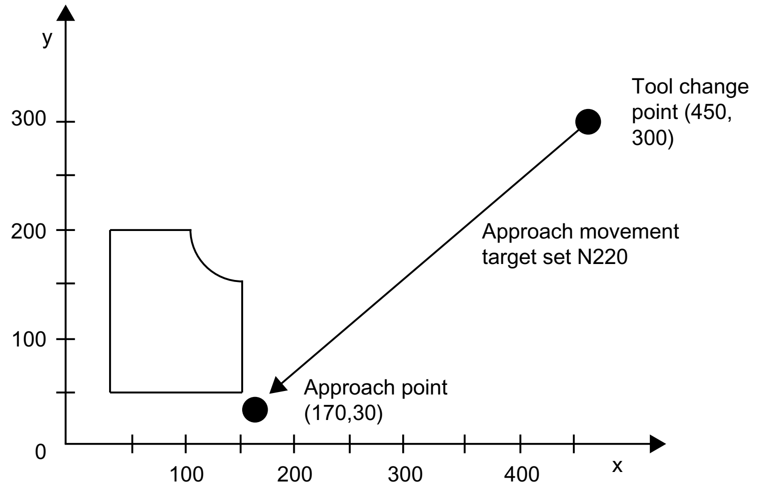

y
300
200
100
0
100
200
300
400
x
Tool change point (450, 300)
Approach movement target set N220
Approach point (170,30)

Figure 10-3 Approach motion for block search to block end point (target block `N220`)

Note

"Block search at contour" with target block `N220` would generate an approach motion to the tool change point (start point of the target block).

Type 2 block search with calculation at contour

Example with automatic tool change after block search with active tool management:

1. to 3.	Same as example for Type 4 block search
4.	Block search at contour, block number `N260`
5. to 10.	Same as example for Type 4 block search

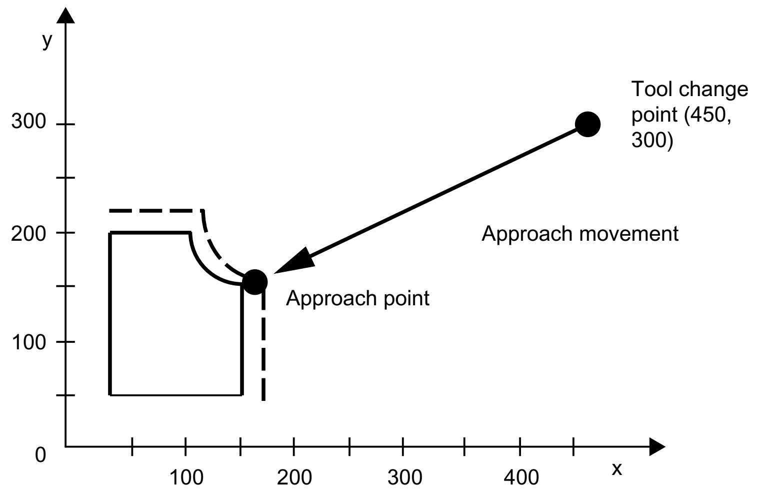


Figure 10-4 Approach motion for block search at contour (target block `N260`)

Note

"Block search to block end point" with target block `N260` would result in Alarm 14040 (circle end point error).

Part programs for Type 4 and Type 2

PROC WORKPIECE_1

Program code	Comment
; Main program	
...	
;Machine contour section 1 with **"CUTTER_1"**tool	
...	
N100 G0 G40 X200 Y200	; Deselect radius compensation
N110 Z100 D0	; Deselect length correction
;End of contour section 1	
;	
;Machine contour section 2 with **"CUTTER_2"**tool	
N200 T="CUTTER_2"	; Preselect tool
N210 WZW	; Call tool change routine
N220 G0 X170 Y30 Z10 S3000 M3 D1	; Approach block for contour section 2
N230 Z-5	; Infeed
N240 G1 G64 G42 F500 X150 Y50	; Start point of contour
N250 Y150	
N260 G2 J50 X100 Y200	
N270 G1 X50	
N280 Y50	
N290 X150	
N300 G0 G40 G60 X170 Y30	; Deselect radius compensation

Program code	**Comment**
`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 point`
`N560 M6`	`; Execute tool change`
`;`	
`END: M17`	
`PROC SUCHLAUF_ENDE SAVE`	
`;`**`ASUP`**` 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 ; block search`
`N1030 TNR_AKTIV = $TC_MPP6[9998,1]`	`; Read T number of active tool`
`N1040 TNR_SUCHLAUF = $P_TOOLNO`	`; Read T number determined by block search`
`N1050 GETSELT(TNR_VORWAHL)`	`; Read T number of preselected tool`
`N1060 IF TNR_AKTIV ==TNR_SUCHLAUF GOTOF ASUP_ENDE`	
`N1070 T = $TC_TP2[TNR_SUCHLAUF]`	`; T selection by tool name`
`N1080 WZW`	`; Call tool change routine`
`N1090 IF TNR_VORWAHL == TNR_SUCHLAUF GOTOF ASUP_ENDE`	
`N1100 T = $TC_TP2[TNR_VORWAHL]`	`; Restore T preselection by tool name`
`ASUP_ENDE:`	
`N1110 M90`	`; Feedback to PLC`
`N1120 REPOSA`	`; ASUP end`

10.7.6 Supplementary conditions

10.7.6.1 Compressor functions (COMPON, COMPCURV, COMPCAD)

- If the target block for block search **type 2** or **type 4** (block search **with calculation** to ...) is in a program section in which a compressor function (`COMPON`, `COMPCURV`, `COMPCAD`) is active, positions are approached on the path calculated by the **compressor** on repositioning. These positions must precisely match the positions on the path programmed in the part program.
- If programmed blocks are eliminated from the part program during compression, these blocks will not be found at the target block in the block search ⇒ Alarm 15370 "Search target not found."

10.8 Block search Type 5 (SERUPRO)

10.8.1 Description of the function

Block search type 5, block search with calculation in the "Program test" mode (SERUPRO, "**Se**arch-**Ru**n by **Pro**gram test") enables a cross-channel block search with calculation at a selectable interruption point. Taking into account existing program coordination commands, all the status data required to continue the program in the interrupted channels is determined during SERUPRO and then the NC and PLC set to a state permitting the program continuation.

Before repositioning with subsequent continuation of the program execution, all the output states that may still be required can be automatically generated via a user-specific ASUP.

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 block search function for HMI and select block search configuration
SeruproEnabled=1	;SERUPRO softkey available for HMI. Default value is (1)
SeruproConfig=1	;Number (1) to (4) of above indicated channel grouping. Default value is (1)

All other channels started with SERUPRO are operated in "Self-Acting SERUPRO" mode. Only the channel in which a target block has been selected can be started with a block search in SERUPRO mode.

Activation

SERUPRO is activated via the HMI. SERUPRO is operated using the "Prog.Test Contour" softkey.

SERUPRO uses REPOS to approach the target block.

Chronological sequence of SERUPRO

1. Via HMI, softkey "Pog. test contour" and the search target are operated.
2. The NC now automatically starts the selected program in "Program test" mode.
   - In this mode, axes are not traversed.
   - Auxiliary functions $A_OUT and the direct PLC IO are output.
   - The auxiliary functions of the target block are not output.
3. The NC stops at the beginning of the target block, deselects the program test internally, and displays the stop condition "Wait: Search target found".
4. If the user-specific ASUP "PROG_EVENT.SPF" is available, it is started automatically.
5. Repositioning is performed with the next NC start (REPOS).
   The REPOS operation is performed via a system ASUP and can be extended using the "Editable ASUP" function.

Boundary conditions for block search SERUPRO

The SERUPRO function may only be activated in "AUTOMATIC" mode and may only be aborted in program state (channel state RESET).

If in normal mode **only** the PLC starts commonly several channels, then this can be simulated by SERUPRO in each channel.

With machine data setting:
MD10708 $MN_SERUPRO_MASK, bit 1 = 0,
alarm 16942: "Channel %1 Start program command action %2<ALNX> not possible"
aborts the simulation if part program command `START` is used.

Machine data:
MD10707 $MN_PROG_TEST_MASK
allows shutdown in the stopped state and has no effect on the SERUPRO operation. The default setting allows program testing to be deactivated only in the RESET state.

Note

After program testing has been deactivated, a REPOS operation is initiated that is subject to the same restrictions as a SERUPRO approach operation. Any adverse effects can be inhibited using an ASUP.

Controlling SERUPRO behavior

For the functions listed below as an example, the SERUPRO behavior can be set specifically for the NC:

- Programmed stop (`M0`)
- Program coordination command `START`
- Group SERUPRO
- Cross-channel exiting of SERUPRO
- Override

MD10708 $MN_SERUPRO_MASK = <behavior with SERUPRO>

Channel-specific basic settings for SERUPRO

The channel-specific basic settings are normally specified with the following machine data after a part program start:

MD20112 $MC_START_MODE_MASK= basic settings>

You can specify your own basic settings for SERUPRO which replace the basic settings from MD20112:

MD22620 $MN_START_MODE_MASK_PRT = <SERUPRO basic settings>

The SERUPRO basic settings must be explicitly released via:

MD22621 $MC_ENABLE_START_MODE_MASK_PRT = 1

NC/PLC interface signal "Block search via program test is active"

The block search via program test is displayed using the NC/PLC interface signal:
DB21, ... DBX318.1 == 1

The interface signal is set from the start of the block search until the target block is inserted into the main run.

For user-defined ASUP after the SERUPRO operation

Note

If the machine manufacturer decides to start an ASUP after the SERUPRO operation as described in point 7, the following must be observed:

Stopped status acc. to point 6:

Machine data:
MD11602 $MN_ASUP_START_MASK
and
MD11604 $MN_ASUP_START_PRIO_LEVEL
allow the NC to start the ASUP from the stopped status automatically via the FC9 block.

Acknowledgment of FC9 only after completion of REPOS block:

The ASUP can only be signaled as complete from the FC9 block with "ASUP Done" if the REPOS block has also been completed.

Deselection of assigned REPOS operation after point 8:

The start of the ASUP deselects the assigned REPOS operation!

Therefore, the ASUP 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 ASUP with `M17` or `RET`.

Special handling of ASUP:

As a basic rule, an ASUP that ends with REPOS and is started from stopped state receives special treatment.

The ASUP stops automatically before the REPOS block and indicates this via:

DB21, ... DBX318.0 (ASUP is stopped)

Automatic ASUP start

The ASUP in path:
/_N_CMA_DIR/_N_PROG_EVENT_SPF
is started automatically in SERUPRO approach with machine data:
MD11450 $MN_SEARCH_RUN_MODE, bit 1 = 1
according to the following sequence:

1. The SERUPRO operation has been performed completely.
2. The user presses "NC start".
3. The ASUP is started.
4. The NC stops automatically **before** the `REPOS` part program command and the message "Press NC start to continue the program" appears.
5. The user presses "NC start" again.
6. The NC executes the repositioning motion and continues the part program at the target block.

Note

The automatic ASUP start with MD11450 requires **Starts** to continue the program.

The procedure is in this respect similar to other block search types.

10.8.2 Repositioning to the contour (REPOS)

The "Reposition to the contour" (REPOS) function can be used to continue an interrupted machining at the interrupted location. Unlike REPOS, SERUPRO permits the "refetching" or "repetition" of a program section. For this purpose, once SERUPRO has found the target block, the contour is positioned at the location selected with REPOS mode and the machining continued.

SERUPRO: Set REPOS response

The REPOS behavior, i.e. the behavior in the repositioning block, is set with the following machine data:
MD11470 $MN_REPOS_MODE_MASK = <REPOS mode>

<REPOS mode>		
Bit	**Value**	**Meaning**
0	0	An interrupted delay time is repeated
	1	An interrupted delay time is continued
1	-	Reserved
2	0	DB31, ... DBX10.0 (REPOS delay) **not** considered in the repositioning deceleration block.
	1	DB31, ... DBX10.0 (REPOS delay) considered in the repositioning deceleration block.
3	0	**SERUPRO**: Only path axes traverse in the repositioning block
	1	**SERUPRO**: Path and positioning axes traverse concurrently in the repositioning block
4	0	**REPOS**: Only path axes traverse in the repositioning block
	1	**REPOS**: Path and positioning axes traverse concurrently in the repositioning block
5	0	During the interruption, changed feedrates and spindle speeds act only after the first part program block following the interruption location
	1	Modified feedrates and spindle speeds during the interruption are valid immediately from the interruption point onward, i.e. they are already valid in the residual block and are given priority. This behavior relates to every REPOS operation.
6	0	SERUPRO: In the repositioning block, neutral axes and positioning spindles traverse as path axes.
	1	SERUPRO: In the repositioning block, neutral axes and positioning spindles traverse as command axes. Neutral axes and positioning spindles are repositioned after SERUPRO. For neutral axes, where it is not permissible to reposition them, the REPOS delay must be set: DB31, ... DBX10.0 = 1 (REPOS delay)
7	0	The REPOS delay is **not** enabled.
	1	The REPOS delay is enabled. Axes with active REPOS delay (DB31, ... DBX10.0 == 1), which are neither geometry nor orientation axes, are **not** traversed when repositioning.

⚠ CAUTION

Risk of collision

MD11470 $MN_REPOS_MODE_MASK, Bit 3 or Bit 4 = 1

The user alone is responsible for ensuring that the concurrent traversal of the axes in the repositioning block does not cause any collision on the machine.

Repositioning with controlled REPOS

The REPOS mode can be specified for the path axes via the NC/PLC interface:

DB21, ... DBX31.0 - 2 (REPOS mode)

The REPOS mode is programmed in the NC program, and defines the approach behavior (see Section "Repositioning with controlled REPOS (Page 528)").

The REPOS behavior of individual axes can also be controlled via NC/PLC interface signals, and must be enabled using 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, REPOS of individual axes can be prevented temporarily and also delayed. NC/PLC interface signals can be used to subsequently re-enable or to continue blocking individual channel axes that REPOS would like to traverse.

⚠ DANGER

Risk of collision

Signal DB31, ... DBX2.2 (delete distance-to-go) produces the following dangerous behavior when function "Prevent repositioning of individual axes" is selected:

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 the example below: Axis is programmed incrementally

Example: Rotary axis A is programmed incrementally

Rotary axis A is the fourth machine axis.

- Rotary axis A is positioned at 11° before the REPOS operation.
  In the interruption block, i.e. in the target block of SERUPRO, rotary axis A should be moved to 27°.
  Any number of blocks later, rotary axis A is moved incrementally through 5°:
  `N1010 POS[A]=IC(5) FA[A]=1000`
  When interface signal DB34 DBX10.0 = 1 is set (REPOS delay), rotary axis A is not moved in the REPOS operation, and is moved with `N1010` to 32°. The user may have to consciously acknowledge the movement from 11° to 27°.

DANGER

Risk of collision

As, after the interruption, the axis is incrementally programmed, it moves to 16° instead of 32°.

- Starting 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 is then executed.
- Repositioning 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)
  MD11470 $MN_REPOS_MODE_MASK.bit 4 = 1: for each REPOS

Note

If neither bit 3 nor bit 4 is set, in that this phase non-path axes are repositioned in the residual block.

Delayed approach of axis with REPOS offset

If the axis-specific interface signal DB31, ... DBX10.0 (REPOS delay) is set with the positive edge of DB21, ... DBX31.4 (REPOS mode change), the REPOS offset for this axis is only traversed through when programmed the next time.

Whether this axis is currently subject to a REPOS offset can be read via synchronized actions with $AA_REPOS_DELAY.

⚠ CAUTION

Risk of collision

DB31, ... DBX10.0 (REPOS delay) have no effect on machine axes, which form a path.

Whether an axis is a path axis can be determined using DB31, ... DBX76.4 (path axis).

Acceptance timing of REPOS SIGNALS

With the positive edge of DB21, ... DBX31.4 (REPOS mode change), the following REPOS signals are transferred into the NC:

- Channel-specific: DB21, ... DBX31.0 - 2 (REPOS mode)
- Axis-specific: DB31, ... DBX10.0 (REPOS delay)

The level of the REPOS signals refers to the actual main run block. There are two different cases:

1. **One** repositioning block of a currently active REPOS operation is contained in the main run. The active REPOS operation is canceled, restarted and the REPOS offsets are influenced by the REPOS signals mentioned above:
2. **No** repositioning block of a currently active REPOS operation is contained in the main run. Each future REPOS operation wanting to reapproach the current main program block is influenced by the REPOS signals mentioned above.

Note

In a running ASUP, DB21, ... DBX31.4 (REPOS mode change) does not act on the final REPOS, unless the signal is inadvertently set at the instant in time that the REPOS blocks are being executed.

In the 1st case, the signal is allowed only in the stopped state.

Response to RESET**:**

- The NC has already acknowledged the PLC signal:
  DB21, ... DBX31.4 == 1 (REPOS mode change) **AND**
  DB21, ... DBX319.0 == 1 (REPOS mode change acknowledgment)
  If, in this situation, a channel reset occurs, then the active REPOS mode is cleared:
  DB21, ... DBX319.1 - 3 = 0 (active REPOS mode)
- The NC has still not acknowledged the PLC signal:
  DB21, ... DBX31.4 == 1 (REPOS mode change) **AND**
  DB21, ... DBX319.0 == 0 (REPOS mode change acknowledgment)
  If, in this situation, a channel reset occurs, then acknowledgment of the REPOS mode change and the active REPOS mode are cleared:
  DB21, ... DBX319.0 = 0 (REPOS mode change acknowledgment)
  DB21, ... DBX319.1 - 3 = 0 (active REPOS mode)

Controlling SERUPRO approach with NC/PLC interface signals

The SERUPRO approach can be used with:DB21, ... DBX31.4 (REPOS mode change) and the associated signals in the following phases:

- From "Search target found" to "Start SERUPRO ASUP"
- From "SERUPO-ASUP stops automatically before REPOS" to "Target block is executed"

While the SERUPRO ASUP is being executed, e.g. in the program section before the REPOS operation, the interface signal does not affect the SERUPRO positioning

REPOS operations with NC/PLC interface signals

Control REPOS mit NC/PLC interface signals

REPOS offsets can be influenced with the following NC/PLC interface signals:

- DB21, ... DBX31.0 - 2 (REPOS mode)
- DB21, ... DBX31.4 (REPOS mode change)
- DB31, ... DBX10.0 (REPOS delay)
- DB31, ... DBX72.0 (REPOS delay)

REPOS acknowledgment signals

The following NC/PLC interface signals can be used to acknowledge **from the NC**, functions that control the REPOS response via PLC:

- DB21, ... DBX319.0 (REPOS mode change acknowledgment)
- DB21, ... DBX319.1 - 3 (active REPOS mode)
- DB21, ... DBX319.5 (REPOS acknowledgment delay)
- DB31, ... DBX70.0 (REPOS offset)
- DB31, ... DBX70.1 (REPOS offset valid)
- DB31, ... DBX70.2 (REPOS delay acknowledgment)
- DB31, ... DBX76.4 (path axis)

For further information, see "REPOS offset in the interface"

REPOS acknowledgment operations

If the NC recognizes a REPOS mode change (DB21, ... DBX31.4 == 1), then this is acknowledged by the PLC with DB21, ... DBX319.0 = 1.

Note

If the NC has not yet acknowledged interface signal DB21, ... DBX31.4 (REPOS MODE CHANGE) with interface signal DB21, ... DBX319.0 (REPOS mode change acknowledgment) , then a channel reset in this situation causes the program to be canceled, and the REPOS, which is to be used to control the REPOS mode, is not executed.

A REPOS mode issued from the PLC is acknowledged by the NC using the following interface signals:

- DB21, ... DBX319.1 - 3 (active REPOS mode)
- DB31, ... DBX10.0 (REPOS delay)
- DB31, ... DBX70.2 (REPOS delay acknowledgment)

Example

- Instant in time ②: An NC program in block `N20` is stopped with an NC stop. All axes are braked along their parameterized braking ramps to standstill.
- Instant in time ③: After the PLC user program has set the "REPOS mode", the NC accepts the REPOS mode with the 0/1 edge of "REPOS mode change".
- Instant in time ④: "REPOS mode change acknowledgment" remains set until the ASUP is initiated.
- Instant in time ⑤: The REPOS operation begins in the ASUP.
- Instant in time ⑥: The residual block of the ASUP is reloaded.

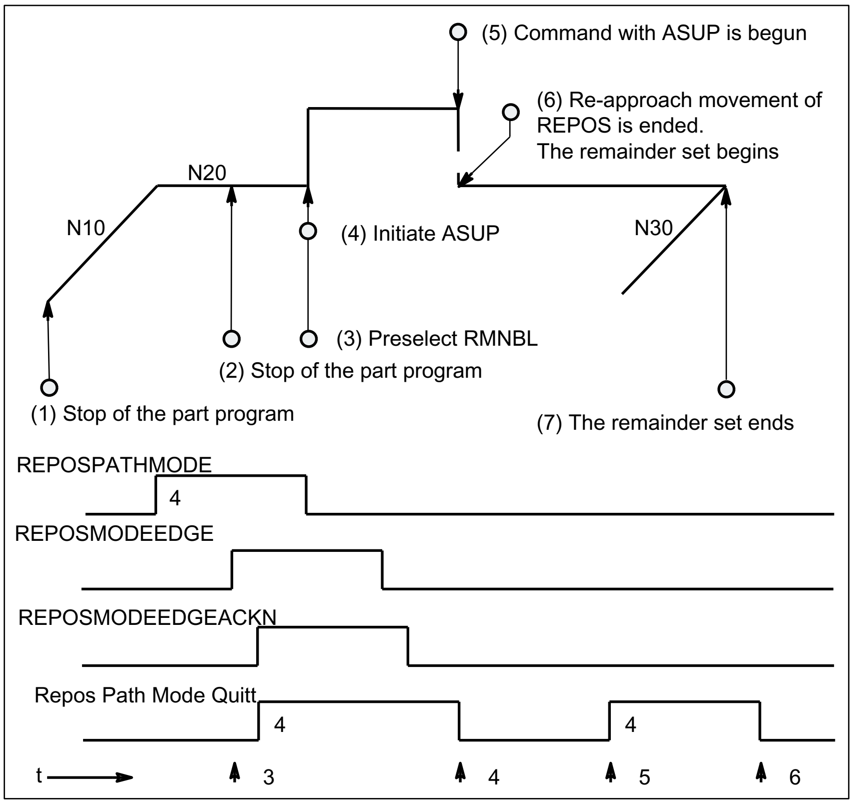

(5) Command with ASUP is begun
(6) Re-approach movement of REPOS is ended. The remainder set begins
N20
N10
(4) Initiate ASUP
N30
(3) Preselect RMNBL
(2) Stop of the part program
(1) Stop of the part program
(7) The remainder set ends
REPOSPATHMODE
4
REPOSMODEEDGE
REPOSMODEEDGEACKN
Repos Path Mode Quitt
4
4
t
3
4
5
6

Figure 10-5 REPOS sequence in part program with timed acknowledgment signals from NC

NC sets acknowledgment 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 ASUP is executed, the NC sets the "Repos Path Mode Ackn" again (→ Time (5)). If no REPOSPATHMODE has been preselected via an NC/PLC interface signal, the programmed REPOS mode is displayed.

"Repos Path Mode Ackn" is canceled when the residual block is activated (→ Time (6)). The part program block `N30` following the block at → Time (2) is resumed.

Interface signal:
DB31, ... DBX70.2 (REPOS delay acknowledgment) is analogously defined.

DB31, ... DBX70.1 (REPOS offset valid) = 1, if:

DB21, ... DBX319.1-319.3 (active REPOS mode) = 4 (RMNBL).

Valid REPOS offset

At the end of the SERUPRO operation, the user can read out the REPOS offset via the axis/spindle NC/PLC interface signal (NC→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 ASUP or the automatic ASUP start.

Updating the REPOS offset in the range of validity

Between the SERUPRO end and SERUPRO start, the axis can be moved in JOG mode with a mode change.

In JOG mode, the user manually moves the axis over the REPOS offset path in order to set interface signal:
DB31, ... DBX70.0 (REPOS offset) to the value 0.

Within the range of validity, the axis can also be traversed using FC18, whereby the interface signal
DB31, ... DBX70.0 (REPOS offset) is continuously updated.

Displaying the range of validity

The range of validity of the REPOS offset is indicated with interface signal:

DB31, ... DBX70.1 (REPOS offset valid)

It is indicated whether the REPOS offset calculation was valid or invalid:

Value 0:	The REPOS offset of this axis is calculated correctly.
Value 1:	The REPOS offset of this axis cannot be calculated, as the REPOS has not yet occurred, e.g. it is at the end of the ASUP, or no REPOS is active.

REPOS offset after an axis interchange

The group signal:DB21, ... DBX319.5 (REPOS delay) 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 axis-specific NC/PLC interface signal DB31, ... DBX76.4 (path axis) is 1 if the axis is part of the path grouping.

This signal shows the status of the current block to be executed during block change. Subsequent status changes are ignored.

If the SERUPRO operation is ended with "Search target found", DB31, ... DBX76.4 (path axis) refers to the target block.

10.8.2.1 Repositioning with controlled REPOS

Once SERUPRO has been used to find the target block, prior to continuing the interrupted program, a REPOS operation for repositioning the contour is performed. The REPOS mode "Reposition at the block start point of the target block" (RMBBL) is active by default. The REPOS mode can be defined user-specific via the NC/PLC interface:

DB21, ... DBX31.0 - .2 (REPOS mode)

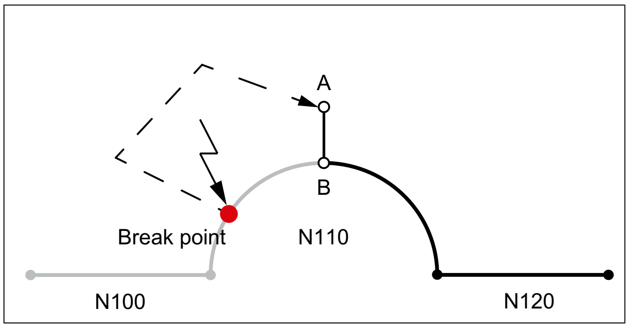
Reference:
A detailed description of the interface signals can be found in the NC Variables and Interface Signals List Manual

REPOS mode RMNBL repositioning to the next point on the path

in the `RMNBL` REPOS mode, positioning is made for the REPOS start position from the next nearest contour point.

Example

The program is interrupted at any point in the block N110. The axes were then traversed to position (A), e.g. manually. Once SERUPRO has found target block N110, the REPOS operation with REPOS mode `RMNBL` is performed. With regard to the REPOS start position (A), point (B) is the next nearest point of the contour. The REPOS operation is completed when point (B) is reached. The programmed contour of the interrupted program is traversed again starting at point (B).



Specifying the REPOS mode via the NC/PLC interface

The REPOS mode can be specified via the following NC/PLC interface signal:
DB21, ... DBX31.0 - .2 (REPOS mode)

Note

`RMNBL` is a general REPOS extension and it is not restricted to SERUPRO.

`RMIBL` and `RMBBL` behavior identically for SERUPRO.

DB21, ... DBX31.0 - .2 (REPOS mode) affects only the traversing motion of the path axes.

The behavior of the other axis can be changed individually using interface signal DB31, ... DBX10.0 (REPOSDELAY). The REPOS offset is not applied immediately, but only when it is next programmed.

For further information on the programming of the repositioning point, see:

References:
Programming Manual, Job Planning; Path Behavior, Section: Repositioning on contour

---

10.8.3 Accelerate block search

Machine data settings

The speed of execution of the SERUPRO operation can be influenced via the following machine data.

MD22600 $MC_SERUPRO_SPEED_MODE = <value>

<Value>	Meaning
0	Program test with block search velocity / dry run feedrate: • Axes: MD22601 $MC_SERUPRO_SPEED_FACTOR * dry run feedrate • Spindles: MD22601 $MC_SERUPRO_SPEED_FACTOR * programmed speed Dynamic limitations of axes / spindles are **not** considered.
1	Program test with programmed velocity: • Axes: Dry run feedrate • Spindles: Programmed speed Dynamic limitations of axes / spindles are considered.
2	Program test with dry run feedrate Under program test, traversing is performed with the programmed velocity / speed. Dynamic limitations of axes / spindles are considered.
3	Program test with block search velocity Under program test, traversing is performed with the following velocity: • Axes: MD22601 $MC_SERUPRO_SPEED_FACTOR * programmed feedrate • Spindles: MD22601 $MC_SERUPRO_SPEED_FACTOR * programmed speed. Dynamic limitations of axes / spindles are **not** considered. **Note** When revolutional feedrate is active (e.g. `G95`), the programmed feedrate is not multiplied by MD22601 $MC_SERUPRO_SPEED_FACTOR but only the programmed spindle speed. This results, here too, in an increase in the effective path velocity by MD22601 $MC_SERUPRO_SPEED_FACTOR.

Supplementary conditions

Main axes

MD22600 $MC_SERUPRO_SPEED_MODE acts on the following main run axes with SERUPRO:

- PLC axes
- Command axes
- Positioning axes
- Reciprocating axes

Synchronized actions

NOTICE
Actions of synchronized action may not be executed with SERUPRO
Because during SERUPRO other actual values (e.g. axis positions) are generated internally that are different from those in normal program execution, with SERUPRO, conditions of synchronized actions that check the actual values (e.g. axis positions) are no longer recognized as true (TRUE) and that the action part of the synchronized action is therefore not executed.

Revolutional feedrate

Effects of MD22601 $MC_SERUPRO_SPEED_FACTOR during DryRun:

- Switchover of `G95/G96/G961/G97/G971` to `G94`
- Tapping and thread cutting: Normal dry run velocity.

Tapping without compensating chuck

- With "tapping without compensating chuck" (`G331`/`G332`), the spindle is interpolated under position control in a path grouping. The drilling depth (linear axis), the thread pitch, and speed (spindle) are defined.
  During dry run, the velocity of the linear axis is specified, the speed remains constant, and the pitch is adjusted.
  After SERUPRO, a position for the spindle results that is deviates from normal mode because the spindle has revolved fewer times in SERUPRO.

10.8.4 SERUPRO ASUB

SERUPRO ASUP special points

Special points should be noted for SERUPRO ASUP 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 ASUP

G74 reference point approach

If command `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` command into account!

Tool management

If tool management is active, the following setting is recommended:

MD20310 $MC_TOOL_MANAGEMENT_MASK Bit 20 = 0

The tool management command generated during the SERUPRO operation is thus **not** output to the PLC!

The tool management command has the following effect:

- The NC acknowledges the commands automatically.
- No magazine data is changed.
- Tool data is not changed.
  Exception:
  The tool enabled during the test mode can assume 'active' state. In this way, the wrong tool may be on the spindle after the SERUPRO operation.
  Remedy:
  The user starts a SERUPRO ASUP that is actually traversed. Prior to the start, the user can start an ASUP that loads the correct tool.

SERUPRO operation: Functionality: In sequence steps 2. to 6.
SERUPRO ASUP: Functionality: The sequence of step 7.

In addition, machine data setting MD20310 $MC_TOOL_MANAGEMENT_MASK Bit 11 = 1 is required because the ASUP may have to repeat a T selection.

Systems with tool management and auxiliary spindle are not supported by SERUPRO!

Example

Tool change subprogram

Program code	Comment
`PROC L6`	`; Tool change routine`
`N500 DEF INT TNR_AKTUELL`	`; Variable for active T number`
`N510 DEF INT TNR_VORWAHL`	`; Variable for preselected T number`
	`; Determine current tool`
`N520 STOPRE`	`; In program testing`
`N530 IF $P_ISTEST`	`; From the program context`
`N540 TNR_AKTUELL = $P_TOOLNO`	`; The "current" tool is read`
`N550 ELSE`	`; Otherwise, the tool of the spindle is read out.`
`N560 TNR_AKTUELL = $TC_MPP6[9998,1]`	`; Read tool T number on the spindle`
`N570 ENDIF`	
`N580 GETSELT(TNR_VORWAHL)`	`; Read T number of the preselected tool of the master spindle`
	`; Execute tool change only if tool is not yet current`
`N590 IF TNR_AKTUELL <> TNR_VORWAHL`	`; Approach tool change point`
`N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0`	
`N610 M206`	`; Execute tool change`
`N620 ENDIF`	
`N630 M17`	

ASUP for calling the tool change routine after type 5 block search

Program code	Comment
`PROC ASUPWZV2`	
`N1000 DEF INT TNR_SPINDEL`	`; Variable for active T number`
`N1010 DEF INT TNR_VORWAHL`	`; Variable for preselected T number`
`N1020 DEF INT TNR_SUCHLAUF`	`; Variable for T number determined in block search`
`N1030 TNR_SPINDEL = $TC_MPP6[9998,1]`	`; Read tool T number on the spindle`
`N1040 TNR_SUCHLAUF = $P_TOOLNO`	`; Read T number determined by block search, i.e. this 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`	`; ASUP end`

In both of the programs PROC L6 and PROC ASUPWZV2, the tool change is programmed with M206 instead of M6.
ASUP "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 ASUP is started, the spindle is not accelerated to the speed specified in the program because the SERUPRO ASUP is intended to move the new tool into the correct position at the workpiece after the tool change.

A spindle ramp-up is performed with SERUPRO ASUP as follows:

- SERUPRO operation has finished completely.
- The user starts the SERUPRO ASUP via function block FC 9
  in order to ramp up the spindle.
- The start after M0 in the ASUP does not change the spindle status.
- SERUPRO ASUP 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 ASUP.

Note

Modifications for REPOS of spindles:

The transitions of speed control mode and positioning mode must be taken into consideration in the event of modifications in SERUPRO approach and spindle functionality.

For further information on the operating mode switchover of spindles, see Section "Modes (Page 1274)".

10.8.5 Selfacting SERUPRO

Self-acting 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.

A special channel, the "serurpoMasterChan", can also be defined for **each** "Self-acting SERUPRO". A search target can be defined in this channel.

The "Self-acting SERUPRO" function supports the SERUPRO cross-channel block search.

Function

The "Self-acting SERUPRO" operation cannot be used to find a search target. If the search target is not reached, then no channel is stopped. However, in certain situations the channel is temporarily stopped. Generally, the channel waits for another channel. Examples are: Wait marks, couplings or axis interchange.

The wait phase occurs:

During this wait phase, the NC checks the "seruproMasterChan" channel whether it has reached a search target. The wait phase is exited when a search target is not reached.

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 the Reset state again.

There is no SERUPRO approach.

Starting a group of channels

If a group of channels is only started with "Self-acting SERUPRO", then all channels are ended with "RESET".

Exception:
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 selects a particularly important channel from the channel group for which a search target is to be explicitly selected (target channel).
- The HMI will then start SERUPRO on the target channel and "Self-acting SERUPRO" in the remaining channels of the channel group.

The operation is complete when **every** relevant channel has deleted "seruproActive".

"Self-acting 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".

A search target is not specified for the dependent channels started from the target channel.

10.8.6 Locking a program section for "Continue machining at the contour"

Programmed interrupt pointer

If because of manufacturing and/or process-related reasons, "Continue machining at the contour" may not be possible within a certain program section at a program abort, this program section can be locked for the target block of a block search.

If after a program abort there is a block search for the interruption point within the locked program section for "Continue machining at the contour", the last executable block (main run block) before the start of the locked section is used by the control as target block (hold block).

Programming

Syntax

`IPTRLOCK()`

Functionality

Marks the beginning of the program section as of which "Continue machining at the contour" is locked. The next executable block (main run block) in which `IPTRLOCK` becomes active is now used as target block for a block search with "Continue machining at the interruption point", until the release with `IPTRUNLOCK`. This block is referred to as the **hold block** in the following.

Effectiveness: Modal

Syntax

`IPTRUNLOCK()`

Marks the end of the program section locked for "Continue machining at the contour". As of the next executable block (main run block) in which `IPTRLOCK` becomes active, the current block is used again as target block for a block search with "Continue machining at the interruption point". This block is referred to as the **release block** in the following.

Effectiveness: Modal

Example

Program code	Comment
...	
N010 IPTRLOCK()	; Locked area: Start
N020 R1=R1+1	
N030 G4 F1	**; Hold block**
...	; Locked area
N200 IPTRUNLOCK()	; Locked area: End
N220 R1=R1+1	
N230 G4 F1	**; Release block**
...	

Supplementary conditions

- `IPTRLOCK` acts within a program (*.MPF, *.SPF) at the most up to the end of the program (`M30`, `M17`, `RET`). `IPTRUNLOCK` implicitly becomes active at the end of the program.
- Multiple programming of `IPTRLOCK` within a program does not have a cumulative effect. With the first programming of `IPTRUNLOCK` within the program or when the end of the program is reached, all previous `IPTRLOCK` calls are terminated.
- If there is a subprogram call within a locked area, "Continue machining at the contour" is also locked for this and all following subprogram levels. The lock also cannot be cancelled within the called subprogram through explicit programming of `IPTRUNLOCK`.

Example: Nesting of locked program sections in two program levels

With the activation of the "Continue machining at the contour" lock in PROG_1, "Continue machining at the contour" is also locked for PROG_2 and all following program levels.

Program code	Comment
PROC PROG_1	; Program 1
...	
N010 IPTRLOCK()	
N020 R1=R1+1	
N030 G4 F1	**; Hold block**
...	; Locked area: Start
N040 PROG_2	; Locked area
...	; Locked area: End
N050 IPTRUNLOCK()	

```
Program code                 Comment
N060 R2=R2+2
N070 G4 F1                   ; Release block
...
```

```
Program code                 Comment
PROC PROG_2                  ; Program 2
N210 IPTRLOCK()              ; Ineffective due to program 1
...
N250 IPTRUNLOCK()            ; Ineffective due to program 1
...
N280 RET                     ; Ineffective due to program 1
```

Example 3: Multiple programming of IPTRLOCK

```
Program code                 Comment
PROC PROG_1                  ; Program 1
...
N010 IPTRLOCK()
N020 R1=R1+1
N030 G4 F1                   ; Hold block
...                          ; Locked area: Start
N150 IPTRLOCK()              ; Locked area
...                          ; Locked area
N250 IPTRLOCK()              ; Locked area
...                          ; Locked area: End
N360 IPTRUNLOCK()
N370 R2=R2+2
N380 G4 F1                   ; Release block
...
```

System variable

The status of the current block can be determined via the system variable $P_IPTRLOCK:

$P_IPTRLOCK	Meaning
FALSE	The current block is **not** within a program section locked for "Continue machining at the contour"
TRUE	The current block is within a program section locked for "Continue machining at the contour"

Automatic function-specific "Continue machining at the contour" lock

For various couplings, the activation/deactivation of the "Continue machining at the contour" lock can be performed automatically channel-specifically with the activation/deactivation of the coupling:

MD22680 $MC_AUTO_IPTR_LOCK, bit x

Bit	Value	Meaning
0	Electronic gear (EGON / EGOF)	
	1	Automatic "Continue machining at the contour" lock is active
	0	Automatic "Continue machining at the contour" lock is not active
1	Axial master value coupling (LEADON / LEADOF)	
	1	Automatic "Continue machining at the contour" lock is active
	0	Automatic "Continue machining at the contour" lock is not active

This program section begins with the last executable block **before** the activation and ends with the deactivation.

The automatic interrupt pointer is not active for couplings that were activated or deactivated via synchronized actions.

Example: Automatically declaring axial master value coupling as search-suppressed:

```
Program code                       Comment
N100 G0 X100
N200 EGON(Y,"NOC",X,1,1)           ; Search-suppressed program section starts.
N300 LEADON(A,B,1)
...
N400 EGOFS(Y)
...
N500 LEADOF(A,B)                   ; Search-suppressed program section ends.
N600 G0 X200
```

A program abort within search-suppressed program section (N200 - N500) always provides the interrupt pointer with N100.

NOTICE

Unwanted state caused by function overlappings

If there is an overlap of the "Programmable interrupt pointer" and "Automatic interrupt pointer" functions via machine data, the NC selects the largest possible search-suppressed area.

A program may need a coupling for almost all of the runtime. In this case, the automatic interrupt pointer would always point to the start of the program and the SERUPRO function would in fact be useless.

10.8.7 Behavior during POWER ON, mode change and RESET

SERUPRO is inactive at Power On. The operating mode change is permitted during SERUPRO. RESET aborts SERUPRO, the internally selected program test is deselected again. SERUPRO cannot be combined with other block search types.

10.8.8 Supplementary conditions

10.8.8.1 STOPRE in the target 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: 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: Read and correctly calculate external work offset.

```
N10 G1 X1000 F100
N20 G1 X1000 F500
N30 G1 X1000 F1000
N40 G1 X1000 F5000
N50 SUPA G1 F100 X200            ; move external work offset to 200
N60 G0 X1000
N70 ...
```

Via an implicit STOPRE before N50, the NC can read and correctly calculate the current work offset.
For a SERUPRO operation on the N50 search target, repositioning is on the implicit STOPRE in the SERUPRO approach and the velocity is determined from N40 with F5000.

Implicit preprocessing stop

Situations in which an implicit preprocessing stop is issued:

1. In all blocks in which one of the following variable access operations occurs:
   – Programming of a system variable beginning with $A...
   – Programming of a system variable beginning with attribute SYNR/SYNRW
2. For the following commands:
   – Part program MEACALC, MEASURE commands
   – Programming of SUPA (suppress frame and online corrections)
   – Programming of CTABDEF (start of the curve table definition)
   – Part program WRITE/DELETE command (write/delete file)
   – Before the first WRITE/DELETE command of a sequence of such commands
   – Part program EXTCALL command
   – Part program GETSELT, GETEXET commands
   – For tool and active FTOCON tool fine correction
3. For the following command processing:
   – End processing of a type 1 block search ("block search without calculation")
   – End processing of a type 2 block search with calculation ("block search at contour end point")

10.8.8.2 SPOS in target block

If a spindle is programmed with M3/M4 and a switch made to SPOS in the target block, the spindle is switched to SPOS at the end of the SERUPRO operation ("Search target found" status).

DB31, ... DBX84.5 = TRUE (active spindle mode: positioning mode)

10.8.8.3 Travel to fixed stop (FXS)

During repositioning (REPOS), the "Travel to fixed stop" function (`FXS`) is repeated automatically. Every axis is taken into account. The torque programmed last before the search target is used as torque.

System variable

The system variables for "Travel to fixed stop" have the following meaning with SERUPRO:

- $AA_FXS: Progress of the program simulation
- $VA_FXS: Real machine state

The two system variables always have the same values **outside** the SERUPRO function.

ASUP

A user-specific ASUP can be activated for SERUPRO.

References

For detailed information on the SERUPRO block search, see Section "Detailed description (Page 318)".

10.8.8.4 Travel with limited torque/force (FOC)

During repositioning (REPOS), the "Travel with limited torque/force" function (`FOC`) is repeated automatically. Every axis is taken into account. The torque programmed last before the search target is used as torque.

System variable

The system variables for "Travel with limited torque/force" have the following meaning with SERUPRO:

- $AA_FOC: Progress of the program simulation
- $VA_FOC: Real machine state

Supplementary condition

A changing torque characteristic **cannot** be implemented during repositioning.

Example

A program traverses axis X from 0 to 100 and switches "Travel with limited torque/force" (`FOC`) on every 20 increments for 10 increments. This torque characteristic is usually generated with non-modal `FOC` and cannot be performed during repositioning (REPOS). Instead, axis X is traversed from 0 to 100 with or without limited torque/force in accordance with the last programming.

References

For detailed information on the SERUPRO block search, see Section "Detailed description (Page 318)".

10.8.8.5 Synchronous spindle

The synchronous spindle can be simulated.

The synchronous spindle operation with main spindle and any number of following spindles can be simulated in all existing channels with SERUPRO.

For further information about synchronous spindles, see:

References:

Function Manual, Extended Functions; Synchronous Spindle (S3)

10.8.8.6 Couplings and master-slave

Setpoint and actual value couplings

The SERUPRO operation is a program simulation in Program Test mode with which setpoint and actual value couplings can be simulated.

Specifications for EG simulation

For simulation of EG, the following definitions apply:

1. Simulation always takes place with setpoint coupling.
2. If not all leading axes are under SERUPRO, the simulation is aborted with Alarm 16952 "Reset Clear/No Start". This can occur with cross-channel couplings.
3. Axes that have only one encoder from the NC point of view and are moved externally, cannot be simulated correctly. These axes must not be integrated in couplings.

⚠ CAUTION

Incorrect simulation

In order to be able to simulate couplings correctly, they must have been switched off previously.

This can be performed with machine data MD10708 $MA_SERUPRO_MASK.

Specifications for coupled axes

The SERUPRO operation simulates coupled axes always assuming that they are setpoint couplings. In this way, the end points are calculated for **all** axes that are used as target points for SERUPRO approach. The coupling is also active with "Search target found". The path from the current point to the end point is carried out for SERUPRO approach with the active coupling.

LEADON

The following specifications apply for the simulation of axial master value couplings:

1. Simulation always takes place with setpoint coupling.
2. SERUPRO approach takes place with active coupling and an overlaid motion of the following axis in order to reach the simulated target point.

The following axis that is moved solely by the coupling cannot always reach the target point. In SERUPRO approach, an overlaid linear motion is calculated for the following axis to approach the simulated point!

Reaching simulated target point for LEAD with JOG

At the time of "Search target found", the coupling is already active, especially for the JOG motions. If the target point is not reached, SERUPRO approach can be used to traverse the following axis with active coupling and an overlaid motion to the target point.

Note

For further information on the repositioning of axis couplings, see Section "Repositioning to the contour (REPOS) (Page 521)".

Master-slave

A system ASUP can be started automatically after the block search has finished. In this ASUP, the user can control the coupling state and the associated axis positions subsequently. The required information is provided via the following system variables:

System variable	Description
$P_SEARCH_MASLD[<slave axis>]	Position offset between slave and master axis when the link is closed.
$AA_MASL_STAT[<slave axis>]	Current state of a master/slave coupling
$P_SEARCH_MASLC[<slave axis>]	Status: The state of the coupling was changed during the block search
The system variables are deleted when the coupling is switched on with `MASLON`.	

Note

The coupled axes must be in the same channel when the block search is executed.

References

Further information on the master-slave coupling can be found in:

Function Manual, Special Functions; Section "TE3: Speed/torque coupling, master-slave"

Example

- **System ASUP**
  - Path and name: /_N_CMA_DIR/PROGEVENT.SPF
  - Master axis: X
  - Slave axis: Y

Program code	
`PROG PROGEVENT`	
`   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`	
`   ...`	
`RET`	

- **Machine data**
  To ensure that the ASUP starts automatically, the following machine data must be set:
  - NC-specific machine data:
    - MD11604 $MN_ASUP_START_PRIO_LEVEL = 100
    - MD11450 $MN_SEARCH_RUN_MODE = 'H02'
  - Channel-specifically for the channel in which the ASUP is started or generally for all channels:
    - MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK, bit<n> = TRUE
      n: For all required event-driven program calls (prog events)
    - MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit<n> = TRUE
      n: For all **required user** interrupts

NOTICE

System interrupts

MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bits 8 to 31 enable the system interrupts.

Bit 8 / interrupt 9 starts an ASUP that contains traversing motions.

Axis couplings

- Acceleration of the processing speed and leading axis and following axes in different channels
  For a leading axis whose following axes are assigned to a different channel than that of the leading axis, the setting for acceleration of the execution speed has no effect (MD22601 (Page 529)$MC_SERUPRO_SPEED_FACTOR):
- Coupled motion
  The coupled motion function (`TRAILON`) is supported by SERUPRO.
  For further information on coupled motion with `TRAILON` and `TRAILOF`, see:
  References:
  - Function Manual, Special Functions; Axis Couplings (M3)
  - Programming Manual, Job Planning; Section "Axis couplings"
- Gantry axes
  The gantry axis function is supported by SERUPRO.
  For further information on the functionality of gantry axes, see:
  References:
  Function Manual, Special Functions; Section "G1: Gantry axes"
- Tangential control
  The tangential follow-up of individual axes function is supported by SERUPRO.
  Additional information about the tangential control can be found in:
  References:
  Function Manual, Special Functions; Section "T3: Tangential control"

10.8.8.7 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 → NC
DB31, ... DBX3.7 (program test axis/spindle enable). If the real servo enable is missing during program test or SERUPRO, the effect on the axes/spindles is as follows:

- As soon as the simulated program run intends to move an axis/spindle, the message "Waiting for axis enable" or "Waiting for spindle enable" is displayed and the simulation is stopped.
- If during a simulated motion, NC/PLC interface signal DB31, ... DBX3.7 (program test axis/spindle enable) is then canceled, alarm 21612: "Channel %1 axis %2 NC/PLC interface signal 'controller enable' reset during motion" is activated.

Autonomous axis operations

Autonomous single-axis operations are axes controlled by the PLC that can also be simulated on SERUPRO. During SERUPRO operation, as in normal operation, the PLC can take over or give up control of an axis. If required, this axis can also be traversed using FC18. The PLC takes over control of the axis **before** the approach block and is responsible for positioning this axis. This is valid for all block search types.

For further information about autonomous single-axis operations, see:

References:
Function Manual, Extended Functions; Positioning Axes (P2)

Axis replacement

Problem: A program moves an axis and gives up control before the target block with WAITP(X). X is thus not subject to REPOS and the axis is not taken into account in SERUPRO approach.

Via the machine data MD11470 $MN_REPOS_MODE_MASK, the following behavior can be achieved for SERUPRO-REPOS:

The neutral axes are moved as "command axes" in the SERUPRO-REPOS. The axis interpolates without a path context even if it was last programmed as a path axis. In this scenario, the velocity results from MD32060 $MA_POS_AX_VELO. After SERUPRO approach, this axis is again neutral.

Neutral axes that are however not allowed to be repositioned must receive the axial NC/PLC interface signal "REPOSDELAY". This deletes the REPOS movement.

Example:

After SERUPRO, one axis is deliberately moved in the synchronized action via technology cycles. The command axes are always moved in the approach block, never in the target block. The target block can only be changed if all command axes have been moved to the end.

⚠ CAUTION

The PLC-controlled axis is not repositioned

Axes enabled by RELEASE(X) before the target block are not repositioned.

10.8.8.8 Gear stage change

Operational sequences

The gear stage change (GSC) requires physical motions from the NC in order to be able to engage a new gear.
In the SERUPRO operation, no gear stage change is required and is carried out as follows:

Some gears can only be changed when controlled by the NC, since either the axis must oscillate or a certain position must be approached beforehand.

The gear stage change can be suppressed selectively for DryRun, program test, and SERUPRO using bits 0 to 2 in MD35035 $MA_SPIND_FUNCTION_MASK.

The gear stage change must then be performed in REPOS; this will work even if the axis involved is to be in "speed control mode" at the target block. In other cases, the automatic gear stage change is denied with an alarm if, for example, the axis was involved in a transformation or coupling between the gear stage change and the target block.

Note

For further information on gear stage changes in DryRun, Program test and SERUPRO, see Section "S1: Spindles (Page 1273)".

10.8.8.9 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.

10.8.8.10 NC/PLC interface signals

REPOS offset available

If a REPOS offset has resulted for an axis during SERUPRO, this is displayed via the axial NC/PLC interface at the end of the SERUPRO operation:

DB31, ...DBX70.0 == 1 (REPOS offset available)

Validity of the REPOS offset

The REPOS offset becomes invalid at the start of a SERUPRO ASUB or NC start to resume the machining:

DB31, ... DBX70.1 == 1 (REPOS offset invalid)

The axis can be traversed manually in JOG mode or via the PLC user program using FC 18 between the end of the SERUPRO operation and NC start to resume the machining. If the REPOS offset is traversed completely, the interface signal is reset.

10.8.8.11 Making the initial settings more flexible

Basic setting / basic SERUPRO setting

Machine data MD20112 $MC_START_MODE_MASK defines the basic 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 a basic setting that differs from the normal part program start. The new setting must therefore be stored in the machine data:

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 cou-
                                             pling not configured
$MC_START_MODE_MASK = 'H400'                 ; will be 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
```

10.8.8.12 Compressor functions (COMPON, COMPCURV, COMPCAD)

- If the target block for block search **type 2** or **type 4** (block search **with calculation** to ...) is in a program section in which a compressor function (`COMPON`, `COMPCURV`, `COMPCAD`) is active, positions are approached on the path calculated by the **compressor** on repositioning. These positions must precisely match the positions on the path programmed in the part program.
- If programmed blocks are eliminated from the part program during compression, these blocks will not be found at the target block in the block search ⇒ Alarm 15370 "Search target not found."

10.8.9 System variable

Overview of the system variables relevant for SERUPRO:

System variable	Meaning
$AC_ASUP, bit 20	ASUP activation reason: $AC_ASUP, Bit 20 == 1 ⇒ system ASUP active, reason: SERUPRO search goal reached
$AC_SERUPRO	SERUPRO status: $AC_SERUPRO == 1 ⇒ SERUPRO is active
$P_ISTEST	Program test status: SERUPRO active ⇒ $P_ISTEST == 1
$P_SEARCHL	Most recently active block-search type: $P_SEARCHL == 5 from the start of SERUPRO to reset or end of program
$AC_REPOS_PATH_MODE	REPOS mode for repositioning the contour after a SERUPRO

10.9 Program operation

Definition

Program mode is present when in the AUTOMATIC or MDI operating modes, NC programs or PC program blocks are processed.

NC/PLC interface signals

The program mode can be influenced by the PLC user program via mode group- and channel-specific NC/PLC interface signals or returns the appropriate feedbacks to the PLC user program.

A function-specific overview of the NC/PLC interface signals can be found in:

References

- Function Manual, Basic Functions; Section "Z1: NC/PLC interface signals
- Function Manual, Extended Functions; Section "Z2: NC/PLC interface signals
- Function Manual, Special Functions; Section "Z3: NC/PLC interface signals

An overview of all NC/PLC interface signals can be found in:

References

List Manual, NC Variable and Interface Signals; Section "Interface signals - Overview".

10.9.1 Initial settings

Basic settings can be programmed in channel-specific machine data for each channel. These basic settings affect, for example, G groups and auxiliary function output.

Auxiliary function output

The timing for output of auxiliary functions can be predefined via machine data AUXFU_x_SYNC_TYPE (MD22200, 22210, 22220, 22230, 22240, 22250, 22260), (output timing for M, S, T, H, F, D, E functions). For more detailed explanations, see Section "H2: Auxiliary function outputs to PLC (Page 401)".

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 basic setting is automatically active during program start or in Reset until it is deselected by a G command from the same G group.

Via MD22510 $MC_GCODE_GROUPS_TO_PLC (G commands which are output to interface NC-PLC after block change / RESET), the output of the G commands to the PLC interface can be activated.

References

A list of G groups with the associated G commands can be found in:

Programming Manual, Fundamentals

Basic configurations of the NC language scope for SINUMERIK solution line

For SINUMERIK 840D sl, certain basic configurations of the NC language scope can be generated (configurable) via machine data. The options and functions of the NC language scope is specially tailored (configured) to the needs of the user.

10.9.1.1 Machine data

NC language scope

The way that non-active options and functions should be handled with language commands can be set with the following machine data:

MD10711 $MN_NC_LANGUAGE_CONFIGURATION = <value>

Value	Meaning
0	All language commands are available. Whether or not the needed function is activated can only be recognized upon execution.
1	All language commands are available. Language commands for non-enabled options are recognized during the program interpretation ⇒ Alarm 12553 "Option/function is not active".
2	Only the language commands of the enabled options are available. Language commands for non-enabled options are recognized during the program interpretation ⇒ Alarm 12550 "Name not defined or option/function available".

Value	Meaning
3	All language commands are available. Language commands of not activated functions are recognized during the program interpretation ⇒ Alarm 12553 "Option/function is not active". **Example:** 1. Option set for the cylinder surface transformation. 2. The cylinder surface transformation in the machine data MD24100 $MC_TRAOF_TYPE_1 is **not** activated 3. Alarm 12553 is issued already when programming the `TRACYL` command.
4	Only the language commands of the active functions are available. Language commands of non-active functions are not recognized ⇒ Alarm 12550 "Name not defined or option/ function not available". **Note** Whether the associated language commands are generally unavailable in the Siemens language or whether this is true only on the corresponding system cannot be distinguished in this scenario.

10.9.1.2 Programming

The function "STRINGIS(...)" checks whether the specified character string is available as element of the NC programming language in the actual language scope.

The following elements of the NC programming language can be checked:

- G commands of all existing G groups
- DIN or NC addresses
- Functions
- Procedures
- Keywords
- System data, such as machine data $M... , setting data $S... or option data $O...
- System variables $A... , $V... , $P...
- Arithmetic parameters R...
- Cycle names of activated cycles
- GUD and LUD variables
- Macro names
- Label names

Definition

```
INT STRINGIS(STRING <name>)
```

Syntax

```
<return value> = STRINGIS(<name>)
```

Meaning

`STRINGIS():`	Test function with return value
`<name>:`	Character string to be tested

`<return value>:`	The return value is coded in the first three decimal places `yxx`		
	000	The specified string is not an element of the current language scope [1].	
	100	The specified string is an element of the NC programming language, but currently cannot be programmed (option/function is inactive).	
	2xx	The specified string is a programmable element of the current language scope (option/function is active). Detailed information is contained in the first and second decimal places:	
		xx	**Meaning**
		01	DIN address or NC address[2]
		02	G command (e.g. G04, INVCW)
		03	Function with return value
		04	Function without return value
		05	Keyword, e.g. DEFINE
		06	Machine ($M...), setting ($S...) or option data ($O...)
		07	System parameter, e.g. system variable ($...) or arithmetic parameter (R...)
		08	Cycle (the cycle must be loaded to the NC and the cycle program must be active [3])
		09	GUD variable (the GUD variable must be defined in the GUD definition files and the GUD variables activated)
		10	Macro name (the macro must be defined in the macro definition files and macros activated) [4]
		11	LUD variable of the actual part program
		12	ISO G command (ISO language mode must be active)
	400	The specified string is an NC address that was not identified as DIN address or NC address (xx==01) or macro name (xx==10), and is not G or R [2]	
	y00	No specific assignment possible	

1) Depending on the control, under certain circumstances, only a subset of the Siemens NC language commands are known, e.g. SINUMERIK 802D sl. For these controls, for strings that are principally Siemens NC language commands, a value of 0 is returned. This behavior can be changed using MD10711 $MN_NC_LANGUAGE_CONFIGURATION. If MD10711 = 1, then a value of 100 is always returned for Siemens NC language commands.

2) NC addresses are the following letters: A, B, C, E, I, J, K, Q, U, V, W, X, Y, Z. These NC addresses can also be programmed with an address extension. The address extension can be specified when checking with STRINGIS. Example: 201 == STRINGIS("A1").
The letters: D, F, H, L, M, N, O, P, S, T are NC addresses or auxiliary functions that are defined by the user. A value of 400 is always returned for these. Example: 400 == STRINGIS("D"). These NC addresses cannot be specified with address extension when checking with STRINGIS.
Example: 000 == STRINGIS("M02"), but 400 == STRINGIS("M").

3) Names for cycle parameters **cannot** be checked with STRINGIS.

4) Address defined as macro, e.g. G, H, M, L, is identified as macro.

Examples

In the following examples it is assumed that the specified NC language element, unless noted otherwise, can in principle be programmed in the control.

1. String "T" is defined as auxiliary function:
   `400 == STRINGIS("T")`
   `000 == STRINGIS ("T3")`
2. String "X" is defined as axis:
   `201 == STRINGIS("X")`
   `201 == STRINGIS("X1")`
3. String "A2" is defined as NC address with extension:
   `201 == STRINGIS("A")`
   `201 == STRINGIS("A2")`
4. String "INVCW" is defined as named G command:
   `202 == STRINGIS("INVCW")`
5. String "$MC_GCODE_RESET_VALUES" is defined as machine data:
   `206 == STRINGIS("$MC_GCODE_RESET_VALUES")`
6. String "GETMDACT" is an NC language function:
   `203 == STRINGIS("GETMDACT ")`
7. String "DEFINE" is a keyword:
   `205 == STRINGIS("DEFINE")`
8. String "$TC_DP3" is a system parameter (tool length component):
   `207 == STRINGIS("$TC_DP3")`
9. String "$TC_TP4" is a system parameter (tool size):
   `207 == STRINGIS("$TC_TP4")`
10. String "$TC_MPP4" is a system parameter (magazine location status):
    - Tool magazine management is active: `207 == STRINGIS("$TC_MPP4") ;`
    - Tool magazine management is not active: `000 == STRINGIS("$TC_MPP4")`

    See also the paragraph below: Tool magazine management.
11. String "MACHINERY_NAME" is defined as GUD variable:
    `209 == STRINGIS("MACHINERY_NAME")`
12. String "LONGMACRO" is defined as macro:
    `210 == STRINGIS("LONGMACRO")`
13. String "MYVAR" is defined as LUD variable:
    `211 == STRINGIS("MYVAR")`
14. String "XYZ" is a command that is not known in the NC, GUD variable, macro or cycle name:
    `000 == STRINGIS("XYZ")`

Supplementary conditions

Tool magazine management

If the tool magazine management function is not active, `STRINGIS()` supplies the system parameters of the tool magazine management, independent of the machine data

- MD10711 $MN_NC_LANGUAGE_CONFIGURATION

always a value of 000.

ISO Mode

If the "ISO Mode" function is active:

`STRINGIS()` checks the specified string initially as SINUMERIK G command:

- MD18800 $MN_MM_EXTERN_LANGUAGE (activation of external NC languages)
- MD10880 $MN_ MM_EXTERN_CNC_SYSTEM (control system to be adapted)

For active "ISO Mode", `STRINGIS()` checks the specified character string first whether it belongs to the SINUMERIK G command.

If the string is not a SINUMERIK G command, it is subsequently checked whether it is an ISO G command.

Programmed switchover of the ISO Mode

Programmed switchovers with the `G290` (SINUMERIK mode) and `G291` (ISO Mode) commands have no effect on `STRINGIS()`.

10.9.2 Selection and start of an NC program

NC/PLC interface signals

Selection

An NC program can be selected only if the relevant channel is in the reset status.

- DB21, ... DBX35.7 == 1 (reset)

Start

An NC program is started by two different events:

1. DB21, ... DBX7.1 = 1 (NC start), the signal is normally initiated by pressing the "NC start" key on the machine control panel.
2. `START` command in an NC program of another active channel. The channel must be in the AUTOMATIC or MDI mode and in the "reset" or "interrupted" status.
   - DB21, ... DBX35.7 == 1 (reset)
   - DB21, ... DBX35.6 == 1 (interrupted)

Initial conditions

An NC program can be started only when the following initial conditions are satisfied.

- DB11 DBX4.4 == 1 (mode group ready)
- DB11 DBX0.7 == 0 (mode group reset)
- DB21, ... DBX1.7 == 0 (activate program test)
- DB21, ... DBX7.0 == 0 (NC start lock)
- DB21, ... DBX7.2 == 0 (NC stop at block limit)
- DB21, ... DBX7.3 == 0 (NC stop)
- DB21, ... DBX7.4 == 0 (NC stop, axes plus spindles)
- DB21, ... DBX7. 7 == 0 (reset)
- DB10 DBX56.1 == 0 (emergency stop)
- **No** axis or NC-specific alarm may be pending

Execution of command

The part program or the part program block is automatically executed and the following interface signals are set:

- DB21, ... DBX35.5 (channel status reset)
- DB21, ... DBX35.0 (program status running)

The program is processed until the end of the program has been reached or the channel is interrupted or aborted by a STOP or RESET command.

References

A detailed description of the interface signals can be found in the NC Variables and Interface Signals List Manual.

Alarms

Under certain conditions the START command will have no effect and one of the following alarms will be triggered:

- 10200 "No NC Start permitted with active alarm"
- 10202 "No NC Start permitted with active command"
- 10203 "No NC Start permitted for non-referenced axes"

References:
Diagnostics Manual, Alarms

10.9.3 Program interruption

NC/PLC interface signals

Requirements

A program interruption is performed only when the channel and the NC program are active:

- DB21, ... D35.5 == 1 (channel: active)
- DB21, ... D35.0 == 1 (program: running)

Program interruption

The program processing can be interrupted by the following events:

- DB21, ... DBX7.2 == 1 (NC stop at block limit)
- DB21, ... DBX7.3 == 1 (NC stop)
- DB21, ... DBX7.4 == 1 (NC stop, axes plus spindles)
- DB21, ... DBX2.0 == 1 (single block)
- Programmed `M00` or `M01` command in the processed NC program

The channel and the NC program are then in the "interrupted" status:

- DB21, ... D35.6 == 1 (channel interrupted)
- DB21, ... D35.3 == 1 (program interrupted)

References

A detailed description of the interface signals can be found in the NC Variables and Interface Signals List Manual.

Sequence

The following actions are performed when a program interruption is detected:

- Interrupt the program processing:
  - At the next block limit for the following events: "NC stop at block limit", M00, M01 or single block
  - Immediately: All other events
- The traversing axes of the channel are stopped via a braking ramp. The braking of the axes can extend over several blocks.
- The block indicator shows the current block at the point of interruption.
- The auxiliary functions that have not been output before the point of interruption are no longer output.

Possible actions in the interrupt state

Various functions can be performed in the channel during a part program interruption, for example:

- **Overstoring**
  References
  Operating Manual for SINUMERIK Operate, Section "Machine workpiece" > "Overstoring"
- **Block search**
  References
  Function Manual, Basic Functions; Section "Mode group, channel, program operation, reset response (K1)" > "Block search" or "Block search type 5 SERUPRO"
- **Repositioning to the contour (REPOS)**
  References
  Function Manual, Basic Functions; Section "Mode group, channel, program mode, reset response (K1)" > "Block search type 5 SERUPRO" > "REPOS" > "Repositioning with controlled REPOS"
- **Oriented tool retraction**
  References
  - Programming Manual, Job Planning; Section "Tool offsets"
  - Description of Functions, Basic Functions; Section "Tool offsets (W1)" > "Orientable toolholders" > ""
- **ASUP** (see Section "Asynchronous subprograms (ASUPs) (Page 587)").
- **DRF function, offset of the workpiece zero**
  References
  Function Manual, Extended Functions; Manual traversing and manual handwheel traversing (H1)
- **Continue the interrupted NC program**
  - `START` command from another channel
    References
    Programming Manual, Job Planning; Section "Flexible NC programming" > "Program coordination (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)"
  - NC/PLC interface:
    DB21, ... DBX7.1 = 1 (NC start)

10.9.4 Channel reset

Function

A channel reset causes an NC program being processed in automatic mode or program block being processed in the MDI mode to be terminated.

The NC program or the program block cannot be continued at the point of interruption. After completion of the channel reset, all axes and spindles of the channel are in the "exact stop" status (exception: follow-up mode).

Sequence

The NC reset causes the following actions to be performed in the channel:

- The program preparation is stopped immediately.
- Axes and spindles are braked via their parameterized braking ramps to standstill.
- Any auxiliary functions of the current block not yet output are no longer output to the PLC.
- The block indicator is reset to the start of the selected NC program.
- All reset alarms of the channel are cleared from the display.

Rules

- A channel reset is performed in any channel state.
- A channel reset is not aborted by any other command.

NC/PLC interface signals

Request: Channel reset

A channel reset is requested via the following NC/PLC interface signals:

- DB21, … DBX7.7 = 1 (reset)

Request: Mode group reset

A mode group reset initiates a channel reset in all channels of a mode group.

A mode group reset is requested via the following NC/PLC interface signals:

- DB11, ... DBX0.7 = 1 (mode group reset)

Feedback: Channel reset completed

- DB21, … DBX35.7 == 1 (channel state reset)

Feedback: Mode group reset completed

- DB11, ... DBX6.5 == 1 (mode group reset)

References

A detailed description of the interface signals can be found in the NC Variables and Interface Signals List Manual.

10.9.5 Program status

The status of the selected NC program is displayed in the interface for each channel.

All program states can occur in the AUTOMATIC and MDI modes. In all other modes or machine functions, the program status is aborted or interrupted.

NC/PLC interface signals

The following program states are displayed at the NC/PLC interface (DB21, ...):

- DB21, ... DBX35.4 ("aborted")
- DB21, ... DBX35.3 ("interrupted")
- DB21, ... DBX35.2 ("stopped")
- DB21, ... DBX35.1 ("waiting")
- DB21, ... DBX35.0 ("running")

A detailed description of the interface signals can be found in the NC Variables and Interface Signals List Manual.

Status changes

The program status is influenced by commands, NC/PLC interface and alarms. Starting at the "running" program status, the table shows the following status based on the associated event.

Initial status of the program: "running"					
Event	Following status of the program				
	Abor-ted	Inter-rupted	Stop-ped	Wait-ing	Run-ning
DB21, ... DBX7.7 (reset)	x				
DB21, ... DBX7.3 (NC stop)			x		
DB21, ... DBX7.2 (NC stop at the block limit)			x		
DB21, ... DBX7.4 (NC stop, axes and spindles)			x		
DB21, ... DBX6.1 (read-in disable)					x
DB21, ... DBX6.0 (feedrate stop)					x
DB21, ... DBX12.3 / 16.3 / 20.3 (feedrate stop, geo axis 1 / 2 / 3)					x
DB21, ... DBB4; feedrate override = 0%					x
DB31, ... DBX4.3 (feed/spindle stop)					x
DB21, ... DBX194.2 / DBX197.6 (M02 / M30 in the block)	x				
DB21, ... DBX194.0 / DBX194.1 (M00 / M01 in the block)			x		
DB21, ... DBX0.4 (single block)			x		
DB21, ... DBX6.2 (delete distance-to-go)					x
Auxiliary functions output to PLC but not yet acknowl-edged			x		
WAIT command in the program				x	
Alarm with "NOREADY" system response		x			

10.9.6 Channel status

The channel status for each channel is displayed in all operating modes at the NC/PLC interface (DB21, ...).

NC/PLC interface signals

The channel states are indicated by the following signals in the NC/PLC interface:

- DB21, ... DBX35.7 ("reset")
- DB21, ... DBX35.6 ("interrupted")
- DB21, ... DBX35.5 ("active")

A detailed description of the interface signals can be found in the NC Variables and Interface Signals List Manual.

Status changes

The channel status is influenced by commands and NC/PLC interface signals. Starting at the "active" channel status, the table shows the following status based on the associated event.

Initial status of the channel: "active"			
Event	**Following status of the channel**		
	"Reset"	**"Interrupted"**	**"Active"**
DB21, ... DBX7.7 (reset)	x		
DB21, ... DBX7.3 (NC stop)		x	
DB21, ... DBX7.2 (NC stop at the block limit)		x	
DB21, ... DBX7.4 (NC stop, axes and spindles)		x	
DB21, ... DBX6.1 (read-in disable)			x
DB21, ... DBX6.0 (feedrate stop)			x
DB21, ... DBX12/16/20.3 (feedrate stop, geometry axis 1/2/3)			x
DB21, ... DBB4; feedrate override = 0%			x
DB31, ... DBX4.3 (feed/spindle stop)			x
DB21, ... DBX194.2/DBX197.6 (M02/M30 in the block)	x		
DB21, ... DBX194.0/DBX194.1 (M00/M01 in the block)		x	
DB21, ... DBX0.4 (single block)		x	
DB21, ... DBX6.2 (delete distance-to-go)			x
Auxiliary functions output to PLC but not yet acknowl-edged			x
WAIT command in the program			x

The "active" channel status is achieved when an NC program or NC program block is being executed or when axes are traversed in JOG mode.

10.9.7 Responses to operator and program actions

Status transitions

The following table shows the channel and program states that result after certain operator and program actions.

The left-hand part of the table lists the various states of the channel and of the program selected in the channel, and the active operating mode.

The right-hand part of the table lists the operating/program actions and the following status.

Status	Channel status			Program status					Operating mode			Operating or program action => following status
	R	U	A	R	U	S	W	A	A	M	J	
1		x						x	x			RESET ⇒ 4
2		x						x		x		RESET ⇒ 5
3		x						x			x	RESET ⇒ 6
4	x			x					x			NC start ⇒ 13; mode change ⇒ 5 or 6
5	x			x						x		NC start ⇒ 14; mode change ⇒ 4 or 6
6	x			x							x	Direction key ⇒ 15; mode change ⇒ 4 or 5
7		x		x						x		NC start ⇒ 14
8		x		x							x	NC start ⇒ 15
9		x			x				x			NC start ⇒ 13; mode change ⇒ 10 or 11
10		x			x					x		NC start ⇒ 16; mode change ⇒ 9 or 11
11		x			x						x	Direction key ⇒ 17; mode change ⇒ 9 or 10
12		x				x			x			NC start ⇒ 13; mode change ⇒ 10 or 11
13			x					x	x			NC stop ⇒ 12
14			x	x						x		NC stop ⇒ 7; for block end ⇒ 5
15			x	x							x	NC stop ⇒ 8; for JOG end ⇒ 6
16			x		x					x		NC stop ⇒ 10; for block end ⇒ 10
17			x		x						x	NC stop ⇒ 11; for JOG end ⇒ 11
18			x				x		x			Reset ⇒ 4; wait for other channel ⇒ 18

Channel status	**Program status**	**Modes**
R → aborted	R → aborted	A → automatic
U → interrupted	U → interrupted	M → MDI
A → running	S → stopped	J → JOG
	W → waiting	
	A → running	

10.9.8 Example of a timing diagram for a program run

Program code

```
N10 G01 G90 X100 M3 S1000 F1000 M170
N20 M0
```

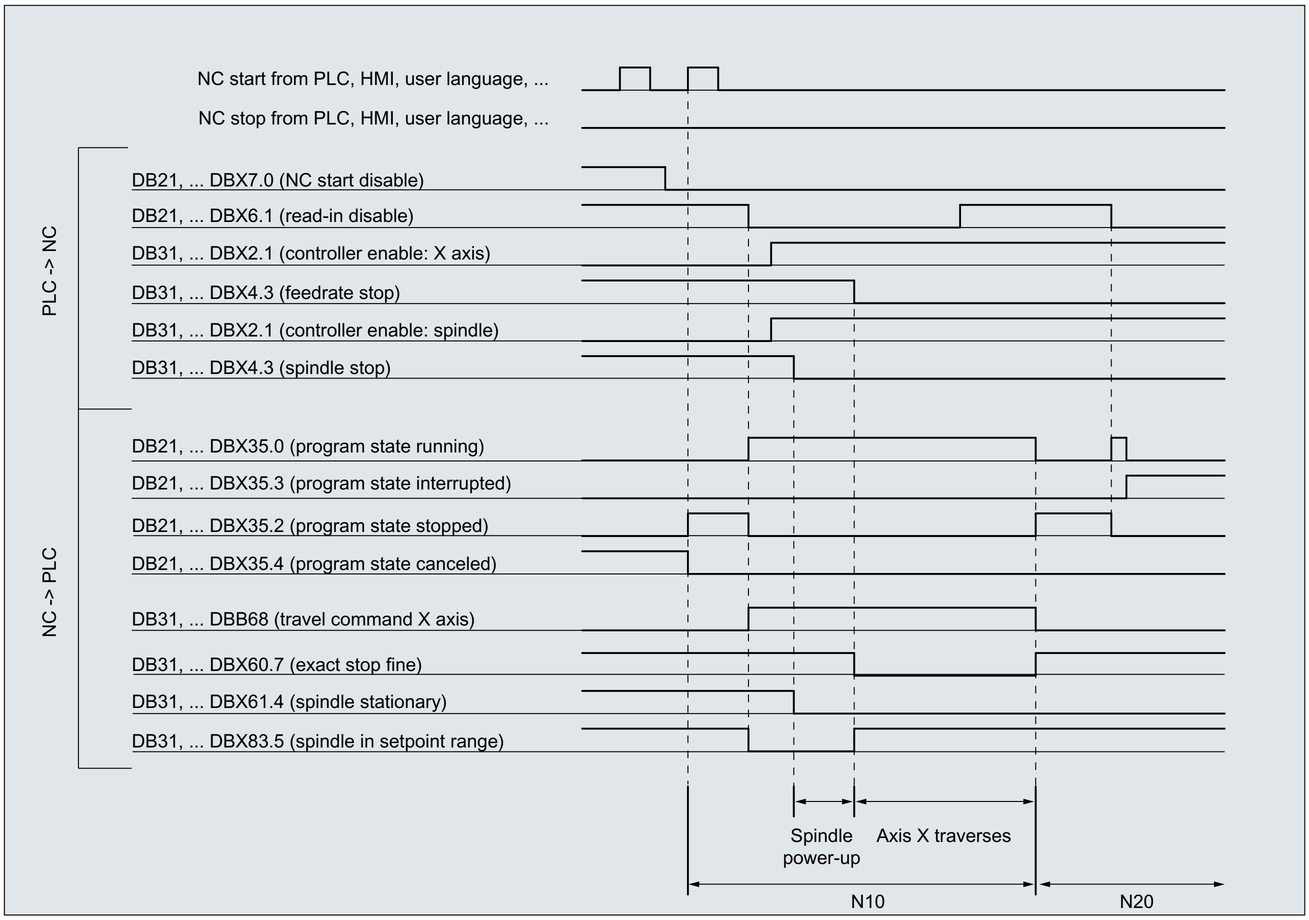

NC start from PLC, HMI, user language, ...
NC stop from PLC, HMI, user language, ...
PLC -> NC
DB21, ... DBX7.0 (NC start disable)
DB21, ... DBX6.1 (read-in disable)
DB31, ... DBX2.1 (controller enable: X axis)
DB31, ... DBX4.3 (feedrate stop)
DB31, ... DBX2.1 (controller enable: spindle)
DB31, ... DBX4.3 (spindle stop)
NC -> PLC
DB21, ... DBX35.0 (program state running)
DB21, ... DBX35.3 (program state interrupted)
DB21, ... DBX35.2 (program state stopped)
DB21, ... DBX35.4 (program state canceled)
DB31, ... DBB68 (travel command X axis)
DB31, ... DBX60.7 (exact stop fine)
DB31, ... DBX61.4 (spindle stationary)
DB31, ... DBX83.5 (spindle in setpoint range)
Spindle power-up
Axis X traverses
N10
N20

Figure 10-6 Signal characteristics during the program

10.9.9 Program jumps

10.9.9.1 Return jump to the start of the program (GOTOS)

Function

With the function "Jump back to start of the program" the control jumps back from a part program to the beginning of the program. The program is then processed again.

As compared to the function "Program jumps to jump marks", with which a repeated processing of the program can also be implemented, the function "Jump back to the start of the program" offers the following advantages:

- The programming of a jump mark at the start of the program is not necessary.
- The program restart can be controlled through the NC/PLC interface signal:
  DB21, ... DBX384.0 (control program branching)

- The timer for the program runtime can be reset to "0" at the restart of the program.
- The timer for workpiece counting can be incremented by "1" at program restart.

Application example

The function is used, if the processing of subsequent workpieces is to be done through an automatic program restart, e.g. in case of turning machine with bar loader/changer.

NC/PLC interface signals

The jump back takes place only when the following NC/PLC interface signal is set:

DB21, ... DBX384.0 (control program branching) = **1**

Parameterization

Program runtime

The runtime of the selected NC program is stored in the system variable $AC_CYCLE_TIME. When starting a new program, the system variable is automatically reset to "0" (see Section " Program runtime (Page 685) ")

Via the following machine data it can be set that the system variable $AC_CYCLE_TIME is reset to "0" even in case of a program restart through the function "jump back to start of program":

MD27860 $MC_PROCESSTIMER_MODE.Bit 8 = <value> (activate the program runtime measurement)

Bit	Value	Meaning
8	0	$AC_CYCLE_TIME is **not** reset to "0" by the function "jump back to start of program".
	1	$AC_CYCLE_TIME is reset to "0" by the function "jump back to start of program".

Note

In order that the setting of bit 8 can become effective, the measurement of the current program runtime must be active (MD27860 bit 1 = 1).

Workpiece count

After the part program end (`M02` / `M30`) has been attained, the activated workpiece counters ($AC_TOTAL_PARTS / $AC_ACTUAL_PARTS / $AC_SPECIAL_PARTS) are incremented by "1" (see Section "Workpiece Counter (Page 695)").

Via the following machine data it can be set that the activated workpiece counter is incremented even in case of a program restart through the function "jump back to start of program":

MD27880 $MC_PART_COUNTER.Bit <n> = <value> (activating the workpiece counter)

Bit	Value	Meaning: In case of a program restart through the function "jump back to start of program", the workpiece counter:
7	0	$AC_**TOTAL**_PARTS **is not** incremented.
	1	$AC_**TOTAL**_PARTS is incremented.

Bit	Value	Meaning: In case of a program restart through the function "jump back to start of program", the workpiece counter:
11	0	$AC_**ACT**_PARTS **is not** incremented.
	1	$AC_**ACT**_PARTS is incremented.
15	0	$AC_**SPECIAL**_PARTS **is not** incremented.
	1	$AC_**SPECIAL**_PARTS is incremented.

Programming

Syntax

```
GOTOS
```

Meaning

`GOTOS`:	Return to the start of the current program	
	Preprocessing stop:	Yes
	Effective:	Non-modal

Example:

Programming	Comment
`N10 ...`	`; Start of the program`
`...`	
`IF ...`	
`   N100 GOTOS`	`Return to the program start (N10)`
`ENDIF`	
`...`	
`RET`	

10.9.10 Program section repetitions

10.9.10.1 Programming

The program section repetition permits the repetition of program sections within an NC program.

The program lines or program sections to be repeated are identified by jump markers (labels).

Note

Jump markers (labels)

Jump markers are always located at the beginning of a block. If a program number exists, the jump marker is located immediately after the block number.

The following rules apply when naming jump markers:

- Number of characters:
  - Minimum 2
  - Maximum 32
- Permissible characters are:
  - Letters
  - Digits
  - Underscores
- The first two characters must be letters or underscores.
- The name of the jump marker is followed by a colon (":").

Syntax

1. REPEATB: Repeat single program line

```
<jump marker>: ...
...
REPEATB <jump marker> P=<n>
```

2. REPEAT + jump marker: Repeat program section between jump marker and REPEAT statement

```
<jump marker>: ...
...
REPEAT <jump marker> P=<n>
```

3. REPEAT + jump marker_1 + jump marker_2: Repeat section between two jump markers

```
<start jump marker>: ...
...
<end jump marker>: ...
...
REPEAT <start jump marker> <end jump marker> P=<n>
```

Note

It is not possible to nest the `REPEAT` statement with the two jump markers in parentheses. If the `<start jump marker>` appears before the `REPEAT` statement and the `<end jump marker>` is not reached before the `REPEAT` statement, the section between the `<start jump marker>` and the `REPEAT` statement will be repeated.

4. REPEAT + jump marker + ENDLABEL: Repeat section between jump marker and ENDLABEL:

```
<jump marker>: ...
...
ENDLABEL: ...
...
REPEAT <jump marker> P=<n>
```

Note

It is not possible to nest the `REPEAT` statement with the `<jump marker>` and the `ENDLABEL` in parentheses. If the `<jump marker>` appears before the `REPEAT` statement and the `ENDLABEL` is not reached before the `REPEAT` statement, the section between the `<jump marker>` and the `REPEAT` statement will be repeated.

Meaning

`REPEATB`:	Command for repeating a program line
`REPEAT`:	Command for repeating a program section
`<jump marker>`:	The `<jump marker>` identifies: ● `REPEATB`: The program line to be repeated ● `REPEAT`: The start of the program section to be repeated
	The program line identified by the `<jump marker>` can appear before or after the `REPEAT`/`REPEATB` statement. The search initially commences towards the start of the program. If the jump marker is not found in this direction, the search continues in the direction towards the end of the program. **Exception:** If the program section between the jump marker and the `REPEAT` statement needs to be repeated (see 2. under Syntax), the program line identified by the `<jump marker>` has to appear **before** the `REPEAT` statement, since in this case the search runs **only** towards the beginning of the program. If the line with the `<jump marker>` contains further statements, they are executed again on each repetition.
`ENDLABEL`:	Keyword that marks the end of a program section to be repeated. If the line with the `ENDLABEL` contains further statements, they are executed again on each repetition. `ENDLABEL` can be used more than once in the program.

P:	Address for specifying the number of repetitions **Note:** If no number is specified for P=<n>, the program section is repeated just once.	
<n>:	Number of repetitions	
	Type:	INT
	The program section to be repeated is repeated <n> times. After the last repetition, the program is resumed at the program line following the REPEAT/ REPEATB command.	

Examples

Example 1: Repeat individual program line

```
Program code                      Comment
N10 POSITION1: X10 Y20
N20 POSITION2: CYCLE(0,,9,8)      ;Position cycle
N30 ...
N40 REPEATB POSITION1 P=5         ; Execute block N10 five times.
N50 REPEATB POSITION2             ; Execute block N20 once.
N60 ...
N70 M30
```

Example 2: Repeat program section between jump marker and REPEAT statement:

```
Program code                 Comment
N5 R10=15
N10 Begin: R10=R10+1         ; Start of the program section
N20 Z=10-R10
N30 G1 X=R10 F200
N40 Y=R10
N50 X=-R10
N60 Y=-R10
N70 Z=10+R10
N80 REPEAT BEGIN P=4         ; Execute section from N10 to N70 four times.
N90 Z10
N100 M30
```

Example 3: Repeat section between two jump markers

```
Program code                     Comment
N5 R10=15
N10 Begin: R10=R10+1             ; Start of the program section
N20 Z=10-R10
N30 G1 X=R10 F200
N40 Y=R10
N50 X=-R10
N60 Y=-R10
```

Program code	Comment
N70 END: Z=10	; End of the program section
N80 Z10	
N90 CYCLE(10,20,30)	
N100 REPEAT BEGIN END P=3	; Execute section from N10 to N70 three times.
N110 Z10	
N120 M30	

Example 4: Repeat section between jump marker and ENDLABEL

Program code	Comment
N10 G1 F300 Z-10	
N20 BEGIN1:	; Start program section 1
N30 X10	
N40 Y10	
N50 BEGIN2:	; Start program section 2
N60 X20	
N70 Y30	
N80 ENDLABEL: Z10	
N90 X0 Y0 Z0	
N100 Z-10	
N110 BEGIN3: X20	; Start program section 3
N120 Y30	
N130 REPEAT BEGIN3 P=3	; Execute section from N110 to N120 three times.
N140 REPEAT BEGIN2 P=2	; Execute section from N50 to N80 twice.
N150 M100	
N160 REPEAT BEGIN1 P=2	; Execute section from N20 to N80 twice.
N170 Z10	
N180 X0 Y0	
N190 M30	

Example 5: Milling, drilling position with different technologies

Program code	Comment
N10 CENTER DRILL()	; Load centering drill.
N20 POS_1:	; Start program section 1; drilling position 1
N30 X1 Y1	
N40 X2	
N50 Y2	
N60 X3 Y3	
N70 ENDLABEL:	
N80 POS_2:	; Start program section 2; drilling position 2
N90 X10 Y5	
N100 X9 Y-5	
N110 X3 Y3	
N120 ENDLABEL:	; End program sections 1 and 2

Program code	Comment
N130 DRILL()	; Drilling cycle
N140 THREAD(6)	; Threading cycle
N150 REPEAT POS_1	; Repeat program section once from POS_1 up to ENDLABEL.
N160 DRILL()	; Drilling cycle
N170 THREAD(8)	; Threading cycle
N180 REPEAT POS_2	; Repeat program section once from POS_2 up to ENDLABEL.
N190 M30	

Further information

- Program section repetitions can be nested. Each call uses a subprogram level.
- If `M17` or `RET` is programmed while processing a program section repetition, the program section repetition is canceled. The program is resumed at the block following the `REPEAT` line.
- In the actual program display, the program section repetition is displayed as a separate subprogram level.
- If the level is canceled during the program section processing, the program resumes at the point after the program section repetition call.
  Example:

Program code	Comment
N5 R10=15	
N10 BEGIN: R10=R10+1	;Width
N20 Z=10-R10	
N30 G1 X=R10 F200	
N40 Y=R10	; Interrupt level
N50 X=-R10	
N60 Y=-R10	
N70 END: Z10	
N80 Z10	
N90 CYCLE(10,20,30)	
N100 REPEAT BEGIN END P=3	
N120 Z10	; Resume program execution.
N130 M30	

- Check structures and program section repetitions can be used in combination. There should be no overlap between the two, however. A program section repetition should appear within a check structure branch or a check structure should appear within a program section repetition.
- If jumps and program section repetitions are mixed, the blocks are executed purely sequentially. For example, if a jump is performed from a program section repetition, processing continues until the programmed end of the program section is found.
  Example:

Program code
`N10 G1 F300 Z-10`
`N20 BEGIN1:`
`N30 X=10`
`N40 Y=10`
`N50 GOTOF BEGIN2`
`N60 ENDLABEL:`
`N70 BEGIN2:`
`N80 X20`
`N90 Y30`
`N100 ENDLABEL: Z10`
`N110 X0 Y0 Z0`
`N120 Z-10`
`N130 REPEAT BEGIN1 P=2`
`N140 Z10`
`N150 X0 Y0`
`N160 M30`

Note

The `REPEAT` statement should appear after the traversing block.

10.9.11 Event-driven program call (PROG_EVENT)

10.9.11.1 Function

Function

The "Event-driven program call" (PROG_EVENT) function starts, for the occurrence of a selected event in the NC, a user-specific NC program (PROG_EVENT-program).

Application examples

Initial setting for functions or initialization of system or user variables.

Events

The triggering events are selected with the machine data MD20108 $MC_PROG_EVENT_MASK (see Section "Parameterization (Page 574)").

PROG_EVENT program

The name of the PROG_EVENT program is set with the machine data MD11620 $MN_PROG_EVENT_NAME up (see Section "Parameterization (Page 574)").

The PROG_EVENT program is executed in the channel in which the event occurred.

The PROG_EVENT program is executed with the lowest priority and so can be interrupted by a user ASUP.

Processing sequences

Processing sequence when activated by an event: Program start

Initial state:

Channel:	Reset status
Mode:	AUTOMATIC (optional: Overstore (active)
	MDI
	TEACH IN

1. A program starts in the channel
2. Initialization sequence with evaluation of:
   - MD20112 $MC_START_MODE_MASK
3. Implicit call of the PROG_EVENT program as a subprogram
4. Processing of the data part of the main program
5. Processing of the program part of the main program

Processing sequence when activated by an event: Program end

Initial state

Channel:	Active
Mode:	AUTOMATIC (optional: Overstore (active)
	MDI
	TEACH IN

1. In the channel, the end of program block is exchanged in the executed program.
2. End of program is executed, evaluation of the following machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
3. Implicit call of the PROG_EVENT program as ASUP
4. A reset is executed in the channel, evaluation of the following machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

Sequence when activated by an event: Channel reset

Initial state:

Channel:	Any
Mode:	Any

1. Control activates reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
2. Implicit call of the PROG_EVENT program as ASUP
3. Control activates reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

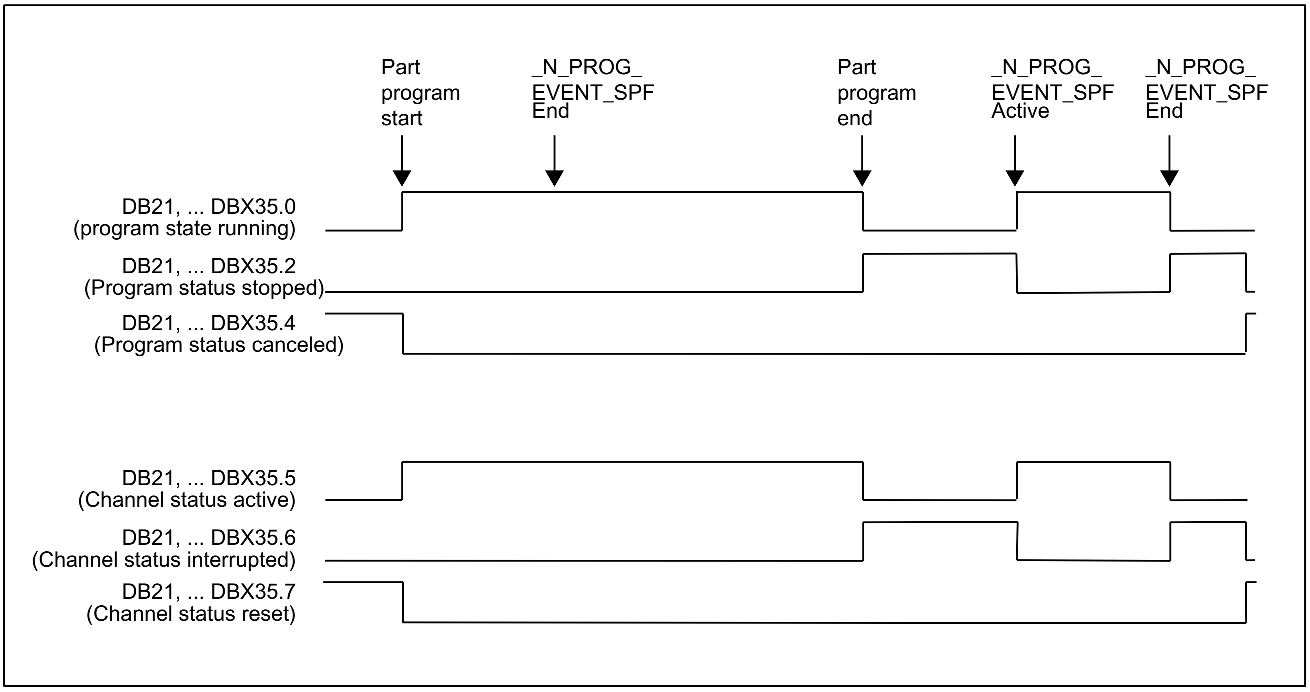
Sequence when activated by an event: Power-up of the NC

1. Control activates after power-up reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
2. Implicit call of the PROG_EVENT program as ASUP
3. Control activates reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

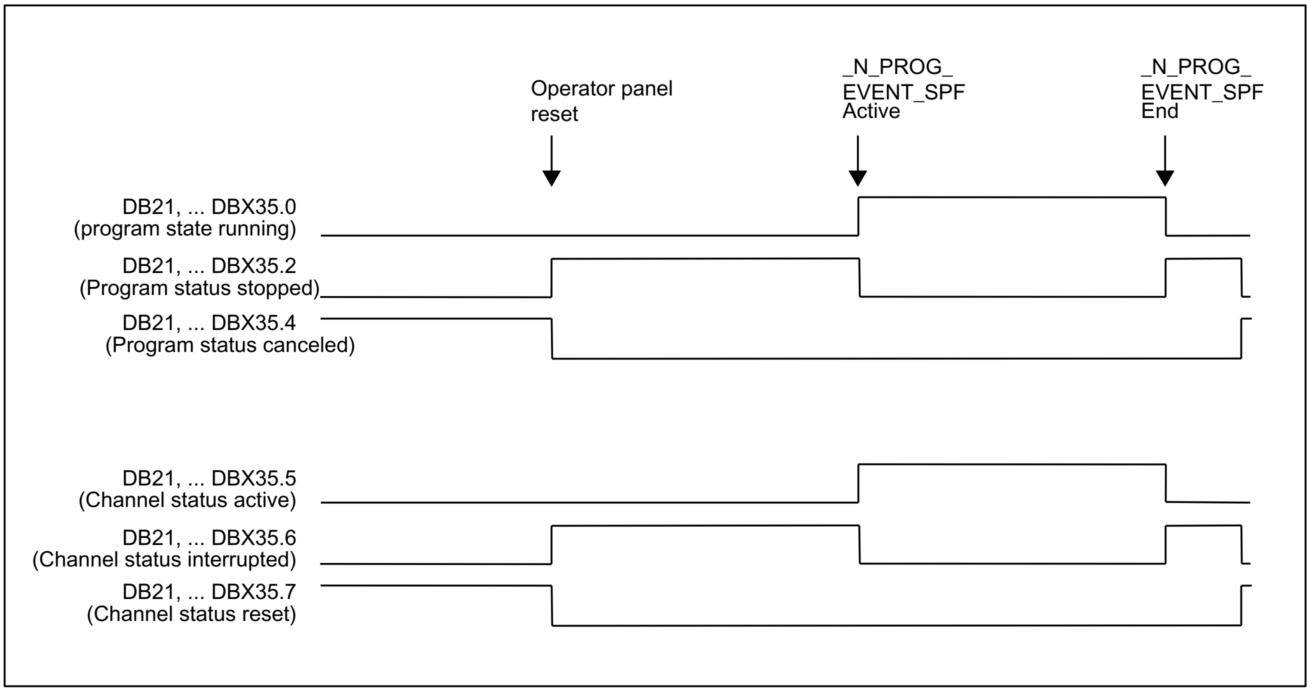
NC/PLC interface signals Change of "program status" and "channel status"

The following diagrams show the changes of the various NC/PLC interface signals for "program status" and "channel status" during the event-driven program call sequence.

Signal sequence when activated by program start and program end


Part program start
_N_PROG_ EVENT_SPF End
Part program end
_N_PROG_ EVENT_SPF Active
_N_PROG_ EVENT_SPF End
DB21, ... DBX35.0 (program state running)
DB21, ... DBX35.2 (Program status stopped)
DB21, ... DBX35.4 (Program status canceled)
DB21, ... DBX35.5 (Channel status active)
DB21, ... DBX35.6 (Channel status interrupted)
DB21, ... DBX35.7 (Channel status reset)

Signal sequence when activated by a channel reset


Operator panel reset
_N_PROG_ EVENT_SPF Active
_N_PROG_ EVENT_SPF End
DB21, ... DBX35.0 (program state running)
DB21, ... DBX35.2 (Program status stopped)
DB21, ... DBX35.4 (Program status canceled)
DB21, ... DBX35.5 (Channel status active)
DB21, ... DBX35.6 (Channel status interrupted)
DB21, ... DBX35.7 (Channel status reset)

NC/PLC interface signals: DB21, ... DBX35.4 (program status aborted) and DB21, ... DBX35.7 (channel status reset)

- The interface signals are set only when the PROG_EVENT program has completed again.
- The interface signals are not set between:
  - Program end and PROG_EVENT program start
  - Channel reset and PROG_EVENT program start

NC/PLC interface signals DB21, ... DBX376.0 ... 4 (triggering events)

The initiating event is displayed using interface signals DB21, ... DBX376.0 ... 4:

Bit	Value	Event
0	1	Program start from channel state "Reset"
1	1	End of program
2	1	Channel reset
3	1	Powering-up of the NC
4	1	1. Program start after block search (see "Automatic start of an ASUP after block search (Page 510)")

If the PROG_EVENT program has completed or been canceled with a channel reset, the interface signals are cleared again.

The interface signals are available for at least the duration of a PLC cycle.

10.9.11.2 Parameterization

Event selection

The events that start the PROG_EVENT program are set channel-specific using the machine data:

MD20108 $MC_PROG_EVENT_MASK.Bit <n> = 1

Bit	<value>	Meaning: Event
0	1	Part program start
1	1	Part program end
2	1	Channel reset
3	1	Power-up
5	1	Safety program during power-up

Note

MD20108 $MC_PROG_EVENT_MASK is not evaluated in the simulation.

PROG_EVENT program

The PROG_EVENT program (default: _N_PROG_EVENT_SPF) must be loaded and enabled.

Default setting

As default setting, when an event occurs, user program _N_CMA_DIR/ **_N_PROG_EVENT_SPF** is executed.

The PROG_EVENT program must be loaded and enabled.

User-specific setting

If, for an event, a different PROG_EVENT program than the default setting is to be executed, the NC program name must be entered into the following machine data:

MD11620 $MN_PROG_EVENT_NAME = <program name>

Search path

The PROG_EVENT program must be located in one of the following cycle directories. The following search path is passed through when the parameterized event occurs:

1. /_N_CUS_DIR/ (user cycles)
2. /_N_CMA_DIR/ (manufacturer cycles)
3. /_N_CST_DIR/ (standard cycles)

The first program found with the name specified in MD11620 $MN_PROG_EVENT_NAME is executed.

Note

- The specified program name is checked syntactically as in case of a subprogram name, i.e. the first two characters must be letters or underscores (no digits). Prefix (_N_) and suffix (_SPF) of the program names are added automatically, if not specified.
- The same protection mechanisms that can be activated for cycles (protection levels for writing, reading, etc.) are activated.

Behavior when starting a user ASUP

The behavior of the function "event-driven program call" upon start of a user ASUP from the reset channel state can be set channel-specific with the machine data:

MD20109 $MC_PROG_EVENT_MASK_PROPERTIES.Bit 0 = <value>

Bit	Value	Meaning
0	0	The PROG_EVENT program is started when one of the events parameterized with MD20108 occurs.
	1	The PROG_EVENT program is **not** started when one of the events parameterized with MD20108 occurs.

Behavior when the single-block processing is set

The behavior of the "event-driven program call" function for set single-block processing can be set channel-specific with the following machine data:

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK.Bit <n> = <value>

Bit	Value	Meaning: In the PROG_EVENT program, the single-block processing for event:
0	0	Part program start: **acts**
	1	Part program start: is **suppressed**
1	0	Part program end: **acts**
	1	Part program end: is **suppressed**
2	0	Channel reset: **acts**
	1	Channel reset: is **suppressed**
3	0	Power-up: **acts**
	1	Power-up: is **suppressed**

If the single-block processing is suppressed, the PROG_EVENT program is processed without interruption.

Note

- MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK affects all single-block processing types.
- The single-block processing in the PROG_EVENT program can be disabled with the following setting:
  MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, bit 0 = 1 (prevent single-block stop)
  The settings in MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK then do not act.

Behavior when the read-in disable is set

The behavior of the "event-driven program call" function in case of set read-in disable (DB21, ... DBX6.1 = 1) can be set channel-specific for each initiating event with the following machine data:

MD20107 $MC_PROG_EVENT_IGN_INHIBIT.Bit <n> = <value>

Bit	Value	Meaning: In the PROG_EVENT program, the read-in disable for event:
0	0	Part program start: **acts** If the PROG_EVENT program is terminated with the `RET` command, **no** executable block is created
	1	Part program start: is **suppressed** If the PROG_EVENT program is terminated with the `RET` command, **an** executable block analog to `M17` is created.
1	0	Part program end: **acts**
	1	Part program end: is **suppressed**
2	0	Channel reset: **acts**
	1	Channel reset: is **suppressed**

Bit	Value	Meaning: In the PROG_EVENT program, the read-in disable for event:
3	0	Power-up: **acts**
	1	Power-up: is **suppressed**

Suppress updating of the display of the program and channel states.

To prevent flickering when displaying the program and channel states on the HMI user interface, the display refresh can be suppressed for the execution of the normally very brief PROG_EVENT program. The program and the channel status before activation of the PROG_EVENT program then remain visible in the display.

MD20192 $MC_PROG_EVENT_IGN_PROG_STATE.Bit <n> = <value>

Bit	Value	Meaning: While processing the PROG_EVENT program, the refresh of the display of the program and channel states for event:
1	0	Part program end: **Not suppressed**
	1	Part program end: **Suppressed**
2	0	Channel reset: **Not suppressed**
	1	Channel reset: **Suppressed**
3	0	Power-up: **Not suppressed**
	1	Power-up: **Suppressed**

Note

The system variables $AC_STAT and $AC_PROG are not affected by this function, i.e. in the running event-driven user program, $AC_STAT is set to "active" and $AC_PROG to "running".

NC/PLC interface signals DB21, ... DBX35.0-7 ("Program state ..." and "Channel state ...") also remain unaffected.

Response for DB21, ... DBX7.2 / 3 / 4 (NC stop ...)

The response of the "event-driven program call" function for "NC stop", "NC stop at block limit" and "NC stop axes plus spindle" can be set channel-specific for the part program end, channel reset and ramp up events with the following machine data:

MD20193 $MC_PROG_EVENT_IGN_STOP.Bit <n> = <value>

Bit	Value	Meaning: The PROG_EVENT-program for NC stop and event
1	0	Part program end: **is stopped/prevented**
	1	Part program end: **is processed completely**
2	0	Channel reset: **is stopped/prevented**
	1	Channel reset: **is processed completely**
3	0	Power-up: **is stopped/prevented**
	1	Power-up: **is processed completely**

Application example

An edge change of the interface signal DB21, ... DBX7.3 (NC stop) initiated by the operator by pressing the NC Stop key during a channel reset or power-up is ignored during the execution of the PROG_EVENT program and so an undesired stop behavior at the machine is prevented.

Note

Programming `DELAYFSTON`/`DELAYFSTOF` in the PROG_EVENT program cannot replace the behavior set with MD20193. NC stop before executing the `DELAYFSTON` command can still cause an interruption.

10.9.11.3 Programming

PROG_EVENT program

End of program

The following must be kept in mind, if the user program is to be activated through the part program start.

- The user program must be ended with `M17` or `RET`.
- A return jump with the `REPOS` command is not permissible.

Block display

The display can be suppressed in the current block display using the `DISPLOF` attribute in the `PROC` statement.

Communication to the PLC user program

User M functions programmed in the PROG_EVENT program can inform the PLC user program, e.g. about the processing status of the PROG_EVENT program.

System variable

Query for triggering event

The initiating event can be queried in the PROG_EVENT program with the following system variables:

<value> = $P_PROG_EVENT (event-driven program call active)

Value	Meaning: Activation by
1	**Part program start**
2	**Part program end**
3	**Channel reset**
4	**Power-up**
5	After output of the last action block after **Block search** (see "Automatic Start of an ASUP after block search (Page 510)")

Query of the current channel

The channel in which the PROG_EVENT program is processed can be determined with the following system variables:

<value> = $P_CHANNO (query the current channel number)

Note

The PROG_EVENT program is processed in the channel in which the initiating event occurred.

Power-up is an event that takes place concurrently in all channels.

10.9.11.4 Boundary conditions

Emergency stop / alarm

If an emergency stop or a mode group / NC-specific alarm is present for a channel reset or after power-up, the PROG_EVENT program is processed only after the emergency stop or the error has been acknowledged in all channels.

Note

The "power-up" event occurs in all channels concurrently.

10.9.11.5 Examples

Example 1: Call the PROG_EVENT program for all events

Parameterization

MD20108 $MC_PROG_EVENT_MASK = 'H0F'	Call of _N_PROG_EVENT_SPF for: • Part program start • Part program end • Channel reset • Power-up

Programming

Program code	Comment
`PROC PROG_EVENT DISPLOF`	
`; Processing for part program start`	
`IF ($P_PROG_EVENT==1)`	
`   MY_GUD_VAR=0`	`; Initialize GUD variable`
`   RET`	
`ENDIF`	

Program code	Comment
`; Processing for part program end and channel reset`	
`IF ($P_PROG_EVENT==2) OR ($P_PROG_EVENT==3)`	
`   DRFOF`	`; Deactivate DRF offsets`
`   IF $MC_CHAN_NAME=="CHAN1"`	
`      CANCEL(2)`	`; Clear modal synchronized action 2`
`   ENDIF`	
`   RET`	
`ENDIF`	
`; Sequence for power-up`	
`IF ($P_PROG_EVENT==4)`	
`   IF $MC_CHAN_NAME=="CHAN1"`	
`      IDS=1 EVERY $A_INA[1]>5.0 DO $A_OUT[1]=1`	
`   ENDIF`	
`   RET`	
`ENDIF`	
`RET`	

Example 2: ; Call the PROG_EVENT program for channel reset

Parameterization

MD20108 $MC_PROG_EVENT_MASK = 'H04'	Call of _N_PROG_EVENT_SPF for: • Operator panel reset

Programming

Program code	Comment
`PROC PROG_EVENT DISPLOF`	
`N10 DRFOF`	`; Deactivate DRF offsets`
`N20 M17`	

Example 3: Initialization of the function

Parameterization

Machine data assignment, extract from the commissioning file (_N_INITIAL_INI)

Program code	Comment
`...`	
`CHANDATA(3)`	`; Initialization for channel 3`
`$MC_PROG_EVENT_IGN_INHIBIT='H01F'`	
`$MC_PROG_EVENT_MASK='H04'`	`; Event: Channel reset`
`...`	

Meaning

The PROG_EVENT program _N_CMA_DIR/_N_PROG_EVENT_SPF is started for channel reset in the 3rd channel and processed until program end, irrespective of whether or not the read-in disable is activated or deactivated.

10.9.12 Influencing the stop events by stop delay areas

10.9.12.1 Function

Stop delay area

The response to a stop event can be influenced by the conditioned interruptible area in the NC program. Such a program area is called a stop delay area.

Within the stop delay areas there should be no stop and the feedrate should not be changed. Stops do not take effect until the program section has been completed.

This has the following advantages:

- A program section is processed without a drop in velocity.
- If the user aborts the program after a stop with reset, the aborted program block is after the protected section. This program block is a suitable search target for a subsequent block search.
- The following main run axes are not stopped as long as a stop delay area is in progress:
  - Command axes
  - Positioning axes that travel with POSA

Application

Example: Machining a thread.

Definition

The definition of a stop delay area is made in the part program with the predefined DELAYFSTON and DELAYFSTOF procedures (see "Programming (Page 584)").

Stop events

Overview of the events that cause a stop:

Event	Reaction
DB21, ... DBX7.7 (reset) or DB11 DBX20.7 (mode group reset)	Immediate
NC program: M30	Alarm 16954
User interrupt to start an ASUP: FC 9 or DB10 DBB1 (setting the digital NC inputs for the PLC)	Delayed
DB21, ... DBX6.2 / DB31, ... DBX2.2 (clear distance-to-go) channel/axis-specific	Immediate

Event	Reaction
DB21, ... DBX6.3 (clear the number of subprogram repetitions)	Delayed
DB21, ... DBX6.4 (program level abort)	Delayed
Stop because of single block • In the stop delay area: The NC stops at the end of the 1st block outside the stop delay area. • Single block is active before the stop delay area: NC stops at each block limit, i.e. also in the stop delay area: DB21, ... DBX7.2 (NC stop at the block limit) **This deselects the stop delay area!**	Delayed
DB11 DBX21.7 (activate single-block type A)	Delayed
DB11 DBX21.6 (activate single-block type B)	Delayed
DB21, ... DBX7.4 (NC stop) and DB11 DBX0.6 (mode group stop, axes plus spindles)	Immediate
DB21, ... DBX7.3 (NC stop at block limit) and DB11 DBX0.5 (mode group stop)	Delayed
NC program: Stop because of empty overstore buffer	Alarm 16954
NC program: Programmed or implicit STOPRE	Alarm 16954
NC program: M0 or M1	Alarm 16954
DB21, ... DBX7.2 (NC stop at the block limit)	Delayed
Subprogram end should always deselect the stop delay area.	System error
NC program: WAITM	Alarm 16954
NC program: WAITE	Alarm 16954
NC program: INIT with parameter "S"	Alarm 16954
NC program: MMC (STRING, CHAR)	Alarm 16954
DB21, ... DBB26 (activate/deactivate skip block)	Delayed
DB21, ... DBB26 (deactivate skip block)	Delayed
DB21, ... DBX0.6 (activate DryRun)	Delayed
DB21, ... DBX0.6 (deactivate DryRun)	Delayed
DB21, ... DBX6.1 (read-in disable)	Delayed
Alarm: Alarm configuration STOPATENDBYALARM	Immediate
Alarm: Alarm configuration STOPBYALARM	Immediate
Internal: Stop after alarm for empty IPO buffer	Immediate
Internal: Stop after alarm for empty IPO buffer	Immediate
NC program: Alarm 16954 for LFON	Alarm 16954
NC program: WAITMC	Alarm 16954
NC program: NEWCONF	Alarm 16954
NC program: NEWCONF	Alarm 16954
OPI: PI "_N_SETUDT"	Delayed
Extended stop and retract	Delayed
DB31, ... DBX3.0 (accept external work offset)	Delayed
OPI: PI "_N_FINDST" STOPRUN	Alarm 16955
Alarms with NOREADY response	Immediate

Event	Reaction
DB21, ... DBX7.2 == 1 (NC stop at block limit) DB21, ... DBX7.3 == 1 (NC stop) DB21, ... DBX7.4 == 1 (NC stop, axes plus spindles)	Delayed
DB21, ... DBX6.1 == 1 (read-in disable)	Delayed
DB21, ... DBX2.0 == 1 (single block)	Delayed

Reaction

- **Immediate**
  Stops immediately, even in the stop delay area. Designated as a "hard stop event".
- **Delayed**
  Does not stop (even short-term) until 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 continued, an illegal action occurred in the stop delay area.
- **Alarm 16957**
  The program area (stop delay area) enclosed by DELAYFSTON and DELAYFSTOF could not be activated. As a result, every stop will take effect immediately and is not subject to a delay! This will always occur when the deceleration begins before the stop delay area but ends within the stop delay area. Likewise, if the stop delay area is entered with an override of 0, the stop delay area also cannot be activated. (Example: A G4 before the stop delay area allows the user to reduce the override to 0. The next block in the stop delay area then begins with override 0 and the described alarm situation occurs.)
  Note
  MD11411 $MN_ENABLE_ALARM_MASK (activation of warnings) Bit 7 activates this alarm.

Note

Some NC events are stopped for only a short time, in order to perform a switching operation, and restart immediately. This includes, e.g. an interrupt that stops the NC program briefly in order to start the associated ASUP. 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".

Note

If a "hard" stop event coincides with the "stop delay area", the entire "stop delay area" is deselected! Thus, if any other stop occurs in this program section, it will be stopped immediately. A new program setting (new DELAYFSTON) must be made in order to start a new stop delay area.

10.9.12.2 Parameterization

Machine data

Stop response for G331/G332

For tapping without compensating chuck (G331, G332), the stop response can be set as follows:

MD11550 $MN_STOP_MODE_MASK

Bit	Value	Meaning
0	0 (default)	Results in an implicit stop delay range if G331/G332 is active - and path motion or a dwell time (G4) was also programmed.
	1	A stop while G331/G332 is active is possible in the following situations: ● The continuous path mode is interrupted (G64). ● A dwell time (G4) is programmed between the G33x blocks. NC commands DELAYFSTON and DELAYFSTOF must be used to define a stop delay range.

10.9.12.3 Programming

Defining a stop delay range (DELAYFSTON, DELAYFSTOF)

The predefined DELAYFSTON and DELAYFSTOF procedures are used to define a conditionally interruptible range in the part program (stop delay range).

Note

DELAYFSTON and DELAYFSTOF are **not** permitted in synchronized actions!

Syntax

```
DELAYFSTON
...
DELAYFSTOF
```

Meaning

`DELAYFSTON`:	Defining the start of a stop delay range	
	Alone in the block:	Yes
`DELAYFSTOF`:	Define the end of the stop delay area	
	Alone in the block:	Yes

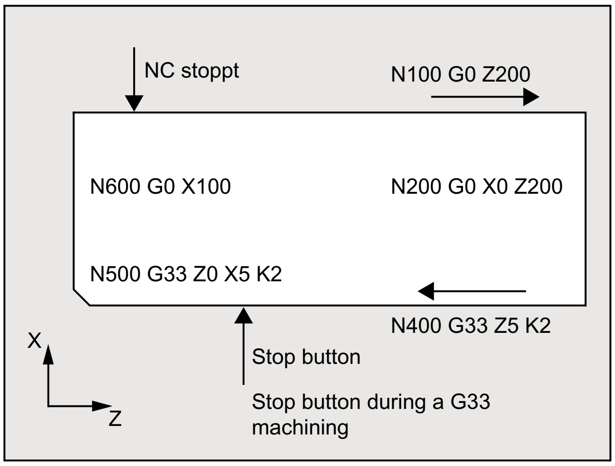
Programming example

The following program block is repeated in a loop:

Program code

```
...
N99 MY_LOOP:
N100 G0 Z200
N200 G0 X0 Z200
N300 DELAYFSTON
N400 G33 Z5 K2 M3 S1000
N500 G33 Z0 X5 K3
N600 G0 X100
N700 DELAYFSTOF
N800 GOTOB MY_LOOP
...
```

In the following diagram it can be seen that the user pressed "Stop" in the stop delay range, and the NC started braking outside the stop delay range, i.e. in block N100. That causes the NC to stop at the beginning of N100.



Additional information

End of subprogram

DELAYFSTOF is activated implicitly at the end of the subprogram in which DELAYFSTON is called.

Nesting

If subprogram 1 calls subprogram 2 in a stop delay area, the whole of subprogram 2 is a stop delay area. In particular, DELAYFSTOF in subprogram 2 has no effect.

Example:

```
Program code                  Comment
N10010 DELAYFSTON             ; Blocks with N10xxx program level 1.
N10020  R1 = R1 + 1
N10030  G4 F1                 ; Stop delay area starts.
...
N10040  subprogram2
...
...                           ; Interpretation of subprogram 2.
N20010 DELAYFSTON             ; Ineffective, repeated start, 2nd level.
...
N20020 DELAYFSTOF             ; Ineffective, end at another level.
N20030  RET
N10050 DELAYFSTOF             ; Stop delay end of range at the same level.
...
N10060  R2 = R2 + 2
N10070  G4 F1                 ; Stop delay area ends. From now, stops act immediately.
```

System variables

The following system variables can be queried to determine whether part program processing is currently in a stop delay area:

- in the part program with $P_DELAYFST
- in synchronized actions with $AC_DELAYFST

Value	Meaning
0	Delay stop range not active
1	Delay stop area active

10.9.12.4 Supplementary conditions

Overlap

If two stop delay areas overlap, one from DELAYFSTON/DELAYFSTOF and the other from MD11550 $MN_STOP_MODE_MASK, the largest possible stop delay area will be generated.

Dwell time (G4)

G4 is permitted in the stop delay area. Other part program commands that cause a stop in the meantime (e.g. WAITM) are not permitted and trigger the alarm 16954.

Override

If the override is changed **before** a stop delay area, the override takes effect **in** the stop delay area.

If the override is changed **in** the stop delay area, the change takes effect **after** the stop delay area.

Note

Override = 0

If the override is reduced to 0 **before** a stop delay area, the stop delay area cannot be activated.

Feed disable

DB21, ... DBX6.0 The feed disable has no effect in the stop delay area; the program does not stop until after the stop delay area.

Single block

If the single block is activated in the stop delay area, the NC stops at the end of the first block outside the stop delay area.

If the single block is already selected before the stop delay area, the NC stops at each block limit, i.e. also in the stop delay area! **This deselects the stop delay area.**

10.10 Asynchronous subprograms (ASUPs)

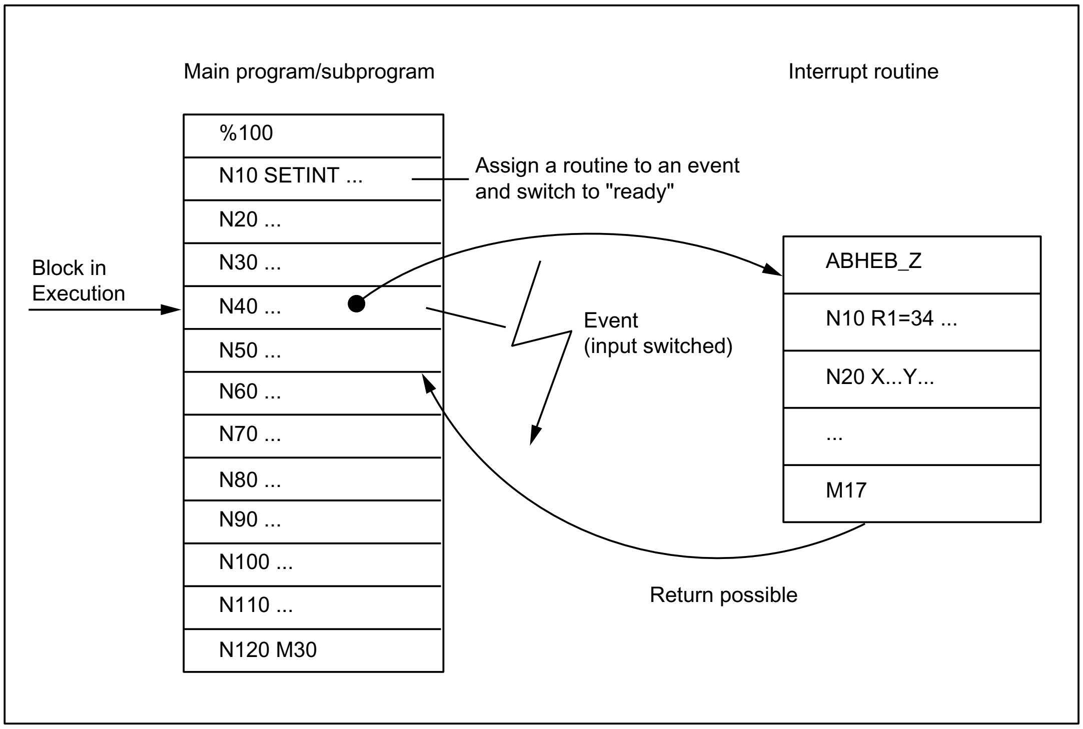
10.10.1 Function

Note

The terms "asynchronous subprogram", "ASUP" and "interrupt routine" used interchangeably in the description below refer to the same functionality.

General

Asynchronous subprograms (ASUPs) are NC programs that are started in an NC channel as response to asynchronous events (interrupt input signals, process and machine states). The activation of an ASUP interrupts an NC program currently executing. The NC program can be continued at the interrupt position when the ASUP ends.



The NC program being executed in the channel can be protected from being interrupted by an ASUP either completely or only in sections. See Section "Programming (SETINT, PRIO) (Page 597)" "Flexible programming".

Definition

To make an ASUP (interrupt routine) from an NC program, the NC program must be assigned an interrupt signal via the `SETINT` command (see Section "Programming (SETINT, PRIO) (Page 597)") or via the "ASUP" PI service (see Section "PI service: ASUP (Page 993)").

Interrupt signals

- A total of eight I/O inputs are available as interrupt signals.
- The I/O input signals can be influenced via the PLC user program.
- The first four I/O inputs are the four fast inputs of the NCU module. The signal states can be read via the NC/PLC interface in the DB10 data block. The input signals can also be locked via the NC/PLC interface in the DB10 data block.

For further information about PLC control of the fast NC inputs (interrupt signals) see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)".

References:
Function Manual, Extended Functions; A4: Digital and analog NC I/O

Call

During program mode

During program mode, i.e. in AUTOMATIC or MDI mode, an ASUP can always be called.

Outside the program mode

Outside the program mode, an ASUP can be called in the following modes, machine functions and states:

- JOG, JOG REF
- MDI Teach In, MDI Teach In REF, MDI Teach In JOG, MDI REF, MDI JOG
- AUTOMATIC, program states "stopped", "ready"
- Axis state "Not referenced"

If an ASUP is started during JOG or JOG REF, the current traversing is aborted.

Activation

An ASUP is activated via:

- 0/1 edge of the interrupt signal, triggered by a 0/1 edge at the associated fast NC input
- Call of the "Function call ASUP" (see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)")
- Setting an output via synchronized action which is parameterized on an interrupt via short-circuit (see "Examples (Page 600)")
  References:
  Function Manual, Synchronized Actions

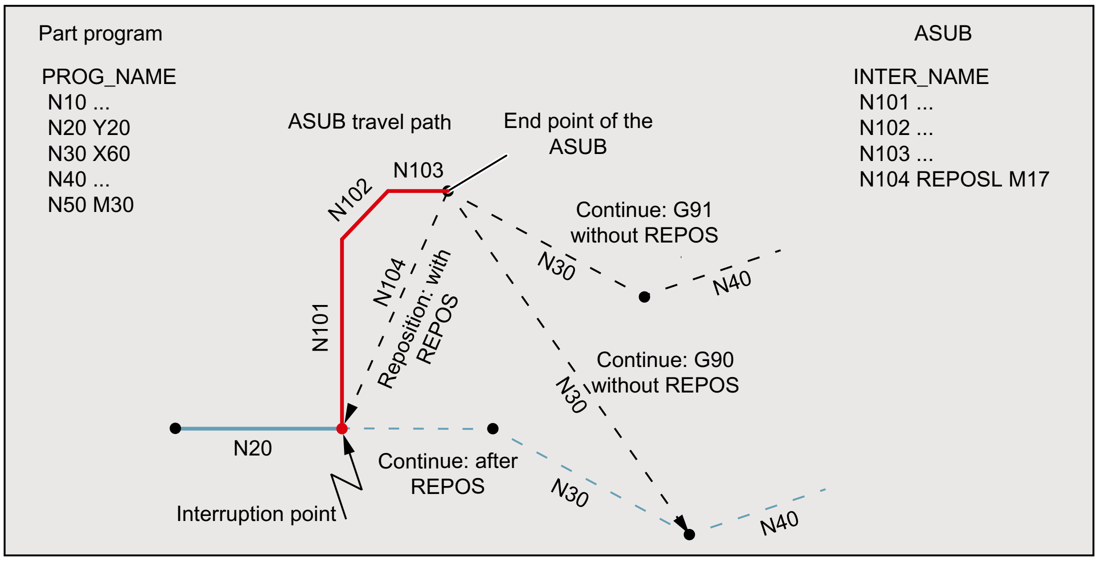
Display

The activation of an ASUP is shown channel-specifically with the following NC/PLC interface signal:

DB21, … DBX378.0 == 1 (ASUP active)

10.10.1.1 Execution sequence of an ASUP in program mode

1. Decelerating the axes
   Upon activation of the ASUP, all machine axes are decelerated to a standstill according to the deceleration ramp (MD32300 $MA_MAX_AX_ACCEL), and the axis positions are saved.
2. 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 commands 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 also buffered so they remain available after the end of the ASUP.
   Exceptions where no reorganization is possible:
   - In thread cutting blocks
   - With complex geometries (e.g. spline or radius compensation)
3. Processing the ASUP
   The ASUP is started automatically on completion of the reorganization.
   The ASUP is processed as a normal subprogram (nesting depth, etc.).
4. End of the ASUP
   After the end identifier (`M02`, `M30`, `M17`) of the ASUP has been processed, the axis traverses by default to the end position programmed in the part program block following the interruption block.
   A `REPOS` statement must have been programmed at the end of the ASUP if return positioning to the point of interruption is required, e.g. `N104 REPOSL M17`



10.10.1.2 ASUP with REPOSA

If an NC program is stopped by an NC stop or alarm, and an ASUP with REPOSA is subsequently initiated from the PLC user program using FC9, then typically, the following sequence is obtained:

- The ASUP and/or the traversing motion programmed in it are executed:
  - Program status: "Stopped"
  - DB21, ... DBX318.0 (ASUP is stopped) = 1
- A new stop is executed before repositioning to the contour (REPOS).
- The operator initiates an NC start to reposition to the contour (REPOS):
  - DB21, ... DBX318.0 (ASUP is stopped) = 0
  - The reapproach movement is executed.
- At the end of the reapproach motion, the FC9 acknowledge signal "ASUP done" is set and the path of the interrupted NC program is continued.

Note

The NC/PLC interface signal DB21, ... DBX318.0 (ASUP is stopped) is set only in the following case:

Interrupt in the program mode and in the "interrupted" channel state.

Note

For ASUPs without REPOS, the FC9 acknowledge signal "ASUP done" and the reset of the NC/PLC interface signal DB21, ... DBX318.0 = 0 (ASUP is stopped) coincide from a timing perspective.

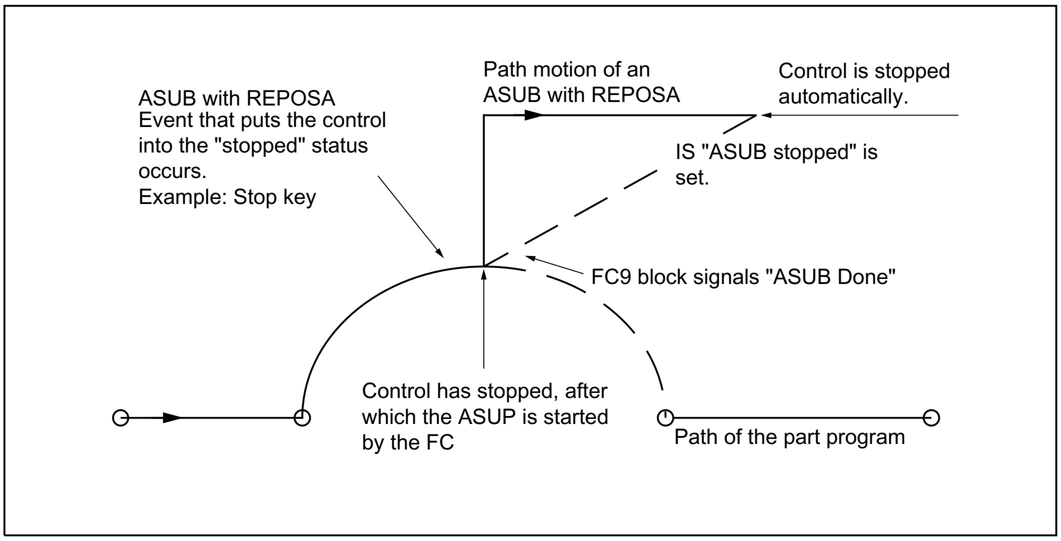

ASUB with REPOSA
Event that puts the control into the "stopped" status occurs.
Example: Stop key
Path motion of an ASUB with REPOSA
Control is stopped automatically.
IS "ASUB stopped" is set.
FC9 block signals "ASUB Done"
Control has stopped, after which the ASUP is started by the FC
Path of the part program

Figure 10-7 Schematic sequence: ASUP with REPOSA

10.10.1.3 NC response

The different responses in the various states for channel, mode group or NC on an activated ASUP are listed in the following table:

Status	ASUP start	Control system response
Program is active	Interrupt, (PLC)	1. Fast retraction or stop axes 2. Interrupt the program for the duration of the ASUP 3. Approach of the interruption point, if REPOS in ASUP 4. Continuation of the part program
Reset	Interrupt, (PLC)	The ASUP is executed like a main program. Reset (without `M30`) is executed at the end of the ASUP. The next control system status depends on the following machine data: MD20110 $MC_RESET_MODE_MASK MD20112 $MC_START_MODE_MASK **References:** Function Manual Basic Functions; Axes, Coordinate Systems, Frames (K2), Section: "Workpiece-related actual-value system"
Program mode (AUTOMATC or MDI) + channel stopped	Interrupt, (PLC)	ASUP is executed. At the end of the ASUP the STOP state is reapplied. If REPOS in the ASUP: • The ASUP processing is stopped before the approach block. • The approach movement can be initiated with the Start key.
	Start key	Once the ASUP has been executed, processing of the interrupted program is resumed.
Manual mode + channel stopped	Interrupt, (PLC)	Control system assumes the status "internal program execution mode" for the addressed channel (not evident externally) and then activates the ASUP. The selected operating mode remains valid. The original status is resumed after execution of the ASUP (`M17`).
JOG AUTO Teach-In AUTO Teach reference point.	Interrupt, (PLC)	Stop processing, evaluate: MD11602 $MN_ASUP_START_MASK MD11604 $MN_ASUP_START_PRIO_LEVEL Internal switchover to "internal program execution mode" if appropriate, activate ASUP, restore status prior to ASUP start. Any `LIFTFAST` defined with `SETINT` is not activated in JOG mode.
MDI JOG, MDI Teach-In, MDI Teach reference point.	Interrupt, (PLC)	
Manual mode + channel running	Interrupt, (PLC)	The current active motion is stopped. The distance-to-go is deleted. The remaining sequence of operations is the same as for "Manual mode, channel stopped".
User alarm 65500 - 65999, channel in reset	Interrupt, (PLC)	When the following machine data is set, the user ASUP is processed despite the "NC Start disable" alarm response: MD20194 $MC_IGNORE_NONCSTART_ASUP

Status	ASUP start	Control system response
Processing of INITIAL.INI	**Not possible**	The signal "Interrupt request not possible" is generated.
Block search		
Alarm that cannot be removed by NC Start.		
Digitalizing active		
Channel in fault condition		
User alarm other than 65500 - 65999, channel in reset, MD20194 not active	**Not possible**	The "request was canceled because of an alarm" signal is generated.

10.10.2 Commissioning: Machine data

10.10.2.1 NC-spec.: Mode-group-specific NC/PLC interface signals and operating mode switchover

The machine data specifies the effectiveness of the mode-group-specific NC/PLC interface signals of DB11 and the channels in which an operating mode switchover is performed:

MD11600 $MN_BAG_MASK = <Value>

Value	Meaning
0	The mode-group-specific NC/PLC interface signals of DB11 are **effective**. An internal operating mode switchover is performed in **all** channels of the mode group.
1	The mode-group-specific NC/PLC interface signals of DB11 are **not** effective. An internal operating mode switchover is performed **only** in the channel in which an **ASUP** is active.
2	The mode-group-specific NC/PLC interface signals of DB11 are **effective**. An internal operating mode switchover is performed **only** in the channel in which an **ASUP** is active.
3	The mode-group-specific NC/PLC interface signals of DB11 are **not** effective. An internal operating mode switchover is performed **only** in the channel in which an **ASUP** is active.

Note

Multi-channel systems

If the "Manual traversing during an ASUP interruption in JOG mode" function is to be possible in multi-channel systems (see below), MD11600 $MN_BAG_MASK must be set to "2" or "3".

See also

Programming (SETINT, PRIO) (Page 597)

10.10.2.2 NC-spec.: ASUP start enable

With the machine data, you can specify which stop reasons are to be ignored for an ASUP start:

MD11602 $MN_ASUP_START_MASK, <Bit> = <Value>

Bit	Value	Meaning
0	0	The ASUP is **not** started when a stop is present from the Stop key, M0 or M01.
	1	The ASUP is **started** even when a stop is present from the Stop key, M0 or M01.
2	0	When an ASUP is started, the channel-specific machine data is effective when there is a read-in disable in the channel: • MD20107 $MC_PROG_EVENT_IGN_INHIBIT • MD20116 $MC_IGNORE_INHIBIT_ASUP
	1	The start of a user or system ASUP is enabled in all channels of the NC even when there is a read-in disable in the relevant channel. The settings in the channel-specific machine data are **not** effective: • MD20107 $MC_PROG_EVENT_IGN_INHIBIT • MD20116 $MC_IGNORE_INHIBIT_ASUP **Notice** The start enable is active in all channels of the NC despite a read-in disable. Not only for the user, but also for the system ASUPs. It is therefore **strongly recommended** that the **channel-specific** enables are used instead of the NC-specific enable.
3	0	If an ASUP was started in JOG mode and interrupted by an NC stop, axes **cannot** be traversed manually in this state.
	1	If an ASUP was started in JOG mode and interrupted by an NC stop, axes can be traversed manually in this state. The ASUP can be continued with NC start. A REPOS operation is then performed automatically. **Note** If more than one channel is parameterized in the control, the following machine data must also be set: MD11600 $MN_BAG_MASK, bit 1 = 1

Manual start enable

If an ASUP is not started **automatically** due to the parameterized start enables, the ASUP can still be started via NC/PLC interface signal (DB21, ... DBX7.1) from the PLC user program or by manual actuation of NC start.

Note

The ASUP for "fast retraction from the contour" (`LIFTFAST`) is started in every case.

10.10.2.3 NC-spec.: Effectiveness of the parameterized start enables

With the machine data, you set up to which ASUP priority (Page 597), starting from the highest priority, the settings in MD11602 $MN_ASUP_START_MASK are effective:

MD11604 $MN_ASUP_START_PRIO_LEVEL = <ASUP priority>

Example

MD11604 $MN_ASUP_START_PRIO_LEVEL = 5

The settings in MD11602 $MN_ASUP_START_MASK are effective for ASUPs of priorities 1 → 5.

10.10.2.4 Channel-spec.: Start enable despite non-referenced axes

With the machine data, you can set for which interrupts the associated ASUP is started despite the parameterized "NC start disable without reference point" function (MD20700 $MC_REFP_NC_START_LOCK):

MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit (1 - <Interrupt>) = TRUE

NOTICE

System interrupts

With MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bits 8 - 31, the ASUPs assigned to the system interrupts are enabled.

Bit 8 / interrupt 9 starts an ASUP that contains traversing motions.

NC-specific ASUP start enable

If MD11602 $MN_ASUP_START_MASK, bit 2 == TRUE, the ASUP start enable is set for all channels of the NC despite the parameterized channel-specific "NC start disable without reference point" function (MD20700 $MC_REFP_NC_START_LOCK). It is therefore **strongly recommended** that the **channel-specific** enable is used instead of the NC-specific enable.

10.10.2.5 Channel-spec.: Start enable despite read-in disable

With the machine data, you can set for which interrupts the associated ASUP is started despite the read-in disable present in the channel (DB21, ... DBX6.1):

MD20116 $MC_IGNORE_INHIBIT_ASUP, bit (1 - <Interrupt>) = TRUE

NOTICE

System interrupts

With MD20116 $MC_IGNORE_INHIBIT_ASUP, bits 8 - 31, the ASUPs assigned to the system interrupts are enabled.

Bit 8 / interrupt 9 starts an ASUP that contains traversing motions.

NC-specific ASUP start enable

If MD11602 $MN_ASUP_START_MASK, bit 2 == TRUE, the channel-specific settings in MD20116 $MC_IGNORE_INHIBIT_ASUP are ignored in all channels of the NC. It is therefore **strongly recommended** that the **channel-specific** enable is used instead of the NC-specific enable.

10.10.2.6 Channel-spec.: Continuous execution despite single block

With the machine data, you can set for which interrupts the associated ASUP is executed **continuously**, i.e. without block-by-block interruption, despite active single block processing in the channel (DB21, ... DBX0.4):

MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP, bit (1 - <Interrupt>) = TRUE

Supplementary conditions

The settings in MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP are only active for a single block SBL1 (main run single block).

NOTICE

System interrupts

With MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP, bits 8 - 31, the ASUPs assigned to the system interrupts are enabled.

Bit 8 / interrupt 9 starts an ASUP that contains traversing motions.

NC-specific ASUP start enable

If MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, bit 1 == TRUE, the channel-specific settings in MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP are ignored in all channels of the NC. It is therefore **strongly recommended** that the **channel-specific** enable is used instead of the NC-specific enable.

10.10.2.7 Channel-spec.: Refreshing the display

With the machine data, you can set that the display is not refreshed while the ASUP is being executed to avoid a flickering of the display of the program and the channel states on the user interface when executing a very short ASUP:

MD20191 $MC_IGN_PROG_STATE_ASUP, bit (1 - <Interrupt>) = TRUE

Note

NC/PLC interface signal

The following NC/PLC interface signal is set when executing an ASUP with suppressed display:

DB21, … DBX378.1 = 1 ("silent" ASUP active)

System variables and NC/PLC interface signals

The system variables and NC/PLC interface signals for the program state and the channel state are not affected by the suppression of the display during the execution of an ASUP:

- $AC_STAT (channel state)
- $AC_PROG (program state)
- DB21, ... DBX35.5 - 7 (channel state)
- DB21, ... DBX35.0 - 4 (program state)

10.10.3 Programming: System variables

10.10.3.1 REPOS option ($P_REPINF)

In conjunction with ASUPs, program execution sequences can be generated for which there is no unambiguous return to a repositioning point on the contour (REPOS).

The system variable can be used to read in the ASUP whether a REPOS is possible.

<value> = $P_REPINF

Value	Meaning
0	Repositioning with REPOS not possible because: • Not called in the ASUP • ASUP ran from reset state • ASUP ran from JOG
1	Repositioning with REPOS possible in ASUP

10.10.3.2 Activation event ($AC_ASUP)

The following information with regard to the event that caused the activation of the ASUP can be read via the $AC_ASUP system variable:

- The reason why the ASUP was activated, e.g. Bit 0: User interrupt "ASUP with Blsync"
- How the ASUP was activated, e.g. Bit 0: NC/PLC interface signal, digital-analog interface
- How is it possible to continue, e.g. Bit 0: Freely selectable REORG or RET

10.10.4 Programming (SETINT, PRIO)

Assignment: Interrupt to the NC program

The `SETI` command assigns an NC program to an interrupt. This makes the NC program into an ASUP.

Syntax

```
SETINT(<n>) <NC program>
```

Meaning

`SETINT`:	Assignment of an NC program to an interrupt signal	
`<n>`:	Number of the interrupt signal	
	Value range:	0, 1, 2, ... 8
`<NC program>`:	Program name	

Example

Program code	Comment
`N20 SETINT(3) LIFT_Z`	`; IF input 3 == 1`
	`; THEN start ASUP "LIFT_Z"`

Together with `SETINT` additionally the following statements can be programmed:

- `LIFTFAST`
  When the interrupt signal arrives, a "Fast retraction of the tool from the contour" is performed before the ASUP starts. The motion direction for the fast retraction is specified by the program statement `ALF`.
- `BLSYNC`
  Upon receiving the interrupt signal, the current program block is processed and only then is the ASUP started.

Note

The assignment interrupt signal ↔ part program is cleared when the following happens:

- Channel in Reset state
- `CLRINT` statement in the part program

Priorities

If several interrupts are activated by `SETINT` in an NC program, the assigned NC programs or ASUPs must be assigned different priorities.

Syntax

```
PRIO=<value>
```

Meaning

`PRIO`:	Keyword for defining the priority of the interrupt
`<value>`:	Priority: 1, 2, 3 ... 128. Whereby 1 is the highest priority.

Example

Program code	Comment
`N20 SETINT(3) PRIO=2 LIFT_Z`	`; IF input 3 == 1`
	`; THEN start ASUP "LIFT_Z"`
`N30 SETINT(2) PRIO=3 LIFT_X`	`; IF INPUT 2 == 1`
	`; THEN start ASUP "LIFT_X"`

The ASUPs are executed in the sequence of the priority values if the inputs 2 and 3 have switched simultaneously:

1. "LIFT_Z"
2. "LIFT_Z"

Additional interrupt-specific commands

Command	Meaning
`SAVE`	If the `SAVE` command is used in an ASUP, the G commands, frames and transformations active in the interrupted NC program before the interruption take effect again at the end of the ASUP.
`DISABLE` `ENABLE`	The `DISABLE` `ENABLE` command pair can be set to protect program sections from being interrupted by ASUPs. The assignment of an interrupt signal to an NC program made with `SETINT` is retained. The ASUP is then started with the next 0/1 edge of the interrupt signal after `ENABLE`.
`CLRINT(<n>)`	Delete the assignment of interrupt signals n to the NC program assigned with `SETI`.

References

Programming Manual, Job Planning; Section: "Flexible NC Programming" > "Interrupt routine (ASUP)"

See also

Programming (Page 602)

10.10.5 Restrictions

Cross-mode ASUP start

Settings to be checked

- MD11600 $MN_BAG_MASK
- MD11604 $MN_ASUP_START_PRIO_LEVEL
- Interrupt assignment priority

Recommended settings

NC-specific machine data:

- MD11600 $MN_BAG_MASK = 'H3'

  Note

  Note with this setting, that the mode-group-specific NC/PLC interface signals of DB11 no longer affect the channel in which the ASUP is executed. If this behavior is not desired, alternatively the setting MD11600 $MN_BAG_MASK = 'H2' can be used (see "Commissioning: Machine data (Page 593)").

  MD11602 $MN_ASUP_START_MASK = 'H5'
- MD11604 $MN_ASUP_START_PRIO_LEVEL = 7

Channel-specific machine data for the channel in which the ASUP is started or generally for all channels:

- MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK, bit <n> = TRUE
  <n>: For all required event-driven program calls (prog events)
- MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit <n> = TRUE
  <n>: For all **required user** interrupts

NOTICE

System interrupts

With MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bits 8 - 31, the system interrupts are enabled.

Bit 8 / interrupt 9 starts an ASUP that contains traversing motions.

10.10.6 Examples

Activation of an ASUP by an interrupt from a synchronized action

1. Parameterize two active digital I/O bytes:
   - MD10350 $MN_FASTIO_DIG_NUM_INPUTS = 2
   - MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS = 2
2. Parameterize a short-circuit with OR operation from output 9 to input 9:
   - Input 1, input byte 2 = (output 1, output byte 2) **OR** (HW input signal 1, input byte 2):
     MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[0] = 'H0102B102'
3. Assign the HW input byte to the SETINT interrupt programming:
   - Input byte 2:
     MD21210 $MC_SETINT_ASSIGN_FASTIN = 2
4. Define input as ASUP trigger:
   - Input 1 in the second input byte, i.e. absolute input 9, starts the SYNCASUP program
     `SETINT(1) PRIO=1 SYNCASUP`
5. Define a synchronized action to set the output:
   - Synchronized action with ID 1 always sets output 9 to 1 when the value of the standardized path parameter becomes >= 0.5:
     `IDS=1 EVERY $$AC_PATHN >= 0.5 DO $A_OUT[9]=1`
     The short-circuit of output 9 to input 9 causes interrupt 1 and the "SYNC" NC program to be started as ASUP.

10.11 User-specific ASUB for RET and REPOS

10.11.1 Function

Function

The control software includes a Siemens-specific ASUP to implement the NC program end (RET) and repositioning on the contour (REPOS) functions. The machine tool manufacturer can replace the system ASUP with a user-specific ASUP.

⚠ DANGER

Programming fault

The machine manufacturer has sole responsibility for ensuring the correct operation of the user-specific ASUP that replaces the Siemens-specific ASUP ("ASUP.SYF").

10.11.2 Parameter assignment

Installation

ASUP name

The user-specific ASUP must be given the following name:

- **_N_ASUP_SPF**

ASUP directories

The user-specific ASUP "_N_ASUP_SPF" must be stored in one of the two directories:

- **_N_CMA_DIR** (manufacturer directory)
- **_N_CUS_DIR** (user directory)

Activation and search path

Bits 0 and 1 of the following machine data specify the event for which the user-specific ASUP "_N_ASUP_SPF" is activated.

Bit 2 specifies the search start for activation of the user-specific ASUP "_N_ASUP_SPF".

MD11610 $MN_ASUP_EDITABLE, bit 0, 1, 2 = <value>

Bit	Value	Meaning
0 and 1	0	The user-specific ASUP is activated neither for program end (**RET**) nor for repositioning on the contour (**REPOS**).
	1	The **user-specific** ASUP is activated for **RET**. The **system-specific** ASUP is activated for **REPOS**.
	2	The **system-specific** ASUP is activated for **RET**. The **user-specific** ASUP is activated for **REPOS**.
	3	The **user-specific** ASUP is activated for **RET** and **REPOS**.
2	0	The user-specific ASUP is searched for first in the user directory **_N_CUS_DIR**.
	1	The user-specific ASUP is searched for first in the manufacturer directory **_N_CMA_DIR**.

Defining a level of protection

If a user-specific ASUP is to be used for RET and/or REPOS (MD11610 $MN_ASUP_EDITABLE ≠ 0), a protection level can be defined for the user-specific routine "_N_ASUP_SPF". The level of protection can have values in the range 0 - 7. The setting is made via the following machine data:

MD11612 $MN_ASUP_EDIT_PROTECTION_LEVEL <protection level of the user-specific ASUP>

For further information about protection levels, refer to:
References:
Commissioning Manual; level of protection concept

Behavior when the single-block processing is set

The following machine data specifies that the system-specific ASUP and the user-specific ASUP "_N_ASUP_SPF" should be processed without interruption, despite active single-block processing:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, bit 0 = <value>

Bit	Value	Meaning
0	0	For active single-block mode, a stop is made in **every ASUP**.
	1	For active single-block mode, the ASUP is processed **without interruption**.

10.11.3 Programming

Determining the cause of the ASUB activation

The cause of the activation of the ASUB can be read bit-coded via the system variable $AC_ASUP.

Continuation

When using the system ASUB, the behavior for the continuation after execution of the actions is permanently specified within the ASUB:

- System ASUB 1 → continuation with `RET` (subprogram return)
- System ASUB 2 → continuation with `REPOS` (repositioning)

The description of the system variables specifies the behavior with regard to the system ASUB for each cause at "Continued for".

Note

Continued for user-specific ASUB

It is recommended for user-specific ASUBs that the appropriate continuation of the system ASUB be retained.

Cause: Change of operating mode ($AC_ASUP, bit 9 == 1)

At a change of operating mode, the continuation depends on the machine data:

MD20114 $MC_MODESWITCH_MASK (interruption of MDA through mode change)

- Bit 0 == 0: System ASUB 1 → continuation with `RET`
- Bit 0 == 1: System ASUB 2 → continuation with `REPOS`

References

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

List Manual, System Variables

10.12 Perform ASUB start for user alarms

10.12.1 Function

Description

An ASUP can be initiated in various situations, either by the user, the system or event-controlled. A user alarm with the "NC Start disable" alarm response prevents the ASUP start in some situations. For a pending alarm response, a user ASUP from reset acknowledges with the alarm 16906.

With the appropriate setting of a channel-specific machine data it is possible to start and perform an ASUP although a user alarm with the alarm response "NC Start disable" is active. The alarm response is overridden for the ASUP start and permits its execution.

Note

NC alarms with the "NC Start disable" alarm response are not affected by the override. A user ASUP from reset is still not possible and will be rejected with the alarm 16906.

User alarms that can be overridden

The number range for the user alarms that can be overridden is classified as follows:

Number range	Effect	Delete
65000 - 65499	Display, NC Start disable	Reset
65500 - 65999	Display, NC Start disable (not for ASUPs for set MD20194)	Reset
66000 - 66999	Display, NC Start disable, motion standstill after executing the pre-decoded blocks	Reset
67000 - 67999	Display	Cancel
68000 - 68999	Display, NC Start disable, immediate interpolator stop	Reset
69000 - 69999	Display, NC Start disable, stop at next block end	Reset

Note

Although it is possible to override the "NC Start disable" alarm response for the following alarm number ranges, the other responses of the associated alarms ensure for the stop conditions. The stop conditions cannot be overridden with the function described in this section:

- 66000 - 66999
- 68000 - 68999
- 69000 - 69999

10.12.2 Activation

Setting

Each ASUP channel can be set separately with the following channel-specific machine data:

MD20194 $MC_IGNORE_NONCSTART_ASUP (ASUP start permitted despite pending alarm response "NC Start disable" for certain user alarms)

A change of the MD setting acts only with the `NEWCONF` part program command or from the user interface per softkey.

Sequence

The normal sequence for the ASUP start has the following form:

- Set machine data appropriate with MD20194 $MC_IGNORE_NONCSTART_ASUP and activate with `NEWCONF`.
- Start the part program.
  A user alarm from the number ranges that can be overridden appears, e.g. alarm 65500. This can occur from a synchronized action or with a part program command.
- Despite alarm, the part program is processed to the program end `M02`, `M30` or `M17`.
  The Reset channel state is active.
- A started user ASUP from reset is now performed.

Note

An ASUP start from a running or stopped part program is also permitted for a pending "NC Start disable" alarm response. It is irrelevant whether a user alarm or an NC alarm with the "NC Start disable" alarm response is involved.

Note

If the part program is stopped during the processing of the user ASUP, the ASUP can no longer be continued with the NC Start key, e.g. for `M0` in the ASUP or for user stop. The "NC Start disable" alarm response is rejected with the alarm 16906. The previously generated user alarm can be acknowledged only with reset.

10.12.3 Examples

10.12.3.1 User ASUB from reset - example 1

MD20194 is not set in the application

Main program

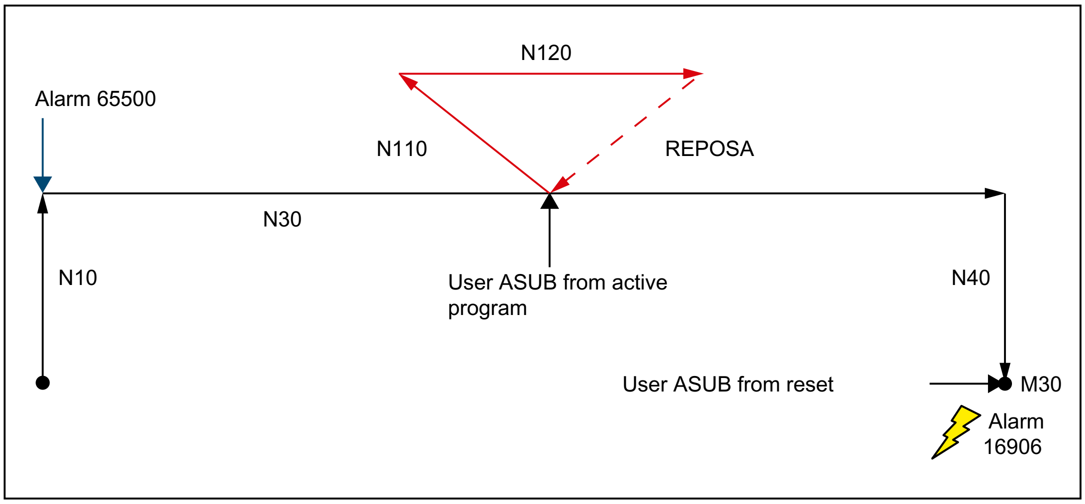
```
Program code
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30
```

User ASUB

```
Program code
N110 G91 G0 X-10 Z5
```

```
Program code
N120 X20
N130 REPOSA
```

Sequence



The block `N10` is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block `N30` is loaded and processed. If the user ASUB is used in the block middle, it is performed despite pending "NC Start disable" alarm response. The `REPOSA` resumes at the program interruption and processes the part program to the program end `M30`. If the user ASUB is now used, it will be rejected with alarm 16906. The NC state is reset.

10.12.3.2 User ASUB from reset - example 2

The application is the normal case, MD20194 is set.

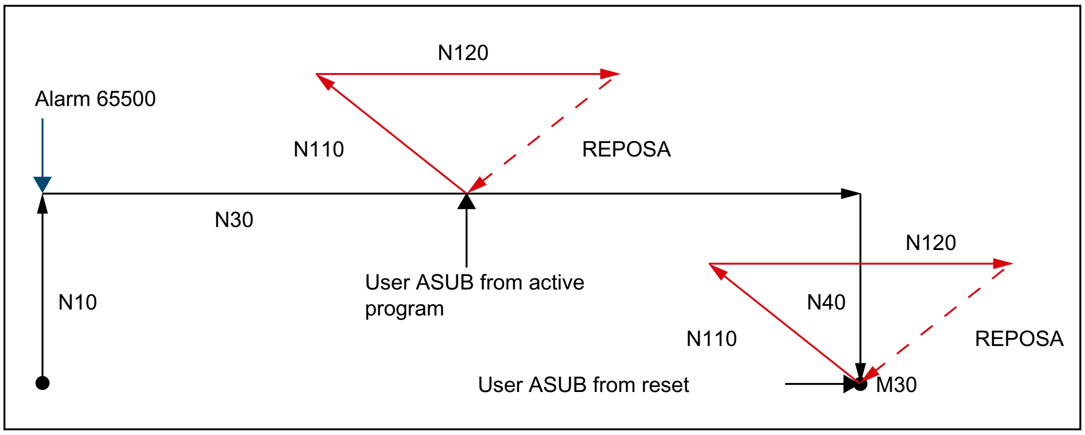
Main program

```
Program code
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30
```

User ASUB

```
Program code
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA
```

Sequence



The machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set for ASUB channel 1 and activated with `NEWCONF`. The block `N10` is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block `N30` is loaded and processed. If the user ASUB is used in the block middle, it is performed despite pending "NC Start disable" alarm response. The `REPOSA` resumes at the program interruption and processes the part program to the program end `M30`. If the user ASUB is now used, it is repeated because of the set machine data MD20194.

10.12.3.3 User ASUB with M0

MD20194 is set in the application.

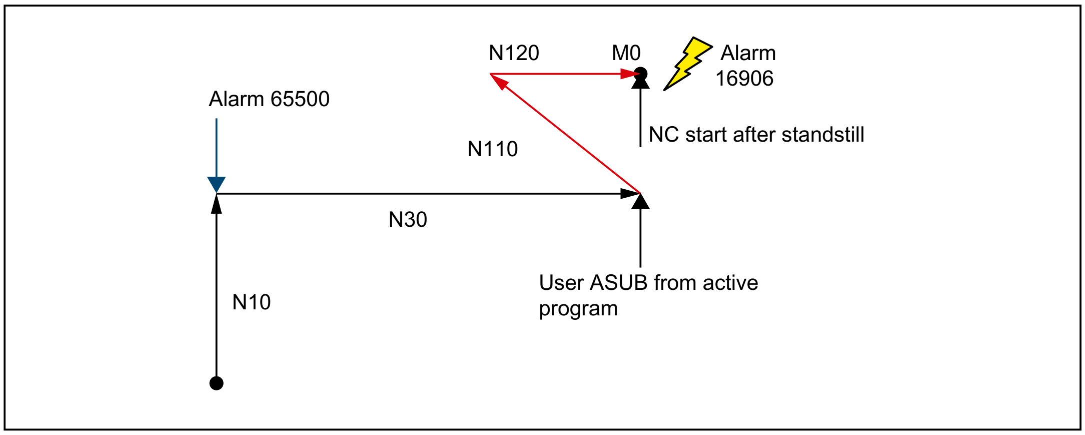
Main program

```
Program code
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30
```

User ASUB

```
Program code
N110 G91 G0 X-10 Z5
N120 X10
N122 M0
N124 X10
N130 REPOSA
```

Sequence



The machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set for ASUB channel 1 and activated with `NEWCONF`. The block `N10` is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block `N30` is loaded and processed. If the user ASUB is used in the block middle, it is performed despite pending "NC Start disable" alarm response. The part program stops at block `N122` because of `M0`. The ASUB can no longer be continued when the NC Start key is pressed. The "NC Start disable" alarm response is rejected with the alarm 16906. The previously generated 65500 alarm can be acknowledged only with reset.

10.12.3.4 User ASUB with stop

MD20194 is set in the application.

Main program

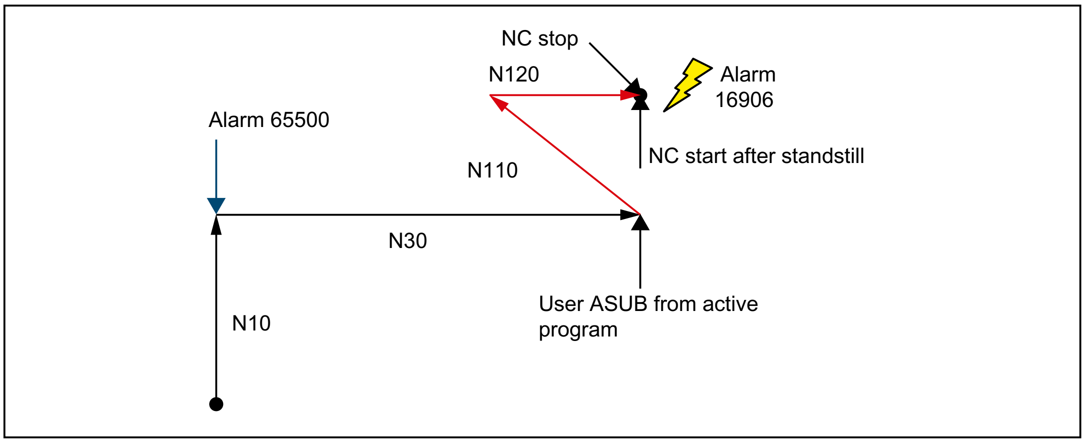
```
Program code
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
```

```
Program code
N50 M30
```

User ASUB

```
Program code
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA
```

Sequence



The machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set for ASUB channel 1 and activated with `NEWCONF`. The block `N10` is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block `N30` is loaded and processed. If the user ASUB is used in the block middle, it is performed despite pending "NC Start disable" alarm response. If the NC Stop key is pressed for `N120` in the block middle, the ASUB stops. The ASUB can no longer be continued when the NC Start key is pressed. The "NC Start disable" alarm response is rejected with the alarm 16906. The previously generated 65500 alarm can be acknowledged only with reset.

10.12.3.5 User ASUB from stopped

MD20194 is or not set in the application.

Main program

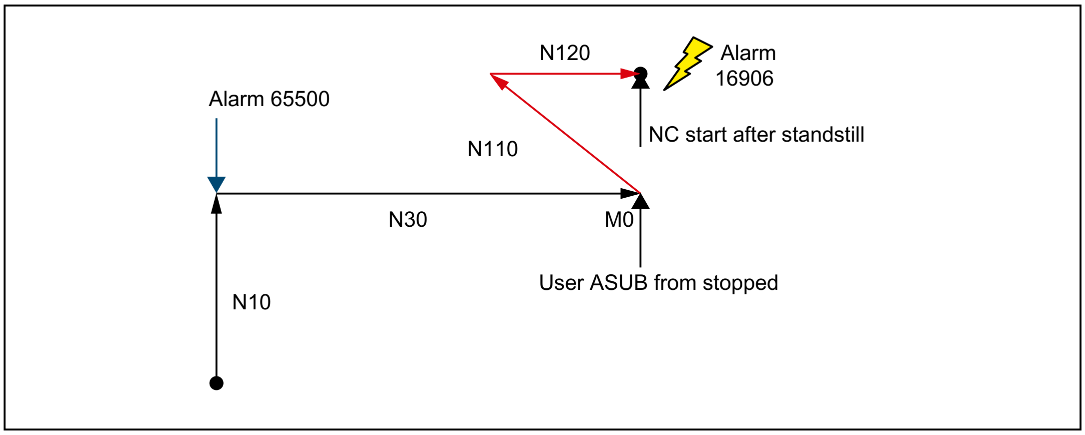
```
Program code
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X50
N35 M0
N38 X100
```

```
Program code
N40 Z0
N50 M30
```

User ASUB

```
Program code
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA
```

Sequence



The block `N10` is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block `N30` is loaded and processed. The part program stops at block `N35` because of `M0`. The NC state is stopped. If the user ASUB is now used, it is performed despite the stopped NC state and pending "NC Start disable" alarm response. It does not matter whether or not the machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set. After block `N120`, the ASUB remains stationary before `REPOSA`. Repositioning is possible only at the next NC Start. The NC Start, however, is rejected with the alarm 16906 because of the "NC Start disable" alarm response. The previously generated 65500 alarm can be acknowledged only with reset.

10.13 Single block

The "Single block" function allows an NC program to be executed block by block (non-modally) in three different ways.

The following single block types are available:

- **"SBL1: single block coarse"** (stop after each main run block)
  After each completely executed main program block, the NC program and/or processing is stopped.
- **"SBL2: calculation block"** (stop after each decoded block)
  After each decoded block, the NC program and/or processing is stopped.
  If several main run blocks are generated from a decoded block, a stop is made after each main run block.

  Note

  Thread cutting blocks

  For thread cutting blocks, the NC program is not stopped and machining is not stopped.

- **"SBL3: single block fine"** (stop after each decoded block, also in a cycle)
  As with SBL2, however, the NC program is stopped and machining is stopped after each block, also within a cycle.

User interface: Selecting the single block type

The SBL1, SBL2 or SBL3 single block type is selected from the HMI user interface:

Operating area "Machine" > "Influence prog." > Menu: "Program control"

10.13.1 Parameterization

Interface signals

Activation

The "single block" function is activated on a channel-specific basis using the NC/PLC interface signal:

DB21, ... DBX0.4 (activate single block)

Machine data

Deactivate single block machining (MD10702, MD20106, MD20117)

- Using the machine data, for certain machining situations and program types it can be selected that in spite of active single block function, a stop is not made:
  MD10702 $MN_IGNORE_SINGLEBLOCK_MASK

  Note

  - By programming (Page 613) `SBLON`/`SBLOF`within an ASUB or subprogram, single block machining can be explicitly activated/deactivated.
  - For single block type "SBL2: calculation block" the machine data only acts for system ASUBs, user ASUBs and subprograms with attribute `DISPLOF`.

- The behavior of event-driven program calls (program events) regarding a single block, is set using:
  MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK
- The behavior of interrupt programs (ASUB) regarding a single block, is set using:
  MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP
  If an ASUP is activated during the single block, execution of the ASUP is completed. The single block does not act until after the ASUP or in the first main program block in which single-block suppression is not activated. If for a transition from ASUB to the NC program the path velocity is too large so that braking to standstill in the following block is not possible, e.g. for active continuous-path mode G64, the deceleration may extend over several following blocks.

  Note

  By programming (Page 613) `SBLON` within an ASUB, in this case single block machining **cannot** be reactivated.

Setting data

Debug mode for single block "SBL2: calculation block" (SD42200)

As a result of the leading decoding of the blocks, the reference between the main block-related actual block display and the displayed variable values can be lost at the user interface. It is possible under certain circumstances that variable values are displayed that are not plausible.

With the following channel-specific setting data, for an active single block "SBL2: calculation block" a preprocessing stop can be executed for each block. This suppresses the premachining of part program blocks and maintains the relationship between the current block display and the variable values display.

SD42200 $SC_SINGLEBLOCK2_STOPRE (activate debug mode for SBL2)

Note

Contour deviation

When executing traversing blocks, single block type "**SBL2: calculation block**" in the **debug mode** contour deviations can occur.

10.13.2 Programming

10.13.2.1 Deactivating/activating single block machining (SBLOF, SBLON)

Deactivating single block machining for the complete NC program:

If deactivating single block machining (`SBLOF`) is programmed in the first of line (`PROC`) of a **main program**, then this remains valid until the end of the NC program or until the NC program is canceled. The NC program is then executed without stopping when in the single block mode.

If deactivating single block machining (`SBLOF`) is programmed in the first of line (`PROC`) of a **subprogram**, then this remains valid until the end of the NC program or until the NC program is canceled. With the programmed return command, the decision is made whether to stop at the end of the subprogram:

- Return jump with `M17`: Stop at the end of the subprogram
- Return jump with `RET`: **No** stop at end of subprogram

Deactivating single block machining within the NC program

If the deactivation of single block machining (`SBLOF`) is programmed in a block within an NC program, then single block machining is deactivated from this block onward up to the next programmed activation of single block machining (`SBLON`) - or at the end of the active subprogram level.

Syntax

```
SBLOF
SBLON
```

Meaning

`SBLOF`:	Predefined procedure to deactivate single block machining	
	Alone in the block:	yes, possible in the `PROC` block
	Effective:	Modal
`SBLON`:	Predefined procedure to activate single block machining	
	Alone in the block:	Yes
	Effective:	Modal

10.13.2.2 Supplementary conditions

Single block suppression and block display

The current block display can be suppressed in subprograms using `DISPLOF`. If `DISPLOF` is programmed together with `SBLOF`, then for single block stops within the subprogram, the subprogram call is displayed.

Special issues relating to various single block types

- "SBL2: calculation block" AND MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, bit 12 == 1:
  A stop is not made in the `SBLON` block.
- "SBL3: single block fine": command `SBLOF` is suppressed

Single block suppression for asynchronous subprograms (ASUB)

In order that an ASUB is executed without stopping, even when single block machining is active, `PROC` together with `SBLOF` must be programmed in the first program line of the ASUB.

Example

Program code	Comment
N10 **PROC** ASUP1 **SBLOF** DISPLOF	
N20 IF $AC_ASUP=='H200'	
N30 RET	; No REPOS for mode change.
N40 ELSE	
N50 REPOSA	; REPOS in all other cases.
N60 ENDIF	

10.13.2.3 Examples

Example 1: Single-block suppression within a program

```
Program code                  Comment
N10 G1 X100 F1000
N20 SBLOF                     ; Deactivate single block.
N30 Y20
N40 M100
N50 R10=90
N60 SBLON                     ; activate single block
N70 M110
N80 ...
```

The range between `N20` and `N60` is handled just like a block when the single block mode is activated.

Example 2: Subprogram without stopping

Main program

```
Program code
...
N100    G1 X10 G90 F200
N120    X-4 Y6
N130    CYCLE1
```

```
Program code
N140    G1 X0
N150 M30
```

Subprogram

```
Program code                                  Comment
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
```

Even when single block machining is active, the cycle is completely executed.

Example 3: ASUB with single block suppression and not visible

Processing a ASUB started by the PLC user program should not be interrupted, even when single block machining is active. Further, the ASUB should not be visible.

```
Program code
N100 PROC NV SBLOF DISPLOF        ; single block and display suppression
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
```

Example 4: Specific stopping in the subprogram

Assumptions:

- Single-block execution is active.
- MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, bit12 = 1

Main program

Program code	**Comment**
`N10 G0 X0`	`; single block stop`
`N20 X10`	`; single block stop`
`N30 **CYCLE**`	`; Traversing block generated by the cycle.`
`N50 G90 X20`	`; single block stop`
`M30`	

Subprogram

Program code	**Comment**
`**PROC** CYCLE **SBLOF**`	`; single block suppression`
`N100    R0 = 1`	
`N110    **SBLON**`	`; no single stop due to MD10702, bit12 = 1`
`N120    X1`	`; single block stop`
`N140    **SBLOF**`	
`N150    R0 = 2`	
`RET`	

Example 5: Single-block suppression for program nesting

Assumption: Single-block execution is active.

Program code	**Comment**
`N10 X0 F1000`	`; single block stop`
`N20 UP1(0)`	
`      **PROC** UP1(INT _NR) **SBLOF**`	`; deactivate single block for UP1`
`      N100 X10`	
`      N110 UP2(0)`	
`            PROC UP2(INT _NR)`	
`            N200 X20`	
`            N210 **SBLON**`	`; activate single block`
`            N220 X22`	`; single block stop`
`            N230 UP3(0)`	
`                  PROC UP3(INT _NR)`	
`                  N300 **SBLOF**`	`; deactivate single block.`
`                  N305 X30`	
`                  N310 **SBLON**`	`; activate single block`
`                  N320 X32`	`; single block stop`
`                  N330 **SBLOF**`	`; deactivate single block.`

Program code	Comment
`            N340 X34`	
`            N350 M17`	; single block stop (M17)
`        N240 X24`	; single block stop (N210)
`        N250 M17`	; single block stop (M17)
`    N120 X12`	
`    N130 M17`	; single block stop (M17)
`N30 X0`	; single block stop
`N40 M30`	; single block stop

10.13.3 Mode group-specific single block type A / B

For a mode group-specific single block, in a channel (control channel) the NC program is executed in single blocks. In the control channel, single block must be activated using the NC/PLC interface signal (DB21 ... DBX0.4).

In the other channels of the mode group (dependent channels), the particular NC program is executed block by block corresponding to the single block type A or B selected corresponding to the mode group using the NC/PLC interface signal (DB11 DBX1.6 / 7). In the dependent channels, it is not permissible that the NC/PLC interface signal (DB21 ... DBX0.4) is set.

Single block type A / B

- Single block type A: If the control channel stops, then also the dependent channels immediately stop, comparable with NC stop.
- Single block type B: If the control channel stops, then also the dependent channels stop at the particular end of block, comparable with NC stop at the block limit.

Interface signals

Control channel

- DB21, ... DBX0.4 (activate single block)

All channels of the mode group

- DB21, ... DBX7.1 (NC Start)

Mode group

- DB11 DBX1.6 (single block, type B)
- DB11 DBX1.7 (single block, type A)

Schematic execution sequence for single block type A

Precondition: All mode group channels are in the "Reset" or "Interrupted" state.

1. PLC user program: Select single block in the control channel, DB21 ... DBX0.4 = 1
2. PLC user program: Select single block type A for the mode group, DB11 DBX1.7 = 1

3. PLC user program: All channels of the mode group start, DB21 ... DBX0.4 = 1
4. The control channel stops at the end of the block.
5. All dependent channels receive an internal signal to immediately stop machining.
6. All mode group channels are in the "Interrupted" state once all of the dependent channels have reached the particular end of the braking phase.

Schematic execution sequence for single block type B

Precondition: All mode group channels are in the "Reset" or "Interrupted" state.

1. PLC user program: Select single block in the control channel, DB21 ... DBX0.4 = 1
2. PLC user program: Select single block type B for the mode group, DB11 DBX1.6 = 1
3. PLC user program: All channels of the mode group start, DB21 ... DBX0.4 = 1
4. The control channel stops at the end of the block.
5. All dependent channels receive an internal signal to stop machining at the end of the block.
6. All mode group channels are in the "Interrupted" state once all of the dependent channels have reached the particular end of the block.

10.13.4 Supplementary conditions

10.13.4.1 SBL2 single block type and block-related synchronized actions

For single block type **"SBL2: calculated block"** for a block-related synchronized action, the next stop is only executed after the next main program block. No stop is made in the initial blocks located between the synchronized action and the next main program block.

10.13.4.2 Programmed stop (M0), single block and single block type switchover

Initial situation: In a channel, an NC program is stopped by an **M0** programmed in it, and in the channel a **single block is active** (DB21, ... DBX0.4 == 1)

If, in this situation the **single block type** is toggled between SBL1 or SBL3 and SBL2 at the user interface **several times**, then alarm 16922 "Maximum nesting depth exceeded" is displayed.

10.14 Program control

Options

1. Function selection (via operator interface or PLC)
2. Activation of skip levels

3. Adapting the size of the interpolation buffer
4. Program display modes via an additional basic block display
5. Execution from external source (buffer size and number)
6. Execution from external subroutines

10.14.1 Function selection from the user interface or PLC user program

Sequence

Selection of a function is made via the SINUMERIK Operate user interface in the "Automatic" > "Program control" operating area by setting the corresponding selection signal in the HMI/ PLC interface.

Depending on the value of FB1 parameter `MMCToIf`, the selection signal of the HMI/PLC interface signal is transmitted from the basic PLC program to the corresponding activation signal of the NC/PLC interface:

- "TRUE": Transmission
- "FALSE": No transmission

By default, the value of the parameter `MMCToIf` == "TRUE".

User-specific activation

A function may also be activated user-specifically with the PLC user program directly by setting the corresponding activation signal in the NC/PLC interface.

The FB1 parameter `MMCToIf` must thus be set to "FALSE", otherwise the NC/PLC interface would be overwritten with the values of the HMI/PLC interface.

Feedback

An acknowledgment is returned for some functions upon activation (refer to the following table).

NC/PLC interface signals

Table 10-2 Program control: Interface signals

Function	Selection (HMI → PLC)	Activation (PLC → NC)	Feedback (NC → PLC)
"Skip block" (SKP) /0 - /7	DB21, ... DBX26.0 - 7	DB21, ... DBX2.0 - 7	---
"Skip block" (SKP) /8 - /9	DB21, ... DBX27.0 - 1	DB21, ... DBX31.6 - 7	
Dry run feed rate (DRY)	DB21, ... DBX24.6	DB21, ... DBX0.6	DB21, ... DBX318.6
"Rapid traverse override" (RG0)	DB21, ... DBX25.3	DB21, ... DBX6.6	---
Single block (SBLx)	Preselection of SBL1, SBL2 or SBL3 via program control display of HMI		
SBL1 "Single block main run"	Machine control panel (MCP): EB n + 5, bit 2	DB21, ... DBX0.4	---
SBL2 "Decoding single block"			
SBL3 "In cycle"			
Programmed stop (M01)	DB21, ... DBX24.5	DB21, ... DBX0.5	DB21, ... DBX32.5

Function	Selection (HMI → PLC)	Activation (PLC → NC)	Feedback (NC → PLC)
Associated help function (M-1)	DB21, ... DBX24.4	DB21, ... DBX30.5	DB21, ... DBX318.5
Handwheel offset (DRF)	DB21, ... DBX24.3	DB21, ... DBX0.3	DB21, ... DBX33.3
Program test (PRT)	DB21, ... DBX25.7	DB21, ... DBX1.7	DB21, ... DBX33.7

References

- NC Variables and Interface Signals List Manual
- Operating Manual, HMI Advanced "Machine operating area"

10.14.2 Activation of skip levels

Function

It is possible to skip blocks which are not to be executed every time the program runs. Blocks to be skipped are indicated in the part program by the character "/" before the block number.

The skip levels in the part program are specified by "/0" to "/9".

Only one skip level can be specified for each part program block.

Parameterization

The number of skip levels is defined using machine data:

MD51029 $MM_MAX_SKP_LEVEL (max. number of skip levels in the NC program)

Programming

Blocks which are not to be executed in every program pass (e.g. program test blocks) can be skipped according to the following schematic.

Program code	Comment
`/N005`	`; Block skipped, (DB21, ... DBX2.0) 1st skip level`
`/0 N005`	`; Block skipped, (DB21, ... DBX2.0) 1st skip level`
`/1 N010`	`; Block skipped, (DB21, ... DBX2.1) 2nd skip level`
`/2 N020`	`; Block skipped, (DB21, ... DBX2.2) 3rd skip level`
`/3 N030`	`; Block skipped, (DB21, ... DBX2.3) 4th skip level`
`/4 N040`	`; Block skipped, (DB21, ... DBX2.4) 5th skip level`
`/5 N050`	`; Block skipped, (DB21, ... DBX2.5) 6th skip level`
`/6 N060`	`; Block skipped, (DB21, ... DBX2.6) 7th skip level`
`/7 N070`	`; Block skipped, (DB21, ... DBX2.7) 8th skip level`
`/8 N080`	`; Block skipped, (DB21, ... DBX31.6) 9th skip level`
`/9 N090`	`; Block skipped, (DB21, ... DBX31.7) 10th skip level`

Activation

The 10 skip levels "/0" to "/9" are activated by the PLC by setting the PLC → NC interface signals.

The function is activated from the HMI via the "Program control" menu in the "Machine" operating area:

- For skip levels "/0" to "/7":
  Via the interface HMI → PLC DB21, ... DBB26 (skip block selected).
- For skip levels "/8" to "/9":
  Via the interface HMI → PLC DB21, ... DBX27.0 to DBX27.1.

References:
Operating Manual

Note

The levels to be skipped can only be changed when the control is in the STOP/RESET state.

10.14.3 Adapting the size of the interpolation buffer

MD28060

The channel-specific interpolator executes prepared blocks from the interpolation buffer during the part program run. The maximum number of blocks requiring space in the interpolation buffer at any given point in time is defined by the memory configuring MD28060 $MM_IPO_BUFFER_SIZE (number of NC blocks in the IPO buffer (DRAM)). For some applications it may be meaningful not to use the full buffer capacity in order to minimize the "interval" between preparation and interpolation.

SD42990

The number of blocks in the interpolation buffer can be restricted dynamically to a **smaller** value than in MD28060 $MC_MM_IPO_BUFFER_SIZE (number of NC blocks in the IPO buffer (DRAM)), minimum 2 blocks, with the setting data SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER (max. number of blocks in the IPO buffer).

Values of setting data SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER:

Value	Effect
< 0	No interpolation buffer limit active. The max. possible IPO buffer as set in MD 28060: MM_IPO_BUFFER_SIZE is activated.
or 1	The minimum permissible IPO buffer with two blocks is activated.
< MM_IPO_BUFFER_SIZE	The IPO buffer is activated with no more than the maximum specified number of blocks.
>= MM_IPO_BUFFER_SIZE	The IPO 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 retained at the end of the program. 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 the IPO buffer to
                                              five NC blocks
N110 ...
N120 ...
............
N200 $SC_MAX_BLOCKS_IN_IPOBUFFER = -1         ; Cancellation of the IPO buffer
                                              limitation
N210 ...
............
```

10.14.4 Program display modes via an additional basic block display

Basic block display (only for ShopMill/ShopTurn)

A second so-called basic block display can be used with the existing block display to show all blocks that produce an **action on the machine**.

LookAhead basic block display

The actually approached end positions are shown as an absolute position. The position values refer either to the workpiece coordinate system (WCS) or the settable zero system (SZS).

The number of LookAhead display blocks stored in the display buffer depends on the number of prepared blocks in the NC 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, ASUP start), the display blocks stored for LookAhead are deleted and preprocessed again afterwards.

Processed values

Values processed in the basic block display coincide with the:

- Selected tools
- Feedrate and spindle speed
- Actually approached position values
  Exceptions:
  With active tool radius compensation, deviations can occur.
  For modulo axes, the programmed value is displayed in the basic block display. This value can also lie outside the modulo range.

Note

Generally the positions are represented in the WCS or the SZS.

The basic block display can be activated or deactivated with setting data

SD42750 $SC_ABSBLOCK_ENABLE.

10.14.5 Basic block display for ShopMill/ShopTurn

Configure basic block display

The basic block display can be configured via the following machine data:

NC machine data for basic block display	Meaning:
MD28400 $MC_MM_ABSBLOCK	Activate basic block display
MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[2]	Size of the display buffer
Machine data display	Position values to be set:

MD9004 $MM_DISPLAY_RESOLUTION	for metric dimensional notation
MD9011 $MM_DISPLAY_RESOLUTION_INCH	for imperial dimensional notation
MD9010 $MM_SPIND_DISPLAY_RESOLUTION	Coordinate system to be set for spindles display resolution
MD9424 $MM_MA_COORDINATE_SYSTEM	for WCS or SZS actual value display

This display machine data is copied to NC machine data
MD17200 $MN_GMMC_INFO_UNIT[0] (global HMI information) to MD17200 $MN_GMMC_INFO_UNIT[3]. This permits access from the NC to this display machine data.

Activation

The basic block display is activated by MD 28400 $MC_MM_ABSBLOCK (activate block display with absolute values) by means of Power On. If MD28400 $MC_MM_ABSBLOCK is set to 1, a channel-specific display buffer (FIFO) is created during power-up.

Size of display buffer (FIFO) = (MD28060 $MC_MM_IPO_BUFFER_SIZE (number of NC blocks in the IPO buffer) + MD28070 $MC_MM_NUM_BLOCKS_IN_PREP (number of blocks for the block preparation)) multiplied by 128 bytes. For the standard machine data setting, this represents a size of 6 KB.

Optimize the size of display buffer:
The memory requirement can be optimized by entering a value between 128 and 512. The display blocks prepared in the display buffer are transferred to the HMI via a configurable upload buffer.

The maximum size of the 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].

Supplementary conditions

If the length of a display block configured in MD28400 $MC_MM_ABSBLOCK is exceeded, this display block is truncated accordingly. To represent this, the "..." string is appended to the block end.

For preprocessed cycles
(MD10700 $MN_PREPROCESSING_LEVEL > 1 (program preprocessing level)), 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 calculation.
- Polar coordinate programming is not shown in Cartesian system.

Radius/diameter values

Diameter values shown in the basic block display and position display may be needed as a radius for internal calculation. These values for measurements in radius/diameter according to G group 29 can be manipulated using the following options:

- G command DIAMCYCOF (extension of channel-specific diameter programming)
  This G command deactivates the channel-specific diameter programming during the cycle execution. In this way, calculations in the cycle can always be made in the radius. The position display and the basic block display are continued according to the state of the diameter programming before DIAMCYCOF.
  In the basic block display, the last displayed value is retained.
- G command DIACYCOFA[AX] (axis-specific diameter programming)
  This G command deactivates the axis-specific diameter programming during the cycle execution. In this way, calculations in the cycle can always be made in the radius. In the position display and in the basic block display, this continues according to the state of the diameter programming before DIACYCOFA[AX] indicator.
  In the basic block display, the last displayed value is retained.
- MD27100 $MC_ABSBLOCK_FUNCTION_MASK (parameterize the block display with absolute values)

Bit0 = 1	Setpoints of the transverse axis are always displayed as diameter values in the basic block display.

Behavior for active compressor

Two display blocks are generated for active compressor with G group 30 not equal to COM. The

- first contains the G command of the active compressor, the
- second contains the string "..." as character for missing display blocks

Example:

```
G0 X10 Y10 Z10            ; block still prepared for the basic block display
COMPCAD                   ; compressor for optimized surface quality (CAD program)
...                       ; String as sign that the display blocks are missing
```

To prevent bottlenecks of the NC performance, the basic block display is disabled automatically. As a sign that the display blocks are missing, a display block with the string "..." is generated.
All display blocks are always generated in the single block.

10.14.6 Structure for a DIN block

Structure of display block for a DIN block

Basic structure of display block for a DIN block

- Block number/label
- G command of the first G group
(only if changed as compared to the last machine function block).
- Axis position
(sequence corresponding to MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in the channel)).
- Further modal G commands
(only if changed as compared to the last machine function block).
- Other addresses as programmed.

The display block for the basic block display is directly derived from the programmed part program blocks according to the following rules:

- Macros are expanded.
- Skip identifiers and comments are omitted.
- Block number and labels are transferred from the original block, but omitted if DISPLOF is active.
- The number of decimal places is defined in display machine data MD 9004, MD 9010 and MD 9011 via the HMI.

HMI display machine data	Access in the NC 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_RESOLU-TION	MD17200 $MN_GMMC_INFO_NO_UNIT[2]
MD9424 $MM_MA_COORDINATE_SYSTEM	MD17200 $MN_GMMC_INFO_NO_UNIT[3]

- Programmed axis positions are represented as absolute positions in the coordinate system (WCS / SZS) specified in MD9424 $MM_MA_COORDINATE_SYSTEM (coordinate system for actual value display)

Note

The modulo correction is omitted for modulo axes, which means that positions outside the modulo range can be displayed. It also means that the basic block display differs from the position display in which values are always modulo-converted.

Examples

Comparisons between display block (original block) and basic block display:

- **Programmed positions** are displayed as absolute.
  The addresses AP/RP are displayed with their programmed values.

Original block:	Display block:
`N10 G90 X10.123`	`N10 X10.123`
`N20 G91 X1`	`N20 X11.123`

- **Address assignments** (non-DIN addresses) are displayed in the form <address> = <constant>.

Original block:	Display block:
`N110 R1 = -67.5 R2 = 7.5`	
`N130 Z = R1 RND = R2`	`N130 Z-67.5 RND = 7.5`

- **Address indices** (address extensions) are displayed as constants <address> [<constant>] = <constant>.

Original block:	Display block:
`N220 DEF AXIS AXIS_VAR = X`	
`N240 FA[ AXIS_VAR] = R2`	`N240 FA[X] = 1000`

- **DIN addresses without address extension** are displayed in the form <din_address> <constant>.

Original block:	Display block:
`N410 DEF REAL FEED = 1.5`	
`N420 F = FEED`	`N420 F1.5`

The following applies for **H functions**: Each programmed value is display irrespective of the output type to the PLC.
(MD22110 $MC_AUXFU_H_TYPE_INT (type of H auxiliary function is integer)).

- For **Tool selection by tool command**
  Display information is generated in the form T<value> or T=<string>. If an address extension has been programmed, this is displayed as well.
  If several spindles have been configured or the "Tool change via master toolholder" function (MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER (toolholder number)) is active, the T number is always output with address extension.
  If no address extension has been programmed, the number of the master spindle or the master toolholder is used instead (T<spindle_number/tool_holder>=).
- For the **Spindle programming** via S, M3, M4, M5, M19, M40 - M45 and M70 (or MD 20094 $MC_SPIND_RIGID_TAPPING_M_NR (M function for switching over in the controlled axis operation)) the following regulation applies regarding the address extension:
  If an address extension has been programmed, then this is also resolved.
  If several spindles have been configured, then the address extension is also output.
  If no address extension has been programmed, the number of the master spindle is used (S<spindle_number>=).
- **Indirect G command programming** in form G[<group>] = <expression> is substituted by the corresponding G command.

Original block:	Display block:
N510 R1=2	
N520 G[8]= R1	N520 G54

- **Modal G commands** that do not generate an executable block are collected and output with the display block of the next executable block if permitted by the syntax (DIN block). If this is not the case (e.g. predefined subprogram call TRANSMIT), a separate display block containing the modified G commands is placed in front of the next executable block.

Original block:	Display block:
N610 G64	G64
N620 TRANSMIT	N620 TRANSMIT

- A display block is always generated for **part program lines** in which the addresses **F and FA** appear (including for MD22240 $MC_AUXFU_F_SYNC_TYPE = 3 (output time of the F functions)).

Original block:	Display block:
N630 F1000	N630 F1000
N640 X100	N640 X100

- The **display blocks generated for the block display** are derived **directly** from the programmed part program blocks. If intermediate blocks (e.g. tool radius compensation G41/G42, radius/chamfer RNDM, RND, CHF, CHR) are generated in the course of contour preprocessing, these are assigned the display information from the part program block on which the motion is based.

Original block:	Display block:
`N710 Y157.5 G42`	`N710 Y157.5 G42`
`N720 Z-67.5 RND=7.5`	`N720 Z-67.5 RND=7.5`

- With the **EXECTAB command** (processing a table of contour elements), the block generated by EXECTAB is shown in the display block.

Original block:	Display block:
`N810 EXECTAB (KTAB[5])`	`N810 G01 X46.147 Z-25.38`

- With the **EXECSTRING command**, the block generated via EXECSTRING is displayed in the display block.

Original block:
`N910 DEF STRING[40] PROGSTRING = "N905 M3 S1000 G94 Z100 F1000 G55"`
`N920 EXECSTRING(PROGSTRING)`

Original block:
`N905 Z100 G55 G94 M3 S1000 F1000`

10.14.7 Execution from external

Function

The "execution from external source" function can be used to execute programs from an external program memory that cannot be saved in the NC memory due to their size.

Note

Protected cycles (_CPF files) can **not** be executed from an external program memory.

External program memory

External program memory can be found on the following data carriers:

- Local drive
- Network drive
- USB drive

Note

Only the USB interfaces on the operator panel front or the TCU can be used as interface for the processing of an external program on a USB drive.

NOTICE

Tool/workpiece damage caused by the USB FlashDrive

It is recommended that a USB-FlashDrive is not used for the execution of an external subprogram. A communication interruption to the USB FlashDrive during the execution of the subprogram due to contact problems, failure, abort through trigger or unintentional unplugging, results in an immediate machining stop. The tool and/or workpiece could be damaged.

Applications

- **Direct execution from external programs**
  In principle, any program that is accessible via the directory structure of the interface in the "Execution from external" HMI mode can be selected and executed.
- **Execution of external subprograms from the part program**
  The external subprogram is called through the part program command `EXTCALL` with specification of a call path (optional) and the subprogram name (→ see "Executing external subprograms (EXTCALL) (Page 631)").

Parameter assignment

A reloading memory (FIFO buffer) must be reserved in the dynamic NC memory for executing a program in "execution from external source" mode.

Size of FIFO buffer

The size of the FIFO buffer is set in the machine data:

MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE (FIFO buffer size for processing from external)

Note

Programs with jump commands

For external programs that contain jump commands (`GOTOF`, `GOTOB`, `CASE`, `FOR`, `LOOP`, `WHILE`, `REPEAT`, `IF`, `ELSE`, `ENDIF` etc.) the jump destinations must lie within the post loading memory.

Note

ShopMill/ShopTurn programs

The contour descriptions added at the file end mean the ShopMill and ShopTurn programs must be stored completely in the read-only memory.

Number of FIFO buffers

One FIFO buffer must be provided for each one of the programs that are executed simultaneously in "execution from external source" mode.

The number of FIFO buffers is set in the machine data:

MD18362 $MN_MM_EXT_PROG_NUM (number of externally executed program levels executable simultaneously)

Behavior on reset, power-on

External program calls are aborted using reset and power-on and the particular FIFO buffers are erased.

A main program selected from an external program memory is selected again automatically after a power-on if the same program memory is still available and execution of EXTCALL calls has been activated in MD9106 $MM_SERVER_EXTCALL_PROGRAMS.

10.14.8 Executing external subprograms (EXTCALL)

Function

Individual machining steps for producing complex workpieces may involve program sequences that require so much memory that they cannot be stored in the NC memory.

In such cases, the user has the option of executing the program sequences as subprograms from an external program memory in the "Execution from external source" mode with the help of the `EXTCALL` part program command.

Preconditions

The following preconditions are applicable to the execution from external subprograms:

- The subprograms must be accessible via the directory structure of the operator interface.
- A reloading memory (FIFO buffer) must be reserved for each subprogram in the dynamic NC memory.

Note

Subprograms with jump commands

For external subprograms that contain jump commands (`GOTOF`, `GOTOB`, `CASE`, `FOR`, `LOOP`, `WHILE`, `REPEAT`, `IF`, `ELSE`, `ENDIF` etc.) the jump destinations must lie within the post loading memory.

The size of the post loading memory is set via:

MD18360 MM_EXT_PROG_BUFFER_SIZE

ShopMill/ShopTurn programs

The contour descriptions added at the file end mean the ShopMill and ShopTurn programs must be stored completely in the read-only memory.

Parameterization

The path for the external subprogram directory can be preset using setting data:

SD42700 $SC_EXT_PROG_PATH (program path for the EXTCALL external subprogram call)

The entire path of the program to be called along with the subprogram path or name specified during programming is derived therefrom.

Programming

An external subprogram is called by means of parts program command `EXTCALL`.

Syntax: `EXTCALL("<path/><program name>")`

Parameter:

`<path>`:	Absolute or relative path data (**optional**) Type: STRING
`<program name>`:	The program name is specified without prefix "_N_". The file extension ("MPF", "SPF") can be attached to program names using the "_" or "." character (**optional**). Type: STRING

Note

Path specification: Short designations

The following short designations can be used to specify the path:

- **LOCAL_DRIVE:** for local drive
- **CF_CARD:** for CompactFlash Card
- **USB:** for USB front connection

CF_CARD: and **LOCAL_DRIVE:** can be alternatively used.

EXTCALL call with absolute path name

If the subprogram exists at the specified path, it will be executed following the `EXTCALL` call. If it does not exist, program execution is canceled.

EXTCALL call with relative path name / without path name

In the event of an `EXTCALL` call with a relative path name or without a path name, the available program memories are searched as follows:

- If a path name is preset in SD42700 $SC_EXT_PROG_PATH, the data specified in the `EXTCALL` call (program name or with relative path name) is searched for first, starting from this path. The absolute path results from linking the following characters:
  - The path name preset in SD42700
  - The "/" character as a separator
  - The subprogram path or name programmed in `EXTCALL`
- If the called subprogram is not found at the preset path, the data specified in the `EXTCALL` call is then searched for in the user-memory directories.
- The search ends when the subprogram is found for the first time. If the search does not produce any hits, the program is canceled.

Example

Execute from local drive

Main program:

Program code
`N010 PROC MAIN`
`N020 ...`
`N030 EXTCALL ("ROUGHING")`
`N040 ...`
`N050 M30`

External subprogram:

Program code
`N010 PROC ROUGHING`

Program code
`N020 G1 F1000`
`N030 X= ... Y= ... Z= ...`
`N040 ...`
`...`
`...`
`N999999 M17`

The "MAIN.MPF" main program is stored in NC memory and is selected for execution.

The "SCHRUPPEN.SPF" or "SCHRUPPEN.MPF" subprogram to be subsequently loaded is on the local drive in the directory "/user/sinumerik/data/prog/WKS.DIR/WST1.WPD".

The subprogram path is preset in SD42700:

SD42700 $SC_EXT_PROG_PATH = "LOCAL_DRIVE:WKS.DIR/WST1.WPD"

Note

Without the path being specified in the SD42700, the `EXTCALL` operation for this example would have to be programmed as follows:

```
EXTCALL("LOCAL_DRIVE:WKS.DIR/WST1.WPD/SCHRUPPEN")
```

10.15 Execution from external storage (EES) (option)

10.15.1 Function

Note

To use the function, the "Expanded CNC user storage" or "Execute from external storage (EES)" licensed option is required!

Function

Using the EES (Execution from External Storage) function, users have the option of having the NCK execute programs **directly from an external program storage**.

The following drives are available as external memory:

Drive	Symbolic name [1]	SINUMERIK 840D sl	SINUMERIK 828D
NC Extend (previously "Lo-cal drive")	CF_CARD LOCAL_DRIVE SYS_DRIVE	CF card of NCU or Local hard disk of a PCU	System CF card of the PPU
USER CF	CF_CARD LOCAL_DRIVE	-	User CF card of the PPU
Network drive			
Statically managed USB drive			

[1] For SINUMERIK 840D sl, the symbolic names LOCAL_DRIVE, CF_CARD, and SYS_DRIVE are permanently assigned to the drive NC Extend (⇒ NC Extend can be addressed via LOCAL_DRIVE, CF_CARD, and SYS_DRIVE).
For SINUMERIK 828D, the assignment of the symbolic names LOCAL_DRIVE, CF_CARD and SYS_DRIVE to NC Extend can be configured. In this way, the system CF card of the PPU can also be addressed via the symbolic names LOCAL_DRIVE and CF_CARD if, for example, no USER CF drive exists.

NOTICE

Tool/workpiece damage caused by the USB FlashDrive

A USB FlashDrive **cannot** be recommended when executing an external program. A communication abort to the USB FlashDrive during the execution of the program due to contact problems, failure, abort through trigger or unintentional unplugging, results in an uncontrolled stop of the machining. The tool and/or workpiece could be damaged.

Requirements

The following prerequisites apply when using EES:

- The "Expanded CNC user storage" or "Execute from external storage (EES)" licensed option must be set.
- The drives, which are used at the control as external memory, must be configured as a logical drives (see "Commissioning (Page 637)").

Operating mode

Depending on the option available and the drive configuration, various EES operating modes are possible. The active operating mode of a control is displayed using machine data MD18045 $MN_EES_MODE_INFO.

MD18045	Operating mode	Option	External memory	
			SINUMERIK 840D sl	SINUMERIK 828D
= 0	EES not active	-	-	
= 1	Local EES active	6FC5800-0AP77-0YB0 CNC user memory expanded	The use of EES on an NCU is limited to the expanded user memory (100 MB) of the CF card. If EES is used on a PCU, the entire free memory can be used by NC Extend.	The use of EES is limited to the expanded user storage (100 MB) of the system CF card.
			EES via network or USB is not possible.	
		6FC5800-0AP12-0YB0 additional HMI user memory on CF card of NCU.	If EES is used on an NCU, with the additional option, the local user memory can be expanded up to 6 GB (depending on MD9111). If EES is used on a PCU, the additional option is not required.	-
= 2	Global EES active	6FC5800-0AP75-0YB0 Execution from external storage (EES)	EES can be used for all available external storage.	EES can be used for all available external storage. **Note:** EES via network additionally requires enabling the HMI function "Network drive management" (MD19730 Bit 2 = 1).
		6FC5800-0AP12-0YB0 additional HMI user memory on CF card of NCU.	If EES is used on an NCU, with the additional option, the local user memory can be expanded up to 6 GB (depending on MD9111). If EES is used on a PCU, the additional option is not required.	-
= 5	Local EES active + Global part program memory	The same as MD18045 = 1, only that on the expanded user memory a global part program memory (Page 639) is set up.		
= 6	Global EES active + Global part program memory	The same as MD18045 = 2, only that on an external user memory a global part program memory (Page 639) is set up.		

Properties

The EES function can replace the "Execution from external (Page 629)" and "Executing external subprograms (EXTCALL) (Page 631)" functions.

The EES function has the following advantages:

- Standard program handling throughout the system
- No restrictions regarding the commands that can be used
  The restrictions on "execution from external source" and "execution of external subprograms (EXTCALL)", e.g. no backward jumps, limitation of the jump distance of jump commands by the size of the reload memory, are eliminated with EES.
- Programs can be moved between different program storages (NC, GDIR, external drive) significantly easier.
- There are practically no restrictions regarding the part program size and the number of programs (this is only limited by the capacity of the external data memory).
- Uniform syntax for the subprogram call, independent of the storage location of the subprogram (an EXTCALL call is not required).
- Network drives can be used jointly by more than one station (PCU/NCU). Prerequisite is a uniform drive configuration for these stations. This results in a uniform view of the programs for all stations.
- Because of the uniform view of the external program storage for all stations, changes to the programs stored there consistently apply to all stations.

10.15.2 Commissioning

10.15.2.1 Configuring the drives

To use the EES function, the drives used with the control system must be declared.

References:

- Commissioning Manual, base software and operating software
- Universal/Turning/Milling/Grinding Operating Manual

After activating the new drive configuration, the programs can be freely distributed among the available drives.

Note

All the previous drives may no longer be available in the newly created drive configuration. Access to the programs on these drive is then no longer possible.

Remedy: First copy the programs stored on these drives to a drive that can still be accessed.

Note

Because protected cycles (_CPF files) are only executed from the NC part program memory for system reasons, they **cannot** be stored for execution on an external program memory.

The previous NC part program memory with the MPF.DIR, SPF.DIR and WKS.DIR directories is not absolutely necessary when using EES. A system can also be configured without using the NC part program memory.

NOTICE

Executing programs that are not visible

Even if the NC part program memory was removed from the drive configuration, it is always still available in the system itself. This especially means that when executing the program it is possible that programs that still exist there are inadvertently executed from the SPF.DIR.

Remedy: If the system is configured without NC part program memory in the drive configuration, any programs still stored there should be deleted manually.

If the NC part program memory is still to be used, then it should not be completely removed from the system, only assigned an appropriate protection level when required.

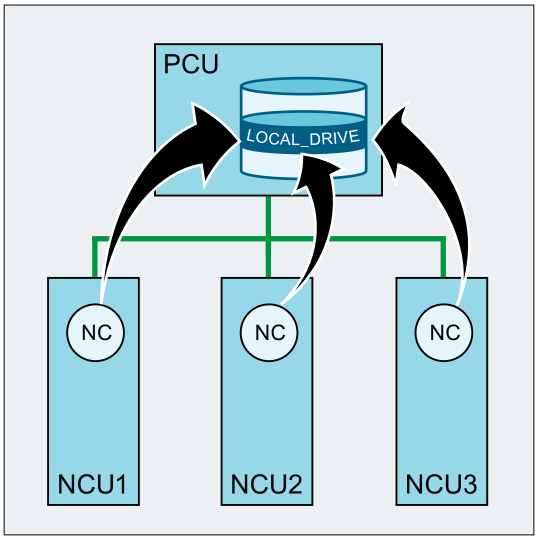
Drives can be shared by more than one station (PCU/NCU). A standard drive configuration for these stations means that there is a standard program view that is independent of the particular station.

Note

The CF card of an NCU/PPU **cannot** be used by several stations.

Example:

Several NCUs jointly use a program memory on the local hard disk of the PCU (LOCAL_DRIVE).



Note

If external program memories are used together at different stations in the EES mode, then the following rules must be observed.

- A program **cannot** be simultaneously edited from several stations.
- Programs, which are being executed, can **no** longer be changed.

SINUMERIK 840D sl only

For operation with an external HMI, the drives must be configured on the external HMI! The drive configuration (logdrive.ini) must be loaded into the corresponding NCU from the external HMI. A softkey is available for the transfer on the dialog for the drive configuration.

In systems in which multiple NCs work together, the drive configuration must be identical for all NCs. This is achieved by distributing the logdrive.ini file to all of the NCUs listed in the mmc.ini file. The configurations existing there are therefore overwritten.

10.15.2.2 Global part program memory (GDIR)

When declaring the drives, one of the drives can be designated the global part program memory (GDIR).

References:
Operating Manual; Section: "Managing programs" > "Setting up drives"

The system automatically creates the MPF.DIR, SPF.DIR and WKS.DIR directories on the drive that acts as the GDIR. These three directories form the GDIR.

The GDIR only plays a role for the EES function. Depending on the drive configuration, the GDIR replaces or extends the NC part program memory. However, it is not mandatory to set up a GDIR for the EES operation.

The directories and files of the GDIR can be addressed in the part program in the same way as in the passive file system. Therefore, a compatible relocation of an NC program with path specification is possible for the passive file system to the GDIR.

The GDIR extends the search path for subprograms, which are called without specifying an absolute path.

GDIR replaces the NC part program memory

If the NC part program memory is completely empty in the MPF.DIR, SPF.DIR and WKS.DIR directories, then the GDIR replaces the NC part program memory. The previous NC search path is emulated 1:1 by the GDIR.

Selecting the main program on an external archive/data storage medium

Search sequence for the subprograms:

1. Actual directory on an external archive/data storage medium
2. SPF.DIR in the GDIR memory

3. The drive referenced using CALLPATH
4. Cycles

GDIR extends the NC part program memory

When the NC part program memory is not empty in the MPF.DIR, SPF.DIR and WKS.DIR directories, then the search sequence for the subprograms depends on where the main program is archived (active directory).

Selecting the main program in the NC part program memory (MPF.DIR or xxx.WPD in WKS.DIR)

Search sequence for the subprograms:

1. Actual directory in the NC part program memory
2. SPF.DIR in the NC part program memory
3. The drive referenced using CALLPATH
4. Cycles

Selecting the main program on an external archive/data storage medium released for EES

Search sequence for the subprograms:

1. Actual directory on an external archive/data storage medium
2. SPF.DIR in the NC part program memory
3. SPF.DIR in GDIR
4. The drive referenced using CALLPATH
5. Cycles

Note

To define the search sequence, also see MD11625 $MN_FILE_ONLY_WITH_EXTENSION and MD11626 $MN_CYCLES_ONLY_IN_CYCDIR!

Note

An external drive can also be referenced using the CALLPATH statement.

10.15.2.3 Settings for file handling in the part program for EES

Program names unique throughout the system

If external program memories are used together at different stations in EES mode, file operations performed simultaneously on different stations (`WRITE`, `DELETE`, …) result in access conflicts. To avoid such access conflicts, we recommend setting up a name space for file names on each station that is unique throughout the system.

Name space unique throughout the system

A name space for file names that is unique throughout the system is achieved, for example, by associating the file names with the EES-specific name of the NC parameterized in the machine data and the channel number of the channel in which the program is executed. For example, when the program is executed, the following code generates the file name (MYFILE _NC1_1.SPF), which is unique throughout the system, by appending the EES-specific name of the NC (NC1) and the channel number (channel 1) to the program name.

```
$MN_EES_NC_NAME="NC1"
N10 DEF STRING[31] FILENAME
N20 FILENAME="MYFILE_" << $MN_EES_NC_NAME << "_" << $P_CHANNO <<
".SPF"
```

Parameterization

The EES-specific name of the NC is set in the NC-specific machine data:

MD10125 $MN_EES_NC_NAME = <NC name>

Note

Name of the NC unique throughout the system

To avoid access conflicts, the EES-specific name of the NC must be unique throughout the system. The responsibility for this resides exclusively with the user/machine manufacturer.

When calling the program, only search for files with file ID

In order to speed up the program search for subprogram calls during EES operation, we recommend that you limit the search to files **with** file ID (e.g. SPF, MPF, etc.).

MD11625 $MN_FILE_ONLY_WITH_EXTENSION = **1**

Note

MD11625 has no effect on the program search when processing external subprograms with EXTCALL.

Reference:
Description of the search path for the subprogram call, see the Programming Manual Job Planning.

Only search for programs with interface in the cycle directories

In order to speed up the program search for subprogram calls during EES operation, we recommend the search for subprograms which have created an interface description (by means of PROC statement), and whose interface description was generated from the one of the cycle directories (CUS, CMA, CST) be limited to the cycle directories:

MD11626 $MN_CYCLES_ONLY_IN_CYCDIR = **1**

Note

MD11626 has no effect on subprograms whose interface was created using an EXTERNAL declaration. A search is made in all program directories.

NOTICE
No search success for cycles outside the cycle directories
Cycles in the current directory and global subprogram directory are no longer found with the setting MD11626 = 1!
Remedy: Always store cycles in the cycle directories.

10.15.2.4 Memory configuration

Reducing the end user program memory in the passive file system

With active EES, the end user program memory in the passive file system can be reduced:

MD18352 $MN_MM_U_FILE_MEM_SIZE (end user memory for part programs / cycles / files)

The released memory can then be used, for example, for tool data or manufacturer cycles (MD18353 $MN_MM_M_FILE_MEM_SIZE).

References:
For detailed information on the memory configuration, see Function Manual, Extended Functions

Releasing reload memory

The EES function can replace the "Execution from external source" and "Execution of external subprograms (EXTCALL)" functions.

In order to execute subprograms from part programs with EXTCALL calls with EES instead of the "Execution of external subprograms (EXTCALL)" function, the EXTCALL calls must be changed to CALL calls, and the path specifications adapted if required.

After a complete changeover, the reload memory (FIFO buffer) required for the "Execution from external source" and "Execution of external subprograms (EXTCALL)" can be released:

MD18362 $MN_MM_EXT_PROG_NUM (number of externally executed program levels executable simultaneously) = 0

10.15.3 Supplementary conditions

Teach in

In the EES mode, it is **not** possible to use the "Teach In" function in the AUTOMATIC operating mode.

10.16 Process Datashare - output to an external device/file

10.16.1 Function

With the "Process DataShare" function, it is possible to write data from a part program to an external device or to an external file; for instance, to log production data or to control additional equipment at a control system.

Availability

The function is available:

- Only in the real NC (**not** in the SNC and VNC simulation software).
- Only in part programs (**not** in synchronized actions).
- Parallel in all machining channels of the NC for all available (configured) output devices.

External devices/files

External devices/files can be:

- Files on the local CF card
  Local CF card is the memory referred to from the HMI using the symbolic identifier LOCAL_DRIVE. For SINUMERIK 840D sl this is the local drive, for SINUMERIK 828D, the user CF card.

  Note

  For SINUMERIK 840D sl, the option "Additional xxx MB HMI user memory on CF card of the NCU" is required for output to the LOCAL_DRIVE device. For SINUMERIK 828D a user CF card must be available and an option is not required.

- Files on a network drive
- V.24 interface

  Note

  For SINUMERIK 840D sl, the NCU option module RS232 interface is required for output at the V.24 interface. For SINUMERIK 828D output is realized at the integrated V.24 interface (precondition: MD51233 $MNS_ENABLE_GSM_MODEM = 0).

Maximum number of opened external devices

More than one external device/file can also be assigned in a part program / channel. A maximum of ten output devices can be simultaneously opened across all NC channels. In addition, there are two entries reserved for Siemens cycles.

A maximum of five jobs may be active simultaneously to the output devices.

Usage mode

For each output device, when opening the device, it can be specified as to whether the device is to be exclusively used by just one channel or whether several channels can output to the device ("shared" mode).

Behavior for part program end / channel reset

With part program end and channel reset, all of the external devices/files that have been opened in the channel are closed.

Use of the function for data transfer to the control

NOTICE

Data security

If the Process Datashare function is used to send data from an external device via the Ethernet-X130 interface to the control, there is the possibility that the data on the control could be falsified by a third-party and is no longer consistent. When using the function, ensure that the network is protected against access by third-parties.

10.16.2 Commissioning

The external devices to be used are configured in the /oem/sinumerik/nck/extdev.ini or /user/sinumerik/nck/extdev.ini file. If both files are available, then the entries in the user area have priority. The file can be updated in the operating area COMMISSIONING under SYSTEMDATEN/CF card.

Note

It is not necessary to configure in the extdev.ini file when using LOCAL_DRIVE and CYC_DRIVE. The two devices are always available as soon as the corresponding option is set or the user CompactFlash card is available.

The external devices to be used are defined/listed in the section [ExternalDevices] of the extdev.ini file. A serial device (/dev/v24) and up to nine files or directories (/dev/ext/1…9) can be specified as device. The Linux notation is used when specifying devices. Lines, which start with ";", are comments and are overread.

With the exception of /dev/v24, the devices can be declared as directory path - terminated with an attached "/" – or as file path – i.e. with attached fully qualified path, ending with a file name (without a terminating "/"). When used in a part program, a file name (path) must also be specified for a device with directory path.

Except for /dev/v24, a device is defined using the three items separated by a comma for "Server", "Path" and the optional "Write mode".

For the files or directories (this then applies to all files in the directory), it can be specified as to whether the file should be overwritten after it has been opened ("O" = Overwrite) or whether the outputs should be attached to the file ("A" = Append). The default value is "A". A file/directory that does not exist is newly created when opening.

For the device V.24 interface, only the settings for baud rate, data bits, stop bits, parity, protocol and possibly end are specified in this sequence.

For files that are generated/saved on the LOCAL_DRIVE, LOCAL_DRIVE_MAX_FILESIZE data can be used to set a maximum file size in bytes - this is then valid as standard for all files. The file size is checked when executing an EXTOPEN command in Append mode. Optionally, the write mode ("O" = Overwrite, "A" = Append) can be defined using the LOCAL_DRIVE_FILE_MODE data. The default value is "A".

Note

A copy template for the extdev.ini configuration file is available in the /siemens/sinumerik/nck directory.

Note

Changes to the extdev.ini file only become effective after an NC restart/boot.

Note

USB devices

For SINUMERIK 828D, "usb" (without partition data!) can also be defined as target for a USB device inserted at the front. The device at the USB can be addressed from the part program only directly using a symbolic device identifier "/dev/ext/x".

For SINUMERIK 840D sl, only statically connected USB interfaces of a TCU can be configured as USB devices. The configuration is realized using the SERVER:/PATH type as specification for "Server" in the sense above, whereby SERVER is the TCU name and /PATH designates the USB interface. The USB interfaces of a TCU are addressed using "dev0-0", "dev0-1", "dev1-0". The path data always starts with "/Partition", whereby the partition can be specified using a two-digit partition number or its partition name – and where required, is extended with a file path up to the required target, e.g.:

/dev/ext/8 = "TCU4:/dev0-0, /01/, A"

/dev/ext/8 = "TCU4:/dev0-0, /01/mydir.dir/"

/dev/ext/8 = "TCU4:/dev0-0, /myfirstpartition/Mydir.dir/myfile.txt, O"

Example

[ExternalDevices]

```
; Comment line
; example for V24
; /dev/v24 = "9600, 8, 1, none, rts [, etx]"
; examples for network drives
; /dev/ext/1 = "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE/, /, A"
; /dev/ext/2 = "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE, /myfile.txt, O"
; /dev/ext/3 = "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE, /mydir/, A"
; /dev/ext/4 = "SERVER:/dev0-0, /01/, A"
; …
; SINUMERIK 828 only (USB)
; /dev/ext/9 = "usb, / [ , O]"
; default: Partition number = 1

; SIEMENS only
; /dev/cyc/1= "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE, /mydir/, A"
; /dev/cyc/2= "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE/mydir, /, A"

LOCAL_DRIVE_MAX_FILESIZE = 50000
LOCAL_DRIVE_FILE_MODE = "O"
```

Effectiveness of the EXTOPEN parameter <WriteMode>

By specifying the write mode, when configuring in the extdev.ini file as well as for an EXTOPEN call, authorization conflicts can occur, which are then acknowledged with EXTOPEN - possibly with error:

Value from extdev.ini	**Value of the EXTOPEN parameter**		
	"OVR"	**"APP"**	**-**
"O"	O	Error	O
"A"	Error	A	A
-	O	A	A
	Explanation: O: "Overwrite" mode is active. A: "Append" mode is active. Error: EXTOPEN call is acknowledged with error.		

LOCAL_DRIVE: File attribute

The files created with EXTOPEN on LOCAL_DRIVE are allocated the following file attributes:

• Owner:	"user"	Read/write rights set
• Group:	"operator"	Read/write rights set

10.16.3 Programming

The writing of data from a part program to an external device/file is performed in three steps:

1. Open the external device/file
   The external device/file is opened for the channel for writing using the EXTOPEN command.
2. Writing data
   The output data can be processed using the string functions of the NC language, e.g. SPRINT. The WRITE command is used for writing.
3. Close the external device/file
   The external device/file assigned in the channel is released again using the EXTCLOSE command, when the end of the program is reached (M30) or for a channel reset.

Syntax

```
DEF INT <Result>
DEF STRING[<n>] <Output>
…
EXTOPEN(<Result>,<ExtDev>,<SyncMode>,<AccessMode>,<WriteMode>)
…
<Output>="data output"
WRITE(<Result>,<ExtDev>,<Output>)
…
EXTCLOSE(<Result>,<ExtDev>)
```

Meaning

`EXTOPEN`:	Pre-defined procedure to open an external device/file		
`<Result>`:	**Parameter 1:** Result variable By using the result variable value, it can be evaluated in the program as to whether the operation was successful and processing is then appropriately continued.		
	Type:	INT	
	Values:	0	No error
		1	External device cannot be opened
		2	External device is not configured
		3	External device with invalid path configured
		4	No access rights for external device
		5	Usage mode: External device already "exclusively" occupied
		6	Usage mode: External device already being "shared"
		7	File length longer than LOCAL_DRIVE_MAX_FILESIZE
		8	Maximum number of external devices has been exceeded
		9	Option for LOCAL_DRIVE not set
		11	V.24 interface has already been assigned with Easy-Message function (only 828D)
		12	Write mode: Data contradicts extdev.ini
		16	Invalid external path has been programmed
		22	External device not mounted

`<ExtDev>`:	**Parameter 2:** Symbolic identifier for the external device/file to be opened	
	Type:	STRING
	The symbolic identifier comprises: 1. the logical device name 2. where relevant, followed by a file path (attached using "/"). The following **logical device names** have been defined:	
	`"LOCAL_DRIVE"`:	Local CF card (pre-defined)
	`"CYC_DRIVE"`:	Reserved drive name for use in SIEMENS cycles (pre-defined)
	`"/dev/ext/1"`, ... `"/dev/ext/9"`:	Available network drives **Note:** It is necessary to configure in the extdev.ini file!
	`"/dev/cyc/1"`, `"/dev/cyc/2"`:	Reserved drive names for use in SIEMENS cycles **Note:** It is necessary to configure in the extdev.ini file!
	`"/dev/v24"`:	V.24 interface **Note:** It is necessary to configure in the extdev.ini file!
	File path: ● A file path must be specified for "LOCAL_DRIVE" and "CYC_DRIVE" e.g. "LOCAL_DRIVE/my_dir/my_file.txt" ● The logical device names "/dev/ext/1...9" and "/dev/cyc/1...2" can be configured: – To already refer to a file, in which case only the logical device names may be specified, e.g.: "/dev/ext/4" – Or to a directory, in which case a file path must be specified, e.g.: "/dev/ext/5/my_dir/my_file.txt" ● It is not permissible that a file path is attached to "/dev/v24".	
	Note: For the logical device names "/dev/ext/1...9", "/dev/v24" and "/dev/cyc/1...2" uppercase/lowercase is ignored; uppercase/lowercase is significant for specifying a path to a file. Only uppercase letters are permissible for "LOCAL_DRIVE" and "CYC_DRIVE".	

`<SyncMode>`:	**Parameter 3:** Processing mode for the WRITE commands to this device/file		
	Type:	STRING	
	Values:	`"SYN"`:	Synchronous writing Program execution is stopped until the write operation has been completed. Successfully completing the synchronous write operation can be checked by evaluating the error variables of the WRITE command.
		`"ASYN"`:	Asynchronous writing Program execution is not interrupted by the WRITE command. **Note.** In this mode, the result variable of the WRITE command does not provide any information and always has the value 0 (no error). In this particular mode, there is no certainty that the WRITE command was successful.
`<AccessMode>`:	**Parameter 4:** Usage mode for this device/file		
	Type:	STRING	
	Values:	`"SHARED"`:	Device/file is requested in the "shared" mode. Other channels can also use the device, i.e. also open in this mode.
		`"EXCL"`:	Device/file is exclusively used in the channel; no other channel can use the device.
`<WriteMode>`:	**Parameter 5:** Write mode for the WRITE commands to this file/device (optional)		
	Type:	STRING	
	Values:	`"APP"`:	Attaching The file is always kept regarding its contents; write calls are attached at the end.
		`"OVR"`:	Overwrite The contents of the file are deleted and re-generated using the subsequent write calls.
	Note: Using this parameter, the write mode configured in the extdev.ini file cannot be overwritten. In the case of a conflict, then the EXTOPEN call is acknowledged with error.		

`WRITE`:	Pre-defined procedure to write output data

`EXTCLOSE`:	Pre-defined procedure to close an external device/file that has been opened		
`<Result>`:	**Parameter 1:** Result variable		
	Type:	INT	
	Values:	0	No error
		16	Invalid external path has been programmed
		21	Error when closing the external device
`<ExtDev>`:	**Parameter 2:** Symbolic identifier for the external device/file description to be closed, see EXTOPEN! **Note:** The identifier must be identical to the identifier specified in the EXTOPEN call!		

Example

```
Program code
N10       DEF INT RESULT
N20       DEF BOOL EXTDEVICE
N30       DEF STRING[80] OUTPUT
N40       DEF INT PHASE
N50       EXTOPEN(RESULT,"LOCAL_DRIVE/my_file.txt","SYN","SHARED")
N60       IF RESULT > 0
N70          MSG("Error for EXTOPEN:" << RESULT)
N80       ELSE
N90          EXTDEVICE=TRUE
N100      ENDIF
…
N200      PHASE=4
N210      IF EXTDEVICE
N220         OUTPUT=SPRINT("End phase: %D",PHASE)
N230         WRITE(RESULT,"LOCAL_DRIVE/my_file.txt",OUTPUT)
N240      ENDIF
…
```

10.16.4 Supplementary conditions

Effect on continuous path mode

The EXTOPEN, WRITE and EXTCLOSE commands trigger a preprocessing stop and therefore interrupt the continuous path mode.

Behavior during block search

During "block search with calculation" with WRITE, no output is made. However, the EXTOPEN and EXTCLOSE commands are collected and set active with NC start after the search target was reached. The following WRITE commands therefore find the same environment as for the normal program processing.

For a block search with calculation in the (SERUPRO) "Program test" mode, EXTOPEN, WRITE and EXTCLOSE are executed just the same as for normal program processing.

10.17 System settings for power-up, RESET / part program end and part program start

Concept

The behavior of the control can be set via the machine data for the following events:

- Run-up (power-on)
- Reset / part program end
- Part program start

The control-system response after:	Can be set with:
Run-up (power on) *)	MD20110 $MC_RESET_MODE_MASK MD20144 $MC_TRAFO_MODE_MASK MD20150 $MC_GCODE_RESET_VALUES
RESET / part program end	MD20110 $MC_RESET_MODE_MASK MD20150 $MC_GCODE_RESET_VALUES MD20152 $MC_GCODE_RESET_MODE
Part program start	MD20112 $MC_START_MODE_MASK MD20110 $MC_RESET_MODE_MASK
*) see also POWER ON (Page 829)	

System settings after run-up

MD20110 $MC_RESET_MODE_MASK, bit 0 = 0 or 1

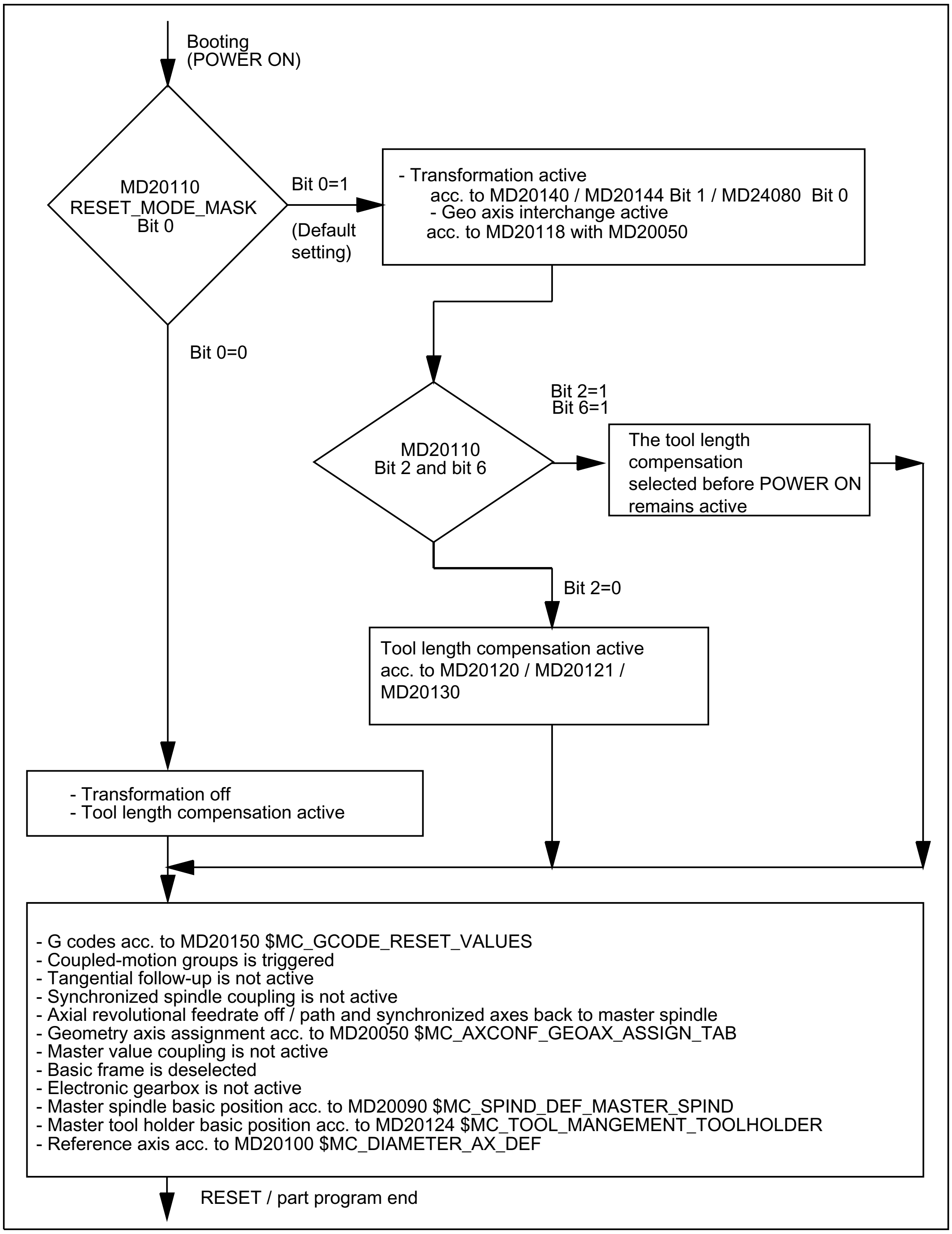

Booting
(POWER ON)
MD20110
RESET_MODE_MASK
Bit 0
Bit 0=1
(Default
setting)
- Transformation active
acc. to MD20140 / MD20144 Bit 1 / MD24080 Bit 0
- Geo axis interchange active
acc. to MD20118 with MD20050
Bit 0=0
Bit 2=1
Bit 6=1
MD20110
Bit 2 and bit 6
The tool length
compensation
selected before POWER ON
remains active
Bit 2=0
Tool length compensation active
acc. to MD20120 / MD20121 /
MD20130
- Transformation off
- Tool length compensation active
- G codes acc. to MD20150 $MC_GCODE_RESET_VALUES
- Coupled-motion groups is triggered
- Tangential follow-up is not active
- Synchronized spindle coupling is not active
- Axial revolutional feedrate off / path and synchronized axes back to master spindle
- Geometry axis assignment acc. to MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB
- Master value coupling is not active
- Basic frame is deselected
- Electronic gearbox is not active
- Master spindle basic position acc. to MD20090 $MC_SPIND_DEF_MASTER_SPIND
- Master tool holder basic position acc. to MD20124 $MC_TOOL_MANGEMENT_TOOLHOLDER
- Reference axis acc. to MD20100 $MC_DIAMETER_AX_DEF
RESET / part program end

Figure 10-8 System settings after run-up (power-on)

System settings after reset / part program end and part program start

MD20110 $MC_RESET_MODE_MASK, bit 0 = 0 or 1

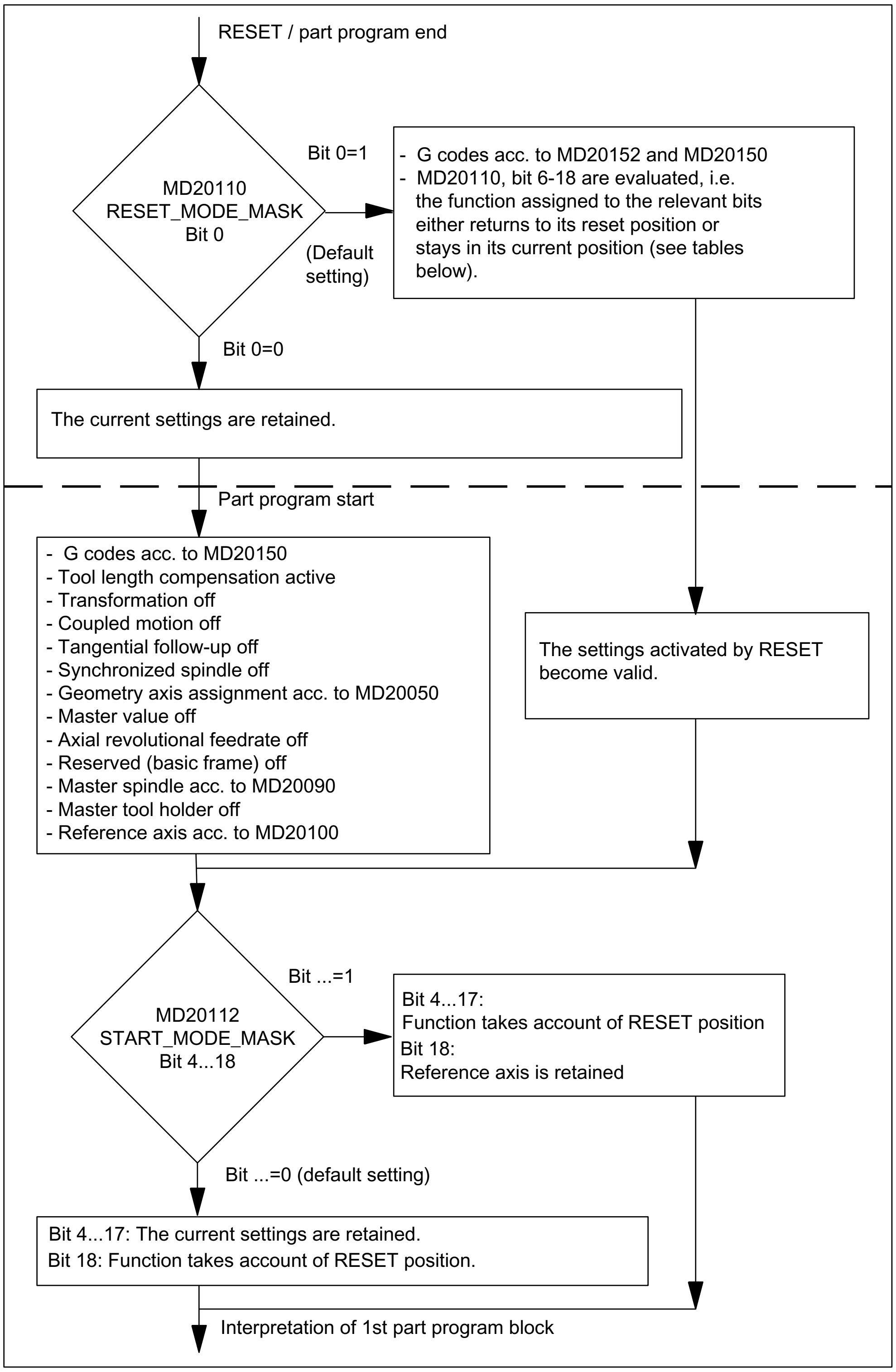

RESET / part program end
MD20110
RESET_MODE_MASK
Bit 0
Bit 0=1
(Default setting)
- G codes acc. to MD20152 and MD20150
- MD20110, bit 6-18 are evaluated, i.e. the function assigned to the relevant bits either returns to its reset position or stays in its current position (see tables below).
Bit 0=0
The current settings are retained.
Part program start
- G codes acc. to MD20150
- Tool length compensation active
- Transformation off
- Coupled motion off
- Tangential follow-up off
- Synchronized spindle off
- Geometry axis assignment acc. to MD20050
- Master value off
- Axial revolutional feedrate off
- Reserved (basic frame) off
- Master spindle acc. to MD20090
- Master tool holder off
- Reference axis acc. to MD20100
The settings activated by RESET become valid.
MD20112
START_MODE_MASK
Bit 4...18
Bit ...=1
Bit 4...17:
Function takes account of RESET position
Bit 18:
Reference axis is retained
Bit ...=0 (default setting)
Bit 4...17: The current settings are retained.
Bit 18: Function takes account of RESET position.
Interpretation of 1st part program block

Figure 10-9 System settings after reset / part program end and part program start

G command effective after run-up and reset / part program end

The G code that is effective in every G group after run-up (power-on) and reset / part program end is set in the following machine data:

MD20150 $MC_GCODE_RESET_VALUES[<**G group**>] = <default-G code>

MD20152 $MC_GCODE_RESET_MODE[<**G group**>] = <value>

Value	Description: Per G group
0	The default G command from MD20150 $MC_GCODE_RESET_VALUES takes effect.
1	The last active/current G command takes effect.

Control basic setting after run-up, reset / part program end and part program start

The control basic setting after run-up (power-on), reset / part program end and part program start is defined in the following machine data:

- MD20110 $MC_RESET_MODE_MASK (definition of the control basic setting after **run-up** and **reset / part program end**)
- MD20112 $MC_START_MODE_MASK (definition of the control basic setting after **part program start**)

References

Detailed Machine Data Description

Relevant machine data

Machine data	Meaning
MD20120 $MC_TOOL_RESET_VALUE	Tool length compensation during run-up, reset / part program end
MD20121 $MC_TOOL_PRESEL_RESET_VALUE	Preselect tool on Reset
MD20130 $MC_CUTTING_EDGE_RESET_VALUE	Tool cutting-edge length compensation on run-up
MD20140 $MC_TRAFO_RESET_VALUE	Run-up transformation data block
MD20144 $MC_TRAFO_MODE_MASK	Selection of the kinematic transformation function
MD20150 $MC_GCODE_RESET_VALUES	Basic setting of the G groups
MD20152 $MC_GCODE_RESET_MODE	Reset behavior of the G groups
MD21330 $MC_COUPLE_RESET_MODE_1	Coupling cancellation response
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB	Assignment of geometry axis to channel axis
MD20118 $MC_GEOAX_CHANGE_RESET	Allow automatic geometry axis change

Example

Activate reset setting on reset:

- MD20110, bit 0 = 1
- MD20112 = 0

Transformation remains with reset / part program start:

- MD20110, bit 0 = 1
- MD20110, bit 7 = 1
- MD20112 = 0

Tool length compensation is retained after reset / part program start:

- MD20110, bit 4 = 1
- MD20110, bit 6 = 1
- MD20112 = 0

Active level (bit 4) and settable frame (bit 5) remain active after reset and are reset on part program start:

- MD20110, bit 4 = 1
- MD20110, bit 5 = 1
- MD20112, bit 4 = 1
- MD20112, bit 5 = 1

Note

MD20110/MD20112, bit 5 and bit 6

If MD20110/MD20112 are parameterized so that tool length compensation or a frame is active on a part program start in the automatic or MDI mode, the first programming of the axes must use absolute measurements (because of the traversing of the offset).

Exception: With MD42442/MD42440 the offsetting process for `G91` is suppressed.

10.17.1 Tool withdrawal after POWER ON with orientation transformation

Function

If a part program with a machining operation with tool orientation is aborted due to a power failure or reset, it is possible to select the previously active transformation and generate a frame in the direction of the tool axis after the control has run up (power on). The tool can then be retracted in JOG mode by means of a retraction movement towards the tool axis.

Requirement

The active measuring systems must have a machine reference for all the machine axes involved in the transformation. See Section "Automatic restoration of the machine reference (Page 1264)".

Parameterization

The following machine data must be set so that the last active transformation is retained after POWER ON:

- MD20144 $MC_TRAFO_MODE_MASK, bit 1 = 1
- MD20110 $MC_RESET_MODE_MASK, bit 0 = 1
- MD20110 $MC_RESET_MODE_MASK, bit 7 = 1

See also Section "System settings for power-up, RESET / part program end and part program start (Page 652)".

Programming

Wait for machine reference WAITENC

With the command WAITENC, the system waits channel-specific in a program until there is a valid machine reference for all the active measuring systems of the parameterized axes. See the "Requirement" section above. The parameter assignment of the axes is performed via:

MD34800 $MA_WAIT_ENC_VALID = 1

Application

In the user program (…/_N_CMA_DIR/_N_PROG_EVENT_SPF) to be called event-controlled when running up (requirement: MD20108 bit 3 = 1), the system must wait using the command `WAITENC` until the valid axis positions are available.

A frame that positions the tool axis in the direction of the X, Y or Z axis can then be generated using the NC language command `TOROTX`/`TOROTY`/`TOROTZ`.

Example

Orientation transformation and orientation axes with incremental encoders.

Configuration:	Meaning:
MD10720 $MN_OPERATING_MODE_DEFAULT [0] = 6	Run-up in JOG mode.
MD30240 $MA_ENC_TYPE [0, <axis>] = 1	Incremental measuring system.
MD34210 $MA_ENC_REFP_STATE [0, <axis>] = 3	Enable the restoration of axis positions for incremental encoders.
MD20108 $MC_PROG_EVENT_MASK = ’H9’	Activate event-controlled using program PROG_EVENT during run-up and at the start of the part program.
MD20152 $MC_GCODE_RESET_MODE [52] = 1	Obtain TOFRAME via reset.
MD20110 $MC_RESET_MODE_MASK = ’HC1’	Obtain transformation and tool offset via reset.
MD20144 $MC_TRAFO_MODE_MASK = ’H02’	Obtain transformation via POWER OFF.

Event-driven user program (…/_N_CMA_DIR/_N_PROG_EVENT_SPF):

```
Program code                          Comment
; Example with activation of the frame, which aligns the WCS in the tool direction, when running up and
  resetting with part program start.
```

```
Program code                          Comment
IF $P_PROG_EVENT == 4                 ; Run-up
    IF $P_TRAFO <> 0                  ; Transformation has been selected.
        WAITENC                       ; Wait for valid axis positions of the orientation axes.
        TOROTZ                        ; Rotate the Z axis of the WCS towards the tool axis.
    ENDIF
    M17
ENDIF

IF $P_PROG_EVENT == 1                 ; Start of the part program.
    TOROTOF                           ; Reset the tool frame.
    RET
ENDIF
```

The `WAITENC` command essentially corresponds to the following program sequence (example for 5-axis machine with AB kinematics):

```
Program code                                Comment
WHILE TRUE                                  ; Wait for a measuring system.
    IF (($AA_ENC_ACTIVE[X]==TRUE) AND ($AA_ENC_ACTIVE[Y]==TRUE) AND ($AA_ENC_ACTIVE[Z]==TRUE) AND
    ($AA_ENC_ACTIVE[A]==TRUE) AND ($AA_ENC_ACTIVE[B]==TRUE)) GOTOF GET_LABEL
    ENDIF
    G4 F0.5                                 ; 0.5 s wait time
ENDWHILE
:Position synchronization
GET_LABEL: GET(X,Y,Z,A,B,)
```

Continuing machining

AUTOMATIC mode

For automatic execution of programs in the AUTOMATIC mode, all the machine axes, whose actual position of the active measuring system has been restored, must be referenced.

MDI mode and overstore

In the MDI mode and for the overstore function, machining can also be performed, without referencing the axes, with restored positions. To do this, NC start with restored positions must be enabled explicitly for a specific channel:

MD20700 $MC_REFP_NC_START_LOCK = 2

Supplementary condition

Axes with incremental encoders and without actual value buffering

It is to be assumed that axes with incremental encoders and **without** actual value buffering are clamped with sufficient speed in the event of a power failure to prevent them drifting from their last position setpoint.

10.18 Replacing functions by subprograms

10.18.1 Overview

Function

User-specific auxiliary functions (e.g. M101) do not trigger any system functions. They are only output to the NC/PLC interface. The functionality of the auxiliary function must be implemented by the user / machine manufacturer in the PLC user program. A description will be provided as to how a user-specific subprogram call can be configured (replacement subprogram) instead of the output to NC/PLC interface, which is the default setting.

Function M101 is then still programmed in the part program. However, when executing the part program, the substitute subprogram is called. Therefore, the NC replaces the function by a subprogram call. This results in the following advantages:

- When adapting to the production process, an existing, tested and proven part program can still be used, unchanged. The changes required are then shifted into the user-specific subprograms.
- The functionality can be implemented within the substitute subprogram with the full functional scope of the NC language.
- The communication overhead between NC and PLC is not required.

Functions that can be replaced

The following functions can be replaced by subprograms:

Auxiliary functions	
M	Switching functions
T	Tool selection
TCA	Tool selection independent of the tool status
D	Tool offset
DL	Additive tool offset

Spindle-related functions during active synchronous spindle coupling	
M40	Automatic gear stage change
M41 - M45	Gear stage selection 1 ... 5
SPOS	Spindle positioning
SPOSA	Spindle positioning
M19	Spindle positioning

10.18.2 Replacement of M, T/TCA and D/DL functions

10.18.2.1 Replacement of M functions

General Information

The following conditions are applicable for replacing the M functions:

- Per block only one M function is replaced.
- A block in which an M function is to be replaced, must **not** contain the following elements:
  - M98
  - Modal subprogram call
  - Subprogram return
  - Part program end
- M functions that trigger system functions must not be replaced by a subprogram (see Section "Non-replaceable M functions").

Parameterization

M function and subprogram

M functions and the replacement subprograms are parameterized in the following machine data:

- MD10715 $MC_M_NO_FCT_CYCLE[<**Index**>] = <M function number>
- MD10716 $MC_M_NO_FCT_CYCLE_NAME[<**Index**>] = "<subprogram name>"

The M function and the corresponding replacement subprogram are connected through the same index.

Example: M function M101 is replaced by subprogram SUB_M101 and M function M102 by SUB_M102:

MD10715 $MC_M_NO_FCT_CYCLE[**0**]	= 101
MD10716 $MC_M_NO_FCT_CYCLE_NAME[**0**]	= "SUB_M101"
MD10715 $MC_M_NO_FCT_CYCLE[**1**]	= 102
MD10716 $MC_M_NO_FCT_CYCLE_NAME[**1**]	= "SUB_M102"

System variable for transferring information

For a freely selectable M function, information regarding the M function that has been replaced and additional functions (T, TCA, D, DL) for evaluation in the replacement subprogram are made available via the system variable (see Section "System variable (Page 665)"). The data contained in the system variables refers to the block in which the M function to be replaced is programmed.

The M function is selected with the index of machine data MD10715 $MC_M_NO_FCT_CYCLE[<**Index**>] in which the M function to be replaced has been parameterized:

MD10718 $MC_M_NO_FCT_CYCLE_PAR = <**Index**>

Note

For an M function replacement with transfer of information via system variable, the address extension and function value of the M function must be programmed as constant values.

Permissible programming:

- `M<function value>`
- `M=<function value>`
- `M[<address extension>]=<function value>`

Illegal programming:

- `M=<variable1>`
- `M[<variable2>]=<variable1>`

Programming

Rules for replacing M functions:

- The replacement subprogram is called at the block end
- Within the replacement subprogram, no M functions are replaced
- In an ASUB, the M function is also replaced if the ASUB was started within the replacement subprogram.

M functions that cannot be replaced

The following M functions trigger system functions as pre-defined auxiliary functions and must **not** be replaced by a subprogram:

- M0 ... M5
- M17, M30,
- M19
- M40 ... M45
- M98, M99 (only for MD18800 $MN_MM_EXTERN_LANGUAGE ≠ 0)

User-specific M functions parameterized via machine data must also not be replaced by a subprogram as they also trigger system functions.

Machine data	Meaning
MD10714 $MN_M_NO_FCT_EOP	M function for spindle active after RESET
MD10804 $MN_EXTERN_CHAN_M_NO_SET_INT	M function for ASUB activation (external mode)

Machine data	Meaning
MD10806 $MN_EXTERN_CHAN_M_NO_DISABLE_INT	M function for ASUB deactivation (external mode)
MD10814 $MN_EXTERN_M_NO_MAC_CYCLE	Macro call via M function
MD20094 $MC_SPIND_RIGID_TAPPING_M_NR	M function for switchover to controlled axis mode
MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR	M function for switchover to controlled axis mode (external mode)
MD22254 $MC_AUXFU_ASSOC_M0_VALUE	Additional M function for program stop
MD22256 $MC_AUXFU_ASSOC_M1_VALUE	Additional M function for conditional stop
MD26008 $MC_NIBBLE_PUNCH_CODE	Definition of M functions (for nibble-specific)
MD26012 $MC_PUNCHNIB_ACTIVATION	Activation of punching and nibbling functions

Note

Exception

The M function parameterized with MD22560 $MC_TOOL_CHANGE_M_CODE (tool change with M function) must not be replaced with a subprogram.

10.18.2.2 Replacing T/TCA and D/DL functions

Supplementary conditions

For replacing functions T, TCA, D and DL, the following supplementary conditions apply:

- A maximum of one function replacement is active per block.
- A block with the function replacement must **not** contain the following elements:
  - M98
  - Modal subprogram call
  - Subprogram return
  - Part program end
- If the multitool slot number is programmed with address MTL for the multitool select with T = slot number, the T replacement also replaces the MTL address. The programmed values can be queried in the replacement subprogram using the $C_MTL_PROG and $C_MTL system variables.

Parameterization: Replacement subprogram

The replacement subprogram is specified function-specific in the machine data:

Function	Machine data
T	MD10717 $MN_T_NO_FCT_CYCLE_NAME
TCA	MD15710 $MN_TCA_CYCLE_NAME
D/DL	MD11717 $MN_D_NO_FCT_CYCLE_NAME

Note

It is recommended that the same subprogram is used to replace T, TCA and D/DL functions.

Parameterization: Behavior regarding D or DL function with simultaneous T function

When D or DL and T functions are simultaneously programmed in a block, the D or DL number is either transferred as parameter to the replacement subprogram or the D or DL function is executed before calling the replacement subprogram. The behavior is configurable via:

MD10719 $MN_T_NO_FCT_CYCLE_MODE (parameterization of the T function replacement)

Bit	Value	Meaning
0	0	The D or DL number is available in the subprogram in the form of a system variable (initial state).
	1	The D or DL number is calculated directly in the block. **Note:** This function is only active if the tool change was configured with M function: MD22550 $MC_TOOL_CHANGE_MODE = 1 otherwise the D or DL values are always transferred.

System variable for transferring information

The replacement subprogram is provided with all of the information relevant to the functions programmed in the block via system variables (see Section "System variable (Page 665)").

The data contained in the system variables refers to the block in which the function to be replaced was programmed.

Parameterization: Time that the replacement subprogram is called

The call time of the replacement subprogram is set via:

MD10719 $MN_T_NO_FCT_CYCLE_MODE, bit 1 and bit 2

Bit 2	Bit 1	Time that the replacement subprogram is called
0	0	At the end of the block After the replacement subprogram has been executed, the interpretation is resumed with the program line following the line that triggered the replacement operation.
0	1	At block start After the replacement subprogram has been executed, the program line, which resulted in the replacement subprogram being called, is interpreted. The T address and the D or DL address and the M function for the tool change are no longer processed.
1	-	At block start and block end The replacement program is called twice.

System variable for the call time

System variable $P_SUB_STAT can be used to read whether the substitution is active, and if so, when the replacement subprogram – referred to the block – was called up:

Value	Meaning
0	Replacement not active
1	Replacement active, subprogram call is made at the block start
2	Replacement active, subprogram call is made at the block end

Example: Replacement of the T function

Parameterization	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 0	Tool change with T function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Name of the subprogram to replace the T function
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0	Call time: End of block

Programming	Comment
`N110 D1`	`; D1`
`N120 G90 G0 X100 Y100 Z50`	`; D1 is active.`
`N130 D2 X110 Z0 T5`	`; D1 remains active. The T function is replaced at the block end with the MY_T_CYCLE subprogram call. D2 provides MY_T_CYCLE in a system variable.`

A detailed example for replacement of the T function can be found in Section: "Examples of M/T function replacement at a tool change (Page 666)".

10.18.2.3 System variable

General Information

The replacement subprogram is provided with all of the information relevant to the functions programmed in the block (T or TCA, D or DL, M) via system variables.

Exception

D or DL number is not transferred if:

- MD10719 $MN_T_NO_FCT_CYCLE_MODE, bit 0 = 1
- MD22550 $MC_TOOL_CHANGE_MODE = 1

AND

- D or DL are programmed together with the T or M function in a block.

CAUTION

Values do not act

The values provided for the replacement subprogram in the system variables are not yet effective. It is the sole responsibility of the user / machine manufacturer to resolve this by using the appropriate programming in the replacement subprogram.

System variable

System variable	Meaning
$C_M_PROG	TRUE, if the M function has been programmed
$C_M	For $C_M_PROG == TRUE, contains the value of address M We must differentiate between two cases here: • $C_M supplies the value if, for the tool change with M function, a subprogram is configured with parameter transfer: MD10715 MN_M_NO_FCT_CYCLE • If only one subprogram is configured for the addresses T and/ or D/DL and if in the program the M function for the tool change is programmed together with one of the addresses to be replaced, then $C_M supplies the value: MD22560 $MC_TOOL_CHANGE_M_CODE
$C_AUX_VALUE[0]	Value of the replaced M function
$C_ME	For $C_M_PROG == TRUE, contains the value of the address extension of the M function
$C_AUX_EXT[0]	Address extension of the M function (identical to $C_ME)
$C_AUX_IS_QUICK[0]	TRUE, if the M function was programmed with quick output to the PLC
$C_T_PROG	TRUE, if the T function was programmed
$C_T	For $C_T_PROG == TRUE, contains the value of the T function

System variable	Meaning
$C_TE	Contains for: • $C_T_PROG == TRUE • $C_TS_PROG == TRUE the value of the address extension of the T function
$C_TS_PROG	TRUE, if for the T or TCA replacement, a tool name has been programmed.
$C_TS	For $C_TS_PROG == TRUE, contains the tool name programmed for the T or TCA replacement
$C_TCA	TRUE, if the TCA replacement is active
$C_DUPLO_PROG	TRUE, if the duplo number of the TCA replacement has been programmed
$C_DUPLO	For $C_DUPLO_PROG == TRUE, contains the value of the programmed duplo number
$C_THNO_PROG	TRUE, if the toolholder/spindle number of the TCA replacement has been programmed
$C_THNO	For $C_THNO_PROG == TRUE, contains the value of the programmed toolholder/spindle number
$C_D_PROG	TRUE, if the D function has been programmed
$C_D	For $C_D_PROG == TRUE, contains the value of the D function
$C_DL_PROG	TRUE, if the DL function was programmed
$C_DL	For $C_DL_PROG == TRUE, contains the value of the DL function
$P_SUB_STAT	Block-related time when the replacement subprogram is called
$C_MTL_PROG	TRUE if address MTL has been programmed
$C_MTL	For $C_MTL_PROG == TRUE, contains the value of address MTL

10.18.2.4 Example: Replacement of an M function

Example 1

The function M6 is replaced by calling the subprogram "SUB_M6".

The information relevant for a tool change should be transferred using system variables.

Parameterization

Machine data	Meaning
MD10715 $MN_M_NO_FCT_CYCLE[**2**] = 6	Tool change with M6
MD10716 $MN_M_NO_FCT_CYCLE_NAME[**2**] = "SUB_M6"	Replacement subprogram for M6
MD10718 $MN_M_NO_FCT_CYCLE_PAR = **2**	Information transfer using system variables

Main program

Programming	Comment
`PROC MAIN`	

Programming	Comment
...	;
N10 T1 D1 M6	; M6 is replaced by subprogram "SUB_M6"
	;
...	;
N90 M30	

Subprogram "SUB_M6"

Programming	Comment
PROC SUB_M6	
N110 IF $C_T_PROG==TRUE	; IF address T is programmed
N120 T[$C_TE]=$C_T	; Execute T selection
N130 ENDIF	; ENDIF
N140 M[$C_ME]=6	; Execute tool change.
N150 IF $C_D_PROG==TRUE	; IF address D is programmed
N160 D=$C_D	; Execute D selection
N170 ENDIF	; ENDIF
N190 M17	

Example 2

The new tool is prepared for changing with the T function. The tool change is only realized with function M6. The T function is replaced by calling the subprogram "MY_T_CYCLE". The D / DL number is transferred to the subprogram.

Parameterization

Parameterization	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 1	Tool change prepared with T function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Replacement subprogram
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0	Transfer of the D/DL number

Main program

Program code	Comment
N210 D1	;
N220 G90 G0 X100 Y100 Z50	; D1 is active.
N230 D2 X110 Z0 T5	; D1 remains active, programmed D2 is transferred
	; to the subprogram as variable
N240 M6	; Execute tool change

Example 3

The new tool is prepared for changing with the T function. The tool change is only realized with function M6. The T function is replaced by calling the subprogram "MY_T_CYCLE". The D / DL number is **not** transferred to the subprogram.

Parameterization

Parameterization	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 1	Tool change prepared with T function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Replacement subprogram
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1	No transfer of the D/DL number

Main program

```
Program code                       Comment
N310 D1
N320 G90 G0 X100 Y100 Z50          ; D1 is active.
N330 D2 X110 Z0 T5                 ; D2 is active and is not transferred as variable
                                   to
                                   ; the replacement subprogram.
N340 M6                            ; Execute tool change.
```

Example 4

The functions T and M6 are replaced by the subprogram "MY_T_CYCLE".

The parameters are transferred to the subprogram when replacing M6.

If M6 is programmed together with D or DL in the block, the D or the DL number is also transferred as parameter to the subprogram if no transfer of the D/DL number has been parameterized:

MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1

Parameterization

Configuration	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 1	Tool change with M function
MD22560 $MC_TOOL_CHANGE_M_CODE = 6	M code for tool change
MD10715 $MC_M_NO_FCT_CYCLE[3] = 6	M function to be replaced
MD10716 $MC_M_NO_FCT_CYCLE_NAME[3] = "MY_T_CYCLE"	Replacement subprogram for the M function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Replacement subprogram for the T function
MD10718 $MN_M_NO_FCT_CYCLE_PAR = 3	Parameter transfer to the replacement subprogram for M6
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1	No transfer of the D/DL number

Main program

```
Program code                       Comment
N410 D1
N420 G90 G0 X100 Y100 Z50          ; D1 is active.
N330 D2 X110 Z0 T5 M6              ; D1 remains active, D2 and T5 are transferred to
                                   the M6 replacement subprogram as variable.
```

10.18.2.5 Example: Replacement of a T and D function

The functions T and D are replaced by calling the subprogram "D_T_SUB_PROG". The following should also be true for the example:

- The tool change is realized with address T.
- The subprogram is called at the start of the block.
- The tool management is not active.
- Axis B is an indexing axis with Hirth gearing.

Parameterization

Machine data	Meaning
MD11717 $MN_**D**_NO_FCT_CYCLE_NAME = "**D_T_SUB_PROG**"	Replacement subprogram for D function
MD10717 $MN_**T**_NO_FCT_CYCLE_NAME = "**D_T_SUB_PROG**"	Replacement subprogram for M function
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 'H2'	Call at block start
MD22550 $MC_TOOL_CHANGE_MODE = 0	Tool change with T function

Main program

Programming	Comment
`PROC MAIN`	
`...`	`;`
`N10 G01 F1000 X10 T1=5 D1`	`; T and D function replaced by calling`
	`; "D_T_SUB_PROG" at start of block`
`...`	`;`
`N90 M30`	

Subprogram "D_T_SUB_PROG"

Programming	Comment
`N1000 PROC D_T_SUB_PROG DISPLOF SBLOF`	
`N4100 IF $C_T_PROG==TRUE`	`; IF address T is programmed`
`N4120    POS[B]=CAC($C_T)`	`; Approach indexing position`
`N4130    T[$C_TE]=$C_T`	`; Select tool (T selection)`
`N4140 ENDIF`	`; ENDIF`
`N4300 IF $C_T_PROG==TRUE`	`; IF address D is programmed`
`N4320    D=$C_D`	`; Select offset (D selection)`

Programming	Comment
N4330 ENDIF	; ENDIF
N4400 IF $C_DL_PROG==TRUE	; IF address DL is programmed
N4420 D=$C_DL	; Select insert offset
N4430 ENDIF	; ENDIF
N9999 RET	

10.18.2.6 Behavior in the event of a conflict

Conflict case

A conflict is present if several functions are programmed in one block and the functions should be replaced with different subprograms:

- Addresses D and DL replaced with subprogram:
  MD11717 $MN_FCT_CYCLE_NAME = "**D_SUB_PROG**"
- Address T replaced with subprogram:
  MD10717 $MN_FCT_CYCLE_NAME = "**T_SUB_PROG**"
- M function M6 replaced with subprogram:
  MD10715 $MN_M_NO_FCT_CYCLE[0] = 6
  MD10716 $MN_M_NO_FCT_CYCLE_NAME[0] = "**M6_SUB_PROG**"
  MD10718 $MN_M_NO_FCT_CYCLE_PAR = 0
  MD22550 $MC_TOOL_CHANGE_MODE = 1
  MD22560 $MC_TOOL_CHANGE_M_CODE = 6

Resolution

A conflict is resolved corresponding to the following table:

The following are programmed in one program line:			Called subprogram:
D and/or DL	T or TCA	M6	
–	–	x	**M6**_SUB_PROG
–	x	–	**T**_SUB_PROG
–	x	x	**M6**_SUB_PROG
x	–	–	**D**_SUB_PROG
x	–	x	**M6**_SUB_PROG
x	x	–	**T**_SUB_PROG
x	x	x	**M6**_SUB_PROG

10.18.3 Replacement of spindle functions

10.18.3.1 General information

Function

When a coupling is active the following spindle functions can be replaced for leading spindles:

- `M40`: Automatic gear stage change
- `M41` ... `M45` Programmed gear stage change
- `SPOS`, `SPOSA` and `M19`: Spindle positioning

Boundary conditions

- To replace a spindle function, the following conditions must be met:
  – The programmed spindle must be the leading spindle of an active coupling.
  – Leading and following spindle are located in the same channel. This is only detected if the leading spindle is located in the channel in which the coupling was closed. If the leading spindle is changed to another channel, a gear stage change or positioning of this spindle does not call the replacement subprogram.
  – A programmed gear stage change must result in a real gear stage change. For this purpose, the programmed and active gear stage must differ.
- In a block, only one spindle function can be replaced. Multiple replacements lead to the termination of the program processing. The spindle functions, which are to be replaced, must then be distributed over several blocks.

Parameterization

Spindle function

The spindle functions to be replaced by the subprogram are selected in the machine data:

MD30465 $MA_AXIS_LANG_SUB_MASK

Bit	Meaning	
0	Gear-stage change automatic (`M40`) and directly (`M41`-`M45`)	
	Val-ue	Meaning
	0	No replacement
	1	Replacement through the subprogram set in MD15700 and MD15702
1	Spindle positioning with `SPOS` / `SPOSA` / `M19`	
	Val-ue	Meaning
	0	No replacement
	1	Replacement through the subprogram set in MD15700 and MD15702

Subprogram: Name

The name of the replacement subprogram is entered in the machine data:

MD15700 $MN_LANG_SUB_NAME = "<subprogram name>"

Subprogram: Path

The path of the replacement subprogram is set in the machine data:

MD15702 $MN_LANG_SUB_PATH = <value>

Value	Meaning
0	**Manufacturer** cycle folder: /_N_**CMA**_DIR
1	**User** cycle folder: /_N_**CUS**_DIR
2	**Siemens** cycle folder: /_N_**CST**_DIR

System variable: Time that the replacement subprogram is called

The time that the replacement subprogram is called can be read using the system variable $P_SUB_STAT:

Value	Meaning
0	Replacement not active
1	Replacement active, subprogram call is made at the block start
2	Replacement active, subprogram call is made at the block end

Block processing

If the replacement subprogram is called at the block start, after processing the replacement subprogram, the block that initiated the call is processed. The replaced commands are no longer processed.

If the replacement subprogram is called at the block end, the block that initiated calling the replacement subprogram is first processed without the commands to be replaced. The replacement subprogram is then subsequently called.

10.18.3.2 Replacement of M40 - M45 (gear stage change)

Function

When a coupling is active, the commands for gear stage change (M40, M41 ... M45) of the leading spindle are replaced by calling a user-specific subprogram.

Parameterization

Activation

- MD30465 $MA_AXIS_LANG_SUB_MASK, bit 0 = 1

Time that the subprogram is called

- `M40`
  The time of the call cannot be set. The replacement subprogram is always called at the block start.
- `M41` ... `M45`
  The call time depends on the configured output behavior of the auxiliary function to the PLC (see below MD22080):
  – Output **before** or **during** motion: Subprogram call at the **start of the block**.
  – Output **after** motion: Subprogram call at the **end of the block**

  MD22080 $MC_AUXFU_PREDEF_SPEC[12 ... 16] (output behavior for `M41` ... `M45`)

Bit	Value	Meaning
5	1	Output of the auxiliary function **before** the motion
6	1	Output of the auxiliary function **during** the motion
7	1	Output of the auxiliary function **after** the motion

System variable to transfer information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Chapter "System variable (Page 674)"). The data refer exclusively to the block, in which the function to be replaced has been programmed.

10.18.3.3 Replacement of SPOS, SPOSA, M19 (spindle positioning)

Function

When a coupling is active, the positioning commands (`SPOS`, `SPOSA` or `M19`) of a leading spindle are replaced by calling a user-specific subprogram (replacement subprogram).

Application example

When machining workpieces in parallel on a double-spindle machine, the spindles are coupled through a coupling factor not equal to 1. When changing the tool, they must be brought to the same position. The replacement subprogram opens the coupling, separately positions the spindles at the tool change position and then recloses the coupling.

Parameterization

Activation

- MD30465 $MA_AXIS_LANG_SUB_MASK, bit 1 = 1

Time that the replacement subprogram is called

- `SPOS`, `SPOSA`
  The time of the call cannot be set. The replacement subprogram is always called at the block start.
- `M19`
  The call time depends on the configured output behavior of the auxiliary function to the PLC (see below MD22080):
  - Output **before** or **during** motion: Subprogram call at the **start of the block**.
  - Output **after** motion: Subprogram call at the **end of the block**

MD22080 $MC_AUXFU_PREDEF_SPEC[9]		
Bit	**Value**	**Meaning**
5	1	Output of the auxiliary function **before** the motion
6	1	Output of the auxiliary function **during** the motion
7	1	Output of the auxiliary function **after** the motion

System variable for transferring information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Chapter "System variable (Page 674)"). The data refer exclusively to the block, in which the function to be replaced has been programmed.

10.18.3.4 System variable

System variable	Meaning
$P_SUB_AXFCT	TRUE, if `M40`, `M41` ... `M45` replacement is active
$P_SUB_GEAR	Programmed or calculated gear stage Outside the replacement subprogram: Gear stage of the master spindle
$P_SUB_AUTOGEAR	TRUE, if M40 was active in the block that had initiated the replacement operation. Outside the replacement subprogram: Actual setting in the interpreter
$P_SUB_LA	Contains the axis name of the leading spindle of the active coupling, which had triggered the replacement operation. **Note** If the variable is used outside the replacement subprogram, program processing is cancelled with an alarm.
$P_SUB_CA	Contains the axis name of the following spindle of the active coupling, which had triggered the replacement operation. **Note** If the variable is called outside the replacement subprogram, program processing is cancelled with an alarm.
$P_SUB_AXFCT	Contains the active replacement types corresponding to MD30465 $MA_AXIS_LANG_SUB_MASK
$P_SUB_SPOS	TRUE, if the `SPOS` replacement is active

System variable	Meaning	
$P_SUB_SPOSA	TRUE, if the `SPOSA` replacement is active	
$P_SUB_M19	TRUE, if the `M19` replacement is active	
$P_SUB_SPOSIT	Contains the programmed spindle position **Note** If the variable is called outside the replacement subprogram, program processing is cancelled with an alarm.	
$P_SUB_SPOSMODE	Contains the position approach mode for the programmed spindle position:	
	Value	**Meaning**
	0	No change of the position approach mode
	1	AC
	2	IC
	3	DC
	4	ACP
	5	ACN
	6	OC
	7	PC
	Note: If the variable is called outside the replacement subprogram, program processing is cancelled with an alarm.	
$P_SUB_STAT	Block-related time when the replacement subprogram is called	

10.18.3.5 Example: Gear stage change

In the subprogram, all commands to change the gear stage `M40`, `M41` ... `M45` are replaced.

Parameterization

Machine data	Meaning
MD15700 $MN_LANG_SUB_NAME = "LANG_SUB"	Subprogram
MD15702 $MN_LANG_SUB_PATH = 0	Manufacturer's folder
MD22080 $MC_AUXFU_PREDEF_SPEC[12] = 'H21'	M41: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'	M42: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'	M43: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[15] = 'H21'	M44: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[16] = 'H21'	M45: Output prior to motion
MD30465 $MA_AXIS_LANG_SUB_MASK[AX5] = 'H0001'	Replace gear change commands

Main program

```
Programming                              Comment
PROC MAIN
N110 COUPON(S2,S1)                       ; Close the synchronous spindle coupling
```

Programming	Comment
N120 G01 F100 X100 S5000 M3 M43	; Subprogram call due to M43
N130 M40	; Switch on automatic gear stage change
N140 M3 S1000	; Subprogram call due to S1000
	; and as a result an initiated automatic
	; gear stage change
N9999 M30	

Replacement subprogram "LANG_SUB", version 1

Optimized for simplicity and velocity by directly addressing the spindles (S1: Leading spindle, S2: Following spindle).

Programming	Comment
N1000 PROC LANG_SUB DISPLOF SBLOF	
N1100 IF($P_SUB_AXFCT ==1)	; Replacement due to gear stage change
N1140 DELAYFSTON	; Start of stop delay area
N1150 COUPOF(S2,S1)	; Open synchronous spindle coupling
N1160 ;gear stage change separately for leading and following spindles	
N1170 M1=$P_SUB_GEAR M2=$P_SUB_GEAR	
N1180 DELAYFSTON	; End of stop delay area
N1190 COUPON(S2,S1)	; Close the synchronous spindle coupling
N1200 ENDIF	
...	
N9999 RET	

Replacement subprogram "LANG_SUB", version 2

Flexibility through indirect addressing using the system variable (leading spindle: $P_SUB_LA, Folgespindel: $P_SUB_CA).

Programming	Comment
N1000 PROC LANG_SUB DISPLOF SBLOF	
N1010 DEF AXIS _LA	; Bit memory for leading axis / leading spindle
N1020 DEF AXIS _CA	; Bit memory for following axis / following spindle
N1030 DEF INT _GEAR	; Bit memory for gear stage
N1100 IF($P_SUB_AXFCT==1)	; Replacement due to gear stage change
N1110 _GEAR=$P_SUB_GEAR	; Gear stage to be activated
N1120 _LA=$P_SUB_LA	; Axis name of the leading spindle
N1130 _CA=$P_SUB_CA	; Axis name of the following spindle
N1140 DELAYFSTON	; Start of stop delay area
N1150 COUPOF(_CA,_LA)	; Open synchronous spindle coupling
N1160 ;gear stage change for leading and following spindles	
N1170 M[AXTOSPI(_LA)]=_GEAR M[AXTOSPI(_CA)]=_GEAR	

Programming	Comment
N1180 DELAYFSTOF	; End of stop delay area
N1190 COUPON(_CA,_LA)	; Close the synchronous spindle coupling
N1200 ENDIF	
...	
N9999 RET	

10.18.3.6 Example: Spindle positioning

In the subprogram, only the replacement of commands `SPOS` and `SPOSA` is explicitly executed. Additional replacements should be supplemented in essentially the same fashion.

Parameterization

Machine data	Meaning
MD30465 $MA_AXIS_LANG_SUB_MASK[AX5] = 'H0002'	Replace positioning commands
MD22080 $MC_AUXFU_PREDEF_SPEC[9] = 'H0021'	Output of `M19` to the PLC before motion

Setting data	Meaning
SD43240 $SA_M19_SPOS[AX5] = 260	Spindle position for `M19` = 260
SD43250 $SA_M19_SPOSMODE[AX5] = 4	Position approach mode for `M19`: "Approach in the positive direction (ACP)"

Main program

Programming	Comment
PROC MAIN	
...	
N210 COUPON(S2,S1)	; Activate synchronous spindle coupling
N220 SPOS[1]=100	; Position leading spindle with SPOS
...	
N310 G01 F1000 X100 M19	; Position leading spindle with M19

Replacement subprogram "LANG_SUB", version 1

Optimized for simplicity and velocity by directly addressing the spindles (S1: Leading spindle, S2: Following spindle).

Programming	Comment
N1000 PROC LANG_SUB DISPLOF SBLOF	
N2100 IF($P_SUB_AXFCT==2)	

```
Programming                             Comment
N2110 ;Replacement of SPOS/SPOSA/M19 for active synchronous spindle coupling
N2185    DELAYFSTON                     ; Start of stop delay area
N2190    COUPOF(S2,S1)                  ; Open synchronous spindle coupling
N2200                                   ; Position leading and following spindles
N2210    IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)
N2220    ;SPOS and SPOSA are mapped to SPOS
N2230       CASE $P_SUB_SPOSMODE OF \
             0 GOTOF LABEL1_DC \
             1 GOTOF LABEL1_IC \
             2 GOTOF LABEL1_AC \
             3 GOTOF LABEL1_DC \
             4 GOTOF LABEL1_ACP \
             5 GOTOF LABEL1_ACN \
             DEFAULT GOTOF LABEL_ERR
LABEL1_DC:  SPOS[1]=DC($P_SUB_SPOSIT) SPOS[2]=DC($P_SUB_SPOSIT)
            GOTOF LABEL1_CONT
LABEL1_IC:  DELAYFSTOF
            SPOS[1]=IC($P_SUB_SPOSIT) SPOS[2]=IC($P_SUB_SPOSIT)
            DELAYFSTON
            GOTOF LABEL1_CONT
LABEL1_AC:  SPOS[1]=AC($P_SUB_SPOSIT) SPOS[2]=AC($P_SUB_SPOSIT)
            GOTOF LABEL1_CONT
LABEL1_ACP: SPOS[1]=ACP($P_SUB_SPOSIT) SPOS[2]=ACP($P_SUB_SPOSIT)
            GOTOF LABEL1_CONT
LABEL1_ACN: SPOS[1]=ACN($P_SUB_SPOSIT) SPOS[2]=ACN($P_SUB_SPOSIT)
LABEL1_CONT:
N2250    ELSE                           ; Position the spindle with M19
N2270       M1=19 M2=19                 ; Leading and following spindles
N2280    ENDIF                          ; End replacement SPOS, SPOSA
N2285    DELAYFSTOF                     ; End of stop delay area
N2290    COUPON(S2,S1)                  ; Activate synchronous spindle coupling
N2410 ELSE
N2420    ;from here processing further replacements
...
N3300 ENDIF                             ; End replacements
...
N9999 RET                               ; Normal end of program
LABEL_ERR: SETAL(61000)                 ; Error has occurred
```

Replacement subprogram "LANG_SUB", version 2

Flexibility through indirect addressing using the system variable (leading spindle: $P_SUB_LA, Folgespindel: $P_SUB_CA).

```
Programming                            Comment
N1000 PROC LANG_SUB DISPLOF SBLOF
N1010 DEF AXIS _LA                     ; Leading axis/spindle
N1020 DEF AXIS _CA                     ; Following axis/spindle
N1030 DEF INT _LSPI                    ; Leading spindle number (programmed
                                       ; spindle)
N1040 DEF INT _CSPI                    ; Following spindle number
...
N2100 IF($P_SUB_AXFCT==2)
N2110 ; Replacement of SPOS/SPOSA/M19 for active ; synchronous spindle coupling
N2120    _LA=$P_SUB_LA                 ; Axis name of the leading spindle
N2130    _CA=$P_SUB_CA                 ; Axis name of the following spindle
N2140    _LSPI=AXTOSPI(_LA)            ; Number of the leading spindle
N2180    _CSPI=AXTOSPI(_LA)            ; Number of the following spindle
N2185    DELAYFSTON                    ; Start of stop delay area
N2190    COUPOF(_CA,_LA)               ; Deactivate synchronous spindle coupling
N2200                                  ; Position leading and following spindles:
N2210    IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)
N2220    ;SPOS and SPOSA are mapped to SPOS
N2230       CASE $P_SUB_SPOSMODE OF
             0 GOTOF LABEL1_DC \
             1 GOTOF LABEL1_IC \
             2 GOTOF LABEL1_AC \
             3 GOTOF LABEL1_DC \
             4 GOTOF LABEL1_ACP \
             5 GOTOF LABEL1_ACN \
             DEFAULT GOTOF LABEL_ERR
LABEL1_DC:  SPOS[_LSPI]=DC($P_SUB_SPOSIT) SPOS[_CSPI]=DC($P_SUB_SPOSIT)
            GOTOF LABEL1_CONT
LABEL1_IC:  DELAYFSTOF
            SPOS[_LSPI]=IC($P_SUB_SPOSIT) SPOS[_CSPI]=IC($P_SUB_SPOSIT)
            DELAYFSTON
            GOTOF LABEL1_CONT
LABEL1_AC:  SPOS[_LSPI]=AC($P_SUB_SPOSIT) SPOS[_CSPI]=AC($P_SUB_SPOSIT)
            GOTOF LABEL1_CONT
LABEL1_ACP: SPOS[_LSPI]=ACP($P_SUB_SPOSIT) POS[_CSPI]=ACP($P_SUB_SPOSIT)
            GOTOF LABEL1_CONT
LABEL1_ACN: SPOS[_LSPI]=ACN($P_SUB_SPOSIT) POS[_CSPI]=ACN($P_SUB_SPOSIT)
LABEL1_CONT:
N2250    ELSE                          ; Position spindles with M19
N2270       M[_LSPI]=19 M[_CSPI]=19
```

Programming	Comment
`N2280    ENDIF`	
`N2285    DELAYFSTOF`	`; End of stop delay area`
`N2290    COUPON(_CA,_LA)`	`; Activate synchronous spindle coupling`
`N2410 ELSE`	
`N2420    ;from here processing further replacements`	
`...`	
`N3300 ENDIF`	
`...`	
`N9999 RET`	`; Normal end of program`
`LABEL_ERR: SETAL(61000)`	`; Error has occurred`

10.18.4 Properties of the subprograms

General rules

- The subprogram called when making the replacement can contain the command `PROC` and the attribute `SBLOF` and `DISPLOF`.
- The replacement is also made in the ISO language mode. However, the replacement subprograms are exclusively processed in the standard language mode (Siemens). There is an implicit switchover into the standard language mode. The original language mode is reselected with the return jump from the replacement subprogram.
- System variables are exclusively used to transfer information to the replacement subprogram. Transfer parameters are not possible.
- The behavior for a single block and attribute `SBLOF` depends on the setting in:
  MD10702 IGNORE_SINGLEBLOCK_MASK, bit 14 (prevent single-block stop)

Value	Meaning
0	The replacement subprogram behaves like a "normal" subprogram: • Return jump with `M17`: Stop at the end of the subprogram **Note** The output of the M function at the PLC depends on: MD20800 $MC_SPF_END_TO_VDI, bit 0 (subprogram end to PLC) - Bit 0 = 0: No output - Bit 0 = 1: M17 is output to the PLC. • Return jump with `RET`: No stop at the end of the replacement subprogram
1	In the block, in which the replacement subprogram is called, **only one stop is made**. Regardless of whether: • The subprogram was called at the block start and/or at the block end • Other subprograms are called in the subprogram • The subprogram is exited with `M17` or `RET` The single-block stop takes place for the replacement of M functions at the end of the replacement subprogram. For the replacement of T and D/DL functions, the time of the single-block stop depends on when the subprogram is called: • Call at block start: Single-block stop at the end of the block • Call at the block end: Single-block stop at the end of the replacement subprogram

- For replacement subprograms with the attribute `DISPLOF` in the block display, the program line is displayed as actual block, which resulted in the subprogram being called.
- In the replacement subprogram, areas or the complete replacement subprogram can be protected against interruptions, such as NC stop, read-in inhibit etc., using the `DELAYFSTON` and `DELAYFSTOF` commands.
- Replacements do not occur recursively, i.e. the function that has led to the replacement subprogram call is no longer replaced if it is programmed again in the replacement subprogram.

Output of auxiliary functions to PLC

When replacing auxiliary functions, calling the replacement subprogram does not initiate that the auxiliary function is output to the PLC. The auxiliary function is only output if the auxiliary function is reprogrammed in the replacement subprogram.

Behavior during block search

The replacement subprogram is also called in the block search modes "Block search with calculation" and "Block search with calculation in the program test mode" (SERUPRO). Any special features must be implemented in the replacement subprogram using the system variable: $P_SEARCH, $AC_SERUPRO.

Regarding collecting actions for "block search with calculation", replacement subprograms behave just like "normal" subprograms.

10.18.5 Restrictions

- Function replacements are not permitted in:
  - Synchronized actions
  - Technology cycles
- There must be no blockwise synchronized actions in front of a block that contains functions at the beginning to be replaced. See the paragraph below "Example for: Non-modal synchronized actions".
- Only the actions required for the respective replacements can be performed in the replacement subprogram.
- In a block, in which the replacement subprogram is called at the block end, the following should be observed:
  - No modal subprogram call should be active
  - No subprogram return jump should be programmed
  - No program end should be programmed

  Note

  The controller does not monitor whether the function to be replaced has been realized in the replacement subprogram.

Example of: Non-modal synchronized actions

MD30465 $MA_AXIS_LANG_SUB_MASK, bit 0 = 1 (gear stage change)

```
Program code
...
N1000 WHENEVER $AA_IM[X2] <= $AA_IM[X1] + 0.5 DO $AA_OVR[X1]=0
N1010 G1 X100 M43
...
```

If, in block `N1010`, the function `M43` initiates that a replacement subprogram is called, machining is interrupted and an alarm is output.

10.19 Renaming/locking NC commands

Function

The machine manufacturer or end user can change the names of existing NC commands with the "Rename/lock NC commands" function.

Application

The function can be used for the following purposes:

- Improve the readability of part programs
- Lock NC commands
- User-specific extension of NC functions

Parameterization

The renaming/locking of NC commands is made via the machine data:

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[<n>] (list of reconfigured NC codes)

An even index [<n>] contains the original name of the command. The following uneven index contains the new name of the command.

Note

For SINUMERIK 828D, the entries in the even indexes (original command name) are predefined and cannot be changed. Only the uneven indexes (new command name) can be written.

An empty string ("") as new name means that there is no new name for the command. Consequently, the command is locked and can no longer be programmed.

Changes in MD10712 take effect on Power On.

Examples

Example 1 (840D sl): Rename command G00 and deactivate G01

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0] = "G00"

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1] = "RAPIDTRAVERSE"

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[2] = "G01"

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[3] = ""

After activation with Power On, rapid traverse can no longer be programmed with the `G00` command, but only with the `RAPIDTRAVERSE` command. `G01` can no longer be programmed.

Example 2 (828D): Change the preassigned value for SINUMERIK 828D

To adapt to the ISO dialect mode, the indexes [0] to [7] are preassigned for the SINUMERIK 828D so that the `G505` and `G506` commands are renamed as `G58` and `G59`. Originally `G58` and `G59` were programmable frames, `G505` and `G506` were settable frames. The renaming makes `G58` and `G59` into settable frames.

If this preassignment is not desired, and G58 and G59 should remain programmable frames, the renaming must be undone as follows:

Index for MD10712	Command name	
	Predefined	New
[0]	"G58"	"G58"
[1]	**""**	**"G58"**
[2]	"G59"	"G59"
[3]	**""**	**"G59"**
[4]	"G505"	"G505"
[5]	**"G58"**	**"G505"**
[6]	"G506"	"G506"
[7]	**"G59"**	**"G506"**

Example 3 (828D): Rename the _TCA command as TC

The TC command for SINUMERIK 828D is renamed as standard as _TCA. The TCA function has a cycle. If this cycle is not used, the _TCA command must be renamed again as TC:

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[**8**] = "_TCA"

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[**9**] = "TCA"

Example 4 (828D): Reroute the existing command to a user-specific cycle.

The existing WAITM command should be replaced by a cycle in which user-specific functions are available.

1. Rename the existing WAITM command with MD10712 as _WAITM:
   MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[**20**] = "WAITM"
   MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[**21**] = "_WAITM"
2. Create a part program with the name WAITM.
   The interface for PROC WAITM must be identical with that for the NC function WAITM. This part program now replaces the existing WAITM NC function. Every WAITM command from the part program, also those from Siemens cycles and Shopmill/Shopturn cycles, calls the user-specific WAITM subprogram. In this WAITM subprogram, the user-specific functions are executed and, if required, the renamed _WAITM NC function called.

```
Program code                                   Comment
N5 PROC WAITM(<flag no.>,<channel no.>)
...                                            ; User-specific functions
N100 _WAITM(<flag no.>,<channel no.>)
...                                            ; User-specific functions
M30
```

10.20 Program runtime / part counter

Information on the program runtime and workpiece counter are provided to support the machine tool operator.

This information can be processed as system variables in the NC and/or PLC program. This information is also available to be displayed on the operator interface.

10.20.1 Program runtime

10.20.1.1 Function

The "Program runtime" function provides various timers to monitor technological processes, which can be read into the part program and into synchronized actions via system variables. There are two types of timers:

1. Standard timers
   Standard timers are always active
2. Timers that can be activated
   Timers that can be activated must be activated via machine data.

Standard timers

Time since the last control power-up

System variable	Meaning
$AN_SETUP_TIME	Time since the last control power-up with default values in minutes. Is automatically reset to "0" in each control power-up with default values.
$AN_POWERON_TIME	Time since the last normal control power-up ("warm restart") in minutes. Is automatically reset to "0" in each normal control power-up.

Program runtime

The timers to measure the program runtimes are only available in AUTOMATIC mode.

System variable (channel-specific)	Description
$AC_ACT_PROG_NET_TIME	Actual net runtime of the current program in seconds Net runtime means that the time, in which the program was stopped, has been deducted. If, in the AUTOMATIC operating mode, a part program is restarted from the RESET channel state, then $AC_ACT_PROG_NET_TIME is automatically reset to "0". Additional properties: • The reset button does not reset $AC_ACT_PROG_NET_TIME back to "0", but rather only stops the timer. • When starting an ASUB, $AC_ACT_PROG_NET_TIME is set to "0" and also counts the runtime of the ASUB. At the end of an ASUB, it behaves just the same as for the RESET button: The timer is only held, but is not set to "0". • $AC_ACT_PROG_NET_TIME is **not** reset when starting an event-controlled program (PROG_EVENT). The program runtime is only counted further if it involves a start, M30 or a search PROG_EVENT. • The behavior of $AC_ACT_PROG_NET_TIME for GOTOS and override = 0% can be parameterized using MD27850 (refer to Section "Parameterization") **Tip:** With $AC_PROG_NET_TIME_TRIGGER, $AC_ACT_PROG_NET_TIME can be manipulated further.
$AC_OLD_PROG_NET_TIME	Net runtime in seconds of the program that has just been correctly ended "Correctly ended" means that the program was not interrupted with RESET, but instead ended properly with `M30`. If a new program is started, $AC_OLD_PROG_NET_TIME remains unscanned, till `M30` is reached again. Additional properties: • $AC_OLD_PROG_NET_TIME is set to "0" if the currently selected program is edited. • $AC_OLD_PROG_NET_TIME is not changed at the end of an ASUB or an event-controlled program (PROG_EVENT). **Tip:** The implied copying process of $AC_ACT_PROG_NET_TIME after $AC_OLD_PROG_NET_TIME takes place only when $AC_PROG_NET_TIME_TRIGGER is not written.

System variable (channel-specific)	Description	
$AC_OLD_PROG_NET_TIME_COUNT	Changes to $AC_OLD_PROG_NET_TIME After POWER ON, $AC_OLD_PROG_NET_TIME_COUNT is at "0". $AC_OLD_PROG_NET_TIME_COUNT is always increased if the control has newly written to $AC_OLD_PROG_NET_TIME. If the user terminates the running program with RESET, $AC_OLD_PROG_NET_TIME and $AC_OLD_PROG_NET_TIME_COUNT remain unchanged. With $AC_OLD_PROG_NET_TIME_COUNT it can thus be ascertained, whether $AC_OLD_PROG_NET_TIME was written. Example: If two programs running consecutively have the same runtime and were ended correctly, then the user can identify this via the changed value in $AC_OLD_PROG_NET_TIME_COUNT.	
$AC_PROG_NET_TIME_TRIGGER	Trigger for the selective measurement of program sections. The function is triggered by the writing of the value to the variable:	
	Value	**Function**
	0	Not active
	1	Terminates the measurement • $AC_OLD_PROG_NET_TIME = $AC_ACT_PROG_NET_TIME • $AC_ACT_PROG_NET_TIME = 0 and then continues to run
	2	Starts the measurement • $AC_ACT_PROG_NET_TIME = 0 • $AC_OLD_PROG_NET_TIME is not changed
	3	Stops the measurement • $AC_OLD_PROG_NET_TIME is not changed • $AC_ACT_PROG_NET_TIME is not changed
	4	Continues the stopped measurement • $AC_ACT_PROG_NET_TIME continues to run • $AC_OLD_PROG_NET_TIME is not changed
All system variables are reset to 0 as a result of POWER ON!		

Note

Residual time for a workpiece

If the same workpieces are machined one after the other then using the following timer values the remaining residual time for a workpiece can be determined.

- Processing time for the last workpiece produced (see $AC_OLD_PROG_NET_TIME)
- Current processing time (see $AC_ACT_PROG_NET_TIME)

The residual time is displayed on the user interface in addition to the current processing time.

Note

Using STOPRE

The system variables $AC_OLD_PROG_NET_TIME and $AC_OLD_PROG_NET_TIME_COUNT do not generate any implicit preprocessing stop. This is uncritical when used in the part program if the value of the system variables comes from the previous program run. However, if the trigger for the runtime measurement ($AC_PROG_NET_TIME_TRIGGER) is written very frequently and as a result $AC_OLD_PROG_NET_TIME changes very frequently, then an explicit `STOPRE` should be used in the part program.

Timers that can be activated

Program runtime

The timers to measure the program runtimes are only available in AUTOMATIC mode.

System variable (channel-specific)	Description
$AC_OPERATING_TIME	Total runtime of NC programs in Automatic mode (in s) In the automatic mode, the runtimes of all programs between NC start and end of program / reset are summed up. The default is not to count in NC stop and override = 0%. Continued counting can be activated at an override of 0% via MD27860. The value is automatically reset to "0" every time the control powers up.
$AC_CYCLE_TIME	Runtime of the selected NC program (in seconds) The runtime between NC Start and end of program / NC reset is measured in the selected NC program. The default is not to count in NC stop and override = 0%. Continued counting can be activated at an override of 0% via MD27860. The value is automatically reset to "0" every time a new NC program starts up. MD27860 can be set to ensure that this value will be deleted even if there is a jump to the beginning of the program with `GOTOS` or in the event of ASUBs and PROG_EVENTs starting.
$AC_CUTTING_TIME	Processing time in seconds The runtime of the path axes (at least one is active) is measured in all NC programs between NC Start and end of program / NC reset without rapid traverse active. MD27860 can be used to set whether measuring should only be conducted when the tool is active or independent of the tool state. The measurement is interrupted when a dwell time is active. The value is automatically reset to "0" every time the control powers up with default values.

10.20.1.2 Commissioning

Activation/deactivation

The timer that can be activated is switched-in/switched-out using machine data:

MD27860 $MC_PROCESSTIMER_MODE, bit 0 - 2 = <value>

Bit	Value	Meaning
0	0	Timer for $AC_OPERATING_TIME not active.
	1	Timer for $AC_OPERATING_TIME active.
1	0	Timer for $AC_CYCLE_TIME not active.
	1	Timer for $AC_CYCLE_TIME active.
2	0	Timer for $AC_CUTTING_TIME not active.
	1	Timer for $AC_CUTTING_TIME active.

Parameterization

Behavior of the timer that is always active

The behavior of the timer that is always active for GOTOS and override = 0% is set using machine data:

MD27850 $MC_PROG_NET_TIMER_MODE

Bit	Value	Meaning
0	0	$AC_ACT_PROG_NET_TIME is not reset to "0" in case of a jump with `GOTOS` to the program start (initial setting).
	1	With a jump with GOTOS to the start of the program, $AC_ACT_PROG_NET_TIME is reset to "0", the value is first saved in $AC_OLD_PROG_NET_TIME and the program counter $AC_OLD_PROG_NET_TIME_COUNT is incremented.
1	0	$AC_ACT_PROG_NET_TIME is not increased for override = 0%. This means that the program runtime is measured without the time for which the override was set to "0" (basic setting).
	1	$AC_ACT_PROG_NET_TIME is also increased for override = 0%. This means that the program runtime is measured with the time for which the override was set to "0".

Behavior of the timer that can be activated

The behavior of the timer that can be activated for certain functions (e.g. test run feedrate, program test) is set using machine data:

MD27860 $MC_PROCESSTIMER_MODE

Bit	Value	Meaning
4	0	No measurement during active dry run feedrate.
	1	Measurement during active dry run feedrate.
5	0	No measurement during program test.
	1	Measurement during program test.
6	Only for bit 1 = 1 (timer for $AC_CYCLE_TIME is active)	
	0	$AC_CYCLE_TIME is reset to "0" also in case of Start through ASUB and PROG_EVENTs.
	1	$AC_CYCLE_TIME is not reset to "0" in case of Start through ASUB and PROG_EVENTs.

Bit	Value	Meaning
7	Only for bit 2 = 1 (timer for $AC_CUTTING_TIME is active)	
	0	Timer for $AC_CUTTING_TIME counts only for the active tool.
	1	Timer for $AC_CUTTING_TIME counts independent of the tool.
8	Only for bit 1 = 1 (timer for $AC_CYCLE_TIME is active)	
	0	$AC_CYCLE_TIME is not reset to "0" in case of a jump with `GOTOS` to the program start.
	1	$AC_CYCLE_TIME is reset to "0" in case of a jump with `GOTOS` to the program start.
9	Only for bit 0, 1 = 1 (timer for $AC_OPERATING_TIME and $AC_CYCLE_TIME are active)	
	0	Counting of the program runtime is not continued at an override of 0%.
	1	Counting of the program runtime continues at an override of 0%.

10.20.1.3 Supplementary conditions

Block search

No program runtimes are determined through block searches.

REPOS

The duration of a REPOS process is added to the current processing time ($AC_ACT_PROG_NET_TIME).

10.20.1.4 Examples

Example 1: Parameterization of the runtime measurement via MD27860

Activating the runtime measurement for the active NC program and hence no measurement in case of active dry run feedrate and program test:

MD27860 $MC_PROCESSTIMER_MODE = 'H2'

Activating the measurement for the tool action time and measurement also with active dry run feedrate and program test.

MD27860 $MC_PROCESSTIMER_MODE = 'H34'

Activating the measurement for the total runtime and the processing time with an active tool, including measurement with a program test:

MD27860 $MC_PROCESSTIMER_MODE = 'H25'

Activating the measurement for the total runtime and the machining time (independent of the tool), including measurement with a program test:

MD27860 $MC_PROCESSTIMER_MODE = 'Ha5'

Activating the measurement for the processing time with an active tool, including measurements at an override = 0%, but not with a trial run feed active:

MD27860 $MC_PROCESSTIMER_MODE = 'H22'

Example 2: Measuring the duration of "mySubProgrammA"

Program code

```
...
N50 DO $AC_PROG_NET_TIME_TRIGGER=2
N60 FOR ii= 0 TO 300
N70 mySubProgrammA
N80 DO $AC_PROG_NET_TIME_TRIGGER=1
N95 ENDFOR
N97 mySubProgrammB
N98 M30
```

After the program has processed line `N80`, the net runtime of "mySubProgrammA" is located in $AC_OLD_PROG_NET_TIME.

The value from $AC_OLD_PROG_NET_TIME:

- Is kept beyond `M30`.
- Is updated each time the loop is run through.

Example 3: Measuring the duration of "mySubProgrammA" and "mySubProgrammC"

Program code

```
N10 DO $AC_PROG_NET_TIME_TRIGGER=2
N20 mySubProgrammA
N30 DO $AC_PROG_NET_TIME_TRIGGER=3
N40 mySubProgrammB
N50 DO $AC_PROG_NET_TIME_TRIGGER=4
N60 mySubProgrammC
N70 DO $AC_PROG_NET_TIME_TRIGGER=1
N80 mySubProgrammD
N90 M30
```

10.20.2 Workpiece counter

10.20.2.1 Function

Various counters with a range of values from 0 to 999,999,999 are available with the "Workpiece counter" function in the form of channel-specific system variables. Read and write access to the system variables is possible.

The following channel-specific machine data can be used to control counter activation, counter reset timing and the counting algorithm.

System variables for workpiece counting

System variable	Meaning
$AC_REQUIRED_PARTS	Number of workpieces to be produced (setpoint number of workpieces) In this counter the number of workpieces at which the actual workpiece count ($AC_ACTUAL_PARTS) will be reset to "0" can be defined. MD27880 can be used to activate the generation of the display alarm: "Channel %1 workpiece target = %2 reached" and of the channel-specific NC/PLC interface signal: DB21, DBX317.1 (workpiece target reached) .
$AC_TOTAL_PARTS	Total number of completed workpieces (actual workpiece total) This counter specifies the total number of workpieces produced since the start time. The value is only automatically reset to "0" when the control runs up with default values.
$AC_ACTUAL_PARTS	Number of completed workpieces (actual workpiece total) This counter registers the total number of workpieces produced since the start time. On condition that $AC_REQUIRED_PARTS > 0, the counter is automatically reset to "0" when the required number of workpieces ($AC_REQUIRED_PARTS) is reached.
$AC_SPECIAL_PARTS	Number of workpieces selected by the user This counter supports user-specific workpiece counts. An alarm can be defined to be output when the setpoint number of workpieces is reached ($AC_REQUIRED_PARTS). Users must reset the counter themselves.

Note

All workpiece counters are set to "0" when the control runs up with default values and can be read and written independent of their activation.

10.20.2.2 Commissioning

Activation

The workpiece counter is activated with the machine data:

MD27880 $MC_PART_COUNTER (activation of workpiece counters)

Bit	Value	Meaning
0	1	$AC_REQUIRED_PARTS is active
1	0	Alarm/signal output for: $AC_ACTUAL_PARTS = $AC_REQUIRED_PARTS
	1	Alarm/signal output for: $AC_SPECIAL_PARTS = $AC_REQUIRED_PARTS
4	1	$AC_TOTAL_PARTS is active.
5	0	$AC_TOTAL_PARTS is incremented by the value "1" through M02/M30.
	1	$AC_TOTAL_PARTS is incremented by the value "1" through the M command defined with MD27882[**0**].
6	0	$AC_TOTAL_PARTS is also active for program test / block search.
7	1	$AC_TOTAL_PARTS is incremented by the value "1" upon a jump back with `GOTOS`.

Bit	Value	Meaning
8	1	$AC_ACTUAL_PARTS is active
9	0	$AC_ACTUAL_PARTS is incremented by the value "1" through M02/M30.
	1	$AC_ACTUAL_PARTS is incremented by the value "1" through the M command defined with MD27882[**1**].
10	0	$AC_ACTUAL_PARTS is also active for program test / block search.
11	1	$AC_ACTUAL_PARTS is incremented by the value "1" upon a jump back with `GOTOS`.
12	1	$AC_SPECIAL_PARTS is active.
13	0	$AC_SPECIAL_PARTS is incremented by the value "1" through M02/M30.
	1	$AC_SPECIAL_PARTS is incremented by the value 1 through the M command defined with MD27882[**2**].
14	0	$AC_SPECIAL_PARTS is also active for program test / block search.
15	1	$AC_SPECIAL_PARTS is incremented by the value "1" upon a jump back with `GOTOS`.

Parameter assignment

Workpiece counting with user-defined M command

If the corresponding bit is set in MD27880, then the count pulse can be triggered via a user-defined M command parameterized via the following machine data instead of via the end of program `M2`/`M30`.

MD27882 $MC_PART_COUNTER_MCODE[<n>] (workpiece counting with user-defined M command)

<n>	Meaning
0	MD27882[**0**] defines the M command in which $AC_**TOTAL**_PARTS is incremented.
1	MD27882[**1**] defines the M command in which $AC_**ACTUAL**_PARTS is incremented.
2	MD27882[**2**] defines the M command in which $AC_**SPECIAL**_PARTS is incremented.

The respective workpiece counter is incremented by "1", when a user-defined M command is called.

Protection level for workpiece counting

Using the following machine data, the protection level for activating/deactivating the workpiece counting at the user interface is set.

MD51074 $MN_ACCESS_WRITE_WPC_COUNTER = <protection level>

10.20.2.3 Supplementary conditions

Mode change / NC RESET

The counters are not affected by a mode change or NC RESET.

$AC_REQUIRED_PARTS ≤ 0

Where $AC_REQUIRED_PARTS ≤ 0 and MD27880 $MC_PART_COUNTER, bit 0 = 1, the counting procedure and the identity comparison set with MD27880 are **not** conducted for all the active counters.

10.20.2.4 Examples

Activation of the workpiece counter $AC_REQUIRED_PARTS:

- MD27880 $MC_PART_COUNTER = 'H3'

$AC_REQUIRED_PARTS is active

Display alarm for: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

Activation of the workpiece counter $AC_TOTAL_PARTS:

- MD27880 $MC_PART_COUNTER = 'H10'
- MD27882 $MC_PART_COUNTER_MCODE[0] = 80

$AC_TOTAL_PARTS is active; the counter is incremented by the value 1 with each `M02`.

$MC_PART_COUNTER_MCODE[0] has no significance.

Activation of the workpiece counter $AC_ACTUAL_PARTS:

- MD27880 $MC_PART_COUNTER = 'H300'
- MD27882 $MC_PART_COUNTER_MCODE[1] = 17

$AC_TOTAL_PARTS is active; the counter is incremented by a value of "1" with each `M17`.

Activation of the workpiece counter $AC_SPECIAL_PARTS:

- MD27880 $MC_PART_COUNTER = 'H3000'
- MD27882 $MC_PART_COUNTER_MCODE[2] = 77

$AC_SPECIAL_PARTS is active.

With each `M77` the following is realized: $AC_SPECIAL_PARTS + 1

Deactivation of the workpiece counter $AC_ACTUAL_PARTS:

- MD27880 $MC_PART_COUNTER = 'H200'
- MD27882 $MC_PART_COUNTER_MCODE[1] = 50

$AC_ACTUAL_PARTS is inactive

Activation of all counters:

- MD27880 $MC_PART_COUNTER = 'H3313'
- MD27882 $MC_PART_COUNTER_MCODE[0] = 80
- MD27882 $MC_PART_COUNTER_MCODE[1] = 17
- MD27882 $MC_PART_COUNTER_MCODE[2] = 77

$AC_REQUIRED_PARTS is active

Display alarm for: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

$AC_TOTAL_PARTS is active; the counter is incremented by the value 1 with each `M02`.

$MC_PART_COUNTER_MCODE[0] has no significance.

$AC_ACTUAL_PARTS is active; the counter is incremented by a value of "1" with each `M17`.

$AC_SPECIAL_PARTS is active; the counter is incremented by a value of "1" with each `M77`.

Workpiece counter $AC_ACTUAL_PARTS is not processed during the program test / block search:

- MD27880 $MC_PART_COUNTER = 'H700'
- MD27882 $MC_PART_COUNTER_MCODE[1] = 75

$AC_ACTUAL_PARTS is active; the counter is incremented by a value of "1" with each `M75`, apart from during the program test and search.

Cancellation of the count modes in the MD27880 $MC_PART_COUNTER with bit 0 = 1:

- MD27882 $MC_PART_COUNTER_MCODE[0] = 41
- MD27882 $MC_PART_COUNTER_MCODE[1] = 42
- MD27882 $MC_PART_COUNTER_MCODE[2] = 43

Program code	Comment
...	
N100 $AC_REQUIRED_PARTS=-10	; value <0: Set counting.
N200 M41 M43	; no counting.
N300 M42	
...	
N500 $AC_REQUIRED_PARTS=52	; value > 0: Counting in accordance with MD27880 activated.
N501 M43	; counting.
N502 M42 M41	; counting.
...	

10.21 Data lists

10.21.1 Function

Various counters with a range of values from 0 to 999,999,999 are available with the "Workpiece counter" function in the form of channel-specific system variables. Read and write access to the system variables is possible.

The following channel-specific machine data can be used to control counter activation, counter reset timing and the counting algorithm.

System variables for workpiece counting

System variable	Meaning
$AC_REQUIRED_PARTS	Number of workpieces to be produced (setpoint number of workpieces) In this counter the number of workpieces at which the actual workpiece count ($AC_ACTUAL_PARTS) will be reset to "0" can be defined. MD27880 can be used to activate the generation of the display alarm: "Channel %1 workpiece target = %2 reached" and of the channel-specific NC/PLC interface signal: DB21, DBX317.1 (workpiece target reached) .
$AC_TOTAL_PARTS	Total number of completed workpieces (actual workpiece total) This counter specifies the total number of workpieces produced since the start time. The value is only automatically reset to "0" when the control runs up with default values.
$AC_ACTUAL_PARTS	Number of completed workpieces (actual workpiece total) This counter registers the total number of workpieces produced since the start time. On condition that $AC_REQUIRED_PARTS > 0, the counter is automatically reset to "0" when the required number of workpieces ($AC_REQUIRED_PARTS) is reached.
$AC_SPECIAL_PARTS	Number of workpieces selected by the user This counter supports user-specific workpiece counts. An alarm can be defined to be output when the setpoint number of workpieces is reached ($AC_REQUIRED_PARTS). Users must reset the counter themselves.

Note

All workpiece counters are set to "0" when the control runs up with default values and can be read and written independent of their activation.

10.21.2 Machine data

10.21.2.1 General machine data

Displaying machine data

Number	Identifier: $MM_	Description
SINUMERIK Operate		
9421	MA_AXES_SHOW_GEO_FIRST	Display geo axes of channel first
9422	MA_PRESET_MODE	PRESET / basic offset in JOG.
9423	MA_MAX_SKP_LEVEL	Maximum number of skip levels

NC-specific machine data

Number	Identifier: $MN_	Description
10010	ASSIGN_CHAN_TO_MODE_GROUP	Channel valid in mode group
10125	EES_NC_NAME	NCU name for the generation of unique NC program names in the EES mode
10280	PROG_FUNCTION_MASK	Comparison commands ">" and "<"
10700	PREPROCESSING_LEVEL	Program preprocessing level
10702	IGNORE_SINGLEBLOCK_MASK	Prevent single block stop
10707	PROG_TEST_MASK	Program test modes
10708	SERUPRO_MASK	Block change modes
10710	PROG_SD_RESET_SAVE_TAB	Setting data to be updated
10711	NC_LANGUAGE_CONFIGURATION	Manner of handling the languages, whose related option or function is not activated.
10713	M_NO_FCT_STOPRE	M function with preprocessing stop
10715	M_NO_FCT_CYCLE	M function to be replaced by subprogram
10716	M_NO_FCT_CYCLE_NAME	Subroutine name for M function replacement
10717	T_NO_FCT_CYCLE_NAME	Name of the tool change cycle for T function replacement
10718	M_NO_FCT_CYCLE_PAR	M function replacement with parameters
10719	T_NO_FCT_CYCLE_MODE	Parameter assignment for T function replacement
11450	SEARCH_RUN_MODE	Block search parameter settings
11470	REPOS_MODE_MASK	Repositioning properties
11600	BAG_MASK	Mode group response to ASUB
11602	ASUP_START_MASK	Ignore stop conditions for ASUB
11604	ASUP_START_PRIO_LEVEL	Priorities, starting from which $MN_ASUP_START_MASK is effective
11610	ASUP_EDITABLE	Activating a user-specific ASUB program
11612	ASUP_EDIT_PROTECTION_LEVEL	Protection level of user-specific ASUB program
11620	PROG_EVENT_NAME	Program name for PROG_EVENT
11625	FILE_ONLY_WITH_EXTENSION	When calling the program, only search for files with a file ID
11626	CYCLES_ONLY_IN_CYCDIR	Only search for programs with interface in the cycle directories
11717	D_NO_FCT_CYCLE_NAME	Subroutine name for D function replacement
15700	LANG_SUB_NAME	Name for replacement subprogram
15702	LANG_SUB_PATH	Call path for replacement subprogram
17200	GMMC_INFO_NO_UNIT	Global HMI info (without physical unit)
17201	GMMC_INFO_NO_UNIT_STATUS	Global HMI status info (without physical unit)
18045	EES_MODE_INFO	Mode in which the EES function operates
18360	MM_EXT_PROG_BUFFER_SIZE	FIFO buffer size for execution from external source (DRAM)
18362	MM_EXT_PROG_NUM	Number of program levels that can be processed simultaneously from external

10.21.2.2 Channel-specific machine data

Basic machine data

Number	Identifier: $MC_	Description
20000	CHAN_NAME	Channel name
20050	AXCONF_GEOAX_ASSIGN_TAB	Assignment of geometry axis to channel axis
20060	AXCONF_GEOAX_NAME_TAB	Channel axis name in the 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	Basic setting of master spindle in channel
20100	DIAMETER_AX_DEF	Geometry axis with transverse axis function
20106	PROG_EVENT_IGN_SINGLEBLOCK	Prog events ignore the single block
20107	PROG_EVENT_IGN_INHIBIT	Prog events ignore the read-in disable
20108	PROG_EVENT_MASK	Event-driven program calls
20109	PROG_EVENT_MASK_PROPERTIES	Prog events properties
20114	MODESWITCH_MASK	Setting for REPOS
20116	IGNORE_INHIBIT_ASUP	Execute user ASUPs completely in spite of read-in disable
20117	IGNORE_SINGLEBLOCK_ASUP	Process user ASUPs completely in spite of single-block processing
20160	CUBIC_SPLINE_BLOCKS	Number of blocks for C spline
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
20191	IGN_PROG_STATE_ASUP	Do not display the execution of the interrupt routine on the operator panel
20192	PROG_EVENT_IGN_PROG_STATE	Do not display the execution of the program events on the operator panel
20193	PROG_EVENT_IGN_STOP	Prog events ignore the Stop key
20194	IGNORE_NONCSTART_ASUP	ASUP start permitted despite pending alarm response "NC Start disable" for certain user alarms
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value for DISC
20230	CUTCOM_CURVE_INSERT_LIMIT	Maximum angle for intersection calculation with tool radius compensation
20240	CUTCOM_MAXNUM_CHECK_BLOCKS	Blocks for predictive contour calculation with tool radius compensation
20250	CUTCOM_MAXNUM_DUMMY_BLOCKS	Maximum number of blocks without traversing motion for TRC
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20400	LOOKAH_USE_VELO_NEXT_BLOCK	Look Ahead to programmed following block velocity
20430	LOOKAH_NUM_OVR_POINTS	Number of override points for Look Ahead
20440	LOOKAH_OVR_POINTS	Override switch points for LookAhead
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity

Number	Identifier: $MC_	Description
20600	MAX_PATH_JERK	Pathrelated maximum jerk
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid motions
20700	REFP_NC_START_LOCK	NC start disable without reference point
20750	ALLOW_GO_IN_G96	G0 logic for G96, G961
20800	SPF_END_TO_VDI	Subprogram end to PLC
21000	CIRCLE_ERROR_CONST	Circle end point monitoring constant
21010	CIRCLE_ERROR_FACTOR	Circle end point monitoring factor
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system for automatic Frame definition
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour
21210	SETINT_ASSIGN_FASTIN	HW assignment of the ext. NC input byte for NC program interrupt:
21202	LIFTFAST_WITH_MIRROR	Lift fast with mirror

Block search

Number	Identifier: $MC_	Description
20128	COLLECT_TOOL_CHANGE	Tool change commands to the PLC after block search
22600	SERUPRO_SPEED_MODE	Velocity for block search type 5
22601	SERUPRO_SPEED_FACTOR	Velocity factor for block search type 5
22621	ENABLE_START_MODE_MASK_PRT	Switches MD22620: START_MODE_MASK_ PRT free for SERUPRO block search run
22622	DISABLE_PLC_START	Allow part program start via PLC
22680	AUTO_IPTR_LOCK	Disable interrupt pointer

Reset response

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Initial setting after RESET / parts program end
20112	START_MODE_MASK	Basic setting for NC start after part program start
20118	GEOAX_CHANGE_RESET	Allow automatic geometry axis change
20120	TOOL_RESET_VALUE	Tool length compensation when powering-up (RESET / part program end)
20121	TOOL_PRESEL_RESET_VALUE	Preselected tool on RESET
20130	CUTTING_EDGE_RESET_VALUE	Tool cutting edge length compensation when powering-up (RESET / part program end)
20140	TRAFO_RESET_VALUE	Transformation data set when powering-up (RESET / part program end)
20150	GCODE_RESET_VALUES	Initial setting of the G groups
20152	GCODE_RESET_MODE	Reset behavior of G groups

Number	Identifier: $MC_	Description
20156	MAXNUM_GCODES_EXT	Reset behavior of the external G groups
22620	START_MODE_MASK_PRT	Initial setting at special NC Start after power-up and at RESET

Auxiliary function settings

Number	Identifier: $MC_	Description
22000	AUXFU_ASSIGN_GROUP	Auxiliary function group
22010	AUXFU_ASSIGN_TYPE	Auxiliary function type
22020	AUXFU_ASSIGN_EXTENSION	Auxiliary function extension
22030	AUXFU_ASSIGN_VALUE	Auxiliary function value
22200	AUXFU_M_SYNC_TYPE	Output timing of M functions
22210	AUXFU_S_SYNC_TYPE	Output timing of S functions
22220	AUXFU_T_SYNC_TYPE	Output timing of T functions
22230	AUXFU_H_SYNC_TYPE	Output timing of H functions
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions
22250	AUXFU_D_SYNC_TYPE	Output timing of D functions
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET
22410	F_VALUES_ACTIVE_AFTER_RESET	F function active after RESET
22510	GCODE_GROUPS_TO_PLC	G commands that are output to the NC/PLC interface on block change / RESET
22550	TOOL_CHANGE_MODE	New tool offset for M function
22560	TOOL_CHANGE_M_CODE	M function for tool change

Memory settings

Number	Identifier: $MC_	Description
27900	REORG_LOG_LIMIT	Percentage of IPO buffer for log file enable
28000	MM_REORG_LOG_FILE_MEM	Memory size for REORG (DRAM)
28010	MM_NUM_REORG_LUD_MODULES	Number of blocks for local user variables for REORG
28020	MM_NUM_LUD_NAMES_TOTAL	Number of local user variables (DRAM)
28040	MM_LUD_VALUES_MEM	Memory size for local user variables (DRAM)
28050	MM_NUM_R_PARAM	Number of channelspecific R parameters (SRAM)
28060	MM_IPO_BUFFER_SIZE	Number of NC blocks in IPO buffer (DRAM)
28070	MM_NUM_BLOCKS_IN_PREP	Number of blocks for block preparation (DRAM)
28080	MM_NUM_USER_FRAMES	Number of settable Frames (SRAM)
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles (DRAM)
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for compile cycles (DRAM)
28400	MM_ABSBLOCK	Activating basis blocks with absolute values
28402	MM_ABSBLOCK_BUFFER_CONF	Dimension size of upload buffer
28500	MM_PREP_TASK_STACK_SIZE	Stack size of preparation task (DRAM)

Program runtime and workpiece counter

Number	Identifier: $MC_	Description
27860	PROCESSTIMER_MODE	Activate the runtime measurement
27880	PART_COUNTER	Activate the workpiece counter
27882	PART_COUNTER_MCODE[ ]	Workpiece counting via M command

10.21.2.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30465	AXIS_LANG_SUB_MASK	Substitution of NC language commands
30550	AXCONF_ASSIGN_MASTER_CHAN	Reset position of channel for axis change
30600	FIX_POINT_POS	Fixed value positions of axes with G75
33100	COMPRESS_POS_TOL	Maximum deviation with compensation

10.21.3 Setting data

10.21.3.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42010	THREAD_RAMP_DISP	Acceleration behavior of axis when thread cutting
42100	DRY_RUN_FEED	Dry run feedrate
42200	SINGLEBLOCK2_STOPRE	Activate debug mode for SBL2
42444	TARGET_BLOCK_INCR_PROG	Continuation mode after block search with calculation
42700	EXT_PROG_PATH	Program path for external subroutine call EXTCALL
42750	ABSBLOCK_ENABLE	Enable basic block display
42990	MAX_BLOCKS_IN_IPOBUFFER	Maximum number of blocks in the interpolation buffer

10.21.4 Signals

10.21.4.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Emergency stop	DB10.DBX56.1	DB2600.DBX0.1

10.21.4.2 Signals to mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
AUTOMATIC mode	DB11.DBX0.0	DB3000.DBX0.0
MDA mode	DB11.DBX0.1	DB3000.DBX0.1
JOG mode	DB11.DBX0.2	DB3000.DBX0.2
Mode change disable	DB11.DBX0.4	DB3000.DBX0.4
Mode group stop	DB11.DBX0.5	DB3000.DBX0.5
Mode group stop axes plus spindles	DB11.DBX0.6	DB3000.DBX0.6
Mode group reset	DB11.DBX0.7	DB3000.DBX0.7
Machine function teach in	DB11.DBX1.0	DB3000.DBX1.0
Machine function REPOS	DB11.DBX1.1	-
Machine function REF	DB11.DBX1.2	DB3000.DBX1.2

10.21.4.3 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Selected mode AUTOMATIC	DB11.DBX4.0	-
Selected mode MDA	DB11.DBX4.1	-
Selected JOG mode	DB11.DBX4.2	-
Selected machine function teach in	DB11.DBX5.0	-
Selected machine function REPOS	DB11.DBX5.1	-
Selected machine function REF	DB11.DBX5.2	-
Active mode AUTOMATIC	DB11.DBX6.0	DB3100.DBX0.0
Active mode MDA	DB11.DBX6.1	DB3100.DBX0.1
Active mode JOG	DB11.DBX6.2	DB3100.DBX0.2
Mode group ready	DB11.DBX6.3	DB3100.DBX0.3
Mode group reset performed	DB11.DBX6.4	DB3100.DBX0.4
NC internal JOG active	DB11.DBX6.5	-
All channels in the reset state	DB11.DBX6.7	DB3100.DBX0.7
Active machine function teach in	DB11.DBX7.0	DB3100.DBX1.0
Active machine function REPOS	DB11.DBX7.1	-
Active machine function REF	DB11.DBX7.2	DB3100.DBX1.2

10.21.4.4 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate DRF	DB21, ...DBX0.3	DB320x.DBX0.3
Activate single block	DB21, ...DBX0.4	DB320x.DBX0.4
Activate M01	DB21, ...DBX0.5	DB320x.DBX0.5
Activate dry run feedrate	DB21, ...DBX0.6	DB320x.DBX0.6

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
PLC action completed	DB21, ...DBX1.6	-
Activate program test	DB21, ...DBX1.7	DB320x.DBX1.7
Skip block levels: /0 to /7	DB21, ...DBX2.0-7	DB320x.DBX2.0-7
Read-in disable	DB21, ...DBX6.1	DB320x.DBX6.1
Program level abort	DB21, ...DBX6.4	DB320x.DBX6.4
NC start disable	DB21, ...DBX7.0	DB320x.DBX7.0
NC start	DB21, ...DBX7.1	DB320x.DBX7.1
NC Stop at block limit	DB21, ...DBX7.2	DB320x.DBX7.2
NC stop	DB21, ...DBX7.3	DB320x.DBX7.3
NC Stop axes plus spindles	DB21, ...DBX7.4	DB320x.DBX7.4
Reset	DB21, ...DBX7.7	DB330x.DBX3.7
REPOSPATHMODE	DB21, ...DBX31.0-2	-
REPOSMODEEDGE	DB21, ...DBX31.4	-

10.21.4.5 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
DRF selected	DB21, ...DBX24.3	DB170x.DBX0.3
Select NC associated M01	DB21, ...DBX24.4	-
M01 selected	DB21, ...DBX24.5	DB170x.DBX0.5
Dry run feedrate selected	DB21, ...DBX24.6	DB170x.DBX0.6
REPOSPATHMODE 0 - 2	DB21, ...DBX25.0-2	-
Feedrate override selected for rapid traverse	DB21, ...DBX25.3	DB170x.DBX1.3
REPOS MODE EDGE	DB21, ...DBX25.4	-
Program test selected	DB21, ...DBX25.7	DB170x.DBX1.7
Skip block selected: /0 - /7	DB21, ...DBX26.0-7	DB170x.DBX2.0-7
Skip block selected: /8	DB21, ...DBX27.0	DB170x.DBX3.0
Skip block selected: /9	DB21, ...DBX27.1	DB170x.DBX3.1
REPOSPATHMODE 0 - 2	DB21, ...DBX31.0-2	-
REPOS MODE EDGE	DB21, ...DBX31.4	-
Skip block active /8	DB21, ...DBX31.6	DB320x.DBX15.6
Skip block active /9	DB21, ...DBX31.7	DB320x.DBX15.7
Execution from external active	DB21, ...DBX32.0	DB330x.DBX0.0
Action block active	DB21, ...DBX32.3	DB330x.DBX0.3
Approach block active	DB21, ...DBX32.4	DB330x.DBX0.4
M0/M1 active	DB21, ...DBX32.5	DB330x.DBX0.5
Last action block active	DB21, ...DBX32.6	DB330x.DBX0.6
Block search active	DB21, ...DBX33.4	DB330x.DBX1.4
M02/M30 active	DB21, ...DBX33.5	DB330x.DBX1.5
Transformation active	DB21, ...DBX33.6	DB330x.DBX1.6

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Program test active	DB21, ...DBX33.7	DB330x.DBX1.7
Program state: Running	DB21, ...DBX35.0	DB330x.DBX3.0
Program state: Waiting	DB21, ...DBX35.1	DB330x.DBX3.1
Program state: Stopped	DB21, ...DBX35.2	DB330x.DBX3.2
Program state: Interrupted	DB21, ...DBX35.3	DB330x.DBX3.3
Program state: Aborted	DB21, ...DBX35.4	DB330x.DBX3.4
Channel state: Active	DB21, ...DBX35.5	DB330x.DBX3.5
Channel state: Interrupted	DB21, ...DBX35.6	DB330x.DBX3.6
Channel state: Reset	DB21, ...DBX35.7	DB330x.DBX3.7
Interrupt handling active	DB21, ...DBX36.4	-
Channel is ready	DB21, ...DBX36.5	DB330x.DBX4.5
Read-in enable is ignored	DB21, ...DBX37.6	-
Stop at the end of block with SBL is suppressed	DB21, ...DBX37.7	-
Number of the active G command of G group 1 – n (8-bit integer)	DB21, ...DBB208-271	DB350x.DBB0-63
Workpiece setpoint reached	DB21, ...DBX317.1	DB330x.DBX4001.1
ASUP is stopped	DB21, ...DBX318.0	DB330x.DBX4002.0
Block search via program test is active	DB21, ...DBX318.1	-
REPOS MODE EDGEACKN	DB21, ...DBX319.0	-
Repos Path Mode Ackn: 0 - 2	DB21, ...DBX319.1-3	-
Repos DEFERAL Chan	DB21, ...DBX319.5	-
Display of the triggering event in case of event-driven program call	DB21, ...DBX376.0-7	DB330x.DBB4004
ASUP is active	DB21, ...DBX378.0	DB330x.DBB4006.0
ASUP with suppressed display update is active	DB21, ...DBX378.1	DB330x.DBB4006.1

10.21.4.6 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
REPOSDELAY	DB31, ...DBX10.0	-

10.21.4.7 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
REPOS offset	DB31, ...DBX70.0	-
REPOS offset valid	DB31, ...DBX70.1	-
REPOS Delay Ackn	DB31, ...DBX70.2	-
REPOSDELAY	DB31, ...DBX72.0	-
Path axis	DB31, ...DBX76.4	DB390x.DBX1002.4

K2: Axis Types, Coordinate Systems, Frames 11

11.1 Brief description

11.1.1 Axes

Machine axes

Machine axes are the axes that actually exist on a machine tool.

Channel axes

Every geometry axis and every special axis is assigned to a channel and, therefore, a channel axis. Geometry axes and additional axes are always traversed in "their" channel.

Geometry axes

The three geometry axes always make up a fictitious rectangular coordinate system, the basic coordinate system (BCS).

By using FRAMES (offset, rotation, scaling, mirroring), it is possible to image geometry axes of the workpiece coordinate system (WCS) on the BCS.

Special axes

In contrast to geometry axes, no geometrical relationship is defined between the special axes.

Path axes

Path axes are interpolated together (all the path axes of a channel have a common path interpolator).

All the path axes of one channel have the same acceleration phase, constant travel phase and delay phase.

Positioning axes

Positioning axes are interpolated separately (each positioning axis has its own axis interpolator). Each positioning axis has its own feedrate and acceleration characteristic.

Synchronized axes

Synchronous axes are interpolated together with path axes (all path axes and synchronous axes of one channel have a common path interpolator).

All path axes and all synchronous axes of a channel have the same acceleration phase, constant travel phase and deceleration phase.

Axis configuration

The machine data below are used to assign the geometry axes, special axes, channel axes and machine axes as well as the names of the individual axis types:

MD20050 $MC_AXCONF_GEOAX_ASIGN_TAB (assignment of geometry axis to channel axis)

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

MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in channel)

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

MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX (assignment of spindle to machine axis)

Replaceable geometry axes

The "Replaceable geometry axes" function allows the geometry axes in a grouping to be replaced by other channel axes.

Axes that are initially configured as synchronous special axes in a channel can replace any selected geometry axis in response to a program command.

Link axis

Link axes are axes, which are physically connected to another NCU and whose position is controlled from this NCU. Link axes can be assigned dynamically to channels of **another** NCU. Link axes are not local axes from the perspective of a particular NCU.

The **axis container** concept is used for the dynamic modification of the assignment to an NCU. Axis replacement with `GET` and `RELEASE` from the part program is not available for link axes across NCU boundaries.

The link axes are described in
References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

Axis container

An axis container is a circular buffer data structure, in which local axes and/or link axes are assigned to channels. The entries in the circular buffer can be **shifted cyclically**.

In addition to the direct reference to local axes or link axes, the link axis configuration in the logical machine axis image also allows references to axis containers.

This type of reference consists of:

- Axis container number
- A slot (circular buffer location within the corresponding container)

The entry in a circular buffer location contains:

- A local axis
  or
- A link axis

The axis container function is described in
References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

11.1.2 Coordinate systems

MCS

The machine coordinate system (MCS) has the following properties:

- It is defined by the machine axes.
- The machine axes can be perpendicular to each other to form Cartesian system or arranged in any other way.
- The names of the machine axes can be defined.
- The machine axes can be linear or rotary axes.

BCS

The basic coordinates system (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.

External work offset

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 + 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.

11.1.3 Frames

A frame is a closed calculation rule (algorithm) that translates one Cartesian coordinate system into another.

Frame components

A frame consists of the following components:

Frame components		Programmable with:
Offset	Coarse offset	`TRANS` `ATRANS` (additive translation component) `CTRANS` (work offset for multiple axes) `G58` (axial work offset)
	Fine offset	`CFINE` `G59` (axial work offset)

Frame components	Programmable with:

Rotation	`ROT` / `ROTS`
	`AROT` / `AROTS`
	`CROTS`

Scaling	`SCALE` `ASCALE`
Mirroring	`MIRROR` `AMIRROR`

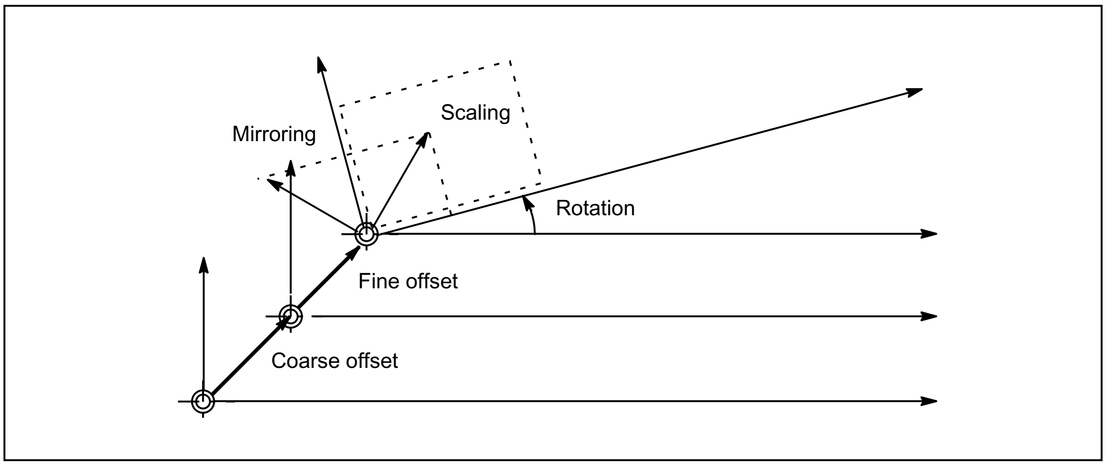


Figure 11-1 Frame components

Coarse and fine offsets

As the assignment of machine axes to channel axes and, in particular, to geometry axes, can be different in all channels, there is consequently no unique cross-channel geometric relationship between the channel axes. Therefore, only offset, scaling and mirroring is possible for NCU-global frames. Rotations are not possible.

G58: Absolute axis-specific programmable work offset (coarse offset)

The absolute component of the translatory offset (coarse offset) is programmed axis-specifically with `G58`. The additive component of the translatory offset (fine offset) is retained.

G59: Additive axis-specific programmable work offset (fine offset)

The additive component of the translatory offset (fine offset) is programmed axis-specifically with `G59`. The absolute component of the translatory offset (coarse offset) is retained.

`G59` can only be used when the fine offset has been released:

- MD18600 $MN_MM_FRAME_FINE_TRANS = TRUE
- MD24000 $MC_FRAME_ADD_COMPONENTS = TRUE

Rotation

Orientations in space are defined via frame rotations as follows:

- Rotation with `ROT` defines the individual rotations for all geometry axes.
- Solid angles with `ROTS`, `AROTS`, `CROTS` define the orientation of a plane in space.
- Frame rotation with `TOFRAME` defines a frame with a Z axis pointing in the tool direction.

Scaling

`SCALE` is used to program the programmable scale factors for all geometry axes and special axes.

If a new scaling is to be based on a previous scaling, rotation, translation or mirroring, then `ASCALE` must be programmed.

Mirroring

The following machine data is used to set how the mirroring is to be performed:

MD10610 $MN_MIRROR_REF_AX

Concatenation

Frames and frame components can be combined using the concatenation operator "`:`". The channel-specific active frame `$P_ACTFRAME` results, for example, from the chaining of all active frames of the channel:

```
$P_ACTFRAME =    $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
                 $P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
                 $P_ACTBFRAME : $P_IFRAME : $P_GFRAME :
                 $P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME:
                 $P_PFRAME : $P_ISO4FRAME : $P_CYCFRAME
```

Supplementary conditions

Incremental dimensioning G91

Incremental programming with `G91` is defined such that the offset value is traversed additively to the incrementally programmed value when a work offset is selected.

The behavior depends on the setting in the setting data:

SD42440 $SC_FRAME_OFFSET_INCR_PROG (work offsets in frames)

Value	Meaning
1	Work offset is applied on `FRAME` and incremental programming of an axis (= default setting).
0	Only the programmed path is traversed.

Consistency

When writing, reading and activating frames, e.g. using channel coordination, the user is solely responsible for achieving consistent behavior within the channels. Cross-channel activation of frames is not supported.

11.2 Axes

11.2.1 Overview

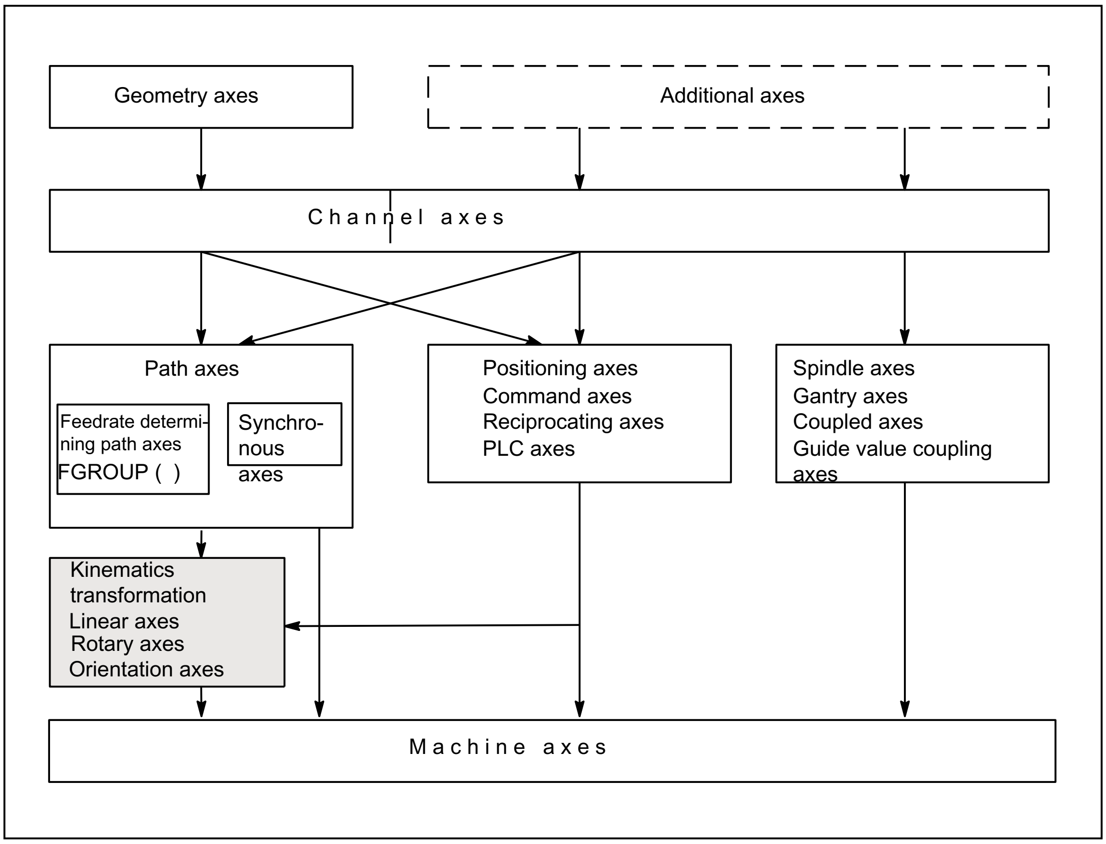


Figure 11-2 Relationship between geometry axes, special axes and machine axes

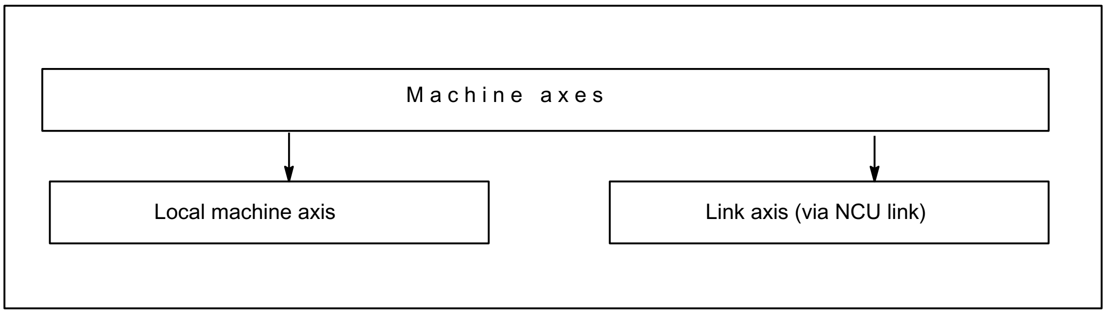


Figure 11-3 Local and external machine axes (link axes)

11.2.2 Machine axes

Meaning

Machine axes are the axes that actually exist on a machine tool.

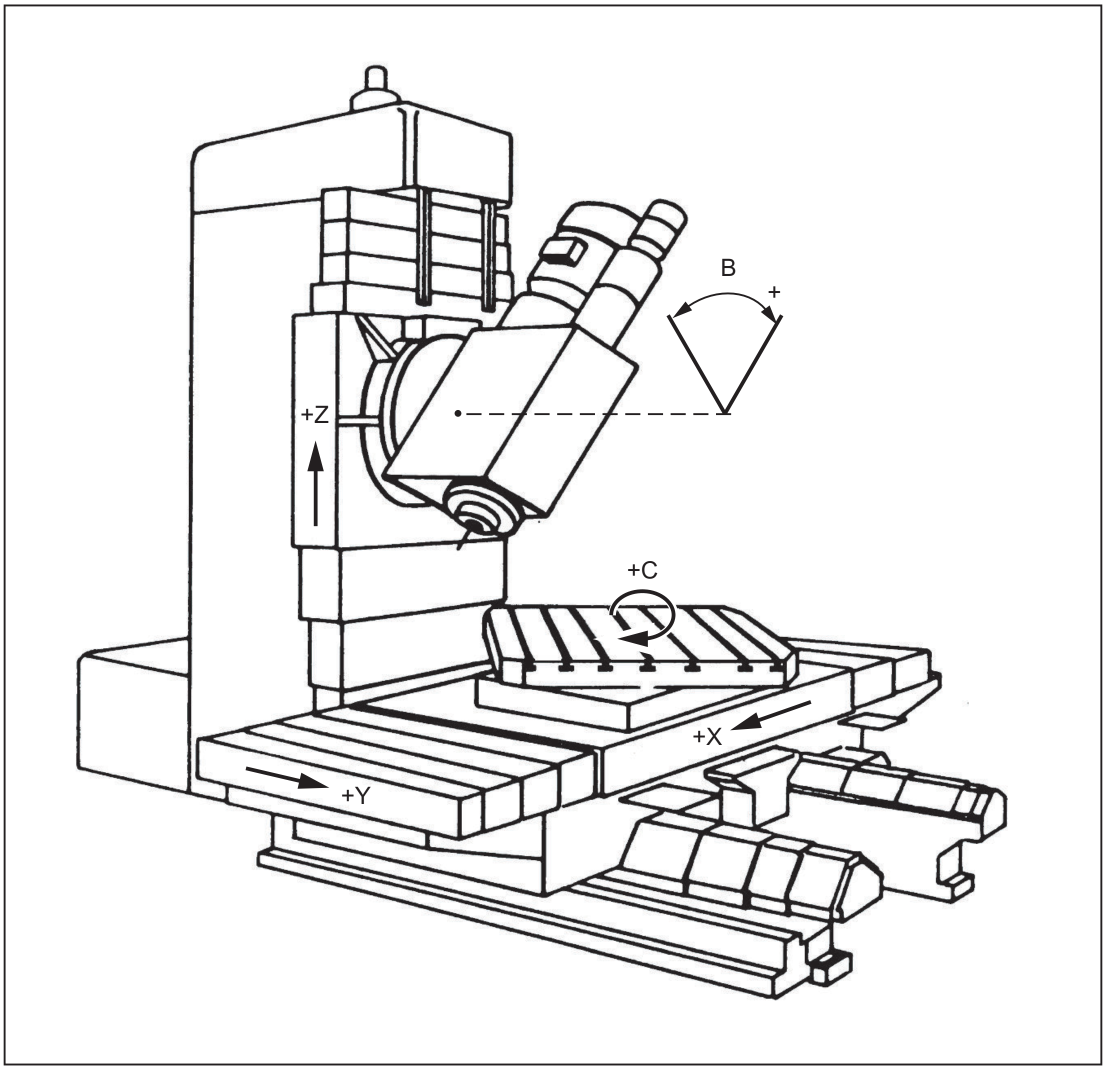


Figure 11-4 Machine axes X, Y, Z, B, S on a Cartesian machine

Application

The following can be machine axes:

- Geometry axes X, Y, Z
- Orientation axes A, B, C

- Loader axes
- Tool turrets
- Axes for tool magazine
- Axes for automatic tool changer
- Spindle sleeves
- Axes for pallet changers
- Etc.

11.2.3 Channel axes

Meaning

Each geometry axis and each special axis is assigned to a channel. Geometry axes and additional axes are always traversed in "their" channel.

11.2.4 Geometry axes

Meaning

The three geometry axes always make up a fictitious rectangular coordinate system.

By using FRAMES (offset, rotation, scaling, mirroring), it is possible to image geometry axes of the workpiece coordinate system (WCS) on the BCS.

Application

Geometry axes are used to program the workpiece geometry (the contour).

Plane selection `G17`, `G18` and `G19` (DIN 66217) always refers to the three geometry axes. That is why it is advantageous to name the three geometry axes X, Y and Z.

11.2.5 Special axes

Significance

In contrast to geometry axes, no geometrical relationship is defined between the special axes.

Note

Geometry axes have an exactly defined relationship in the form of a rightangled coordinate system.

Special axes are part of the basic coordinate system (BCS). With FRAMES (translation, scaling, mirroring), special axes of the workpiece coordinate system can be mapped on the basic coordinate system.

Application

Typical special axes are:

- Rotary axes
- Machine tool axes
- Tool revolver axes
- Loader axes

11.2.6 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.

11.2.7 Positioning axes

Meaning

Positioning axes are interpolated separately (each positioning axis has its own axis interpolator). Each positioning axis has its own feedrate and acceleration characteristic. Positioning axes can be programmed in addition to path axes (even in the same block). Path axis interpolation (path interpolator) is not affected by the positioning axes. Path axes and the individual positioning axes do not necessarily reach their block end points at the same time.

Instructions `POS` and `POSA` are used to program positioning axes and define block change criteria:

- `POS`
  Block change takes place when the path axes and positioning axes have reached their block end points.
- `POSA`
  Block change takes place when the path axes have reached their end of block position. Positioning axes continue to traverse beyond block limits to their block end point.

Concurrent positioning axes differ from positioning axes in that they:

- Only receive their block end points from the PLC
- Can be started at any time (not at block limits)
- Do not affect the execution of current part programs.

Application

Typical positioning axes are:

- Loaders for moving workpieces away from machine
- Tool magazine/turret

Reference

For further information, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)" and "S1: Spindles (Page 1273)".

References:

- Function Manual, Extended Functions; Positioning Axes (P2)
- Function Manual, Special Functions; Gantry Axes (G1)
- Function Manual, Special Functions; Axis Couplings and ESR (M3)
- Function Manual, Extended Functions; Oscillation (P5)
- Function Manual, Synchronized Actions

11.2.8 Main axes

Meaning

A main axis is an axis that is interpolated by the main run.

This interpolation can be started as follows:

- From synchronized actions
  (as command axes due to an event via block-related, modal or static synchronized actions)
- From the PLC via special function blocks in the basic PLC program
  (named as a concurrent positioning axis or a PLC axis)
- Via the setting data or from the part program
  (as an asynchronous or block-synchronous oscillating axis)

Channel control

An axis interpolated by the main axis reacts in terms of:

- NC stop
- Alarm handling
- Program control
- End of program
- RESET

Note

The response at the end of the program varies. The axis movement need not always be completed and, therefore, may carry on beyond the end of the program.

Application

Certain axes in the main run can be decoupled at the channel response triggered by the NC program sequence and controlled from the PLC. These axes are also interpolated in the main run and respond independently for the channel and program sequence.

A PLC-controlled axis can then be controlled independently by the NC. This concerns the following actions:

- The sequence for canceling the axis (equivalent to delete distancetogo)
- Stopping or interrupting the axis
- Continuing the axis (continue sequence of motion)
- Resetting the axis to its basic status

11.2.9 Synchronized axes

Meaning

Synchronous axes are components of the path axes, which are not referenced in order to calculate the tool path velocity. They are interpolated together with path axes (all path axes and synchronous axes of one channel have a common path interpolator).

All path axes and all synchronous axes of a channel have the same acceleration phase, constant travel phase and deceleration phase.

The feedrate (path feedrate) programmed under address F applies to all the path axes programmed in a block but not to the synchronous axes.

Synchronous axes take the same time to cover the programmed path as the path axes.

FGROUP command

The command FGROUP specifies whether the axis is a feed-defining **path axis** (used to calculate the path velocity) or a **synchronous axis** (not used to calculate the path velocity).

Example

Program code	Comment
`N05 G00 G94 G90 M3 S1000 X0 Y0 Z0`	`;`
`N10 FGROUP(X,Y)`	`; Axes X and Y are path axes, Z is a synchronous axis`
`N20 G01 X100 Y100 F1000`	`; Progr. feedrate 1000 mm/min. Feedrate of axis X = 707 mm/min. Feedrate of axis Y = 707 mm/min.`
`N30 FGROUP (X)`	`; Axis X is a path axis, axis Y is a synchronous axis`
`N20 X200 Y150`	`; Progr. Feedrate 1000 mm/min Feedrate of Axis X = 1000 mm/min Feedrate of Axis Y is set to 500 mm/min, because only half the distance is to be traversed.`

Note

The channel axis name must be used for the FGROUP command.

This is defined by the machine data:

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

Application

In the case of helical interpolation FGROUP can be programmed to determine whether:

- The programmed feedrate should be valid on the path
  (all three programmed axes are path axes)
- The programmed feedrate should be valid on the circle
  (two axes are path axes and the infeed axis is a synchronous axis)

11.2.10 Axis configuration

Assigning geometry, special, channel and machine axes and drives

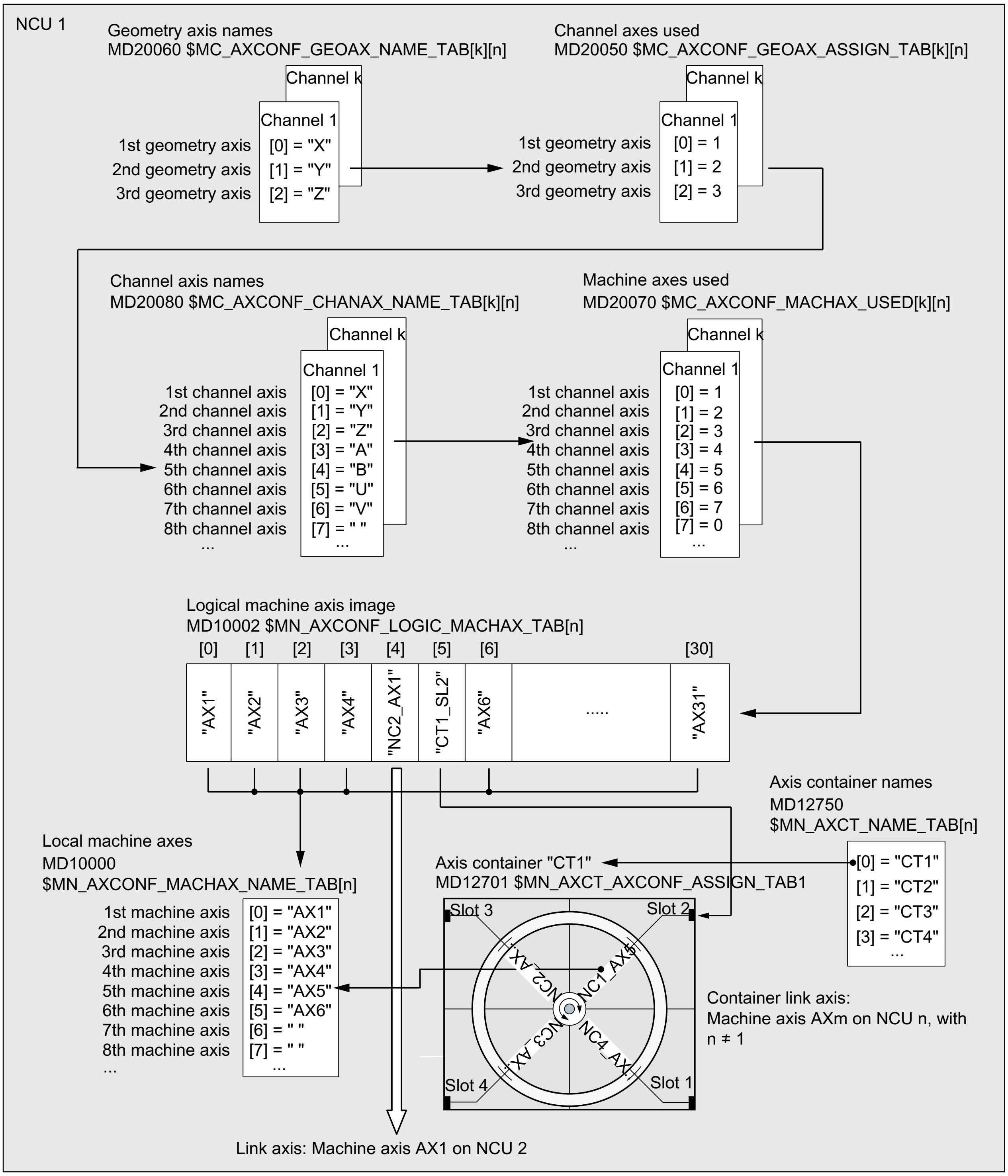

NCU 1
Geometry axis names
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[k][n]
Channel k
Channel 1
1st geometry axis [0] = "X"
2nd geometry axis [1] = "Y"
3rd geometry axis [2] = "Z"
Channel axes used
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[k][n]
Channel k
Channel 1
1st geometry axis [0] = 1
2nd geometry axis [1] = 2
3rd geometry axis [2] = 3
Channel axis names
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[k][n]
Channel k
Channel 1
1st channel axis [0] = "X"
2nd channel axis [1] = "Y"
3rd channel axis [2] = "Z"
4th channel axis [3] = "A"
5th channel axis [4] = "B"
6th channel axis [5] = "U"
7th channel axis [6] = "V"
8th channel axis [7] = " "
...
Machine axes used
MD20070 $MC_AXCONF_MACHAX_USED[k][n]
Channel k
Channel 1
1st channel axis [0] = 1
2nd channel axis [1] = 2
3rd channel axis [2] = 3
4th channel axis [3] = 4
5th channel axis [4] = 5
6th channel axis [5] = 6
7th channel axis [6] = 7
8th channel axis [7] = 0
...
Logical machine axis image
MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[n]
[0] "AX1"
[1] "AX2"
[2] "AX3"
[3] "AX4"
[4] "NC2_AX1"
[5] "CT1_SL2"
[6] "AX6"
.....
[30] "AX31"
Axis container names
MD12750
$MN_AXCT_NAME_TAB[n]
[0] = "CT1"
[1] = "CT2"
[2] = "CT3"
[3] = "CT4"
...
Local machine axes
MD10000
$MN_AXCONF_MACHAX_NAME_TAB[n]
1st machine axis [0] = "AX1"
2nd machine axis [1] = "AX2"
3rd machine axis [2] = "AX3"
4th machine axis [3] = "AX4"
5th machine axis [4] = "AX5"
6th machine axis [5] = "AX6"
7th machine axis [6] = " "
8th machine axis [7] = " "
...
Axis container "CT1"
MD12701 $MN_AXCT_AXCONF_ASSIGN_TAB1
Slot 3
Slot 2
Slot 4
Slot 1
NC2_AX.
NC1_AX5
NC3_AX.
NC4_AX.
Container link axis:
Machine axis AXm on NCU n, with n ≠ 1
Link axis: Machine axis AX1 on NCU 2

Figure 11-5 Axis assignment

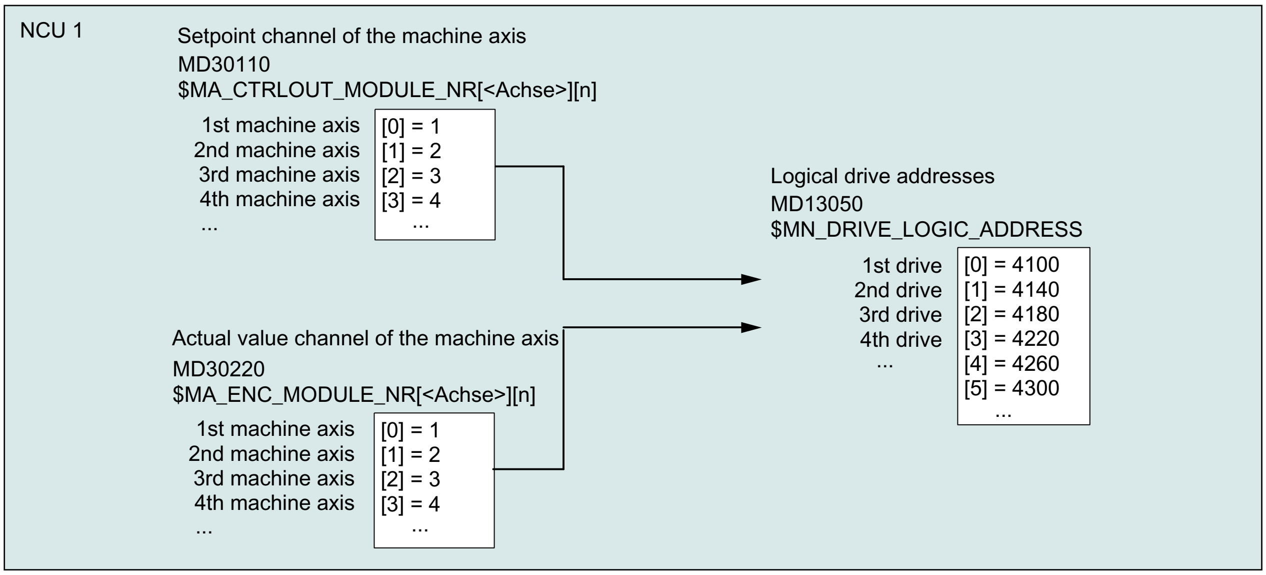


Figure 11-6 Drive assignment

Supplementary conditions

- Leading zeros for user-defined axis names are ignored:
  MD10000 `$MN_AXCONF_MACHAX_NAME_TAB[0] = X01 corresponds to X1
- The geometry axes must be assigned to the channel axes in ascending order without any gaps.
- All channel axes that are not geometry axes are special axes.

Channel axis gaps

Normally, each channel axis defined in machine data MD20080 $MC_AXCONF_CHANAX_NAME_TAB must be assigned a machine axis via MD20070 $MC_AXCONF_MACHAX_USED.

In order to simplify commissioning series of machines with a different number of machine axes, channel axes may also be defined, which are not assigned to any machine axis. As a result, gaps can occur in the numbering sequence of the channel axes.

Any channel axis gaps must be explicitly enabled:

MD11640 $MN_ENABLE_CHAN_AX_GAP = 1

Without being enabled, a value of 0 in machine data MD20070 $MC_AXCONF_MACHAX_USED stops the assignment of further channel axes to machine axes.

Example

Channel axis B is not assigned to a machine axis.

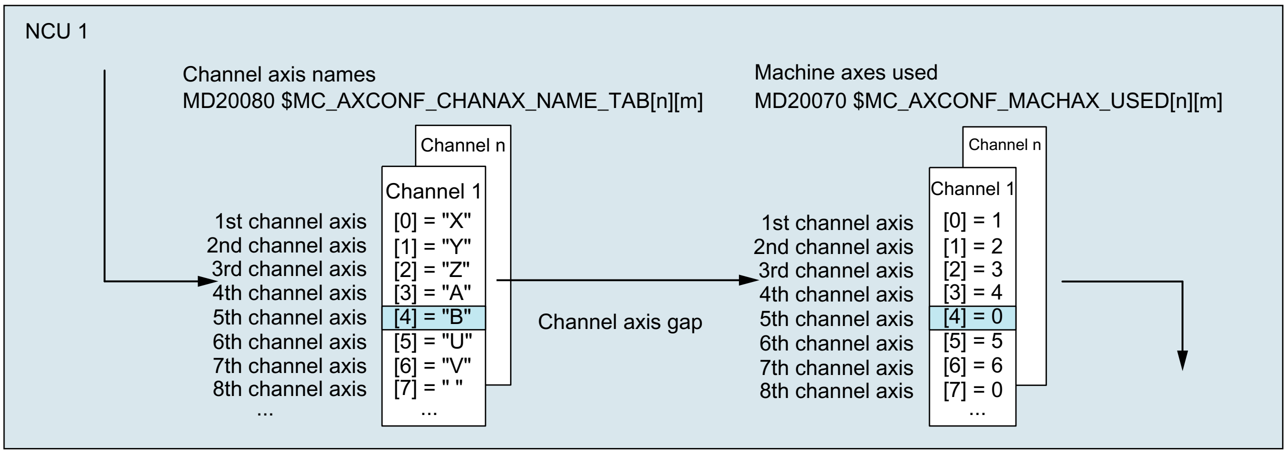


Figure 11-7 Axis configuration with channel axis gap (excerpt)

Supplementary conditions

- Channel axes without assigned machine axes (channel axis gaps) are, regarding the number and indexing of the channel axes, treated just like normal channel axes with associated machine axes.
- If a channel axis without assigned machine axis (channel axis gap) is defined as geometry axis, then this is rejected without an alarm.

11.2.11 Link axes

Meaning

A link axis is a machine axis that is not on the NCU from which it is traversed. The name of a local machine axis is not entered in the machine data for the logical machine axis image of the traversing NCU, but the NCU and machine axis name of the NCU to which it is physically connected.

As an example, machine axis AX1 of NCU2 should be traversed from NCU1:

- NCU1: MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[n] = NC2_AX1

Requirement

The NCUs involved must be connected using link communication as a requirement for using link axes. The link axes and link communication functions are described in detail in:

References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

11.3 Zeros and reference points

11.3.1 Reference points in working space

Zeros and reference points

The neutral position of the machine is obtained from the coordinate axes and the constructive characteristics of the machine. The zero of the coordinate system is obtained by defining a suitable reference point on the machine in its neutral position.

The position of the coordinate systems (MCS, BCS, BZS, SZS, WCS) is determined by means of zeros.

Zero points		Reference points	
⊕	**M** = Machine zero	⊕	**R** = Reference point
⊕	**W** = Workpiece zero	⊕	**T** = Toolholder reference point

Machine zero M

The machine zero M defines the machine coordinate system MCS. All other reference points refer to the machine zero.

Workpiece zero W

The workpiece zero W defines the workpiece coordinate system in relation to the machine zero M. The programmed part-program blocks are executed in the workpiece coordinate system WCS.

Reference point R

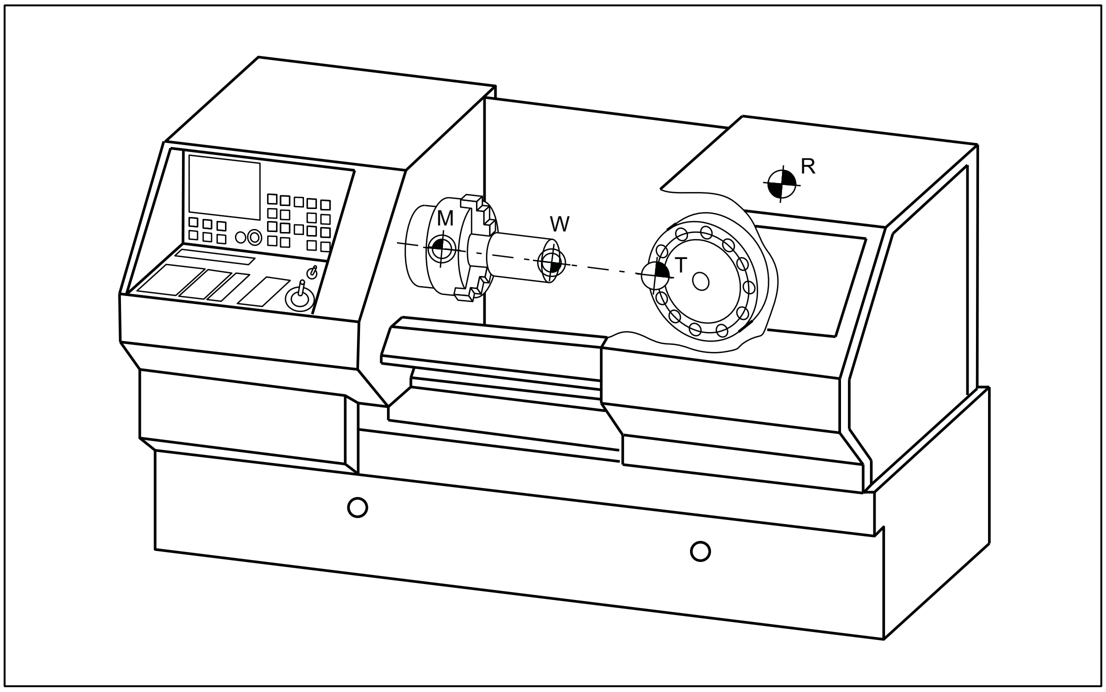
The position of the reference point R is defined by cam switches. Reference point R calibrates the position measuring system.

With incremental encoders, the reference point must be approached every time the control power is switched on. The control can only then work with the measuring system and transfer all position values to the coordinate systems.

Toolholder reference point T

The toolholder reference point T is located on the toolholder locator. By entering the tool lengths, the control calculates the distance between the tool tip (TCP Tool Center Position) and the toolholder reference point.

Example: Zeros and reference points on a turning machine



11.3.2 Position of coordinate systems and reference points

Control POWER ON

For incremental measuring probes, the reference point must be approached each time the control is activated so that the control can transfer all position values to the coordinate system.

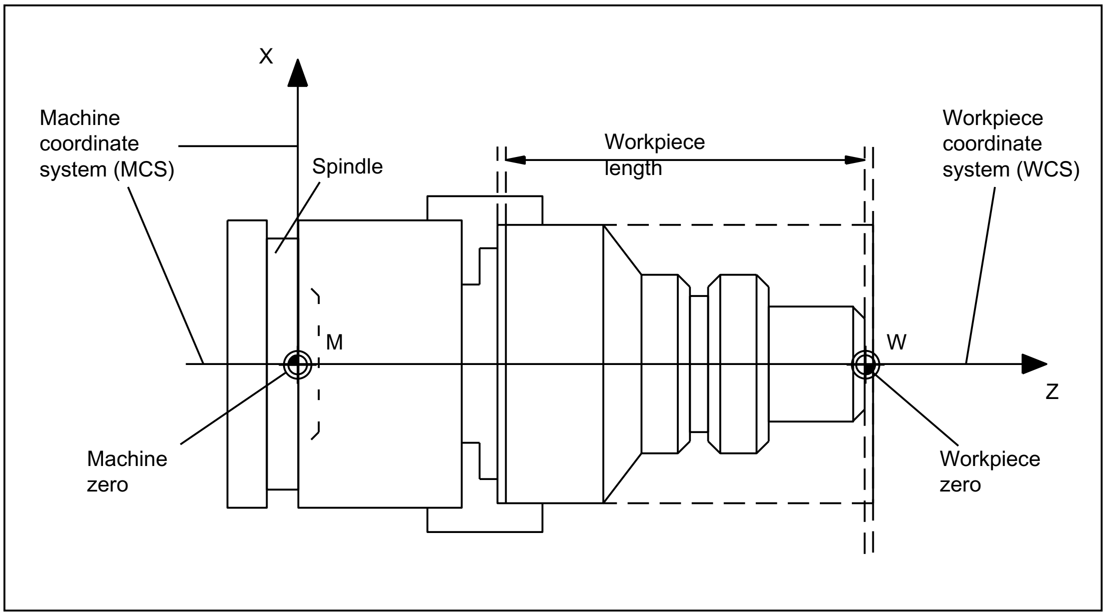

X
Machine coordinate system (MCS)
Spindle
Workpiece length
Workpiece coordinate system (WCS)
M
W
Z
Machine zero
Workpiece zero

Figure 11-8 Position of coordinate systems by machine zero M and workpiece zero W

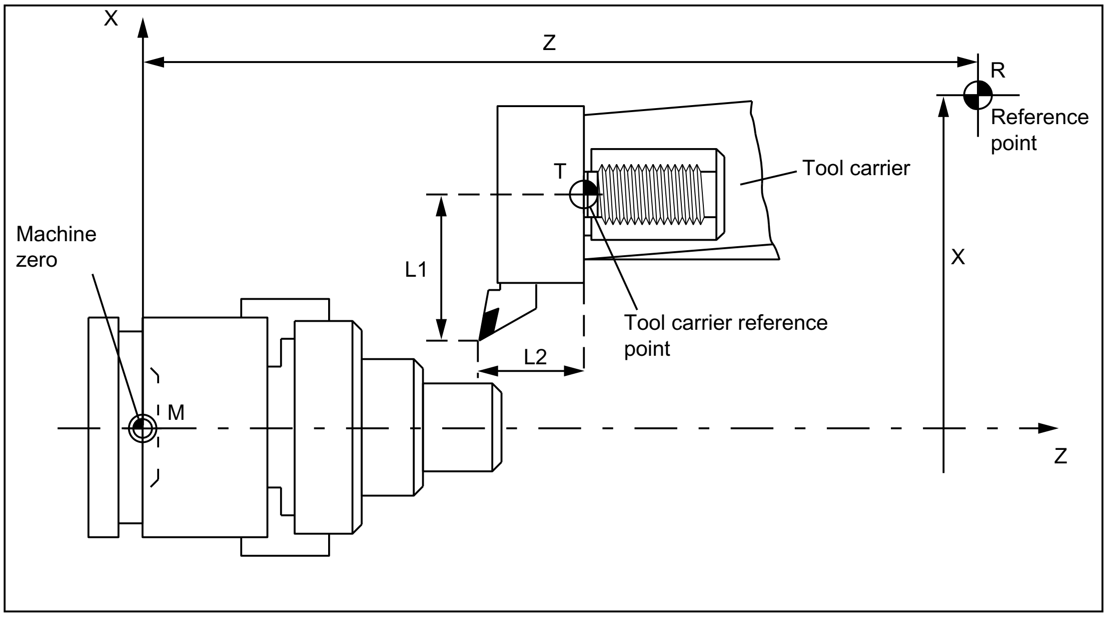


Figure 11-9 Position of reference point in relation to machine zero

11.4 Coordinate systems

11.4.1 Overview

Definitions

DIN 66217 stipulates that when programming machine tools, right-angled, rectangular (Cartesian) coordinate systems must be used. The positive directions of the coordinate axes can be determined using the "right-hand rule".

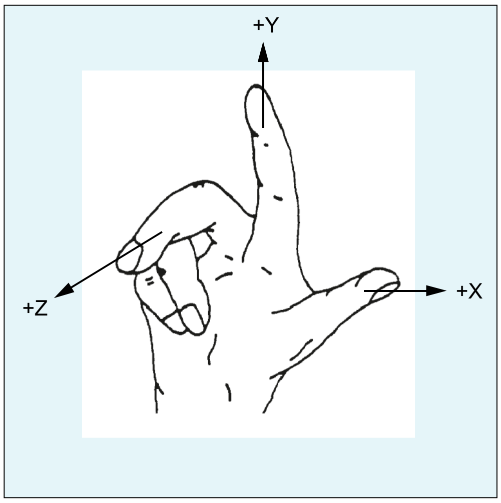


Figure 11-10 Right_hand_rule

The coordinate system used when programming is referred to the workpiece. Programming is realized independent of whether the tool or the workpiece is moved. When programming, it is always assumed that the tool traverses relative to the coordinate system of the workpiece, which is intended to be stationary.

Positive (clockwise) rotation of rotary axes can also be defined using the "right-hand rule". If the thumb of the right hand points towards the positive direction of a coordinate axis (linear axis), then the fingers point in the positive direction of rotation of the associated rotary axis.

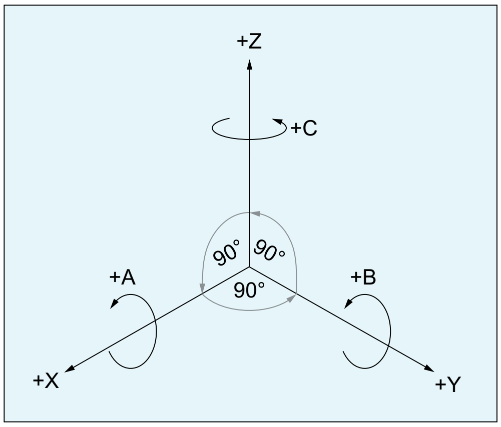


Figure 11-11 Clockwise, rectangular Cartesian coordinate system

Coordinate systems

The following coordinate systems are defined for machine tools:

Coordinate system	Abbreviation	Remark
Workpiece **C**oordinate **S**ystem	WCS	Programming the traversing motion of the geometry axes for machining the workpiece is written in the WCS.
Settable **Z**ero **S**ystem	SZS	Coordinate transformation using frames: WCS ⇒ SZS
Basic **Z**ero **S**ystem	BZS	Coordinate transformation using frames: SZS ⇒ BZS:
Basic **C**oordinate **S**ystem	BCS	Coordinate transformation using frames: BZS ⇒ BCS:
Machine **C**oordinate **S**ystem	MCS	Coordinate transformation using frames and kinematic transformation: BCS ⇒ MCS: The programmed traversing motion of the geometry axes in WCS is mapped to the machine axes of MCS using frames and kinematic transformations. A kinematic transformation is used to derive the BCS from the MCS. In this way, functions such as manual traverse, for example, are effective not in the MCS but in the BCS.

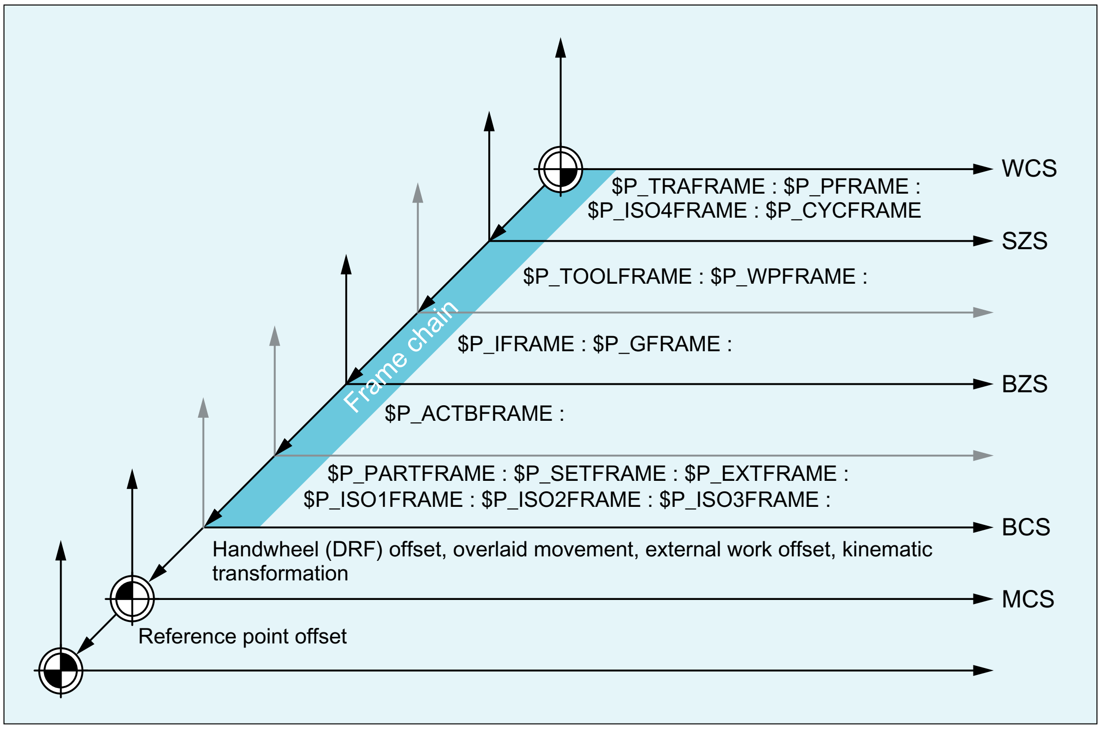

WCS
$P_TRAFRAME : $P_PFRAME :
$P_ISO4FRAME : $P_CYCFRAME
SZS
$P_TOOLFRAME : $P_WPFRAME :
Frame chain
$P_IFRAME : $P_GFRAME :
BZS
$P_ACTBFRAME :
$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
BCS
Handwheel (DRF) offset, overlaid movement, external work offset, kinematic transformation
MCS
Reference point offset

Figure 11-12 Coordinate systems, frames, and kinematic transformations

11.4.2 Machine coordinate system (MCS)

Machine coordinate system (MCS)

The machine coordinate system (MCS) is made up of all physically available machine axes.

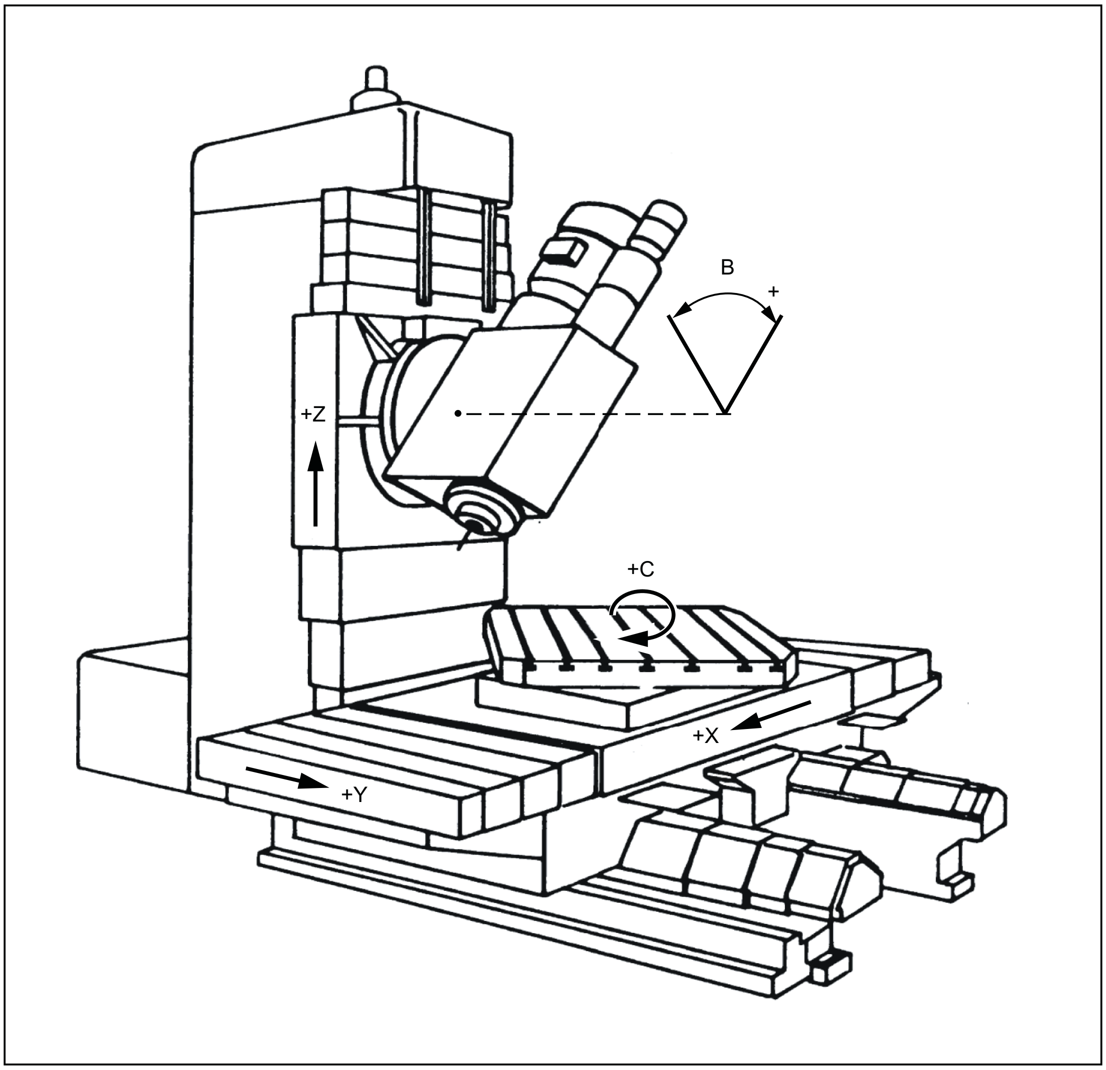


Figure 11-13 MCS with machine axes X, Y, Z, B, C (5-axis milling machine)

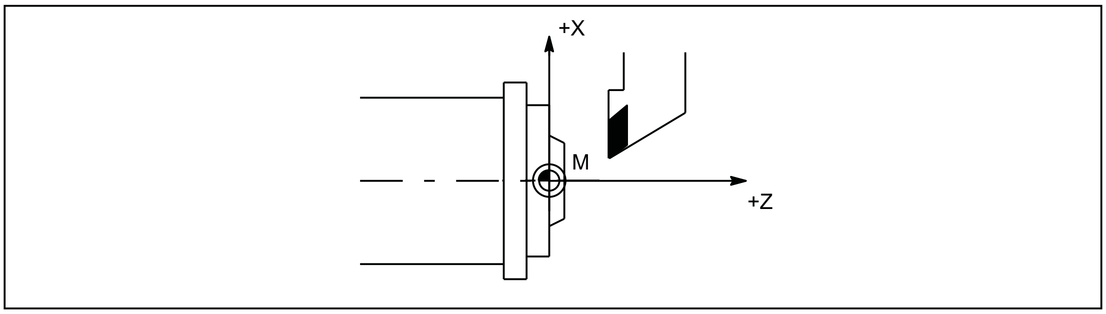


Figure 11-14 MCS with machine axes X, Z (turning machine)

Axial preset offset

The reference point of the control in the machine coordinate system (machine zero) can be reset via the "Preset offset (PRESETON)" function.

CAUTION

Loss of the encoder adjustment

After a preset offset, the appropriate machine axis is in the "Not referenced" state! This means that when using absolute encoders, the encoder adjustment is lost and must be performed again (e.g. by calibration with a laser interferometer). The use of PRESETON in combination with absolute encoders is therefore not recommended.

Note

We recommend that the function is **only** used for machine axes that do not require a reference point.

In order to restore the original machine coordinate system, the machine axis must be re-referenced, e.g. with G74 (reference point approach).

The machine axes are not moved with the preset offset.

References

- Programming Manual, Fundamentals
  Section: "Supplementary commands" > "Reference point approach (`G74`)"
- Programming Manual, Job Planning
  Section: "Coordinate transformations (FRAMES)" > "Preset offset (`PRESETON`)

11.4.2.1 Actual value setting with loss of the referencing status (PRESETON)

Function

The `PRESETON()` procedure sets a new actual value for one or more axes in the machine coordinate system (MCS). This corresponds to work offset of the axis MCS. The axis is not traversed.

A preprocessing stop with synchronization is triggered by `PRESETON`. The actual position is not assigned until the axis is at standstill.

If the axis is not assigned to the channel for `PRESETON`, the next steps depend on the replacement behavior parameterized in the following machine data:

MD30552 $MA_AUTO_GET_TYPE

Referencing status

By setting a new actual value in the machine coordinate system, the referencing status of the machine axis is reset.

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

It is recommended that `PRESETON` only be used for axes that do not require a reference point.

To restore the original machine coordinate system, the measuring system of the machine axis must be referenced again, e.g. through active referencing from the part program (`G74`).

⚠ CAUTION

Loss of the referencing status

By setting a new actual value in the machine coordinate system with `PRESETON`, the referencing status of the machine axis is reset to "not referenced/synchronized".

Programming

Syntax

```
PRESETON(<axis_1>, <value_1> [, <axis_2>, <value_2>, ... <axis_8>,
<value_8>])
```

Meaning

`PRESETON`:	Actual value setting with loss of the referencing status	
	Preprocessing stop:	Yes
	Alone in the block:	Yes
`<axis_x>`:	Machine axis name	
	Type:	AXIS
	Value range:	Machine axis names defined in the channel
`<value_x>`:	New actual value of the machine axis in the machine coordinate system (MCS)	
	The input is made in the current valid system of units (inch/metric)	
	An active diameter programming (`DIAMON`) is taken into account	
	Type:	REAL

System variable

$AC_PRESET

The axis-specific system variable $AC_PRESET provides the vector from the zero point of the currently offset MCS' to the zero point of the original MCS$_0$ after the referencing of the machine axis.

$AC_PRESET<axis> = $AC_PRESET<axis> + "current actual position of the axis in the MCS" - "`PRESETON` actual position"

The work offset can be undone with the system variables:

```
PRESETON(<axis>, $VA_IM + $AC_PRESET[<axis>]) ; "current actual
position of the axis in the MCS'" + "offsets"
```

Example

Program code

```
N10 G1 X10 F5000
N20 PRESETON(X, $AA_IM[X]+70) ; actual value = 10 + 70 = 80 =>
                              ; $AC_PRESET = $AC_PRESET - 70
```

Supplementary conditions

Axes for which `PRESETON` must not be used

- Traversing path axes
- Traversing positioning axes
- Traversing command axes in spindle mode
- Traversing concurrent positioning axes (FC18)
- Axes involved in a transformation
- Axes on which one or more of the following safety Safety Integrated functions are active:
  - Safe software limit switches (SE):
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bit 1 = 1
  - Safe software cams, safe cam track (SN):
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bits 8 ... 15 = 1
- Reciprocating axes
- Hirth axes
- Synchronized axes of a gantry grouping
- Axes for which the reference point approach from the part program (G74) is active
- Slave axis of a speed/torque coupling (master-slave)

Geometry axes

- `PRESETON` can be used on a stationary geometry axis when a further geometry axis is not being traversed in the channel at the same time.
- `PRESETON` can be used on a stationary geometry axis even when a further geometry axis is being traversed in the channel at the same time, but this axis is in the "neutral axis" state or traversing as a command axis.

Program code	Comment
`N10 G0 X0 Y0`	`; X, Y: Geometry axes`
`N15 RELEASE(Y)`	`; Neutral axis`
`N20 PRESETON(Y,20)`	`; Actual position Y in the MCS = 20`
`N30 G0 X40`	`; Geometry axis X traverses`
`N40 M30`	

Example: Another geometry axis (X) simultaneously traverses in the "**neutral axis**" state

Program code	Comment
`N10 G0 X0 Y0`	`; X, Y: Geometry axes`
`N20 POS[X]=40 FA[X]=1000`	`; X command axis`
`N30 DO PRESETON(Y,20)`	`; Actual position Y in the MCS = 20`
`N40 M30`	

Example: Geometry axis (X) traverses simultaneously as **command axis**

PLC-controlled axes

`PRESETON` can be used on PLC-controlled axes according to their current type.

Spindle states

The following table shows the reactions that occur when `PRESETON` is used on a spindle in a synchronized action:

PRESETON in the NC program			
Spindle mode	**Traversing status**	**Assigned to the NC program**	**Main axis**
Open-loop speed controlled mode	In motion	Alarm 22324	Alarm 22324
	Stationary	+	+
Positioning mode `SPOS`	In motion	-	+
	Stationary	+	+
Positioning across block boundaries `SPOSA`	In motion	Alarm 10610	-
Axis mode	In motion	-	+
	Stationary	+	+
+: Possible -: Not possible			

Axis couplings

- Leading axes: The sudden change of the leading axis position caused by `PRESETON` is not traversed in the following axes. The coupling is not changed.
- Following axes: Only the overlaid position component of the following axis is affected by `PRESETON`.

Gantry grouping

If `PRESETON` is used on the guide axis of a gantry grouping, the work offset is also performed in all synchronized axes of the gantry grouping.

Indexing axes

`PRESETON` can be used on indexing axes.

Software limit switches, operating range limit, protection areas

If the axis position is outside the specified limits after a work offset by `PRESETON`, an alarm is not displayed until an attempt is made to traverse the axis.

Block search with calculation

`PRESETON` commands are collected during the block search and executed with the NC start to continue the NC program.

Position-dependent NC/PLC interface signals

The status of the position-dependent NC/PLC interface signals is redetermined based on the new actual position.

Example: Fixed point positions

- Parameterized fixed point positions: MD30600 $MA_FIX_POINT_POS[0...3] = <fixed point position 1...4>
- NC/PLC interface signals DB31, ... DBX75.3 ... 5 (JOG approach fixed point: reached)

If the axis is at a fixed point position with the exact stop tolerance, the associated NC/PLC interface signal is set. The NC/PLC interface signal is reset when the actual value is set by `PRESETON` to a different value outside the exact stop tolerance around the fixed point position.

DRF offset

A DRF offset of the axis is deleted by `PRESETON`.

Overlaid movement $AA_OFF

An overlaid movement from a synchronized action with $AA_OFF is not affected by `PRESETON`.

Online tool offset FTOC

An active online tool offset from a synchronized action with FTOC remains active even after `PRESETON`.

Axis-specific compensations

Axis-specific compensations remain active after `PRESETON`.

JOG mode

`PRESETON` must only be used on a stationary axis.

JOG mode, REF machine function

`PRESETON` must **not** be used.

11.4.2.2 Actual value setting without loss of the referencing status (PRESETONS)

Function

The `PRESETONS()` procedure sets a new actual value for one or more axes in the machine coordinate system (MCS). This corresponds to work offset of the axis MCS. The axis is not traversed.

A preprocessing stop with synchronization is triggered by `PRESETONS`. The actual position is not assigned until the axis is at standstill.

If the axis for `PRESETONS` is not assigned to the channel, the next steps depend on the axis-specific configuring of the axis replacement behavior:

MD30552 $MA_AUTO_GET_TYPE

Referencing status

By setting a new actual value in the machine coordinate system (MCS) with `PRESETONS`, the referencing status of the machine axis is **not** changed.

Requirements

- **Encoder type**
  `PRESETONS` is only possible with the following encoder types of the active measuring system:
  - MD30240 $MA_ENC_TYPE[<measuring system>] = 0 (simulated encoder)
  - MD30240 $MA_ENC_TYPE[<measuring system>] = 1 (raw signal encoder)
- **Referencing mode**
  `PRESETONS` is only possible with the following referencing modes of the active measuring system:
  - MD34200 $MA_ENC_REFP_MODE[<measuring system>] = 0 (reference point approach not possible)
  - MD34200 $MA_ENC_REFP_MODE[<measuring system>] = 1 (referencing of incremental, rotary or linear measuring systems: zero pulse on the encoder track)

Commissioning

Axis-specific machine data

Actual value setting without loss of the referencing status (`PRESETONS`) must be set axis-specifically:

MD30455 $MA_MISC_FUNCTION_MASK, bit 9 = 1

Note

PRESETON deactivated

Activation of the "Actual value setting **without**loss of the referencing status `PRESETONS`" function deactivates the "Actual value setting **with** loss of the referencing status `PRESETON`" function. The options mutually exclude each other.

Programming

Syntax

```
PRESETONS(<axis_1>, <value_1> [, <axis_2>, <value_2>, ... <axis_8>,
<value_8>])
```

Meaning

`PRESETONS`:	Actual value setting with loss of the referencing status	
	Preprocessing stop:	Yes
	Alone in the block:	Yes
`<axis_x>`:	Machine axis name	
	Type:	AXIS
	Value range:	Machine axis names defined in the channel
`<value_x>`:	New current actual value of the machine axis in the machine coordinate system (MCS)	
	The value refers to the active system of units (inch / metric)	
	An active diameter programming (`DIAMON`) is taken into account	
	Type:	REAL

System variable

$AC_PRESET

The axis-specific system variable $AC_PRESET provides the vector from the zero point of the currently offset MCS' to the zero point of the original MCS_0 after the referencing of the machine axis.

$AC_PRESET<axis> = $AC_PRESET<axis> + "current actual position of the axis in the MCS" - "`PRESETONS` actual position"

The work offset can be undone with the system variables:

```
PRESETONS(<axis>, $VA_IM + $AC_PRESET[<axis>]) ; "current actual
position of the axis in the MCS'" + "offsets"
```

Example

Work offset of the X axis MCS by 70 units.

The programmed end position of the X axis (command axis) is transformed to the new MCS with `PRESETONS`.

Program code

```
N10 G1 X10 F5000
N20 PRESETONS(X, $AA_IM[X]+70) ; actual value = 10 + 70 = 80 =>
                               ; $AC_PRESET = $AC_PRESET - 70
```

Supplementary conditions

Axes for which `PRESETONS` must not be used

- Traversing positioning axes
- Traversing command axes in spindle mode
- Traversing concurrent positioning axes (FC18)
- Axes involved in a transformation
- Traversing path axes
- Reciprocating axes
- Axes on which one or more of the following safety Safety Integrated functions are active:
  - Safe software limit switches (SE):
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bit 1 = 1
  - Safe software cams, safe cam track (SN):
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bits 8 ... 15 = 1
- Hirth axes
- Synchronized axes of a gantry grouping
- Axes for which the reference point approach from the part program (G74) is active
- Slave axis of a speed/torque coupling (master-slave)

Geometry axes

- `PRESETONS` can be used on a stationary geometry axis when a further geometry axis is not being traversed in the channel at the same time.
- `PRESETONS` can be used on a stationary geometry axis even when a further geometry axis is being traversed in the channel at the same time, but this axis is in the "neutral axis" state or traversing as a command axis.
  Example: Another geometry axis (X) simultaneously traverses in the "**neutral axis**" state

Program code	Comment
`N10 G0 X0 Y0`	`; X, Y: Geometry axes`
`N15 RELEASE(Y)` 1)	`; Neutral axis`
`N20 PRESETONS(Y,20)`	`; Actual position Y in the MCS = 20`
`N30 G0 X40`	`; Geometry axis X traverses`
`N40 M30`	
1) **Note** The release of an axis in the action part of a synchronized action does not ensure that the release is on time. `N20 ID=1 WHEN 20.0 < $AA_IM[X] DO RELEASE(Y) PRESETONS(Y,20) ; NOT recommended`!	

Example: Another geometry axis (X) is traversing at the same time as **command axis**

Program code	Comment
`N10 G0 X0 Y0`	`; X, Y: Geometry axes`
`N20 POS[X]=40 FA[X]=1000`	`; X command axis`
`N30 PRESETONS(Y,20)`	`; Actual position Y in the MCS = 20`
`N40 M30`	

PLC-controlled axes

`PRESETONS` can be used on PLC-controlled axes according to their current type.

Spindle states

The following table shows the reactions that occur when `PRESETONS` is used on a spindle in a synchronized action:

PRESETONS in the NC program			
Spindle mode	**Traversing status**	**Assigned to the NC program**	**Main axis**
Speed control mode	In motion	Alarm 22324	Alarm 22324
	Stationary	+	+
Positioning mode `SPOS`	In motion	-	+
	Stationary	+	+
Positioning across block boundaries `SPOSA`	In motion	Alarm 10610	-

PRESETONS in the NC program			
Spindle mode	**Traversing status**	**Assigned to the NC program**	**Main axis**
Axis mode	In motion	-	+
	Stationary	+	+
+: Possible -: Not possible			

Axis couplings

- Leading axes: The sudden change of the leading axis position caused by `PRESETONS` is not traversed in the following axes. The coupling is not changed.
- Following axes: Only the overlaid position component of the following axis is affected by `PRESETONS`.

Gantry grouping

If `PRESETONS` is used on the guide axis of a gantry grouping, the work offset is also performed in all synchronized axes of the gantry grouping.

Indexing axes

`PRESETONS` can be used on indexing axes.

Software limit switches, operating range limit, protection areas

If the axis position is outside the specified limits after a work offset by `PRESETONS`, an alarm is not displayed until an attempt is made to traverse the axis.

Block search with calculation

`PRESETONS` commands are collected during the block search and executed with the NC start to continue the NC program.

Position-dependent NC/PLC interface signals

The status of the position-dependent NC/PLC interface signals is redetermined based on the new actual position.

Example: Fixed point positions

- Parameterized fixed point positions: MD30600 $MA_FIX_POINT_POS[0...3] = <fixed point position 1...4>
- NC/PLC interface signals DB31, ... DBX75.3 ... 5 (JOG approach fixed point: reached)

If the axis is at a fixed point position with the exact stop tolerance, the associated NC/PLC interface signal is set. The NC/PLC interface signal is reset when the actual value is set by `PRESETONS` to a different value outside the exact stop tolerance around the fixed point position.

DRF offset

A DRF offset of the axis is deleted by `PRESETONS`.

Overlaid movement $AA_OFF

An overlaid movement from a synchronized action with $AA_OFF is not affected by `PRESETONS`.

Online tool offset FTOC

An active online tool offset from a synchronized action with FTOC remains active even after `PRESETONS`.

Axis-specific compensations

Axis-specific compensations remain active after `PRESETONS`.

JOG mode

`PRESETONS` must only be used on a stationary axis.

JOG mode, REF machine function

`PRESETONS` must **not** be used.

Synchronized actions

In a synchronized action for an axis, if several `PRESETONS` instructions are included, then only the last instruction in the sequence from left to right is actually executed.

Example:

Program code

```
N10 ID=1 WHEN TRUE DO PRESETONS(X,40) PRESETONS(X,39) PRESETONS(X,38)
; This is equivalent to:
N10 ID=1 WHEN TRUE DO PRESETONS(X,38)
```

11.4.3 Basic coordinate system (BCS)

Basic coordinate system (BCS)

The basic coordinate system (BCS) consists of three mutually perpendicular axes (geometry axes) as well as other special axes, which are not interrelated geometrically.

Machine tools without kinematic transformation

BCS and MCS always coincide when the BCS can be mapped onto the MCS withouth kinematic transformation (e.g. `TRANSMIT` / face transformation, 5-axis transformation and up to three machine axes).

On such machines, machine axes and geometry axes can have the same names.

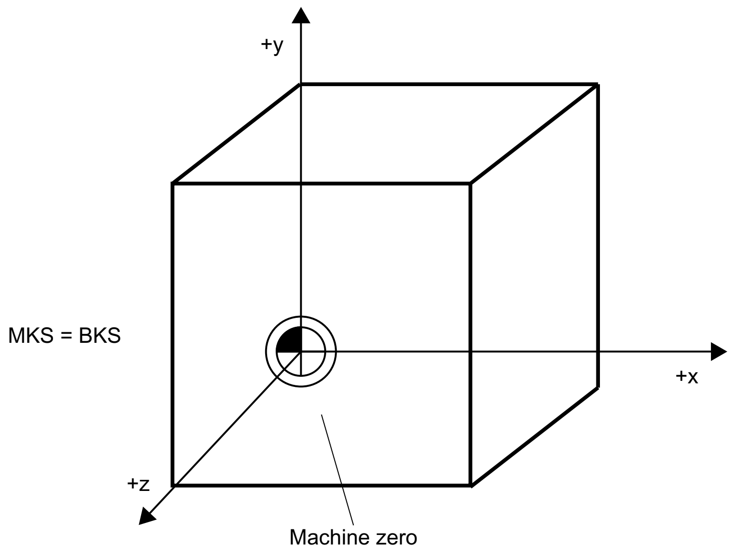


Figure 11-15 MCS=BCS without kinematic transformation

Machine tools with kinematic transformation

The BCS and MCS do not coincide when the BCS is mapped onto the MCS with kinematic transformation (e.g. `TRANSMIT` / face transformation, 5-axis transformation or more than three axes).

On such machines the machine axes and geometry axes must have different names.

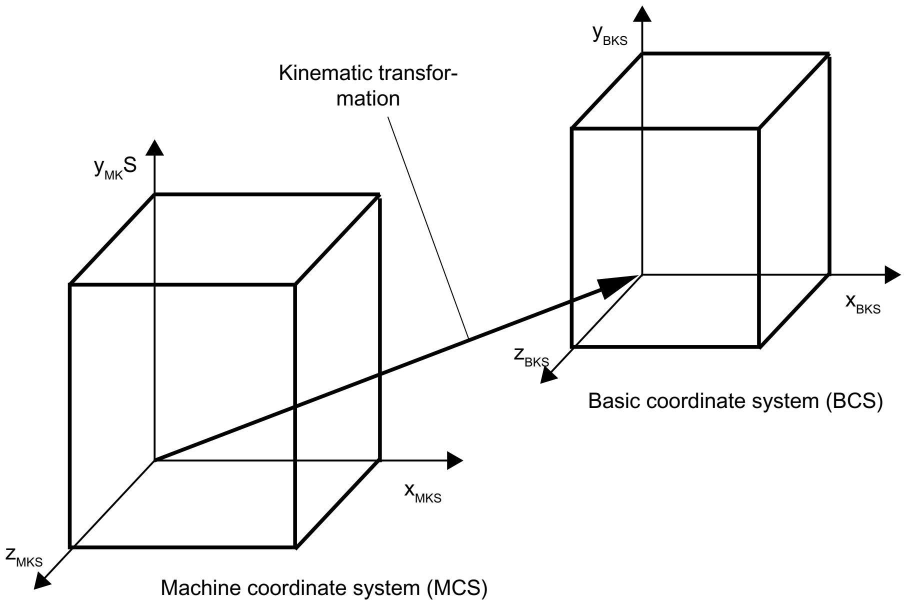


Figure 11-16 Kinematic transformation between the MCS and BCS

Machine kinematics

The workpiece is always programmed in a two- or three-dimensional, right-angled coordinate system (WCS). However, such workpieces are being programmed ever more frequently on machine tools with rotary axes or linear axes not perpendicular to one another. Kinematic transformation is used to represent coordinates programmed in the workpiece coordinate system (rectangular) in real machine movements.

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

Function Manual, Extended Functions; Kinematic Transformation (M1)

11.4.4 Basic zero system (BZS)

Basic zero system (BZS)

The basic zero system (BZS) is the basic coordinate system with a basic offset.

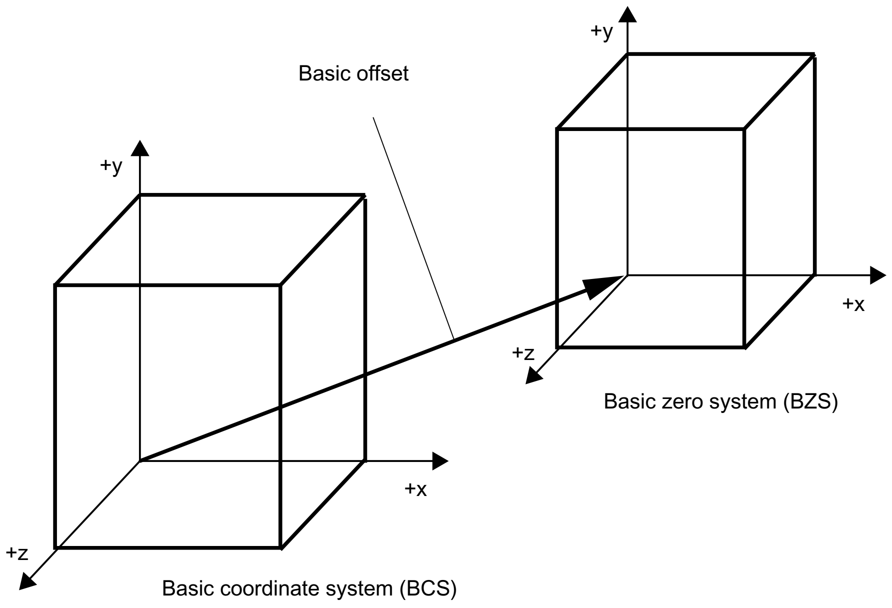


Figure 11-17 Basic offset between BCS and BZS

Basic offset

The basic offset describes the coordinate transformation between BCS and BZS. It can be used, for example, to define the palette zero.

The basic offset comprises:

- External work offset
- DRF offset
- Superimposed motion

- Chained system frames
- Chained basic frames

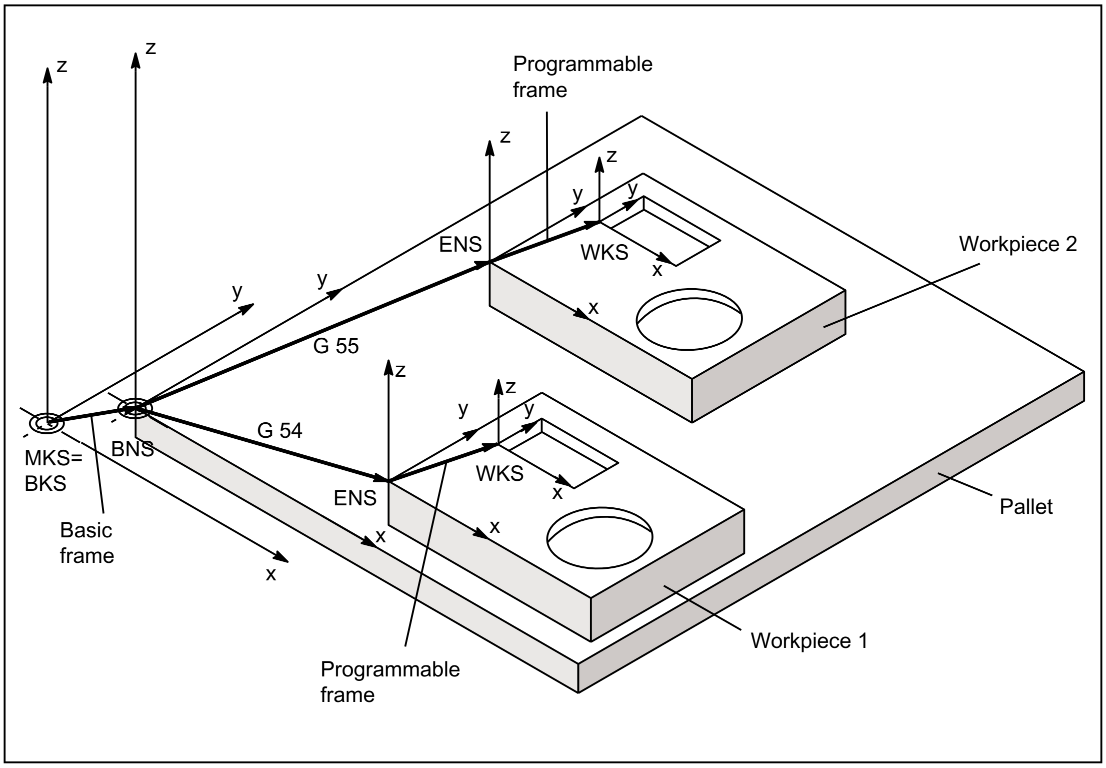


Figure 11-18 Example of the use of the basic offset

The following applies:

- 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 ASUP can be started via the PLC using `FC9` in order to execute the appropriate `G` command.

Note

Recommendation to the machine manufacturer

For your own applications, use the 3rd basic offset onwards.

The 1st and 2nd basic offset are reserved for PRESET and the "Zero offset external".

11.4.5 Settable zero system (SZS)

Settable zero system (SZS)

The "settable zero system" (SZS) is the workpiece coordinate system WCS with a programmable frame (viewed from the perspective of the WCS). The workpiece zero is defined by the settable FRAMES `G54` to `G599`.

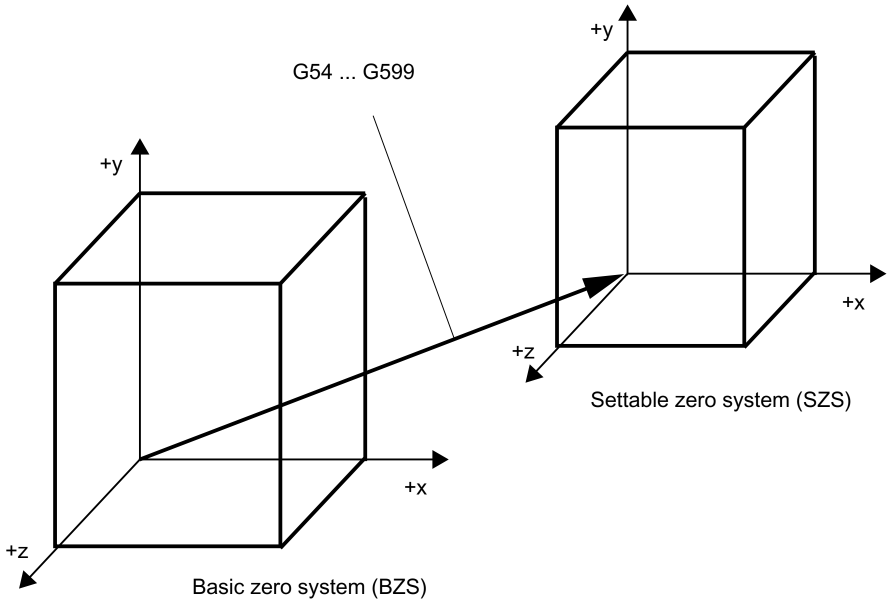


Figure 11-19 Settable FRAME `G54` ... `G599` between BZS and SZS

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 parameterization takes place through the machine data:

MD9424 $MM_MA_COORDINATE_SYSTEM (coordinate system for actual value display)

Value	Meaning
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

11.4.6 Workpiece coordinate system (WCS)

Workpiece coordinate system (WCS)

The workpiece coordinate system (WCS) is the programming basis.

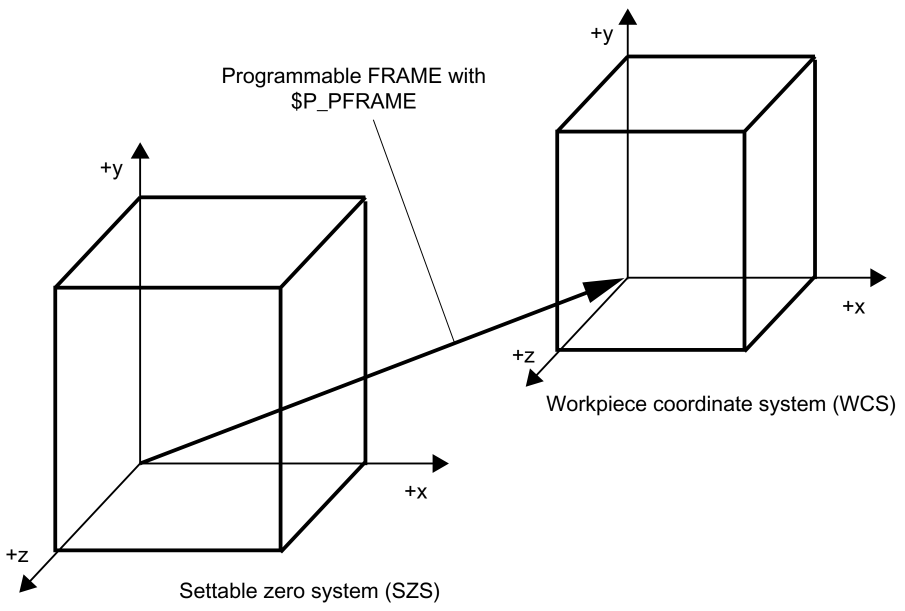
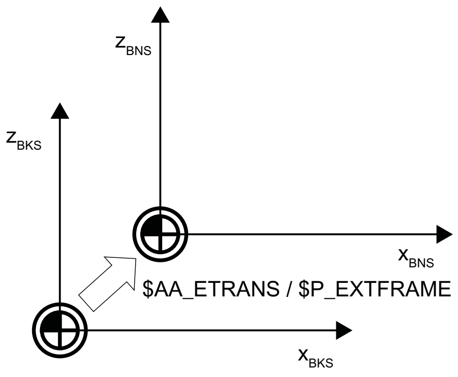

+y
Programmable FRAME with $P_PFRAME
+y
+x
+z
Workpiece coordinate system (WCS)
+x
+z
Settable zero system (SZS)

Figure 11-20 Programmable FRAME between SZS and WCS

11.4.7 Additive offsets

11.4.7.1 External work offsets

The external work offset is a linear offset between the basic coordinate system (BCS) and the basic zero system (BZS).



The external work offset $AA_ETRANS is effective in two ways depending on the machine data parameterization:

1. The $AA_ETRANS system variable has a direct effect as offset value after activation by the NC/PLC interface signal.
2. The value of the $AA_ETRANS system variable is taken into the system frame $P:EXTFRAME and the data management frame $P_EXTFR after activation by the NC/PLC interface signal. The active complete frame $P_ACTFRAME is then recalculated.

Machine data

In conjunction with the $AA_ETRANS system variable, a distinction is made between two procedures which are selected via the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit1 = <value>

<Value>	Meaning
0	Function: Direct writing of $AA_ETRANS[<axis>] by the PLC, HMI or NC program. Enable to traverse the work offset of $AA_ETRANS[<axis>] in the next possible traversing block: DB31, ... DBX3.0
1	Function: Activation of the active system frame $P:EXTFRAME and the data management frame $P_EXTFR Enable to traverse the work offset of $AA_ETRANS[<axis>] through: DB31, ... DBX3.0. The following is performed in the channel: • Stop of all traversing movements in the channel (except command and PLC axes) • Preprocessing stop with subsequent reorganization (STOPRE) • Coarse offset of the active frame $P_EXTFRAME[<axis>] = $AA_ETRANS[<axis>] • Coarse offset of the data management frame $P_EXTFR[<axis>] = $AA_ETRANS[<axis>] • Recalculation of the active complete frame $P_ACTFRAME • Traversing of the offset in the programmed axes. • Continuation of the interrupted traversing movement or the NC program

Programming

Syntax

```
$AA_ETRANS[<axis>] = <value>
```

Meaning

`$AA_ETRANS`:	System variable to buffer the external work offset
`<axis>`:	Channel axis
`<value>`:	Offset value

NC/PLC interface signal

Activation of the external work offset:

DB31, ... DBX3.0 = 0 → 1 ⇒ $P_EXTFRAME[<axis>] = $P_EXTFR[<axis>] = $AA_ETRANS[<axis>]

Suppression: External work offset

- The `SUPA` command suppresses the external work offset while the block is being processed.
- Active external work offsets are suppressed for the duration of the reference point approach through the `G74` command (reference point approach) and the equivalent operator actions in the reference point approach mode.
- With `G74`, i.e. "Automatic" or "MDI" mode, the previously active external work offset automatically becomes active again with the next traversing movement in the block.
- After a mode change from the reference point approach mode, the NC/PLC interface signal for the referenced axes must be set for reactivation.

11.4.7.2 DRF offset

The DRF offset enables the adjustment of an additive incremental work offset for geometry and additional axes in the basic coordinate system **through handwheel**.

System variable

The DRF offset can be read from the axis-specific system variable:

$AC_DRF[<Axis>]

References

Function Manual, Extended Functions; Manual traversing and manual handwheel traversing (H1); Section: DRF offset

11.4.7.3 Reset behavior

The reset and Power On behavior of the $P_EXTFRAME active frame and of the $P_EXTFR data management frame can be set with machine data:

Machine data

- The reset behavior for the system frame active of the $P_EXTFRAME external work offset in the channel is set with the following machine data:
- MD24006 $MC_CHSFRAME_RESET_MASK, bit 1 = <value>

Value	Meaning **The active system frame of the external work offset after channel/program end reset is:**
0	Not active
1	Active

- The Power On behavior of the channel-specific system frame of the $P_EXTFR external work offset of the data management is set with the following machine data:
  MD24008 $MC_CHSFRAME_POWERON_MASK, bit 1 = <value>

Value	Meaning **The $P_EXTFR system frame of the external work offset for Power On**
0	Not deleted
1	Deleted

11.4.8 Axis-specific overlay ($AA_OFF)

11.4.8.1 Function

Axis-specific system variable $AA_OFF[<axis>] can specify an absolute position or an incremental distance of the specified axis in a synchronous action, for example. The resulting traversing motion is then executed in parallel with the traversing motions of the channel of the axis.

$AA_OFF[<axis>] = <value>

11.4.8.2 Commissioning

Machine data

Parameterization of the overlaid motion

The following settings for the overlaid movement are made in the axis-specific machine data:

36750 $MA_AA_OFF_MODE, bit<n> = <value>

Bit	Value	Meaning
0	0	Interpretation of the value of $AA_OFF as the **absolute position**
	1	Interpretation of the value of $AA_OFF as the **incremental path**
1	0	The overlaid movement is deselected for channel reset.
	1	The overlaid movement maintained beyond channel reset.

Bit	Value	Meaning
2	0	In JOG mode, an overlaid movement is **not** retracted.
	1	In JOG mode, an overlaid movement is retracted.
3	0	A superimposed motion is interrupted for NC stop.
	1	A superimposed motion is **not** interrupted for NC stop.

System variable

Integrated path of the axis overlay

The integrated value of the overlaid movement can be read from the axis-specific system variable.

<value> = $AA_OFF_VAL[<axis>]

11.4.8.3 Programming: Deselecting overlays axis-specifically (CORROF)

The following axis-specific overlays are deleted with the CORROF procedure:

- Additive work offsets (DRF offsets) set via handwheel traversal
- Position offsets programmed via the $AA_OFF system variable
- Overlays of the tool orientation programmed via the $AC_OFF_... system variable

A preprocessing stop is initiated through the deletion of an overlay value and the position component of the deselected overlaid movement is transferred to the position in the basic coordinate system. Whereby, no axis is traversed.

The position value that can be read via the $AA_IM system variable (current **MCS** setpoint of the axis) **does not change** in the **machine** coordinate system.

The position value that can be read via the $AA_IW system variable (current **WCS** setpoint of the axis) **changes** in the **workpiece**coordinate system because it now contains the deselected component of the overlaid movement.

Note

`CORROF` can be programmed in an **NC program**.

`CORROF` must **not** be programmed in a **synchronized action**.

Syntax

```
CORROF(<Axis>,"<String>"[,<Axis>,"<String>"])
```

Meaning

`CORROF`:	Procedure for the deselection of the following offsets and overlays of an axis: • DRF offset • Position offsets ($AA_OFF) • Overlay of the tool orientation ($AC_OFF_...)	
	Effective-ness:	Modal
`<Axis>`:	Axis identifier (channel, geometry or machine axis identifier)	
	Data type:	AXIS
`<String>`:	Character string for the definition of the overlay type	
	Data type:	BOOL
	Value	**Meaning**
	`DRF`	DRF offset
	`AA_OFF`	Position offset ($AA_OFF)
	`OFF_ORI`	Overlay of the tool orientation ($AC_OFF_...) **Note** The deselection of the overlay of the tool orientation is performed by deleting the axis-specific offsets of the orientation axes. An **arbitrary** channel axis can be specified as <Axis> parameter.

Examples

Example 1: Axis-specific deselection of a DRF offset (1)

A DRF offset is generated in the X axis by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

```
Program code             Comment
N10 CORROF(X,"DRF")      ; CORROF has the same effect as DRFOF here.
...
```

Example 2: Axis-specific deselection of a DRF offset (2)

A DRF offset is generated in the X and Y axes by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

```
Program code             Comment
; Only the DRF offset of the X axis is deselected; the DRF offset of the Y axis is
retained.
; With DRFOF, both offsets would have been deselected.
N10 CORROF(X,"DRF")
...
```

Example 3: Axis-specific deselection of a $AA_OFF position offset

```
Program code                                 Comment
; A position offset == 10 is interpolated for the X axis.
```

```
Program code                                               Comment
N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5


...
; The position offset of the X axis is deselected: $AA_OFF[X]=0
; The X axis is not traversed.
; The position offset is added to the current position of the X axis.
N80 CORROF(X,"AA_OFF")
...
```

Example 4: Axis-specific deselection of a DRF offset and a $AA_OFF position offset (1)

A DRF offset is generated in the X axis by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

```
Program code                                               Comment
; A position offset of 10 is interpolated for the X axis.
N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5
...
; Only the DRF offset and the position offset of the X axis are deselected.
; The DRF offset of the Y axis is retained.
N70 CORROF(X,"DRF",X,"AA_OFF")
...
```

Example 5: Axis-specific deselection of a DRF offset and a $AA_OFF position offset (2)

A DRF offset is generated in the X and Y axes by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

```
Program code                                               Comment
; A position offset == 10 is interpolated for the X axis.
N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5
...
; The DRF offset of the Y axis and the position offset of the X axis are deselected.
; The DRF offset of the X axis is retained.
N70 CORROF(Y,"DRF",X,"AA_OFF")
...
```

Further information

$AA_OFF_VAL

Once the position offset has been deselected by means of $AA_OFF, system variable $AA_OFF_VAL (integrated distance of axis overlay) for the corresponding axis will equal zero.

$AA_OFF in JOG mode

Also in JOG mode, if $AA_OFF changes, the position offset will be interpolated as an overlaid movement if this function has been enabled via machine data MD 36750 $MA_AA_OFF_MODE.

$AA_OFF in synchronized action

If a synchronized action which immediately resets $AA_OFF (`DO $AA_OFF[<axis>]=<value>`) is active when the position offset is deselected using the `CORROF(<axis>,"AA_OFF")`, then $AA_OFF will be deselected and not reset, and alarm 21660 will be displayed. However, if the synchronized action becomes active later, e.g. in the block after `CORROF`, $AA_OFF will remain set and a position offset will be interpolated.

Automatic channel axis exchange

If an axis that is active in another channel has been programmed for a `CORROF`, it will be fetched into the channel with an axis exchange (requirement: MD30552 $MA_AUTO_GET_TYPE > 0) and the position offset and/or the DRF offset deselected.

11.5 Frames

11.5.1 Frame types

A frame is a data structure that contains values for offset (`TRANS`), fine offset (`FINE`), rotation (`ROT`), mirroring (`MIRROR`) and scaling (`SCALE`) for axes.

When activating the frame, using the frame values, a static coordinate transformation for the axes contained in the frame is performed using a defined algorithm.

Axis-specific frame

An axis-specific frame contains the frame values of an axis.

Example data structure of an axis-specific frame for geometry axis X:

Axis	TRANS	FINE	ROT	MIRROR	SCALE
X	10.0	0.1	0.0	0	1

Channel-specific frame

A channel-specific frame contains frame values for all channel axes (geometry, special and machine axes).

Rotations (ROT) are only included in the calculation for geometry axes.

A channel-specific frame is only active in the channel in which the frame is defined.

Example of the data structure of a channel-specific frame:

- Geometry axes: X, Y, Z
- Special axes: A
- Machine axes: AX1

Axis	TRANS	FINE	ROT	MIRROR	SCALE
X	10.0	0.1	**0.0**	0	1
Y	0.0	0.0	**0.0**	1	1
Z	0.0	0.0	**45.0**	0	1
A	2.0	0.1	0.0	0	2
AX1	0.0	0.0	0.0	0	0

Global frame

A global frame contains the frame values for all machine axes.

A global frame is active in all channels of the NC.

Example data structure of global frame:

- Machine axes: AX1, ... AX5

Axis	TRANS	FINE	ROT	MIRROR	SCALE
AX1	10.0	0.1	-	0	1
AX2	0.0	0.0	-	1	1
AX3	0.0	0.0	-	0	1
AX4	2.0	0.1	-	0	2
AX5	0.0	0.0	-	1	1

11.5.2 Frame components

11.5.2.1 Translation

Programming

The programming of the translation or coarse offset can be performed via the following commands:

- Example of data management frames `$P_UIFR`
  - Complete frame: `$P_UIFR[<n>] = CTRANS(<K1>,<V1>[,<K2>,<V2>][,<K3>,<V3>])`
    with Km = coordinate x, y or z and Vm = offset m
  - Frame component: `$P_UIFR[<n>,<k>,TR] = <V>`
    with K = coordinate x, y or z and V = offset
- Example of programmable frame
  - TRANS `<K1> <V1> [<K2> <V2>][<K3> <V3>]`
    with Km = coordinate x, y or z and Vm = offset m

Programs examples:

Program code	Remark
`$P_UIFR[1] = CTRANS(X,10,Y,10)`	`Complete frame`
`$P_UIFR[1,X,TR] = 10`	`Frame components`
`TRANS X=10 Y=10`	`Programmable frame`

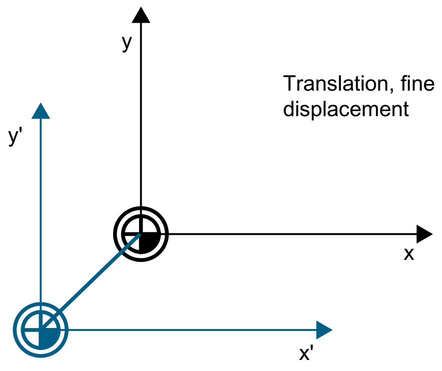


Figure 11-21 Offset in the Z direction

11.5.2.2 Fine offset

Parameterization

The fine offset is enabled via the machine data:

MD18600 $MN_MM_FRAME_FINE_TRANS = <value>

Value	Meaning
0	The fine offset cannot be entered or programmed.
1	Fine offset is possible for settable frames, basic frames and the programmable frame via command or program.

Programming

The programming of the translation or coarse offset can be performed via the following commands:

- Example of data management frames `$P_UIFR`
  - Complete frame: `$P_UIFR[<n>] = CFINE(<K1>,<V1>[,<K2>,<V2>][,<K3>,<V3>])`
    with Km = coordinate x, y or z and Vm = offset m
  - Frame component: `$P_UIFR[<n>,<K>,FI] = <V>`
    with K = coordinate x, y or z and V = offset
- Example of programmable frame
  - TRANS `<K1> <V1> [<K2> <V2>][<K3> <V3>]`
    with Km = coordinate x, y or z and Vm = offset m

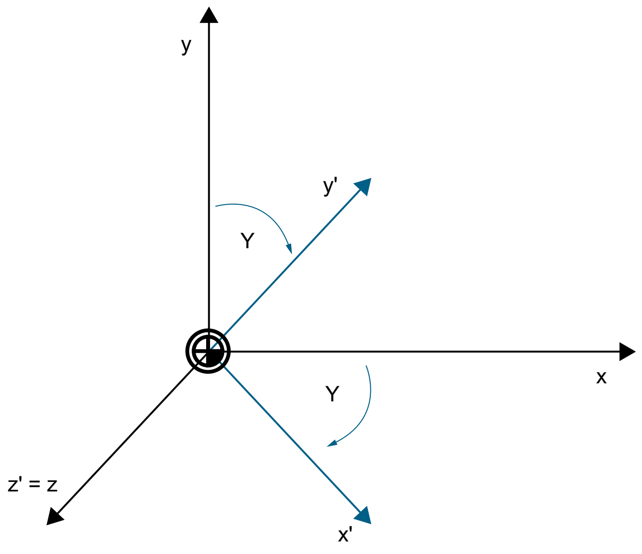
Programming examples:

Program code	Remark
`$P_UIFR[1] = CTRANS(X,10,Y,10)`	Complete frame
`$P_UIFR[1,X,TR] = 10`	Frame components
`TRANS X=10 Y=10`	Programmable frame

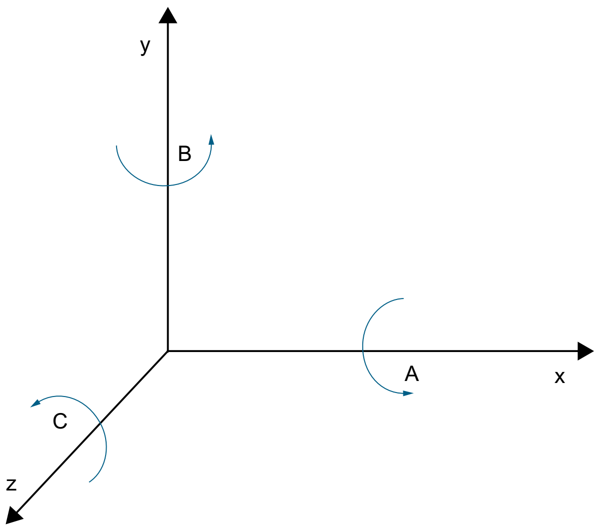
11.5.2.3 Rotation Overview (geometry axes only)

Function

The direction of rotation around the coordinate axes is determined by means of a right-hand, rectangular coordinate system with axes x, y and z. If the rotary motion is in a clockwise direction when looking in the positive direction of the coordinate axis, the direction of rotation is positive. A, B and C designate rotations whose axes are parallel to the coordinate axes.



The following figure shows the new position of the coordinate system x', y' and z' after the rotation around z with γ = -45°



Parameterization of the rotation sequence

The following machine data is used to set around which coordinate axes and in which order the rotations are performed when more than one angle of rotation is programmed:

MD10600 $MN_FRAME_ANGLE_INPUT_MODE = <value>

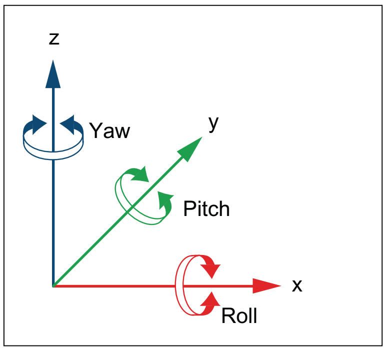
Value	Meaning
1	Euler angles in zy'x" convention (RPY angles)
2	Euler angles in zx'z" convention

Note

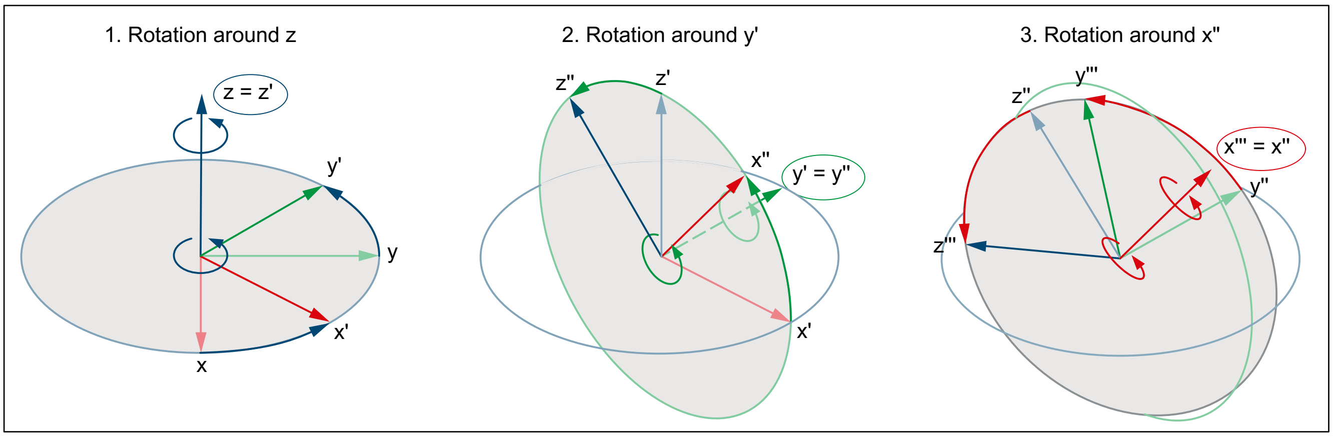
For historical reasons, Euler angles in zx'z" convention can be used. However, it is strongly recommended that only Euler angles in zy'x" convention (RPY angles) be used (see Section Rotation with a Euler angles: ZY'X" convention (RPY angles) (Page 753)).

11.5.2.4 Rotation with a Euler angles: ZY'X" convention (RPY angles)

Euler angles in the ZY'X" convention are also called RPY angles. RPY is derived from rolling, pitching and yawing:



For the ZY'X" convention, the rotations are carried out in the following sequence:



Value range

With RPY angles, programmed values can only be unambiguously calculated back within the following value ranges:

-180	≤	**x**	≤	180
-90	<	**y**	<	90
-180	≤	**z**	≤	180

Programming: Writing all rotation components

When programming the rotating components of a frame using`CROT`, `ROT` or `AROT`, all rotating components are always written to. Rotating components not explicitly programmed are assigned a value of 0°.

Syntax

```
<frame> = CROT([<1st GAx>,<angle>,][<2nd GAx>,<angle>,][<3rd
GAx>,<angle>])
ROT [<1st GAx> <Angle>] [<2. GAx> <Angle>] [<3. GAx> <Angle>]
AROT [<1st GAx> <Angle>] [<2. GAx> <Angle>] [<3. GAx> <Angle>]
```

Meaning

`CROT`:	Absolute rotation
`<Frame>`:	Arbitrary active or data management frame
`ROT`:	Absolute rotation Reference frame: Programmable frame `$P_PFRAME`, reference point: Zero point of the current workpiece coordinate system set with `G54` ... `G57`, `G505` ... `G599`
`AROT`:	Additive rotation Reference frame: Programmable frame `$P_PFRAME`, reference point: Zero point of the current workpiece coordinate system set with `G54` ... `G57`, `G505` ... `G599`

`<nth GAx>`:	Name of the nth geometry axis around which rotation is to be performed with the specified angle. The value 0° is implicitly set as angle of rotation for a geometry axis that has not been programmed. Assignment of geometry axis to rotary axis:	
	Geometry axis	**Rotary axis**
	1st geometry axis	x''
	2nd geometry axis	y'
	3rd geometry axis	z
`<angle>`:	Angle specification in degrees.	
`[...]`:	The data in square brackets are optional.	

Programming: Writing to a rotation component

When explicitly programming a rotation component of a frame, only the programmed rotation component is written. Rotation components that have not been programmed remain unchanged.

Syntax

```
<frame>[<index>,<GAx>,RT] = <angle>
```

Meaning

`<Frame>`:	Arbitrary active frame or data management frame
`<Index>`:	Array index of the frame, e.g. `$P_UIFR[0 ... n]`
`<GAx>`:	Name of the geometry axis around which rotation is to be performed with the specified angle.
`RT`:	Keyword for rotation "RoTation"
`<Angle>`	Angle specification in degrees.

Reading back the rotation components

In general, the same values are obtained when reading back the rotation components of a frame as those that were programmed:

Programmed	Values when reading back		
	x, RT	y, RT	z, RT
`<Frame> = CROT(X,45,Y,30,Z,-20)`	45	30	-20

Values outside the value range

Programmed values outside a value range are mapped on the range limits:

Programmed	Values when reading back		
	x, RT	y, RT	z, RT
`<Frame> = CROT(X,190,Y,0,Z,-200)`	-170	0	160

Note

It is recommended that when writing the rotation components of the frame, the specified value ranges are observed so that the same values are obtained when reading back the rotation components.

Gimbal lock

Gimbal lock designates a geometric problem in which the rotation components can no longer be unambiguously calculated back from the position vector. Gimbal lock occurs in RPY angles with an angular position of the rotation component y = 90°. In this case, the rotation components are converted by the control system after being written so that the following applies:

- Rotation component z = rotation component z - rotation component x
- Rotation component x = 0°
- Rotation component y = 90°

Programmed	Values when reading back		
	x, RT	y, RT	z, RT
`<Frame> = CROT(X,30,Y,90,Z,40)`	0	90	40 - 30 = 10

CAUTION

Different values for reading back the rotation component z

Because of the different conversion times after writing the **complete frame** or the writing of **individual rotation components** of a **data management frame** and the writing of **individual rotation components** of an **active frame**, different values can be read back for rotation component z.

Differences when writing the complete frame and frame components

Two cases must be distinguished when writing the rotation components of a frame:

1. Writing the complete frame: `<Frame> = CROT(X,a,Y,b,Z,c)`
   When writing the complete frame, the conversion is immediately at the time of writing.
2. Writing individual rotation components, e.g. rotation around X: `<Frame>[0,X,RT]= <a>`
   When writing individual rotation components, the conversion depends on the storage location of the frame:
   – Data management frames
     With data management frames, the conversion is at the time of activation of the frame based on the rotation components written by this time. With regard to the conversion of a data management frame, a data management frame therefore behaves in the same way after writing individual rotation components as when writing the complete frame.
   – Active frames
     In the case of active frames, the conversion is immediately at the time of writing of the rotation component.

Example

- Writing the complete frame
  The **conversion** is made **in each block** after the **complete frame** has been written.

Programmed	Values when reading back		
	x, RT	y, RT	z, RT
`N10 <Frame> = CROT(X,0,Y,90,Z,90)`	0	90	90
`N20 <Frame> = CROT(X,90,Y,90)`	0	90	**-90** 1)
`N30 <Frame> = CROT(X,90,Y,90,Z,90)`	0	90	**0** 1)
1) Different values compared to the writing of **individual rotation components** of an **active frame**			

- Writing individual rotation components of a **data management frame**
  The **conversion** is performed **on the activation of the data management frame.** In the example, at any time after N30.

Programmed	Values when reading back		
	x, RT	y, RT	z, RT

`N10 <Data management frame>[0,X,RT] = 0`
`N20 <Data management frame>[0,Y,RT] = 90`
`N30 <Data management frame>[0,Z,RT] = 90`
	0	90	90

`N10 <Data management frame>[0,X,RT] = 90`
`N20 <Data management frame>[0,Y,RT] = 90`
`N30 <Data management frame>[0,Z,RT] = 0`
	0	90	**-90** 1)

`N10 <Data management frame>[0,X,RT] = 90`
`N20 <Data management frame>[0,Y,RT] = 90`
`N30 <Data management frame>[0,Z,RT] = 90`
	0	90	**0** 1)

1) Different values compared to the writing of **individual rotation components** of an **active frame**			

- Writing individual rotation components of an **active frame**
  Any required **conversion** is performed **immediately on writing the rotation component.**
  The stored initial values of the active frame are: x = 0, y = 0, z = 0.

Programmed	Values when reading back		
	x, RT	y, RT	z, RT

`N10 <Active frame>[0,X,RT] = 0`
`N20 <Active frame>[0,Y,RT] = 90`
`N30 <Active frame>[0,Y,RT] = 90`
	0	90	90

`N10 <Active frame>[0,X,RT] = 90`
`N20 <Active frame>[0,Y,RT] = 90`
`N30 <Active frame>[0,Y,RT] = 0`
	0	90	**0** 1)

`N10 <Active frame>[0,X,RT] = 90`
`N20 <Active frame>[0,Y,RT] = 90`
`N30 <Active frame>[0,Y,RT] = 90`
	0	90	**90** 1)

1) Different values compared to the writing of the **complete frame** or the writing of **individual rotation components** of a **data management frame**			

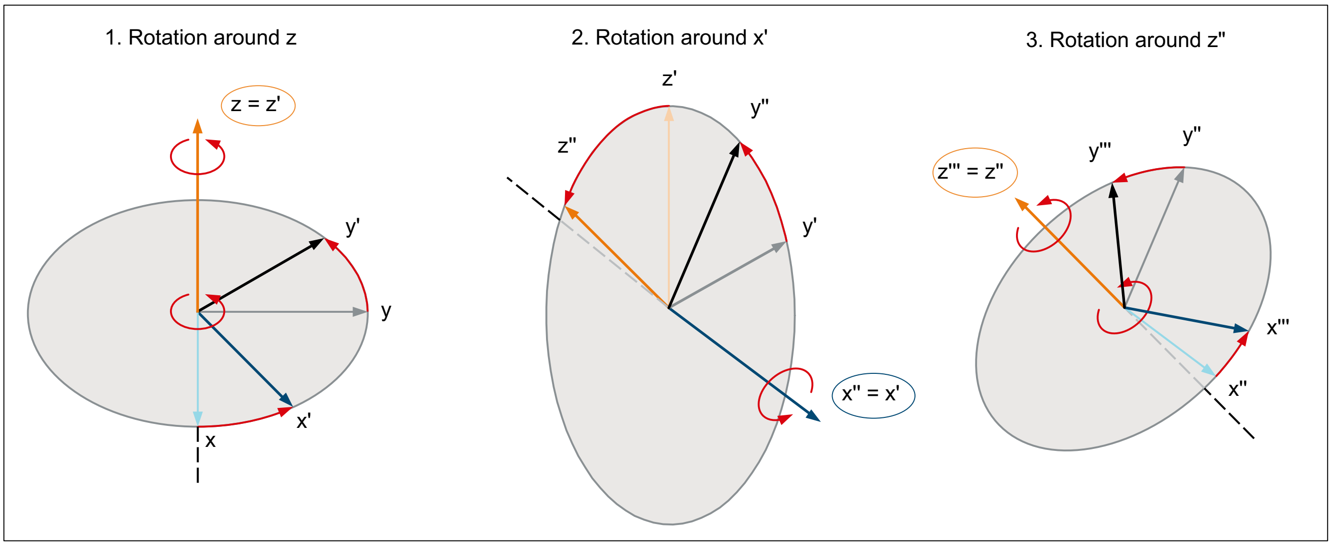
11.5.2.5 Rotation with a Euler angles: ZX'Z" convention

With Euler angles, the rotations are in the order z, x', z".

Note

Recommended use

For historical reasons, Euler angles in zx'z" convention can be used. However, it is strongly recommended that only Euler angles in zy'x" convention (RPY angles) be used (see Section Rotation with a Euler angles: ZY'X" convention (RPY angles) (Page 753)).



Assignment of rotary axis to geometry axis

Rotary axis	Geometry axis in channel
z	3rd geometry axis
x'	1st geometry axis
z"	3rd geometry axis

Value range

Data from Euler angles can only be unambiguously calculated back within the following value ranges:

0	<=	**x**	<	180
-180	<=	**y**	<=	180
-180	<=	**z**	<=	180

For data outside the specified value ranges, a modulo conversion is made referred to the value of the particular range limit.

Note

It is recommended that when writing the rotation components of the frame, the specified value ranges are observed so that the same values are obtained when reading back the rotation components.

11.5.2.6 Rotation in any plane

CRPL - Constant Rotation Plane

The predefined function "**C**onstant **R**otation **Pl**ane" enables a rotation to be programmed for a frame in an arbitrary plane (`G17`, `G18`, `G19`) without specifying the name of a geometry axis. This enables rotations to be programmed in the third plane when only two geometry axes are present in the channel due to the specific machine constellation.

Syntax

```
CRPL(<rotary axis>,<angle of rotation>)
```

Meaning

`CRPL`:	Rotation in any plane
`<rotary axis>`:	Axis around which the rotation is performed Type: INT

Value	Meaning
0	Rotation in the active plane
1	Rotation around Z
2	Rotation around Y
3	Rotation around X

`<angle of rotation>`: Angle in degrees through which the rotation is performed

Type: REAL

It is strongly recommended to observe the specified angular ranges. If the limits are not observed, then an unambiguous reverse calculation is not possible. Angles outside the limits are not rejected.

RPY angle:	X	-180 <= `<angle of rotation>` <= 180
	Y	-90 <= `<angle of rotation>` <= 90
	Z	-180 <= `<angle of rotation>` <= 180
ZX'Z" convention:	X	-180 <= `<angle of rotation>` <= 180
	Y	0 <= `<angle of rotation>` <= 180
	Z	-180 <= `<angle of rotation>` <= 180

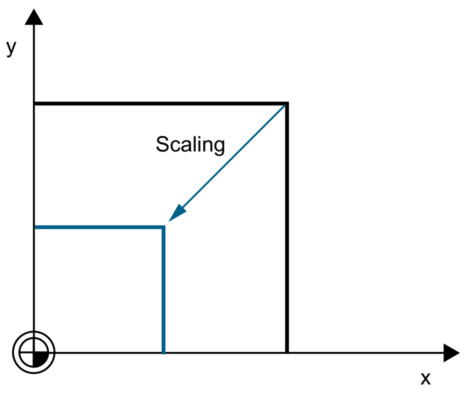
Chaining with frames

`CRPL()` can be chained with frames and known frame functions such as `CTRANS()`, `CROT()`, `CMIRROR()`, `CSCALE()`, `CFINE()` etc.

Examples:

```
$P_PFRAME = $P_PFRAME : CRPL(0,30.0)
$P_PFRAME = CTRANS(X,10) : CRPL(1,30.0)
$P_PFRAME = CROT(X,10) : CRPL(2,30.0)
$P_PFRAME = CRPL(3,30.0) : CMIRROR(Y)
```

11.5.2.7 Scaling

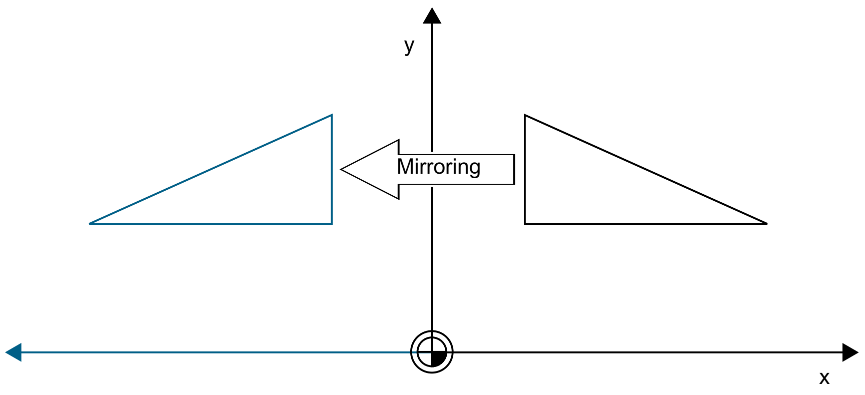


Programming

The program commands below are used to program the scaling:

```
$P_UIFR[1] = CSCALE(x,1,y,1)
SCALE x = 1y = 1
$P_UIFR[1,x,sc] = 1
```

11.5.2.8 Mirroring



Programming

The program commands below are used to program a mirroring:

`$P_UIFR[1] = CMIRROR(x,1,y,1)`

`MIRROR x = 1y = 1`

`$P_UIFR[1,x,mi] = 1`

11.5.2.9 Chain operator

Frame components or complete frames can be combined into a complete frame using the chain operator (:).

11.5.2.10 Programmable axis name

Geometry, channel and machine axis names can be used in the frame commands. The programmed axis must be known to the channel-specific frames in the channel.

SPI

When programming frame commands, the `SPI(<spindle number>)` axis function can be used in place of an axis name.

`SPI(<spindle number>)` forms the reference of the spindle to the channel axis.
→ refer to MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[] (assignment of spindle to machine axis)

The following frame commands 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 name or machine axis name of the axis belonging to the spindle is always output when decompiling frames, even when axis names have been programmed in the part program with `SPI(..)`.

If the spindle is assigned e.g., to the Channel Axis A then the programming:

```
N10 $P_UIFR[1] = CTRANS(SPI(1),33.33,X,1):CSCALE(SPI(1),
33.33):CMIRROR(SPI(1))
```

during recompilation:

```
$P_UIFR[1]=CTRANS(X,1,A,33.33):CSCALE(A,33.33):CMIRROR(A)
```

If a spindle and an assigned axis are programmed in a frame command, then Alarm 16420 "Axis % multiply programmed" is output.

Example:

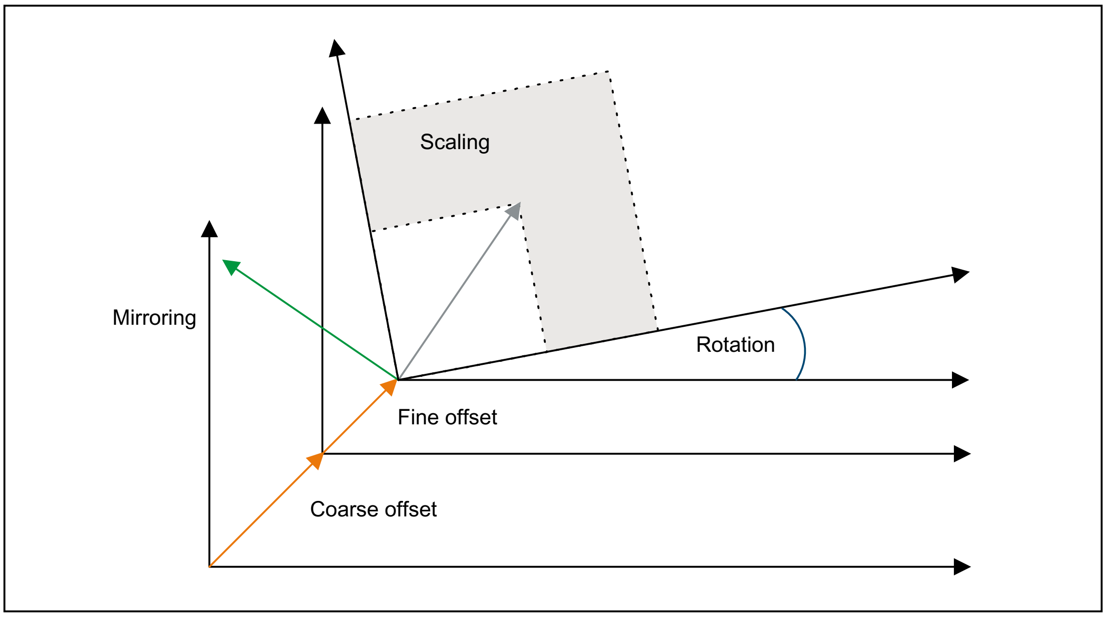
```
$P_UIFR[1] = CTRANS(SPI(1),33.33,X,1,A,44)
```

(The spindle is assigned to Axis A.)

Programming examples

```
$P_PFRAME[SPI(1),TR]=22.22

$P_PFRAME=CTRANS(X, axis value,Y,axis value,SPI(1),axis value)

$P_PFRAME=CSCALE(X,Scale,Y,scale,SPI(2),scale)

$P_PFRAME=CMIRROR(S1,Y,Z)

$P_UBFR=CTRANS(A,10):CFINE(SPI(1),0.1)
```

11.5.2.11 Coordinate transformation



The formulae below are used to discover the coordinate transformation for geometry axes:

WCS → BCS	$\vec{v} = \underline{R} * \underline{S} * \underline{M} * \vec{v}' + \vec{t}$
BCS → WCS	$\vec{v}' = inv(\underline{M}) * inv(\underline{S}) * inv(\underline{R}) * (\vec{v} - \vec{t})$

V:	Position vector in BCS
V':	Position vector in WCS

11.5.3 Data management frames and active frames

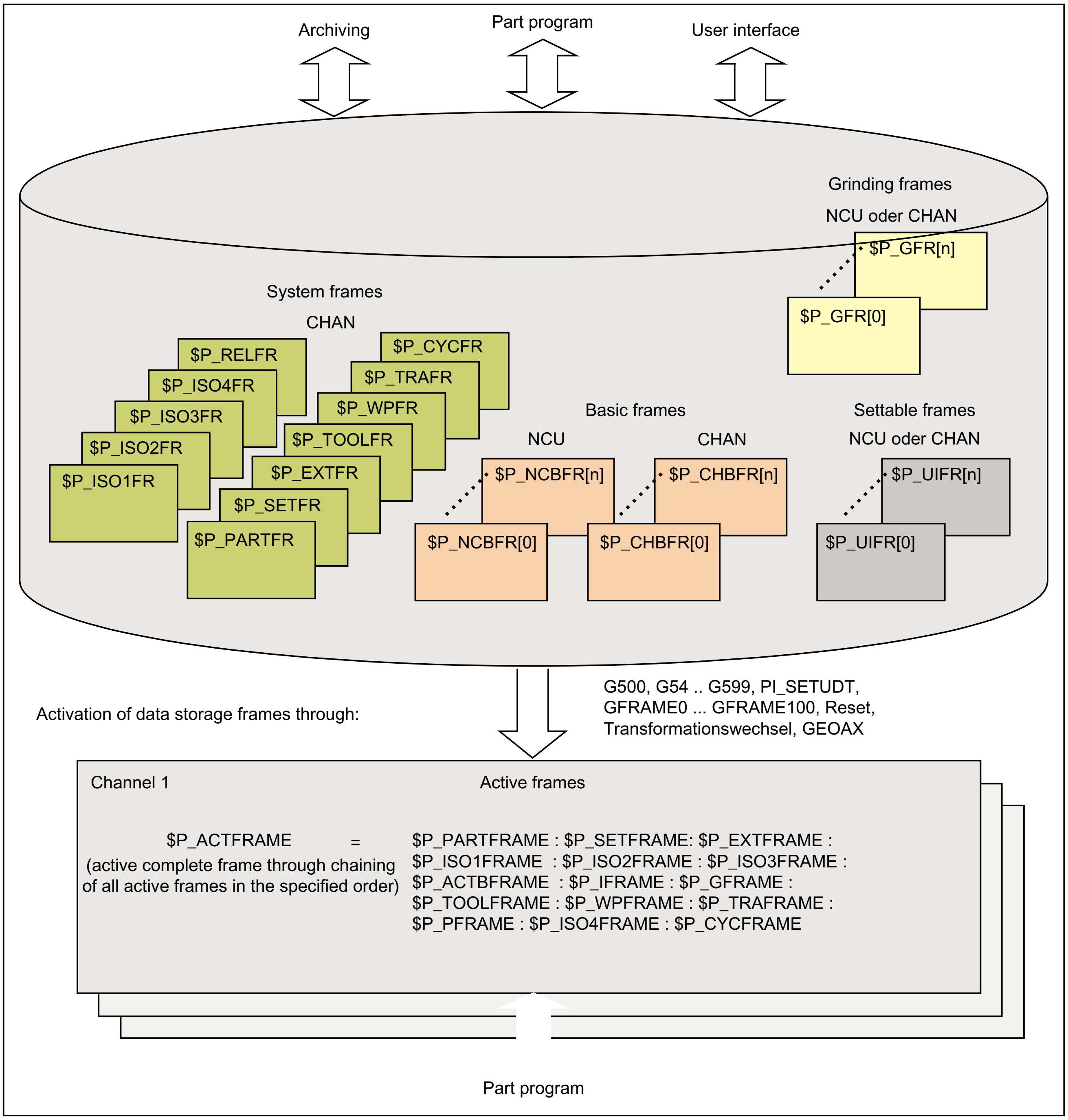
11.5.3.1 Overview

Frame types

The following frame types are available:

- System frames (`$P_PARTFR`, ... see figure)
- Basic frames (`$P_NCBFR[<n>]`, `$P_CHBFR[<n>]`)
- Grinding frames (`$P_GFR[<n>]`)
- Settable frames (`$P_UIFR[<n>]`)
- Programmable frame (`$P_PFRAME`)

For all frame types except the programmable frame, one or more frames exist in the data management (data management frames) in addition to the frame active in the channel. For the programmable frame, only the frame active in the channel exists.


Archiving
Part program
User interface
Grinding frames
NCU oder CHAN
$P_GFR[n]
$P_GFR[0]
System frames
CHAN
$P_RELFR
$P_ISO4FR
$P_ISO3FR
$P_ISO2FR
$P_ISO1FR
$P_CYCFR
$P_TRAFR
$P_WPFR
$P_TOOLFR
$P_EXTFR
$P_SETFR
$P_PARTFR
Basic frames
NCU
CHAN
$P_NCBFR[n]
$P_NCBFR[0]
$P_CHBFR[n]
$P_CHBFR[0]
Settable frames
NCU oder CHAN
$P_UIFR[n]
$P_UIFR[0]
Activation of data storage frames through:
G500, G54 .. G599, PI_SETUDT, GFRAME0 ... GFRAME100, Reset, Transformationswechsel, GEOAX
Channel 1
Active frames
$P_ACTFRAME =
(active complete frame through chaining of all active frames in the specified order)
$P_PARTFRAME : $P_SETFRAME: $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
$P_ACTBFRAME : $P_IFRAME : $P_GFRAME :
$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME :
$P_PFRAME : $P_ISO4FRAME : $P_CYCFRAME
Part program

Writing frames

Data management frames and active frames can be written from the part program. Only data management frames can be written via the user interface.

Archiving frames

Only data management frames can be archived.

11.5.3.2 Activating data management frames

Data management frames become active frames as a result of the following actions:

- "Settable frames" G group: `G54` ... `G57, G500, G505` ... `G599`
- "Grinding frames" G group: `GFRAME0` ... `GFRAME100`
- RESET and MD20110 $MC_RESET_MODE_MASK, bit14 == 1 (the current setting of the basic frame is retained)
- Transformation changeover
- Changing the geometry axis assignment `GEOAX`
- From the HMI with PI service "_N_SETUDT"

Activation from the HMI

The activation of a data management frame from the HMI with PI service "_N_SETUDT" only becomes active in the channel after a hot restart for the current part program.

The activation is effective in the reset state if the following machine data is set:

MD9440 $MM_ACTIVATE_SEL_USER_DATA (set active offset immediately)

Activating system frames

System frames are activated by:

- Programming the corresponding system function in the part program
- Operator control at SINUMERIK Operate

Note

Modifying system frames of the data management

In principle, system frames of the data management can be modified by the cycle programmer and activated using a `G500`, `G54`...`G599` command. However, this option should only be used with reservation.

Activating data management frames

The behavior when activating data management frames is set using the following machine data:

MD24050 $MC_FRAME_SAA_MODE (save and activate data management frames)

Bit	Val-ue	Meaning
0	0	@@@
	1	• Data management frames are **only** activated by programming the system variables $P_CHBFRMASK, $P_NCBFRMASK and $P_CHSFRMASK. • `G500`...`G599` activates the appropriate settable frame.
1	0	Data management frames are implicitly described by functions, such as `TOROT`, `PAROT`, external work offset and transformations.
	1	Data management frames are **not** implicitly described by functions, such as `TOROT`, `PAROT`,external work offset and transformations.

Activation of system frames via system variable $P_CHSFRMASK

The system frames of the data management can be activated using system variable **$P_CHSFRMASK**. The value of the variables is specified as bit coded according to the machine data:
MD28082 $MC_MM_SYSTEM_FRAME_MASK (system frames of the data management)

The corresponding system frame of the data management in the channel is activated by setting a bit of system variable $P_CHSFRMASK to 1. For a value of 0, the currently active system frame in the channel remains active.

Activating system frames after RESET

After RESET, the system frames in the channel are activated whose bits are set in the following machine data:

MD24006 $MC_CHSFRAME_RESET_MASK (active system frames after Reset)

Activating system frames for TCARR, PAROT and TOROT, TOFRAME

The system frames for `TCARR`, `PAROT` and `TOROT`, `TOFRAME` are activated according to the setting in the following machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

When changing over geometry axes using transformation selection/deselection or the `GEOAX` command, the actual total frame $P_ACTFRAME is either deleted or is re-calculated using the new geometry axis constellation and activated. The system frames and all other frames are conditioned again in relation to the geometry axes.

11.5.3.3 NCU-global and channel-specific frames

- Settable frames and grinding frames can only be configured as NCU-global **or** channel-specific frames.
- Basic frames can be configured as NCU-global **and** channel-specific frames.
- An NCU-global frame has the same effect in all channels of the NCU.

- All channels of an NCU can read and write NCU-global frames equally.
- As the assignment of machine axes to channel axes and, in particular, to geometry axes, can be different in all channels, there is consequently no unique cross-channel geometric relationship between the channel axes. Therefore, only offset, scaling and mirroring is possible for NCU-global frames. Rotations are not possible.

Note

Program coordination

The coordination of channel-specific accesses to NCU-global frames is the sole responsibility of the user. It is recommended that the commands for the program coordination be used.

References

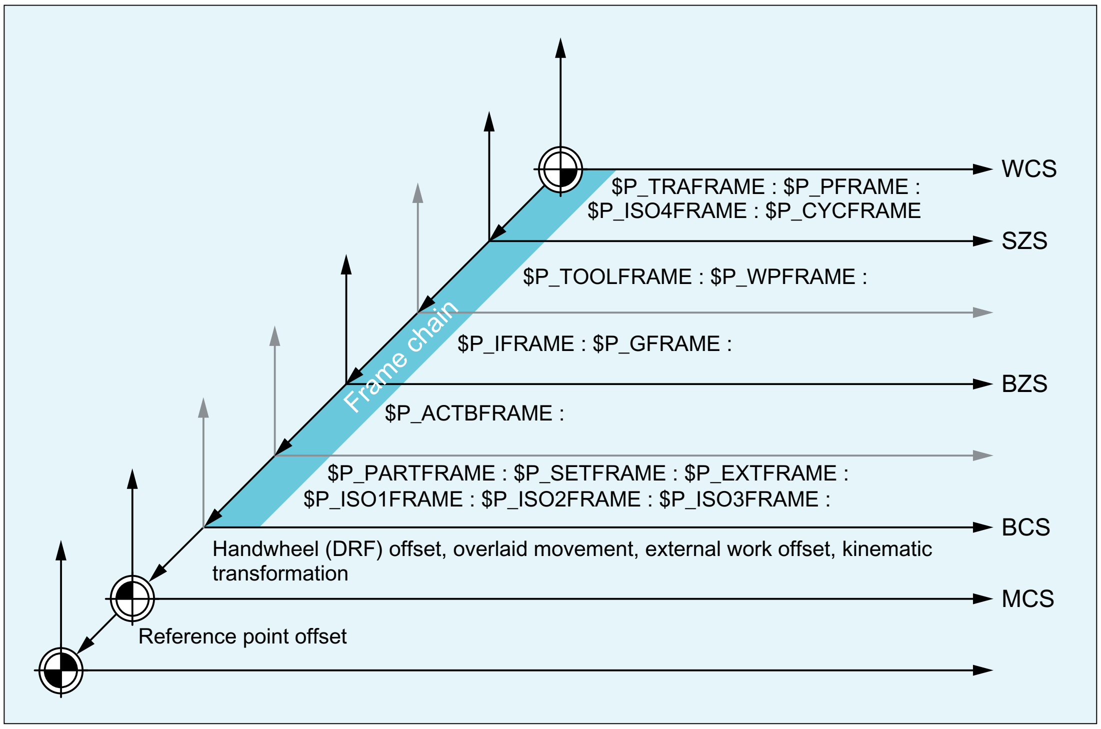
Programming Manual, Job Planning; Section "Flexible NC programming" > "Program coordination (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)"

11.5.4 Frame chain and coordinate systems

11.5.4.1 Overview

The figure below shows the frame chain for the current complete frame. The frame chain is located between the basic coordinate system (BCS) and the workpiece coordinate system (WCS).

The settable zero system (SZS) corresponds to the WCS transformed by the programmable frame. The basic zero system (BZS) still includes the current settable frame. The system frame for the external work offset exists only if it has been configured. Otherwise, the external work offset is traversed as overlaid movement of the axis.


WCS
$P_TRAFRAME : $P_PFRAME :
$P_ISO4FRAME : $P_CYCFRAME
SZS
$P_TOOLFRAME : $P_WPFRAME :
Frame chain
$P_IFRAME : $P_GFRAME :
BZS
$P_ACTBFRAME :
$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
BCS
Handwheel (DRF) offset, overlaid movement, external work offset, kinematic transformation
MCS
Reference point offset

WCS:	**W**orkpiece **C**oordinate **S**ystem
SZS:	**S**ettable **Z**ero **S**ystem
BZS:	**B**asic **Z**ero **S**ystem
BCS:	**B**asic **C**oordinate **S**ystem
MCS:	**M**achine **C**oordinate **S**ystem

Complete frame

The current complete frame `$P_ACTFRAME` results from the chaining of all active frames of the frame chain:

```
$P_ACTFRAME =    $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
                 $P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
                 $P_ACTBFRAME : $P_IFRAME : $P_GFRAME :
                 $P_TOOLFRAME :$P_WPFRAME : $P_TRAFRAME :
                 $P_PFRAME :$P_ISO4FRAME : $P_CYCFRAME
```

11.5.4.2 Relative coordinate systems

Relative coordinate systems display the current setpoint positions of the axes which lie relative to a specified reference point in the active displayed coordinate system. No programming can be done regarding the relative coordinate systems. Only the axis positions in these systems can be read via the system variables.

The new display coordinate systems lie relative to WCS and SZS coordinate system and result through transformation of the WCS or SZS axis positions with the active system frame $P_RELFRAME. The relative coordinate systems can not only be displaced linearly, but also rotated, mirrored, compressed or expanded.

The position indicator for axis setpoints is done in WCS or in SZS. The configuring is done via HMI machine data. Always only one display-coordinate system is active in the channel. For this reason only one relative frame is provided which generates both relative coordinate systems in the same ratio. The HMI displays the relative coordinates according to the configuration.

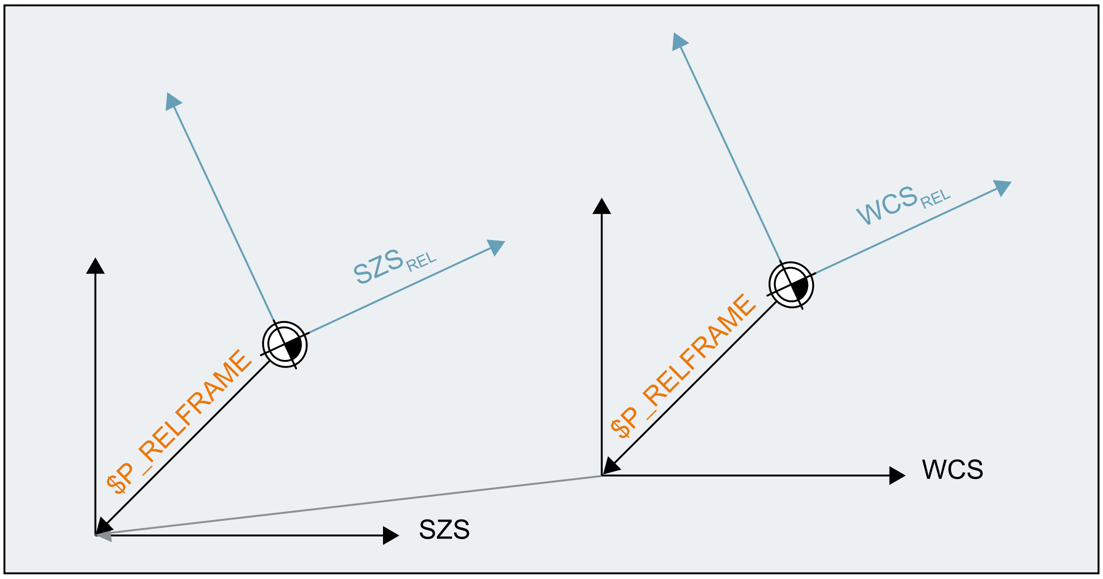


Figure 11-22 Relative coordinate systems

The "Relative coordinate systems" function is activated with MD51036 $MNS_ENABLE_COORDINATE_REL=1.

The data maintenance frame $P_RELFR can be written in the part program and via BTSS. All the frame components can be modified.

The active system frame $P_RELFRAME can be written in the part program and via BTSS.

The configuring of the system frame $P_RELFR is done via the following machine data:

Machine data	Bit	Meaning
MD28082 $MC_MM_SYSTEM_FRAME_MASK	11	Creation of $P_RELFR; with this, relative coordinate systems become existent.
MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK	11	Data maintenance frame $P_RELFR
MD24006 $MC_CHSFRAME_RESET_MASK.	11	$P_RELFR becomes active at Reset
MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK	11	$P_RELFR is deleted at Reset
MD24008 $MC_CHSFRAME_POWERON_MASK	11	$P_RELFR is deleted at PowerOn

The axis position in the relative coordinate system WCS_{Rel} can be read via the variable $AA_PCS_REL[ax]. The variable can be read in part program, BTSS and via synchronized actions.

The axis position in the relative coordinate system SZS_{Rel} can be read via the variable $AA_ACS_REL[ax]. The variable can be read in part program, BTSS and via synchronized actions.

The setting of a relative reference point via the operator panel is done via the general command interface for the workpiece and tool measuring. The system frame $P_RELFR for relative coordinate systems is calculated and activated as follows:

- $AC_MEAS_TYPE = 14
- PI-services _N_SETUDT(6, 7)

An example of setting the relative axis positions can be found in:
References:
Function Manual, Extended functions; Measurement (M5),
Section "Measurement of geometry and special axes (meas. type 14, 15)"

11.5.4.3 Selectable SZS

Within a cycle, machining is performed in a cycle-specific workpiece coordinate system (WCS). The cycle-specific WCS results from SZS transformed by the programmable frame $P_PFRAME and/or cycle frame $P_CYCFRAME programmed for the cycle.

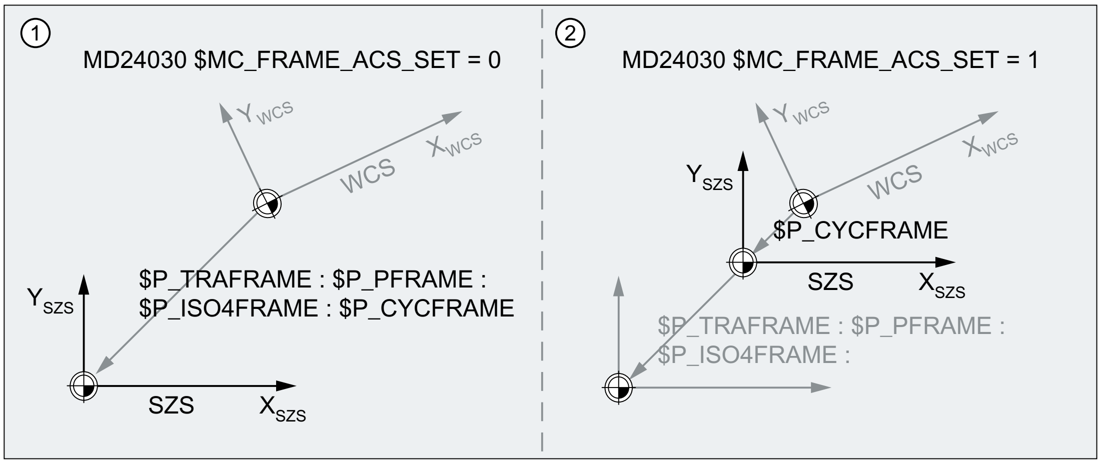
If a cycle is interrupted by a machine operator, e.g. through an NC stop, traversing should then be performed in the original coordinate system (SZS) valid before the activation of the cycle.

Machine data

The specification how the SZS is to be calculated from the cycle-specific WCS is set via the following machine data.

MD24030 $MC_FRAME_ACS_SET = <value>

<value>	Meaning
0	SZS = WCS transformed with $P_TRAFRAME, $P_PFRAME, $P_ISO4FRAME and $P_CYCFRAME
1	SZS = WCS transformed only with $P_CYCFRAME


①
MD24030 $MC_FRAME_ACS_SET = 0
Y_WCS
X_WCS
WCS
$P_TRAFRAME : $P_PFRAME :
$P_ISO4FRAME : $P_CYCFRAME
Y_SZS
SZS
X_SZS
②
MD24030 $MC_FRAME_ACS_SET = 1
Y_WCS
X_WCS
WCS
Y_SZS
$P_CYCFRAME
SZS
X_SZS
$P_TRAFRAME : $P_PFRAME :
$P_ISO4FRAME :

① SZS = WCS transformed with $P_TRAFRAME, $P_PFRAME, $P_ISO4FRAME and $P_CYCFRAME

② SZS = WCS transformed only with $P_CYCFRAME

Effects

The reconfiguration of the SZS has an effect on:

- SZS-related actual values: Actual-value displays, system variables, e.g. $AA_IEN, etc.
- Manual traversing (JOG) of geometry axes in the SZS

11.5.4.4 Manual traversing of geometry axes either in the WCS or in the SZS ($AC_JOG_COORD)

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 traversing 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.

During manual traversing in the JOG mode, the geometry axes can be traversed either in the workpiece coordinate system (WCS) or in the settable zero system (SZS).

System variables

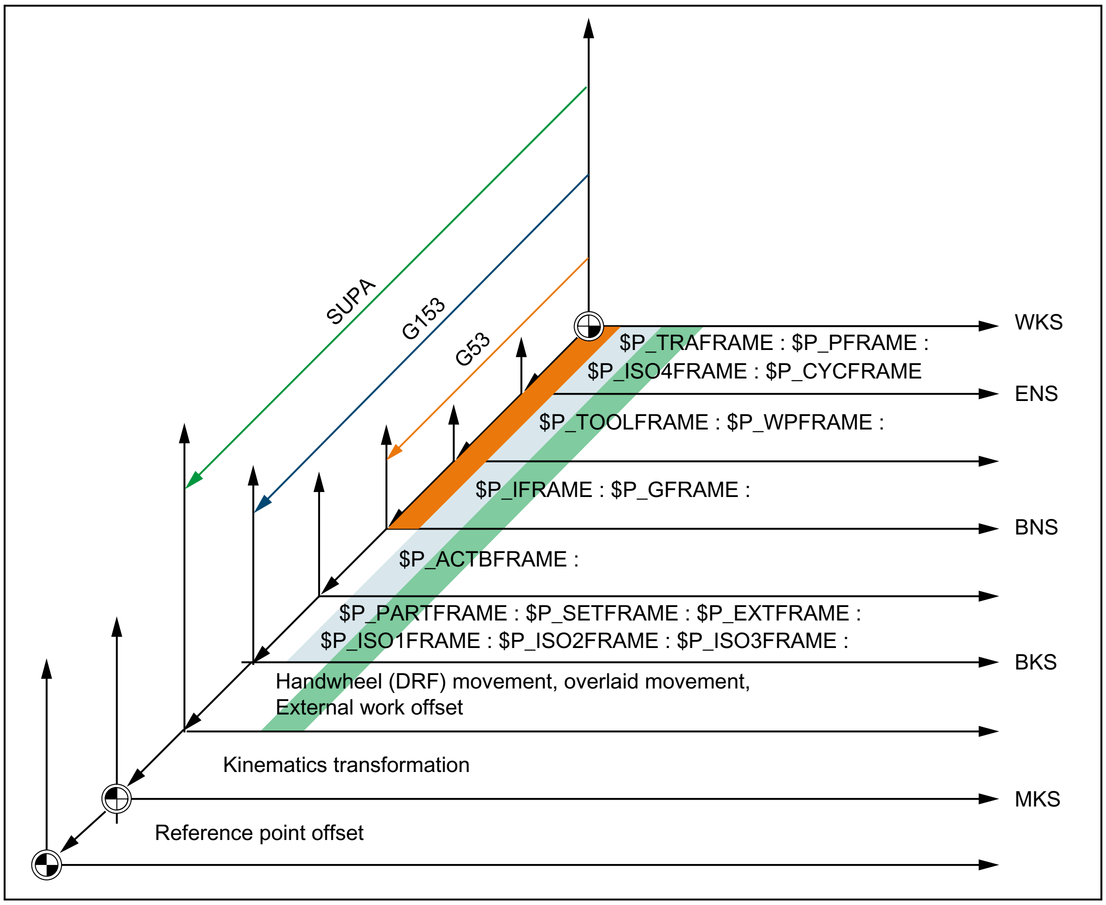
During manual traversing in the JOG mode, the geometry axes can be traversed either in the workpiece coordinate system (WCS) or in the settable zero system (SZS). The selection is made via the system variable $AC_JOG_COORD:

$AC_JOG_COORD = <value>

<Value>	Meaning
0	Workpiece coordinate system (WCS)
1	Settable zero system (SZS)

11.5.4.5 Suppression of frames

Suppression of frames is performed channel-specifically via the commands `G53`, `G135` and `SUPA` described in the following. Activation of the frame suppression results in jumps in the position display (HMI) as well as in the position values in system variables that refer to the WCS, SZS or BZS. The behavior can be set via machine data.


SUPA
G153
G53
WKS
$P_TRAFRAME : $P_PFRAME :
$P_ISO4FRAME : $P_CYCFRAME
ENS
$P_TOOLFRAME : $P_WPFRAME :
$P_IFRAME : $P_GFRAME :
BNS
$P_ACTBFRAME :
$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
BKS
Handwheel (DRF) movement, overlaid movement,
External work offset
Kinematics transformation
MKS
Reference point offset

Machine data

The behavior of position displays (HMI) and position values in system variables is specified via the following machine data:

MD24020 $MC_FRAME_SUPPRESS_MODE, bit<n> = <value> (positions during frame suppression)

Bit	Val-ue	Meaning
0	0	Position values (OPI) **with** frame suppression [1)]
	1	Position values (OPI) **without** frame suppression [2)]

Bit	Val-ue	Meaning
1	0	Position values in system variables **with** frame suppression [1)]
	1	Position values in system variables **without** frame suppression [2)]
1) Jump of the position value 2) **No** jump of the position value		

Programming

Com-mand	Meaning

`G53`:	Nonmodal suppression of the following frames:
	`$P_TRAFRAME : $P_PFRAME : $P_ISO4FRAME : $P_CYCFRAME`
	`$P_IFRAME : $P_GFRAME : $P_TOOLFRAME : $P_WPFRAME :`

`G153`:	Non-modal suppression of the frames as for `G53` plus following frames:
	`$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_ACTBFRAME`
	`$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :`

`SUPA`:	Implicit preprocessing stop and non-modal suppression of the frames as for `G53` and `G135` plus following frames: • Handwheel offsets (DRF) • Overlaid movement • External work offset
`G500`:	Modal activation of the `G500` frame. The `G500` frame should normally be a zero frame.
`DRFOF`:	Deactivation (deletion) of the handwheel offsets (DRF)

11.5.5 Frames of the frame chain

11.5.5.1 Overview

The following frames are available:

- Settable frames (`G500, G54` ... `G57`, `G505` ... `G599`)
- Grinding frames (`GFRAME0` ... `GFRAME100`)
- Basic frames
- Programmable frame
- System frames

11.5.5.2 Settable frames ($P_UIFR[<n>])

Machine data

Channel-specific settable frames

The number of channel-specific settable frames is set with the following machine data:

MD28080 $MC_MM_NUM_USER_FRAMES = <number>

System variable index n = 0, 1, 2, ... <number> - 1

NCU-global settable frames

The number of NCU-global settable frames is set with the following machine data:

MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES = <number>

System variable index n = 0, 1, 2, ... <number> - 1

If the machine data has a value > 0, there are **no channel-specific** settable frames. The machine data to set the channel-specific settable frames is then not evaluated.

Reset position of the G group of the settable frames

The reset position or which of the G commands of the settable frame-specific **8th G group** take effect after channel reset or Power On is set in:

MD20150 $MC_GCODE_RESET_VALUES[**7**] = <value>

Value	G command
1	`G500`
2	`G54`
3	`G55`
4	`G56`
5	`G57`
6	`G505`
...	...
100	`G599`

Reset behavior of the grinding-specific G group

The reset behavior of the settable frame-specific **8th G group** is set in:

MD20152 $MC_GCODE_RESET_MODE[**7**] = <value>

Value	Meaning **After channel reset or part program end:**
0	The settable frame-specific G command in accordance with **MD20150** is active.
1	The currently active settable frame-specific G command remains active.

Note

MD20110 $MC_RESET_MODE_MASK

The MD20152 $MC_GCODE_RESET_MODE machine data is evaluated only for:

MD20110 $MC_RESET_MODE_MASK, bit 0 == 1

System variables

$P_UIFR[<n>] (settable frames of the data management)

System variable $P_UIFR[<n>] can be used to read and write the settable frames of the data management. The new values are not immediately active in the channel when writing a settable frame of the data management. The activation in the channel only takes effect with the programming of a work offset `G500, G54`...`G599`.

For NCU-global frames, the changed settable frame of the data management is active in each channel of the NCU which executes a `G500, G54`..`G599` command.

The settable frames in the data management are also stored during a data backup.

$P_IFRAME (active settable frame)

System variable $P_IFRAME can be used to read and write the settable frame **active** in the channel. The new values are immediately active in the channel when writing the settable frame.

In the case of NCU-global settable frames, the changed active frame acts only in the channel in which the new values were programmed. If the changed NCU-global settable frame is to be active in all channels of the NCU, the settable frame active in the channel **and** the corresponding settable frame of the data management must be written together:

$P_UIFR[<n>] = $P_IFRAME = <new value>

- $P_UIFR[<n>] (settable frame in the data management)
- $P_IFRAME (settable frame active in the channel)

For the changed settable frame to take effect in another channel, it must be activated in this channel with the appropriate command, e.g. `G54`.

$P_UIFRNUM (number of active settable frame)

System variable $P_UIFRNUM can be used to read the index <n> of the settable frame of the data management active in the channel:

Settable frame active in the channel $P_IFRAME == $P_UIFR[$P_UIFRNUM]

$P_UIFRNUM	$P_IFRAME == $P_UIFR[<n>], where n =
0	0
1	1
2	2
...	...
99	99

Programming

Commands for activating a settable frame in the channel

The programming of a `G500, G54`...`G599` command activates the settable frame of the $P_UIFR[<n>] data management in the channel or the active $P_IFRAME settable frame set identical to the settable frame of the $P_UIFR[<n>] data management:

`G<x>` ⇒ $P_IFRAME = $P_UIFR[<n>]

Command	Active settable frame $P_IFRAME
`G500`	$P_UIFR[0]
`G54`	$P_UIFR[1]
`G55`	$P_UIFR[2]
`G56`	$P_UIFR[3]
`G57`	$P_UIFR[4]
`G505`	$P_UIFR[5]
...	...
`G599`	$P_UIFR[99]

Supplementary conditions

Writing of settable frames through HMI/PLC

Only the settable frames of the **data management** can be written from the HMI or the PLC user program.

11.5.5.3 Grinding frames $P_GFR[<n>]

Grinding frames are additional work offsets or "fit-dependent corrections" available particularly for the grinding technology. They are added to the work offset of the adjustable frames (Page 774).

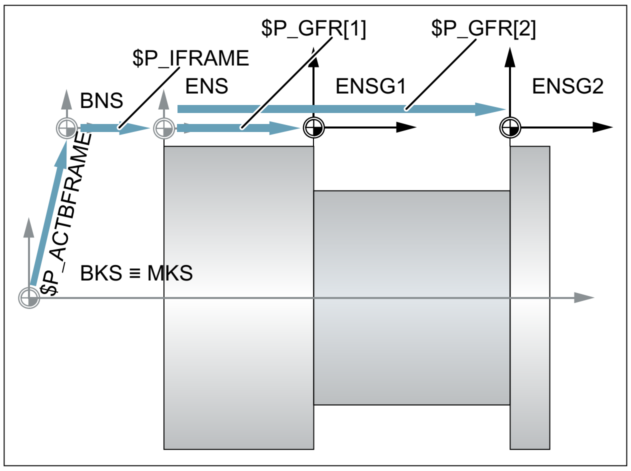


Figure 11-23 Grinding frames

Based on the base zero system (BZS), the settable zero system (SZS) results from the chaining of the frames active in the channel:

$P_IFRAME: **$P_GFRAME** : $P_TOOLFRAME :$P_WPFRAME

Machine data

Number of channel-specific grinding frames

The number of channel-specific grinding frames is set in:

MD28079 $MN_MM_NUM_G_FRAMES = <number>

with <number> = 0, 1, 2, ... maximum number

Number of NCU-global grinding frames

The number of NCU-global grinding frames is set in:

MD18603 $MN_MM_NUM_GLOBAL_G_FRAMES = <number>

with <number> = 0, 1, 2, ... maximum number

If the machine data has a value > 0, there are **no channel-specific** grinding frames. The machine data to set the channel-specific grinding frames is then not evaluated.

Reset position of the grinding-specific G group (64)

The reset position or which of the G commands of the grinding-specific **64th G group** take effect after channel reset or Power On is set in:

MD20150 $MC_GCODE_RESET_VALUES[**63**] = <value>

Value	G command
1	`GFRAME0` (zero frame)
2	`GFRAME1`
3	`GFRAME2`
...	...
101	`GFRAME100`

Reset behavior of the grinding-specific G group

The reset behavior of the grinding-specific **64th G group** is set in:

MD20152 $MC_GCODE_RESET_MODE[**63**] = <value>

Value	Meaning After channel reset or part program end:
0	The grinding-specific G command in accordance with **MD20150** is active.
1	The currently active grinding-specific G command remains active.

Note

MD20110 $MC_RESET_MODE_MASK

The MD20152 $MC_GCODE_RESET_MODE machine data is evaluated only for:

MD20110 $MC_RESET_MODE_MASK, bit 0 == 1

System variables

$P_GFR[<n>] (grinding frames of the data management)

System variable $P_GFR[n] can be used to read and write the grinding frames of the data management. The new values are not immediately active in the channel when writing a grinding frame. The activation in the channel only takes effect with the programming of the appropriate command `GFRAME0`...`GFRAME100`. For NCU-global frames, the changed frame only becomes active in those channels of the NCU, which execute a `GFRAME0` ... `GFRAME100` command.

The grinding frames in the data management are also stored during a data backup.

Note

Display (SINUMERIK Operate)

The grinding frames of the data management are displayed in a separate window of the SINUMERIK Operate user interface.

- References
  Operating Manual Grinding; Section "Setting up the machine" > "Work offsets" > "Displaying and editing seat-related fine offset"

Delete (SINUMERIK Operate)

The grinding frames of the data management can be deleted individually or, for example, after a workpiece change, all together via the SINUMERIK Operate user interface.

- References
  Operating Manual Grinding; Section "Setting up the machine" > "Work offsets" > "Displaying and editing seat-related fine offset"

$P_GFRAME (active grinding frame)

The grinding frame **active** in the channel can be read from and written to the $P_GFRAME system variable. The new values are immediately active in the channel when writing the active grinding frame.

In the case of NCU-global grinding frames, the changed frame acts only in the channel in which the new frame values were programmed.

If the changed NCU-global grinding frame is to be active in all channels of the NCU starting from one channel, the $P_GFRAME grinding frame active in the grinding frame and the grinding frame in the $P_GFR[<n>] data management must be written together:

$P_GFRAME = $P_GFR[<n>] = <new value>

For the changed grinding frame in the $P_GFR[<n>] data management to take effect in another channel, it must be activated in this channel with the appropriate command `GFRAME<n>`.

$P_GFRNUM (number of the active grinding frame)

System variable $P_GFRNUM can be used to read the index <n> of the grinding frame of the data management active in the channel:

Grinding frame active in the channel $P_GFRAME == $P_GFR[$P_GFRNUM]

$P_GFRAME grinding frame activated with the G command GFRAME<n>:	$P_GFRNUM
`GFRAME0`	0
`GFRAME1`	1
`GFRAME2`	2
...	...
`GFRAME100`	100

Programming

Command for activating a grinding frame in the channel

The programming of the `GFRAME<n>` command makes the associated grinding frame of the $P_GFR[<n>] data management active in the channel. This sets the active $P_GFRAME grinding frame identical to the $P_GFR[<n>] grinding frame of the data management:

`GFRAME<n>` ⇒ $P_GFRAME = $P_GFR[<n>]

Command	Grinding frame activated in the channel
`GFRAME0`	$P_GFR[0] (**null frame**)
`GFRAME1`	$P_GFR[1]
...	...
`GFRAME100`	$P_GFR[100]

Syntax

```
GFRAME<n>
```

Meaning

`GFRAME<n>`:	Activation of the grinding frame <n> of the data management	
	G group:	64
	Basic position:	MD20150 $MC_GCODE_RESET_VALUES[**63**]
	Effective:	Modal
`<n>`:	Number of the grinding frame	
	Value range:	0, 1, 2, ... 100

Supplementary conditions

Writing of grinding frames through HMI/PLC

Only the grinding frames of the **data management** can be written from the HMI or the PLC user program.

11.5.5.4 Channel-specific basic frames[<n>]

Machine data

Number of channel-specific basic frames

The number of channel-specific basic frames is set with the following machine data:

MD28081 $MC_MM_NUM_BASE_FRAMES = <number>

System variable index n = 0, 1, 2, ... <number> - 1

System variables

$P_CHBFR[<n>] (channel-specific basic frames of the data management)

System variable $P_CHBFR[<n>] can be used to read and write the channel-specific basic frames of the data management. The new values are not immediately active in the channel when writing a channel-specific basic frame. The activation in the channel only takes effect with the programming of the appropriate command `G500, G54`...`G599`.

The channel-specific basic frames in the data management are also stored during a data backup.

$P_CHBFRAME[<n>] (active channel-specific basic frames)

System variable $P_CHBFRAME[<n>] can be used to read and write the **active** channel-specific basic frames. When writing an active channel-specific basic frame, the new values take effect immediately through the recalculation of the active complete basic frame $P_ACTBFRAME.

System variables for compatibility reasons

$P_UBFR (first channel-specific basic frame of the data management)

The system variable is retained for reasons of compatibility, although it is redundant for the $P_CHBFR[0] variables.

The basic frame with array index 0 is not activated simultaneously when writing to the predefined $P_UBFR variable, but rather activation only takes place on execution of a `G500, G54,.G599` statement. For NCU-global frames, the changed frame only becomes active in those channels of the NCU, which execute a `G500, G54`..`G599` command. The variable is used primarily for storing write operations to the basic frame on HMI or PLC. The variable can also be read and written in the program.

$P_UBFR is identical to $P_CHBFR[0]. One basic frame always exists in the channel by default, so that the system variable is compatible with older versions. If there is no channel-specific basic frame, an alarm: "Frame: Statement not permitted" is output on a read or write access.

$P_BFRAME (first active channel-specific basic frame)

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 array index 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: "Frame: Statement not permitted" is output on a read or write access.

Supplementary conditions

Writing of basic frames from HMI/PLC

Only the basic frames of the **data management** can be written from the HMI or the PLC user program.

11.5.5.5 NCU-global basic frames $P_NCBFR[<n>]

Machine data

Number of NCU-global basic frames

The number of NCU-global basic frames is set with the following machine data:

MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES = <number>

System variable index n = 0, 1, 2, ... <number> - 1

System variables

$P_NCBFR[<n>] (NCU-global basic frames of the data management)

System variable $P_NCBFR[<n>] can be used to read and write the NCU-global basic frames of the data management. The new values are not immediately active in the channel when writing an NCU-global basic frame. The activation in the channel only takes effect with the programming of the appropriate command `G500, G54`...`G599`.

The NCU-global basic frames in the data management are also stored during a data backup.

$P_NCBFRAME[<n>] (current NCU-global basic frames)

System variable $P_NCBFRAME[<n>] can be used to read and write the **active** NCU-global basic frames. When writing an active NCU-global basic frame, the new values take effect immediately through the recalculation of the active complete basic frame $P_ACTBFRAME.

If the changed NCU-global basic frame is to be active in all channels of the NCU starting from one channel, the NCU-global basic frame active in the channel and the NCU-global frame in the data management must be written together:

$P_NCBFR[<n>] = $P_NCBFRAME = <new value>

- $P_NCBFR[<n>] (NCU-global basic frame of the data management)
- $P_NCBFRAME (NCU-global basic frame active in the channel)

For the changed NCU-global frame to take effect in another channel, it must be activated in this channel with the appropriate command `G500, G54`..`G599`.

Programming

A channel-specific settable frame of the data management $P_UIFR[<n>] becomes the settable frame $P_IFRAME active in the channel through the appropriate command (`G54` ... `G57`, `G505` ... `G599` and `G500`).

Command	Activation of the NCU-global and channel-specific basic frames of the data management
G500	$P_CHBFR[0] : $P_NCBFR[0]
G54	$P_CHBFR[1] : $P_NCBFR[1]
G55	$P_CHBFR[2] : $P_NCBFR[2]
G56	$P_CHBFR[3] : $P_NCBFR[3]
G57	$P_CHBFR[4] : $P_NCBFR[4]
G505	$P_CHBFR[5] : $P_NCBFR[5]
...	...
G599	$P_CHBFR[99] : $P_NCBFR[99]

11.5.5.6 Active complete basic frame $P_ACTBFRAME

Function

All active NCU-global and channel-specific basic frames are combined into the complete basic frame `$P_ACTBFRAME`:

```
$P_ACTBFRAME = $P_NCBFRAME[0] : ... : $P_NCBFRAME[<n>] :
               $P_CHBFRAME[0] : ... : $P_CHBFRAME[<n>]
```

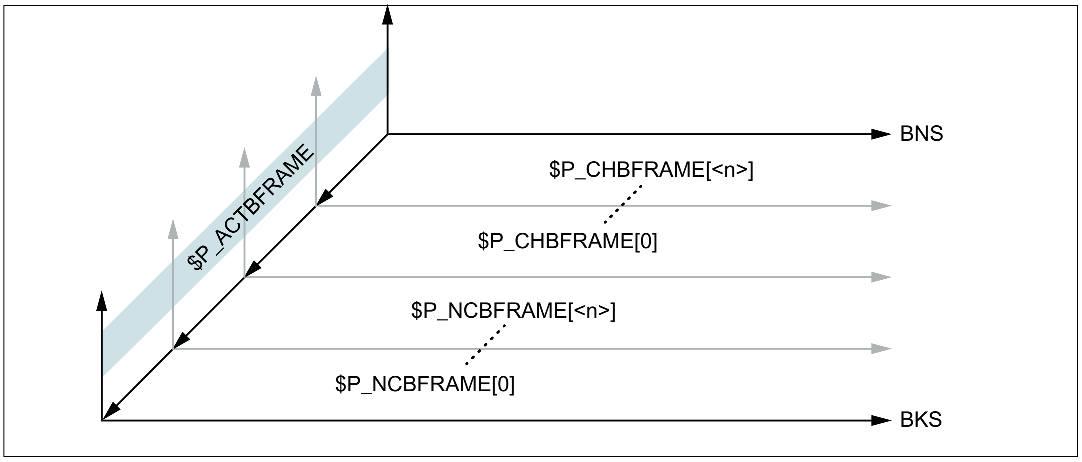


Figure 11-24 Complete basic frame

Machine data

Reset response

Which basic frames are active after a reset (channel reset, end of program reset or Power On) is set via the machine data:

MD20110 $MC_RESET_MODE_MASK, bit0 = 1 and bit14 = 1

- Bit 1 = 0: Default value ⇒ reset behavior corresponding to the setting of the further bits
- Bit14 = 0: The basic frames are completely deselected with reset.
- Bit14 = 1: With reset, the machine data settings are taken over in the system variables and the basic frames selected therein become active:
  - $P_NCBFRMASK = MD10613 $MN_NCBFRAME_RESET_MASK
  - $P_CHBFRMASK = MD24002 $MC_CHBFRAME_RESET_MASK

  Example

Machine data setting	Active basic frames
$P_NCBFRMASK = MD10613 $MN_NCBFRAME_RE-SET_MASK = 'H81'	$P_NCBFRAME[0] : $P_NCBFRAME[7]

Programming

Basic frame masks

The basic frames that are linked to form the complete basic frame are selected via the basic frame masks `$P_NCBFRMASK` and `$P_CHBFRMASK`.

The appropriate basic frame is selected by setting a bit in the basic frame **mask**:

- `$P_NCBFRMASK, bit` **0, 1, 2, ... n** ⇒ `$P_NCBFRAME[`**0, 1, 2, ... n**`]`
- `$P_CHBFRMASK, bit` **0, 1, 2, ... n** ⇒ `$P_NCHFRAME[`**0, 1, 2, ... n**`]`

The basic frame masks `$P_NCBFRMASK` and `$P_CHBFRMASK` can only be read/written in the NC program. The basic frame masks can be read via the OPI.

After writing a basic frame mask, the active complete basic frame `$P_ACTBFRAME` and complete frame `$P_ACTFRAME` are recalculated.

Example

Program code	Comment
`$P_NCBFRMASK = 'H81'`	`; Active NCU-global basic frames: $P_NCBFRAME[0] : $P_NCBFRAME[7]`

11.5.5.7 Programmable frame $P_PFRAME

Programmable frames are available only as active frames.
This frame is reserved for the programmer.

The programmable frame remains at RESET, if:

MD24010 $MC_PFRAME_RESET_MODE (reset mode for programmable frame) = 1

This functionality is especially important after a RESET if one still wants to retract out of an oblique hole.

MIRROR

Mirrorings of a geometry axis were previously (up to SW-P4) related to a defined reference axis only using the machine data:
MD10610 $MN_MIRROR_REF_AX
(reference axis for the mirroring).

From the user's point of view, this definition is difficult to understand. When mirroring the z axis, the display showed that the x axis was mirrored and the y axis had been rotated through 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.

As of SW P5, there is the option to clearly display the mirroring of an axis. Mirroring is then not mapped to mirroring of a reference axis and rotations of other axes.

This setting can be configured using:

MD10610 $MN_MIRROR_REF_AX = 0

`MIRROR` and `AMIRROR` are used to expand the programming of the programmable frame. Previously, the specified value of the coordinate axis, e.g. the value 0 for `MIRROR X0` was not evaluated, but rather the `AMIRROR` had a toggle function, i.e. `MIRROR X0` activates mirroring and an additional `AMIRROR X0` deactivates it. `MIRROR` always has an absolute effect and `AMIRROR` an additive effect.

The
MD10612 $MN_MIRROR_TOGGLE = 0 ("Mirror Toggle")
machine data setting can be used to define that the programmed values are evaluated.
A value of 0, as in`AMIRROR X0`, deactivates the mirroring of the axis, and values not equal to 0 cause the axis to be mirrored if it is not already mirrored.

Reading or writing mirroring component-by-component is independent of the machine data:

MD10612 $MN_MIRROR_TOGGLE

A value = 0 means that the axis is not mirrored and a value = 1 means that the axis will always be mirrored, irrespective of whether it has already been mirrored or not.

```
$P_NCBFR[0,x,mi]=1          ; x axis is always mirrored.
$P_NCBFR[0,x,mi]=0          ; x axis mirroring is OFF.
```

Axis-specific replacement G58, G59 (only 840D sl)

The translation component of the programmable frame is split into an absolute component and a component for the total of all additively programmed translations. The absolute component can be changed using `TRANS`, `CTRANS` or by writing the translation components, in which the additive component is set to zero. `G58` changes only the absolute translation component for the specified axis; the total of additively programmed translations is retained.

```
G58 X... Y... Z... A... ...
```

`G59` is used for axis-specific 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:

MD24000 $MC_FRAME_ADD_COMPONENTS (frame components for G58 / G59) == TRUE

The table below shows the effect of various program commands on the absolute and additive translation.

	Coarse or absolute offset	Fine or additive offset
`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

11.5.5.8 Channelspecific system frames

Channel-specific system frames are only written by system functions, such as actual value setting, scratching, external work offset and inclined machining.

Machine data

Parameterization of the channel-specific system frames

Only channel-specific system frames whose system functions are actually used should be configured for memory space reasons.

Per channel, each system frame occupies approx. 1 KB static and approx. 6 KB dynamic memory.

The channel-specific system frames are parameterized in the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit<n>

Bit	Value	System frame available:
0	1	$P_SETFR: Actual value setting and scratching
1	1	$P_EXTFR: External work offset via system frames

Bit	Value	System frame available:
2	1	$P_PARTFR: TCARR and PAROT with an orientable toolholder
3	1	$P_TOOLFR: TOROT and TOFRAME
4	1	$P_WPFR: Frame for workpiece reference points
5	1	$P_CYCFR: Frame for cycles
6	1	$P_TRAFR: Frame for selection and deselection of transformations
7	1	$P_ISO1FRAME : Frame for G51.1 mirroring (ISO)
8	1	$P_ISO2FRAME : Frame for G68 2DROT (ISO)
9	1	$P_ISO3FRAME : Frame for G68 3DROT (ISO)
10	1	$P_ISO4FRAME: Frame for G51 scale (ISO)
11	1	$P_RELFR: Frame for relative coordinate systems

Parameterization of the SZS (ACS) coordinate system

The following machine data is used to specify which system frames form the SZS (ACS) coordinate system

MD24030 $MC_FRAME_ACS_SET = <value>

<value>	Meaning: The SZS (ACS) coordinate system comprises
0	$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :$P_ACTBFRAME : $P_IFRAME : $P_GFRAME : $P_TOOLFRAME : **$P_WPFRAME**
1	$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :$P_ACTBFRAME : $P_IFRAME : $P_GFRAME : $P_TOOLFRAME : **$P_WPFRAME : $P_TRAFRAME :$P_PFRAME : $P_ISO4FRAME**

System variables

Channel-specific system frames of the data management

The channel-specific system frames of the data management can be read and written via the following frame variables:

System variable	Meaning: System frame of the data management for
$P_SETFR	Actual value setting and scratching (**Set Fr**ame)
$P_EXTFR	External work offset (**Ext Fr**ame)
$P_PARTFR	TCARR and PAROT for orientable toolholder (**Part Fr**ame)
$P_TOOLFR	TOROT and TOFRAME (**Tool Fr**ame)
$P_WPFR	Workpiece reference points (**W**ork**P**iece **Fr**ame)
$P_CYCFR	Cycles (**Cyc**le **Fr**ame)
$P_TRAFRAME	Transformations (**Tra**nsformation **Fr**ame)
$P_ISO1FR	G51.1 mirroring (ISO)
$P_ISO2FR	G68 2DROT (ISO)
$P_ISO3FR	G68 3DROT (ISO)
$P_ISO4FR	G51 scale (ISO)
$P_RELFR	Relative coordinate systems

Note

Cycle programming

The frame variables of the system frames are only for the cycle programming. Therefore, in NC programs the system frames should not be written directly by the user, but rather only via system functions such as `TOROT`, `PAROT`, etc.

Channelspecific active system frames

System variables of the active channel-specific system frames:

System variable	Meaning: Active system frame for
$P_SETFRAME	Actual value setting and scratching (**Set** **Fr**ame)
$P_EXTFRAME	External work offset (**Ext** **Fr**ame)
$P_PARTFRAME	`TCARR` and `PAROT` for orientable toolholder (**Part** **Fr**ame)
$P_TOOLFRAME	`TOROT` and `TOFRAME` (**Tool** **Fr**ame)
$P_WPFRAME	Workpiece reference points (**W**ork**P**iece **Fr**ame)
$P_CYCFRAME	Cycles (**Cyc**le **Fr**ame)
$P_TRAFRAME	Transformations (**Tra**nsformation **Fr**ame)
$P_ISO1FRAME	`G51.1` mirroring (ISO)
$P_ISO2FRAME	`G68` 2DROT (ISO)
$P_ISO3FRAME	`G68` 3DROT (ISO)
$P_ISO4FRAME	`G51` scale (ISO)
$P_RELFRAME	Relative coordinate systems

If a channel-specific system frame of the data management is not parameterized, the following applies for the corresponding active system frame: $P_<system frame> == null frame.

Channel-specific active SZS (ACS) complete frame

The system variable $P_ACSFRAME is used to read which system frames form the SZS (ACS) coordinate system. The specification is made via the MD24030 $MC_FRAME_ACS_SET machine data described above. See Subsection "Machine data" > "Parameterization of the SZS (ACS) coordinate system"

System variable	Meaning: Active system frame for
$P_ACSFRAME	System frames that form the SZS (ACS) coordinate system in accordance with the parameterization in MD24030 $MC_FRAME_ACS_SET

11.5.6 Implicit frame changes

11.5.6.1 Switching geometry axes

Which channel axes are the geometry axis of the channel can change by activating/deactivating a transformation and with the `GEOAX()` command.

Four different settings for handling the current $P_ACTFRAME complete frame can be made with the following machine data:

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = <value>

<value>	Meaning
0	Delete 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.
1	New calculation on receipt of the rotations The current complete frame is calculated again when the geometry axes are switched over, whereby the frame components of the new geometry axes are effective. The rotations of the geometry axes which were programmed before the switchover remain effective for the new geometry axes. For `TRANSMIT`, `TRACYL` and `TRAANG`, see Section "Selecting and deselecting transformations: General (Page 790)".
2	New calculation only when no rotations were active The current complete frame is calculated again when the geometry axes are switched over, whereby the frame components of the new geometry axes are effective. If rotations are active in the current basic frames, the current settable frame or the programmable frame before the switchover, it is aborted with the alarm "Frame: Geometry axis switchover not allowed". For `TRANSMIT`, `TRACYL` and `TRAANG`, see Section "Selecting and deselecting transformations: General (Page 790)".
3	Transformation: Reset / GEOAX(): New calculation on receipt of the rotations • Transformation: The current frame is deleted when selecting and deselecting transformations. • `GEOAX()`: With `GEOAX()`, the current complete frame is calculated again and the translations, scalings and mirrorings of the new geometry axes come into effect. The rotations of the geometry axes which were programmed before the switchover remain effective for the new geometry axes.

The workpiece geometry is described by a coordinate system that is formed by the geometry axes. A channel axis is assigned to each geometry axis and a machine axis is assigned to each channel axis. An axis-specific frame exists for each machine axis (system frame, basic frame, settable frame, programmable frame). If a different machine axis is assigned to a geometry axis, the machine axis provides its own axis-specific frame components. The new geometry in the channel is then generated by the new contour frames resulting from the new geometry axes (up to three).

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 A is to become a geometry axis X through `geo axis` substitution. The substitution is to give the programmable frame a translation component of 10 in the X axis. The current settable frame is to be retained.

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

Program code	Comment
`$P_UIFR[1] = CROT(X,10,Y,20,Z,30)`	`; Frame is retained after geo axis substitution.`

Program code	Comment
`G54`	`; Settable frame becomes active.`
`TRANS A10`	`; Axial offset of A is also substituted.`
`GEOAX(1, A)`	`; A becomes the 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 is probably 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

Program code

```
$MC_AXCONF_CHANAX_NAME_TAB[0] = "CAX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "CAY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "CAZ"
$MC_AXCONF_CHANAX_NAME_TAB[3] = "A"
$MC_AXCONF_CHANAX_NAME_TAB[4] = "B"
$MC_AXCONF_CHANAX_NAME_TAB[5] = "C"

$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"
$MC_AXCONF_GEOAX_NAME_TAB[1]="Y"
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=5
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=6

$MC_TRAFO_AXES_IN_1[0] = 4
$MC_TRAFO_AXES_IN_1[1] = 5
$MC_TRAFO_AXES_IN_1[2] = 6
$MC_TRAFO_AXES_IN_1[3] = 1
$MC_TRAFO_AXES_IN_1[4] = 2
```

Program:

Program code

```
$P_NCBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
$P_CHBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
$P_IFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(Z,45)
$P_PFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(X,10,Y,20,Z,30)
```

Program code	Comment
TRAORI	; Transformation sets GEOAX(4,5,6)
	; $P_NCBFRAME[0] = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3)
	; $P_ACTBFRAME =CTRANS(X,8,Y,10,Z,12,CAX,2,CAY,4,CAZ,6)
	; $P_PFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3):CROT(X,10,Y, 20,Z,30)
	; $P_IFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3):CROT(Z,45)
TRAFOOF	; Deactivation of the transformation sets GEOAX(1,2,3)
	; $P_NCBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
	; $P_CHBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
	; $P_IFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(Z,45)
	; $P_PFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(X,10,Y,20,Z, 30)

11.5.6.2 Selecting and deselecting transformations: General

As a rule, the assignment of geometry axes to the channel axes changes when selecting and deselecting transformations. It is not possible to uniquely assign axis-specific frame components to geometric contour frame components when carrying out transformations, in which rotary axes become linear axes and vice versa. The contour frame must be conditioned using special treatment for such non-linear transformations.

The mode, set with MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1 and 2, is expanded in such a way as to take the above transformations into account.

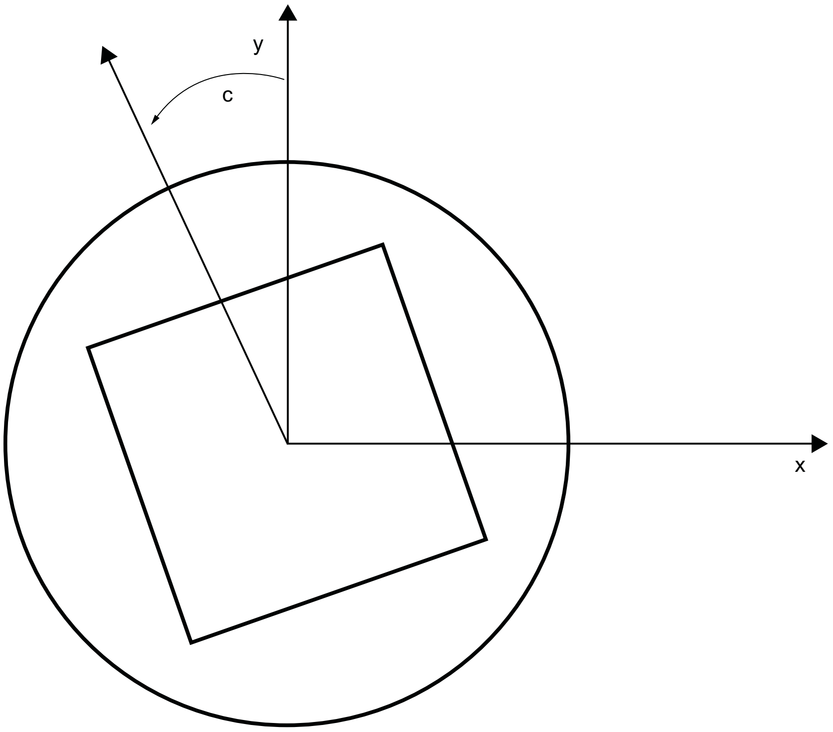
When selecting transformations, the contour frame is connected to the axis-specific frames. The virtual geometry axis of the TRANSMIT, TRACYL and TRAANG transformations is subject to special treatment.

Note

Transformations with virtual axes

When selecting TRANSMIT or TRACYL, offsets, scaling and mirroring of the real Y axis are not accepted in the virtual Y axis. Offsets, scaling and mirroring of the virtual Y axis are deleted for TRAFOOF.

11.5.6.3 Selecting and deselecting transformations: TRANSMIT



Transmit expansions

The machine data below can be used to take the axis-specific complete frame of the TRANSMIT rotary axis, i.e. translation, mirroring and scaling, into account in the transformation:

- MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 1
- MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 1

A rotary axis offset can, for example, be entered by compensating the oblique position of a workpiece in a frame within a frame chain. As a rule, this offset can also be included in the transformation as an offset in the rotary axis. A C axis offset, as in the figure above, then leads to corresponding X and Y values.

- MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 2
- MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 2

With this setting, the axis-specific offset of the rotary axis is taken account of in the transformation up to the SZS. The axis-specific offsets of the rotary axis included in the SZS frames are entered into the transformation frame as rotation. This setting is only effective if the transformation frame has been configured.

Frame expansions:

The expansions described below are only valid for the following machine data settings:

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

The selection of transformation TRANSMIT produces a virtual geometry axis, coupled by way of the rotary axis, which is merely included in the contour frame but does not have a reference to an axis-specific frame. The geometric value results from the rotation of a rotary axis. All other geometry axes accept their axis-specific components when the transformation is selected.

Components:

- Translations
  When selecting TRANSMIT, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.
- Rotations
  Rotations before the transformation are accepted.
- Mirrorings
  Mirroring of the virtual axis is deleted. Mirroring of the rotary axis can be taken into account in the transformation.
- Scalings
  Scaling of the virtual axis is deleted. Scaling of the rotary axis can be taken into account in the transformation.

Example: Machine data

```
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK='H41'            ; TRAFRAME, SETFRAME
$MC_CHSFRAME_RESET_MASK='H41'             ; Frames are active after Reset.
$MC_CHSFRAME_POWERON_MASK='H41'           ; Frames are deleted for Power On.

$MN_FRAME_GEOAX_CHANGE_MODE=1             ; Frames are calculated after switchover
                                          of the geo axis.
$MC_RESET_MODE_MASK='H4041'               ; Basic frame is not deselected after Re-
                                          set.
;$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 the 1st transformation
$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 the 2nd transformation
$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
```

Example: Part program

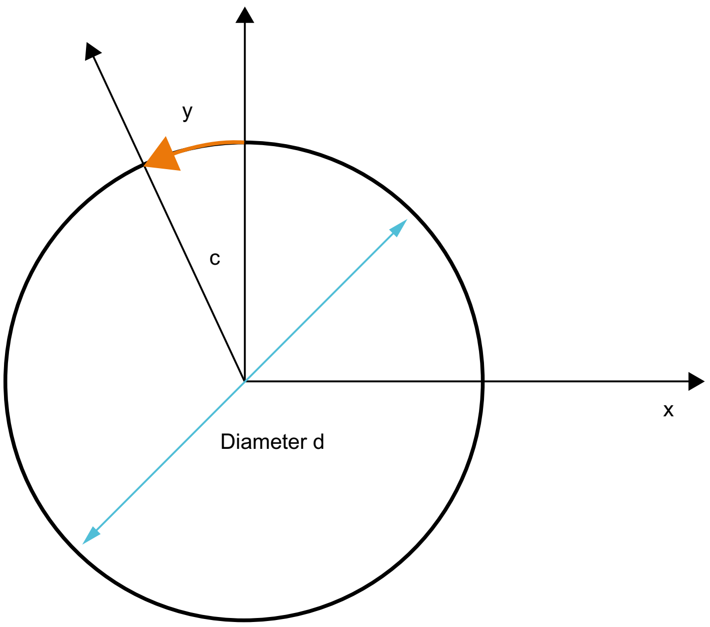
```
; Frame settings
N820 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,4)
N830 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N840 $P_UIFR[1] = $P_UIFR[1] : cmirror(x,c)
N850
N860 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,c,15)
N870


; Tool selection, clamping compensation, plane selection
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900


; Approach start position
N920 G0 X20 Z10
N930
N940 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N950 setal(61000)
N960 endif
N970 if $P_BFRAME <> $P_CHBFR[0]
N980 setal(61000)
N990 endif
N1000 if $P_IFRAME <>
CTRANS(X,1,Y,2,Z,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
N1060 if $P_ACTFRAME <>
CTRANS(X,11,Y,22,Z,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1070 setal(61000)
N1080 endif
N1090
N1100 TRANSMIT(2)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1130 setal(61000)
N1140 endif
```

```
N1180 if $P_IFRAME <>
CTRANS(X,1,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1190 setal(61000)
N1200 endif
N1240 if $P_ACTFRAME <>
CTRANS(X,11,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330


; Set frame
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370


; Four-edge roughing
N1390 G1 X10 Y-10 G41 OFFN=1; allowance 1 mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440


; Tool change
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490


; Four-edge finishing
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
N1540 X10
N1550 Y-10
N1560
```

```
; Deselect frame
 N2950 m30 N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
N1670 if $P_IFRAME <>
TRANS(X,11,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1680 setal(61000)
N1690 endif
N1730 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1740 setal(61001)
N1750 endif
N1760
N1770 TRAFOOF
N1780
N1790 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N1800 setal(61000)
N1810 endif
N1820 if $P_BFRAME <> $P_CHBFR[0]
N1830 setal(61000)
N1840 endif
N1850 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1860 setal(61000)
N1870 endif
N1880 if $P_IFRAME <> $P_UIFR[1]
N1890 setal(61000)
N1900 endif
N1910 if $P_ACTFRAME <>
TRANS(X,21,Y,22,Z,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1920 setal(61002)
N1930 endif
N1940
N2010 $P_UIFR[1] = ctrans()
N2011 $P_CHBFR[0] = ctrans()
N2020 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,0)
N2021 G54
N2021 G0 X20 Y0 Z10 C0
```

```
N2030 TRANSMIT(1)
N2040 TRANS x10 y20 z30
N2041 ATRANS y200
N2050 G0 X20 Y0 Z10
N2051 if $P_IFRAME <> CTRANS(X,1,Y,0,Z,3,CAY,2)
N2052 setal(61000)
N2053 endif
N2054 if $P_ACTFRAME <> CTRANS(X,11,Y,20,Z,33,CAY,2):CFINE(Y,200)
N2055 setal(61002)
N2056 endif
N2060 TRAFOOF
N2061 if $P_IFRAME <> $P_UIFR[1]
N2062 setal(61000)
N2063 endif
N2064 if $P_ACTFRAME <> CTRANS(X,11,Y,2,Z,33):CFINE(Y,0)
N2065 setal(61002)
N2066 endif
```

11.5.6.4 Selecting and deselecting transformations: TRACYL



TRACYL expansions:

The machine data below can be used to take the axis-specific 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 axis-specific offset of the rotary axis is taken account of in the transformation up to the SZS. The axial offsets of the rotary axis included in the SZS frames are entered into the transformation frame as offsets on the peripheral surface. This setting is only effective if the transformation frame has been configured.

Frame expansions:

The expansions described below are only valid for the following machine data settings:

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

The selection of transformation TRACYL produces a virtual geometry axis on the peripheral surface, coupled via the rotary axis, which is only taken into account in the contour frame but does not have a reference to an axis-specific frame. All components of the virtual geometry axis are deleted. All other geometry axes accept their axis-specific components when the transformation is selected.

Components:

- Translations
  When selecting TRACYL, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.
- Rotations
  Rotations before the transformation are accepted.
- Mirrorings
  Mirroring of the virtual axis is deleted. Mirroring of the rotary axis can be taken into account in the transformation.
- Scalings
  Scaling of the virtual axis is deleted. Scaling of the rotary axis can be taken into account in the transformation.

Example: Machine data

```
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK = 'H41'           ; TRAFRAME, SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41'            ; Frames are active after Reset.
$MC_CHSFRAME_POWERON_MASK = 'H41'          ; Frames are deleted for Power On.

$MN_FRAME_GEOAX_CHANGE_MODE = 1            ; Frames are calculated after switchover
                                           of the geo axis.
```

```
$MC_RESET_MODE_MASK = 'H4041'             ; Basic frame is not deselected after Re-
                                          set.
;$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 wall offset is the 3rd transformation

$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
```

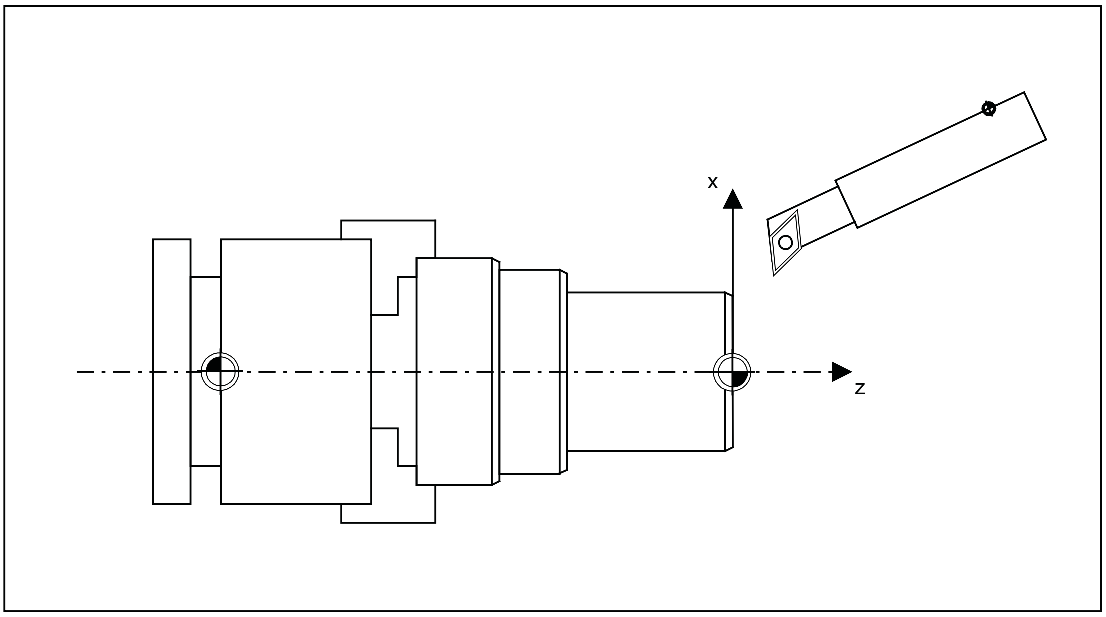
Example: Part program

```
;Simple traversing test with groove side offset
N450 G603
N460


; Frame settings
N500 $P_UIFR[1] = CTRANS(x,1,y,2,z,3,b,4)
N510 $P_UIFR[1] = $P_UIFR[1] : CROT(x,10,y,20,z,30)
N520 $P_UIFR[1] = $P_UIFR[1] : CMIRROR(x,b)
N530
N540 $P_CHBFR[0] = CTRANS(x,10,y,20,z,30,b,15)
N550
N560 G54
N570


; Continuous-path mode with selected smoothing
N590 G0 x0 y0 z-10 b0 G90 F50000 T1 D1 G19 G641 ADIS=1 ADISPOS=5
N600
N610 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N620 setal(61000)
N630 endif
N640 if $P_BFRAME <> $P_CHBFR[0]
N650 setal(61000)
N660 endif
N670 if $P_IFRAME <>
TRANS(X,1,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N680 setal(61000)
N690 endif
N700 if $P_IFRAME <> $P_UIFR[1]
N710 setal(61000)
N720 endif
N730 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N740 setal(61000)
N750 endif
N760


; Transformation ON
N780 TRACYL(40.)
N790
N800 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N810 setal(61000)
N820 endif
```

```
N830 if $P_CHBFR[0] <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N840 setal(61000)
N850 endif
N860 if $P_IFRAME <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N870 setal(61000)
N880 endif
N890 if $P_UIFR[1] <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N900 setal(61000)
N910 endif
N920 if $P_ACTFRAME <>
TRANS(X,11,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N930 setal(61001)
N940 endif
N950
N960 $P_UIFR[1,x,tr] = 11
N970 $P_UIFR[1,y,tr] = 14
N980
N990 g54
N1000
N1010 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N1020 setal(61000)
N1030 endif
N1040 if $P_BFRAME <> $P_CHBFR[0]
N1050 setal(61000)
N1060 endif
N1070 if $P_IFRAME <>
TRANS(X,11,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1080 setal(61000)
N1090 endif
N1100 if $P_IFRAME <> $P_UIFR[1]
N1110 setal(61000)
N1120 endif
N1130 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1140 setal(61001)
N1150 endif
N1160


; Transformation off
N1180 TRAFOOF
N1190
N1200 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N1210 setal(61000)
```

```
N1220 endif
N1230 if $P_BFRAME <> $P_CHBFR[0]
N1240 setal(61000)
N1250 endif
N1260 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1270 setal(61000)
N1280 endif
N1290 if $P_IFRAME <> $P_UIFR[1]
N1300 setal(61000)
N1310 endif
N1320 if $P_ACTFRAME <>
TRANS(X,21,Y,22,Z,33,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1330 setal(61002)
N1340 endif
N1350
N1360 G00 x0 y0 z0 G90
N1370
N1380 m30
```

11.5.6.5 Selecting and deselecting transformations: TRAANG



Frame expansions:

The expansions described below are only valid for the following machine data settings:

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

Components:

- Translations
  When selecting TRAANG, translations of the virtual axis are deleted.
- Rotations
  Rotations before the transformation are accepted.
- Mirrorings
  Mirrorings of the virtual axis are taken over.
- Scalings
  Scalings of the virtual axis are taken over.

Example: Machine data

```
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK = 'H1'           ; SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41'           ; Frames are active after RESET.
$MC_CHSFRAME_POWERON_MASK = 'H41'         ; Frames are deleted for "Power On".

$MN_FRAME_GEOAX_CHANGE_MODE = 1           ; Frames are calculated after switchover
                                          of the geo axis.

$MC_RESET_MODE_MASK = 'H4041'             ; Basic frame is not deselected after RE-
                                          SET.
;$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 the 1st transformation

$MC_TRAFO_TYPE_1 = 1024
```

```
$MC_TRAFO_AXES_IN_1[0] = 4                ; Inclined axis
$MC_TRAFO_AXES_IN_1[1] = 3                ; Axis is parallel to z
$MC_TRAFO_AXES_IN_1[2] = 2
$MC_TRAFO_AXES_IN_1[3] = 0
$MC_TRAFO_AXES_IN_1[4] = 0

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 3

$MC_TRAANG_ANGLE_1 = 85.
$MC_TRAANG_PARALLEL_VELO_RES_1 = 0.
$MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.

$MC_TRAANG_BASE_TOOL_1[0] = 0.0
$MC_TRAANG_BASE_TOOL_1[1] = 0.0
$MC_TRAANG_BASE_TOOL_1[2] = 0.0
```

```
; TRAANG is the 2nd transformation

$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
```

Example: Part program

```
; Frame settings
N820 $P_UIFR[1] = ctrans(x,1,y,2,z,3,b,4,c,5)
```

```
N830 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N840 $P_UIFR[1] = $P_UIFR[1] : cmirror(x,c)
N850
N860 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,b,40,c,15)
N870


; Tool selection, clamping compensation, plane selection
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900


; Approach start position
N920 G0 X20 Z10
N930
N940 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,40,C,15)
N950 setal(61000)
N960 endif
N970 if $P_BFRAME <> $P_CHBFR[0]
N980 setal(61000)
N990 endif
N1000 if $P_IFRAME <>
TRANS(X,1,Y,2,Z,3,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
N1060 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1070 setal(61000)
N1080 endif
N1090
N1100 TRAANG(,1)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1130 setal(61000)
N1140 endif
N1150 if $P_BFRAME <> $P_CHBFR[0]
N1160 setal(61000)
N1170 endif
N1180 if $P_IFRAME <>
CTRANS(X,1,Y,2,Z,3,CAX,1,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1190 setal(61000)
N1200 endif
N1210 if $P_IFRAME <> $P_UIFR[1]
```

```
N1220 setal(61000)
N1230 endif
N1240 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,CAX,11,B,44,C,20):CROT(X,10,Y,20,Z,
30):CMIRROR(X,CAX,C)
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330


; Set frame
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370


; Four-edge roughing
N1390 G1 X10 Y-10 G41 OFFN=1; allowance 1 mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440


; Tool change
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490


; Four-edge finishing
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
N1540 X10
N1550 Y-10
N1560
```

```
; Deselect frame
N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
N1670 if $P_IFRAME <>
TRANS(X,11,Y,14,Z,3,CAX,1,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1680 setal(61000)
N1690 endif
N1700 if $P_IFRAME <> $P_UIFR[1]
N1710 setal(61000)
N1720 endif
N1730 if $P_ACTFRAME <>
TRANS(X,21,Y,34,Z,33,CAX,11,B,44,C,20):CROT(X,10,Y,20,Z,
30):CMIRROR(X,CAX,C)
N1740 setal(61001)
N1750 endif
N1760
N1770 TRAFOOF
N1780
N1790 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,40,C,15)
N1800 setal(61000)
N1810 endif
N1820 if $P_BFRAME <> $P_CHBFR[0]
N1830 setal(61000)
N1840 endif
N1850 if $P_IFRAME <>
TRANS(X,1,Y,14,Z,3,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1860 setal(61000)
N1870 endif
N1880 if $P_IFRAME <> $P_UIFR[1]
N1890 setal(61000)
N1900 endif
N1910 if $P_ACTFRAME <>
TRANS(X,11,Y,34,Z,33,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1920 setal(61002)
N1930 endif
N1940
N1950 m30
```

11.5.6.6 Adapting active frames

The geometry axis constellation can change during the program processing or for `RESET`. The number of existent geometry axes varies between zero to three. For non-existent geometry axes, components in the active frames (e.g. rotations) can cause the active frames for these axis constellations to become invalid. This is indicated with the "Channel %1 block %2 rotation programmed for non-existent geometry axis" alarm. The alarm remains pending until the frames have been changed appropriately.

The automatic adaptation of active frames can be activated with the following machine data:

MD24040 $MC_FRAME_ADAPT_MODE, bit<n> = <value>

Bit	<value>	Meaning
0	1	Rotations in active frames that rotate coordinate axes for which there are no geometry axes, are deleted from the active frames.
1	1	Shearing angles in active frames are made orthogonal.
2	1	Scaling factors of all geometry axes in the active frames are set to 1.

All rotations in the active frames that could cause coordinate axis motions for non-existent geometry axes are deleted with the following machine data:

MD24040 $MC_FRAME_ADAPT_MODE = 1

The data management frames are not changed in the process. Only the possible rotations are accepted for the activation of data management frames.

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)

Program code	; Comment
`N390 G54 G0 X10 Z10 F10000`	
`IF $P_IFRAME <> CROT(Y,45)`	`; Only the rotation around Y is transferred`
`   SETAL(61000)`	
`ENDIF`	

11.5.6.7 Mapped Frames

Overview

The "mapped frames" function supports the cross-channel consistent change of axis-specific frames inside channel-specific or global data management frames. Using axis-specific machine data, it is defined between which axes mapping is realized.

If frame mapping is, e.g. active for machine axes AX1 and AX4, and the axis-specific frame of axis AX1 is changed in a channel-specific data management frame (e.g. basic frame $P_CHBFR[x]) (translation, fine translation, scaling, mirroring), then this frame data for AX1 and AX4 is transferred to all channel-specific data management frames (e.g. basic frame $P_CHBFR[x]) in all channels in which they are parameterized as channel axes.

Frame mapping is not realized when changing the axis-specific frame data for the rotation.

Requirements

The following requirements must be fulfilled for frame mapping:

- The data management frames used for mapping must be configured:
  MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK (system frames)
- **Channel-specific** data management frames must be explicitly enabled for mapping:
  MD10616 $MN_MAPPED_FRAME_MASK (enable frame mapping)

  Note
  For **global** data management frames, mapping is always carried out. An enable is not required.

Parameterization

The parameterization of the mapping relationships is realized in the axis-specific machine data:

MD32075 $MA_MAPPED_FRAME[<AXn>] = "AXm"

AXn, AXm: Machine axis name with n, m = 1, 2, ... max. number of machine axes

Mapping rules

The following rules apply for frame mapping:

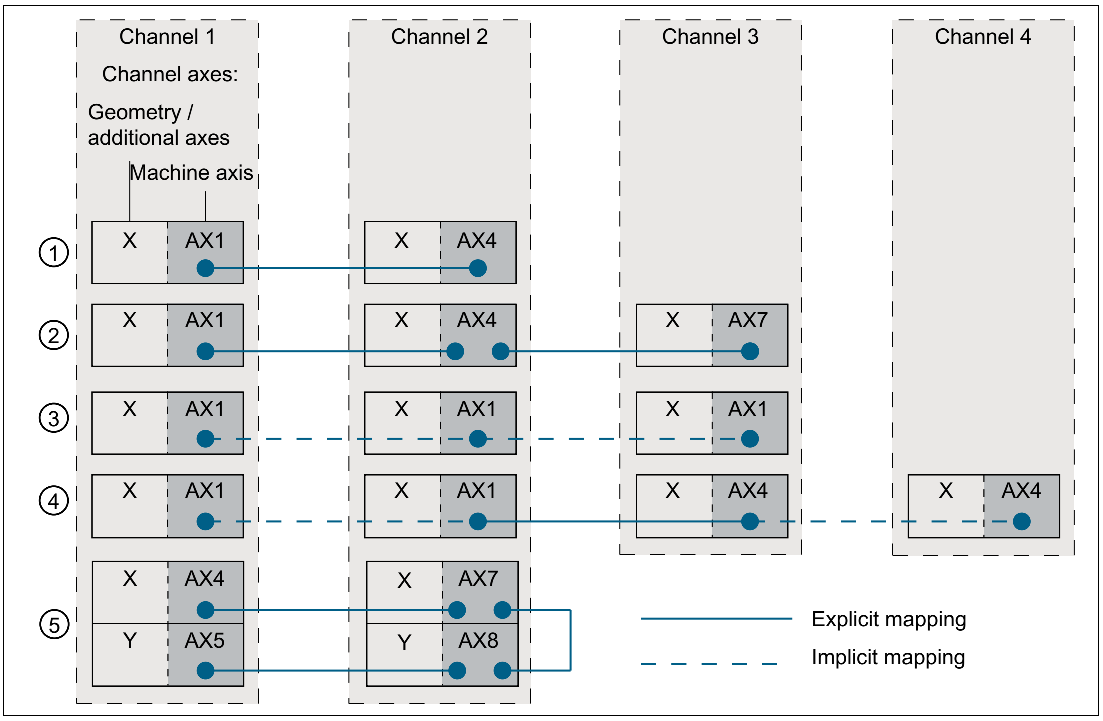
- The mapping is bidirectional.
  An axis-specific frame can be written for axis AXn or AXm. The frame data is always accepted and taken for the other axis.
- All parameterized mapping relationships are always evaluated.
  When writing an axis-specific frame of axes AXn, all mapping relationships are evaluated and the frame data accepted for all directly and indirectly involved axes.
- The mapping is global across all channels.
  When writing an axis-specific frame of axis AXn or AXm for a channel-specific frame, the frame data is accepted for all channels in which AXn or AXm are parameterized as channel axes.

- When writing an axis-specific frame using geometry or special axis name, the mapping relationships are evaluated via the machine axes currently assigned to the geometry or special axis.
- Mapping is frame-specific.
  When writing an axis-specific frame, the frame data is only mapped within the same channel-specific or global data management frame.

Note

Data consistency

The user / machine manufacturer is solely responsible for ensuring that after a frame is written, consistent frame data is available in all channels, e.g. by using channel synchronization.



	Description	Parameterization: $MA_
①	Simple mapping relationship: AX1(K1) ↔ AX4(K2)	MAPPED_FRAME[<AX1>] = "AX4"
②	Chained mapping relationships: AX1(K1) ↔ AX4(K2) ↔ AX7(K3)	MAPPED_FRAME[<AX1>] = "AX4" MAPPED_FRAME[<AX4>] = "AX7"
③	Mapping relationship to itself, with AX1 as channel axis of channels 1, 2 and 3: AX1(K1+K2+K3)	MAPPED_FRAME[<AX1>] = "AX1"
④	Mapping relationship between two axes, the channel axes in two channels are: AX1(K1+K2) ↔ AX4(K3+K4)	MAPPED_FRAME[<AX1>] = "AX4"
⑤	Chained mapping relationships where multiple channel axes can be written in the same channel: AX4(K1) ↔ AX7(K2) ↔ AX8(K2) ↔ AX5(K1)	MAPPED_FRAME[<AX4>] = "AX7" MAPPED_FRAME[<AX7>] = "AX8" MAPPED_FRAME[<AX8>] = "AX5"

Figure 11-25 Mapping examples

Activating the data management frame

Data management frames can be written in the part program and via the user interface of SINUMERIK Operate. The following should be noted when activating the data management frames in the channels written directly and via frame mapping:

- Writing in the part program
  The data management frames must be explicitly activated in each channel (`G500`, `G54` ... `G599`)
- Writing via user interface
  Data management frames are written via the user interface, e.g. by entering new work offsets. A modified data management frame is immediately active in all of the involved channels if none of these channels is in the "Channel active" state. The data management frame is not active in any channel if one of the channels involved is in the "Channel active" state. The activation must then be explicitly programmed in each channel in the part program (`G500`, `G54` ... `G599`). Or, the next time that the channel state changes, it becomes active after "Channel reset".

Example

The following channels and channel axes are parameterized at a control:

- Channel 1
  - Z: Geometry axis
  - AX1: Machine axis of geometry axis Z
- Channel 2
  - Z: Geometry axis
  - AX4: Machine axis of geometry axis Z

The zero point of the Z axis should always be the same in both channels:

- Mapping relationship: $MA_MAPPED_FRAME[AX1] = "AX4"

Part programs in channel 1 and 2

Channel 1	Channel 2
`...`	`...`
`N100 WAIT (10,1,2)`	`N200 WAIT (10,1,2)`
`N110 $P_UIFR[1] = CTRANS(Z, 10)`	
`N120 WAIT (20,1,2)`	`N220 WAIT (20,1,2)`
`N130 G54`	`N230 G54`
`N140 IF ($P_IFRAME[0, Z, TR] <> 10)`	`N230 IF ($P_IFRAME[0, Z, TR] <> 10)`
`N150   SETAL(61000)`	`N250   SETAL(61000)`
`N160 ENDIF`	`N260 ENDIF`
`...`	`...`

Description:

N100 / N200	Channel synchronization for consistent writing and mapping of frame data

N110	Writing of the settable data management frame $P_UIFR[1]: Moving the zero point of the Z axis to 10 mm Mapping the axis-specific frame data: Channel1: Z ≙ AX1 ⇔ channel 2: Z ≙ AX4
N120 / N220	Channel synchronization for consistent activation of new frame data
N130 / N230	Activating the new frame data
N140 / N240	Checking the zero point of the Z axis for: 10 mm

11.5.7 Predefined frame functions

11.5.7.1 Inverse frame

The INVFRAME() function calculates the appropriate inverse frame from a frame.

Function description

The chaining between a frame and its inverse frame always produces a zero frame.
`FRAME: INVFRAME(FRAME) □ null 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, then 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
  ```
- The target frame is the nth channel basic frame $P_CHBFRAME[<n>]:
  ```
  k = $MN_MM_NUM_GLOBAL_BASE_FRAMES
  ```
  - For n = 0, TMP is resolved as:
    ```
    TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
    $P_NCBFRAME[<0...k>]
    ```
  - For n ≠ 0, TMP is resolved as:
    ```
    TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
    $P_NCBFRAME[<0..k>] : $P_CHBFRAME[<0...n> - 1]
    ```

  ```
  $P_CHBFRAME[<n>] = INVFRAME(TMP) : $P_ACTFRAME : $AC_MEAS_FRAME :
  INVFRAME($P_ACTFRAME) : TMP : $P_CHBFRAME[<n>]
  ```
- The 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
  ```

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.

Program code	**Comment**
`DEF INT RETVAL`	
`DEF FRAME TMP`	
`$TC_DP1[1,1]=120`	`; Type`
`$TC_DP2[1,1]=20.`	`; 0`
`$TC_DP3[1,1]= 10.`	`; (z) length correction vector`
`$TC_DP4[1,1]= 0.`	`; (y)`
`$TC_DP5[1,1]= 0.`	`; (x)`
`$TC_DP6[1,1]= 2.`	`; Radius`
`T1 D1`	
`g0 x0 y0 z0 f10000`	
`G54`	
`$P_CHBFRAME[0] = CROT(Z,45)`	
`$P_IFRAME[X,TR] = -SIN(45)`	
`$P_IFRAME[Y,TR] = -SIN(45)`	
`$P_PFRAME[Z,TR] = -45`	
`$AC_MEAS_VALID = 0`	`; Measure corner with four measuring points`
`G1 X-1 Y-3`	`; 1st Approach measuring point`
`$AC_MEAS_LATCH[0] = 1`	`; 1st Store measuring point`
`G1 X5 Y-3`	`; 2. Approach measuring point`
`$AC_MEAS_LATCH[1] = 1`	`; 2. Store measuring point`
`G1 X-4 Y4`	`; 3. Approach measuring point`
`$AC_MEAS_LATCH[2] = 1`	`; 3. Store measuring point`
`G1 X-4 Y1`	`; 4. Approach measuring point`
`$AC_MEAS_LATCH[3] = 1`	`; 4. Store measuring point`
`$AA_MEAS_SETPOINT[X] = 0`	`; Set position setpoint of the corner`
`$AA_MEAS_SETPOINT[Y] = 0`	
`$AA_MEAS_SETPOINT[Z] = 0`	
`$AC_MEAS_CORNER_SETANGLE = 90`	`; Define setpoint angle of intersection`
`$AC_MEAS_WP_SETANGLE = 30`	
`$AC_MEAS_ACT_PLANE = 0`	`; Measuring plane is G17`
`$AC_MEAS_T_NUMBER = 1`	`; Select tool`
`$AC_MEAS_D_NUMBER = 1`	
`$AC_MEAS_TYPE = 4`	`; Set measuring type on corner 1`
`RETVAL = MEASURE()`	`; Start measuring process`
`IF RETVAL <> 0`	
`   SETAL(61000 + RETVAL)`	
`ENDIF`	
`IF $AC_MEAS_WP_ANGLE <> 30`	
`   SETAL(61000 + $AC_MEAS_WP_ANGLE)`	

Program code	Comment
`ENDIF`	
`IF $AC_MEAS_CORNER_ANGLE <> 90`	
`   SETAL(61000 + $AC_MEAS_CORNER_ANGLE)`	
`ENDIF`	
`; Transform measured frame and write to $P_SETFRAME so that a complete frame is created,`	
`; which linked from the old total frame results in the measurement frame.`	
`$P_SETFRAME = $P_ACTFRAME: $AC_MEAS_FRAME: INVFRAME($P_ACTFRAME): $P_SETFRAME`	
`$P_SETFR = $P_SETFRAME`	`; Describe system frames in data management`
`G1 X0 Y0`	`; Approach the corner`
`G1 X10`	`; Retract the rectangle rotated about 30 degrees`
`Y10`	
`X0`	
`Y0`	
`M30`	

11.5.7.2 Additive frame in frame chain

Measurements on the workpiece or calculations in the part program or cycle often produce a frame that is applied additively to the active complete frame. This causes, for example, the WCS and thus the programming zero point to be displaced and/or possibly rotated. The measured frame so exists in a frame variable and has not yet been included in the frame chain calculation.

Function description

The `ADDFRAME()` function calculates from the temporary frame the specified target frame specified by the `<STRING>` parameter so that the new active $P_ACTFRAME complete frame from the chaining of the old active complete frame with the temporary frame gives:

`ERG = ADDFRAME(TMPFRAME,"$P_SETFRAME")` ⇒ $\$P\_SETFRAME_{new}$ = $\$P\_SETFRAME_{old}$ ADD TMPFRAME and $\$P\_ACTFRAME_{new}$ = $\$P\_ACTFRAME_{old}$: TMPFRAME

If an active frame has been specified as a target frame, the new complete frame becomes active at the **preprocessing stage**.

If the target frame is a data management frame, the frame becomes active only when it is activated explicitly in the channel, e.g. part program or cycle.

The function returns a return value for which a user-specific response is possible, e.g. in a user cycle.

Programming

Syntax

```
INT ADDFRAME(<FRAME>,<STRING>)
```

Meaning

`<FRAME>`:	Frame variable with the values to be included additively in the calculation	
	Type	FRAME
`<STRING>`:	The name of an active or data management frame: • Active frames "$P_CYCFRAME", "$P_ISO4FRAME", "$P_PFRAME", "$P_WPFRAME", "$P_TOOLFRAME", "$P_IFRAME", "$P_GFRAME", "$P_CHBFRAME[<n>]", "$P_NCBFRAME[<n>]", "$P_ISO1FRAME", "$P_ISO2FRAME", "$P_ISO3FRAME", "$P_EXTFRAME", "$P_SETFRAME", "$P_PARTFRAME" • Data management frames "$P_CYCFR", "$P_ISO4FR, "$P_TRAFR", "$P_WPFR", "$P_TOOLFR", "$P_UIFR[<n>]", "$P_GFR", "$P_CHBFR[<n>]", "$P_NCBFR[<n>]", "$P_ISO1FR, "$P_ISO2FR, "$P_ISO3FR, "$P_EXTFR", "$P_SETFR", "$P_PARTFR"	
	Type	STRING
Return value:	Possible return values: • 0: OK • 1: Specified target (string) is wrong • 2: Target frame is not configured • 3: Rotation in frame is not permitted	
	Type	INT

11.5.8 Functions

11.5.8.1 Setting zeros, workpiece measuring and tool measuring

Set actual value is initiated by means of an HMI operator action or via measuring cycles. The calculated frame can be written to system frame `SETFRAME`. The position setpoint of an axis in the WCS can be altered during set actual value.

The term "scratching" refers to both the workpiece and tool measurement. The position of the workpiece can be measured in relation to an edge, a corner or a hole. To determine the zero position of the workpiece or the hole, position setpoints can be added to the measured positions in the WCS. The resulting offsets can be entered in a selected frame. In the tool measurement, the length or radius of a tool can be measured using a measured reference part.

The measurements can be initiated by means of an operator action or via measuring cycles. Communication with the NC takes place via predefined system variables. The calculation is performed in the NC when a PI service is activated by means of an HMI operator action or via 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.

11.5.8.2 Axis-specific external work offset

Machine data

The external work offset or the $P_EXTFRAME system frame is activated with the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 1 = TRUE

The magnitude for the external work offset can be specified manually via the HMI user interface, and the PLC user program via OPI or programmed in the part program via the axial system variable $AA_ETRANS[<axis>].

Activation
The external work offset is activated with the interface signal:
DB31, ... DBX3.0 (accept external work offset)

Behavior
Upon activation of the external work offset, the traversing movements of all axes, except command and PLC axes, are stopped immediately and the advance is reorganized. The rough offset of the current system frame and of the system frame in data management is set to the value of the axial system variable $AA_ETRANS[<axis>]. Thereafter, the offset is traversed first and then the interrupted movement is continued.

Behavior for incremental dimension
In case of active incremental dimension `G91` and machine data:
MD42440 $MC_FRAME_OFFSET_INCR_PROG (work offset in frames) = 0
traversing the offset is done in the scope of the external work offset via system frame, despite opposite configuring of the machine data, with the approach block, although it is specified by a frame.

Note

The external work offset always acts absolutely.

11.5.8.3 Toolholder

Translations

With kinematics of type "P" and "M", the selection of a toolholder activates an additive frame (table offset of the orientable toolholder) which takes into account the work offset as a result of the rotation of the table. The work offset is entered in the system frame `$P_PARTFR`. In this case the translatory component of this frame is overwritten. The other frame components are retained.

The system frame `$P_PARTFR` must be enabled via the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 2 = 1 (system frame for TCARR and PAROT)

Note

Alternatively, the offset can also parameterized via machine data to record the table offset:

MD20184 $MC_TOCARR_BASE_FRAME_NUMBER = <number of the basic frame>

This option is only for compatibility reasons to older software versions. You are strongly recommended **not** to use this procedure any longer.

A frame offset as a result of a toolholder change becomes effective immediately on selection of `TCARR=...`. A change in the tool length, on the other hand, only becomes effective immediately if a tool is active.

A frame rotation is not executed with the activation, or a rotation which is already active is not changed. As in case T (only the tool can be rotated), the position of the rotary axes used for the calculation is dependent on the command `TCOFR`/`TCOABS` and determined from the rotation component of an active frame or from the $TC_CARR entries. Activation of a frame changes the position in the workpiece coordinate system accordingly, without compensating motion by the machine itself.

The ratios are shown in the figure below:

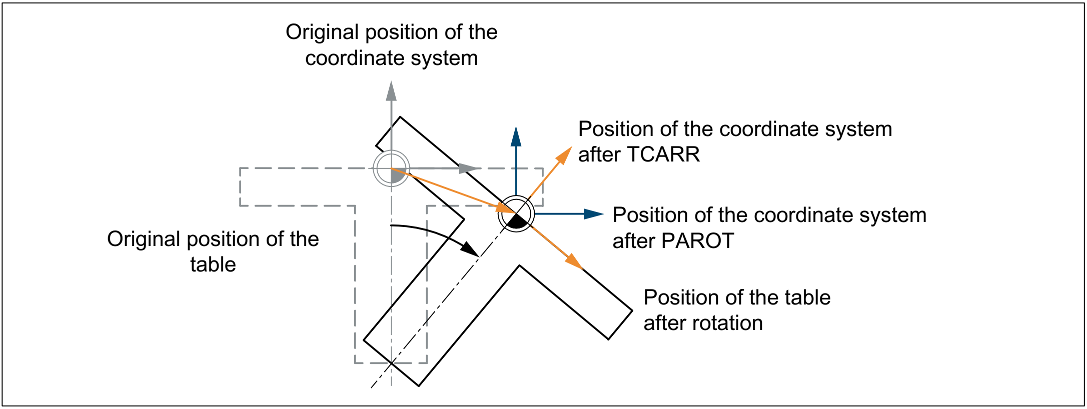

Original position of the coordinate system
Position of the coordinate system after TCARR
Position of the coordinate system after PAROT
Original position of the table
Position of the table after rotation

Figure 11-26 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 work offset.

Rotations

Depending on the machining task, it is necessary to take into account not only a work offset (whether as frame or as tool length) when using a rotary toolholder or table, but also a rotation. However, the activation of an orientable 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 orientable toolholder corresponding to the table. The entire programmable frame including its rotation component remains unchanged.

The rotation component that describes the rotation of the table is then either entered in the system frame `$PARTFR` or in the basic frame parameterized with MD20184 $MC_TOCARR_BASE_FRAME_NUMBER:

$MC_MM_SYSTEM_FRAME_MASK, bit 2 = <value>

Value	Meaning
1	Rotation component → $PARTFR
0	Rotation component → MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

As with the note made in the description of the table offset, the second alternative here is not recommended for use with new systems.

The rotation component of the part frame can be deleted with `PAROTOF`, irrespective of whether this frame is in a basic or a system frame.
The translation component is deleted when a toolholder which does not produce an offset is activated or a possibly active orientable toolholder is deselected with `TCARR=0`.

`PAROT` or `TOROT` take into account the overall orientation change in cases where the table or the tool are oriented with two rotary axes. With mixed kinematics, only the corresponding component caused by a rotary axis is considered. It is thus possible, for example, when using `TOROT`, to rotate a workpiece such that an inclined plane lies parallel to the XY plane fixed in space, whereby rotation of the tool must be taken into account in machining where any holes to be drilled, for example, are not perpendicular to this plane.

Example

On a machine, the rotary axis of the table points in the positive Y direction. The table is rotated by +45 degrees. `PAROT` defines a frame which similarly describes a rotation of 45 degrees around the Y axis. The coordinate system is not rotated relative to the actual environment (marked in the figure with "Position of the coordinate system after TCARR"), but is rotated by -45 degrees relative to the defined coordinate system (position after PAROT). If this coordinate system is defined with `ROT Y-45`, for example, and if the toolholder is then selected with active `TCOFR`, an angle of +45 degrees will be determined for the rotary axis of the toolholder.

Language command `PAROT` is not rejected if no orientable toolholder is active. However, such a call then produces no frame changes.

Machining in direction of tool orientation

Particularly on machines with tools that can be oriented, traversing should take place in the tool direction (typically, when drilling) without activating a frame (e.g. using `TOFRAME` or `TOROT`), on which one of the axes points in the direction of the tool. This is also a problem if, when carrying out inclined machining operations, a frame defining the inclined plane is active, but the tool cannot be set exactly perpendicularly because an indexed toolholder (Hirth tooth system) prevents free setting of the tool orientation. In these cases it is then necessary - contrary to the motion actually requested perpendicular to the plane - to drill in the tool direction, as the drill would otherwise not be guided in the direction of its longitudinal axis (tool breaks).

Incremental traversing

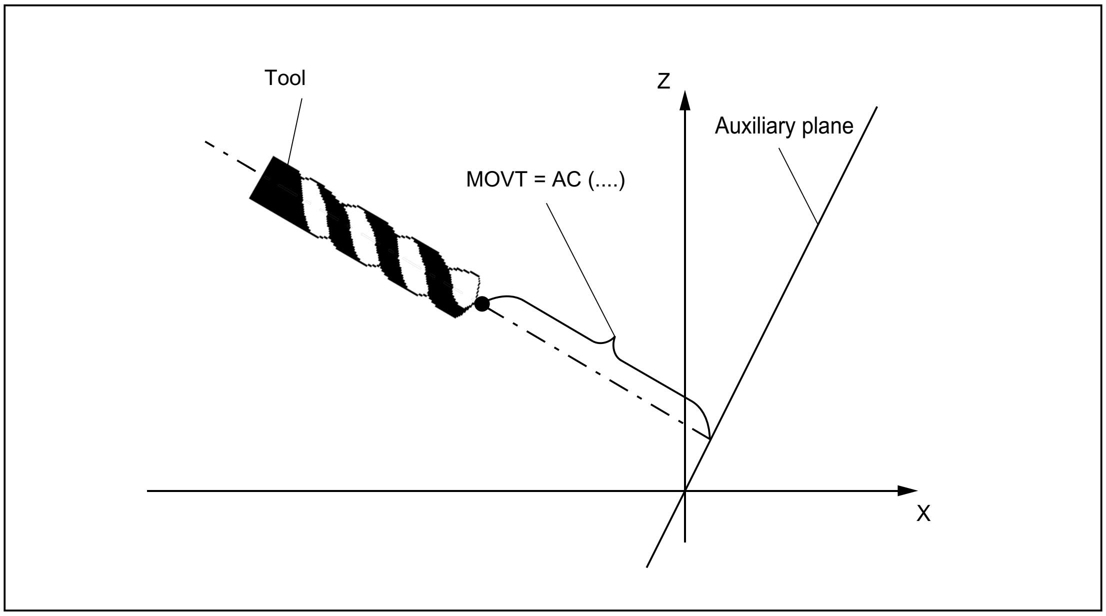
The end point for incremental traversing motion in the tool direction is programmed using `MOVT = <Value>` or `MOVT=IC(<Value>).`

The positive traversing direction is defined from the tool tip to the tool adapter. Corresponding to paraxial machining, e.g. with `G91 Z ....`

Absolute traversing

The end point for absolute incremental traversing motion in the tool direction is programmed using `MOVT=AC(<Value>).`

In this case a plane is defined, which runs through the current zero point, and whose surface normal vector is parallel to the tool orientation. `MOVT` then gives the position relative to this plane (see figure). The reference plane is only used to calculate the end position. Active frames are not affected by this internal calculation.



Programming with `MOVT` is independent of the existence of a toolholder that can be oriented. The direction of the motion is dependent on the active plane.
It runs in the directions of the vertical axes, i.e. with `G17` in Z direction, with `G18` in Y direction and with `G19` in X direction. This applies when no orientable toolholder is active and for the case of an orientable toolholder without rotary tool or with a rotary tool in its basic setting.

`MOVT` acts similarly for active orientation transformation (3-, 4-, 5-axis transformation).

If in a block with `MOVT` the tool orientation is changed simultaneously (e.g. active 5-axis transformation by means of simultaneous interpolation of the rotary axes), the orientation at the start of the block is decisive for the direction of motion of `MOVT`.
With an active 5-axis transformation, the path of the tool center point (TCP) is not affected by the change of orientation, i.e. the path remains a straight line and its direction is determined by the tool orientation at the start of the block.

If `MOVT` is programmed, linear or spline interpolation must be active (`G0,G1`, `ASPLINE`, `BSPLINE`, `CSPLINE`). Otherwise, an alarm is produced.
If a spline interpolation is active, the resultant path is generally not a straight line, since the end point calculated by `MOVT` is treated as if it had been programmed explicitly with X, Y, Z.

It is not permissible to program geometry axes in a block with `MOVT`.

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. The subsequent rotation is realized in the newly rotated coordinate system.

This applies both to programing in one block - and when configuring a frame in several consecutive blocks:

- One block: `N10 ROT X... Y... Z...`
- Several consecutive blocks:
```
N10 ROT Y...
N20 AROT X...
N30 AROT Z...
```

Solid angle

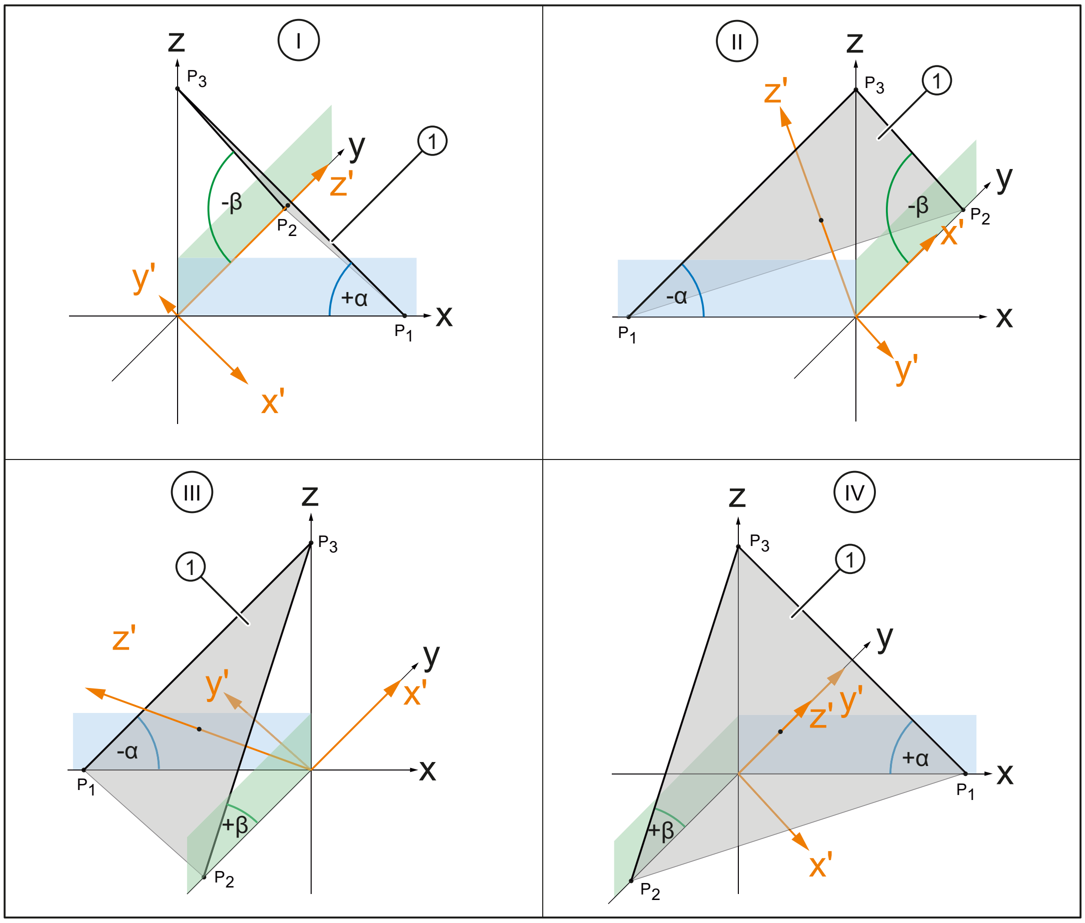
Often, solid angles are specified in workpiece drawings to define inclined surfaces. Solid angles are angles which the intersection lines of the inclined plane form with the main planes (X-Y, Y-Z, Z-X planes) of the workpiece coordinate system (see diagram). The orientation of a plane in space is defined unambiguously by specifying two solid angles. The third solid angle is derived from the first two.

Rotations can be directly defined as solid angle using `ROTS`, `AROTS` and `CROTS` commands.

It is permissible to specify a single solid angle. The rotation which is performed with `ROTS` or `AROTS` in this case is identical to that for `ROT` and `AROT`.

The two axes programmed in the command define a plane. The non-programmed axis defines the associated third axis of a coordinate system. This also means that for the two programmed axes, it is uniquely defined which is the first axis and which is the second axis. The definition corresponds to the definition of a plane for `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`).



I, ..., IV	Quadrants 1 to 4
①	Inclined plane as specification for the new G17 plane
α, β	Solid angle of the inclined plane

Figure 11-27 Rotation around the solid angle

In the diagram, the solid angles are shown for an example of a plane in quadrants I to IV. The inclined plane defines the alignment of the G17 plane after rotating the workpiece coordinate system WCS. The sign of the solid angle specifies the direction around which the coordinate system is rotated around the relevant axis.

1. Rotation around y:
   Rotation of the workpiece coordinate system WCS around the y axis around the signed angle α ⇒
   The x' axis is aligned parallel (colinear) to the intersection lines of the xz plane with the inclined plane
2. Rotation around x':
   Rotation of the new workpiece coordinate system WCS around the x' axis around the signed angle β ⇒
   – The y' axis is aligned in parallel (colinear) to the intersection lines of the zy plane with the inclined plane
   – The z' axis is perpendicular to the inclined plane
   – G17' lies in parallel to the inclined plane

The programming to align the G17 plane of the workpiece coordinate system WCS to the inclined plane is, for each quadrant:

- Quadrant I: `ROTS X<+α> Y<-β>`
- Quadrant II: `ROTS X<-α> Y<-β>`
- Quadrant III: `ROTS X<-α> Y<+β>`
- Quadrant IV: `ROTS X<+α> Y<+β>`

Orientation

The specification of the solid angle does not define the orientation of the two-dimensional coordinate system within the plane (i.e. the angle of rotation around the surface normal vector). The position of the coordinate system is thus determined so that the rotated first axis lies in the plane which is formed by the first and third axes of the non-rotated coordinate system.

This means that

- When programming X and Y, the new X axis lies in the original Z-X plane.
- When programming Z and X, the new Z axis lies in the original Y-Z plane.
- When programming Y and Z, the new Y axis lies in the original X-Y plane.

If the required coordinate system does not correspond to this basic setting, then an additional rotation must be performed with `AROT....`

Parameterization: zy'x" (RPY angle) or zx'z" convention

The programmed solid angles are converted to the equivalent Euler angles according to the the zy'x" convention (RPY angle) or zx'z" convention when entering depending on the following machine data:

MD10600 $MN_FRAME_ANGLE_INPUT_MODE

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 work 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 work 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 orientable toolholder. This language command is also especially useful for 5-axis transformations.

The new language command TOROT ensures consistent programming with active orientable toolholders for each kinematics type.

TOFRAME or TOROT defines frames whose Z direction points in the tool direction. This definition is suitable for milling, where G17 is usually active. However, particularly with turning or, more generally, when G18 or G19 is active, it is desirable that frames which will be aligned on the X or Y axis, can be defined. For this purpose, the following commands are available in G group 53:

- TOFRAMEX, TOFRAMEY, TOFRAMEZ
- TOROTX, TOROTY, TOROTZ

The appropriate frames can be defined with these commands. The functions of TOFRAME and TOFRAMEZ or of TOROT and TOROTZ are identical to one another.

The frames resulting from TOROT or TOFRAME can be written in a separate system frame ($P_TOOLFR). The programmable frame is then retained unchanged.

- Requirement: MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 3 = 1

When programming TOROT or TOFRAME, etc. response is identical, with or without a system frame. Differences occur when the programmable frame is processed further elsewhere.

Note

In new systems, it is recommended that only the intended system frame is used for frames produced by the commands of G group 53.

Example

TRANS is programmed after TOROT. TRANS without specified parameters deletes the programmable frame. In the variant without a system frame, this also deletes the frame component of the programmable frame produced by TOROT, but if the TOROT component is in the system frame, it is retained.

TOROT or TOFRAME, etc. are disabled with language command TOROTOF. TOROTOF deletes the entire system frame $P_TOOLFR. If the programmable frame (old variant) and not the system frame is described by commands TOFRAME, etc. TOROT only deletes the rotation component and leaves the remaining frame components unchanged.

If a rotating frame is already active before language command TOFRAME or TOROT is activated, a request is often made that the newly defined frame should deviate as little as possible from the old frame. This is the case, for example, if a frame definition needs to be modified slightly because the tool orientation cannot be set freely on account of Hirth-toothed rotary axes. The language commands uniquely define the Z direction of the new frame.

Parameterization: Frame definition for TOFRAME, TOROT and PAROT (SD42980)

With the following setting data, the direction of the geometry axes of the actual machining plane (G17: X and Y axis) for the frame definition is defined using `TOFRAME`, `TOROT` (`TOROTY`, `TOROTX`) or `PAROT`.

For a frame calculation, the tool direction is defined, so that the tool direction and applicate (G17: Z axis) of the frame are parallel and perpendicular to the machining plane.

Initially, rotation around the tool axis is arbitrary. Using the setting data, this free rotation can be defined so that the newly defined frame deviates as little as possible from a previously active frame.

In all cases where the setting data is not equal to zero, an active frame remains unchanged if the tool direction of the old and the new frame match one another.

SD42980 $SC_TOFRAME_MODE

TCARR (request toolholder) and PAROT (align workpiece coordinate system on the workpiece)

`TCARR` uses the basic frame identified by following machine data:
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER.

A system frame can be created for `TCARR` and `PAROT` alone, in order to avoid conflicts with systems, which already use all the basic frames.

`PAROT, TOROT` and `TOFRAME` have previously changed the rotation component of the programmable frames. In this case, it is not possible to shut down `PAROT` or `TOROT` separately. On `RESET`, the programmable frame is deleted, which means that after changing the operating mode to JOG, the rotation component of `PAROT` and `TOROT` is no longer available. The user must also have unrestricted access to the programmable frame. Frames produced by `PAROT` and `TOROT` must be able to be archived and reloaded via data backup.

The system frame for `TCARR` and `PAROT` is configured with:
MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 2 = 1

The following machine data is then no longer evaluated:
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

If the system frame for `TCARR` is configured, `TCARR` and `PAROT` describe that corresponding system frame; otherwise the basic frame identified by machine data MD20184 is described.

With kinematics systems of the types P and M, `TCARR` will enter the table offset of the orientable toolholder (work offset resulting from the rotation of the table) as a translation into the system frame. `PAROT` converts the system frame so that a workpiece-related WCS results.

The system frames are stored retentively and therefore retained after a reset. The system frames also remain active in the case of a mode change.

For the display, the commands `PAROT` and `TOROT, TOFRAME` are each assigned to a separate G group.

PAROTOF

PAROTOF is the switch off command for PAROT. This command deletes the rotations in the system frame for PAROT. In so doing, the rotations in the current $P_PARTFRAME and in the data management frame $P_PARTFR are deleted. The position of the coordinate system is then recreated according to TCARR. PAROTOF is in the same G group as PAROT and appears therefore in the G command display.

TOROT (align Z axis of the WCS by rotating the frame parallel to the tool orientation) and TOFRAME (ditto.)

The system frame for TOROT and TOFRAME is activated via the following machine data:
MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 3 = 1

This system frame is located before the programmable frame in the frame chain.
The SZS coordinate system is located before the programmable frame.

TOROTOF

TOROTOF is the switch off command for TOROT and TOFRAME. This command deletes the corresponding system frame. The current $P_TOOLFRAME and the data management frame $P_TOOLFR are also deleted. TOROTOF is in the same G group as TOROT and TOFRAME and appears therefore in the G command display.

Example

Use of an orientable toolholder with resolved kinematics.

```
N10 $TC_DP1[1,1]=120
N20 $TC_DP3[1,1]= 13                ; Tool length 13 mm
; Definition of toolholder 1:
N30 $TC_CARR1[1] = 0                ; X component of the 1st offset vector
N40 $TC_CARR2[1] = 0                ; Y component of the 1st offset vector
N50 $TC_CARR3[1] = 0                ; Z component of the 1st offset vector
N60 $TC_CARR4[1] = 0                ; X component of the 2nd offset vector
N70 $TC_CARR5[1] = 0                ; Y component of the 2nd offset vector
N80 $TC_CARR6[1] = -15              ; Z component of the 2nd offset vector
N90 $TC_CARR7[1] = 1                ; X component of the 1st axis
N100 $TC_CARR8[1] = 0               ; Y component of the 1st axis
N110 $TC_CARR9[1] = 0               ; Z component of the 1st axis
N120 $TC_CARR10[1] = 0              ; X component of the 2nd axis
N130 $TC_CARR11[1] = 1              ; Y component of the 2nd axis
N140 $TC_CARR12[1] = 0              ; Z component of the 2nd axis
N150 $TC_CARR13[1] = 30             ; Angle of rotation of 1st axis
N160 $TC_CARR14[1] = -30            ; Angle of rotation of 2nd axis
N170 $TC_CARR15[1] = 0              ; X component of the 3rd offset vector
N180 $TC_CARR16[1] = 0              ; Y component of the 3rd offset vector
N190 $TC_CARR17[1] = 0              ; Z component of the 3rd offset vector
N200 $TC_CARR18[1] = 0              ; X component of the 4th offset vector
```

```
N210 $TC_CARR19[1] = 0                ; Y component of the 4th offset vector
N220 $TC_CARR20[1] = 15               ; Z component of the 4th offset vector
N230 $TC_CARR21[1] = A                ; Reference for 1st axis
N240 $TC_CARR22[1] = B                ; Reference for 2nd axis
N250 $TC_CARR23[1] = "M"              ; Toolholder type
N260 X0 Y0 Z0 A0 B45 F2000
N270 TCARR=1 X0 Y10 Z0 T1 TCOABS      ; Selection of orientable tool holder
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
N400 M30
```

11.5.9 Subprograms with SAVE attribute (SAVE)

For various frames, the behavior regarding subprograms can be set using the SAVE attribute.

Settable frames G54 to G599

The behavior of the adjustable frames can be set using MD10617 $MN_FRAME_SAVE_MASK.**BIT0** :

- BIT0 = 0
  If using the subprogram, only the values of the active adjustable frame are changed using the system variable $P_IFRAME, but the G command is kept, then the change is also kept after the end of the subprogram.
- BIT0 = 1
  With the end of the subprogram, the adjustable frame, G command and values, active before the subprogram call, are reactivated.

Basic frames $P_CHBFR[ ] and $P_NCBFR[ ]

The behavior of the basic frame can be set using MD10617 $MN_FRAME_SAVE_MASK.**BIT1**:

- BIT1 = 0
  If the active basic frame is changed by the subprogram, the change remains effective even after the end of the subprogram.
- BIT1 = 1
  With the end of the subprogram the basic frame which is active before the subprogram call is reactivated.

Programmable frame

With the end of the subprogram the programmable frame active before the subprogram call is reactivated.

System frames

If the system frames are changed by the subprogram, the change remains effective even after the end of the subprogram.

11.5.10 Data backup

Data blocks

The data backup of the frames is performed in the following data blocks:

- Channel-specific frames ⇒ _N_CHAN<x>_UFR data blocks
- Global frames ⇒ _N_NC_UFR data block

Note

- It is strongly recommended that the following machine data is not changed between the backup and restoration of the saved system frames. Otherwise, it is possible that the saved system frames cannot be loaded.
  MD28082 $MC_MM_SYSTEM_FRAME_MASK (configuration of channel-specific system frames that are included in the channel calculation)
- The data backup is always performed in accordance with geometry axis assignments currently valid in the channel rather than in accordance with the axis constellations originally set in the machine data.

Data backup of system frames

Only the system frames created in the data management are considered for the data backup of the system frames. These system frames were selected during the commissioning of the control with the following machine data:

MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK

Frames not created in the data management are not backed up.

Data backup of NC-global frames

A data backup of NC-global frames is performed only when at least one NC-global frame has been parameterized in one of the following machine data items:

- MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES
- MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES
- MD18603 $MN_MM_NUM_GLOBAL_G_FRAMES

11.5.11 Positions in the coordinate system

The current setpoint positions in the coordinate systems 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 an Actual value display softkey in the MCS/WCS. The machine manufacturer can specify on the PLC which coordinate system corresponds to the workpiece coordinate system on a machine. The HMI requests the corresponding actual values form the NC.

$AA_IM[axis]

The setpoints in the machine coordinate system can be read for each axis with the $AA_IM[axis] variable.

$AA_IEN[axis]

The setpoints in the settable zero system (SZS) can be read for each axis with the $AA_IEN[axis] variable.

$AA_IBN[axis]

The setpoints in the basic zero system (BZS) can be read with the $AA_IBN[axis] variable.

$AA_IW[axis]

The setpoints in the workpiece coordinate system (WCS) can be read with the $AA_IW[axis] variable.

11.5.12 Control system response

11.5.12.1 POWER ON

Frame	Frame conditions after POWER ON
$P_PFRAME programmable frame	Deleted.
$P_IFRAME settable frame	Is retained, depending on: • MD24080 $MC_USER_FRAME_POWERON_MASK, bit 0 • MD20152 $MC_GCODE_RESET_MODE[7]
$P_GFRAME grinding frame	Is retained, depending on: • MD24080 $MC_USER_FRAME_POWERON_MASK, bit 0 • MD20152 $MC_GCODE_RESET_MODE[63]
$P_ACTBFRAME complete basic frame	Is retained, depending on: • MD20110 $MC_RESET_MODE_MASK bit 0 and bit 14 Basic frames can be deleted via machine data: • MD10615 $MN_NCBFRAME_POWERON_MASK • MD24004 $MC_CHBFRAME_POWERON_MASK

Frame	Frame conditions after POWER ON
System frames: $P_PARTFRAME, $P_SET-FRAME, $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_TOOL-FRAME, $P_WPFRAME, $P_TRA-FRAME, $P_ISO4FRAME, $P_RE-LFRAME, $P_CACFRAME	Retained Individual system frames can be deleted via machine data: • MD24008 $MC_CHSFRAME_POWERON_MASK Deletion of system frames is executed in the data management on first priority.
$P_EXTFRAME external work off-set	Is retained, but has to be activated again.
DRF offset	Deleted.

11.5.12.2 Mode change

System frames

The system frames are retained and remain active.

JOG mode

In JOG mode, the frame components of the current frame are only taken into account for the geometry axes if a rotation is active. No other axis-specific frames are taken into account.

PLC and command axes

The response for PLC and command axes can be set via machine data:

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED (frame or HL correction is not permissible)

11.5.12.3 Channel reset / part program end

Reset behavior of the basic frames

The reset behavior of the basic frames is set via the machine data:

MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after channel reset / part program end)

Reset behavior of system frames

The system frames are retained in the data management after a channel reset / part program end.

The machine data below can be used to configure the activation of individual system frames:

MD24006 $MC_CHSFRAME_RESET_MASK, bit<n> = <VALUE> (active system frames after channel-reset / part program end)

Bit	Value	Meaning
0	1	System frame for actual value setting and scratching is active after channel reset / part program end.
1	1	System frame for external work offset is active after channel reset / part program end.
2	---	Is not evaluated.
3	---	Is not evaluated.
4	1	System frame for workpiece reference points is active after channel reset / part program end.
5	1	System frame for cycles is active after channel reset / part program end.
6	---	Reserved, RESET response depends on MD20110 $MC_RESET_MODE_MASK.
7	1	System frame $P_ISO1FR is active after channel reset / part program end.
8	1	System frame $P_ISO2FR is active after channel reset / part program end.
9	1	System frame $P_ISO3FR is active after channel reset / part program end.
10	1	System frame $P_ISO4FR is active after channel reset / part program end.
11	1	System frame $P_RELFR is active after channel reset / part program end.
For bit<n> = 0, the associated system frame is **not** active.		

Reset behavior of the system frames for TCARR, PAROT, TOROT and TOFRAME

The reset behavior of the system frames for `TCARR`, `PAROT`, `TOROT` and `TOFRAME` after channel reset / part program end depends on the reset setting of the G commands.

The setting is made with machine data:

- MD20110 $MC_RESET_MODE_MASK, bit<n> = <value>

Bit	Value	Meaning	
0	0	The current system frame for TCARR and PAROT is retained.	
0	1	**Additional relevant machine data settings**	**Effect**
		MD20152 $MC_GCODE_RESET_MODE[51] = **0** and MD20150 $MC_GCODE_RESET_VALUES[51] = **1**	`PAROTOF`
		MD20152 $MC_GCODE_RESET_MODE[51] = **0** and MD20150 $MC_GCODE_RESET_VALUES[51] = **2**	`PAROT`
		MD20152 $MC_GCODE_RESET_MODE[51] = **1**	`TCARR` and `PAROT` are retained.
		MD20152 $MC_GCODE_RESET_MODE[52] = **0** and MD20150 $MC_GCODE_RESET_VALUES[52] = **1**	`TOROTOF`
		MD20152 $MC_GCODE_RESET_MODE[52] = **0** and MD20150 $MC_GCODE_RESET_VALUES[52] = **2**	`TOROT`
		MD20152 $MC_GCODE_RESET_MODE[52] = **0** and MD20150 $MC_GCODE_RESET_VALUES[52] = **3**	`TOFRAME`
		MD20152 $MC_GCODE_RESET_MODE[52] = **1**	`TOROT` and `TOFRAME` are retained
Description of the further bits for **bit 0 == 1**			
14	0	Chained complete basic frame is deleted.	
	1	The complete basic frame results from the following additional machine data:	
		MD24002 $MC_CHBFRAME_RESET_MASK, bit<n> = 1 n: The nth channel-specific basic frame is calculated into the chained complete basic frame.	
		MD10613 $MN_NCBFRAME_RESET_MASK, bit<n> = 1 n: The nth NCU-global basic frame is calculated into the chained complete basic frame.	

`TCARR` and `PAROT` are two independent functions, which describe the same frame. With `PAROTOF`, the component of `TCARR` is not activated for channel reset / part program end.

- MD20152 $MC_GCODE_RESET_MODE[ ] (RESET response of G groups)
- MD20150 $MC_GCODE_RESET_VALUES (RESET position of G groups)

Frame states after channel reset / part program end

Frame	Status after channel reset / part program end
$P_PFRAME programmable frame	Deleted.
$P_IFRAME settable frame	Is retained, depending on: ● MD20110 $MC_RESET_MODE_MASK ● MD20152 $MC_GCODE_RESET_MODE[7]

Frame	Status after channel reset / part program end
$P_GFRAME grinding frame	Is retained, depending on: • MD20110 $MC_RESET_MODE_MASK • MD20152 $MC_GCODE_RESET_MODE[63]
$P_ACTBFRAME complete basic frame	Is retained, depending on: • MD20110 $MC_RESET_MODE_MASK bit 0 and bit 14 • MD10613 $MN_NCBFRAME_RESET_MASK • MD24002 $MC_CHBFRAME_RESET_MASK
System frames: $P_PARTFRAME, $P_SETFRAME, $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_TOOLFRAME, $P_WPFRAME, $P_TRAFRAME, $P_ISO4FRAME, $P_RELFRAME, $P_CACFRAME	Are retained, depending on: • MD24006 $MC_CHSFRAME_RESET_MASK • MD20150 $MC_GCODE_RESET_VALUES[<n>]
$P_EXTFRAME external work offset	Retained
DRF offset	Retained

Deletion of system frames

The system frames in the data management can be deleted for channel reset / part program end with the following machine data:

MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK, bit<n> = <value>

Bit	Value	Meaning
0	1	System frame for actual value setting and scratching is deleted for channel reset / part program end.
1	1	System frame for external work offset is deleted for channel reset / part program end.
2	---	Reserved, for `TCARR` and `PAROT` see MD20150 $MC_GCODE_RESET_VALUES[ ].
3	---	Reserved, for `TOROT` and `TOFRAME` see MD20150 $MC_GCODE_RESET_VALUES[ ].
4	1	System frame for workpiece reference points is deleted for channel reset / part program end.
5	1	System frame for cycles is deleted for channel reset / part program end.
6	---	Reserved, RESET response depends on MD20110 $MC_RESET_MODE_MASK.
7	1	System frame $P_ISO1FR is deleted for channel reset / part program end.
8	1	System frame $P_ISO2FR is deleted for channel reset / part program end.
9	1	System frame $P_ISO3FR is deleted for channel reset / part program end.
10	1	System frame $P_ISO4FR is deleted for channel reset / part program end.
11	1	System frame $P_RELFR is deleted for channel reset / part program end.
For bit<n> = 0, the associated system frame is **not** deleted.		

11.5.12.4 Part program start

Frame conditions after part program start

Frame	Condition after part program start
$P_PFRAME programmable frame	Deleted.
$P_IFRAME settable frame	Is retained depending on: MD20112 $MC_START_MODE_MASK
$P_GFRAME grinding frame	Is retained depending on: MD20112 $MC_START_MODE_MASK
$P_ACTBFRAME complete basic frame	Retained
System frames: $P_PARTFRAME, $P_SETFRAME, $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_TOOLFRAME, $P_WPFRAME, $P_TRAFRAME, $P_ISO4FRAME, $P_RELFRAME, $P_CACFRAME	Retained
$P_EXTFRAME external work offset	Retained
DRF offset	Retained

11.5.12.5 Block search

Block search with calculation

Data management frames are also modified when carrying out a block search with calculation.

Cancellation of block search

If a block search is aborted with channel reset, then the following machine data can be used to set that all data management frames are reset to the value they had before the block search:

MD28560 $MC_MM_SEARCH_RUN_RESTORE_MODE, bit<n>

Bit	Value	Meaning
0	1	All frames in the data management are restored.

In case of cascaded block searches, the frames are set to the status of the previous block search.

SERUPRO

The "SERUPRO" function is not supported.

11.5.12.6 REPOS

There is no special treatment for frames. If a frame is modified in an ASUB, it is retained in the program. On repositioning with REPOS, a modified frame is included, provided the modification was activated in the ASUB.

11.6 Workpiece-related actual value system

11.6.1 Overview

Definition

The term "workpiece-related actual-value system" designates a series of functions that permit the user:

- To use a workpiece coordinate system defined in machine data after powerup.
  Features:
  - No additional operations are necessary.
  - Effective in JOG and AUTOMATIC modes
- To retain the valid settings for the following after end of program for the next part program:
  - Active plane
  - Settable frame (`G54`-`G57`)
  - Kinematic transformation
  - Active tool offset
- To change between work coordinate system and machine coordinate system via the user interface.
- To change the work coordinate system by operator action (e.g., changing the settable frame or the tool offset).

11.6.2 Use of the workpiece-related actual value system

Requirements, basic settings

The settings described in the previous Section have been made for the system.
The predefined setting after power-up of the HMI software is MCS.

Switchover to WCS

The change to the WCS via the user interface causes the axis positions relative to the origin of the WCS to be displayed.

Switchover to MCS

The change to the MCS via the user interface causes the axis positions relative to the origin of the MCS to be displayed.

Interrelationships between coordinate systems

The figure below shows the interrelationships between the machine coordinate system (MCS) and the workpiece coordinate system (WCS).

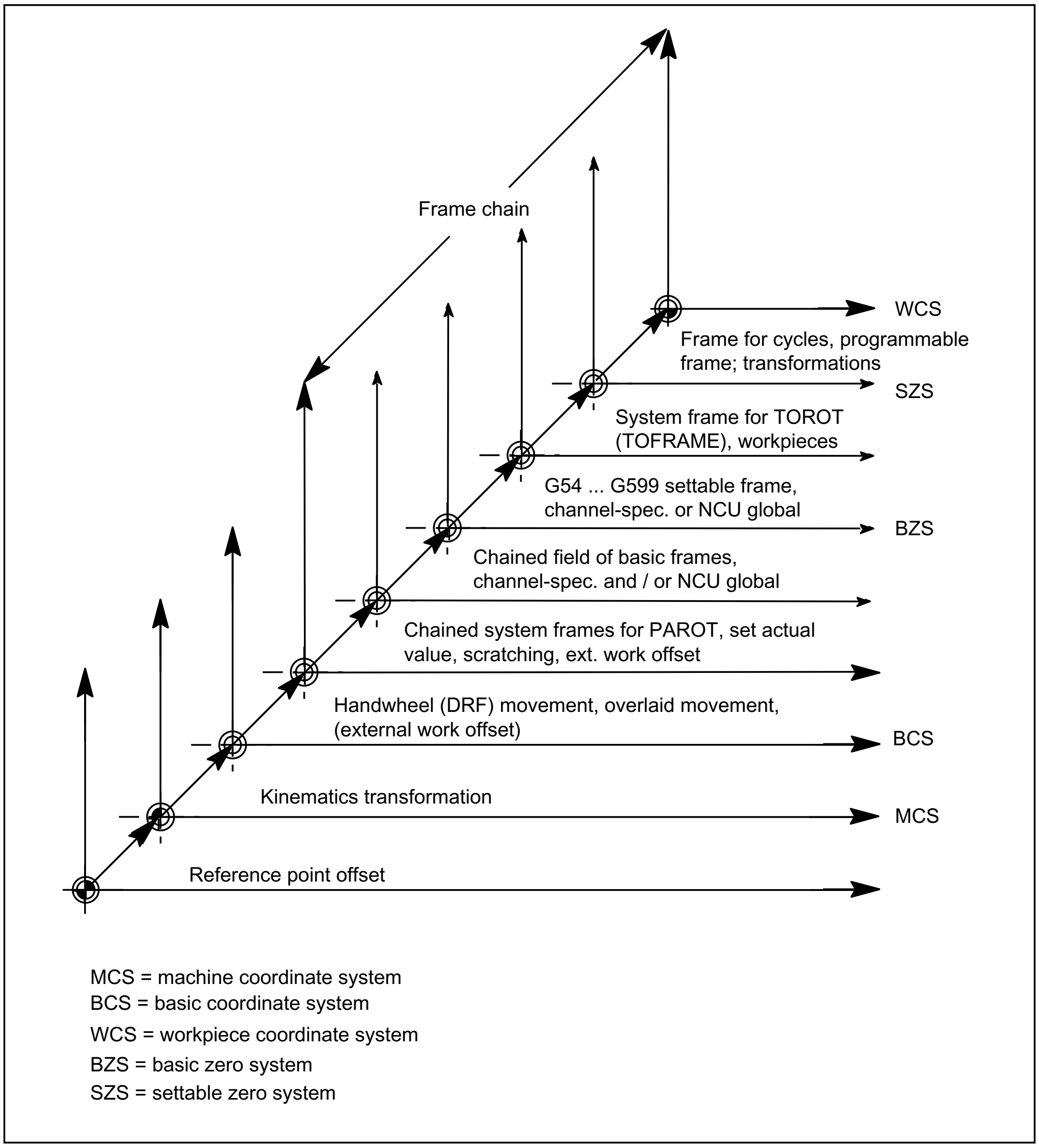

Frame chain
WCS
Frame for cycles, programmable frame; transformations
SZS
System frame for TOROT (TOFRAME), workpieces
G54 ... G599 settable frame, channel-spec. or NCU global
BZS
Chained field of basic frames, channel-spec. and / or NCU global
Chained system frames for PAROT, set actual value, scratching, ext. work offset
Handwheel (DRF) movement, overlaid movement, (external work offset)
BCS
Kinematics transformation
MCS
Reference point offset
MCS = machine coordinate system
BCS = basic coordinate system
WCS = workpiece coordinate system
BZS = basic zero system
SZS = settable zero system

Figure 11-28 Interrelationship between coordinate systems

For further information, see "H2: Auxiliary function outputs to PLC (Page 401)" and "W1: Tool offset (Page 1451)".

References:

- Programming Guide Fundamentals
- Function Manual, Extended Functions; Kinematic Transformation (M1)

- Function Manual, Special Functions; Axis Couplings and ESR (M3);
  Section: Coupled motion, Section: Master value coupling
- Function Manual, Special Functions; Tangential Control (T3)

11.6.3 Special reactions

Overstore

Overstoring in `RESET` state of:

- Frames (zero offsets)
- Active plane
- Activated transformation
- Tool offset

immediately affects the actual-value display of all axes in the channel.

Entry via operator panel front

If operations on the operator panel are used to change the values for
"Active frame" (zero offsets, "Parameters" operating area)
and
"Active tool length compensation" ("Parameters" operating area)
, one of the following actions is used to activate these changes in the display:

- Press the `RESET` key.
- Reselect:
  - Zero offset by the part program
  - Tool offset by the part program
- Reset:
  - Zero offset by overstoring
  - Tool offset by overstoring
- Part program start

MD9440

If the HMI machine data
MD9440 ACTIVATE_SEL_USER_DATA
 for the operator panel front is set, the entered values become active immediately in RESET state.

When values are entered in the part-program execution stop state, they become effective when program execution continues.

Actual-value reading

If the actual value of $AA_IW is read in the WCS after activation of a frame (zero offset) or a tool offset, the activated changes are already contained in the result read even if the axes have not yet been traversed with the activated changes.

The actual values in the settable zero system (SZS) can be read from the part program for each axis using the variable $AA_IEN[axis].

The actual values in the basic zero system (BZS) can be read from the part program for each axis using the variable $AA_IBN[axis].

Actual-value display

The programmed contour is always displayed in the WCS.

The following offsets are added to the MCS:

- Kinematic transformation
- DRF offset/zero offset external
- Active frame
- Active tool offset of the current tool

Switchover by PLC

The actual values can be displayed in the WCS, SZS, BZS or MCS via the PLC. The PLC can define, which coordinate system corresponds to the workpiece coordinate system on a machine.

On MMC power-up the MCS is preset.
With the signal DB19 DBB0.7 "MCS/WCS switchover", it is also possible to switch from the PLC to the WCS.

Transfer to PLC

Depending on machine data
MD20110 / MD20112, bit 1
, the auxiliary functions (`D`, `T`, `M`) are output to the PLC (or not) on selection of the tool length compensation.

Note

If the WCS is selected from the PLC, an operator action can still be used to switch between the WCS and MCS for the relevant mode.
However, when the mode and or area is changed, the WCS selected by the PLC is evaluated and activated (see Section "K1: Mode group, channel, program operation, reset response (Page 479)").

11.7 Restrictions

There are no supplementary conditions to note.

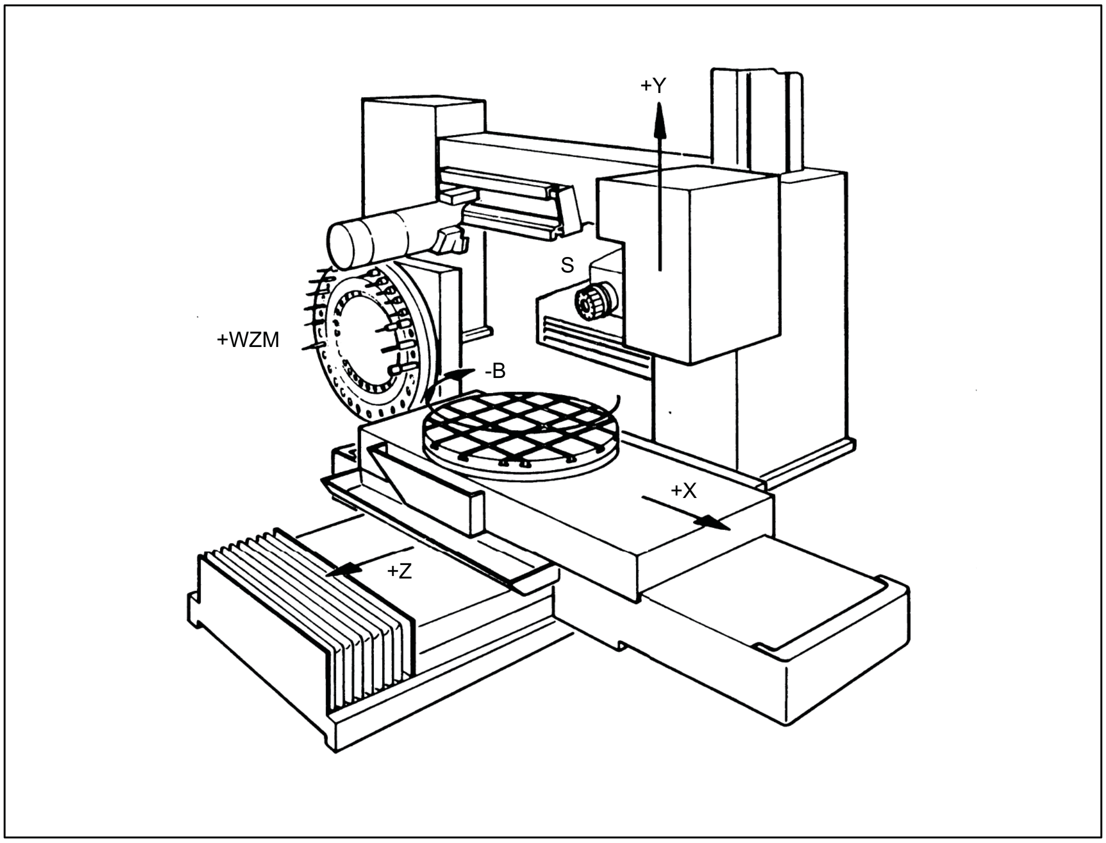
11.8 Examples

11.8.1 Axes

Axis configuration for a 3axis milling machine with rotary table

1. Machine axis: X1	Linear axis
2. Machine axis: Y1	Linear axis
3. Machine axis: Z1	Linear axis
4. Machine axis: B1	Rotary table (for turning for multiface machining)
5. Machine axis: W1	Rotary axis for tool magazine (tool revolver)
6. Machine axis: C1	(Spindle)

1. Geometry axis: X	(1. channel)
2. Geometry axis: Y	(1. channel)
3. Geometry axis: Z	(1. channel)
1. Special axis: B	(1. channel)
2. Special axis: WZM	(1. channel)
1. spindle: S1/C	(1. channel)



Parameterization of the machine data

Machine data	Value
MD10000 AXCONF_MACHAX_NAME_TAB[0]	= X1
MD10000 AXCONF_MACHAX_NAME_TAB[1]	= Y1
MD10000 AXCONF_MACHAX_NAME_TAB[2]	= Z1
MD10000 AXCONF_MACHAX_NAME_TAB[3]	= B1
MD10000 AXCONF_MACHAX_NAME_TAB[4]	= W1
MD10000 AXCONF_MACHAX_NAME_TAB[5]	= C1
MD20050 AXCONF_GEOAX_ASSIGN_TAB[0]	= 1
MD20050 AXCONF_GEOAX_ASSIGN_TAB[1]	= 2
MD20050 AXCONF_GEOAX_ASSIGN_TAB[2]	= 3
MD20060 AXCONF_GEOAX_NAME_TAB[0]	=X
MD20060 AXCONF_GEOAX_NAME_TAB[1]	=Y
MD20060 AXCONF_GEOAX_NAME_TAB[2]	=Z
MD20070 AXCONF_MACHAX_USED[0]	= 1
MD20070 AXCONF_MACHAX_USED[1]	= 2

Machine data	Value
MD20070 AXCONF_MACHAX_USED[2]	= 3
MD20070 AXCONF_MACHAX_USED[3]	= 4
MD20070 AXCONF_MACHAX_USED[4]	= 5
MD20070 AXCONF_MACHAX_USED[5]	= 6
MD20080 AXCONF_CHANAX_NAME_TAB[0]	=X
MD20080 AXCONF_CHANAX_NAME_TAB[1]	=Y
MD20080 AXCONF_CHANAX_NAME_TAB[2]	=Z
MD20080 AXCONF_CHANAX_NAME_TAB[3]	= B
MD20080 AXCONF_CHANAX_NAME_TAB[4]	= WZM
MD20080 AXCONF_CHANAX_NAME_TAB[5]	= S1
MD30300 IS_ROT_AX[3]	= 1
MD30300 IS_ROT_AX[4]	= 1
MD30300 IS_ROT_AX[5]	= 1
MD30310 ROT_IS_MODULO[3]	= 1
MD30310 ROT_IS_MODULO[4]	= 1
MD30310 ROT_IS_MODULO[5]	= 1
MD30320 DISPLAY_IS_MODULO[3]	= 1
MD30320 DISPLAY_IS_MODULO[4]	= 1
MD20090 SPIND_DEF_MASTER_SPIND	= 1
MD35000 SPIND_ASSIGN_TO_MACHAX[AX1]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX2]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX3]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX4]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX5]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX6]	= 1

11.8.2 Coordinate systems

Configuration of a global basic frame

An NC with two channels is required. The following applies:

- Both channels can then write the global basic frame.
- The other channel recognizes the change when the global basic frame is activated again.

- Both channels can read the global basic frame.
- Both channels can activate the global basic frame independently.

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 the 1st 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, because rotations are not permitted for
                                       ; NC-global frames
. . .
```

Part program in the 2nd channel

Code (excerpt)	; Comment
. . .	
N100 $P_NCBFR[0] = CTRANS(x, 10)	; The NC-global basic frame is also effective in the 2nd channel
. . .	
N510 G500 X10	; Activate basic frame
N520 $P_CHBFRAME[0] = CTRANS(x, 10)	; Current frame of the 2nd channel is activated with an offset
. . .	

11.8.3 Frames

Example 1

The channel axis is to be switched to a geometry axis by means of a geometry axis interchange.

The intention of the interchange is to give the programmable frame a translation component of 10 in the X axis.

The current settable frame is to be retained:

FRAME_GEOAX_CHANGE_MODE = 1

Program code	Comment
$P_UIFR[1] = CROT(X,10,Y,20,Z,30)	; Frame is retained after geometry axis interchange.
G54	; Settable frame becomes active.
TRANS A10	; Axis-specific offset of A is also interchanged.
GEOAX(1,A)	; A becomes the X axis
	; $P_ACTFRAME = CROT(X,10,Y,20,Z,30) : CTRANS(X10)

Multiple channel axes can become geometry axes at the same time on a transformation change.

Example 2

Channel axes 4, 5 and 6 become the geometry axes of a 5axis orientation transformation. The geometry axes are thus all substituted before the transformation.

The current frames are changed when the transformation is activated.

The axis-specific frame components of the channel axes that become geometry axes are used in the calculation of the new WCS. Rotations programmed before the transformation are retained. The old WCS is restored when the transformation is deactivated.

The most common application is probably that the geometry axes do not change before and after the transformation and that the frames are to stay as they were before the transformation.

Machine data:

$MN_FRAME_GEOAX_CHANGE_MODE = 1

$MC_AXCONF_CHANAX_NAME_TAB [0] = "CAX"
$MC_AXCONF_CHANAX_NAME_TAB [1] = "CAY"
$MC_AXCONF_CHANAX_NAME_TAB [2] = "CAZ"
$MC_AXCONF_CHANAX_NAME_TAB [3] = "A"
$MC_AXCONF_CHANAX_NAME_TAB [4] = "B"
$MC_AXCONF_CHANAX_NAME_TAB [5] = "C"

$MC_AXCONF_GEOAX_ASSIGN_TAB [0] = 1
$MC_AXCONF_GEOAX_ASSIGN_TAB [1] = 2
$MC_AXCONF_GEOAX_ASSIGN_TAB [2] = 3

$MC_AXCONF_GEOAX_NAME_TAB [0] = "X"
$MC_AXCONF_GEOAX_NAME_TAB [1] = "Y"
$MC_AXCONF_GEOAX_NAME_TAB [2] = "Z"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 5
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 6

$MC_TRAFO_AXES_IN_1 [0] = 4
$MC_TRAFO_AXES_IN_1 [1] = 5
$MC_TRAFO_AXES_IN_1 [2] = 6
$MC_TRAFO_AXES_IN_1 [3] = 1
$MC_TRAFO_AXES_IN_1 [4] = 2

Program:

Program code	**Comment**
`$P_NCBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)`	
`$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`	`; Transformation sets GEOAX(4,5,6)`
	`; $P_NCBFRAME[0] = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3)`
	`; $P_ACTBFRAME = CTRANS(X,8,Y,10,Z,12,CAX,2,CAY,4,CAZ,6)`
	`; $P_PFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3) : CROT(X,10,Y,20,Z,30)`
	`; $P_IFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3) : CROT(Z,45)`
`TRAFOOF`	`; Deactivating transformation sets GEOAX (1,2,3)`

Program code	Comment
	`; $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)`

11.9 Data lists

11.9.1 Machine data

11.9.1.1 Displaying machine data

Number	Identifier: $MM_	Description
SINUMERIK Operate		
9242	MA_STAT_DISPLAY_BASE	Numerical basis for display of moving joint `STAT`
9243	MA_TU_DISPLAY_BASE	Numerical basis for display of rotary axis position `TU`
9244	MA_ORIAXES_EULER_ANGLE_NAME	Display of orientation axes as Euler angle
9245	MA_PRESET_FRAMEIDX	Value storage scratching and PRESET
9247	USER_CLASS_BASE_ZERO_OFF_PA	Availability of basic offset in "Parameters" operating area
9248	USER_CLASS_BASE_ZERO_OFF_MA	Availability of basic offset in Machine operating area
9424	MA_COORDINATE_SYSTEM	Coordinate system for actualvalue display
9440	ACTIVE_SEL_USER_DATA	Active data (frames) are immediately operative after editing
9449	WRITE_TOA_LIMIT_MASK	Applicability of MD9203 to edge data and locationdependent offsets
9450	MM_WRITE_TOA_FINE_LIMIT	Limit value for wear fine
9451	MM_WRITE_ZOA_FINE_LIMIT	Limit value for offset fine

11.9.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB	Machine axis name
10600	FRAME_ANGLE_INPUT_MODE	Sequence of rotation in the frame
10602	FRAME_GEOAX_CHANGE_MODE	Frames and switchover of geometry axes

Number	Identifier: $MN_	Description
10610	MIRROR_REF_AX	Reference axis for mirroring
10612	MIRROR_TOGGLE	Change over mirror
10613	NCBFRAME_RESET_MASK	ActiveNCU-global basic frame after reset
10615	NCBFRAME_POWERON_MASK	Reset global basic frames after Power On
10617	FRAME_SAVE_MASK	Behavior of frames for SAVE subprograms
10650	IPO_PARAM_NAME_TAB	Name of interpolation parameters
10660	INTERMEDIATE_POINT_NAME_TAB	Name of intermediate point coordinates for G2/G3
11640	ENABLE_CHAN_AX_GAP	Channel axis gaps are allowed
18600	MM_FRAME_FINE_TRANS	Fine offset for FRAME (SRAM)
18601	MM_NUM_GLOBAL_USER_FRAMES	Number of globally predefined user frames (SRAM)
18602	MM_NUM_GLOBAL_BASE_FRAMES	Number of global basic frames (SRAM)

11.9.1.3 Channel-specific machine data

Number	Identifier: $MC_	Description
20050	AXCONF_GEOAX_ASSIGN_TAB	Assignment geometry axis to channel axis
20060	AXCONF_GEOAX_NAME_TAB	Geometry axis name in channel
20070	AXCONF_MACHAX_USED	Machine axis number valid in channel
20080	AXCONF_CHANAX_NAME_TAB	Channel axis name in the channel
20110	RESET_MODE_MASK	Definition of basic control settings after RESET / TP end
20118	GEOAX_CHANGE_RESET	Allow automatic geometry axis change
20126	TOOL_CARRIER_RESET_VALUE	Active toolholder on RESET
20140	TRAFO_RESET_VALUE	Transformation record on power-up (RESET / TP-End)
20150	GCODE_RESET_VALUES	Initial setting of the G groups
20152	GCODE_RESET_MODE	RESET response of the G groups
20184	TOCARR_BASE_FRAME_NUMBER	Number of the basic frame for pickup of the table offset
21015	INVOLUTE_RADIUS_DELTA	End point monitoring for evolvents (involutes)
22532	GEOAX_CHANGE_M_CODE	M code for replacement of geometry axes
22534	TRAFO_CHANGE_M_CODE	M code for transformation change
24000	FRAME_ADD_COMPONENTS	Frame components for G58 and G59
24002	CHBFRAME_RESET_MASK	RESET response of channel-specific basic frames
24004	CHBFRAME_POWERON_MASK	Reset channel-specific basic frames after Power On
24006	CHSFRAME_RESET_MASK	Active system frames after reset
24007	CHSFRAME_RESET_CLEAR_MASK	Clear system frames on RESET
24008	CHSFRAME_POWERON_MASK	Reset system frames after POWER ON
24010	PFRAME_RESET_MODE	RESET mode for programmable frame
24020	FRAME_SUPPRESS_MODE	Positions for frame suppression
24030	FRAME_ACT_SET	SZS coordinate system setting
24040	FRAME_ADAPT_MODE	Adapting active frames
24050	FRAME_SAA_MODE	Saving and activating data management frames

Number	Identifier: $MC_	Description
24805	TRACYL_ROT_AX_FRAME_1	Rotary axis offset `TRACYL 1`
24855	TRACYL_ROT_AX_FRAME_2	Rotary axis offset `TRACYL 2`
24905	TRANSMIT_ROT_AX_FRAME_1	Rotary axis offset `TRANSMIT1`
24955	TRANSMIT_ROT_AX_FRAME_2	Rotary axis offset `TRANSMIT2`
28080	MM_NUM_USER_FRAMES	Number of settable Frames (SRAM)
28081	MM_NUM_BASE_FRAMES	Number of basic frames
28082	MM_SYSTEM_FRAME_FRAMES	System frames (SRAM)
28560	MM_SEARCH_RUN_RESTORE_MODE	Restore data after a simulation

11.9.1.4 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32074	FRAME_OR_CORRPOS_NOTALLOWED	`FRAME` or HL offset is not permitted
35000	SPIND_ASSIGN_TO_MACHAX	Assignment spindle to machine axis

11.9.2 Setting data

11.9.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42440	FRAME_OFFSET_INCR_PROG	Work offsets in frames
42980	TOFRAME_MODE	Frame definition for TOFRAME, TOROT and PAROT

11.9.3 System variables

Identifier	Description
$AA_ETRANS[axis]	External work offset
$AA_IBN[axis]	Actual value in basic zero coordinate system (BZS)
$AA_IEN[axis]	Actual value in settable zero point coordinate system (SZS)
$AA_OFF[axis]	Overlaid motion for programmed axis
$AC_DRF[axis]	Handwheel override of an axis
$AC_JOG_COORD	Coordinate system for manual traversing
$P_ACSFRAME	Active frame between BCS and SZS
$P_ACTBFRAME	Active complete basic frame
$P_ACTFRAME	Active complete frame

Identifier	Description
$P_BFRAME	1st active basic frame. Corresponds to $P_CHBFRAME[0]
$P_CHBFR[<n>]	Data management frame: Basic frame can be activated via G500, G54...G599
$P_CHBFRAME[<n>]	Active basic frame
$P_CHBFRMASK	Basic frame mask in the channel
$P_CHSFRMASK	System frame mask
$P_CYCFR	Data management frame: System frame for cycles
$P_CYCFRAME	Active system frame for cycles
$P_EXTFR	Data management frame: System frame for work offset external
$P_EXTFRAME	Active system frame for external work offset
$P_IFRAME	Active settable frame
$P_ISO1FR	Data management frame: System frame for ISO G51.1 mirroring
$P_ISO2FR	Data management frame: System frame for ISO G68 2DROT
$P_ISO3FR	Data management frame: System frame for ISO G68 3DROT
$P_ISO4FR	Data management frame: System frame for ISO G51 Scale
$P_ISO1FRAME	Active system frame for ISO G51.1 Mirroring
$P_ISO2FRAME	Active system frame for ISO G68 2DROT
$P_ISO3FRAME	Active system frame for ISO G68 3DROT
$P_ISO4FRAME	Active system frame for ISO G51 Scale
$P_NCBFR[n]	Data management frame: NCU-global basic frame in the data management, can be activated via G500, G54...G599
$P_NCBFRAME[n]	Active NCU-global basic frame
$P_NCBFRMASK	Global basic frame mask
$P_PARTFR	Data management frame: System frame for TCARR and PAROT
$P_PARTFRAME	Active system frame for TCARR and PAROT with orientable toolholder
$P_PFRAME	Active programmable frame
$P_SETFR	Data management frame: System frame for actual value setting
$P_SETFRAME	Active system frame for actual value setting
$P_TOOLFR	Data management frame: System frame for TOROT and TOFRAME
$P_TOOLFRAME	Active system frame for TOROT and TOFRAME
$P_TRAFRAME	Data management frame: System frame for transformations
$P_TRAFRAME	Active system frame for transformations
$P_UBFR	1st basic frame in the channel in the data management that is activated after G500, G54...G599. Corresponds to $P_CHBFR[0].
$P_UIFR[n]	Data management frame: Settable frame, can be activated via G500, G54 to G599
$P_UIFRNUM	Number of active settable frame
$P_WPFR	Data management frame: System frame for the workpiece
$P_WPFRAME	Active system frame for the workpiece

11.9.4 Signals

11.9.4.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
T function modification	DB21, ...DBX61.0-.2	-
D function modification	DB21, ...DBX62.0-.2	-
T function 1	DB21, ...DBB118-119	DB250x.DBD2000
D function 2	DB21, ...DBB129	DB250x.DBD5000
Number of active function G group 1 8 (bit int)	DB21, ...DBB208	DB350x.DBB0
Number of active function G group 2 8 (bit int)	DB21, ...DBB209	DB350x.DBB1
...	...	…
Number of active function G group 29 8 (bit int)	DB21, ...DBB236	DB350x.DBB28

11.9.4.2 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Accept zero offset external	DB31, ...DBX3.0	-

11.9.4.3 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Spindle / rotary axis	DB31, ...DBX60.0	DB390x.DBX0.0

N2: Emergency stop 12

12.1 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.
- The Emergency Stop function must bring the machine to a standstill according to stop category 0 or 1 (EN 60204).
- 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.

12.2 Relevant standards

Relevant standards

Compliance with the following standards is essential for the emergency stop function:

- EN ISO 12000-1
- EN ISO 12000-2
- EN 418
- EN 60204

Emergency stop

In accordance with EN 418, an emergency stop is a function that:

- Is intended to prevent or diminish developing or existing risks to operating personnel, and damage to the machine or machined materials.
- Is triggered by a single action of a person, if the normal stop function is not suitable for it.

Hazards

In the terms of EN 418, risks may arise from:

- Functional irregularities (machine malfunctions, unacceptable properties of the material to be machined, human error, etc.).
- Normal operation.

Stadard EN ISO 12000-2

In accordance with the basic safety requirement of the EC Machinery Directive regarding emergency stop, machines must be equipped with an 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.

Note

The machine manufacturer is expressly directed to comply with the national and international standards. The SINUMERIK controllers support the machine manufacturer in the implementation of the emergency stop function according to the specifications in the following function description. But the responsibility for the emergency stop function (its triggering, sequence and acknowledgement) rests exclusively with the machine manufacturer.

12.3 Emergency stop control elements

Emergency stop control elements

In accordance with EN 418, emergency stop control elements must be designed so that they latch mechanically on their own and are easy for the operator and others to actuate in the event of an emergency.

The following list includes some possible types of control elements:

- Mushroom pushbutton switches
- Wires/cables, cords, rods
- Puller grips
- In special cases: Foot switches without protective covers

Emergency stop button and control

Actuation of the emergency stop button or a signal derived directly from the button must be routed to the controller (PLC) as a PLC input. In the PLC user program, this PLC input must be forwarded to the NC on the interface signal:

DB10 DBX56.1 (Emergency stop)

Resetting of the emergency stop button or a signal derived directly from the button must be routed to the controller (PLC) as a PLC input. In the PLC user program, this PLC input must be forwarded to the NC on the interface signal:

DB10 DBX56.2 (Acknowledge emergency stop)

Connection conditions

For connecting the emergency stop button see:
References:
Operator Components Manual

12.4 Emergency stop sequence

After actuation of the emergency stop control element, the emergency stop device must operate in the best possible way to prevent or minimize the danger.

"In the best possible way" means that the most favorable delay rate can be selected and the correct stop category (defined in EN 60204) can be determined according to a risk assessment.

Emergency stop sequence in the NC

The predefined (in EN 418) sequence of internal functions implemented to obtain the emergency stop state is as follows in the control system:

1. Part program execution is interrupted.
   All machine axes are braked in the relevant axis-specific parameterized time:
   MD36610 $MA_AX_EMERGENCY_STOP_TIME (time of braking ramp in event of errors)
   The maximum braking ramp that can be achieved thereby, is defned by the maximum brake current of the respective drive. The maximum brake current is achieved by setting a setpoint = 0 (fast braking).
2. Reset interface signal:
   DB11 DBX6.3 (Mode group ready)
3. Set the interface signal:
   DB10 DBX106.1 (emergency stop active)
4. Alarm 3000 "Emergency stop" is displayed.

5. After the expiry of a paramaterized delay time, the servo enables of machine axes are reset.
   The setting of the delay time is programmed in machine data:
   MD36620 $MA_SERVO_DISABLE_DELAY_TIME (OFF delay of the controller enable)
   The following setting rule must be observed: MD36620 ≥ MD36610
6. All machine axes are switched in the follow-up mode within the controller.
   The machine axes are no longer in position control.

Emergency stop sequence at the machine

The emergency stop sequence on the machine is determined solely by the machine manufacturer.

Attention should be paid to the following points in connection to the sequence on the NC:

- The process in the NC is started using the interface signal:
  DB10 DBX56.1 (Emergency stop)
  After the machine axes have come to a standstill, the power supply must be interrupted, in compliance with EN 418.

  Note

  The responsibility for interrupting the power supply rests with the machine manufacturer.

- The digital and analog outputs of the PLC I/O are not influenced by the emergency stop sequence in the NC.
  If individual outputs are required to attain a particular state or voltage level in the event of an emergency stop, the machine manufacturer must implement this in the PLC user program.
- The fast digital outputs of the NCK I/O system are not influenced by the emergency stop sequence in the NC.
  If individual outputs must assume a specific state in case of emergency stop, the machine manufacturer must transmit the desired state to the NC in the PLC user program via interface signals:
  DB10 DBB4-7

  Note

  If the sequence in the NC is not to be executed as described above, then the interface signal DB10 DBX56.1 (emergency stop) must not be set until an emergency stop state defined by the machine manufacturer in the PLC user program is reached.

  As long as the interface signal is not set and no other alarm is pending, all interface signals are operative in the NC. Any emergency stop state defined by the manufacturer (including axis-, spindle- and channel-specific emergency stop states) can therefore be assumed.

12.5 Emergency stop acknowledgement

The emergency stop control element may only be reset as a result of manual manipulation of the emergency stop control element according to EN 418.

Resetting of the emergency stop control element alone must not trigger a restart command.

A machine restart must be impossible until all of the actuated emergency stop control elements have been deliberately reset by hand.

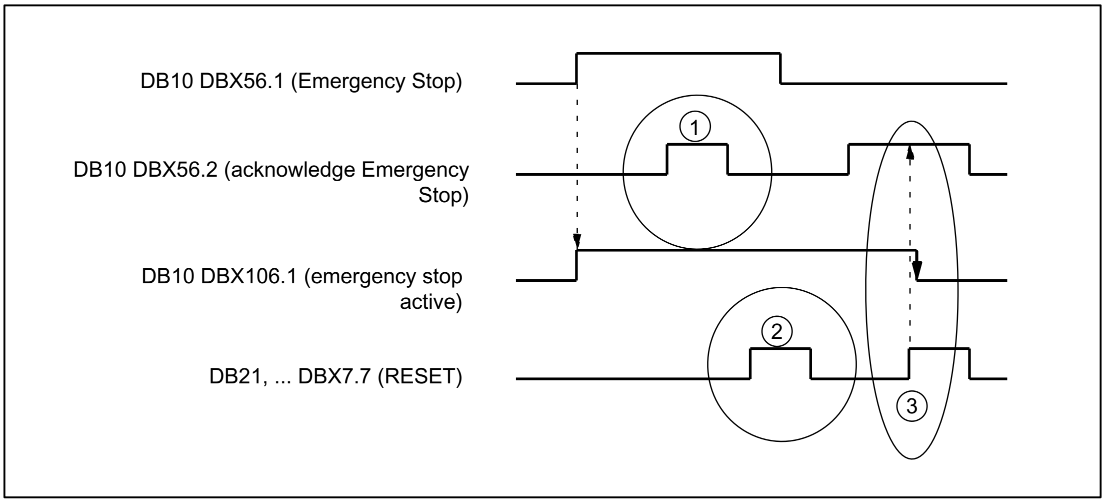
Emergency stop acknowledgment

The EMERGENCY STOP state is only reset if the interface signal:DB10 DBX56.2 (acknowledge EMERGENCY STOP) is set followed by the interface signal:DB11, ... DBX0.7 (mode group reset).

Hence it can be noted that the interface signal DB10 DBX56.2 (acknowledge emergency stop) and the interface signal DB21, ... DBX7.7 (Reset) together are set at least for so long until the interface signal DB10 DBX106.1(emergency stop active) is reset.

Note

The emergency stop state cannot be reset with the interface signal DB21, ... DBX7.7 (Reset) alone.


DB10 DBX56.1 (Emergency Stop)
DB10 DBX56.2 (acknowledge Emergency Stop)
DB10 DBX106.1 (emergency stop active)
DB21, ... DBX7.7 (RESET)
1
2
3

(1) DB10 DBX56.2 (acknowledge emergency stop) is inoperative

(2) DB21, ... DBX7.7 (Reset) is inoperative

(3) DB10 DBX56.2 and DB21, ... DBX7.7 reset DB10 DBX106.1 (emergency stop active)

Figure 12-1 Resetting the emergency stop state

Effects

Resetting the emergency stop state has the following effects:

- Within the controller for all machine axes:
  - The servo enables are set.
  - The follow-up mode is canceled.
  - The position control is activated.
- The following interface signals are set:
  DB31, ... DBX60.5 (position control active)
  DB11 DBX6.3 (mode group ready)
- The following interface signal is reset:
  DB10 DBX106.1 (emergency stop active)
- Alarm 3000 "Emergency stop" is deleted.
- Part program processing is interrupted in all channels of the NC.

PLC and NC I/O

The PLC user program must switch the PLC and NC I/O back to the state for operation of the machine.

POWER OFF / ON (supply off / on)

The emergency stop state can also be reset by switching the controller off and back on (POWER OFF / ON).

Requirement:

During power-up of the controller the interface signal DB10 DBX56.1 (emergency stop) must not be set.

12.6 Data lists

12.6.1 Machine data

12.6.1.1 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
36610	AX_EMERGENCY_STOP_TIME	Length of the braking ramp for error states
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay servo enable

12.6.2 Signals

12.6.2.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Emergency stop	DB10.DBX56.1	DB2600.DBX0.1
Acknowledge Emergency Stop	DB10.DBX56.2	DB2600.DBX0.2

12.6.2.2 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Emergency stop active	DB10.DBX106.1	DB2700.DBX0.1

12.6.2.3 Signals to BAG

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group RESET	DB11.DBX0.7	DB3000.DBX0.7

P1: Transverse axes 13

13.1 Function

Transverse axis

Within the framework of "turning" technology, the transverse axis refers to the machine axis that travels perpendicular to the axis of symmetry of the spindle, in other words, to longitudinal axis Z.

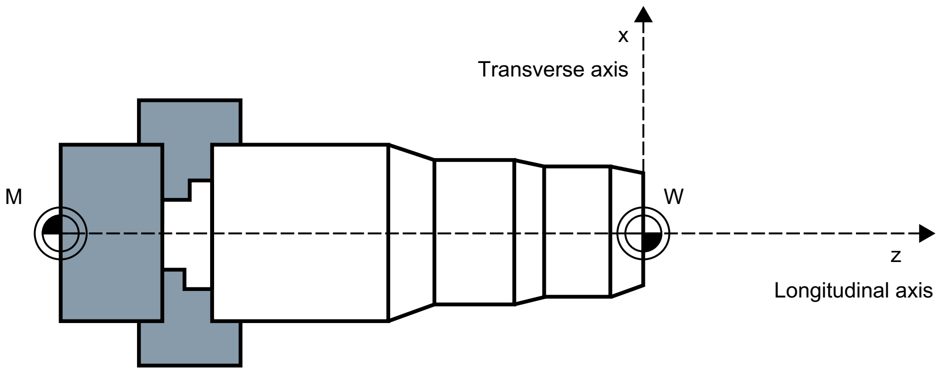


Figure 13-1 Position of the transverse axis in the machine coordinate system

Geometry axis as transverse axis

Every geometry axis of a channel can be defined as a transverse axis.

For the geometry axis as a transverse axis, the following functions can be simultaneously or separately permitted and activated:

- Programming and display in the diameter
- Reference axis for constant cutting speed G96/G961/G962

Several transverse axes in the channel

The introduction several transverse axes in the channel involves a functional decoupling of diameter programming and reference axis for G96/G961/G962. Diameter programming and reference axis for G96/G961/G962 can be active for different transverse axes (see table below).

	Programming and display in the diameter			**Reference axis for G96 / G961 / G962**	
Permissible axis type:	Geometry axis		Linear channel axes	Geometry axis	
Selection in the channel:	one	m of 3	m of n	one	one of 3
Specific effect:	Channel	Axis		Channel	
Machine data:	MD20100	MD30460		MD20100	

	Programming and display in the diameter	Reference axis for G96 / G961 / G962
Programming:	`DIAM*` channel-specific modal G group 29	`SCC[<Axis>]` channel-specific modal Reference axis for G96/G961/G962
Acceptance during axis replacement:	`DIAM*A[<Axis>]` axis-specific modal	
Axis-specific non-modal diametral/radius programming:	`DAC`, `DIC`; `RAC`, `RIC` blockwise axis-specific only programming	
DIAM*: DIAMOF, DIAMON, DIAM90, DIAMCYCOF DIAM*A[<Axis>]: DIAMOFA[<Axis>], DIAMONA[<Axis>], DIAM90A[<Axis>], DIACYCOFA[<Axis>], DIAMCHANA[<Axis>] <Axis>: Axis name for geometry, channel or machine axis name		

Note

Rotary axes are not permitted to serve as transverse axes.

Diameter-related data

DIAMON/DIAMONA[<Axis>]

After activation of the diameter programming with `DIAMON/DIAMONA[<Axis>]` (see "Programming (Page 865)"), the following data refer to diameter dimensions:

- Display data of transverse axis in the workpiece coordinate system:
  - Setpoint and actual position
  - Distance-to-go
  - REPOS offset
- "JOG" mode:
  - Increment (INC) through incremental manual handwheel traversing (depending on the active MD)
- Part program programming:
  - End positions, independent of reference mode (`G90` / `G91`)
  - Interpolation parameters of circular-path programming (`G2` / `G3`) if these are programmed with part program instruction `AC` absolute.
- Actual values read with reference to the workpiece coordinate system (WCS):
  - $AA_MW[<transverse axis>]
    System variable of the measuring functions `MEAS` (measuring with delete distance-to-go) and `MEAW` (measuring without delete distance-to-go)
  - $P_EP[<transverse axis>]
  - $AA_IW[<transverse axis>]

DIAM90/DIAM90A[<Axis>]

After activation of the reference-mode-dependent diameter programming with `DIAM90/DIAM90A[<Axis>]`, the following data is always displayed in relation to diameter regardless of the operating mode (`G90`/`G91`):

- Actual value
- Actual values read with reference to the workpiece coordinate system (WCS):
  - $AA_MW[<transverse axis>]
    System variable of the measuring functions `MEAS` (measuring with delete distance-to-go) and `MEAW` (measuring without delete distance-to-go)
  - $P_EP[<transverse axis>]
  - $AA_IW[<transverse axis>]

DIAMCYCOF/DIACYCOFA[<Axis>]

With `DIAMCYCOF` / `DIACYCOFA[<Axis>]`, a changeover to radius programming takes place within the controller. The diameter programming status that was active before `DIAMCYCOF` or `DIACYCOFA[AX]` continues to be displayed to the HMI.

Permanently radius-related data

For transverse axes, the following data is **always** entered, programmed and displayed in relation to radius:

- Part program programming
  - Interpolation parameters of circular-path programming with CIP
- Offsets:
  - Tool offsets
  - Programmable and configurable frames
  - External work offset
  - DRF and preset offset
  - etc.
- Working area limitation
- Software limit switch
- Feedrate
- Display data with reference to the machine coordinate system
- Display data of the service images for axis, FSD and MSD

Extended functions for data that is always radius-related:

The following applies for PLC axes, via FC18 or axes controlled exclusively from the PLC:

- The dimension for PLC axes in the radius also applies to several transverse axes with diameter function and is independent of channel-specific or axis-specific diameter programming.
- In the JOG mode (Inc) a PLC axis is subordinate to the channel status. If diameter programming is active and MD20624 $MC_HANDWH_CHAN_STOP_COND bit 15 = 0, only half the path of the specified increment is traversed.

13.2 Parameterization

Defining a geometry axis as a transverse axis

Defining **a** geometry axis as a transverse axis in the channel is realized using machine data:

MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function)

Example

MD20100 $MC_DIAMETER_AX_DEF="X" ; geometry axis X is the transverse axis in the channel.

For this axis, diameter programming and assigning a constant cutting speed with G96 / G961 / G962 are both permitted.

Several transverse axes in the channel

Defining additional transverse axes in the channel, for which the functionality of the axis-specific diameter programming is permitted, is performed using the machine data:

MD30460 BASE_FUNCTION_MASK.Bit 2 = <Value>

Bit	Value	Meaning
2	0	Axis-specific diameter programming is **not** permitted.
	1	Axis-specific diameter programming is permitted.

Note

The setting MD30460 bit 2 = 1 is only possible for linear axes.

Supplementary conditions

In a channel, a transverse axis can be defined as channel-specific (MD20100) and axis-specific (MD30460, bit 2) axis at the same time. The channel-specific machine data has the higher priority.

With MD20100, the following channel-specific functions are assigned to the transverse axis during the run-up

The following axis-specific basic position is assigned to the transverse axis during the power-up of the NC: "Transfer of the diameter programming channel status" `DIAMCHANA[<Achse>]`

With the release of the axis-specific diameter programming (MD30460, bit 2 = 1), the following axis-specific operations can be used at the user's end:

- `DIAMONA` (axis-specific modal diameter programming: ON)
- `DIAMOFA` (axis-specific modal diameter programming: OFF)
- `DIAM90A` (axis-specific modal diameter programming for `G90` and `AC`, radius programming for `G91` and `IC`)
- `DIACYCOFA` (axis-specific modal diameter programming: OFF in cycles)
- `DIAMCHANA` (transfer of the diameter programming channel status)

Dimensions for tool parameters

With the following channel-specific machine data, the following tool parameters can be activated as diameter values for all of the transverse axes defined in the channel:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK.<Bit> = <value>

MD20360 $MC_TOOL_PARAMETER_DEF_MASK		
Bit	**Value**	**Meaning**
1	1	Transverse axis tool length as a diameter
2	1	Alarm for wear or tool length as a diameter and plane change
3 1)	0	Work offset $P_EXTFRAME and frames For transverse axes, work offsets in frames are always calculated as radius values.
	1	The work offset (WO) in frames in the transverse axis as a diameter The frame stores the work offsets internally as a radius value. There is no conversion during a change of diameter to radius programming or vice versa.
4	1	Preset value as a diameter
5 1)	0	External work office (axis overlay) For transverse axes, the external work offset is always calculated as a radius value.
	1	External WO of the transverse axis as a diameter There is no conversion during a change of diameter to radius programming or vice versa.
6	1	Actual values of the transverse axis as a diameter
7	1	Display of actual values of the transverse axis as a diameter value
8 1)	1	Display of remaining path in WCS always as a radius
9 1)	For all of the transverse axes for which it is set that the specifications of the handwheel are path specifications (MD11346 $MN_HANDWH_TRUE_DISTANCE == 1), the following applies:	
	0	Half of the path of the specified handwheel increment is traveled if channel-specific or axis-specific diameter programming is active for this axis.
	1	Half of the path of the specified handwheel increment is always traveled.
10	1	Tool portion of an active tool carrier that can be oriented if no tool is active
11	1	Evaluation of $TC_DP6 as a diameter
12	1	Evaluation of $TC_DP15 as wear of the tool diameter

MD20360 $MC_TOOL_PARAMETER_DEF_MASK		
Bit	**Value**	**Meaning**
13 1)	1	When jogging around circles, the circle center point coordinate is always a radius value, see SD42690 $SC_JOG_CIRCLE_CENTRE
14 1)	1	For cycle masks, the absolute values of the transverse axis are always meant as the radius.
15	1	Incremental values of the transverse axis for cycle masks as diameters
1) The function is dependent upon the settings in MD20100 $MC_DIAMETER_AX_DEF and MD30460 $MA_BASE_FUNCTION_MASK, bit 2		

Behavior during manual handwheel traversing

The behavior during manual handwheel traversing (MD11346 $MN_HANDWH_TRUE_DISTANCE == 1 or 3) of a transverse axis while diameter programming is active can be adjusted with the following machine data:

MD20624 $MC_HANDWH_CHAN_STOP_COND.Bit 15 = <value>

Bit	Value	Meaning
15	0	Only the half path of the specified increment is traveled.
	1	The specified increment is traveled completely.

Displaying position values in the diameter

Position values of transverse axes are always displayed as diameter values, if:

MD27100 $MC_ABSBLOCK_FUNCTION_MASK, bit 0 = 1

Channel-specific basic position after power up, reset

The channel-specific basic position after power up or channel reset or the part program end of G group 29 (`DIAMON`, `DIAM90`, `DIAMOF`, `DIAMCYCOF`) is defined by the machine data MD20110 $MC_RESET_MODE_MASK.

MD20110 $MC_RESET_MODE_MASK, bit 0 = 0:

- Power-up: Basic position according to MD20150 $MC_GCODE_RESET_VALUE
- Channel reset or part program end: Basic position according to MD20150 $MC_GCODE_RESET_VALUES

MD20110 $MC_RESET_MODE_MASK, bit 0 = 1:

- After power-up: Basic position according to MD20150 $MC_GCODE_RESET_VALUE
- After channel reset or part program end: Basic position according to MD20152 $MC_GCODE_RESET_MODE

The user can set the respective desired status via an event-driven program call (ProgEvent).

If the basic position after the power-up is `G96` / `G961` / `G962`, a transverse axis must be defined with MD20100 $MC_DIAMETER_AX_DEF, otherwise the alarm 10870 is displayed.

To ensure that the reference axis for G96 / G961 / G962 is retained during a reset, end of part program or start of part program, the following setting must be made:

- Channel reset or part program end: MD20110 $MC_RESET_MODE_MASK, bit 18 = 1
- Part program start: MD20112 $MC_START_MODE_MASK, bit 18 = 1

A reference axis for G96 / G961 / G962 can also be assigned without the parameterization of a transverse axis in MD20100 via SCC[AX]. For this scenario, the constant cutting speed must not already be activated with G96. For additional information, see the following:

References

Programming Manual Fundamentals, Feedrate Control and Spindle Motion
"Constant cutting rate (G96/G961/G962, G97/G971/G972, G973, LIMS, SCC)

13.3 Programming

Transverse axes can be programmed with respect to both diameter and radius. Generally, they are diameter-related, i.e. programmed with doubled path dimension so that the corresponding dimensional information can be transferred to the part program directly from the technical drawings.

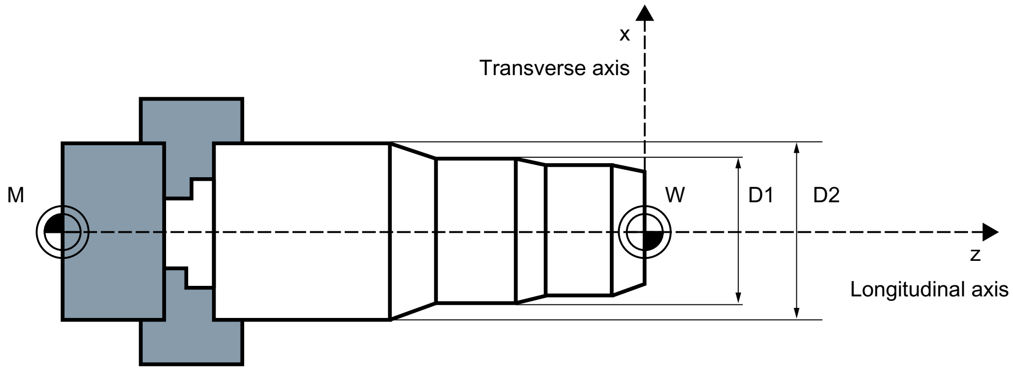


Figure 13-2 Transverse axis with diameter information (D1, D2)

Switching the diameter programming on/off

Channel-specific diameter programming

The activating or deactivating of the diameter programming is done via the modally active part program statements of the G group 29:

- DIAMON: Diameter programming ON
- DIAMOF: Diameter programming OFF, in other words, radius programming ON
- DIAM90: Diameter or radius programming depending on the reference mode:
  - Diameter programming ON in connection with absolute dimensioning G90
  - Radius programming ON in connection with incremental dimensioning G91
- DIAMCYCOF: Radius programming for G90 and G91 ON, for the HMI, the last active G command of this group remains active

Reference is made exclusively to the transverse axis of the channel.

Axis-specific diameter programming for several transverse axes in one channel

---

Note

The additionally specified axis must be activated via MD30460 $MA_BASE_FUNCTION_MASK with bit 2 = 1.

The axis specified must be a known axis in the channel. Geometry, channel or machine axes are permitted.

Programming is not permitted in synchronized actions.

The following axis-specific modal statements can be programmed several times in a part program block:

- `DIAMONA[<Axis>]`: Diameter programming for `G90`, `G91`, `AC` and `IC` ON
- `DIAMOFA[<Axis>]`: Diameter programming OFF, in other words, radius programming ON
- `DIAM90A[<Axis>]`: Diameter or radius programming depending on the reference mode:
  - Diameter programming ON in connection with absolute dimensioning `G90` and `AC`
  - Radius programming ON in connection with incremental dimensioning `G91` and `IC`
- `DIACYCOFA[Axis]`: Radius programming for `G90` and `G91` ON, for the HMI, the last active G command of this group remains active
- `DIAMCHANA[Axis]`: Acceptance of diameter programming channel status
- `DIAMCHAN`: All axes with MD30460, bit 2 = 1 accept the diameter programming channel status

Axis-specific modal statements have priority over the channel setting.

13.4 Supplementary conditions

Axis replacement

Due to a `GET` request from the parts program, the diameter programming status for an additional transverse axis is accepted in the new channel during axis replacement using `RELEASE[<Axis>]`.

Axis replacement in synchronized actions

For axis replacement in synchronized actions, a transverse axis takes the status of the axis-specific diameter programming with it into the new channel if:

- with MD30460, bit 2 = 1 axis-specific diameter programming is permitted for the transverse axis.
- the transverse axis is not subordinated to the channel-specific diameter programming in the releasing channel.

The active dimension can be queried via the system variable $AA_DIAM_STAT[<Axis>].

Axis replacement via axis container rotation

By rotating the axis container, the assignment of a channel axis can change to assignment of a machine axis. The current diameter programming status is retained however for the channel axis after the rotation. This also applies to the current channel and axis status, because the status is the same for all axes of the axis container at the time of the machine data "putting into effect" the status from MD30460 $MA_BASE_FUNCTION_MASK.

13.5 Examples

Example 1

X is a transverse axis defined via MD20100 $MC_DIAMETER_AX_DEF.

Y is a geometry axis and U is an additional axis. These two axes are additional transverse axes with diameter specifications defined in MD30460 $MA_BASE_FUNCTION_MASK with bit 2 = 1.

DIAMON is not active after power up.

Program code	**Comment**
`N10 G0 G90 X100 Y50`	`;no diameter programming is active`
`N20 DIAMON`	`;Channel-specific diameter programming, in effect for X`
`N30 Y200 X200`	`;Dimensions: X in the diameter, Y in the radius`

`N40 DIAMONA[Y]`	`;axis-specific modal diameter programming,`
	`;in effect for Y`

`N50 Y250 X300`	`;Dimensions: X and Y in diameter`
`N60 DIAM90`	`;Dimensions: X G90/AC in the diameter, G91/IC in the ra-dius`
`N70 Y200`	`;Y: continuing, axis-specific modal diameter programming`
`N75 G91 Y20 U=DIC(40)`	`;Dimensions: Y in the diameter, U non-modally IC in the diameter`
`N80 X50 Y100`	`;Dimensions: X in the radius (G91), Y in the diameter`
`N85 G90 X100 U200`	`;Dimensions: X in the diameter, U in the radius`
`N90 DIAMCHANA[Y]`	`;Y accepts the channel status DIAM90`
`N95 G91 X100 Y100`	`;Dimensions: X and Y in the radius(G91)`
`N100 G90 X200 Y200`	`;Dimensions: X and Y in diameter`

Example 2

Transverse axes with diameter specification applied as in the previous example.

X and Y are located in channel 1 and are also known in channel 2; i.e. permitted for axis replacement.

Program code	**Comment**
`Channel 1`	
`N10 G0 G90 X100 Y50`	`;no diameter programming is active`
`N20 DIAMON`	`;Channel-specific diameter programming for X`

Program code	Comment
`N30 Y200 X200`	`;Dimensions: X in the diameter, Y in the radius`
`N40 DIAMONA[Y]`	`;Y axis-specific modal diameter programming`
`N50 Y250 X300`	`;Dimensions: X and Y in diameter`
`N60 SETM(1)`	`;Synchronous marker 1`
`N70 WAIT(1,2)`	`;wait for synchronous marker 1 in channel 2`
`Channel 2`	
`...`	
`N50 DIAMOF`	`;channel 2 no diameter programming active`
`...`	
`N100 WAIT(1,1)`	`;wait for synchronous marker 1 in channel 1`
`N110 GETD(Y)`	`;Axis replacement direct Y`
`N120 Y100`	`;Y is the channel-specific diameter programming` `;subordinated in channel 2; i.e. dimension in the radius`

13.6 Data lists

13.6.1 Machine data

13.6.1.1 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]	Channel axis name in the channel
20100	DIAMETER_AX_DEF	Geometry axis with transverse axis function
20110	RESET_MODE_MASK	Definition of the control basic setting after power-up and RE-SET / part program end
20112	START_MODE_MASK	Definition of the control basic setting for NC start
20150	GCODE_RESET_VALUES[n]	Basic setting of the G groups
20152	GCODE_RESET_MODE[n]	G command basic setting at RESET / part program end
20360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
20624	HANDWH_CHAN_STOP_COND	Definition of the behavior of the manual handwheel traversing
27100	ABSBLOCK_FUNCTION_MASK	Parameterize block display with absolute values

13.6.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30460	BASE_FUNCTION_MASK	Axis functions

P3: Basic PLC program for SINUMERIK 840D sl 14

14.1 Brief description

General

The PLC basic program organizes the exchange of signals and data between the PLC user program and the NC, HMI and MCP areas. In the case of signals and data, a distinction is made between the following groups:

- Cyclic signal exchange
- Event-driven signal exchange
- Messages

Cyclic signal exchange

The cyclically-exchanged signals consist primarily of bit arrays.

- They contain **commands** transferred from the PLC to the NC (such as start or stop) and **status information** from the NC (such as program running, interrupted, etc.).
- The bit fields are organized into signals for:
  - Mode group
  - Channels
  - Axes/spindles
  - General NC 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 NC remain constant throughout a cycle.

Event-driven signal exchange NC → PLC

PLC functions that have to be executed as a function of the workpiece program are triggered by auxiliary functions in the workpiece program. If a block with auxiliary functions is executed, the type of auxiliary function determines whether the NC has to wait for this function to execute (e.g. tool change) or whether the function will be executed together with the workpiece machining process (e.g. tool loading on milling machines with chain magazine).

Data transfer must be as fast and yet as reliable as possible, in order to minimize the effect on the NC processing. Data transfer is, therefore, interrupt- and acknowledgment-driven. The basic program evaluates the signals and data, acknowledges this to the NC and transfers the data to the application interface at the start of the cycle. If the data does not require user acknowledgment, this does not affect NC processing.

Event-driven signal exchange PLC → NC

An "event driven signal exchange PLC → NC" takes place whenever the PLC transfers a request to the NC (e.g. traversing 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. The message signals are transferred to the basic program via a specified bit array. where they are evaluated and, if message events occur, entered in the PLC's interrupt buffer by means of the ALARM S/SQ functions. If an HMI (e.g. SINUMERIK Operate) is being used, the messages are transferred to the HMI and displayed.

PLC/HMI data exchange

In this type of data exchange, the HMI takes the initiative, being referred to as the "client" on the bus system. The HMI polls or writes data. The PLC processes these requests at the cycle control point via the operating system. The PLC basic program is not involved in these exchanges.

Note

The function of the machine is largely determined by the PLC program. Every PLC program in the RAM can be edited with the programming device.

Know-how protection for user blocks

To protect the know-how contained in the the user blocks (OB, FB and FC), they can be encoded with the SBP tool (SIMATIC block protection) contained in SIMATIC STEP 7. These blocks can then no longer be opened, debugged and modified without specifying the password for the encoding.

When encoding, the automation system on whose PLC-CPU the blocks are to be executed, must be specified: SIMATIC and/or SINUMERIK PLC-CPU.

The handling of the blocks, e.g. loading to the CPU, is not affected by the encoding.

Requirement

SIMATIC STEP 7 as of Version 5.5 SP3

14.2 Key data of the PLC CPU

Key data of the PLC CPU

References:
The overview of the key data of the PLC CPU integrated in the SINUMERIK NCU can be found in:
NCU 7x0.3 PN Manual, Section "Technical data"

Note

I/O addresses for integrated drives

The I/O addresses above 4096 are reserved for the integrated drives of the NCU and must not assigned otherwise.

Functions of the basic PLC program

Scope		
	Axes/spindles	31
	Channels	10
	Mode groups	10
Functions		
	Status/control signals	+
	M decoders (M00-99)	+
	G group decoders	+
	Aux. function distributors	+
	Aux. function transfer, interrupt-driven	+
	M decoding acc. to list	+
	Move axes/spindles from PLC	+
	ASUP interface	+
	Error/operating messages	+
	Transfer MCP and HHU signals	+
	Display control handheld unit	+
	Read/write NC variables and GUD	+
	PI services	+
	Tool management	+
	Star/delta switchover	+
M to N		+
Safety Integrated		+
Program diagnostics		+

14.3 PLC operating system version

The PLC operating system version is displayed at:

- User interface of SINUMERIK Operate: "Operating area switchover" > "Diagnostics" > "Version" ⇒ version data / system software NCU: Selection "PLC" > "Details" ⇒ version data / system software NCU/PLC: The PLC operating system version is displayed in the first line is at "PLC 3xx…".
  Note
  The displayed version is SINUMERIK-specific. It is not compatible with the basic SIMATIC CPU.
- SIMATIC STEP 7, HW Config: In the properties of the PLC CPU in the SINUMERIK rack: "Properties - CPU 3xx…" > "Order no. / firmware": xxxx / **Vx.y.z**
  Note
  The version of the basic SIMATIC CPU is displayed.

14.4 PLC mode selector

The PLC mode selector is located on the front of the NCU module. The following PLC operating modes can be set via the PLC mode selector:

S 1)	Meaning	Remark
0	RUN-P	The PLC program can be changed without activation of the password
1	RUN	Only read access operations are possible using a programming device (PG). It is not possible to make changes to the PLC program until the password has been set.
2	STOP	Processing the PLC program is stopped and all PLC outputs are set to substitute values.
3	MRES	The PLC is switched into the STOP state followed by a PLC general reset (default data).
1) Switch position of the PLC mode selector		

References

A detailed description of the position of the PLC mode selector on the front of the NCU module, as well as its use in connection with NC and PLC general reset can be found in:

CNC Commissioning Manual: NC, PLC, Drive:

- Section "Switch-on/power-up" > "Operator control and display elements for power-up"
- Section "Switch-on/power-up" > "NC and PLC general reset"
- Section "General tips" > "Separate NC and PLC general reset"

14.5 Reserve resources (timers, counters, FC, FB, DB, I/O)

Reserve resources (timers, counters, FC, FB, DB, I/O)

The components below are reserved for the basic program:

- **Timer**
  No reservation
- **Counter**
  No reservation
- **FC, FB, DB**
  FC0 to FC29 and FB0 to FB29 are reserved for the basic program. The number range between 1000 and 1023 is also reserved for FCs and FBs. DB1 to DB61, DB71 to DB80 are reserved for data blocks. The number range 1000 to 1099 is also reserved in addition for DB. The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as desired by the user.
- **I/O range**
  The PLC has an I/O address volume of 16384 bytes each for inputs and outputs. The address ranges starting at 4096 are reserved for/occupied by integrated drives. However, diagnostic addresses for modules can be assigned to the highest address range as proposed by STEP 7. The address range between 4080 and 4097 is also assigned for the NC, CP and HMI in rack 0 of the SIMATIC 300 station (for NCU 7x0.3).

14.6 Commissioning hardware configuration of the PLC CPU

The commissioning of the PLC CPU is described in detail in:

References

CNC Commissioning Manual: NC, PLC, Drive:

- Section: "Connect PG/PC to PLC"
- Section: "Commissioning PLC"
- Section: "Basics" > "PLC program"
- Section: "General tips" > "Separate NC and PLC general reset"
- Section: "General tips" > "Integrating PG/PC into the network (NetPro)"

14.7 Starting up the PLC program

14.7.1 Installation of the basic program

The installation of the basic program is described in detail in:

References

CNC Commissioning Manual: NC, PLC, Drive; Section: "Commissioning PLC" > "Creating a PLC program"

Note

Installation/update

Before installing the toolbox for SINUMERIK 840D sl, SIMATIC STEP 7 must be installed.

It is recommended that the hardware expansions for STEP 7 be installed again from the toolbox after an update of STEP 7.

Contents

The OB source programs, including standard parameterization, interface symbols and data-block templates for the handheld unit and M decoding functions are included in the basic program.

14.7.2 Application of the basic program

A new CPU program (e.g. "Turnma1") must be set up in a project by means of the STEP 7 software for each installation (machine).

Remark

The catalog structures of a project and the procedure for creating projects and user programs are described in the relevant SIMATIC documentation.

Procedure

The basic program blocks are copied using the SIMATIC Manager and "File" > "Open" > "Library".

The following components must be copied from the library:

- From the block container: FCs, FBs, DBs, OBs, SFC, SFB, UDT
- The source_files (from the source container): GPOB840D
- Possibly MDECLIST, HHU_DB and others
- The symbols table (from the symbols container)

Compatibility with STEP 7

There are no dependencies between the basic program and current STEP 7 versions.

14.7.3 Version codes

Basic program

The version of the basic program is displayed on the Version screen of the user interface along with the control system type.

The control system type is encoded as follows:

Leftjustified decade of DB17.DBD0 (byte 0)	Control system type
03	SINUMERIK 840D sl (NCU 7x0)

User program version identifications

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 version, however, is not interpreted, but rather the entered string is accepted. The parameterization on this string is done via a pointer on FB1. For this, the data block must be defined symbolically. See the FB1 (Page 965) block description for more information.

The version identification can be formatted in the string as follows:

- xx.yy
- xx.yy.zz
- ww.xx.yy.zz
- vv.ww.xx.yy.zzz
- x.y
- x.y.z
- w.x.y.z
- v.w.x.y.z

In addition to the version identification, a date can be entered in the string, which given the appropriate formatting, is displayed in the HMI in the version screen. The date, however, is not interpreted, but rather the entered string is accepted. A combination with version identifications is possible. The following formats are currently supported:

- 00/00/0000
- 0000/00/00
- 00/00/00

Examples:

- "Test project version 01.02.03 01/01/2015"
- "1.2 2015/01/01 test project"
- "01/01/15 version 01.02 test project"

14.7.4 Machine program

The machine manufacturer creates the machine program using the library routines supplied with the basic program. The machine program contains the logic operations and sequences on the machine. The interface signals to the NC are also controlled in this program. More complex communication functions with the NC, e.g. read/write NC data, tool-management acknowledgments, etc., are activated and executed via the FCs and FBs blocks of the basic program.

The machine program can be created in various STEP 7 creation languages, e.g. STL, LAD, 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 these** blocks.

If blocks that are called by other blocks are subsequently modified in the interface (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) as the program is developed, then the call block and all blocks associated with it must be compiled again. This general procedure applies analogously to instance data blocks for FBs. If this sequence of operations is not observed, timestamp conflicts occur when the data is retranslated into STEP 7. As such, the recompilability of the blocks is not ensured and with the function "Status of block" unnecessary conflicts can also appear. It is, moreover, advisable to generate blocks in ASCII-STL by means of the STEP 7 editor when they have been created in Ladder Diagram or in single statements (incremental mode).

14.7.5 Data backup

The PLC-CPU does not save any symbolic names, but instead only the datatype descriptions of the block parameters VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR and the datatypes of the global data blocks.

Note

No sensible recompilation is possible without the related project for this machine. This especially affects, for instance the function status of the block or the necessary changes done in the PLC-CPU programs later. It is, therefore, necessary to keep a backup copy of the STEP 7 project located in the PLC CPU on the machine. This is a great help for the service case and saves unnecessary consumption of time in restoring the original project.

If the STEP 7 project exists and has been created according to the instructions given above, then symbols can be processed in the PLCCPU on this machine. It may also be advisable to store the machine source programs as ".awl" files in case they are required for any future upgrade.

The source programs of all organization blocks and all instance data blocks should always be available.

14.7.6 PLC series startup, PLC archive

Once the blocks have been loaded to the PLC CPU, a series archive can be generated via the HMI user interface to back up data on the machine. To ensure data consistency, this backup must be created immediately after block loading when the PLC is in the Stop state. It does not replace the SIMATIC project backup as the series archive saves binary data only. For instance, no symbolic information is present here. In addition, no CPU DBs (SFC 22 DBs) or SDBs generated in the CPU are saved.

Selection of the SINUMERIK archiving program

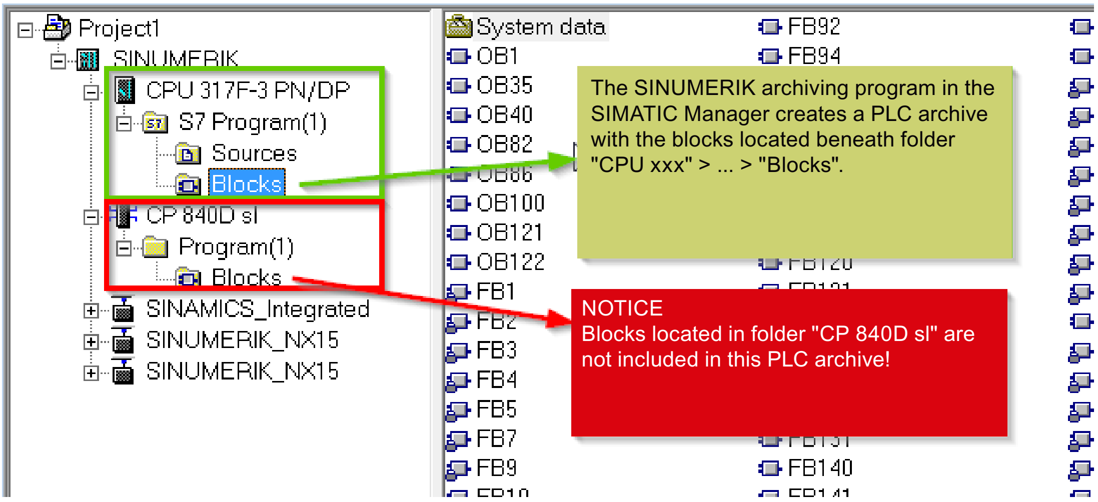
The PLC series archive can be generated directly from the SIMATIC project as an alternative.

1. In the SIMATIC Manager open dialog box "Settings": menu bar "Tools" > "Settings".
2. Open tab "Archiving".
3. In the "Preferred archiving program" drop-down list box, select the SINUMERIK archiving program "SINUMERIK (*.arc)".

Start of the SINUMERIK archiving program

The SINUMERIK archiving program is started in the SIMATIC Manager via the menu command "File" > "Archive".

After you have selected a project and assigned an archive name, the PLC archive is generated. If a project contains several program paths, the S7 program for which the PLC archive will be created can be selected in the dialog box. All blocks contained in folder "CPU ..." > ... > "Blocks" at the selected program path are archived. Blocks in the folder "CP 840D sl" are not included in the PLC archive:

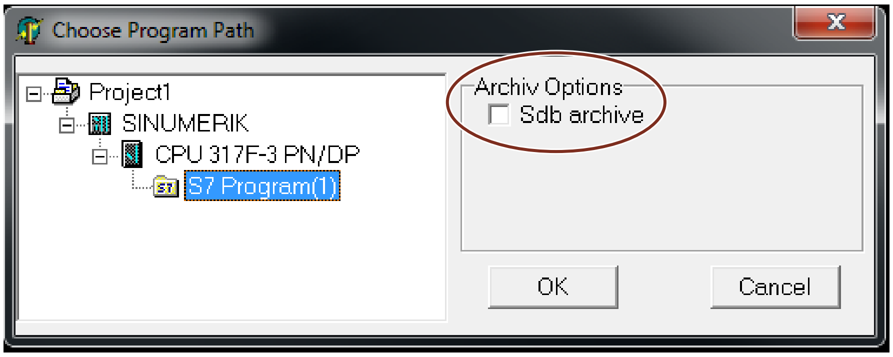


The SINUMERIK archiving program in the SIMATIC Manager creates a PLC archive with the blocks located beneath folder "CPU xxx" > ... > "Blocks".

NOTICE
Blocks located in folder "CP 840D sl" are not included in this PLC archive!

Data blocks that were created with SFC22 (online) in the CPU are also excluded from the archive.

The "Sdb archive" option can be activated or deactivated for the archiving program:



If "Sdb archive" is activated, a PLC archive is created that only contains the system data blocks (SDB) of the selected program path.

Automation

The process of generating a series archive can be automated (comparable to the command interface in STEP 7). In generating this series archive, the command interface is expanded.

The following functions are available for this expansion:

The functions (shown here in VB script) are not available until server instantiations and Magic have been called:

```
Const S7BlockContainer = 1138689, S7PlanContainer = 17829889

Const S7SourceContainer = 1122308

set S7 = CreateObject("Simatic.Simatic.1")

rem Instantiate command interface of STEP 7

Set S7Ext = CreateObject("SimaticExt.S7ContainerExt")

Call S7Ext.Magic("")
```

Functions:

- `Function Magic(bstrVal As String) As Long`
- `Function MakeSerienIB (FileName As String, Option As Long, Container As S7Container) As Long`

Description

Function **Magic**(bstrVal As String) As Long

The call provides access to certain functions. The function must be called once after server instantiation. The value of bstrVal can be empty. This initiates a check of the correct STEP 7 version and path name in Autoexec. The functions are enabled with a return parameter of 0.

Return parameter (-1) = incorrect STEP 7 version

Return parameter (-2) = no entry in Autoexec.bat

Function **MakeSerienIB**(FileName As String, Option As Long, Container As S7Container) As Long

Parameter "Option":

0:	Normal series startup file with general reset.
Bit 0 = 1:	Series startup file without general reset. When project contains SDBs, this option is inoperative. A general reset is then always executed
Bit 1 = 1:	Series startup file with PLC restart

Return value:

0	= OK
-1	= Function unavailable, call Magic function beforehand
-2	= File name cannot be generated
-4	= Container parameter invalid or container block empty
-5	= Internal error (memory request rejected by Windows)
-6	= Internal error (problem in STEP 7 project)
-7	= Write error when generating series startup files (e.g. diskette full)

Use in script

Program code

```
If S7Ext.Magic("") < 0 Then
    Wscript.Quit(1)
End If
    Set Proj1 = s7.Projects("new")
    set S7Prog = Nothing
    Set s7prog = Proj1.Programs.Item(1) 'if there is only one program'
For Each cont In s7prog.Next
    If (Cont.ConcreteType = S7BlockContainer) Then
    'Check block container
    Exit For
    End if
    Cont = Nothing
    Next
Error = S7Ext.MakeSerienIB("f:\dh\arc.dir\PLC.arc", 0, Cont)
'Now error analysis
```

The `For Each` ... `Next` block programmed above can be programmed in the Delphi programming language as follows (The programming in the C, C++ programming languages is similar):

Program code

```
Var
    EnumVar: IEnumVariant;
    rgvar: OleVariant;
    fetched: Cardinal;
```

Program code

```
//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;
```

14.7.7 Software upgrade

A general PLC reset should be performed to achieve a defined initial state before the PLC software is upgraded. In this case, among other things, all user data (program and data blocks) will be deleted. The PLC general reset is described in:

References:
Commissioning Manual CNC: NC, PLC, Drive, General Tips,
Section: PLC general reset

Generating a new SIMATIC S7 project

In normal cases, the new PLC basic program is to be linked-in for a new NCU software version. The basic programs blocks must be loaded into the user project for this purpose. If the following program and data blocks are already in the user project, then these should not be transferred with the blocks of the basic PLC program: OB1, OB40, OB82, OB86, OB100, FC12 and DB4. These have been possibly changed by the user, and should not be overwritten. The new basic program must be linked with the user program. The following procedure must be taken into account:

1. Generate the text or source file of all user blocks before copying the basic PLC program.
2. Copy the new basic program blocks into the SIMATIC S7 project (for a description, see Section "Application of the basic program (Page 874)")
3. All user programs "*.awl" must be recompiled in the correct order! (See also "Machine program (Page 876)")
4. This newly compiled SIMATIC S7 project should then be downloaded with STEP 7 into the PLC.

However, it is normally sufficient to recompile the organization blocks (OBs) and the instance data blocks of the S7 project. This means before upgrading, only the sources for the organization blocks and the instance data blocks have to be generated.

NC variables

The latest NC VAR selector can be used for each NC software version (even earlier versions). The variables can also be selected from the latest list for earlier NC software versions. The data content in DB120 (default DB for variables) does not depend on the software status. This means, variables selected in an older software version need not be reselected when the software is upgraded.

14.7.8 I/O modules (FM, CP modules)

Additional packages for STEP 7 are generally required for more complex I/O modules (FM, CP modules). Support blocks (FC/FB) are provided in these additional packets. The blocks contain specific functions for operating the relevant module. These functions can be parameterized and called in the user program.

Identical numbers

If handling and basic program blocks have identical numbers, the block numbers of the basic program must remain unchanged. The block numbers of the handling blocks must be renamed to free numbers via STEP 7.

14.7.9 Troubleshooting

This section describes problems which may occur, their causes and remedies and should be read carefully before hardware is replaced.

Errors, cause/description and remedy			
Serial no. error information	**Errors**	**Cause/description**	**To correct or avoid errors**
1	No connection via MPI to PLC.	The MPI cable is not plugged in or is defective. Possibly, the STEP 7 software is also not correctly configured for the MPI card.	Test: Create a link with the programmer in the STEP 7 editor by means of connection "Direct_PLC". A number of node addresses must be displayed here. If they do not appear, the MPI cable is defective/not plugged in.
2	PLC cannot be accessed in spite of PLC general reset.	A system data block SDB 0 has been loaded with a modified MPI address. This has caused an MPI bus conflict due to dual assignment of addresses.	Disconnect all MPI cables to other components. Create the link "Direct_PLC" with the programmer. Correct the MPI address.
3	All four LEDs on the PLC flash (DI disaster)	A system error has occurred in the PLC. **Measures:** The diagnostic buffer on the PLC must be read to analyze the system error in detail. To access the buffer, the PLC must be stopped (e.g. set "PLC" switch to position 2). A hardware reset must then be performed. The diagnostic buffer can then be read out with STEP 7. Relay the information from the diagnostic buffer to the Hotline / Development Service. A general reset must be carried out if requested after the hardware RESET. The diagnostic buffer can then be read with the PLC in the Stop state.	Once the PLC program has been RESET or reloaded, the system may return to normal operation. Even in this case, the content of the diagnostic buffer should be sent to the Development Office.

14.8 Coupling of the PLC CPU

14.8.1 General information

A CPU of the S7-300 automation system is used as PLC for the SINUMERIK 840D sl. The PLC-CPU is integrated into the NCU component as a sub-module. A reference to the performance data of the PLC CPU can be found in Section "Key data of the PLC CPU (Page 871)".

14.8.2 Properties of the PLC CPU

The PLC integrated in the SINUMERIK 840D sl generally has the same functionality as the corresponding SIMATIC S7-300 PLC.

For differences, see reference in Section "Key data of the PLC CPU (Page 871)".

Owing to differences in their memory system as compared to a SIMATIC S7-300 PLC, certain functions are not available (e.g. save blocks on memory card, save project on memory card).

Note

As with the PLC integrated in SINUMERIK, there is no automatic start of the PLC after power failure and recovery for a SIMATIC S7-300 PLC when a "PLC stop" is triggered by an operator action on the programming device. For safety reasons, the PLC remains in the stop state with an appropriate diagnostic entry. You can start the PLC only by an operator action on the programming device, "Execute a restart", or via the mode selector "Stop" > "Run" (warm restart).

14.8.3 Interface with integrated PLC

Physical interfaces

With the SINUMERIK 840D sl, the PLC integrated in the NCU offers the option of exchanging signals between the NC and PLC directly via a dual-port RAM.

Data exchange with the operator panel

Data exchange with the operator panel (e.g. TCU/OP) can be performed via Ethernet or PROFIBUS. With a connection via Ethernet, communication takes place via the integrated communication processor (CP 840D sl).

Data exchange with the machine control panel (MCP) and handheld unit (HHU) can be performed via MPI, PROFIBUS or Ethernet.

Programming devices should preferably be connected via Ethernet or via MPI (Multi-Point Interface) directly to the PLC.

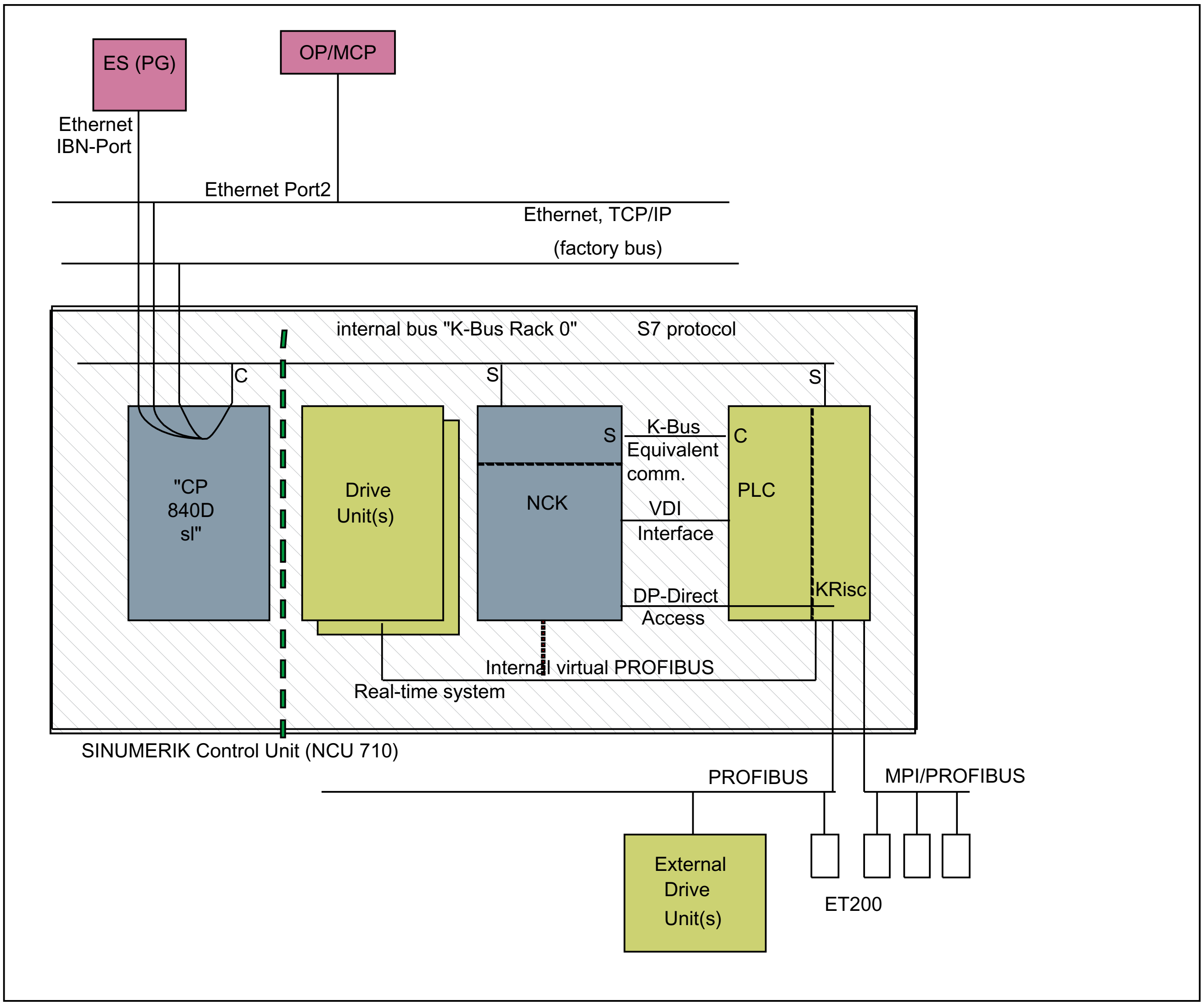


Figure 14-1 NC-PLC coupling on SINUMERIK 840D sl (integrated PLC)

Interface: NC/PLC

The data exchange between NC and PLC is organized by the basic program on the PLC side. The **status information**, such as "Program running", stored by the NC in the NC/PLC interface is copied to data blocks by the basic program at the beginning of the cycle (OB1) and can then be accessed in the user program (user interface). The **control signals** for the NC (e.g. NC start) entered in the interface data block by the user are also written to the internal DPR and transferred to the NC at the start of the cycle.

Workpiece-program-specific **auxiliary functions** transferred to the PLC 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 that NC processing is interrupted (e.g. M06 for tool change), the basic program stops the decoding of the NC block initially for one PLC cycle. The user can then use the "read disable" interface signal to halt the block execution until the tool change has been completed. If, on the other hand, the relevant NC block only contains auxiliary functions, which do not require interruption of the decoding (e.g. M08 for cooling medium on), the transfer of these "fast" auxiliary functions is directly acknowledged in OB40, so that decoding is only insignificantly influenced by the transfer to the PLC.

The evaluation and enabling of the **G commands** transferred from the NC are also alarm-driven and acknowledged, however they are transferred directly to the user interface. If a G command is evaluated at several points in the PLC program, differences in the information of the G command within one PLC cycle may arise.

In the case of **NC actions** triggered and assigned with parameters by the PLC (e.g. traverse concurrent axes), triggering and parameter assignment is performed using FCs and FBs, not interface data blocks. The FCs and FBs belonging to the actions are supplied together with the basic program. The FCs and FBs required must be loaded by the user and called in the PLC program of the machine manufacturer (machine program). For an overview of FC, FB and data blocks, sorted according to basic and extended functions, please refer to Section "Start-up of PLC programs".

Interface: HMI/PLC

HMI/PLC data exchange is performed via the integrated CP, whereby the HMI is always the active partner (client) and the PLC is always the passive partner (server). Data transferred or requested by the HMI is read from and written to the HMI/PLC interface area by the PLC operating system (timing: Cycle control point). From the viewpoint of the PLC application, the data is identical to I/O signals.

Interface: MCP/PLC or HHU/PLC (connection: Ethernet)

MCP/PLC and HHU (HT 2) / PLC data is exchanged via the integrated CP. The CP transfers the MCP/HHU signals to and fetches them from the PLC's internal DPR (Dual-Port RAM). On the PLC side, the basic program handles communication with the user interface. The basic-program parameters (FB1, DB7) define the operand areas (e.g. I/O areas) and the start addresses.

Interface: MCP/PLC (connection: PROFIBUS)

MCP/PLC data exchange takes place via the PLC's PROFIBUS. The MCP's I/O addresses are to be placed in the PLC's process image area and via HW configuration in STEP 7. The MCP*In, MCP*Out pointer variables must be set to the same addresses. The selected DP slave number must be entered in MCP*BusAdr.

Interface: HHU/PLC (connection: MPI)

The HHU/PLC data exchange is performed via the MPI interface on the PLC. The "Communication with global data (GD)" service is used for this purpose (see also STEP 7 User Manual). The PLC operating system handles the transfer of signals from and to the user interface. The STEP 7 "Communication configuration" configuring tool is used to define both GD parameters as well as operand areas (e.g. I/O areas) and their start addresses.

14.8.4 Diagnostic buffer on PLC

The diagnostic buffer of the PLC (readable using STEP 7) will enter diagnostic information on the PLC operating system.

14.9 Interface structure

Interface DBs

Mapping in interface DBs is necessary due to the large number of signals exchanged between the NC and PLC. These are global DBs from the viewpoint of the PLC program. During system start-up, 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.

14.9.1 PLC/NCK interface

General

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

Data interface

The data interface is subdivided into the following groups:

- NC-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 NC.

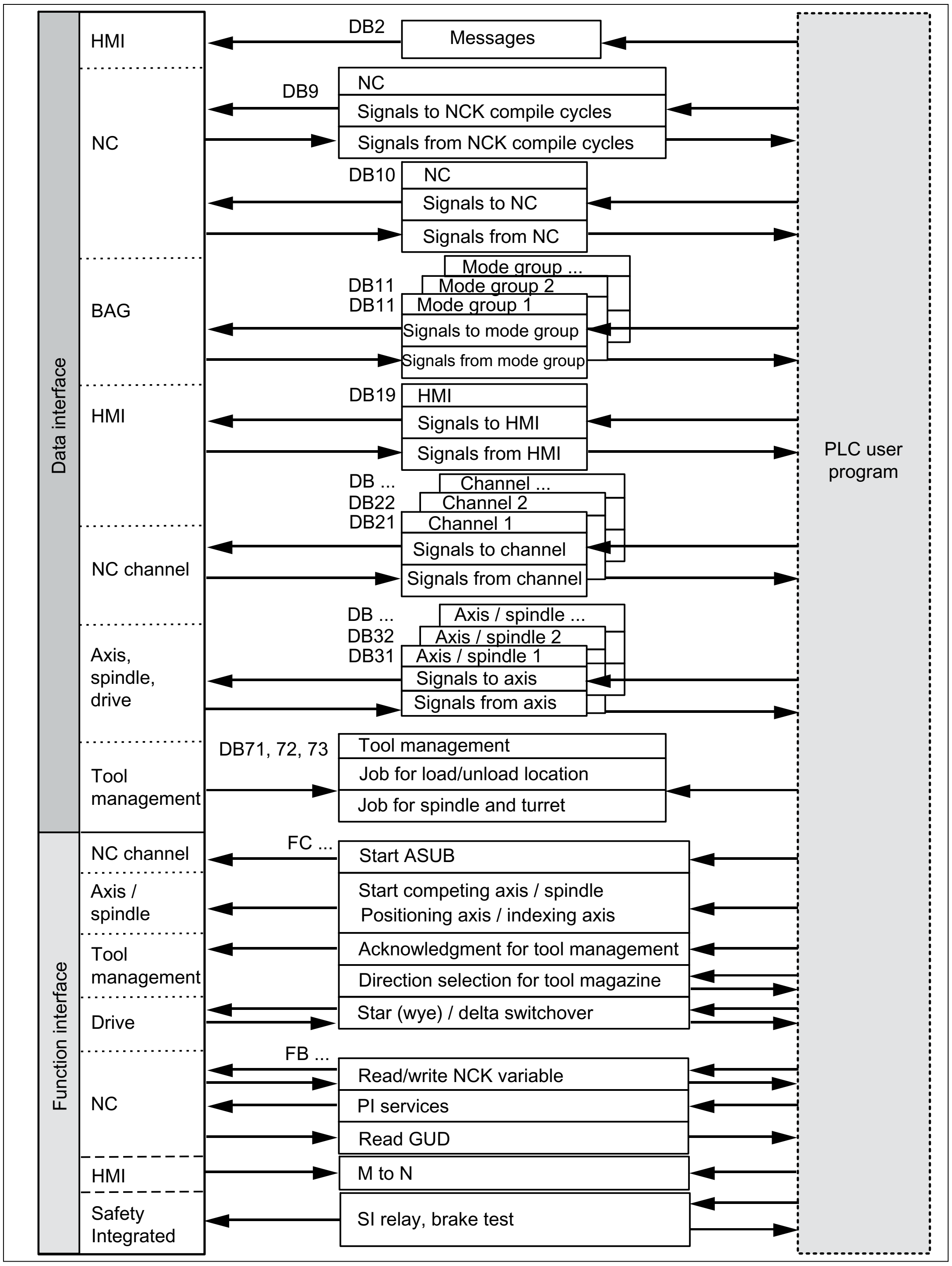


Figure 14-2 PLC/NC user interface

Compile-cycle signals

In addition to the standard signals exchanged between the PLC and NC, an interface data block for compile cycles is also generated if required (DB9). The associated signals, which are dependent on the compile cycles, are transmitted cyclically at the start of OB1. The basic program starts transmission at the lowest address and works up to the highest. First, signals are transferred from the PLC to the NC, then from the NC to the PLC. The user must synchronize the NC and PLC as necessary (e.g. using the semaphore technique). Signal transmission is asynchronous between NC and PLC. This means, for example, that active NC data transmission can be interrupted by the PLC. This can mean that data is not always consistent.

PLC/NC signals

The group of signals from the PLC to NC includes:

- Signals for modifying the high-speed digital I/O signals of the NC
- Keyswitch and emergency stop signals

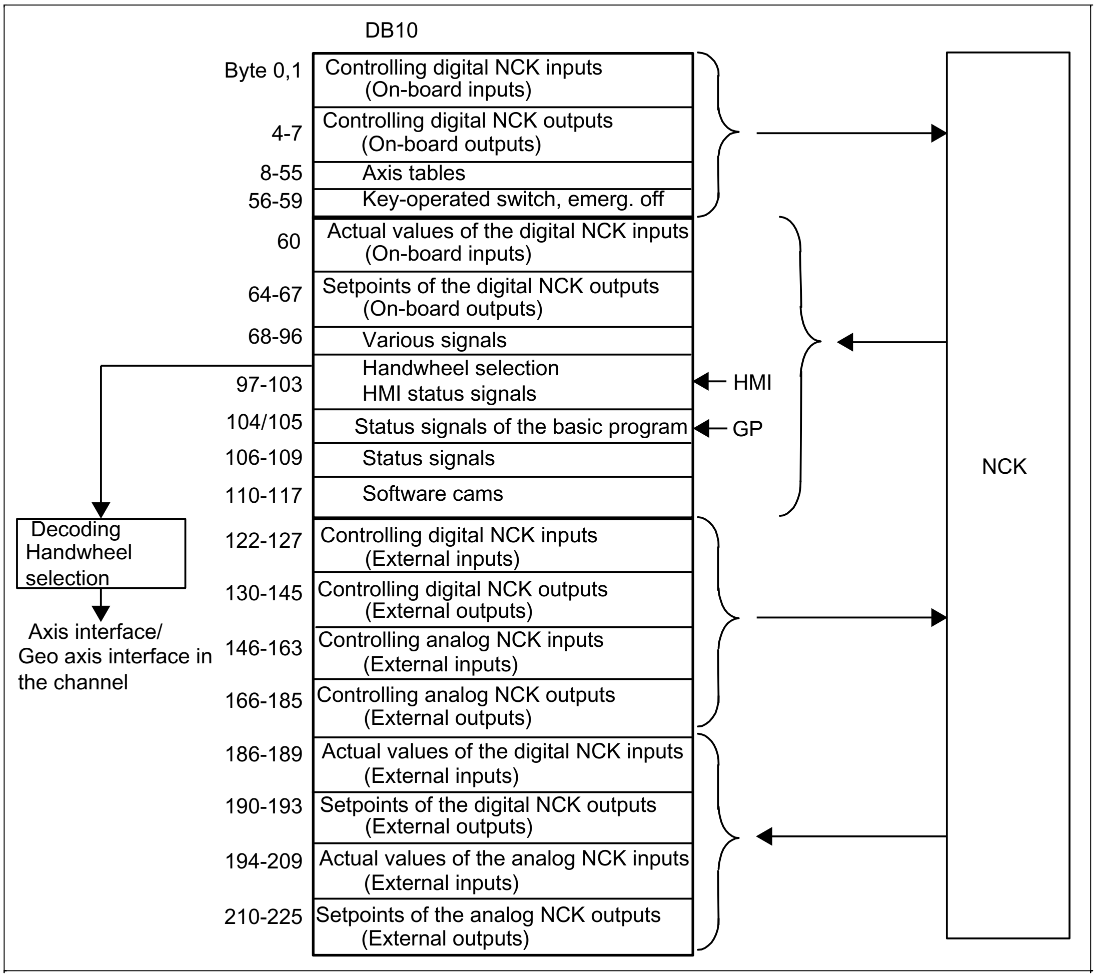

DB10
Byte 0,1
Controlling digital NCK inputs (On-board inputs)
4-7
Controlling digital NCK outputs (On-board outputs)
8-55
Axis tables
56-59
Key-operated switch, emerg. off
60
Actual values of the digital NCK inputs (On-board inputs)
64-67
Setpoints of the digital NCK outputs (On-board outputs)
68-96
Various signals
97-103
Handwheel selection
HMI status signals
HMI
104/105
Status signals of the basic program
GP
106-109
Status signals
110-117
Software cams
122-127
Controlling digital NCK inputs (External inputs)
130-145
Controlling digital NCK outputs (External outputs)
146-163
Controlling analog NCK inputs (External inputs)
166-185
Controlling analog NCK outputs (External outputs)
186-189
Actual values of the digital NCK inputs (External inputs)
190-193
Setpoints of the digital NCK outputs (External outputs)
194-209
Actual values of the analog NCK inputs (External inputs)
210-225
Setpoints of the analog NCK outputs (External outputs)
Decoding Handwheel selection
Axis interface/ Geo axis interface in the channel
NCK

Figure 14-3 PLC/NC interface

NC/PLC signals

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 stored in this group are the HMI handwheel selection signals and the channel 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 NC

The following must be noted with respect to the digital and analog inputs and outputs of the NC:

Inputs:

- All input signals or input values of the NC are also transferred to the PLC.
- The transfer of signals to the NC part program can be suppressed by the PLC. Instead, a signal or value can be specified by the PLC.
- The PLC can also transfer a signal or value to the NC even if there is no hardware for this channel on the NC side.

Outputs:

- All signals or values to be output are also transferred to the PLC.
- The NC can also transfer signals or values to the PLC even if there is no hardware for this channel on the NC side.
- The values transferred by the NC can be overwritten by the PLC.
- Signals and values from the PLC can also be output directly via the NC I/O.

Note

When implementing the digital and analog NC I/O, the information contained in the following documentation must be taken into account:
References:
Function Manual Extended Functions; Digital and analog NC I/O (A4)

PLC / mode group signals

The operating mode signals set by the machine control panel or the HMI are transferred to the operating mode group of the NC. These apply to all NC channels. Several mode groups can be optionally defined in the NC.

The mode group reports its current status to the PLC.

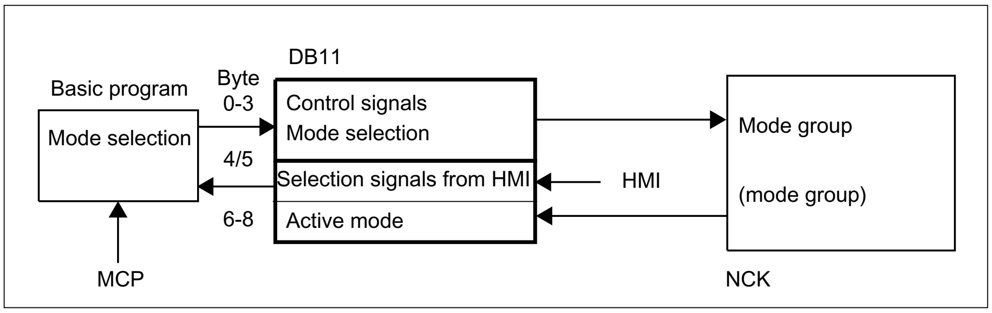


Figure 14-4 PLC / mode group interface

PLC/NC signals

The signal groups below must be considered on the interface:

- Control/status signals
- Auxiliary commands / G commands
- Tool management signals
- NC functions

The **control/status signals** are transferred cyclically at the start of OB1. The signals entered in the channel-specific interface by the HMI (HMI signals are entered by the PLC operating system) are also transferred at this time if they have been defined on the HMI operator panel, not on the MCP.

Auxiliary commands and G commands are entered in the interface data blocks in two ways. First, they are entered with the change signals.

- The **M signals** M00 - M99 (they are transferred from the NC with extended address 0) are also decoded and the associated interface bits set for the duration of one cycle.
- For **G commands**, only the groups selected via machine data are entered in the interface data block.
- The **S values** are also entered together with the related M signals (M03, M04, M05) in the spindle-specific interface. The axis-specific feedrates are also entered in the appropriate axisspecific interface.

When the **tool management (magazine management)** function is activated in the NC, 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 **NC functions** is performed by means of PLC function calls.
The following function calls are available:

- Position a linear axis or rotary axis
- Position an indexing axis
- Start a prepared asynchronous subprogram (ASUP)
- Read/write NC variables
- Update magazine and tool motion

Some of the above functions are described in their own function documentation.

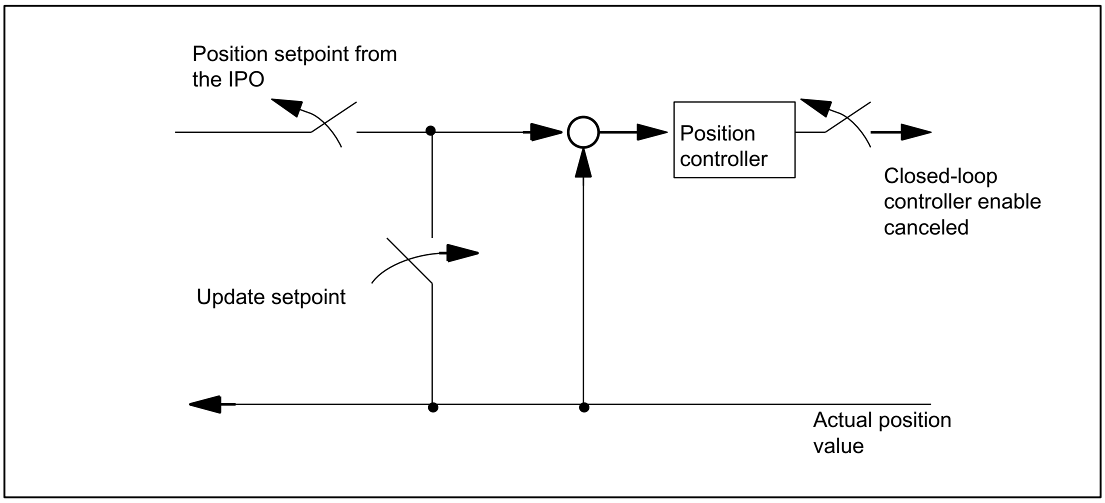

Position setpoint from the IPO
Position controller
Closed-loop controller enable canceled
Update setpoint
Actual position value

Figure 14-5 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 OB1 with the following exceptions:

Exceptions include:

- axis-specific F value
- M value
- S value

An **axis-specific F value** is entered via the M, S, F distributor of the basic program if it is transferred to the PLC during the NC program processing.

The **M and S value** are also entered via the M, S, F distributor of the basic program if one or both values require processing.

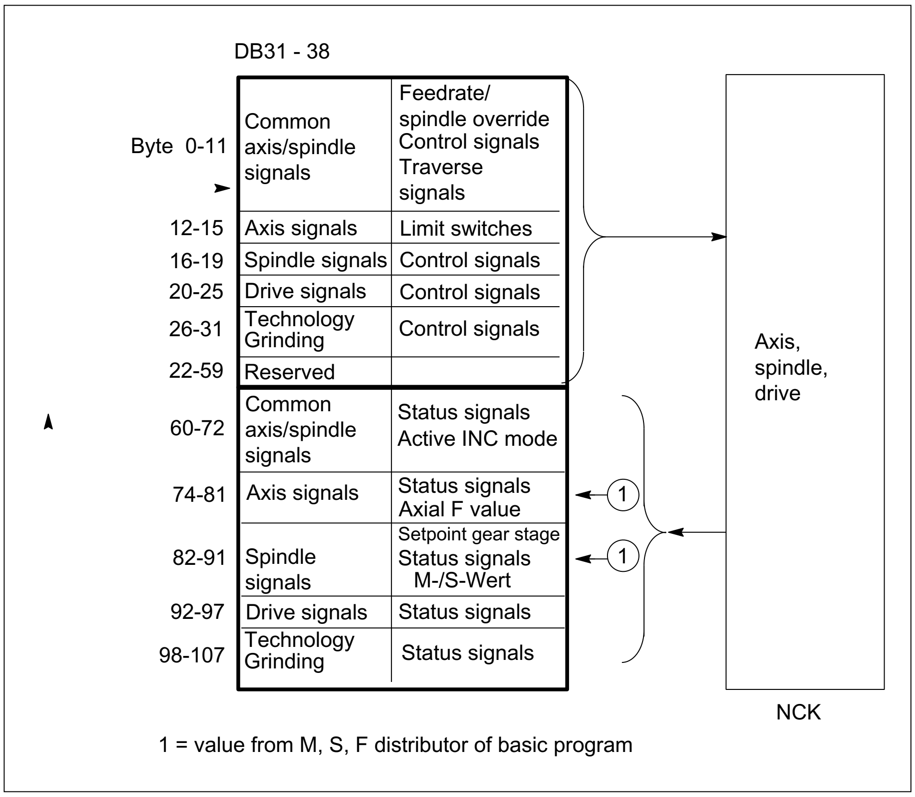


Figure 14-6 Interface between PLC and axes/spindles/drives

14.9.2 Interface PLC/HMI

General

The following groups of functions are required for the PLC/HMI interface:

- Control signals
- Machine operation
- PLC messages
- PLC status display

Control signals

Some control signals are signal inputs, for example, via the machine control panel, which have to be taken into account by the HMI. This group of signals includes, for example, display actual values in MCS or WCS, key disable, etc. These are exchanged with the HMI via a separate interface data block (DB19).

Machine operation

All operator inputs, which lead to response actions on the machine, are monitored by the PLC. Operator actions are usually performed on the machine control panel (MCP). However, it is also possible to perform some operator actions on the HMI, e.g. mode selection.

The PLC operating system enters the operating signals sent by the HMI directly into the interface data blocks. As standard, the basic program routes these operating signals in such a way that, provided equivalent operator actions are available, these can be performed either on the HMI **or** on the MCP. If required, the user can switch off operation via HMI using 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 HMI) via the bus systems once these have issued a ready signal (message service). "Transfer to the node ready" is a function of the PLC operating system. Receipt and interpretation of the messages are executed by the HMI software.
- The PLC user program can also use SFCs (System Function Calls) to enter messages in the diagnostic buffer or ALARM S/ALARM SQ buffer.
- The events are entered in the alarm buffer.
  The associated message texts must be stored on the OP or HMI.

An FC (FC10) 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 alarm buffer.

The message acquisition structure is shown in the figure "Acquisition and signaling of PLC events". The features include:

- Bit fields for events related to the NC/PLC interface are combined in a single data block (DB2) with bit fields for user messages.
- Bit fields are evaluated at several levels by FC10.
  - **Evaluation 1;** acquisition of **group signals**
    A group signal is generated for each group of signals if at least one bit signal is set to "1". This signal is generally linked to the disable signal of the NC/PLC interface (on modules with diagnostic functions). The group signals are acquired completely in cycles.
  - **Evaluation 2;** acquisition of **alarm messages**
    A fixed specification exists to define which signals in a group generate an alarm 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 FB1.

Acknowledgement concept

The following acknowledgement procedures are implemented for error and operational messages:

Operating messages are intended for the display of normal operating states as information for the user. Acknowledgement signals are, therefore, not required for this type of message. An entry is made in the diagnostic status list for incoming and outgoing messages. The HMI maintains an up-to-date log of existing operating messages using the identifiers "operating message arrived" and "operating message gone".

Alarm messages 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 alarm 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 "Alarm gone". This enables the HMI to also maintain an up-to-date log of pending alarm messages. The time of day indicating the time at which the error occurred is maintained for messages, which are still pending (in contrast to a received interrogation).

STEP 7

A tool can be started in the SIMATIC Manager via menu item "Target system" > "CPU messages". Alarms and messages can be displayed by number using this tool. To do this, acivate the "Alarm" tab and enter a check mark under "A" in the upper half of the screen.

User program

The user PLC program merely needs to call the basic program block FC10 with appropriate parameter settings in the cyclic program section and set or reset the bit fields in DB2. All further necessary measures are implemented by the basic program and HMI.

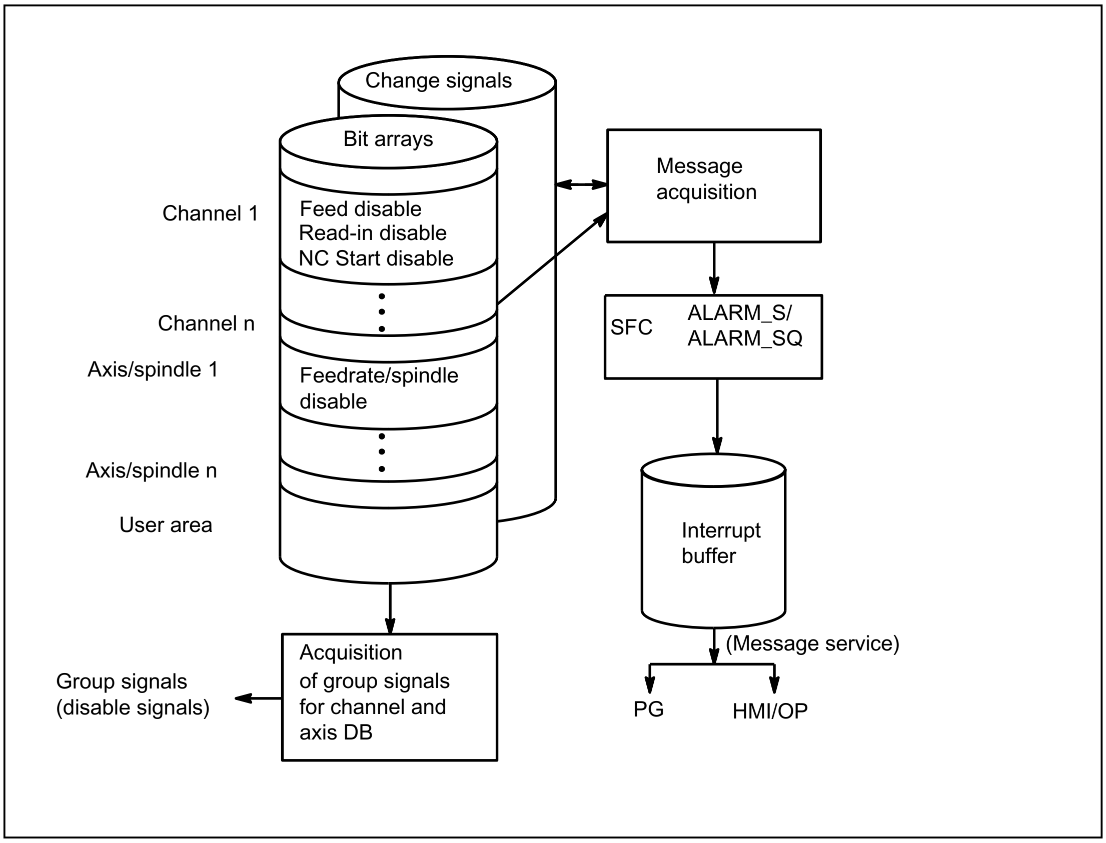


Figure 14-7 Acquisition and signaling of PLC events

Extensions of the PLC alarms via block FC10

The FB1 parameter "ExtendAlMsg" selects the PLC alarm mechanism.

If "ExtendAlMsg:= FALSE" the earlier process of the FC10 with the DB2 is active as bit array data block. The known restrictions regarding the number of channels and axes are applicable.

On the other hand, in case of "ExtendAlMsg:= TRUE" the extension of the FC10 becomes active. DB2 and DB3 are created just as before. The user must set or reset the bits in DB2. The parameter setting via message and alarm and a parameter setting of the numeric value of the 2nd decade of the user alarms are contained in DB5.

The extensions are:

- Support for 10 channels, 31 axes, 64 user areas (the number of user areas should be entered in the FB1 parameter "MsgUser").
- Areas for feed stop, read-in disable, etc. are available without messages. The information from this area is stored on the interface in DB21, DB31 depending on FC10 parameter "ToUserIF" together with the related message bits as group signals. As such, the previous cumbersome handling of the signals is omitted.
- The alarms / messages also get the 16-bit integer additional value (%Z parameter in the alarm text) in addition to the alarm number for the user area 0. The user must write the 16-bit integer values in the DB2 in the array variable ZInfo0 parallel to setting an alarm bit. An integer value is available for each bit in the user area 0, see UDT1002 in the basic program.
- The user messages can be parameteized in the second decade of the message number in the numerical range 0 to 9. The display value of the second decade must be written by the user in the DB5 in the array variable UserDek2No. A number can be defined for each user area, see DB5 in the basic program.
  The value 0 is set by default for the second decade.

The structuring of the DB2 in UDT1002 can be recognized (basic program). In case of new alarm functions, the UDT1002 must be assigned symbolically to the DB2.

At the start of DB2 there are bit arrays for signals without alarm generation. This is followed by an array of size 64 integer values for additional info about the user area 0.

Thereafter follow the areas, which also trigger alams / messages (see List manual) These areas are extended to 10 channels, 31 axes.

Simple implementation of a user program on the new alarms

The source container of the basic program contains the file "udt2_for_Convert.awl", which has the following structural element from UDT1002:

- ChanA as array of 1 ... 8
- AxisA as array of 1 ... 18
- UserA as array of 0 ... 63

This UDT2 is to be compiled via the LAD/FBD/STL editor. The UDT2 must be assigned to the DB2 in the symbol table.

Sources must be generated for components, which have assignments on DB2. Alternatively, sources can naturally be created for all blocks too. The UDT1002 must now be assigned to the DB2 in the symbol table. Thereafter, the sources must be recompiled.

Now all the alarm allocations are assigned to the new data areas in the DB2 and now only the parameter "ExtendAlMsg" at FB1 must be set to True.

After a Power On RESET the alarm behavior is the same as earlier.

14.9.3 PLC/MCP/HHU interface

General

There are different connection options for the machine control panel (MCP) and the handheld unit (HHU). This is in part due to the history of the MCP and HHU. This description focuses primarily on the connection of the Ethernet components.

On the SINUMERIK 840D sl, the machine control panel (MCP) and handheld unit (HHU) are connected via the Ethernet bus, which also links the TCU to the NCU. The advantage of this is that only one bus cable is required to connect the operating unit.

Topology SINUMERIK 840D sl

On the 840 D, the machine control panel and the handheld unit are connected to the CP 840D sl Ethernet bus (see Figure below). Where the connection of further keys and displays is required for customized operator panels, an additional keyboard interface (machine control panel without operating unit) can be used. For each keyboard interface, 64 pushbuttons, switches, etc. and 64 display elements can be connected via ribbon cable.

The signals sent from the MCP are copied to the PLC's DPR (Dual-Port RAM) by the integrated Ethernet CP-840D sl. The basic program of the PLC enters the incoming signals in the input image configured at FB1. The NC-related signals are generally distributed by the basic program to the NC/PLC interface. If required, the signals can be modified by the user.

The signals from the PLC to the MCP (displays) are transferred in the opposite direction.

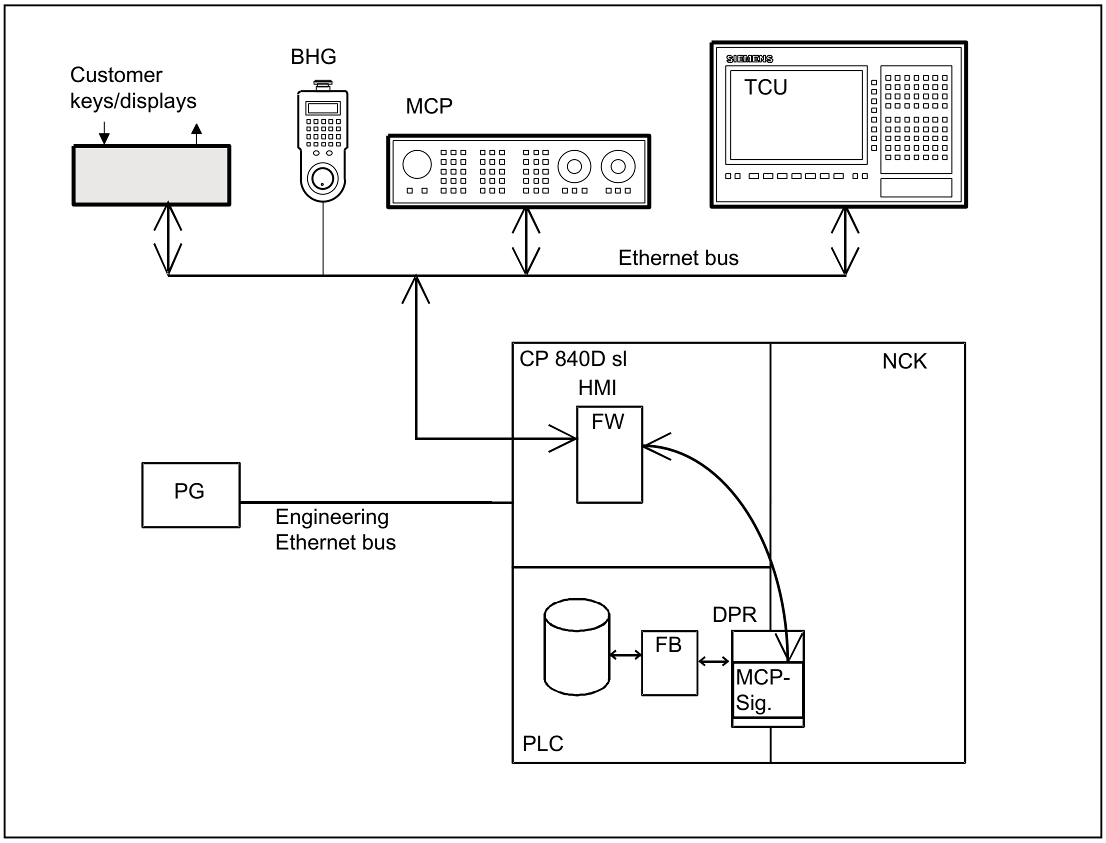


Figure 14-8 Connection of the machine control panel on 840D sl

Bus addresses

On Ethernet components, MAC and IP addresses or logic names are determining factors in respect of communication. The control system's system programs convert logic names into MAC or IP addresses. On the PLC, the numeric component of the logic name is used for communication. This numeric part is specified by the user to the FB1 using parameter "MCPxBusAdr".

The logical name of an MCP or HHU always begins with "DIP". This is followed by a number corresponding to the switch position of the MCP component (e.g. DIP192, DIP17).

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-, NC-, 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, FC19 or the other blocks mentioned above must be called in the **user program**.

Customized keys, which can be used to trigger a wide range of machine functions, must be evaluated directly by the user program. The user program also routes the status signals to the output area for the LEDs.

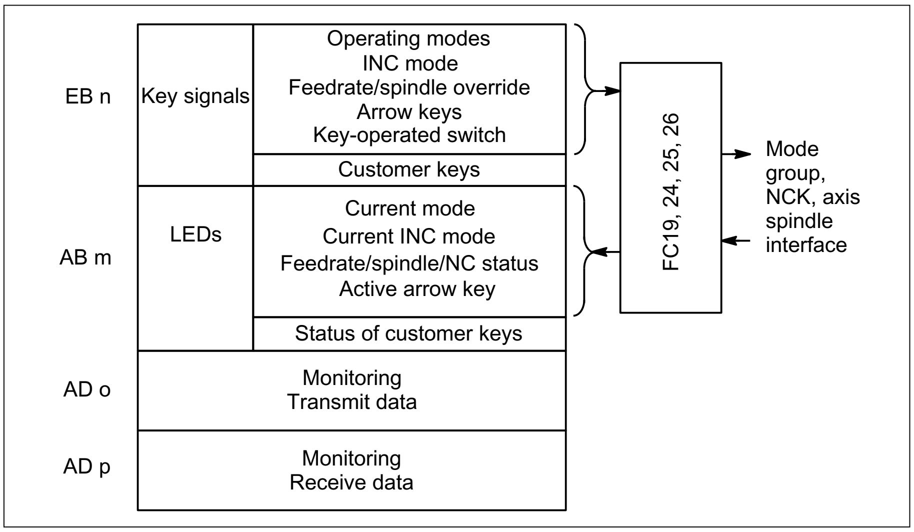


Figure 14-9 Interface to and from machine control panel

14.10 Structure and functions of the basic program

General

The PLC program has a modular structure. The organization blocks (OB) form the interface between the operating system and the basic and user programs.

- Restart (warm restart) with start-up and synchronization (OB100)
- Cyclic operation (OB1)
- Process alarms (OB40)
- Asynchronous errors: Diagnostics alarm (OB82), module failure (OB86)

The calls of the function blocks of the basic and user programs must be programmed by the user in the organization blocks (OB).

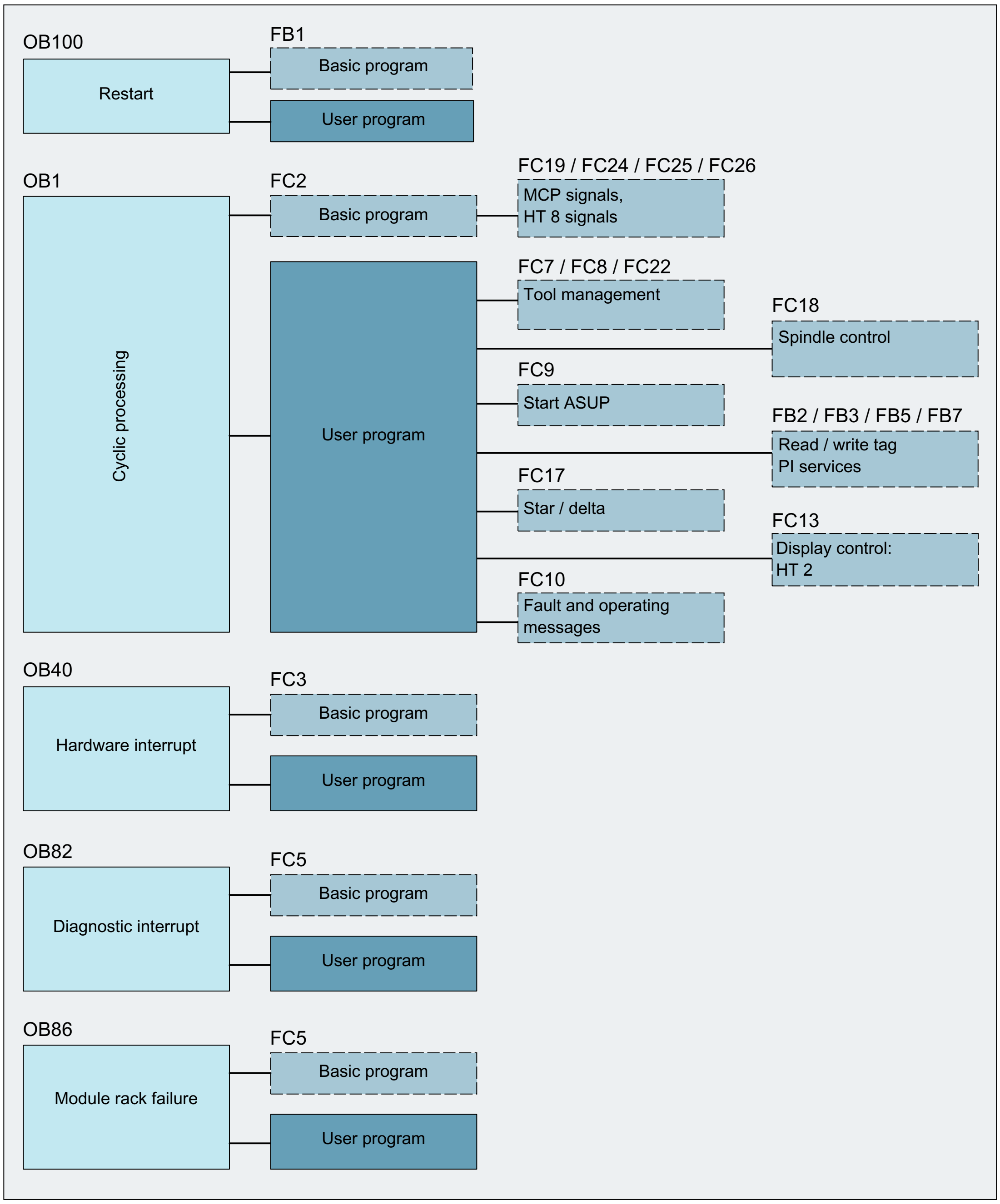


Figure 14-10 Structure of the basic program (principle)

14.10.1 Startup and synchronization of NCK PLC

Loading the basic program

The basic program must be loaded with the S7 tool when the PLC is in the Stop state. This ensures that all blocks in the basic program will be initiated correctly the next time they are called. Otherwise, undefined states can occur in the PLC (e.g. blinking of all PLC LEDs).

Startup,

The synchronization of NC 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 cases of errors, the basic program produces an alarm (visible on HMI) and switches the PLC to the Stop state.

A warm restart is not provided, i.e. following system initialization, the operating system runs organization block OB100 and always commences cyclic execution at the start of OB1.

Synchronization

The PLC is synchronized with the HMI, NC and CP during power-up.

Sign-of-life

After a correct initial start and the first complete OB1 cycle (basic setting cycle) the PLC and NC continuously exchange sign-of-life signals. If the sign of life from the NC fails to materialize, the PLC/NC interface is initialized and the signal "NC CPU ready" in DB10 is set to FALSE.

14.10.2 Cyclic operation (OB1)

General

The NC/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 transferred cyclically; transfer of the auxiliary commands and G commands takes place only on request.

The following functions are performed in the cyclic part of the basic program:

- Transmission of the control/status signals
- Distribution of the auxiliary functions
- M decoding (M00 - M99),
- M, S, F distribution
- Transfer the MCP signals via NC
- Acquisition and conditioning of the user errors and operational 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 modes)
- Channel-specific signals (such as program and feed modifications)
- Axis- and spindle-specific signals (such as feed disable)

Auxiliary and G commands

The auxiliary and G commands have the following characteristics:

- Transfer to the PLC is block-synchronous (referred to a part 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	
	1st value	2nd value	1st value	2nd value	1st value	2nd value
G command		G command		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 NC are output together with the accompanying change signals to the **CHANNEL DB** interface via the auxiliary/G command distributor (see List Manual). The two values of the auxiliary function are transferred to the appropriate data word. The accompanying change signal is activated to 1 for one PLC cycle. When the change signal is reset, the acknowledgment is passed to the NC. The acknowledgment of high-speed auxiliary functions is performed when the basic program detects the auxiliary function.

In addition to distribution of the auxiliary and G commands, selected signals are processed as described below.

M decoder

M functions can be used to transfer switching commands and fixed-point values. Decoded dynamic signals are output to the **CHANNEL DB** interface for standard M functions (range M00 - M99) (signal duration = 1 cycle time).

G group decoders

In the case of G commands sent by the NC, 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 commands are entered in the channel DBs. The entered G commands 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

Depending on the bus connection, the MCP signals are either transferred directly to the PLC or indirectly to the parameterized I/O areas via an internal procedure using the basic program.

User messages

The acquisition and processing of the user error messages and operational messages is performed by an FC in the basic program.

14.10.3 Time-interrupt processing (OB35)

The user must program **OB35** 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 OB35 time setting must be at least 3 ms in order to avoid a PLC CPU stop. The stop is caused by reading of the HMI system state list during powerup of the HMI. This reading process blocks priority class control for approx. 2 ms. The OB35 with a time base set to a rather lower value is then no longer processed correctly.

14.10.4 Process-interrupt processing (OB 40)

A process interrupt **OB40** (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 OB40. The cause of the interrupt is contained in the local data of OB40.

References:
SIMATIC STEP 7 Description or Online Help of STEP 7

14.10.5 Diagnostic alarm, module failure processing (OB82, OB86)

General

In the event of a diagnostic alarm or failure of an I/O module, basic program block OB82 or OB86 is called. The basic program block FC5 "Diagnostic alarm and module failure" (Page 1046) is called from these blocks.

Bus diagnostics

The status of the DP slave modules at PROFIBUS connections MPI/DP, DP1 - or the PROFINET connection PN - is signaled to the user program by the basic program using the "Slaves OK" group signal of the particular bus system:

- DB10.DBX92.0 == 1 (MPI/DP bus: Slaves OK)
- DB10.DBX92.1 == 1 (DP1 bus: Slaves OK)
- DB10.DBX92.2 == 1 (PN bus: Slaves OK)

The group signal is derived from the LED status of the respective bus system (system state list SZL 0x174).

Alarm output

If an error or a failure of a slave is detected at the particular bus system, the following alarm is displayed:

- Alarm 400551 (MPI/DP bus)
- Alarm 400552 (DP1 bus)
- Alarm 400553 (PN bus)

The alarm is automatically deleted again when the error is removed.

Suppression of the alarm output 400551, 400552, 400553

By setting one of the following signals, the alarm is suppressed for the particular bus system:

- DB10.DBX92.4 = 1 (suppress alarm 400551)
- DB10.DBX92.5 = 1 (suppress alarm 400552)
- DB10.DBX92.6 = 1 (suppress alarm 400553)

The alarm is immediately suppressed as soon as the signal is set. Before the fault occurs or while a fault is already active.

When the signal is reset, in the case of a fault, the corresponding alarm is displayed.

The signals are reset when the control system powers up.

Note

By setting the signal, the fault monitoring for the complete bus line is deactivated!

14.10.6 Response to NCK failure

General

During cyclic operation, the PLC continuously monitors NC availability by querying the sign-of-life. If the NC is no longer reacting, the NC/PLC interface is initialized, and the **NC-CPU ready** signal in the area of the **signals from NC (DB 10.DBX 104.7)** is reset. Furthermore, the signals sent from the NC to the PLC and vice versa are set to an initial state.

The PLC itself remains active so that it can continue to control machine functions. However, it remains the responsibility of the user program to set the machine to a safe state.

NC → PLC signals

The signals sent by the NC to the PLC are divided into the following groups:

- Status signals from the NC, channels, axes and spindles
- Modification signals of the auxiliary functions
- Values of the auxiliary functions
- Values of the G commands

Status signals:

The status signals from the NC, channels, axes, and spindles are reset.

Auxiliary function change signals:

Auxiliary function change 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 NC.

G command values:

G command values are reset (i.e. each initialized with the value 0).

PLC → NC signals

The signals sent by the PLC to the NC are divided into control signals and tasks that are transferred by FCs to the NC.

Control signals:

The control signals from the PLC to the NC are frozen; cyclic updating by the PLC basic program is suspended.

Jobs from PLC to NC:

The FCs and FBs, which are used to pass jobs to the NC, 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 NC data) must not be activated in the user program until the **NC-CPU ready** signal is set.

14.10.7 Functions of the basic program called from the user program

General

In addition to the modules of the basic program, which are called at the start of OB1, OB40 and OB100, 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 NC:

- Traversing concurrent axes (FC18)
- Start asynchronous subprograms (ASUPs) (FC9),
- Select NC programs (FB4)

- Control spindle (FC18),
- Read and write variables (FB2, FB3)

Note

Checking and diagnostics of a function call of the basic PLC program

To simplify the checking and diagnostics of a function call (FB or FC) of the basic PLC program that is controlled via a trigger (e.g. via Req, Start parameters) and that provide an execution acknowledgment as output parameter (e.g. via Done, NDR, Error parameters), proceed as follows.

A variable compiled of other signals which produce the trigger for the function call should be set. Start conditions may be reset only as a function of the states of parameters Done, NDR and Error.

The appropriate control mechanism can be placed in front of or behind the function call. If the mechanism is placed after the call, the output variables can be defined as local variables (advantage: Reduction of global variables, markers, data variables and time-related advantages over data variables).

The trigger parameter must be a global variable (e.g. marker, data variable).

Jobs that are still active must be reset from the user program in OB100 (Req, Start, parameters,
etc. from TRUE ⇒ FALSE). A POWER OFF/ON could result in a state in which jobs are still active.

Concurrent axes

The distinguishing features of concurrent axes are as follows:

- They must be defined as such via the NC machine data.
- They can be traversed either from the PLC or from the NC by means of the JOG keys.
- Starting from the PLC is possible in the NC operating modes MDI and AUTOMATIK via FC.
- The start is independent of NC block boundaries.

Function calls are available for positioning axes, indexing axes and spindles (FC18).

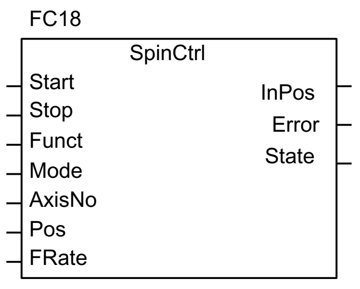
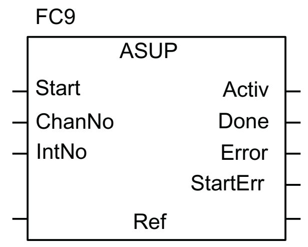


Figure 14-11 FC18 input/output parameters

Asynchronous subprograms (ASUPs)

The ASUP can be used to trigger any functions 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 FB4 PI services (ASUP).

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 FC9 is interrupted by the asynchronous subprogram. An ASUP is started by calling FC9 from the user program by setting the start parameter to 1.



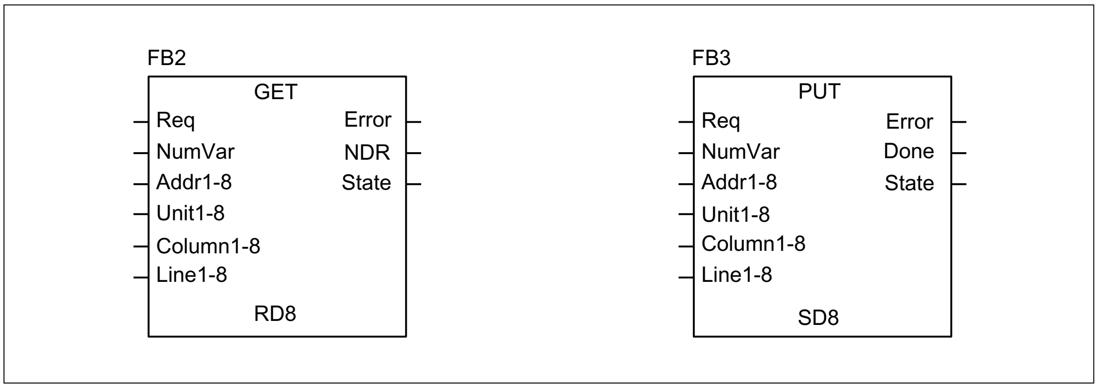
Note

If an asynchronous subprogram has not been prepared by an NC program or by FB4 (ASUP) (e.g. if no interrupt no. has been assigned), a start error is output (StartErr = TRUE).

Read/Write NC variables

NC variables can be read with FB GET while values can be entered in NC variables with FB PUT. The NC 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, a PC software (NC-Var-Selector) 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. NC floating-point values (64-bit) are converted to PLC format (32-bit with 24-bit mantissa and 8-bit exponent) and vice versa]. A loss of accuracy results from the conversion from 64-bit to 32-bit REAL. The maximum precision of 32-bit REAL numbers is approximately 10 to the power of 7.



AG_SEND/AG_RECV functions

The AG_SEND/AG_RECV functions correspond to the functions of the library "SIMATIC_NET_CP" of the S7-300 CPU in STEP 7. Generally, the online help is valid for these functions.
The AG_SEND/AG_RECV functions can be used for data exchange with another station via the integrated "CP 840D sl". A description of the functions is provided in Section "Block descriptions (Page 965)".

Note

Other communication blocks (e.g. BSEND, USEND) which possess a CP343-1 are not supported in SINUMERIK 840D sl.

14.10.8 Symbolic programming of user program with interface DB

General

Note

The basic program library on the CD supplied with the Toolbox for the 840D contains files NST_UDTB.AWL and TM_UDTB.AWL.

The compiled UDT blocks from these two files are stored in the CPU program of the basic program.

A UDT is a data type defined by the user that can, for example, be assigned to a data block generated in the CPU.

Symbolic names of virtually all the interface signals are defined in these UDT blocks.

The UDT numbers 2, 10, 11, 19, 21, 31, 71, 72, 73, 77, 1002, 1071, 1072, 1073 are used.

The assignments have been made as follows:

UDT assignments		
UDT number	**Assignment to interface DB**	**Meaning**
UDT2	DB2	Alarms/messages
UDT10	DB10	NCK signals
UDT11	DB11	Mode group signals
UDT19	DB19	HMI signals
UDT21	DB21 to DB30	Channel signal
UDT31	DB31 to 61	Axis/spindle signals
UDT71	DB71	Tool management: Load/unload locations
UDT72	DB72	Tool management: Change in spindle
UDT73	DB73	Tool management: Change in revolver
UDT77	DB77	MCP and HHU signals with standard SDB 210
UDT1002	DB2	Extended alarms / messages (FB1 parameter "ExtendAlMsg:=TRUE")
UDT 1071	DB 1071	Tool management: Loading/unloading points (multitool)
UDT 1072	DB 1072	Tool management: Change in spindle (multitool)
UDT 1073	DB 1073	Tool management: Change in turret (multitool)

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 UDT31 in the symbol file.

After this input, the STEP 7 program can be programmed in symbols for this interface.

Note

Programs generated with an earlier software version that utilize the interface DBs described above can also be converted into symbol programs. A fully qualified command for data access e.g. "U DB31.DBX60.0" (spindle / rotary axis) is necessary in the program previously created. This command is converted upon activation of the symbolics in the editor "AxisX.E_SpKA".

Description

Abbreviated symbolic names of the interface signals are defined in the two STL files NST_UDTB.AWL and TM_UDTB.AWL.

In order to create the reference to the names of the interface signals, the name is included in the comment after each signal.

The names are based on the English language. The comments are in English.

The symbolic names, commands and absolute addresses can be viewed by means of a STEP 7 editor command when the UDT block is opened.

Note

Unused bits and bytes are listed, for example, with the designation "f56_3".

- "56": Byte address of the relevant data block
- "3": Bit number in this byte

14.10.9 M decoding acc. to list

Function description

Up to 256 M functions with extended address can be decoded from the basic program using the "M decoding according to list" function. The function is activated using FB1 parameter "ListMDecGrp" (number of M groups for decoding). The assignment of the M function with extended address and a signal in the signal list is defined in the decoding list. The signals are also grouped for this purpose.

Decoding list (DB75)

The source file for the decoding list (MDECLIST.AWL) is supplied with the basic program. Data block DB75 is created when the STL source is compiled. Before the function is activated, the decoding list (DB75) must be transferred to the PLC followed by a restart.

An M function is decoded if it is in the decoding list. When decoding the M function, the corresponding signal is set in the signal list as a function of the specific group. When setting a signal in the signal list, the interface signal "Read in inhibit" is set by the basic program in the associated channel of the NC. The interface signal is reset again for the channel as soon as the user resets all of the signals output from this channel in the signal list; i.e. after they have been acknowledged.

Signal list (DB76)

When activating the function in data block DB76, the basic program creates the signal list. From then, for each M signal decoded according to the list, a signal is set in the signal list (DB76) in the corresponding group. At the same time, the "Readin inhibit" interface signal is set in the channel in which the M function has been output. The interface signal is reset again for the channel as soon as the user resets all of the signals output from this channel in the signal list; i.e. after they have been acknowledged.

Highspeed auxiliary functions

When an M function contained in the decoding list is output as "fast help function", no read-in inhibit is set for the corresponding channel of the NC.

The figure below shows the structure of the **M decoding according to list**:

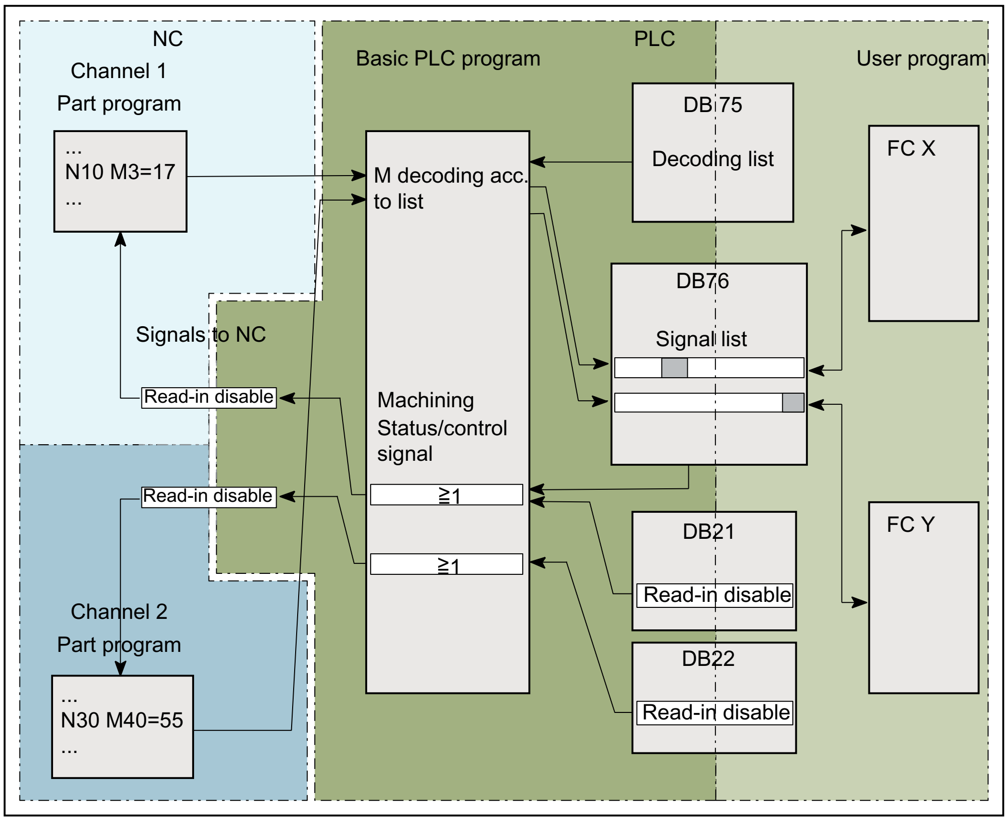


Figure 14-12 M decoding acc. to list

Activation

M decoding is activated using FB1 parameter "**ListMDecGrp**"

The number of M groups to the evaluated and/or decoded is specified using the appropriate parameter. The function is active for a parameter value = 1 ... 16.

- Basic program, OB100, FB1 parameter **ListMDecGrp** = <number of M groups> (also see " FB1: RUN_UP - basic program, start section (Page 965) ").

Properties and structure of the decoding list (DB75)

Properties of the decoding list (DB75):

- There is only one decoding list independent of the channel.
- The decoding list can include a maximum of 16 groups.
- A group has a maximum of 16 signals

- There must be an entry in the decoding list for every group of M functions to be decoded.
- The assignment between the M function with extended address and the signal to be set in the signal list is specified in the decoding list using the first and last M function of the associated group.
  - First M function: Parameter: "MFirstAdr" ≙ signal or bit 0
  - Last M function: Parameter: "MLastAdr" ≙ dependent on the difference to the first M function maximum signal or bit 15

Structure of the decoding list (DB75):

An entry in the decoding lists consists of 3 parameters, each of which is assigned to a group.

Group	Extended M address	First M address of the group	Last M address of the 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 of the signals:

Signal	Type	Value range	Meaning
MExtAdr	INT	0 ... 99	Extended M address
MFirstAdr	DINT	0 to 99.999.999	First M address in group
MLastAdr	DINT	0 to 99.999.999	Last M address in group

Properties of the signal list (DB76)

The signal list (DB76) has the following properties:

- There is only one signal list independent of the channel.
- The signal list can include a maximum of 16 signals for each M group.

Example

3 groups of M functions are to be decoded:

- Group 1: M2 = 1 to M2 = 5
- Group 2: M3 = 12 to M3 = 23
- Group 3: M40 = 55

Structure of the decoding and signal list

Group	Decoding list (DB75)			Signal list (DB76)
	Extended M address	First M address of the group	Last M address of the group	
1	2	1	5	DB76.DBX0.0 ... DBX0.4
2	3	12	23	DB76.DBX2.0 ... DBX3.3
3	40	55	55	DB76.DBX4.0

Program code

```
DATA_BLOCK DB 75
TITLE =
VERSION : 0.0
   STRUCT
      MSigGrp : ARRAY [1 .. 16 ] OF STRUCT
         MExtAdr : INT;
         MFirstAdr : DINT;
         MLastAdr : DINT;
      END_STRUCT ;
BEGIN
   MSigGrp[1].MExtAdr   :=    2; extended M address of the 1st group
   MSigGrp[1].MFirstAdr :=  L#1; first M address of the group
   MSigGrp[1].MLastAdr  :=  L#5; last M address of the group
   MSigGrp[2].MExtAdr   :=    3; extended M address of the 2nd group
   MSigGrp[2].MFirstAdr := L#12; first M address of the group
   MSigGrp[2].MLastAdr  := L#23; last M address of the group
   MSigGrp[3].MExtAdr   :=   40; extended M address of the 3rd group
   MSigGrp[3].MFirstAdr := L#55; first M address of the group
   MSigGrp[3].MLastAdr  := L#55; last M address of the group
END_DATA_BLOCK
```

Structure of FB1 in OB100

To activate the function, insert the parameter for the number of M groups to be decoded "ListMDecGrp".

```
Call FB 1, DB 7(
   ...
   ListMDecGrp := 3;     //M decoding of three groups
   ...
);
```

Description

A restart must be performed after the entry has been made in OB100 and the decoding list (DB75) has been transferred to the PLC. The basic program creates the signal list (DB76) when it restarts.

An NC program is then started, for instance in the 1st channel. An extended M function is included in this (M3=17). When decoding the M function (M3 ≙ group 2), the associated signal (DBW1.5) is set in the signal list (DB76) and the interface signal "Read-inhibit" in the 1st channel. The execution of the NC program is stopped. Further, the "Extended address M function" and the "M function number" are displayed in the channel DB of the 1st channel.

The "Read-in inhibit" signal in the 1st channel is reset once the user has reset all of the signals output from this channel in the signal list (DB76), and has therefore acknowledged them.

14.10.10 PLC machine data

General

The user has the option of storing PLC-specific machine data in the NC. These machine data can then be processed during power-up of the PLC (OB100). This enables, for example, user options, machine expansion levels, machine configurations, etc., to be implemented.

The interface to read this data is in DB20. However, DB20 is only created by the basic program during power-up when user machine data is used, i.e. sum of GP parameters "UDInt", "UDHex" and "UDReal" is greater than ZERO.

Size of the data areas

The sizes of the individual areas, and thus the total length of the DB20, is set by the following PLC machine data:

- MD14504 $MN_MAXNUM_USER_DATA_INT
- MD14506 $MN_MAXNUM_USER_DATA_HEX
- MD14508 $MN_MAXNUM_USER_DATA_FLOAT

User-relevant basic PLC program parameters

The machine data settings are provided to the user via the following basic PLC program parameters:

- "UDInt"
- "UDHex"
- "UDReal"

Data storage

The data is seamlessly stored in the DB20 by the basic PLC program in the following sequence:

1. INT values
2. HEX values (bit arrays)
3. FLOAT values

INT and FLOAT values are saved in S7 format.

The hexadecimal values are stored in DB20 in the order in which they are input (use as bit arrays).

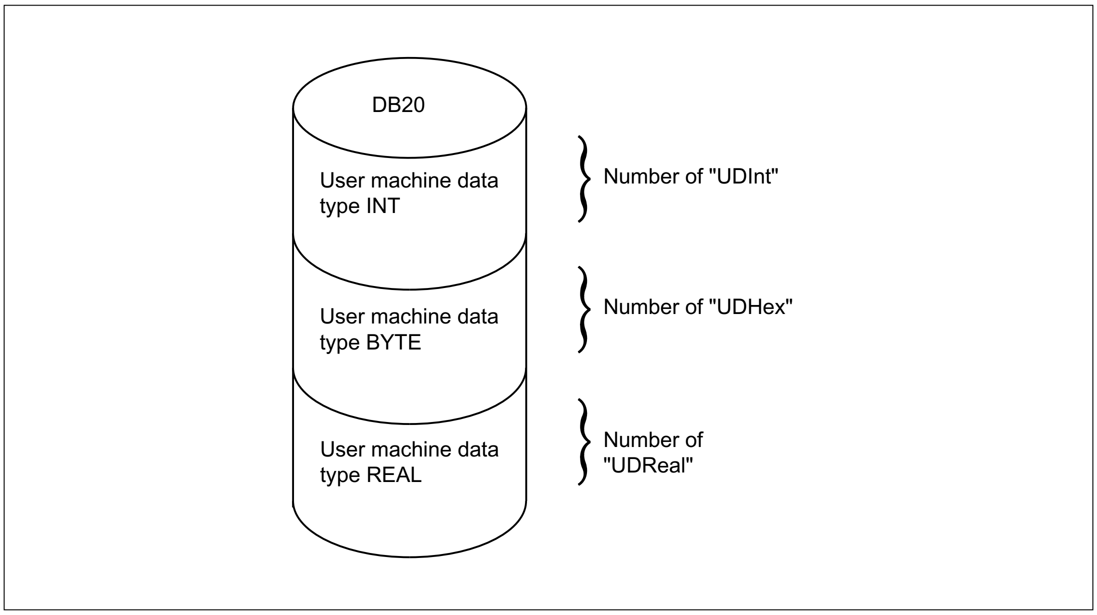

DB20
User machine data type INT
User machine data type BYTE
User machine data type REAL
Number of "UDInt"
Number of "UDHex"
Number of "UDReal"

Figure 14-13 User data in DB20

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.

Example

For the project in the example, four INT values, two HEX values for bit information, and one FLOAT value are needed.

Machine data:

MD14510 $MN_USER_DATA_INT[0] =	123
MD14510 $MN_USER_DATA_INT[1] =	456
MD14510 $MN_USER_DATA_INT[2] =	789
MD14510 $MN_USER_DATA_INT[3] =	1011
...	
MD14512 $MN_USER_DATA_HEX[0] =	12
MD14512 $MN_USER_DATA_HEX[1] =	AC
...	
MD14514 $MN_USER_DATA_FLOAT[0] =	123.456

Basic PLC program parameters (OB100):

```
CALL FB1, DB7(
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;
```

Basic PLC program parameters (scan at 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:

DB20	
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 UDT20
   STRUCT
      UDInt :   ARRAY [0 .. 3] OF INT;
      UDHex0 :  ARRAY [0 .. 15]OF BOOL;
      UDReal :  ARRAY [0 .. 0] OF REAL;    //Description as field, for
                                           later expansions
   END_STRUCT;
END_TYPE
```

Note

ARRAY OF BOOL are always sent to even-numbered addresses. For this reason, an array range of 0 to 15 must generally be selected in the UDT definition or all Boolean variables specified individually.

Although only a REAL value is used initially in the example, a field (with one element) has been created for the variable. This ensures that extensions can be made easily in the future without the symbolic address being modified.

Symbolic accesses

An entry is made in the symbol table to allow data access in symbolic form:

Symbol	Operand	Data type
UData	DB20	UDT20

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];
...
```

14.10.11 Configuration machine control panel, handheld unit, direct keys

General

Up to two machine control panels and one handheld unit can be in operation at the same time. There are various connection options (Ethernet/PROFINET, PROFIBUS) for the machine control panel (MCP) and handheld unit (HHU). It is possible to connect two MCPs to different bus systems (mixed operation is only possible on Ethernet and PROFIBUS). This can be achieved using FC1 parameter "MCPBusType". In this parameter, the right-hand decade (units position) is responsible for the first MCP and the left-hand decade (tens position) for the second MCP.

Parameterization of components is always performed by calling the basic program block FC1 in OB100. FC1 saves its parameters in the associated instance data block (DB7, symbolic

"LBP_ConfigData"). 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 FC19, FC24, FC25, FC26 and FC13. Further, the addresses for status information, PROFIBUS or Ethernet/PROFINET are also to be defined. The default time settings for timeout and cyclic forced retriggering should not be changed. Please refer to the Operator Components manual for further information on MCP and HHU components.

Activation

Each component is activated either via the number of machine control panels ("MCPNum" parameter) or, in the case of the handheld unit, via the "BHG" parameter. The MCP and HHU connection settings are entered in FC1 parameters "MCPMPI", "MCPBusType" or "BHG", "BHGMPI".

Handheld unit (HT 2)

In the handheld unit the addressing is done via a parameter of the GD parameter set. This was necessary for reasons of compatibility of the parameter names.

Configuration

Essentially, there are various communication mechanisms for transferring data between the MCP/HHU and PLC. These mechanisms are characterized by the bus connection of the MCP and HHU. In one case (Ethernet), data is transported via the "CP 840D sl".

The parameterization is performed completely via the MCP/HHU parameters in FC1.

In the other case the transmission is via the PLC operating system through the PROFIBUS configuration.

The parameterization is performed via STEP 7 in HW-Config. To enable the basic program to access this data and failure monitoring of MCP/HHU, the addresses set in the FC1 parameters must be made known to the basic program.

An overview of the various coupling mechanisms is shown below. Mixed operation can also be configured.

If an error is detected due to a timeout, an entry is made in the alarm buffer of the PLC CPU (alarms 400260 to 400262). In this case, the input signals from the MCP or from the handheld unit (MCP1In/MCP2In or BHGIn) are reset to 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message deleted by the GP.

Note

The abbreviation "(n.r.)" in the tables below means "not relevant".

Ethernet connection (MCPBusType = 5)

Without further configuration settings being made, communication takes place directly from the PLC GP via the CP 840D sl. The FC1 parameters listed below are used for parameterization.

The numeric part of the logical name of the component must be entered in "MCP1 BusAdr", "MCP2 BusAdr" or "BHGRecGDNo" (corresponds to the bus address of the node). The logical name is defined via switches on the MCP or terminal box.

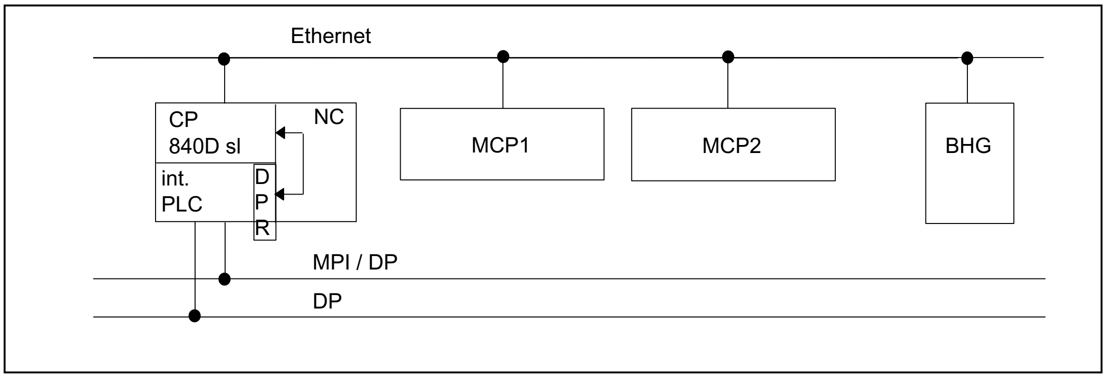


Figure 14-14 Ethernet connection

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 (n.r.)	MCP2StatRec (n.r.)	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen (n.r.)
MCP1Timeout (n.r.)	MCP2Timeout (n.r.)	BHGOutLen (n.r.)
MCP1Cycl (n.r.)	MCP2Cycl (n.r.)	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCP1NotSend	MCP2NotSend	BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPBusType = b#16#55 (via CP 840D sl)		BHGSendGDNo (n.r.)
		BHGSendGBZNo (n.r.)
MCPSDB210= FALSE		BHGSendObjNo (n.r.)
MCPCopyDB77 = FALSE		**BHGMPI = FALSE**
		BHGStop
		BHG NotSend

An error entry is also made in the PLC alarm buffer for timeouts. As a result, the following error messages are output at the HMI:

- 400260: MCP 1 failure
  or
- 400261: MCP 2 failure
- 400262: HHU failure

An MCP or HHU failure is detected immediately after a cold restart even if no data has yet been exchanged between the MCP/HHU and PLC.
The monitoring function is activated as soon as all components have signaled "Ready" after powerup.

Example: OP with direct keys

The direct keys of the OPs at the Ethernet bus should be transferred to the PLC. Previously, the direct keys have been transferred to the PLC via the PROFIBUS or via a special cable connection between OP and MCP.

For connecting the direct keys via the Ethernet, this concerns e.g. the "OP 08T", there is a parameterization in the basic program for activating the data transport. The associated parameters are in the instance DB of FC1 (OpKeyNum to OpKeyBusType, see data table). The parameters are provided by the user in the startup OB100 by connecting the parameters at the FC1 call. The bus address and Op1/2KeyStop can also be modified in the cyclic operation by writing the FC1 instance DB DB7.

The transport of the user data of the direct keys runs in the same way as in the case of Ethernet MCP. The data transport can also be stopped and restarted by writing the DB7-parameter "Op1/2KeyStop". During the Stop phase the address of the direct key module (TCU-index or the MCP-address) can also be changed.

After resetting the Stop signal, a connection to the new address is established.

The status of the respective direct-key interfaces can be read in the interface signal:

DB10.DBX104.3 (OP1Key ready)

or

DB10.DBX104.4 (OP2Key ready)

Address direct keys

For the parameter Op1/2KeyBusAdr, the TCU index is normally to be used. This affects the OPs, such as OP 08T, OP 12T, which for the direct keys **do not** have special cable connection to an Ethernet MCP.

If OPs with direct keys have a special cable connection and these are connected to an Ethernet-MCP, then for the parameter Op1/2KeyBusAdr the address of the MCP (DIP-switch setting of the MCP) is to be used. Only the data stream of the direct keys (2 bytes) is transferred via the direct key interface.

Alarm direct keys

An error entry is also made in the PLC alarm buffer for timeouts. As a result, the following error messages are output at the HMI:

- 400274: Direct key 1 failed
  or
- 400275: Direct key 2 failed

Control unit switching for direct keys

The user connects Op1/2KeyBusAdr with 0xFF and stop = TRUE in startup block OB100. The direct key address of the M-to-N interface is connected to parameter "Op1KeyBusAdr" via the M-to-N block FB9.

Relevant parameters (FC1)		
Direct keys		**e.g. direct keys OP 08T**
OpKeyNum = 1 or 2 (number of OPs with direct keys)		
Op1KeyIn	Op2KeyIn	
Op1KeyOut	Op2KeyOut	
OpKey1BusAdr	Op2KeyBusAdr	Address: TCU index:
Op1KeyStop	Op2KeyStop	
Op1KeyNotSend	Op2KeyNotSend	
OpKeyBusType = b#16#55 (via CP 840D sl)		

MCP identification

Via the identify interface in DB7 it is possible to query the type of the Ethernet component (MCP, HT 2, HT 8 or direct keys) with the relevant parameters at the input/output in cyclic operation:

- Relevant parameters at the input:
  "IdentMcpBusAdr", "IdentMcpProfilNo", "IdentMcpBusType", "IdentMcpStrobe"
- Relevant parameters at the output:
  "IdentMcpType", "IdentMcpLengthIn", "IdentMcpLengthOut"

Here the DIP device address or the TCU index at the parameter "IdentMcpBusAdr" is activated by the user program together with setting of the Strobe signal.

The input parameter "IdentMcpProfilNo" is normally to be set to the value 0. This parameter is to be set to the value 1 only in the identification of the direct keys. The parameter "IdentMcpBusType" currently has no significance for a user program and is to be left in its default value.

After resetting the Strobe signal by the basic program, valid output information becomes available to the user. The resetting of the Strobe signals by the basic program can last for several PLC cycles (up to two seconds).

The output parameters should show the user the size of the data areas for the addressed device. Furthermore, it can be defined here whether an HT 2 or an HT 8 or no device is connected to the terminal box. With this information, the MCP channel or the HHU channel can be activated. During cyclic operation, the parameters can be written symbolically by the user program and read via the symbol names of DB7 (LBP_ConfigData).

Relevant parameters (FB1)		
MCP device identification		**Input parameters, e.g. OP 08T**
Input	**Output**	**Values in direct keys**
IdentMcpBusAdr	IdentMcpType	IdentMcpBusAdr = TCU index
IdentMcpBusProfilNo	IdentMcpLengthIn	IdentMcpBusProfilNo = Value 1
IdentMcpBusType	IdentMcpLengthOut	IdentMcpBusType = Default value
IdentMcpStrobe		
IdentMcpBusProfilNo		**Value**
MCP, HHU, HT 8, HT 2		B#16#0
Direct keys such as e.g. OP 08T, OP 12T		B#16#1
IdentMcpType (Mcp-Type)		
no device connected		0
MCP 483C IE (Compact)		B#16#80
MCP 483C IE		B#16#81
MCP 310		B#16#82
MCP OEM		B#16#83
MCP DMG		B#16#84
HT 8		B#16#85
TCU_DT (direct keys)		B#16#86
MCP_MPP		B#16#87
HT 2		B#16#88
OP 08T (direct keys)		B#16#89

PROFIBUS connection on the DP port (MCPBusType = 3)

In case of PROFIBUS connection of the MCP, this component must be considered in the hardware configuration setting of STEP 7. The MCP is connected to the standard DP bus of the PLC (**not to MPI/DP**). The addresses must be stored in the input and output mapping area. These start addresses must also be stored in the pointer parameters of FC1. The FC1 parameters listed below are used for further parameterization.

There is no PROFIBUS variant of the HHU. For this reason, an Ethernet connection is shown for the HHU in this figure. The PROFIBUS slave address must be stored in the parameters "MCP1BusAdr" and "MCP2BusAdr". Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in "MCPxStatRec".

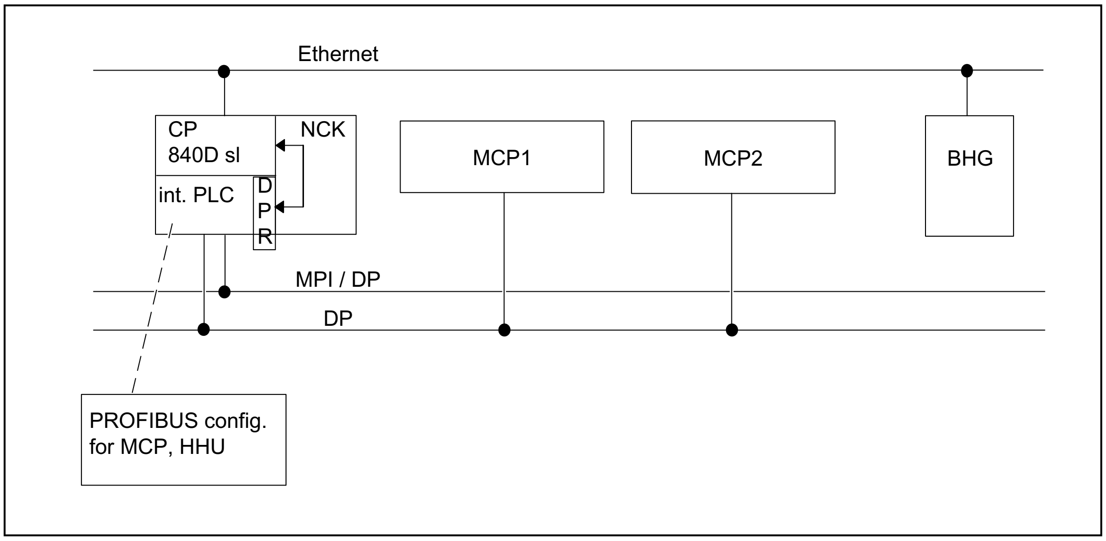


Figure 14-15 PROFIBUS connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		BHG = 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 (n.r.)	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType = b#16#33		BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPSDB210= FALSE		BHGSendGDNo (n.r.)
MCPCopyDB77 = FALSE		BHGSendGBZNo (n.r.)
		BHGSendObjNo (n.r.)
		BHGMPI = FALSE
		BHGStop

MCP failure normally switches the PLC to the STOP state. If this is undesirable, OB 82, OB 86 can be used to avoid a stop. The basic program has, as standard, the OB82 and OB86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. Setting "MCPxStop" := TRUE causes the basic program to deactivate the MCP as a slave via SFC12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The interrupt is deleted when the station recovers.

PROFIBUS connection on the MPI/DP port (MCPBusType = 4)

With the PROFIBUS connection of the MCP, this component must be considered in the STEP 7 hardware configuration. The MCP is connected on the MPI/DP bus of the PLC.

The addresses must be stored in the input and output mapping area. These start addresses must also be stored in the pointer parameters of FC1. The FC1 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 diagram. The PROFIBUS slave address must be stored in the parameters MCP1BusAdr and MCP2BusAdr. Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in MCPxStatRec.

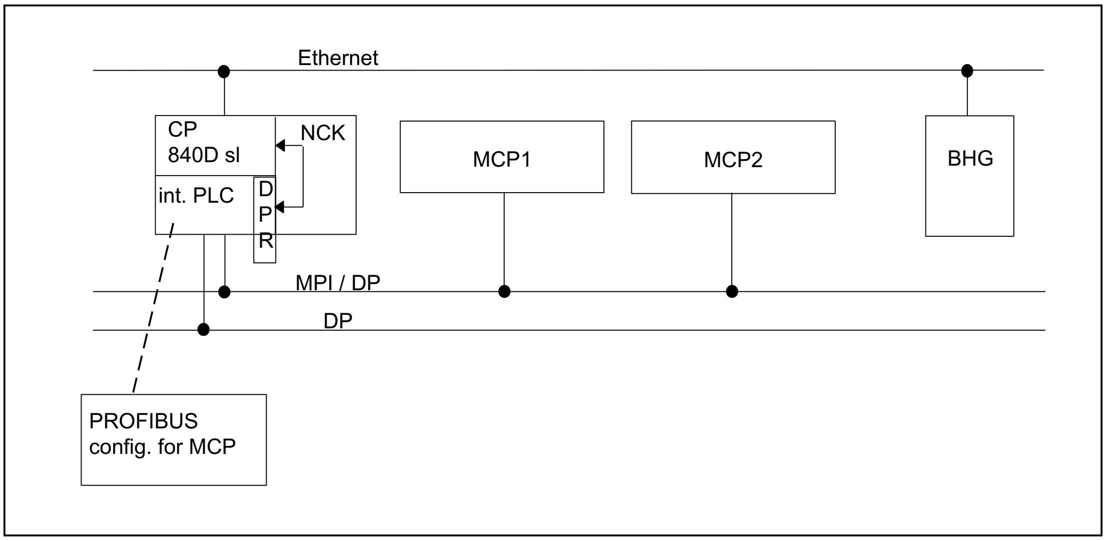


Figure 14-16 PROFIBUS connection on the MPI/DP port

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		HHU = 5 (via CP 840D sl)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend (n.r.)	MCP2StatSend (n.r.)	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl (n.r.)	MCP2Cycl (n.r.)	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType = b#16#44		BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPSDB210= FALSE		BHGSendGDNo (n.r.)
MCPCopyDB77 = FALSE		BHGSendGBZNo (n.r.)
		BHGSendObjNo (n.r.)
		BHGMPI = FALSE
		BHGStop

MCP failure normally switches the PLC to the STOP state. If this is undesirable, then OB82 and OB86 can be used to avoid a PLC stop. The basic program has, as standard, the OB82 and OB86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. Setting MCPxStop:= TRUE causes the basic program to deactivate the MCP as a slave via SFC12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The alarm is deleted when the station returns.

PROFINET connection (MCPBusType = 6)

In case of PROFINET connection of the MCP, this component must be parameterized in the hardware configuration setting of STEP 7. The MCP is coupled with the PROFINET module of the CPU.

When parameterizing the MCP in HW Config, the addresses should be placed in the input and output mapping area. These start addresses must also be stored in the pointer parameters (MCPxIn and MCPxOut) of FC1. This is because signals are transferred between the MCP and basic program via these parameters. The MCP is also monitored using parameter MCPxIn. This is the reason why parameter MCPxBusAdr is not relevant for this MCP variant.

Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in MCPxStatRec.

The PROFINET MCP has its own type which should be applied for parameter MCPBusType.

The FC1 parameters listed below are used for further parameterization. There is no PROFIBUS variant of the HHU. An Ethernet port for the HHU is shown in the diagram.

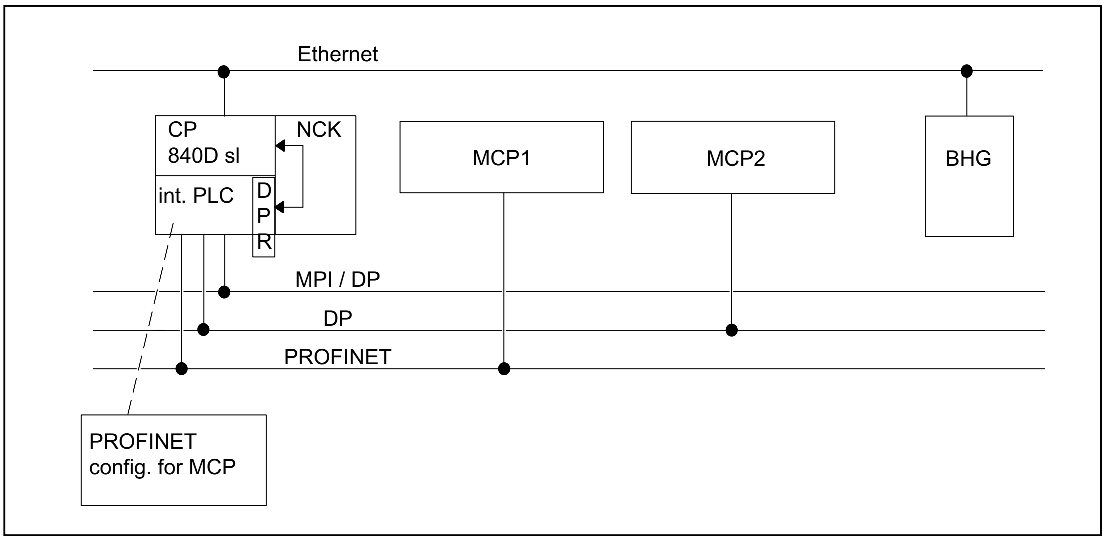


Figure 14-17 PROFINET connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		HHU = 5 (via CP 840D sl)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend (n.r.)	MCP2StatSend (n.r.)	BHGStatSend

Relevant parameters (FB1)		
MCP		**HHU**
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr (n.r.)	MCP2BusAdr (n.r.)	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl	MCP2Cycl	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType = b#16#36 (as in the figure as example) (6 = PROFINET for MCP1) (3 = PROFIBUS for MCP2)		BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPSDB210= FALSE		BHGSendGDNo (n.r.)
MCPCopyDB77 = FALSE		BHGSendGBZNo (n.r.)
		BHGSendObjNo (n.r.)
		BHGMPI = FALSE
		BHGStop

MCP failure normally switches the PLC to the STOP state. If this is undesirable, then OB82 and OB86 can be used to avoid a PLC stop. The basic program has, as standard, the OB82 and OB86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. The input address at parameter MCPxIn is of significance when monitoring for MCPxIn failure.

Setting MCPxStop:= TRUE causes the basic program to deactivate the MCP as a slave via SFC12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The alarm is deleted when the station returns.

14.10.12 Switchover of machine control panel, handheld unit

Only Ethernet variants support switchover/deactivation of an operator component (MCP or HHU) as standard.

PROFIBUS variant

With PROFIBUS variants, this functionality is only possible to a limited extent and with additional user effort.

With the PROFIBUS variant of the MCP, the data area of the DB77 for specified MCP1, MCP2 or HHU can be used for the MCP pointer on FC1. The MCP slave bus address must be set correctly at `MCPxBusAdr` as this is used as the basis for monitoring. A user program copy routine must 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.

Control signals

Parameters `MCP1Stop`, `MCP2Stop` and `HTStop` stop the 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 FC1 call.

Example: Stopping the transfer from the 1st machine control panel:

```
SET;
S LBP_ConfigData.MCP1Stop;
```

Setting parameters `MCP1Stop`, `MCP2Stop`, `HTStop` also results in a suppression or deletion of alarms 400260 to 400262.

Switchover of the bus address

If an existing communication connection to an operator component (MCP or HHU) is to be cancelled and a new communication connection established to a different component (MCP or HHU) with a different communication address, proceed as follows:

1. Stop the communication of the operator component to be disconnected: Parameter `MCP1Stop`, `MCP2Stop` or `HTStop` = 1
2. The communication is stopped when the following applies: DB10, DBX104.0, .1 or .2 == 0
3. Change the bus address:
   - MCP: FC1 parameter `MCP1BusAdr` or `MCP2BusAdr` = <bus address of the new operator component>
   - HHU (Ethernet variant): FC1 parameter `BHGRecGDNo` = <bus address of the new operator component>
4. Enable the communication (possible in the same PLC cycle as point 3): Parameter `MCP1Stop`, `MCP2Stop` or `HTStop` = 0
5. The communication with the new component is active when the following applies: DB10, DBX104.0, .1 or .2 == 1

Switching off the LED flashing of an Ethernet MCP

After a Power On, MCPs generally indicate the completion of the power-up and waiting for a connection to be established by flashing LEDs. The flashing of the LEDs can be switched off as described in the following. Presently, this behavior **cannot** be retentively stored on the MCP.

Requirement

MCP firmware as of V02.02.04

Setting for switching off the flashing

The Send status must be set in `MCPxStop` before the start of communication with the MCP. Before the start of communication means either during power-up (OB100) or during cyclic operation (OB1) before the setting of DB7 parameter `MCPxStop` = FALSE

Setting the Send status: FC1 parameter `MCPxStatSend`, bit 30 = 0 and bit 31 = 1

There is no feedback of the current status.

Example

Extract from OB100: (based on the example for MCP1)

```
CALL "RUN_UP" , "LBP_ConfigData"
   ...
   MCP1StatSend   := P#A 8.0
   ...
   // Deactivate MCP flashing
   SET
   R   A 11.6
   S   A 11.7
   ...
```

14.11 SPL for Safety Integrated

Rather than being a function of the basic program, SPL is a user function. The basic program makes a data block (DB18) available for Safety SPL signals and runs a data comparison to ensure the consistency of SPL program data in the NC.

References:
/FBSI/ Description of Functions Safety Integrated

14.12 Assignment overview

14.12.1 Assignment: NCK/PLC interface

The values of the NC/PLC interface for SINUMERK 840D sl are described in detail in:
References:
Lists sl (Buch2)

14.12.2 Assignment: FB/FC

Number	Meaning
FB15	Basic program
FB1, FC2, FC3, FC5	Basic program
FC0 ... 29	Reserved for Siemens
FB0 ... 29	Reserved for Siemens
FC30 ... 999 1)	Free for user assignment
FB30 ... 999 1)	Free for user assignment

Number	Meaning
FC1000 ... 1023	Reserved for Siemens
FB1000 ... 1023	Reserved for Siemens
FC1024 ... upper limit	Free for user assignment
FB1024 ... upper limit	Free for user assignment

[1] The actual upper limit of the block number (FB/FC) depends on the PLC CPU on which the selected NCU is located.

Note

Values of FC, FB see " Memory requirements of the basic PLC program (Page 949)".

14.12.3 Assignment: DB

Note

Only as many data blocks as are required according to the NC machine data configuration are set up.

Overview of the data blocks			
DB no.	**Designation**	**Name**	**Pack-age**
1		Reserved for Siemens	GP
2 ... 5	PLC-MELD	PLC messages	GP
6 ... 8		Basic program	
9	NC COMPILE	Interface for NC compile cycles	GP
10	NC INTERFACE	Central NC interface	GP
11	Mode group 1	Mode group interface	GP
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 for NC channels	GP
31 ... 61	AXIS 1 ... m	Interfaces for axes/spindles or free for user	GP
62 ... 70		Free for user	
71 ... 74		Tool management	GP

Overview of the data blocks			
DB no.	**Designation**	**Name**	**Pack-age**
75 ... 76		M group decoding	
77		DB 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	

1) The actual upper limit of the block number (DB) depends on the PLC CPU on which the selected NCU is located. Data blocks of channels, axes/spindles, and tool management functions that have not been activated are available.

Note

The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as required by the user.

14.12.4 Assignment: Timers

Timer No.	Significance
T 0 ... T 512 1)	User area

1) The actual upper limit of the timer number (DB) depends on the PLC CPU on which the selected NCU is located.

14.13 PLC functions for HMI (DB19)

14.13.1 Channel selection

Function

The channel displayed on the HMI, e.g. in the machine start screen, can be selected from the PLC user program via the HMI/PLC interface.

Requirement

More than one channel is parameterized in the NC.

Job and acknowledgment interface

DB19	Meaning	
DBX32.0 - .5	PLC → HMI	Function number: 1 = channel selection
DBX32.6	PLC → HMI	Function request
DBX32.7	HMI → PLC	Status: 1 = "function being executed"
DBB33	PLC → HMI	Channel number: 1, 2, 3, ... maximum number of channels Next channel: FF_H
DBB36	HMI → PLC	Error identification: • 0: No error • 1: Invalid function number (DBX32.0 - .5) • 2: Invalid parameter (DBB33 - DBB35) • 3: Error when writing the HMI-internal variable • 10: Channel not present (DBB33)

Functional sequence

PLC → HMI

The PLC user program must maintain the following execution sequence:

1. Check whether the interface is free for a new job:
   - `DB19.DBX32.6 == 0` (function request)
   - `DB19.DBX32.7 == 0` (status)
2. If the interface is free, the job data must be entered and the function request set:
   - `DB19.DBB33 = <channel number>`
   - `DB19.DBX32.0 - .5 = 1` (function number)
   - `DB19.DBX32.6 = 1` (function request)

HMI → PLC

The HMI makes the following responses for **error-free** parameterization:

1. Once the HMI has recognized the function request for channel selection, the status is set to "function being performed" and the function request reset:
   - `DB19.DBX32.7 = 1` (status)
   - `DB19.DBX32.6 = 0` (function request)
2. Once the channel selection has been performed, the status is reset again and value 0 is set as error identification:
   - `DB19.DBX32.7 = 0` (status)
   - `DB19.DBX36 = 0` (error identification)

The HMI makes the following responses for **faulty** parameterization:

- The function request is reset and the appropriate error identification is set:
  - `DB19.DBX32.6 = 0` (function request)
  - `DB19.DBX36 = <error identification>`

14.13.2 Program selection

Function

Preselected programs/workpieces can be selected for machining by the NC via the PLC/HMI interface.

The preselection is implemented by entering programs/workpiece in files (these are known as PLC program lists (*.ppl).

Requirements

The following machine data must be set to allow the HMI to process tasks:

MD9106 $MM_SERVE_EXTCALL_PROGRAMS

In order to activate a sector-specific PLC program list, you must set the appropriate machine data and at least the protection level password:

- Area **User**
  - MD51041 $MN_ENABLE_PROGLIST_USER = 1
  - Protective level password: 3 (users)
  - Program list: /user/sinumerik/hmi/plc/programlist/plc_proglist_user.ppl
- Area **Manufacturer (OEM)**
  - MD51043 $MN_ENABLE_PROGLIST_MANUFACT **=** 1
  - Protective level password: 1 (manufacturer)
  - Program list: /oem/sinumerik/hmi/plc/programlist/plc_proglist_manufacturer.ppl

Structure of a program list

A program list is a text file. Each line contains the following information:

<program number> <program path><program name> [CH=<channel number>]

- Program number
  The program numbers which may be used in a program list depend on the sector:
  - user: 1 - 100
  - Manufacturer (OEM): 201 - 255
- Program path
  The program path must be completely specified in absolute terms.
  For specifying the program path, see:
  References
  Programming Manual, Work Planning, Section "File and Program Administration" > "Program memory" > "Addressing the files of the program memory"
- Channel number
  Specifying the channel number "CH=<channel number>" is optional. It is only required if the NC has more than one channel.

The following excerpt as example shows the structure of the user program list:

Program list: plc_proglist_user.ppl

```
1 //DEV2:/MPFDir/PROG_01.MPF CH=1
2 //DEV2:/MPFDir/PROG_01.MPF CH=2
```

Generating entries in a program list

The entries in a program list (*.ppl) can be directly edited in the file or entered in screen forms in the user interface.

- Via the user interface for the **user** area
  Operating area "Program Manager" > "ETC key (">")" > "Prog. list"
- Via the user interface for the **Manufacturer** area
  Operating area "Commissioning" > "System data" > "ETC key (">")" > "Prog. list"

Program selection: Job interface

Note

The PLC may only request a new job if the last job has been acknowledged by the HMI:
DB19.DBB26 == 0

Program list

DB19.DBB16 = <number of the program list>

Number	Program list
129	/user/sinumerik/hmi/plc/programlist/**plc_proglist_user.ppl**
131	/oem/sinumerik/hmi/plc/programlist/**plc_proglist_manufacturer.ppl**

Program number

The program number refers to the programs contained in the selected program list.

DB19.DBB17 = <program number>

- user area: 1 - 100
- oem area: 201 - 255

Requesting program selection

DB19.DBX13.7 = 1

Program selection: Acknowledgment interface

Job acknowledgment

- DB19.DBX26.7 == 1 (selection identified)
- DB19.DBX26.3 == 1 (program is selected)
- DB19.DBX26.2 == 1 (error when selecting the program, see error ID DB19.DBB27)
- DB19.DBX26.1 == 1 ((job completed)

Error detection

DB19.DBB27 == <error ID>

Error detection	
Value	**Meaning**
0	No error
1	Invalid program list number (DB19.DBB16)
3	User specification Program list plc_proglist_main.ppl not found (only for DB19.DBB16 ≠ 129, 131)
4	Invalid program number (DB19.DBB17)
5	Job list in the selected workpiece could not be opened.
6	Error in job list. (Job list Interpreter returns error)
7	Job list interpreter returns empty job list

Program selection: Job processing

A job to select a program is executed as follows:

1. Checking the acknowledgment byte: DB19.DBB26 == 0
   If the acknowledgment byte is not 0, then the last job has still not been completed.
2. Specifying the program list: DB19.DBB16
3. Specifying the program number: DB19.DBB17
4. Setting the request to select a program: DB19.DBX13.7 = 1

5. Evaluating the acknowledgment and error interface: DB19.DBB26 and DBB27
   The order is still not completed on the HMI side as long as: DB19.DBX26.3 == 1 (active)
   The order has been completed on the HMI side if one of the two signals has been set:
   - DB19.DBX26.1 == 1 (OK)
   - DB19.DBX26.2 == 1 (error)
6. To complete the order, the program selection request must be reset: DB19.DBX13.7 = 0
7. The HMI signals that it is ready to accept a new order by resetting the acknowledgment byte: DB19.DBB26 == 0

14.13.3 Activating the key lock

The operator panel keyboard and a keyboard directly connected to the HMI can be locked using the following interface signal

- 1. HMI: DB19.DBX0.2 = <value>
- 2. HMI: DB19.DBX50.2 = <value>

Value	Meaning
0	Key lock inactive
1	Key lock active

14.13.4 Operating area numbers

The number of the active operating area is normally displayed in: DB19.DBB21

If the HMI monitor is active, the number of the active operating area is no longer displayed in DB19.DBB21, but instead in the user-specific configured area of the HMI monitor (Page 945).

Operating area	Number
Machine	201
Parameters	205
Programming	203
Program Manager	202
Diagnostics	204
Commissioning	206

14.13.5 Screen numbers

The current screen number is normally displayed in: DB19.DBW24

If the HMI monitor is active, the current screen number is no longer displayed in DB19.DBW24, but instead in the user-specific configured area of the HMI monitor (Page 945).

Screen number ranges

The following screen number ranges are available:

- JOG, manual machine (Page 937)
- Reference point approach (Page 942)
- MDA (Page 942)
- AUTOMATIC (Page 942)
- Parameters operating area (Page 943)
- Program operating area (Page 944)
- Program manager operating area (Page 945)
- Diagnostics operating area (Page 945)

14.13.5.1 Screen numbers: JOG, manual machine

JOG mode

Screen	Number
Turning technology	
Cycle start screen for all screens that can be taken over	81
Milling technology	
Cycle start screen for all screens that can be taken over	3
Turning/milling	
Start screen	19
T,S,M	2
Set WO	21
Positioning	4
Face milling	18
Stock removal	80
Cycle start screen for all user screens	91
General settings	1
Multi-channel function settings	106
Collision avoidance settings	107
Measurement log settings	108
Swiveling	60
All G commands	100
Actual zoom value (MCS/WCS)	101
Thread synchronizing	102
Retract	103
Handwheel	104
Action synchronization	105
Turning technology: Workpiece zero	
Workpiece zero (main menu)	30

Screen	Number
User screen	31
User screen	34
User screen	35
User screen	36
User screen	37
User screen	38
User screen	40
Measure edge Z	5
Turning technology: Workpiece, measurement	
Measure tool (main menu)	50
Manual X or user screen	51
Manual Y	71
Manual Z or user screen	52
Zoom or user screen	53
User screen	54
User screen	55
Probe calibration X or user screen	56
Probe calibration Z or user screen	57
Automatic length in Z	58
Automatic length in Y	73
Automatic length in X	59
Milling technology: Workpiece zero	
Workpiece zero (main menu)	30
Measure edge X	5
Measure edge X	22
Measure edge Z	23
User screen	7
Align edge or user screen	31
Distance 2 edges or user screen	32
Right-angled corner	33
Any corner or user screen	8
Rectangular pocket	34
1 hole or user screen	9
2 holes	35
3 holes	36
4 holes	37
Rectangular spigot	38
1 circular spigot or user screen	10
2 circular spigots	39
3 circular spigots	40
4 circular spigots	41
Set up level	42
Probe length calibration or user screen	11

Screen	Number
Probe radius calibration	12
Milling technology: Workpiece, measurement	
Measure tool (main menu)	50
Measure length, manual (with milling tool) or measure length in X, manual (with turning tool) or user screen	16
Measure length in Y, manual (with turning tool)	74
Measure length in Z, manual (with turning tool)	24
Measure diameter, manual or user screen	17
Measure length, automatic (with milling tool) or measure length in X, automatic (with turning tool) or user screen	13
Measure length in Y, automatic (with turning tool)	75
Measure length in Z, automatic (with turning tool)	25
Measure diameter, automatic or user screen	14
User screen	51
Probe calibration or user screen	15
Fixed point calibration or user screen	52
RunMyScreens (only for set JobShopIntegration)	
User screen for the 1st horizontal softkey	96
User screen for the 2nd horizontal softkey	98
User screen for the 3rd horizontal softkey	99
User screen for the 4th horizontal softkey	94
User screen for the 5th horizontal softkey	95
User screen for the 6th horizontal softkey	92
User screen for the 7th horizontal softkey	97
User screen for the 8th horizontal softkey	90
User screen for the 9th horizontal softkey	83
User screen for the 10th horizontal softkey	82
User screen for the 11th horizontal softkey	93
User screen for the 12th horizontal softkey	84
User screen for the 13th horizontal softkey	85
User screen for the 14th horizontal softkey	86
User screen for the 15th horizontal softkey	87
User screen for the 16th horizontal softkey	88

JOG mode, manual machine

DB19.DBB24	
Screen	**Screen number**
Turning/milling	
Taper turning	61
Angle milling	62

DB19.DBB24	
Screen	**Screen number**
Stop	63
Straight line	1300
Straight line all axes	1330
Straight line X alpha	1340
Straight line Z alpha	1350
Circle	1360
Drilling	1400
Center drilling	1410
Drilling, thread centered	1420
Drilling, centering	1433
Drilling, drilling	1434
Drilling, reaming	1435
Drilling, boring	1436
Drilling, deep hole drilling	1440
Drilling, deep hole drilling 2	1441
Drilling, tapping	1453
Drill thread milling	1455
Positions	1473
Position row	1474
Position grid	1477
Position frame	1478
Position circle	1475
Position pitch circle	1479
Obstacle	1476
Turning	1500
Turning, stock removal 1	1513
Turning, stock removal 2	1514
Turning, stock removal 3	1515
Turning, groove 1	1523
Turning, groove 2	1524
Turning, groove 3	1525
Turning, undercut form E	1533
Turning, undercut form F	1534
Turning, undercut thread DIN	1535
Turning, undercut thread	1536
Turning, thread, longitudinal	1543
Turning, thread, taper	1544
Turning, thread, facing	1545
Turning, thread, chain	1546
Turning, cut-off	1550
Milling	1600
Milling, face milling	1610

DB19.DBB24	
Screen	**Screen number**
Milling, rectangular pocket	1613
Milling, circular pocket	1614
Milling, rectangular spigot	1623
Milling, circular spigot	1624
Milling, longitudinal groove	1633
Milling, circumferential groove	1634
Milling, open groove	1635
Milling, multi-edge	1640
Milling, thread milling	1454
Milling, engraving	1670
Contour turning	1200
Contour turning, new contour / last contour	1210
Contour turning, stock removal along contour	1220
Contour turning, contour grooving	1230
Contour turning, contour plunge turning	1240
Contour milling	1100
Contour milling, new contour / last contour	1110
Contour milling, path milling	1120
Contour milling, centering	1130
Contour milling, rough drilling	1140
Contour milling, contour pocket	1150
Turning technology: Simulation	
Side view	1740
Front view	1750
3D view	1760
2-window view	1770
Half section	1780
Turning technology: Simultaneous recording	
Side view	1741
Front view	1751
3D view	1761
2-window view	1771
Machine space	1791
Half section	1781
Milling technology: Simulation	
Top view	1742
3D view	1760
From the front	1744
From the rear	1746
From the Left	1748
From the right	1752
Half section	1780

DB19.DBB24	
Screen	**Screen number**
Turning view	1782
Milling technology: Simultaneous recording	
Top view	1743
3D view	1761
From the front	1745
From the rear	1747
From the Left	1749
From the right	1753
Machine space	1791
Half section	1781
Turning view	1783

14.13.5.2 Screen numbers: Reference point approach

Screen	Number
Actual zoom value MCS/WCS	101

14.13.5.3 Screen numbers: MDA

Screen	Number
MDI	20
All G commands	100
Actual zoom value MCS/WCS	101
Handwheel	104
Action synchronization	105
Program control	210
Settings	250

14.13.5.4 Screen numbers: AUTOMATIC

Screen	Number
Automatic	200
Overstore	202
Program control	210
Block search	220
General settings	250
Multi-channel function settings	106
Collision avoidance settings	107
All G commands	100

Screen	Number
Actual zoom value MCS/WCS	101
Handwheel	104
Action synchronization	105
Turning technology: Simultaneous recording	
Side view	243
Front view	244
3D view	245
2-window view	246
Machine space	247
Half section	253
Milling technology: Simultaneous recording	
Top view	242
3D view	244
From the front	248
From the rear	249
From the Left	251
From the right	252
Machine space	247
Half section	253
Turning view	254

14.13.5.5 Screen numbers: Parameters operating area

Screen	Number
Tool list	600
Tool wear	610
User tool list	620
Magazine	630
Work offset	
Work offset, active	642
Work offset, overview	643
Work offset, basic	644
Work offset, G54 - G509	645
Details of work offset, active, overview, basic or G54 - G509	647
User variable	
R parameters	650
Global GUD 1 (SGUD)	660
Global GUD 2 (MGUD)	661
Global GUD 3 (UGUD)	662
Global GUD 4	663
Global GUD 5	664
Global GUD 6	665

Screen	Number
Global GUD 7	666
Global GUD 8	667
Global GUD 9	668
Channel GUD 1 (SGUD)	690
Channel GUD 2 (MGUD)	691
Channel GUD 3 (UGUD)	692
Channel GUD 4	693
Channel GUD 5	694
Channel GUD 6	695
Channel GUD 7	696
Channel GUD 8	697
Channel GUD 9	698
Local LUD	681
Local LUD/PUD	684
Setting data	
Working area limitation	671
Spindle data	670
Spindle chuck data	672
Ctrl-Energy	
Ctrl-Energy, main menu	6170
Ctrl-Energy, analysis	6171
Ctrl-Energy, profiles	6172
Ctrl-Energy, analysis graphic	6176
Ctrl-Energy, analysis long-term measurement	6177
Ctrl-Energy, analysis details	6179
Ctrl-Energy, compare measurements	6178

14.13.5.6 Screen numbers: Program operating area

Screen	Number
Turning technology: Simulation	
Side view	413
Front view	414
3D view	415
2-window view	416
Half section	423
Milling technology: Simulation	
Top view	412
3D view	414
From the front	418
From the rear	419
From the Left	421

Screen	Number
From the right	422
Half section	423
Turning view	424

14.13.5.7 Screen numbers: Program manager operating area

Screen	Number
NC directory	300
Local drive	325
USB / configured drive1	330
Configured drive2	340
Configured drive3	350
Configured drive4	360
Configured drive5	383
Configured drive6	384
Configured drive7	385
Configured drive8	386

14.13.5.8 Screen numbers: Diagnostics operating area

Screen	Number
Alarm list	500
Messages	501
Alarm log	502
NC/PLC variable	503

14.13.6 HMI monitor

Function

The HMI monitor is an 8-byte data area in a freely selectable data block, in which the HMI can provide the following data for the PLC user program:

- Operating area numbers (Page 936)
- Screen numbers (Page 936)

Parameterization

The data area is configured using the following display machine data:

MD9032 $MM_HMI_MONITOR = "string"

with "string" = "DB<DB number>.DBB<byte address>"

Note

Even byte address

The data area must start at an even byte address.

Structure of the data area

Byte	Meaning
EB n + 0	Active SINUMERIK operating area
EB n + 1	Reserved
EB n + 2	Current screen number
EB n + 3	
EB n + 4	Reserved
...	...
EB n + 7	Reserved

Supplementary conditions

When the HMI monitoring is active, the following PLC/HMI interface signals are no longer processed:

- DB19.DBB10 (PLC hardkeys)
- DB19.DBB21 (active SINUMERIK operating area)
- DB19.DBW24 (current screen number)

Example

Assumptions

- Current operating area: "Machine", number: 201
- Actual screen: "AUTOMATIC" start screen, number: 200
- PLC data area: DB60.DBB10

Parameterization

- MD9032 $MM_HMI_MONITOR = "DB60.DBB10"

Values in the data area

- DB60.DBB10: 201
- DB60.DBW12: 200

14.14 PLC functions for drive components on the integrated PROFIBUS

14.14.1 Overview

Using the function described below, input and output data from drive components on the integrated PROFIBUS can be consistently, cyclically read and written from the PLC user program of the **hardware PLC**. The following boundary conditions must be observed:

- For reading / writing input/output data, the system functions **SFC14 / SFC15** must be used.
- The reading / writing of input/output data is always over the **entire** slot length.
- The function only supports cyclical, **non-equidistant** data transfer.
- An output slot may not already be occupied on the NC side; (e.g. output slots of drives).

14.14.2 Performing a start-up

Preconditions

Before the function is put into operation on the NC side, the following preconditions must be satisfied:

- The drive components on the integrated PROFIBUS of the NCU must be fully configured using SIMATIC STEP 7, HW Config.
- The configuration is loaded into the PLC.

NC machine data

The start addresses of the slots to be transferred cyclically are to be entered into the following machine data:

- Input slots:
  MD10520 $MN_PLCINTERN_LOGIC_ADDRESS_IN[<Index>] = <Slot address>
- Output slots:
  MD10525 $MN_PLCINTERN_LOGIC_ADDRESS_OUT[<Index>] = <Slot address>

<Index>: 0, 1, 2, ... (Max no. of slots) – 1

<Slot address>: The slot address parameterized in HW Config

Note

Maximum quantity of data

The sum of all data to be cyclically transferred in inward and outward directions may not currently exceed **2,048** bytes.

14.14.3 Example

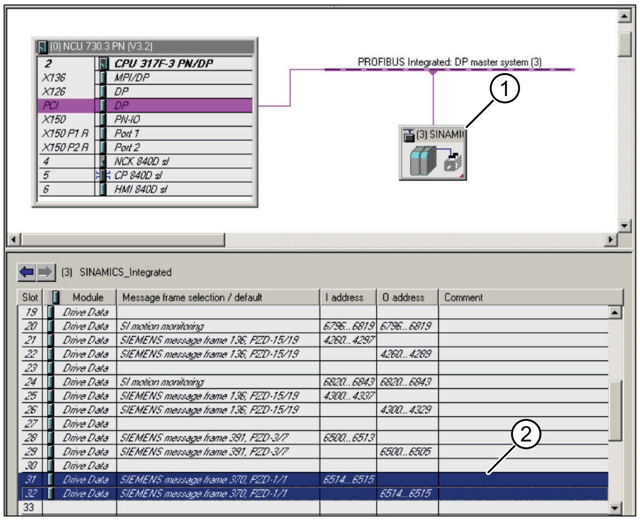
Determining slot addresses

After selecting the DP Slave "SINAMICS_Integrated" on the integrated PROFIBUS "PROFIBUS Integrated: DP master system (3)" in the station window of HW Config, its PROFIdrive message frame and associated slot addresses are displayed in the detailed view.

- Message frame 136: Drive
- Message frame 391: Control Unit
- Message frame **370**: Infeed

The infeed slot addresses required for parameterization in the NC machines are:

- Input slot: Slot 31, address **6514**
- Output slot: Slot 32, address **6514**



(3) SINAMICS_Integrated

Slot	Module	Message frame selection / default	I address	O address	Comment
19	Drive Data				
20	Drive Data	SI motion monitoring	6796...6819	6796...6819	
21	Drive Data	SIEMENS message frame 136, PZD-15/19	4260...4297		
22	Drive Data	SIEMENS message frame 136, PZD-15/19		4260...4289	
23	Drive Data				
24	Drive Data	SI motion monitoring	6820...6843	6820...6843	
25	Drive Data	SIEMENS message frame 136, PZD-15/19	4300...4337		
26	Drive Data	SIEMENS message frame 136, PZD-15/19		4300...4329	
27	Drive Data				
28	Drive Data	SIEMENS message frame 391, PZD-3/7	6500...6513		
29	Drive Data	SIEMENS message frame 391, PZD-3/7		6500...6505	
30	Drive Data				
31	Drive Data	SIEMENS message frame 370, PZD-1/1	6514...6515		
32	Drive Data	SIEMENS message frame 370, PZD-1/1		6514...6515	
33					

Figure 14-18 Infeed PROFIBUS message frames

Setting NC machine data

- Input slots:
  MD10520 $MN_PLCINTERN_LOGIC_ADDRESS_IN[0] = 6514
- Output slots:
  MD10525 $MN_PLCINTERN_LOGIC_ADDRESS_OUT[0] = 6514

Controlling the ALM using FB390 "ALM_Control"

The SINUMERIK hardware PLC is connected to the CU320 of SINAMICS S120 through the internal PROFIBUS. The ALM is connected to the CU via DRIVE-CLiQ.

In SINAMICS S120, a control and status message frame can be defined for each module (the CU, ALM, motor modules, etc.). If this is performed for an ALM, this can be switched on and off from the PLC user program.

The SIMATIC S7 block FB390 "ALM_Control" checks the status of the ALM and enables the user to switch it on or off.

A description of the block and an example project are available for download under the following link to Industry Online Support:

http://support.automation.siemens.com/WW/view/de/49515414

14.15 Memory requirements of the basic PLC program

The basic program consists of basic and optional functions. The **basic functions** include cyclic signal exchange between the NC and PLC. The **Options** include e.g. the FCs, which can be used, if needed.

The table below lists the memory requirements for the basic functions and the options. The data quoted represent guide values, the actual values depend on the current software version.

Memory requirements of blocks for SINUMERIK 840D sl			
Block Type, No.	**Function**	**Remark**	**Block size (bytes)**
	Working memory		
Basic functions in basic program			
FB1, FB15		Must be loaded / on CompactFlash Card	52182
FC2, 3, 5, 12		Must be loaded	470
DB4, 5, 7, 8		Must be loaded	1006
DB2, 3, 17		Are generated by the BP	632
OB1, 40, 100, 82, 86		Must be loaded	398
		Total	55698
PLC/NC, PLC/HMI interface			
DB10	PLC/NC signals	Must be loaded	262
DB11	PLC / mode group signals	Is generated by BP	56
DB19	PLC/HMI signals	Is generated by BP	434
DB21 to 30	PLC/channel signals	Are generated by BP as a function of NCMD: for each DB	416
DB31 to 61	PLC / axis or spindle signals	Are generated by BP as a function of NCMD: for each DB	148
Basic program options			
	Machine control panel		
FC19	Transfer of MCP signals, M variant	Must be loaded when M variant of MCP is instal-led	92

Basic program options			
FC25	Transfer of MCP signals, T variant	Must be loaded when T variant of MCP is installed	92
FC24	Transfer of MCP signals, slim variant	Must be loaded when slim variant of MCP is installed	100
FC26	Transfer of MCP signals, HT8 variant	Must be loaded for HT8	68
	Handheld unit		
FC13	Display control HHU	Can be loaded for handheld units	144
	Error/operating messages		
FC10	Acquisition FM/BM	Load when FM / BM is used	66
	ASUP		
FC9	ASUP start	Load when PLC ASUPs are used	128

Basic program options			
	Star/delta changeover		
FC17	Star/delta switchover of MSD	Load for star/delta switchover	114
	Spindle control		
FC18	Spindle control	Load for spindle control from PLC	132
	PLC/NC communication		
FB2	Read NC variable	Load for Read NC variable	76
DBn	Read NC variable	One instance DB per FB2 call:	270
FB3	Write NC variable	Load for Write NC variable	76
DBm	Write NC variable	One instance DB per FB3 call:	270
FB4	PI services	Load for PI services	76
DBo	PI services	One instance DB per FB4 call:	130
DB16	PI services description	Load for PI services	618
FB5	Read GUD variables	Load for PI services	76
DBp	Read GUD variables	One instance DB per FB5 call:	166
DB15	General communication	Global data block for communication	146
FB7	PI services 2	Load for PI services	76
DBo	PI services 2	One instance DB per FB4 call: every	144
FC21	Transfer	Load with dual-port RAM, ...	164
	M to N		
FB9	Switchover M to N	Load with M to N	58
	Safety Integrated		
FB10	Safety relay	Load with Safety option	74
FB11	Brake test	Load with Safety option	76
DB18	Safety data	DB for Safety	308
	Tool management		

Basic program options			
FC7	Transfer function turret	Load for tool management option	84
FC8	Transfer function	Load for tool management option	132
FC22	Direction selection	Load, when direction selection is needed	138
DB71	Loading locations	Generated by BP as a function of NC MD	40+30*B
DB72	Spindles	Generated by BP as a function of NC MD	40+48*Sp
DB73	Revolver	Generated by BP as a function of NC MD	40+44*R
DB74	Basic function	Generated by BP as a function of NC MD	100+(B+ Sp+R)*22
Compile cycles			
DB9	Interface PLC compile cycles	Is generated by BP as a function of NC option	436

Example:

Based on the memory requirements in the table above, the memory requirements have been determined for two sample configurations (see table below).

Block Type, No.	Function	Remark	Block size (bytes)
	Working memory		

Minimum configuration (1 spindle, 2 axes and T MCP)			
see above	Basic program, base		54688
	Interface DBs		1612
	MCP		92
		Total	56392

Block Type, No.	Function	Remark	Block size (bytes)
	Working memory		

Maximum configuration (2 channels, 4 spindles, 4 axes, T MCP)			
see above	Basic program, base		54688
see above	Interface DBs		2768
see above	MCP		92
see above	Error/operating messages		66
see above	ASUPs	1 ASUP initiation	128
see above	Concurrent axis	For 2 turrets	132
see above	PLC/NC communication	1 x read variable and 1 x write variable	838
see above	Tool management	2 turrets with one loading point each	674

Maximum configuration (2 channels, 4 spindles, 4 axes, T MCP)		
see above	Compile cycles	436
	Total	59822

14.16 Basic conditions and NC VAR selector

14.16.1 Supplementary conditions

14.16.1.1 Programming and parameterizing tools

Hardware

For the PLCs used in SINUMERIK 840D sl, the following equipment is required for the programming devices or PCs:

	Minimum	**Recommendation**
Processor	Pentium	Pentium
RAM (MB)	256	512 or more
Hard disk, free capacity (MB)	500	> 500
Interfaces	MPI, Ethernet incl. cable Memory card	
Graphic	SVGA (1024*768)	
Mouse	Yes	
Operating system	Windows 2000 /XP Professional, STEP 7 version 5.3 SP2 or higher	

The required version of **STEP 7** can be installed on equipment meeting the above requirements in cases where the package has not already been supplied with the programming device.

The following functions are possible with this package:

- Programming
  - Editors and compilers for STL (complete scope of the language incl. SFB/SFC calls), LAD, FBD
  - Creation and editing of assignment lists (symbol editor)
  - Data block editor
  - Input and output of blocks ON/OFF line
  - Insertion of modifications and additions ON and OFF line
  - Transfer of blocks from programming device to the PLC and vice versa
- Parameterizing
  - Parameterizing tool **HW Config** for CPU and I/O device parameterization
  - **NetPro** parameterizing tool for setting the CPU communication parameters
  - Output of system data such as hardware and software version, memory capacity, I/O expansion/assignment
- Testing and diagnostics (ONLINE)
  - Variable status/forcing (I/Os, flags, data block contents, etc.)
  - Status of individual blocks
  - Display of system states (ISTACK, BSTACK, system status list)
  - Display of system messages
  - Trigger PLC stop / restart / general reset from the PG
  - Compress PLC
- Documentation
  - Printout of individual or all blocks
  - Allocation of symbolic names (also for variables in data blocks)
  - Input and output of comments within each block
  - Printout of test and diagnostics displays
  - Hardcopy function
  - Cross-reference list
  - Program overview
  - Assignment plan I/O/M/T/Z/D

- Archiving of utility routines
  - Allocation of the output states of individual blocks
  - Comparison of blocks
  - Rewiring
  - STEP 5 → STEP 7 converter
- Option packages
  - Programming in S7-HIGRAPH, S7-GRAPH, SCL.
    These packages can be ordered from the SIMATIC sales department.
  - Additional packages for configuration modules (e.g. CP3425 → NCM package)

Note

More information about possible functions can be found in SIMATIC catalogs and STEP 7 documentation.

14.16.1.2 SIMATIC documentation required

ReferenceS:

- System description SIMATIC S7
- S7-300 instruction list
- Programming with STEP 7
- User Manual STEP 7
- Programming manual STEP 7; designing of user programs
- Reference manual STEP 7; Instructions list AWL
- Reference manual STEP 7; Ladder Diagram KOP
- Reference manual STEP 7; Default and system functions
- Manual STEP 7; Conversion of STEP 5 programs
- STEP 7 overall index
- Manual CPU 317-2DP

14.16.1.3 Relevant SINUMERIK documents

References:

- Commissioning Manual IBN CNC: NC, PLC, Drive
- Operator Components and Networking Manual
- Function Manual Basic Functions
- Function Manual, Extended Functions:
- Function Manual, Special Functions

- Lists sl (Book1)
- Lists sl (Book2)

14.16.2 NC VAR selector

14.16.2.1 Overview

General

The PC application "NC VAR selector" retrieves the addresses of required NC variables and processes them for access in the PLC program (FB2/FB3). This enables the programmer to select NC variables from the entire range of NC variables, to store this selection of variables, to edit them by means of a code generator for the STEP 7 compiler and finally to store them as an ASCII file (*.AWL) in the machine CPU program. This process is shown in the figure "NC VAR selector".

For storing the files created by NC-VAR-selector a catalog is to be implemented via the Windows Explorer with any catalog name. The selected data of the NC-VAR selector (data.VAR and data.AWL files) must be stored in this catalog. Thereafter, the STL file is to be transferred and compiled via the menu option "Code" → "in STEP 7 Project". The "data.AWL" (STL data) file must then be inserted into the STEP 7 machine project via "Insert", "External Source" in the STEP 7 Manager. The source container must be selected in the manager for this purpose. This action stores this file in the project structure. Once the file has been transferred, these AWL (STL) files must be compiled with STEP 7.

Note

The latest NC VAR selector can be used for each NC software version (even earlier versions). For older NC software versions the variables can also be selected from the latest complete list. The data content in DB120 (default DB for variables) does not depend on the software status. This means, variables selected in an older software version need not be reselected when the software is upgraded.

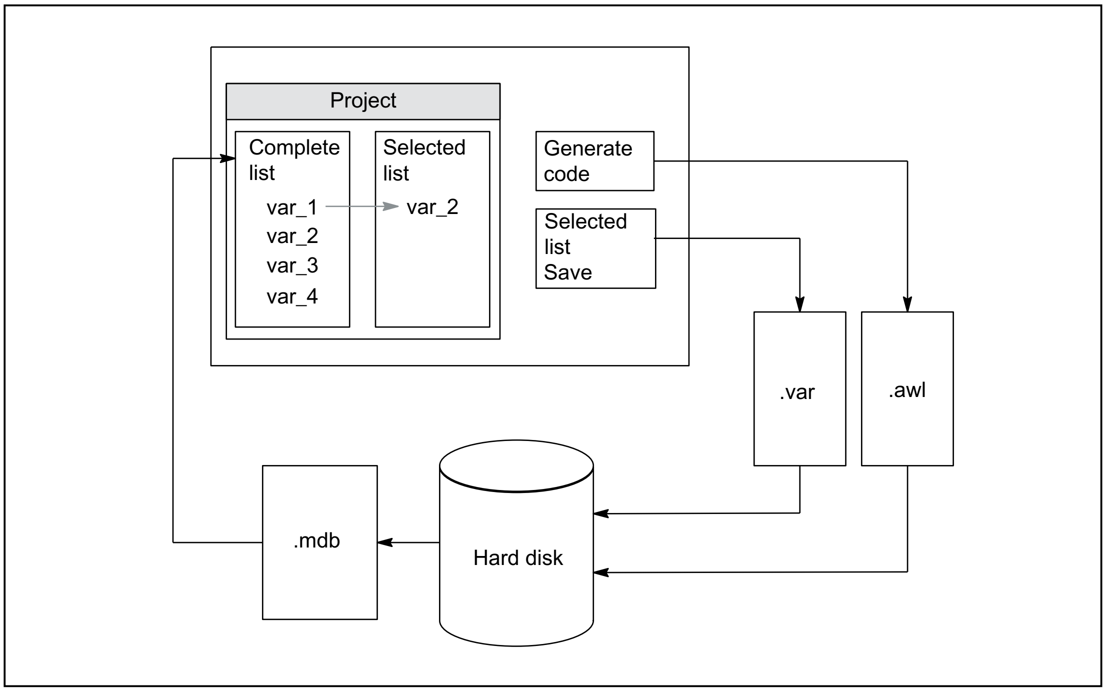


Figure 14-19 NC VAR selector

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.

The following ncv*.mdb variable list is available:

Variables	List
NC variables including machine and setting data:	ncv_NcData.mdb
Parameters of the drive:	ncv_SinamicsServo.mdb

The user can also transfer variables to a second list (separate window). This latter selection of variables can then be stored in an ASCII file or edited as a STEP 7 source file (.awl) and stored.

After generating a PLC data block by means of the STEP 7 compiler, the programmer is able to read or write NC 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 FB5 in the basic program.

Note

The latest version of the "NC VAR selector" is capable of processing all previous NC software versions. It is, therefore, not necessary to install different versions of the "NC VAR selector" in parallel.

System features, supplementary conditions

The PC application "NC VAR selector" requires Windows 2000 or a higher operating system.

The assignment of names to variables is described in:
References:
/List sl (Book1); Section: Variables,
or in the variables help file (integrated in NC VAR selector).

14.16.2.2 Description of functions

Overview

The figure below illustrates how the NC VAR selector is used within the STEP 7 environment.

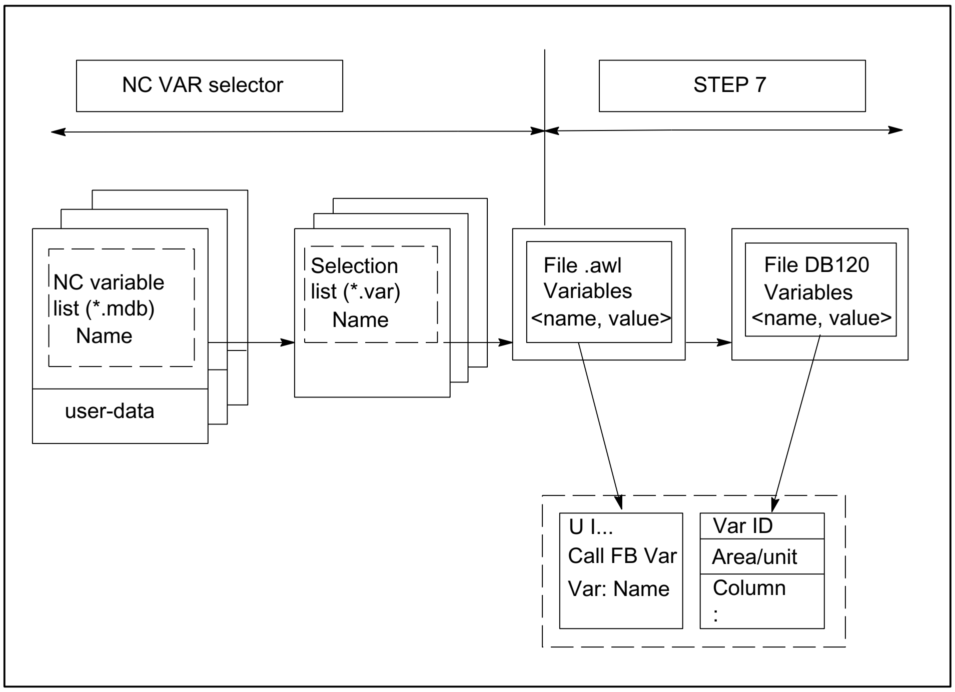

NC VAR selector
STEP 7
NC variable list (*.mdb) Name
user-data
Selection list (*.var) Name
File .awl Variables <name, value>
File DB120 Variables <name, value>
U I...
Call FB Var
Var: Name
Var ID
Area/unit
Column
:

Figure 14-20 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.

- A *.awl file contains the names and alias names of the NC variables, as well as information about their address parameters. Any data block generated from this file will only contain the address parameters (10 bytes per parameter).
- The generated data blocks must always be stored in the machine-specific file storage according to STEP 7 specifications.
- To ensure that the parameterization of the GET/PUT (FB2/3) blocks with respect to NC addresses can be implemented with symbols, the freely assignable, symbolic name of the generated data block must be included in the STEP 7 symbol table.

Basic display / basic menu

After the NC VAR selector has been selected (started), the basic display with all input options (upper menu bar) appears on the screen. All other displayed windows are placed within the general window.

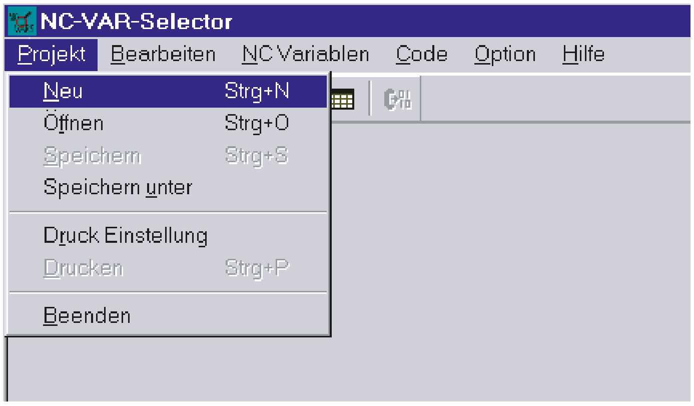


Figure 14-21 Basic display with basic menu

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).

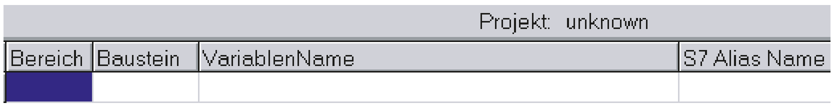


Figure 14-22 Window with selected variables for new project

The selected variables are displayed in a window.

Open an already existing project

Select "Open" under the "Project" menu item to open an existing project (variables already selected). A file selection window is displayed allowing the appropriate project with extension ".var" to be selected.

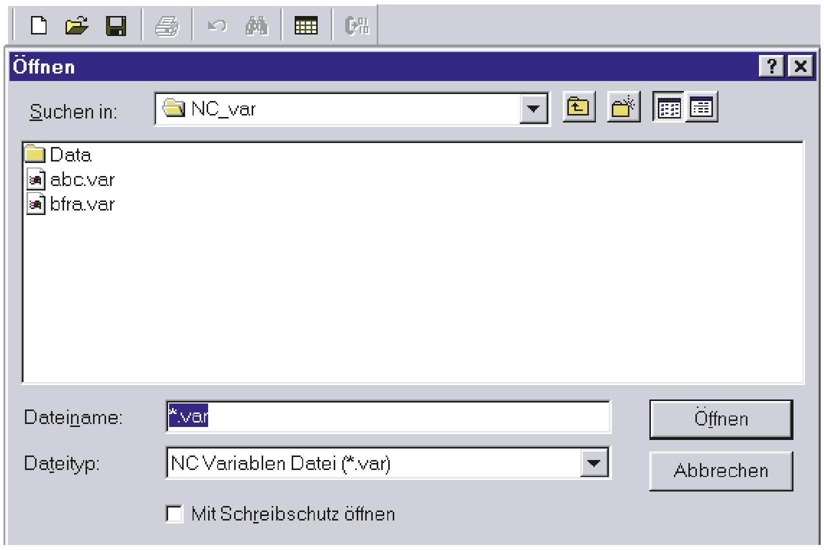


Figure 14-23 Selection window for existing projects

If, after selection of the project, new variables are to be added, a complete list of NC variables must be selected. No complete list need be called if the user only wishes to delete variables from the project.

Storing a project

The variable list is stored using the "Project" > "Save" or "Save As...." menu items.

"Save" stores the variable list under a path, which is already specified. If the project path is not known, then the procedure is as for "Save As....".

"Save As..." displays a window in which the path for the project to be stored can be specified.

Printing a project

The "Print" command under the "Project" menu item can be selected to print a project file. The number of lines per page is selected under the "Print Setting" menu item. The default setting is 77 lines.

Edit menu item

The following operator actions are examples of those, which can be carried out directly with this menu item:

- Transfer variables
- Delete variables

- change alias names
- Find variables

These actions can also be canceled again under Edit.

Undoing actions

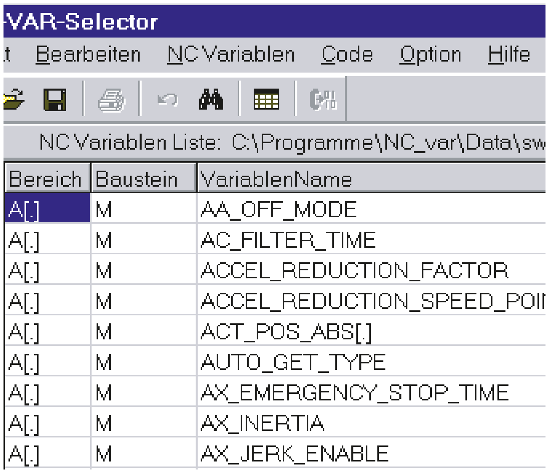
Operator actions relating to the creation of the project file (transfer variables, delete variables, change alias names) can be undone in this menu.

NC variables menu item

The basic list of all variables is saved in NC Var Selector path Data\Swxy (xy stands for software version no., e.g. SW 5.3:=xy=53). This list can be selected as an NC variables list. In case of SINUMERIK 840D sl the basic lists are present in the path Data\Swxy_sl.

Selecting an NC variable list

A list of all the NC variables for an NC version can now be selected and displayed via the "NC Variable List", "Select" menu item.



NC Variablen Liste: C:\Programme\NC_var\Data\sw

Bereich	Baustein	VariablenName
A[.]	M	AA_OFF_MODE
A[.]	M	AC_FILTER_TIME
A[.]	M	ACCEL_REDUCTION_FACTOR
A[.]	M	ACCEL_REDUCTION_SPEED_POI
A[.]	M	ACT_POS_ABS[.]
A[.]	M	AUTO_GET_TYPE
A[.]	M	AX_EMERGENCY_STOP_TIME
A[.]	M	AX_INERTIA
A[.]	M	AX_JERK_ENABLE

Figure 14-24 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.

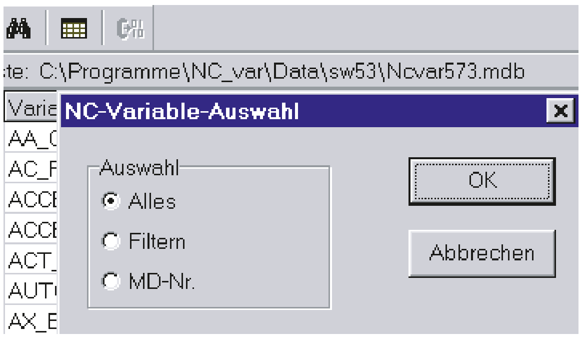


Figure 14-25 Window with filter criteria for displaying list of variables

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

- Select variable
  A variable is selected by means of a simple mouse click and transferred to the window of selected variables by double-clicking. This action can also be undone under the "Edit" menu item.

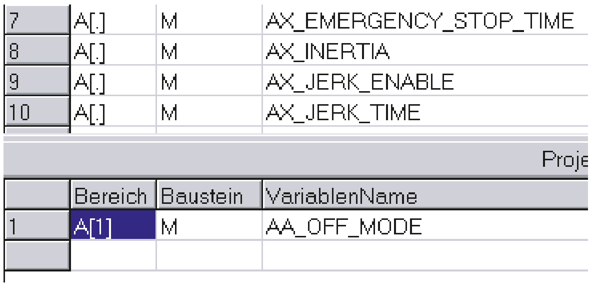
Alias name

The variable names provided can be up to 32 characters in length. To make variables clearly identifiable in the data block to be generated, several ASCII characters are added to the selected name. However, the STEP 7 compiler recognizes only 24 ASCII characters as an unambiguous STEP 7 variable. Since it cannot be precluded that variable names can only be differentiated by the last 8 character positions, **ALIAS** names are used for names, which are too long. When a variable is selected, the length of the STEP 7 name to be used is, therefore, checked. If the name is longer than 24 characters, the user must enter an additional name, which is then used as the alias.

In this case, the user must ensure that the alias name is unambiguous.

Alias input can always be activated by the user in the "Options" menu.
An alias name can then be entered every time a variable is transferred.

It is also possible to edit alias names at a later point in time by double-clicking on the S7 variable name field. This action can also be undone under the "Edit" menu item.



7	A[.]	M	AX_EMERGENCY_STOP_TIME
8	A[.]	M	AX_INERTIA
9	A[.]	M	AX_JERK_ENABLE
10	A[.]	M	AX_JERK_TIME

Proje

	Bereich	Baustein	VariablenName
1	A[1]	M	AA_OFF_MODE

Figure 14-26 Screen with complete list and selected variables

Scrolling

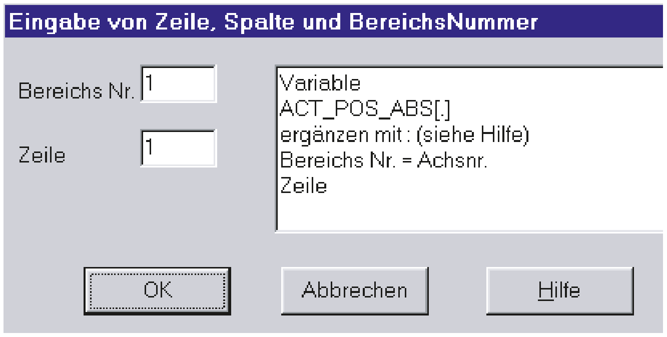
A scroll bar is displayed if it is not possible to display all variables in the window. The remaining variables can be reached by scrolling (page up/down).

Variables in multi-dimensional structures

If variables are selected from multi-dimensional structures, then the column and/or line number as well as the area number must be entered so that the variables can be addressed. The required numbers can be found in the NC variables documentation.

References:
Lists sl (Book1); Variables

By entering a zero (0) as the block number or the line or column index, it is possible to use the variable in the S7 PLC as a pointer to these data. When reading or writing these data via the functions "PUT" and "GET", the optional parameters "UnitX", "ColumnX" and "LineX" must be filled with the necessary information.


Eingabe von Zeile, Spalte und BereichsNummer

Bereichs Nr. 1

Zeile 1

Variable
ACT_POS_ABS[.]
ergänzen mit : (siehe Hilfe)
Bereichs Nr. = Achsnr.
Zeile

Figure 14-27 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 double-click function. It is possible to select several variables for deletion (see Section "Example of search criteria > Selecting variables").

This action can also be undone under the "Edit" menu item.

Note

Deleting of variables results in a change of the absolute addresses of the pointer structures to the variables. When changing the variable selection, it is, therefore, absolutely necessary to **generate** one or several **text files of all user blocksprior to the change**. This is the only way to ensure that the assignment of the variables in FB "GET" or FB "PUT" remains correct, even after recompilation.

Storing a selected list

Once variables have been selected, they can be stored under a project name. The files are stored on a project-specific basis.

A window is displayed for the file to be stored. The project path and name for the file must be selected in the window.

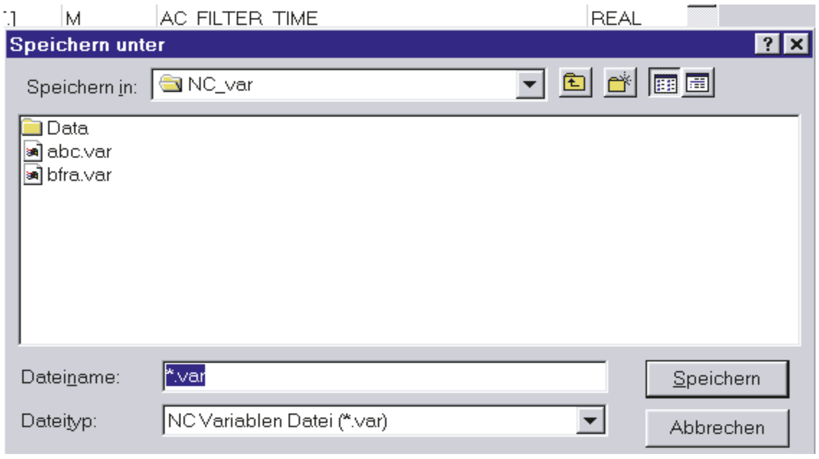


Figure 14-28 Window for project path and name of file to be stored

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 "Mass System" tab, a selection is made to determine how the unit system variables are calculated in the PLC.

Under the "Generate" tab, the project creation is defined for the relevant target system.

Generate

Under this menu item, the STEP 7 file from the selected variable list with extension ".awl" is set.

A file is generated when "Select" is clicked:

An **.awl** file that can be used as an input for the STEP 7 compiler.

A window opens, in which path and name for the .awl file to be generated must be specified for the file to be saved.

In STEP 7 project

The generated STL file is transferred to a selectable SIMATIC project (program path) and compiled. Furthermore, the symbol can also be transferred. This function is available as of STEP 7 Version 5.1. This process takes a longer time owing to the call of STEP 7. Before transferring a new STL file the file window of the STL file is to be closed in the LAD/FBD/STL editor.

Option menu item

The following can be selected under the "Option" menu item:

- The current language
- The mode for alias input (always / > 24 characters)

Help menu item

The information below can be viewed by selecting the corresponding submenu item:

- The Operating Manual
- The Description of Variables

The copyright and the version number can also be displayed.

14.16.2.3 Startup, installation

The Windows application "NC Var selector" is installed using the SETUP program supplied with the package.

14.17 Block descriptions

14.17.1 FB1: RUN_UP - basic program, start section

Function

The synchronization of NC and PLC is performed during start-up. The data blocks for the NC/PLC user interface are created based on the NC configuration defined via machine data and the basic program parameters are verified for plausibility. In the event of an error, FB1 passes an error identifier to the diagnostic buffer and switches the PLC to the stop state.

"Restart" start-up mode

The integrated PLC only supports the start-up mode "Restart". After the basic system initialization, the organization block OB100 "Restart" is always run through first, followed by OB1 "Cyclic mode".

Input parameters

For parameterizing the basic program, only the respective relevant parameters of the function block FB1 must be written with user-specific values. The preset values in the instance data block DB7 of the FB1 do not need to be assigned. The function block FB1 must only be called in the organization block OB100.

Output parameters

The output parameters of function block FB1 can also be read from the cyclical part of the basic program. Two options are available for this purpose:

1. Direct access to the instance data block DB7 of FB1 in symbolic form.
   Example: "L gp_par.MaxChan", with "gp_par" as the symbolic name of DB7
2. A bit memory is assigned to an output parameter when FB1 is parameterized. The bit memory is then read in the basic program in order to determine the value of the output parameter.
   Example: "MaxChan":= MW 20

Note

For assigning the FB1 parameters for MCP and BHG, see "Configuration machine control panel, handheld unit, direct keys (Page 918) ".

Declaration of the function

```
FUNCTION_BLOCK FB1
VAR_INPUT
   MCPNum:            INT:=1;              // 0: No MCP
                                           // 1: 1 MCP (default)
                                           // 2: 2 MCPs
   MCP1In:            POINTER;             // Start addr. MCP1 input signals
```

```
MCP1Out:           POINTER;            // Start addr. MCP1 output signals
MCP1StatSend:      POINTER;            // Status DW for sending MCP1
MCP1StatRec:       POINTER;            // Status DW for receiving MCP1
MCP1BusAdr:        INT:=6;             // default
MCP1Timeout:       S5TIME:= S5T#700MS;
MCP1Cycl:          S5TIME:= S5T#200MS;
MCP2In:            POINTER;            // Start addr. MCP2 input signals
MCP2Out:           POINTER;            // Start addr. MCP2 output signals
MCP2StatSend:      POINTER;            // Status DW for sending MCP2
MCP2StatRec:       POINTER;            // Status DW for receiving MCP2
MCP2BusAdr:        INT;
MCP2Timeout:       S5TIME:= S5T#700MS;
MCP2Cycl:          S5TIME:= S5T#200MS;
MCPMPI:            BOOL:= FALSE;
MCP1Stop:          BOOL:= FALSE;
MCP2Stop:          BOOL:= FALSE;
MCP1NotSend:       BOOL:= FALSE;
MCP2NotSend:       BOOL:= FALSE;
MCPSDB210:         BOOL:= FALSE;
MCPCopyDB77:       BOOL:= FALSE;
MCPBusType:        BYTE=B#16#0;
HHU:               INT:=0;             // Handheld unit interface
                                       // 0: No HHU
                                       // 1: HHU on MPI
                                       // 2: HHU on OPI
BHGIn:             POINTER;            // Transmit data of the handheld
                                       unit
BHGOut:            POINTER;            // Receive data of the handheld
                                       unit
BHGStatSend:       POINTER;            // Status DW for sending HHU
BHGStatRec:        POINTER;            // Status DW for receiving HHU
BHGInLen:          BYTE:= B#16#6;      // Input 6 bytes
BHGOutLen:         BYTE:= B#16#14;     // Output 20 bytes
BHGTimeout:        S5TIME:= S5T#700MS;
BHGCycl:           S5TIME:= S5T#100MS;
BHGRecGDNo:        INT:=2;
BHGRecGBZNo:       INT:=2;
BHGRecObjNo:       INT:=1;
BHGSendGDNo:       INT:=2;
BHGSendGBZNo:      INT:=1;
BHGSendObjNo:      INT:=1;
BHGMPI:            BOOL:= FALSE;
BHGStop:           BOOL:= FALSE;
BHGNotSend:        BOOL:= FALSE;
NCCyclTimeout:     S5TIME:= S5T#200MS;
NCRunupTimeout:    S5TIME:= S5T#50S;
```

```
   ListMDecGrp:      INT:=0;
   NCKomm:           BOOL:= FALSE;
   MMCToIF:          BOOL:=TRUE;
   HWheelMMC:        BOOL:=TRUE;         // Handwheel selection via HMI
   ExtendAlMsg :     BOOL;
   MCP_IF_TCS:       BOOL;
   ExtendChanAxMsg:  BOOL;
   MsgUser:          INT:=10;            // Number of user areas in DB 2
   UserIR:           BOOL:= FALSE;       // User programs in OB40,
                                         // Observe local data expansion!
   IRAuxfuT:         BOOL:= FALSE;       // Evaluate T function in OB40
   IRAuxfuH:         BOOL:= FALSE;       // Evaluate H function in OB40
   IRAuxfuE:         BOOL:= FALSE;       // Evaluate DL function in OB40
   UserVersion:      POINTER;            // Pointer to string variable,
                                         which is displayed in the version
                                         display of the user interface
   OpKeyNum :        INT;
   Op1KeyIn          POINTER;
   Op1KeyOut :       POINTER;
   Op1KeyBusAdr :    INT;
   Op2KeyIn :        POINTER;
   Op2KeyOut :       POINTER;
   Op2KeyBusAdr :    INT;
   Op1KeyStop :      BOOL;
   Op2KeyStop :      BOOL;
   Op1KeyNotSend :   BOOL;
   Op2KeyNotSend :   BOOL;
   OpKeyBusType :    BYTE ;
   IdentMcpBusAdr :  INT;
   IdentMcpProfilNo :BYTE ;
   IdentMcpBusType : BYTE ;
   IdentMcpStrobe :  BOOL;
END_VAR

VAR_OUTPUT
   MaxBAG:           INT;
   MaxChan:          INT;
   MaxAxis:          INT;
   ActivChan:        ARRAY[1..10] OF
                     BOOL;
   ActivAxis:        ARRAY[1..31] OF
                     BOOL;
   UDInt :           INT;
   UDHex:            INT;
   UDReal :          INT;
   IdentMcpType :    BYTE ;
```

```
        IdentMcpLengthIn :BYTE ;
        IdentMcpLengthOut BYTE ;
        :
      END_VAR
```

Description of formal parameters

Signal	Type	Type	Value range	Meaning
MCPNum:	I	INT	0, 1, 2	Number of active MCPs 0: No MCPs available
MCP1In: MCP2In:	I	POINTER	E0.0 ... E120.0 or M0.0 ... M248.0 or DBn DBX0.0 ... DBXm.0	Start address for the input signals of the relevant machine control panel
MCP1Out: MCP2Out:	I	POINTER	A0.0 ... A120.0 or M0.0 ... M248.0 or DBn DBX0.0 ... DBXm.0	Start address for the output signals of the relevant machine control panel
MCP1StatSend: MCP2StatSend:	I	POINTER	A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0	Only for Ethernet MCP: Switch off flashing (see Section "Switch-over of machine control panel, handheld unit (Page 927)")
MCP1StatRec: MCP2StatRec:	I	POINTER	A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0	Currently no significance
MCP1BusAdr: MCP2BusAdr:	I	INT	1, 2, 3 ... 126	DP slave: PROFIBUS address
			192, 193, 194 .. 223	Ethernet MCP: DIP setting
MCP1Timeout: MCP2Timeout:	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for machine control panel
MCP1Cycl: MCP2Cycl:	I	S5time	Recommendation: 200 ms	Relevant only for PROFIBUS
MCPMPI:	I	BOOL	FALSE	Available owing to compatibility
MCP1Stop: MCP2Stop:	I	BOOL	0 (FALSE), 1 (TRUE)	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 (FALSE), 1 (TRUE)	0: Send and receive operation activated 1: Receive machine control panel signals only
MCPSDB210:	I	BOOL	false	Available owing to compatibility
MCPCopyDB77:	I	BOOL	false	Available owing to compatibility

Signal	Typ e	Type	Value range	Meaning
MCPBusType:	I	BYTE	3, 4, 5, 6	b#16#yx: ● Bus type MCP1: lower nibble (x) ● Bus type MCP2: upper nibble (y) 3: PROFIBUS 4: PROFIBUS on the MPI/DP port 5: Ethernet 6: PROFINET Mixed mode is possible (see Section "Con-figuration machine control panel, handheld unit, direct keys (Page 918)")
HHU:	I	INT	0, 5	Handheld unit interface 0: No HHU 5: HHU on Ethernet
BHGIn:	I	POINTER	E0.0 ... E124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0	Start address PLC receive data from HHU
BHGOut:	I	POINTER	A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0	Start address PLC transmit data to HHU
BHGStatSend:	I	POINTER	A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0	Available owing to compatibility
BHGStatRec:	I	POINTER	A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0	Available owing to compatibility
BHGInLen:	I	BYTE	HHU default: B#16#6 (6 Byte)	Available owing to compatibility
BHGOutLen:	I	BYTE	HHU default: B#16#14 (20 Byte)	Available owing to compatibility
BHGTimeout:	I	S5time	Recommendation: 700 ms	Available owing to compatibility
BHGCycl:	I	S5time	Recommendation: 100 ms	Available owing to compatibility
BHGRecGDNo:	I	INT	HHU default: 2	Ethernet DIP switch
BHGRecGBZNo:	I	INT	HHU default: 2	Available owing to compatibility
BHGRecObjNo:	I	INT	HHU default: 1	Available owing to compatibility
BHGSendGDNo:	I	INT	HHU default: 2	Available owing to compatibility
BHGSendGBZNo:	I	Int	HHU default: 1	Available owing to compatibility
BHGSendObjNo:	I	INT	HHU default: 1	Available owing to compatibility
BHGMPI:	I	BOOL	0 (FALSE)	Available owing to compatibility

Signal	Type	Type	Value range	Meaning
BHGStop:	I	BOOL	0 (FALSE), 1 (TRUE)	0: Start transmission of handheld unit signals 1: Stop transmission of handheld unit signals
BHGNotSend:	I	BOOL	0 (FALSE), 1 (TRUE)	0: Send and receive operation activated 1: Receive handheld unit signals only
NCCyclTimeout:	I	S5time	Recommendation: 200 ms	Cyclic sign-of-life monitoring NC
NCRunupTimeout:	I	S5time	Recommendation: 50 s	Power-up monitoring NC
ListMDecGrp:	I	INT	0, 1, 2 ... 16	Activation of expanded M group decoding 0: Not active 1 to 16: Number of M groups
NCKomm:	I	BOOL	0 (FALSE), 1 (TRUE)	PLC NC communication services FB2, 3, 4, 5, 7 1: Active
MMCToIF:	I	BOOL	0 (FALSE), 1 (TRUE)	Transmission of HMI signals to interface (modes, program control, etc.) 1: Active
HWheelMMC:	I	BOOL	0 (FALSE), 1 (TRUE)	0: Handwheel selection via HMI 1: Handwheel selection via user program
ExtendAlMsg :	I	BOOL	0 (FALSE), 1 (TRUE)	Activation of extensions, error and operational messages of the FC10 (see Section "Extensions of the PLC alarms via block FC10" in Section "Interface PLC/HMI (Page 892)")
MCP_IF_TCS	I	BOOL	0 (FALSE), 1 (TRUE)	FC19: Evaluation R11 key 0: No evaluation 1: R11 key acts as manual traversing in tool orientation (see Section "Cartesian manual travel" in Section "FC19: MCP_IFM - transfer of MCP signals to interface (Page 1081)")
ExtendChanAxMsg	I	BOOL	0 (FALSE), 1 (TRUE)	Activation of all areas for error- and operational messages of the FC10 (see Section "FC10: AL_MSG - error and operating messages (Page 1061)")
MsgUser:	I	INT	0, 1, 2 ... 64	Number of user areas (DB2) – depends on parameter "ExtendAIMsg"
UserIR:	I	BOOL	0 (FALSE), 1 (TRUE)	Local data extension OB40 required for processing of signals from the user
IRAuxfuT:	I	BOOL	0 (FALSE), 1 (TRUE)	Evaluate T function in OB40
IRAuxfuH:	I	BOOL	0 (FALSE), 1 (TRUE)	Evaluate H function in OB40
IRAuxfuE:	I	BOOL	0 (FALSE), 1 (TRUE)	Evaluate DL function in OB40

Signal	Typ e	Type	Value range	Meaning
UserVersion:	I	POINTER	P#DBn.DBXx.0	Pointer to string variable, which is displayed in the version display of the user interface The string variable has the following notation (max. 41 characters): **"<Name>** <version **xx.xx.xx>** <date **yy/mm/dd>"** Example: "Test version 07.06.02 13/06/04"
OpKeyNum :	I	INT	0, 1, 2	Number of active direct control key modules 0: No Ethernet direct control keys available
Op1KeyIn: Op2KeyIn:	I	POINTER	P#Ex.0 or P#Mx.0 or P#DBn.DBXx.0.	Start address for the input signals of the affected direct control key modules
Op1KeyOut : Op2KeyOut :	I	POINTER	P#Ax.0 or P#Mx.0 or P#DBn.DBXx.0.	Start address for the output signals of the affected direct control key modules
Op1KeyBusAdr : Op2KeyBusAdr :	I	INT	1, 2, 3 ... 191	Direct control keys via Ethernet: TCU index:
Op1KeyStop : Op2KeyStop :	I	BOOL	0 (FALSE), 1 (TRUE)	0: Start transmission of direct control key signals 1: Stop transmission of direct control key signals
Op1KeyNotSend : Op2KeyNotSend :	I	BOOL	0 (FALSE), 1 (TRUE)	0: Send and receive operation activated 1: Receive direct control key signals only
OpKeyBusType :	I	BYTE	b#16#55	Ethernet
IdentMcpBusAdr :	I	INT	1, 2, 3 ... 254	only IE devices
IdentMcpProfilNo :	I	BYTE	0, 1	Profile of a device 0: Complete device 1: Only direct control keys
IdentMcpBusType :	I	BYTE	b#16#5	only IE devices
IdentMcpStrobe :	I	BOOL	0 (FALSE), 1 (TRUE)	1: Activate query
MaxBAG:	O	INT	1, 2, 3 ... 10	Number of mode groups
MaxChan:	O	INT	1, 2, 3 ... 10	Number of channels
MaxAxis:	O	INT	1, 2, 3 ... 31	Number of axes
ActivChan:	O	ARRAY[1...10] OF BOOL	0 (FALSE), 1 (TRUE)	Bit string for active channels
ActivAxis:	O	ARRAY [1..31] OF BOOL	0 (FALSE), 1 (TRUE)	Bit string for active axes
UDInt :	O	INT	0,	Quantity of INTEGER machine data in DB20

Signal	Type	Type	Value range	Meaning
UDHex:	O	INT	---	Quantity of hexadecimal machine data in DB20
UDReal :	O	INT	---	Quantity of REAL machine data in DB20
IdentMcpType :	O	BYTE	0, B#16#80, B#16#81, B#16#82 ... B#16#89	MCP type
IdentMcpLengthIn :	O	BYTE	---	Length of MCP input data (MCP → PLC)
IdentMcpLengthOut :	O	BYTE	---	Length of MCP output data (PLC → MCP)

If an error occurs during communication with a machine control panel (MCP) or handheld unit (HHU), the following alarms are displayed on the HMI and the input signals (MCP1In, MCP2In or BHGIn) are set to the value zero:

Error case: MCP/HHU

- Alarm 400260: "MCP 1 failure"
- Alarm 400261: "MCP 2 failure"
- Alarm 400262: "HHU failure"

If resynchronization is possible between PLC and MCP/HHU, communication is resumed, the error message on the HMI is deleted by the basic program, and process values are transferred to the input signals (MCP1In, MCP2In or BHGIn) again.

Example: FB1 call in the OB100

```
ORGANIZATION_BLOCK OB100
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 FB1, DB7(               // FB1 call, instance DB: DB7
          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
```

14.17.2 FB2: GET - read NC variable

Function

The FB2 "GET" function block is used to read variables from the NC area.

In order to reference the NC variables, they are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this data block in the S7 symbol table. When calling FB2, the variable addresses are transferred in the following form: Parameter "Addr1" to "Addr8" = "<DB name>.<S7 name>"

Request for reading NC variables

Call of FB2 with positive edge change, parameter "Req" = 0 → 1

S7 names of the NC variables: Parameter "Addr1" to "Addr8" = "NCVAR".<S7 name>"

Pointer for writing the variable values: Parameters "RD1" to "RD8" = "P#<Address>"

Completion of the read request

Read request successfully completed: Parameter "Done" == 1.

Read request completed with error:"Parameter "Error" == 1, error cause in parameter "State"

Prerequisites

- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block DB120 (data interface), the S7 Symbol Editor must be used to assign a symbol (default: NCVAR) in the S7 symbol table of the S7 project. Using this symbol, the NC variable is then specified in the FB2 parameters "Addr<x>", e.g. "ADDR1": = "NCVAR".<NC variable>"

General conditions

- FB2 has multi-instance capability.
- Every call of FB2 must be assigned a separate instance DB from the user area.
- When **channel-specific** variables are read, only variables from exactly **one** channel may be addressed via "Addr1" to "Addr8" if FB2 is called.
- When **drive-specific** variables are read, only variables from exactly **one** SERVO drive object may be addressed via "Addr1" to "Addr8" if FB2 is called. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.
- In a read job, only variables from the same area, channel, or drive object can be read.

Note

Error case
When reading variables from different channels or drive objects, or simultaneously from a channel and a drive object, an error message is output:
- "Error" == TRUE
- "State" == W#16#02

Variable addressing

For some NC variables, it is necessary to select "Area no." and/or "Line" or "Column" from the NC VAR selector. It is possible to select a basic type, i.e. "Area no.", "Line" and "Column" are preassigned "0". The values of the "Area no.", "Line" and "Column" specified by the NC VAR selector are checked for a "0" in FB2. If an NC-VAR selector value == "0", the corresponding value of the FB2 parameter is adopted. To do this, the FB2 parameters "Unit<x>", "Column<x>" and "Line<x>", with <x> = 1 - 8, must be written before FB2 is called.

Table 14-1 Parameter match

FB2 parameter	NC VAR selector
Unit	Area no.
Column	Column
Line	Line

Variables within **one** group can be combined in a job:

Group	Area				
1	C[1]	N	B	O	T
2	C[2]	N	B	O	T
3	V[.]	H[.]	---	---	---
The same rules apply for channels 3 to 10 as for group 1 and group 2 shown in the example.					

Note

The number of usable variables can be less than eight when simultaneously reading several variables of the "String" type.

Declaration of the function

```
FUNCTION_BLOCK FB2
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

Parameter	Type	Type	Value range	Description
Req:	I	BOOL	---	Job start with positive signal edge
NumVar:	I	INT	1 ... 8	Number of variables to be read: "Addr1" - "Addr8"
Addr1 - Addr8:	I	ANY	"DBName".<VarName>	Variable identifiers from **NC Var selector**
Unit1 - Unit8:	I	BYTE	---	Area address, optional for variable addressing
Column1 - Column8:	I	WORD	---	Column address, optional for variable addressing
Line1 - Line8:	I	WORD	---	Line address, optional for variable addressing
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1: Negative acknowledgement of job or execution of job impossible
NDR :	O	BOOL	0 (FALSE), 1 (TRUE)	1: Job successfully executed Data is available
State:	O	WORD	---	See paragraph "Error identifiers"
RD1 - RD8:	I/O	ANY	P#Mm.n BYTE x ... P#DBnr.dbxm.n BYTE x	Target area for read data

Error identifiers

NC variables

State		Description	Note
High byte 1)	Low byte		
1 - 8	1	Access error	---
0	2	Error in job	Incorrect compilation of variables in a job

State		Description	Note
High byte 1)	Low byte		
0	3	Negative acknowledgement, job not executable	Internal error, possible remedy: • Check job data • NC reset
1 - 8	4	Insufficient local user memory available	Data type of the read variable is greater than specified in "RD1" - "RD8"
0	5	Format conversion error	Error on conversion of var. type double: Variable is not within the S7 REAL area
0	6	FIFO full	Job must be repeated since queue is full
0	7	Option not set	FB1 parameter "NCKomm" is not set
1 - 8	8	Incorrect target area (RD)	"RD1" - "RD8" must not be local data
0	9	Transmission occupied	Job must be repeated
1 - 8	10	Error in variable addressing	"Unit" or "Column"/"Line" contains value 0
0	11	Address of variable invalid	Check "Addr" (or variable name), "Area", "Unit"
0	12	NumVar == 0	Check parameter "NumVar"
1 - 8	13	ANY data reference incorrect	Requested "NcVar" data has not been parameterized
1) High byte > 0 ⇒ Number of the variable in which the error occurred			

Drive-specific variables

If an error occurs while reading/writing a drive-specific variable (DB1200.DBX3000.1 == 1), an error number is displayed in the access result which is based on the error numbers defined in the PROFIdrive profile.

State	Description
x	<Error number of PROFIdrive profile> + 20_H or 36_D

Determining the significance of the access result:

1. Calculation of the error number of the PROFIdrive profile
   <Error number of PROFIdrive profile> = result of access - 20_H or 36_D.
2. Determining the significance of the error number of the PROFIdrive profile
   The error numbers of the PROFIdrive profile are described in:
   References
   SINAMICS S120 Drive Functions Function Manual; Chapter "Communication" > "Communication according to PROFIdrive" > "Acyclic communication" > "Structure of orders and responses" > paragraph "Error values in parameter responses"

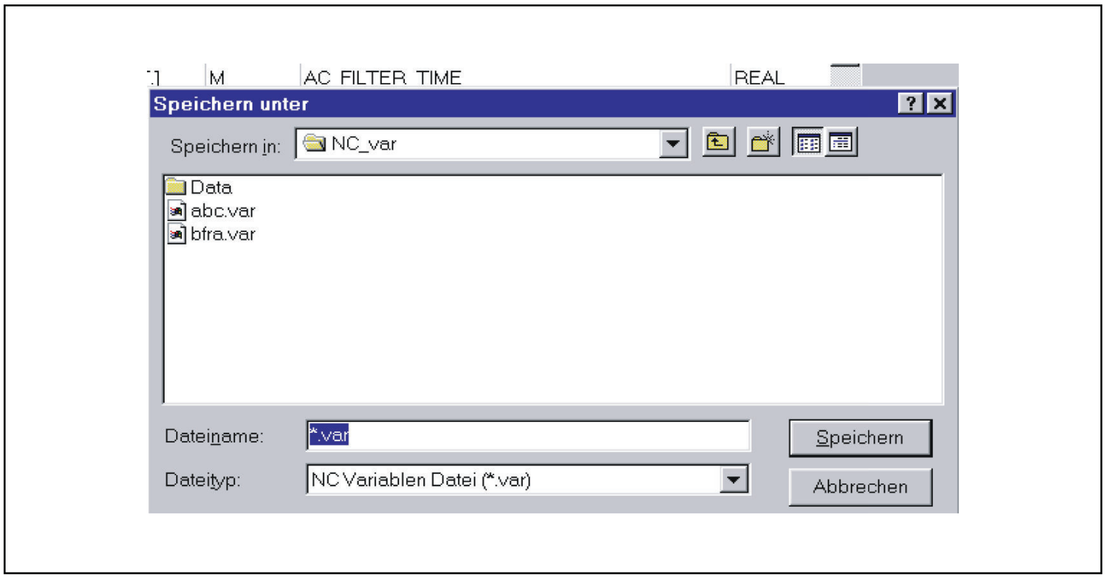
Configuration steps

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.
- Parameterizing the FB2

Pulse diagram



① User: Set request, Req = 0 → 1
② FB2 successfully completed, NDR = 1
User: Reset request, IF NDR == 1 THEN Req = 0
③ User: IF NDR == 1 THEN reset request: 1 → 0
④ FB2 reset job confirmation, NDR = 0
⑤ User: IF NDR == 0 AND Error == 0 THEN reset request Req = 1 → 0 **not permissible**
⑥ FB2 completed with errors, Error = 1
User: Reset request, IF NDR == 1 OR Error == 1 THEN Req = 0, possible further error handling

Call example

Reading three channel-specific machine data from channel 1, whose address specifications are stored in DB120.

Specification of data

The data is selected with the NC VAR selector and saved in the DB120.**VAR** file. Then the DB120.**AWL** file is created from this.

S7 (ALIAS) names are selected.

To adopt the channel designation into the variable name and to delete the characters "[" and "]", which are not permitted in a STEP 7 symbol, new S7 names are selected:

Area	Block	Name	Type	No.	Byte	S7 Name
C[1]	M	MD20070 $MC_AXCONF_MA-CHAX_USED[1]	CHAR	20070	1	C1AxConfMachAxUsed1
C[1]	M	MD20070 $MC_AXCONF_MA-CHAX_USED[2]	CHAR	20070	1	C1AxConfMachAxUsed2
C[1]	M	MD20090 $MC_SPIND_DEF_MAS-TER_SPIND	INT	20090	1	C1SpindDefMasterSpind

S7 symbol table

"NCVAR" is entered in the S7 symbol table as a symbolic name for the data block DB120:

Symbol	Operand	Data type
NCVAR	DB120	DB120

File DB120.AWL must be compiled and transferred to the PLC.

Parameterization of FB2 with instance DB110:

```
DATA_BLOCK DB110         // Unassigned user DB, as instance for FB2
FB2
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 FB2, DB110(
      Req :=           M 100.0,
      NumVar :=        3,             // Read three variables
      Addr1 :=         "NCVAR".C1AxConfMachAxUsed1,
      Addr2 :=         "NCVAR".C1AxConfMachAxUsed2,
      Addr3 :=         "NCVAR".C1SpindDefMasterSpind,
      Error :=         M102.0,
      NDR :=           M100.1,
      State :=         MW104,
      RD1 :=           P#DB99.DBX0.0 BYTE 1,
      RD2 :=           P#DB99.DBX1.0 BYTE 1,
      RD3 :=           P#M110.0 INT 1);
```

Example: Variable addressing

Reading 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 "Line<x>".

```
DATA_BLOCK DB120
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 FB2, DB110(
   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);
```

Classification of data types

Table 14-2 Classification of data types

NC-internal or BTSS data type	S7 data type
DOUBLE	REAL
DOUBLE	REAL2
REAL	REAL
LONG	DINT
INTEGER	DINT
UINT_32	DWORD

NC-internal or BTSS data type	S7 data type
INT_16	INT
UINT_16	WORD
UNSIGNED	WORD
CHAR	CHAR or BYTE
STRING	STRING
BOOL	BOOL
DATETIME	DATE_AND_TIME

Example

For example, to be able to read an NC variable of the type DOUBLE without adapting the format, an ANY pointer with REAL2 type must be specified in the destination area "RDx" (e.g.: P#M100.0 REAL2). If the basic program recognizes REAL2 as the target type when reading a variable of the DOUBLE type, the data is transferred to the PLC data area as a 64-bit floating-point number.

14.17.3 FB3: PUT - write NC variables

Function

The FB3 "PUT" function block is used to write variables from the NC area.

In order to reference the NC variables, they are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this data block in the S7 symbol table. When calling FB3, the variable addresses are transferred in the following form: Parameter "Addr1" to "Addr8" = "<DB name>".<S7 name>

Request for writing NC variables

Call of FB3 with positive edge change, parameter "Req" = 0 → 1

S7 names of the NC variables: Parameter "Addr1" to "Addr8" = "NCVAR".<S7 name>"

Pointer for writing the variable values: Parameters "RD1" to "RD8" = "P#<Address>"

Completion of the write request

Write request successfully completed: Parameter "Done" == 1.

Write request terminated with error: Parameter "Error" == 1, error cause in parameter "State"

Prerequisites

- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block DB120 (data interface), the S7 Symbol Editor must be used to assign a symbol (default: NCVAR) in the S7 symbol table of the S7 project. Via this symbol, the NC variable is then specified in the FB3 parameters "Addr<x>", e.g. "ADDR1":= "NCVAR".<NC-Variable>"

General conditions

- FB3 has multi-instance capability.
- Every call of FB3 must be assigned a separate instance DB from the user area.
- In order to define machine data and GUD without a password, the protection level of the data you want to access must be redefined to the lowest level.
  References:
  - Commissioning Manual; Section: "Protection levels concept"
  - Programming Manual, Job Planning; Section: "Define protection levels for user data"
- When **channel-specific** variables are written, only variables from exactly **one** channel may be addressed via "Addr1" to "Addr8" if FB2 is called.
- When **drive-specific** variables are written, only variables from exactly **one** SERVO drive object may be addressed via "Addr1" to "Addr8" if FB2 is called. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.
- In a write job, only variables from the same area, channel, or drive object can be written.

Note

Error case
When writing variables from different channels or drive objects, or simultaneously from a channel and a drive object, an error message is output:

- "Error" == TRUE
- "State" == W#16#02

Variable addressing

For some NC variables, it is necessary to select "Area no." and/or "Line" or "Column" from the NC VAR selector. It is possible to select a basic type, i.e. "Area no.", "Line" and "Column" are preassigned "0". The values of the "Area no.", "Line" and "Column" specified by the NC VAR selector are checked for a "0" in FB2. If an NC-VAR selector value == "0", the corresponding value of the FB3 parameter is adopted. To do this, the FB3 parameters "Unit<x>", "Column<x>" and "Line<x>", with <x> = 1 - 8, must be written before FB3 is called.

FB3 parameter	NC VAR selector
Unit<x>	Area no.
Column<x>	Column
Line<x>	Line

NC variables within **one** group can be combined in a job:

Group	Area				
1	C[1]	N	B	O	T
2	C[2]	N	B	O	T
3	V[.]	H[.]	---	---	---
The same rules apply for channels 3 to 10 as for group 1 and group 2 shown in the example.					

Note

The number of usable variables can be less than eight when simultaneously writing several variables of the "String" type.

Declaration of the function

```
FUNCTION_BLOCK FB3
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

Signal	Type	Type	Value range	Meaning
Req:	I	BOOL	-	Job start with positive signal edge
NumVar:	I	INT	1 ... 8	Number of variables to be written: "Addr1" - "Addr8"
Addr1 - Addr8:	I	ANY	"DBName".<VarName>	Variable identifiers from **NC Var selector**
Unit1 - Unit8:	I	BYTE	-	Area address, optional for variable addressing
Column1 - Column8:	I	WORD	-	Column address, optional for variable addressing
Line1 - Line8:	I	WORD	-	Line address, optional for variable addressing
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	Negative acknowledgment of job or execution of job impossible
Done:	O	BOOL	0 (FALSE), 1 (TRUE)	Job successfully executed
State:	O	WORD	-	See paragraph "Error identifiers"
SD1 - SD8:	I/O	ANY	P#Mm.n BYTE x... P#DBnr.dbxm.n BYTE x	Data to be written

Error identifiers

NC variables

State		Meaning	Note
High byte 1)	Low byte		
1 - 8	1	Access error	---
0	2	Error in job	Incorrect compilation of variables in a job
0	3	Negative acknowledgment, job not executable	Internal error, try: • Check job • NC reset
1 - 8	4	Data areas or data types do not match or string is empty	Check data in "SD1" - "SD8"
0	6	FIFO full	Job must be repeated since queue is full
0	7	Option not set	FB1 parameter "NCKomm" is not set
1 - 8	8	Incorrect target area (SD)	"SD1" - "SD8" must not be local data
0	9	Transmission occupied	Job must be repeated
1 - 8	10	Error in variable addressing	"Unit" or "Column"/"Line" contains value 0
0	11	Variable address invalid or variable is read-only	Check "Addr" (or variable name), "Area", "Unit"
0	12	NumVar == 0	Check parameter "NumVar"
1 - 8	13	ANY data reference incorrect	Requested "NcVar" data has not been parameterized
1 - 8	15	User data too long	Remedy: Write fewer variables per job or use shorter string variables
1) High byte > 0 ⇒ Number of the variable in which the error occurred			

Drive-specific variables

If an error occurs while reading/writing a drive-specific variable (DB1200.DBX3000.1 == 1), an error number is displayed in the access result which is based on the error numbers defined in the PROFIdrive profile.

State	Meaning
x	<Error number of PROFIdrive profile> + 20_H or 36_D

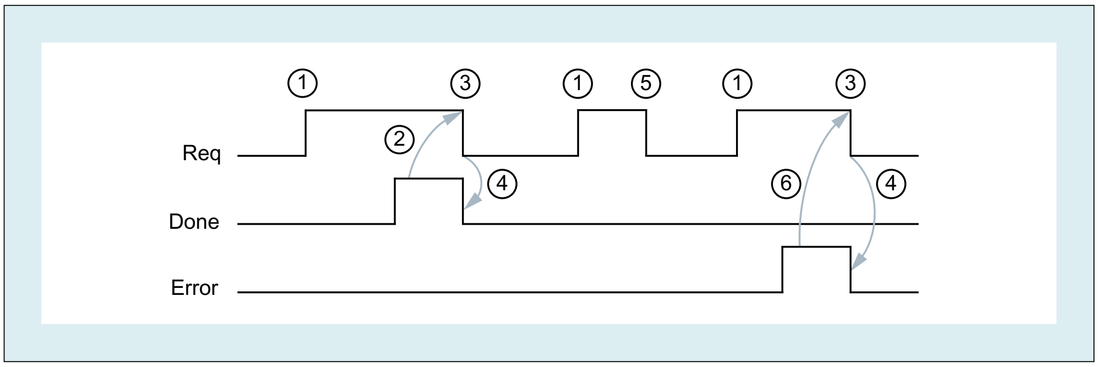
Determining the significance of the access result:

1. Calculation of the error number of the PROFIdrive profile
   <Error number of PROFIdrive profile> = result of access - 20_H or 36_D.
2. Determining the significance of the error number of the PROFIdrive profile
   The error numbers of the PROFIdrive profile are described in:
   References
   SINAMICS S120 Drive Functions Function Manual; Chapter "Communication" > "Communication according to PROFIdrive" > "Acyclic communication" > "Structure of orders and responses" > paragraph "Error values in parameter responses"

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 User: Set request, Req = 0 → 1

② FB3 successfully completed, Done = 1
User: Reset request, IF Done == 1 THEN Req = 0

③ User: IF Done == 1 THEN reset request: 1 → 0

④ FB3 reset job confirmation, Done = 0

⑤ User: IF Done == 0 AND Error == 0 THEN reset request Req = 1 → 0 **not permissible**

⑥ FB3 completed with errors, Error = 1
User: Reset request, IF Done == 1 OR Error == 1 THEN Req = 0, possible further error handling

Call example

Writing of three channel-specific machine data items of channel 1:

Selection of the three data items with NC VAR selector and storage in file DB120.VAR

S7 (ALIAS) names are selected in order to adopt the block designation into the name and to remove the characters [ ], which are not permitted in a STEP 7 symbol.

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 DB120 with the S7 SYMBOL Editor

Symbol	Operand	Data type
NCVAR	DB120	DB120

File DB120.AWL must be compiled and transferred to the PLC.

Calling and parameterizing the FB3 with instance DB111

```
DATA_BLOCK DB111        // Unassigned user DB, as instance for FB3
FB3
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 FB3, DB111(
      Req :=            M 100.0,
      NumVar :=         3,                       // Write three variables
      Addr1 :=          "NCVAR".rpa_5C1RP,
      Addr2 :=          "NCVAR".rpa_11C1RP,
      Addr3 :=          "NCVAR".rpa_14C1RP,
      Error :=          M102.0,
      Done :=           M100.1,
      State :=          MW104,
      SD1 :=            P#DB99.DBX0.0 REAL 1,
      SD2 :=            P#DB99.DBX4.0 REAL 1,
      SD3 :=            P#M110.0 REAL 1);
```

Example: Variable addressing

Writing 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 DB120
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 FB3, DB122(
   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);
```

Classification of data types

See table "Assignment of the data types" in Chapter "FB2: GET - read NC variable (Page 973)"

14.17.4 FB4: PI_SERV - request PI service

Function

The function block FB4 "PI_SERV" is used to start PI services.

The available PI services are described in the following chapters with their specific parameters. An overview of the available PI services can be found in: List of available PI services (Page 992)

Note

Due to the large number of "WVar" parameters, it is recommended that you use the function block FB7 instead of FB4. See Chapter "FB7: PI_SERV2 - request PI service (Page 1025)".

Start of a PI service

Request to start a PI service: Call of FB4 with positive edge change, parameter "Req" = 0 → 1

Completion of a PI service

Job or PI service successfully completed: Parameter "Done" == 1.

Job or PI service completed with error:"Parameter "Error" == 1, error cause in parameter "State"

Prerequisites

- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block DB16 (data interface of the PI services), the S7 Symbol Editor must be used to assign a symbol (default: PI) in the S7 symbol table of the S7 project. The requested PI service is then specified via this symbol in the FB4 parameter "PIService", e.g. "PIService:= "PI".<PI service>"

General conditions

- Every call of FB4 must be assigned a separate instance DB from the user area.
- The start of a PI service (FB4 call with "Req" = 1) is only permitted in the cyclic part of the PLC basic program (OB1).
  If the PI service is not started (FB4 call with "Req" = 0), the parameters can also be written in the start-up part of the PLC basic program (OB100). The PI service can then be started using the already written parameters in the cyclic part of the PLC basic program (OB1) by calling FB4 with "Req" = 1.
- The execution of the PI service generally extends over several PLC cycles.

Declaration of the function

```
FUNCTION_BLOCK FB4
VAR_INPUT
w   Req :                BOOL;
    PIService :          ANY;
    Unit :               INT;
    Addr1 :              ANY;
    Addr2 :              ANY;
    Addr3 :              ANY;
    Addr4 :              ANY;
    WVar1 :              WORD;
    WVar2 :              WORD;
    WVar3 :              WORD;
    WVar4 :              WORD;
    WVar5 :              WORD;
    WVar6 :              WORD;
    WVar7 :              WORD;
    WVar8 :              WORD;
    WVar9 :              WORD;
    WVar10 :             WORD;
END_VAR
VAR_OUTPUT
    Error :              BOOL;
    Done :               BOOL;
    State :              WORD;
END_VAR
```

Description of formal parameters

Signal	Type	Type	Value range	Description
Req:	I	BOOL	0 (FALSE), 1 (TRUE)	Job request
PIService:	I	ANY	"<DBName>".<PI service>	Requested PI service • <DBName>: symbol name for DB16, default: "PI" • <PI service>: List of available PI services (Page 992)
Unit:	I	INT	1, 2, 3 ...	Area number
Addr1 to Addr4:	I	ANY	"<DBName>".<VarName>	Reference to a string Significance depending on selected PI service
WVar1 to WVar10:	I	WORD	-32768 ... 32767 8000_H - $7FFF_H$	INTEGER or WORD variable Significance depending on selected PI service
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	Error status 1: Negative acknowledgment of job or execution of job impossible
Done:	O	BOOL	0 (FALSE), 1 (TRUE)	Job status 1: Job successfully executed
State:	O	WORD	1)	Error detection Only relevant if "Error" == 1

1)	State	Description	Note
	3	Negative acknowledgement, job not executable	Internal error, possible remedy through an NC RESET
	6	FIFO full	Repeat the command
	7	Option not set	FB1 parameter "NCKomm" is not set
	9	Transmission occupied	Repeat the command
	13	Addr1.. Adddr4: Reference invalid	Specify missing string
	14	PI service is unknown	The PI service specified in the "PIService" parameter is unknown. → check the notation/spelling
	15	Addr1.. Adddr4: String too long	Check string lengths

Call example

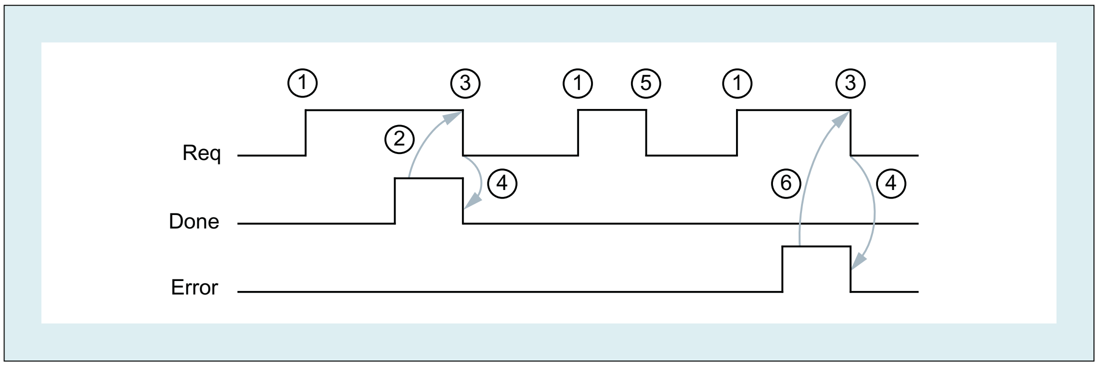
Function: Program selection in channel 1 (main program and workpiece program)

Entry of PI service for DB16 and STR for DB124 using the S7 SYMBOL editor:

Parameterization		
Symbol	**Operand**	**Data type**
PI	DB16	DB16
STR	DB124	DB124

```
DATA_BLOCK DB126         // Instance for FB4, unassigned user DB
FB4
BEGIN
END_DATA_BLOCK
// ------------------------------------------------------------------------
DATA_BLOCK DB124
   STRUCT
      PName:        string[32]:= '_N_TEST_MPF   // Main program
                    ';
      Path:         string[32]:= '/             // Main program path
                    _N_MPF_DIR/';
      PName_WST:    string[32]:= '_N_ABC_MPF';  // Workpiece program
      Path_WST:     string[32]:= '/_N_WCS_DIR/  // Workpiece program path
                    _N_ZYL_WPD';
   END_STRUCT
BEGIN
END_DATA_BLOCK
// -------------------------------------------------------------------------
Function FC "PICall" : VOID
   U  I 7.7;        // Unassigned machine control panel key
   S  M 0.0;        // Activate req.
   U  M 1.1;        // Done completed message
   R  M 0.0;        // Terminate job
   U  I 7.6;        // Manual error acknowledgment
   U  M 1.0;        // Error pending
   R  M 0.0;        // Terminate job

   CALL FB4, DB126 (
      Req:=         M0.0,
      PIService:=   "PI".SELECT,                // PI service: SELECT
      Unit:=        1,                          // Channel 1
      Addr1:=       "STR".Path,                 // Main program: Path
      Addr2:=       "STR".PName,                // Main program: Program
   // Addr1:=       "STR".Path_WST,             // Workpiece: Path
   // Addr2:=       "STR".PName_WST,            // Workpiece: Program
      Error:=       M1.0,
      Done:=        M1.1,
      State:=       MW2
   );
```

Flow diagram



① User: Set request, Req = 0 → 1

② FB4: PI service successfully completed, Done = 1
User: Reset request, IF Done == 1 THEN Req = 0

③ User: IF Done == 1 THEN reset request: 1 → 0

④ FB4: Reset job confirmation, Done = 0

⑤ User: IF Done == 0 AND Error == 0 THEN reset request Req = 1 → 0 **not permissible**

⑥ FB4: PI service completed with errors, Error = 1
User: Reset request, IF Done == 1 OR Error == 1 THEN Req = 0, possible further error handling

14.17.4.1 List of available PI services

General PI services

PI service	Function
ASUP (Page 993)	Assign interrupt
CANCEL (Page 994)	Execute cancel
CONFIG (Page 994)	Reconfiguration of tagged machine data
DIGION (Page 995)	Digitizing on
DIGIOF (Page 995)	Digitizing off
FINDBL (Page 995)	Activate block search
LOGIN (Page 996)	Activate password
LOGOUT (Page 996)	Reset password
NCRES (Page 996)	Trigger NC-RESET
SELECT (Page 997)	Select program for processing for one channel
SETUDT (Page 997)	Sets the current user data to active
SETUFR (Page 998)	Activate user frame
RETRAC (Page 998)	Retraction of the tool in the tool direction

PI services of tool management

PI service	Function
CRCEDN (Page 999)	Create a tool cutting edge with specification of the T number
CREACE (Page 1000)	Create a tool cutting edge with the next higher/free D number

PI service	Function
CREATO (Page 1000)	Create a tool with specification of a T number.
DELECE (Page 1000)	Delete a tool cutting edge
DELETO (Page 1001)	Delete tool
MMCSEM (Page 1001)	Semaphores for various PI services
TMCRTO (Page 1002)	Create a tool with specification of a name, a duplo number
TMFDPL (Page 1004)	Empty location search for loading
TMFPBP (Page 1005)	Empty location search
TMGETT (Page 1006)	T number for the specified tool name with duplo number
TMMVTL (Page 1007)	Prepare magazine location for loading, unload tool
TMPOSM (Page 1008)	Position magazine location or tool
TMPCIT (Page 1009)	Set increment value for workpiece counter
TMRASS (Page 1010)	Reset active status
TRESMO (Page 1010)	Reset monitoring values
TSEARC (Page 1011)	Complex search using search screen forms
TMCRMT (Page 1014)	Create multitool
TMDLMT (Page 1015)	Delete multitool
POSMT (Page 1015)	Position multitool
FDPLMT (Page 1016)	Search/check an empty location within the multitool

14.17.4.2 PI service: ASUP

Function: Assign interrupt

An interrupt number is assigned in the specified channel to a part program that is present in the control and is identified by path and program name.

The PI service ASUB has the same effect as the program instruction SETINT (or CLRINT if the PI service was called without specifying the path name).

Unlike SETINT (or CLRINT), the PI service ASUB remains active even after the end of the program (M30 or channel reset). An assignment activated by the PI service ASUB is not cleared until after a warm restart.

For detailed information about program management, path and file names, see:

References:
Programming Manual, Job Planning; Chapter: "File and Program Management" > "Program Memory".

Description of formal parameters

Signal	Type	Value range	Meaning
PIService:	ANY	"PI".ASUP	Assign interrupt
Unit:	INT	1, 2, 3, ... 10	Channel number
Addr1:	STRING		Path name, e.g. "/_N_MPF_DIR/"
Addr2	STRING		Program name, e.g. "_N_TST_FC9ASUP_MPF"

Signal	Type	Value range	Meaning
WVar1:	WORD	1, 2, 3, ... 8	Interrupt number
WVar2:	WORD	1, 2, 3, ... 8	Priority
WVar3:	WORD	0, 1	LIFTFAST [1] Fast retraction from the contour
WVar4:	WORD	0, 1	BLSYNC [2] Processing of interrupt routine is only to start with the next block change

[1] **References:**
Programming Manual, Job Planning; Chapter: "Flexible NC programming" **>** "Interrupt routine (ASUB)" **>** "Fast retraction from the contour (SETINT, **LIFTFAST**, ALF)"

[2] **References:**
Programming Manual, Job Planning; Chapter: "Flexible NC programming" **>** "Interrupt routine (ASUB)" **>** "Assign and start interrupt routine (SETINT, PRIO, **BLSYNC**)"

Note

The ASUB PI service may only be executed when the specified channel is in the reset state. An ASUB prepared with FB7 can be subsequently initiated with FC9.

References:
Programming Manual, Job Planning; Chapter: "Flexible NC Programming" > "Interrupt routine (ASUB)"

14.17.4.3 PI service: CANCEL

Function: Execute cancel

Triggers the "Cancel" function equivalent to the corresponding "Cancel alarm" button on the user interface (operator panel front).

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".CANCEL	Cancel

14.17.4.4 PI service: CONFIG

Function: Reconfiguration

The reconfiguration command activates machine data which has been entered sequentially by the operator or the PLC, almost in parallel.

The command can only be activated when the controller is in RESET state or the program is interrupted (NC stop at block limit). An FB4 error checkback signal is output if this condition is not fulfilled (state = 3).

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".CONFIG	Reconfiguration
Unit:	INT	1	
WVar1:	INT	1	Classification

14.17.4.5 PI service: DIGION

Function: Digitizing on

Selecting digitizing in the parameterized channel.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".DIGION	Digitizing on
Unit:	INT	1, 2, 3 ... 10	Channel

14.17.4.6 PI service: DIGIOF

Function: Digitizing off

Deactivating digitizing in the parameterized channel.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".DIGIOF	Digitizing off
Unit:	INT	1, 2, 3 ... 10	Channel

14.17.4.7 PI service: FINDBL

Function: Activate block search

A channel is switched to block search mode and the appropriate acknowledgment then transmitted. The block search is then executed immediately by the NC. The block search pointer must already be in the NC at this point in time. The block search can be interrupted at any time by an NC RESET. Once the block search is successfully completed, the normal processing mode is reactivated automatically. NC start then takes effect from the located search target.

It is the sole responsibility of the operator to ensure a collision-free approach path.

Description of formal parameters

Signal	Type	Value range	Meaning
PIService:	ANY	"PI".FINDBL	Block search
Unit:	INT	1 to 10	Channel
WVar1:	WORD	1, 2, 3	Preprocessing mode 1: Without calculation 2: With calculation 3: With main block consideration

14.17.4.8 PI service: LOGIN

Function: Create password

Transfers the parameterized password to the NC. The passwords generally consist of 8 characters. For shorter passwords, the string can be supplemented with empty spaces to attain 8 characters

Example

Password: STRING[8] := 'SUNRISE';

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".LOGIN	Create password
Unit:	INT	1	NC
Addr1:	STRING	8 characters	Password

14.17.4.9 PI service: LOGOUT

Function: Reset password

The password last transferred to the NC is reset.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".LOGOUT	Reset password
Unit:	INT	1	NC

14.17.4.10 PI service: NCRES

Function: Trigger NC-RESET

Triggers an NC-RESET. Parameters "Unit" and "WVar1" must always be set to 0.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".NCRES	Trigger NC-RESET
Unit:	INT	0	-
WVar1:	WORD	0	-

14.17.4.11 PI service: SELECT

Function: Select processing for a channel

A program stored on the NC is selected for one channel for execution. The program must be executable. The path and program name must be stated in full.

For detailed information, please refer to:

References

Programming Manual, Job Planning; Section: "File and Program Management" > "Program Memory".

Possible block types

Block types	
Workpiece directory	WPD
Main program	MPF
Subprogram	SPF
Cycles	CYC
Asynchronous subprograms	ASP
Binary files	BIN

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".SELECT	Program selection
Unit:	INT	1, 2, 3 ... 10	Channel
Addr1:	STRING	-	Path name
Addr2:	STRING	-	Program name

14.17.4.12 PI service: SETUDT

Function: Set current user data active

The current user data, such as tool offsets, basic frames and settable frames are set to active in the next NC block (only in stop state).

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".SETUDT	Activate user data
Unit:	INT	1, 2, 3 ... 10	Channel
WVar1:	WORD	1, 2, 3, 4, 5	User Data Type 1: Active tool offset 2: Active basic frame 3: Active settable frame 4: Active global basic frame 5: Active global settable frame
WVar2:	WORD	0	Reserved
Wvar3:	WORD	0	Reserved

14.17.4.13 PI service: SETUFR

Function: Activate user frames

User frames are loaded to the NC. All necessary frame values must be transferred to the NC first with FB3 "Write variables".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".SETUFR	Activate user frames
Unit:	INT	1, 2, 3 ... 10	Channel

14.17.4.14 PI service: RETRAC

Function: Select JOG retract

Selects the JOG retract mode. The retraction axis, i.e. the geometry axis, with which the retraction is executed can be determined by the NC automatically or specified explicitly.

The mode remains active until it is ended with RESET.

Note

The RETRAC PI service can only be activated in JOG mode in the "Reset" state.

Automatic determination

For automatic determination, the geometric axis is selected as a retraction axis, which is perpendicular (orthogonal) to the currently selected working plane:

- G17: Retraction axis ⇒ 3rd geometry axis (Z)
- G18: Retraction axis ⇒ 2nd geometry axis (Y)
- G19: Retraction axis ⇒ 1st Geometry axis (X)

Note

OPI variable retractState

The active retraction axis can be read via the OPI variable retractState.Bit 2/3.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".RETRAC	Select JOG retract mode
Unit:	INT	1, 2, 3 ... 10	Channel
WVar1:	WORD	0, 1, 2, 3	Retraction axis 0: Automatic determination of the retraction axis by the NC. 1: Retraction axis is the 1st geometry axis of the WCS 2: Retraction axis is the 2nd geometry axis of the WCS 3: Retraction axis is the 3rd geometry axis of the WCS
WVar2:	WORD	0	Reserved. The value must be pre-assigned with 0.

14.17.4.15 PI service: CRCEDN

Function: Creates new cutting edge

If the T number of an existing tool is entered in parameter "T Number" the PI service, then a tool edge for the existing tool is created. In this case, the parameter "D number", the number of the edge to be created, has a value range of from 1 to 9.

If a positive T number is specified as a parameter and the tool for the T number entered does not exist, the PI service is aborted.

If a value of 0 is entered as the T number (model of absolute D numbers), the D number values can range from 1 to 31999. The new cutting edge is set up with the specified D number.

If the specified cutting edge already exists, the PI service is aborted in both cases.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".CRCEDN	Create new cutting edge
Unit:	INT	1 … 10	TOA
WVar1:	INT		T number of tool for which cutting edge must be created. A setting of 00000 states that the cutting edge should not refer to any particular tool (absolute D number).

Signal	Type	Value range	Description
WVar2:	INT	1 ... 9	Edge number of tool cutting edge
		1 ... 31999	

14.17.4.16 PI service: CREACE

Function: Create tool cutting edge

Creation of the cutting edge with the next higher / next unassigned D number for the tool with the transferred T number in TO, TS (if present). The cutting edge for the OEM cutting edge data is set up simultaneously in the TUE block (if one is present).

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".CREACE	Create tool cutting edge
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT		T number

14.17.4.17 PI service: CREATO

Function: Create tool

Create 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.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".CREATO	Create tool
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT		T number

14.17.4.18 PI service: DELECE

Function: Delete a tool cutting edge

If the T number of an existing tool is entered in parameter "T Number" the PI service, then a tool edge for the existing tool is deleted. In this case, the parameter "D number", the number of the edge to be created, has a value range of from 00001 to 00009. If a positive T number is specified as a parameter and the tool for the T number entered does not exist, then the PI service is aborted. If a value of 00000 is entered as the T number (model of absolute D numbers), then the D number values can range from 00001 - 31999. If the specified cutting edge does not exist, then the PI service is aborted in both cases.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".DELECE	Delete cutting edge
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT		T number of tool for which cutting edge must be created. A setting of 00000 states that the cutting edge should not refer to any particular tool (absolute D number).
WVar2:	INT	1 - 9	D number of cutting edge that must be deleted
		1 - 31999	

14.17.4.19 PI service: DELETO

Function: Delete tool

Deletes the tool assigned to the transferred T number with all cutting edges (in TO, in some cases TU, TUE and TG (type 4xx), TD and TS blocks).

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".DELETO	Delete tool
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT		T number

14.17.4.20 PI service: MMCSEM

Function

The PI service is used for testing, setting and resetting of channel-specific semaphores from HMI through the PLC. 10 semaphores are available per channel for protecting critical data areas.

Functions (PI services) are assigned to semaphores 1 to 6. The semaphores 7 to 10 can be freely used.

FB4 return values

- "Done" == TRUE
  The semaphore was set, the critical function can be called.
- "Error" == TRUE AND "State" == 3
  The semaphore was already set, currently the critical function cannot be called.

Schematic sequence for free semaphore

```
Testing and setting the semaphore
IF semaphore == FREE
THEN
   Writing/reading of critical data
   Resetting the semaphore
```

```
ELSE // Semaphore is blocked
...
ENDIF
```

NOTICE

Resetting the semaphore

After blocking the critical data area by setting the semaphore and subsequent reading or writing of the data, the critical data area must be enabled once again by resetting the semaphore, otherwise subsequent blocking will not be possible.

Schematic sequence for blocked semaphore

```
Testing and setting the semaphore
IF semaphore == FREE
THEN
   ...
ELSE // Semaphore is blocked
   Set bit memory for "Function could not be executed, repeat
necessary"
ENDIF
```

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".MMCSEM	Edit semaphore
Unit:	INT	1, 2, 3 ... 10	Channel
WVar1:	INT	1)	PI service-specific number of the semaphore
WVar2:	WORD	0, 1	Job type 0: Reset semaphore 1: Test and set semaphore
1)	**Number**	**PI service**	
	1	TMCRTO (create tool)	
	2	TMFDPL (search for empty location for loading)	
	3	TMMVTL (prepare magazine location for loading, unload tool)	
	4	TMFPBP (search for location)	
	5	TMGETT (search for tool number)	
	6	TSEARC (search for tool)	
	7 ... 10	Freely usable	

14.17.4.21 PI service: TMCRTO

Function: Create tool

Creation of a tool by specifying a name, a duplo number, e.g. with $TC_TP1[y] = Duplo number or $TC_TP2[y] = "<Tool name>". Or optionally using a T number, e.g. with y = <T number>.

The tool is entered in the TV area (tool directory) as available.

The first cutting edge D1 (with zero content) is created in the TO block for corrections.

The first cutting edge D1 (with zero content) is created in the TS block for monitoring data.

In the TUE block, if it is available, the cutting edge D1 is created for the OEM cutting edge data.

The TD block contains identifiers, duplo numbers, and the number of cutting edges (=1) for the T number that is optionally specified or assigned by the NC.

If a TU block exists, it will contain the data block for the tool.

After execution of the PI service, the T number of the tool created is available in the TV block under **TnumWZV**.

Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 1 for TMCRTO. See Section "PI service: MMCSEM (Page 1001)".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMCRTO	Create tool
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT	>0 → the T number is specified -1 → the NC assigns the T number	T number
WVar2:	INT		Duplo number
Addr1:	STRING	Max. 32 characters	Tool name

14.17.4.22 PI service: TMFDPL

Function

Search for empty location for loading, depending on the parameter assignment:

- LocationNumber_To = -1 AND MagazineNumber_To = -1
  Searches all magazines in the specified area (= channel) for an empty location for the tool specified with a T number. After execution of the PI service, the magazine and locations numbers found during the search are listed in the configuration block of the channel (component **magCMCmdPar1** (magazine number) and **magCMCmdPar2** (location number)). LocationNumber_ID and MagazineNumber_ID can be set as search criteria or not (= -1). The PI service is positively or negatively acknowledged depending on the search result.
- LocationNumber_To = -1 AND MagazineNumber_To = magazine number
  An empty location for the tool specified with a T number is searched for in the specified magazine. Location number (reference) and MagazineNumber_Ref can be allocated as search criteria or with -1. The PI service is positively or negatively acknowledged depending on the search result.
- LocationNumber_To = location number AND MagazineNumber_To = magazine number
  The specified location is checked to ensure that it is free for loading of the tool. LocationNumber_ID and MagazineNumber_ID can be set as search criteria or -1. The PI service is acknowledged positively or negatively depending on the search result.

The parameters "WVar1" and "WVar2" are located at the source.

Loading:	If the source is an internal loading magazine, then the parameters are located at the target (a real magazine).
Unloading:	Source is always a real magazine.

Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 2 for TMFDPL.
See Chapter "PI service: MMCSEM (Page 1001)".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMFDPL	Empty location for loading
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT		T number
WVar2:	INT		LocationNumber_To (target)
WVar3:	INT		MagazineNumber_To (target)
WVar4:	INT		LocationNumber_Ref (reference)
WVar5:	INT		MagazineNumber_Ref (reference)

14.17.4.23 PI service: TMFPBP

Function: Empty location search

The PI service searches the specified magazine(s) for an empty location which meets the specified criteria such as tool size and location type.

If the search is successful, the result can be read from the following OPI variables:

- magCMCmdPar1 (magazine number)
- magCMCmdPar2 (location number)

Note

The PI service can only be requested using FB7. See Chapter "FB7: PI_SERV2 - request PI service (Page 1025)".

Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 4 for TMFPBP. See Chapter "PI service: MMCSEM (Page 1001)".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMFPBP	Empty location search
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT		**Magazine number** of the magazine **from** which the search is to be performed
WVar2:	INT		**Location number** of the location in the magazine from "WVar1"
WVar3:	INT		**Magazine number** of the magazine up to which the search is to be performed
WVar4:	INT		**Location number** of the location in the magazine from "WVar3"
WVar5:	INT		Magazine number reference
WVar6:	INT		Location number reference
WVar7:	INT	0, 1, 2 ... 7	Number of required half locations to left
WVar8:	INT	0, 1, 2 ... 7	Number of required half locations to right
WVar9:	INT	0, 1, 2 ... 7	Number of required half locations in upward direction
WVar10:	INT	0, 1, 2 ... 7	Number of required half locations in downward direction
WVar11:	INT		Number of required location type
WVar12:	INT	0, 1, 2, 3, 4	Specifies the required search direction 0: Search strategy as set in $TC_MAMP2 1: Forward 2: Backward 3: Symmetrical

Examples: Setting the search range

From	Location	To	Location	Description
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 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

14.17.4.24 PI service: TMGETT

Function: Determine T number for the specified tool name with duplo number

The PI service is used to determine the T number of a tool via the tool name and duplo number.

The result is written in BTSS variables in the TF block (parameterization, return parameters from TMGETT, TSEARC):

- resultNrOfTools
  - resultNrOfTools == 0: The specified tool was not found
  - resultNrOfTools == 1: The specified tool was found
- resultToolNr: T number of the specified tool with "resultNrOfTools" == 1

Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 5 for TMGETT. See Section "PI service: MMCSEM (Page 1001)".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMGETT	Determining the T number
Unit:	INT	1, 2, 3 ... 10	TOA

Signal	Type	Value range	Description
Addr1:	STRING	Max. 32 characters	Tool name
WVar1:	INT		Duplo number

14.17.4.25 PI service: TMMVTL

Function: Prepare magazine location for loading, unload tool

The PI service is used to load, unload, and relocate tools:

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

Case 1, 3, 4 and 5: The following BTSS variables of the block TM (magazine data: general data) are written:

- magCmd (area no. = TO unit, line = magazine number)
- magCmdState ← "acknowledgment"

Case 2 and 6: The following BTSS variables of the block TMC (magazine data: configuration data) are written:

- magCBCmd (area no. = TO unit)
- magCBCmdState ← "acknowledgment"

Loading

- "WVar2" LocationNumber_From, "WVar3" MagazineNumber_From
  The tool location of the magazine is moved to the loading station/location for loading and the tool is loaded.
- "WVar4" LocationNumber_To== -1
  First, an empty location for the tool is searched for in the magazine. Then the empty location of the magazine is moved to the loading station/location for loading and the tool is loaded. After the PI service is carried out, the empty location number that is found is located in "magCMCmdPar2" (BTSS block TM), **real** magazine of the channel.
- "WVar4" LocationNumber_To == -2
  The tool is loaded into the current tool location of the magazine. After the PI service is carried out, the location number is located in "magCMCmdPar2" (BTSS block TM), **real** magazine of the channel.

Unloading

- "WVar4" LocationNumber_To, "WVar5" MagazineNumber_To
- The tool location of the magazine is moved to the loading station for loading and the tool is unloaded.

In the BTSS block TP (magazine data: location data), the magazine location of the removed tool is designated as free.

Addressing the tool

The tool can be addressed either using a T number or by means of the location and magazine numbers. The value -1 is to be assigned to unused parameters.

Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 3 for TMMVTL. See Chapter "PI service: MMCSEM (Page 1001)".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMMVTL	Make magazine location ready for loading, unloading tool
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT		T number
WVar2:	INT		LocationNumber_From (source)
WVar3:	INT		MagazineNumber_From (source)
WVar4:	INT		LocationNumber_To (target)
WVar5:	INT		MagazineNumber_To (target)

14.17.4.26 PI service: TMPOSM

Function: Position magazine location or tool, depending on the parameter assignment

This PI service can traverse a magazine to position a magazine location at a specified position (e.g. at a loading position). The magazine location can be specified directly or via a tool at the location.

The destination, e.g. the loading location, is specified in the parameters:

- "WVar5" LocationNumber_Ref
- "WVar6" MagazineNumber_Ref

The magazine location to be positioned is specified in the following parameters, depending on the respective specification:

- "WVar1" T number of the tool
  The following parameters are irrelevant here:
  - "Addr1" Tool name = ""
  - "WVar2" duplo number = -1
  - "WVar3" LocationNumber_From = -1
  - "WVar4" MagazineNumber_From = -1
- "Addr1" tool name, "WVar2" duplo number
  The following parameters are irrelevant here:
  - "WVar1" T number of the tool = -1
  - "WVar3" LocationNumber_From = -1
  - "WVar4" MagazineNumber_From = -1
- "WVar3" LocationNumber_From, "WVar4" MagazineNumber_From
  The following parameters are irrelevant here:
  - "Addr1" Tool name = ""
  - "WVar1" T number of the tool = -1
  - "WVar2" duplo number = -1

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMPOSM	Position magazine location or tool
Unit:	INT	1, 2, 3 ... 10	TOA
Addr1:	STRING	Max. 32 characters	Tool name
WVar1:	INT		T number of the tool
WVar2:	INT		Duplo number
WVar3:	INT		LocationNumber_From (source)
WVar4:	INT		MagazineNumber_From (source)
WVar5:	INT		LocationNumber_Ref
WVar6:	INT		MagazineNumber_Ref

14.17.4.27 PI service: TMPCIT

Function: Set increment value for workpiece counter

Incrementing the workpiece counter of the spindle tool

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMPCIT	Set increment value for workpiece counter
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	WORD	0, 1, 2, ... max.	Spindle number; corresponds to the type index in the location data with "Spindle" location type of the buffer magazine in channel. 0: Main spindle
WVar2:	WORD	0 ... max.	Increment value; indicates the number of spindle revolutions after which the workpiece counter is incremented

14.17.4.28 PI service: TMRASS

Function: Resetting the tool status "active"

The PI service sets the status to "inactive" for all of the tools with a tool status "active" or "blocked".

The following events are sensible times for resetting the tool status:

- a negative edge of the NC/PLC interface signal "tool disable ineffective"
- End of program
- Channel reset.

The PI service is intended for the PLC, since it is known here when the disabled tool is finally no longer to be used.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI". TMRASS	Reset active status
Unit:	INT	1, 2, 3 ... 10	TO area

14.17.4.29 PI service: TRESMO

Function: Reset monitoring values

This PI service resets the monitoring values of the designated edges of the designated tools to their setpoints (initial values).
This is only performed for tools with active monitoring.

See also the RESETMON command.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TRESMO	Reset monitoring values
Unit:	INT	1, 2, 3 ... 10	TO area
WVar1:	WORD	- max ... max	Tool number 0: Process all tools > 0:process specified tool < 0: Process all sister tools
WVar2:	WORD	0, 1, 2, ... max.	D number < 0: Reset of the monitoring of the specified cutting edge 0: Reset of monitoring of all cutting edges
Wvar3:	WORD		Monitoring mode that is to be reset (all combinations are possible):
			Bit 0 = 1: Tool-life monitoring
			Bit 1 = 1: Quantity monitoring
			Bit 2 = 1: Wear monitoring
			Bit 3 = 1: Sum-offset monitoring
			0: Reset all active tool-monitoring functions ($TC_TP9).

14.17.4.30 PI service: TSEARC

Function: Complex search using search screen form, depending on the parameter assignment

The PI service is used to search for tools with specific properties within a search range in one or more magazines, beginning with a specific location, up to a specific location.

Note

Active tool management

The PI service is only available if tool management is activated.

Specification options

- Search direction
- Search for next tool with the specified property
- Search for all tools with the specified property

Result

As a result, a list with the internal T-numbers of the tools that are found is created.

Logic operations

For filtering properties, only one AND link is available as a linking option. An OR link must be achieved by the user via several calls of the PI service and subsequent evaluation of the individual results.

Parameterization of the tool properties

The properties of the searched for tools are set in the BTSS block TF (parameterization, return parameters from TMGETT, TSEARC) via the following variables:

- "parMask<X>" parameterization mask
- "parData<X>" comparison values

With <X> = TAD, TAO, TAS, TD

Result list

After completion of the PI service without errors, the search result is located in the BTSS block TF (parameterization, return parameter from TMGETT, TSEARC) in the following variables:

- "resultCuttingEdgeNrUsed" D numbers of cutting edges used since last quantity count
- "resultNrOfCutEdgesUsed" Number of cutting edges since last quantity count
- "resultNrOfTools" Number of found tools
- "resultToolNr" T numbers of found tools
- "resultToolNrUsed" T numbers of cutting edges used since last quantity count

If no tool was found, the number of found tools is zero ("resultNrOfTools" == 0).

Search range specifications

From mag. number `WVar1`	From location number `WVar2`	To mag. number `WVar3`	To location number `WVar4`	Description The following magazine locations are searched:
#M<a>	#L<b>	#M<x>	#L<y>	From: Magazine #M<a>, location #L<b> to: Magazine #M<x>, location #L<y>
#M<a>	-1	#M<a>	-1	From: Magazine #M<a>, first location to: Magazine #M<a>, last location
#M<a>	-1	-1	-1	From: Magazine #M<a>, first location to: Last magazine, last location
#M<a>	#L<b>	-1	-1	From: Magazine #M<a>, location #L<b> to: Last magazine, last location
#M<a>	#L<b>	#M<a>	-1	From: Magazine #M<a>, location #L<b> to: Magazine #M<a>, last location
#M<a>	#L<b>	#M<x>	-1	From: Magazine #M<a>, location #L<b> to: Magazine #M<b>, last location
#M<a>	-1	#M<x>	#L<y>	From: Magazine #M<a>, first location to: Magazine #M<b>, location #L<y>
#M<a>	-1	#M<x>	-1	From: Magazine #M<a>, first location to: Magazine #M<b>, last location
-1	-1	-1	-1	From: First magazine, first location to: Last magazine, last location

Symmetrical search

A symmetrical search, relative to a magazine location, can only be performed if the following conditions have been met:

- The search range must encompass only one magazine: "WVar1" (from: Magazine number) == "WVar3" (to: magazine number)
- Specification of a reference location, i.e. a magazine location for which a symmetrical search is to be performed: "WVar5" (number of the reference magazine) and "WVar6" (number of the reference location)
- For the reference location, a multiple assignment to the magazine to be searched must have been configured in the TPM block.
- "WVar7" (search direction) = 3

The reference location is a buffer location, i.e. a tool location from the buffer magazine or from the internal loading magazine, e.g. swapping station, gripper, loading station. The symmetrical search is made in relation to the magazine location in front of the reference location.

If the magazine location is upstream of the reference location outside of the search range, the PI service will respond as if no suitable location had been found.

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 Chapter "PI service: MMCSEM (Page 1001)".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TSEARC	Complex search using search screen forms
Unit:	INT	1, 2, 3 ... 10	TOA
WVar1:	INT	-1, 1, …	From: Magazine number
WVar2:	INT	-1, 1, …	From: Location number
WVar3:	INT	-1, 1, …	to: Magazine number
WVar4:	INT	-1, 1, …	to: Location number
WVar5:	INT	-1, 1, …	Number of the reference magazine (only relevant for symmetrical search: Search direction ==3)
WVar6:	INT	-1, 1, …	Number of the reference location (only relevant for symmetrical search: Search direction ==3)

Signal	Type	Value range	Description
WVar7:	INT	1, 2, 3	Search direction: 1: Forwards from the first location of the search domain 2: Backwards from the last location of the search domain 3: Symmetrical to the real magazine location, which is located **upstream** of the location specified under "WVar5" (number of the ref-erence magazine) and "WVar6" (number of the reference location)
WVar8:	INT	0, 1, 2, 3	Properties of the tools: 0: all tools, cutting edge-specific 1: first tool, cutting edge-specific 2: all tools, via all cutting edges 3: first tool, via all cutting edges

14.17.4.31 PI service: TMCRMT

Function: Create multitool

The PI service is used to create a new multitool with a defined identifier, an optionally specifiable multitool number, the number of tool locations, and the type of distance coding.

Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" =1 (TMCRTO) must be called. See Chapter: "PI service: MMCSEM (Page 1001)".

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMCRMT	Create multitool
Unit:	INT	1, 2, 3 ... 10	TOA
Addr1:	STRING		Multitool designator (max. 32 characters)
WVar1:	INT	0	Reserved
WVar2:	INT	-1, 1, 2, ... 32000	Multitool number -1: Automatic assignment of the multitool number by NC 1, 2, 3 ... 32000: Multitool number, **note**: The multitool number must be unique
WVar3:	INT	2, 3, 4, ... MAX	Number of tool locations MAX = Parameterized number in MD17504 $MN_MAX_TOOLS_PER_MULTITOOL
WVar4:	INT	1, 2, 3	Type of distance coding

14.17.4.32 PI service: TMDLMT

Function: Delete multitool

The PI service is used to delete a multitool in all of the data blocks in which it is saved. Tools equipped in multitool are not subsequently equipped or loaded, however, continue to be defined if they are not also to be deleted.

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".TMDLMT	Delete multitool
Unit:	INT	1, 2, 3 ... 10	TOA
Addr1:	STRING		Multitool designator (max. 32 characters)
WVar1:	INT	0	Reserved, always 0
WVar2:	INT	-1, 1, 2, ... 32000	Multitool number -1: Delete the multitool with the name specified in "Addr1" 1 ,2, 3 ... 32000: Delete the multitool with the specified multitool number; **note:** Parameter "Addr1" is not evaluated
WVar3:	INT	0, 1	Tools contained in the multitool: 0: Do not delete 1: Delete

14.17.4.33 PI service: POSMT

Function: Position multitool

The PI service is used for positioning a multitool at the programmed location or alternatively at the programmed tool, which is located in one of the locations of the multitool. The tool itself can either be specified using its T number or with its name and duplo number. A multitool can only be positioned if it is at a tool carrier (e.g. spindle) and if no tool offset with regard to this tool carrier is active.

Position specification

Position specification can be specified as one of three variants:

No.	Addr1	WVar1	WVar2	WVar3	WVar4
1	Empty string	Number of the toolholder	Tool number	-1	
2	Tool name	Number of the toolholder	-1	Duplo number	
3	Empty string	Number of the toolholder	-1	-1	Multitool location number

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".POSMT	Position multitool
Unit:	INT	1, 2, 3 ... 10	TOA
Addr1:	STRING		Tool name of the tool to be positioned in the multitool (max. 32 characters)
WVar1:	INT	1, 2, 3, ... 999	Number of the toolholder
WVar2:	INT	-1, 1, 2, 3 ... 32000	Tool number (T number) of the tool to be positioned in the multitool 1, 2, 3, ... 32000: Tool number -1: "Addr1" (tool name) and "WVar3" (duplo number) are used
WVar3:	INT	-1, 1, 2, 3 ... 32000	Duplo number of the tool to be positioned in the multitool 1, 2, 3, ... 32000: Duplo number -1: "WVar2" (T number) is used
WVar4:	INT	1, 2, 3 ... 999	Multitool location number of the toolholder location to which the system should be positioned

14.17.4.34 PI service: FDPLMT

Function: Searching for or checking an empty location within the multitool

The PI service is used to search for a free tool location in a multitool to accommodate the specified tool or for checking whether the specified tool location in the multitool for accommodating the specified tool is free. The tool can be specified using the T number or the identifier and the duplo number.

Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" =x (FDPLMT) must be called. See Chapter "PI service: MMCSEM (Page 1001)".

Position specification

The tool to be positioned in the multitool can be specified as one of three variants:

- "Addr1" = <Empty string>, "WVar1" = <Tool number> and "WVar2" = <Duplo number>
- "Addr1" = <Tool name>, "WVar1" = -1 and "WVar2" = <Duplo number>
- "Addr1" = <Empty string>, "WVar1" = <Tool number> and "WVar2" = -1

The position specification can be assigned via:

- "WVar4" = <Multitool location number>

Description of formal parameters

Signal	Type	Value range	Description
PIService:	ANY	"PI".FDPLMT	Search/check an empty tool location within a multitool
Unit:	INT	1, 2, 3 ... 10	TOA
Addr1:	STRING		Tool name of the tool to be positioned in the multitool (max. 32 characters)
WVar1:	INT	-1, 1, 2, 3, ... 32000	Tool number (T number) of the tool to be positioned in the multitool -1: "Addr1" (tool name) and "WVar2" (duplo number) are used. 1, 2, 3 ... 32000: Tool number
WVar2:	INT	-1, 1, 2, 3, ... 32000	Duplo number of the tool to be positioned in the multitool 1, 2, 3, ... 32000: Duplo number -1: "WVar1" (T number) is used
WVar3:	INT	-1, 1, 2, 3, ... 32000	Number of the multitool 1, 2, 3, ... 32000: Number of the multitool -1: Search of all multitools for an empty location or check of all multitools to see whether the specified tool location is free in one of them for accommodating the tool.
WVar4:	INT	-1, 1, 2, 3, ... 999	Multitool location number of the tool location to which the system should be positioned >0: Multitool location number -1: Search for any empty tool location within the multitool

14.17.5 FB5: GETGUD - read GUD variable

Function

The function block FB5 "GETGUD" is used for reading global user data (GUD) in the NC or channel area.

Request for reading NC variables

Call of FB5 with positive edge change, parameter "Req" = 0 → 1

Parameter "Addr": The pointer to the name of the GUD variables, symbolically with "<Data block>".<Variable name>

Parameters "Area", "Unit", "Index1", and "Index2": Additional information for addressing the variables

When the parameter "CnvtToken" is activated, the user receives a token (variable pointer) for the GUD variable to be read. Using this, the GUD variables can then be read or written via

FB2 and FB3 with parameter "Addr1" ... "Addr8" = "<Token>". Addressing by means of a token is mandatory for **writing** GUD variables. If the token is used to address GUD variable arrays, the parameter "Line1" ... "Line8" = "<array index>" of the FB2/FB3 must also be provided with values.

To read a GUD variable of the DOUBLE type without adapting the format, an ANY pointer of the REAL2 type must be specified in the target area. E.g. P#M100.0 REAL 2. The value of the GUD variables of the DOUBLE type is then adopted in the PLC data area as a 64-bit floating point number.

Completion of the read request

Read request successfully completed: Parameter "Done" == 1.

Read request completed with error:"Parameter "Error" == 1, error cause in parameter "State"

Prerequisites

- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block that will contain the string with the name of the GUD variable, the S7 Symbol Editor must be used to assign a symbol in the S7 symbol table of the S7 project (cf. call example 1 below: DB_GUDVAR). In the data block, a string of suitable length must be created to contain the name of the GUD variable (cf. call example 1 below: "DB_GUDVAR".GUDVar1). This symbol is then passed on to parameter "Addr" of FB5, e.g. Addr := "DB_GUDVAR".GUDVar1. If the token for the GUD variable is required for the following calls of the FB2/FB3, the 10-byte token structure must be created in a DB, e.g. the same DB (cf. call example 1 below: "DB_GUDVAR".GUDVar1Token). This structure is specified in parameter VarToken of the FB5, cf. call example 1 below: FB5, VarToken := "DB_GUDVAR".GUDVar1Token. This token is then specified in parameters "Addr1" … "Addr8" when FB2/FB3 is called, cf. call example 1 below: FB3, Addr1 := "GUDVar1Token".

General conditions

- FB5 has multi-instance capability.
- Every call of FB5 must be assigned a separate instance DB from the user area.
- Reading of a GUD variable (FB5 call with Req = 1) is only permitted in the cyclic part of the PLC basic program (OB1). If the job is not started (FB5 call with Req = 0), the parameters can also be written in the start-up part of the PLC basic program (OB100). The job can then be executed using the already written parameters in the cyclic part of the PLC basic program (OB1) by calling FB5 with Req = 1.
- Only capital letters may be used for the names of GUD variables.
- Reading of a GUD variable generally extends over several PLC cycles.

Note

Error case
When variables from different channels are read, the following feedback message is output:
- "Error" == TRUE
- "State" == W#16#02

Note

To read a variable of the DOUBLE type from NC without adapting the format, an ANY pointer of the REAL2 type must be specified in the target area (e.g.: P#M100.0 REAL 2). If the basic program recognizes REAL 2 as the target type when reading a variable of the DOUBLE type, the data is applied to the PLC data area as a 64-bit floating-point number.

Declaration of the function

```
FUNCTION_BLOCK FB5          //Server name
   KNOW_HOW_PROTECT
   VERSION : 3.0
VAR_INPUT
   Req :         BOOL;
   Addr:         ANY;       // Variable name
   Area          BYTE ;     //Area: NCK = 0, channel = 2
   Unit :        BYTE ;
   Index1:       INT;       // Array index 1
   Index2:       INT;       // Array 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

Signal	Typ e	Type	Value range	Description
Req:	I	BOOL		Job start with positive signal edge
Addr:	I	ANY	"<DBName>".<Var-Name>	Variable name in a variable of the type STRING

Signal	Type	Type	Value range	Description
Area:	I	BYTE	0, 2	Range: 0: NC 2: Channel
Unit:	I	BYTE	1, 2, 3 ... 10	Area == NC: Unit:=1 Area == Channel: Channel number
Index1:	I	INT	0, 1, 2, ... < max. Array index>	Array index 1 For variables without array index: Index1 = 0
Index2:	I	INT	0, 1, 2, ... < max. Array index>	Array index 2 For variables without 2nd array index: Index2 = 0
CnvtToken:	I	BOOL	0 (FALSE), 1 (TRUE)	Activate generation of a 10 byte variable token
VarToken:	I	ANY	"<DBName>".<VarName>	Address of a 10byte token (see example)
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	Negative acknowledgement of job or execution of job impossible
Done:	O	BOOL	0 (FALSE), 1 (TRUE)	Job successfully executed
State:	O	WORD	---	See paragraph "Error identifiers"
RD:	I/O	ANY	P#Mm.n BYTE x ... P#DBnr.dbxm.n BYTE x	Target area for read data

Error identifiers

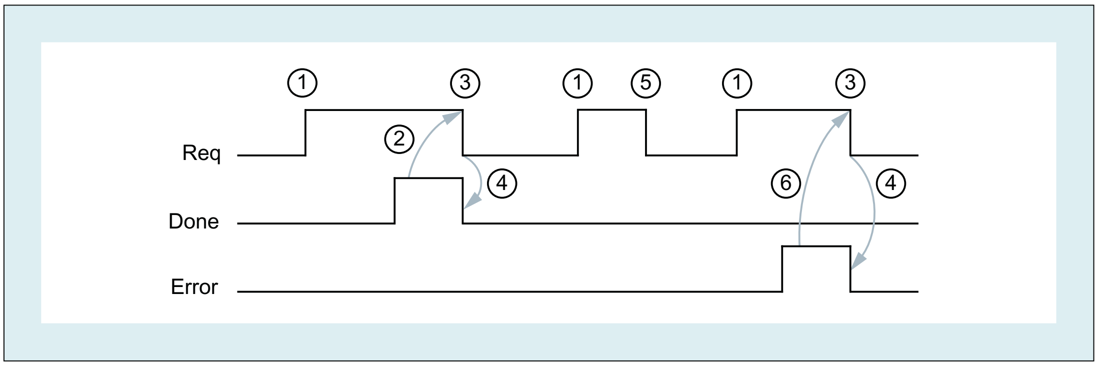
State		Description	Note
WORD H 1)	WORD L		
0	1	Access error	
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgement, job not executable	Internal error, try: NC RESET
0	4	Data areas or data types do not tally	Check data to be read in RD
1	4	Insufficient local user memory available	read variable is longer than specified in RD
0	6	FIFO full	Job must be repeated, since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
0	8	Incorrect target area (SD)	RD may not be local data
0	9	Transmission occupied	Job must be repeated
0	10	Error in addressing	Unit contains value 0
0	11	Address of variable invalid	Check addr (or variable name), area, unit
1 - 8	13	ANY data reference incorrect	String/NcVar data required has not been parameterized

State		Description	Note
WORD H 1)	WORD L		
0	15	String more than 32 characters	GUD variable name too long
1) High byte > 0 ⇒ Number of the variable in which the error occurred			

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	User: Set request, Req = 0 → 1
②	FB5 successfully completed, Done = 1 User: Reset request, IF Done == 1 THEN Req = 0
③	User: IF Done == 1 THEN reset request: 1 → 0
④	FB5 reset job confirmation, Done = 0
⑤	User: IF Done == 0 AND Error == 0 THEN reset request Req = 1 → 0 **not permissible**
⑥	FB5 completed with errors, Error = 1 User: Reset request, IF Done == 1 OR Error == 1 THEN Req = 0, possible further error handling

Call example 1

Read a GUD variable from channel 1:

- Name "GUDVAR1"
- Type: INTEGER

Conversion to a 10 byte variable pointer. See table "Assignment of the data types" in Chapter "FB2: GET - read NC variable (Page 973)"

Reading the GUD variables: FB5 with instance DB111

```
// Data block for GUD variable
DATA_BLOCK DB_GUDVAR   // Assignment in symbol table

STRUCT
   GUDVar1 : STRING[32] := 'GUDVAR1';                // Name is defined by
                                                     user
   GUDVar1Token :
   STRUCT
      SYNTAX_ID : BYTE ;
      area_and_unit : BYTE ;
      column : WORD;
      line : WORD;
      block type : BYTE ;
      NO. OF LINES : BYTE ;
      type : BYTE ;
      length : BYTE ;
   END_STRUCT;
END_STRUCT;

BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB5
DATA_BLOCK DB111

        FB5
BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB3
DATE_BLOCK DB112

        FB3
BEGIN
END_DATA_BLOCK

// Reading of a channel-specific GUD variable from channel 1, with conversion
to a variable pointer
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 FB5, DB111(
           Req          := M 100.0,             // Starting edge for reading
           Addr         := "DB_GUDVAR".GUDVar1, // Name of the GUD
           Area         := B#16#2,              // Channel variable
           Unit         := B#16#1,              // Channel 1
           Index1       := 0,                   // No array index 1
           Index2       := 0,                   // No array index 2
           CnvtToken    := TRUE,                // Request: Conversion into
                                                10-byte token
           VarToken     := "DB_GUDVAR".GUDVar1Token, // Address of the token
           Error        := M 102.0,
           Done         := M 100.1,
           State        := MW 104
           RD           := P#DB99.DBX0.0 DINT 1 // free memory area
);
```

Writing the GUD variables: FB3 with instance DB112

GUD variable token from FB5, parameter: "VarToken" for writing with FB3, parameter "Addr1"

```
CALL FB3, DB112(
           Req          := M 200.0,
           NumVar       := 1,                     // one GUD variable
           Addr1        :=                        // Token
                        "DB_GUDVAR".GUDVar1Token,
           Error        := M 102.0,
           Done         := M 100.1,
           State        := MW 104
           SD1          := P#DB99.DBX0.0 DINT 1
);
```

Call example 2

Read a GUD variable from channel 1:

- Name "GUD_STRING"
- Type: STRING[32]

Conversion to a 10 byte variable pointer.

Reading the GUD variables: FB5 with instance DB111

```
// Data block for GUD variable
DATA_BLOCK DB_GUDVAR    // Assignment in symbol table

STRUCT
    GUDVarS : STRING[32] := 'GUD_STRING';    // Name defined by user
    GUDVarSToken :
    STRUCT
      SYNTAX_ID : BYTE ;
      area_and_unit : BYTE ;
      column : WORD;
      line : WORD;
      block type : BYTE ;
      NO. OF LINES : BYTE ;
      type : BYTE ;
      length : BYTE ;
    END_STRUCT;

    string_of_GUD : STRING[30];     // must at least be so long as
                                    // the definition of 'GUD_STRING'!
    new_name : STRING[30] := 'GUD_123';
END_STRUCT;

BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB5
DATA_BLOCK DB111

        FB5
BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB3
DATE_BLOCK DB112

        FB3
BEGIN
END_DATA_BLOCK

// Reading of a channel-specific GUD variable from channel 1, with conversion
to a variable pointer
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 FB5, DB111(
           Req          := M 100.0,     // Starting edge for reading
           Addr         := "DB_GUDVAR".GUDVarS,
           Area         := B#16#2,      // Channel variable
           Unit         := B#16#1,      // Channel 1
           Index1       := 0,           // No array index
           Index2       := 0,           // No array index
           CnvtToken    := TRUE,        // Request: Conversion into 10-byte
                                        token
           VarToken     := "DB_GUDVAR".GUDVarSToken, // Address of the token
           Error        := M 102.0,
           Done         := M 100.1,
           State        := MW 104
           RD           := "DB_GUDVAR".string_of_GUD
);
```

Writing the GUD variables: FB3 with instance DB112

GUD variable token from FB5, parameter: "VarToken" for writing with FB3, parameter "Addr1"

```
CALL FB3, DB112(
           Req          := M 200.0,
           NumVar       := 1,                // one GUD variable
           Addr1        := "DB_GUDVAR".GUDVarSToken, // Token
           Error        := M 102.0,
           Done         := M 100.1
           State        := MW 104
           SD1          := "DB_GUDVAR".new_name
);
```

Classification of data types

See table "Assignment of the data types" in Chapter "FB2: GET - read NC variable (Page 973)"

14.17.6 FB7: PI_SERV2 - request PI service

Function

With the exception of a higher number of "WVar" parameters ("WVar11" - "WVar16"), function block FB7 has the same functionality as function block FB4. It is recommended that function block FB7 be used instead of function block FB4.

For a detailed description, refer to Chapter "FB4: PI_SERV - request PI service (Page 988)".

Declaration of the function

```
FUNCTION_BLOCK FB7
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
```

14.17.7 FB9: MtoN - operator panel switchover

Function

The function block FB9 "MzuN" is used for switching over operating components (MCP/OP), which are connected via a bus system to one or more NCU control modules.

The interface between the individual operating units (operator panels) and the NCU (PLC) is the M : N interface in data block DB19. FB9 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 control unit switchover:

Tabulated overview of functions:	
Basic function	**Description**
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

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.

Brief description of a few important functions

Active/passive operating mode:

An online HMI can operate in two different modes:

Active mode:	Operator can control and monitor
Passive mode:	Operator can monitor (HMI header only)

After switchover to an NCU, this initially requests active operating mode in the PLC of the online NCU. If two control units are linked online simultaneously to an NCU, one of the two is always in active mode and the other in passive mode. The operator can request active mode on the passive HMI at the press of a button.

MCP switchover

As an option, an MCP assigned to the HMI can be switched over at the same time. To achieve this, the MCP address must be entered in the "mstt_adress" parameter of the NETNAMES.INI configuration file on the HMI and "MCPEnable" must be set to TRUE. The MCP of the passive HMI is deactivated so that there is only ever one active MCP on an NCU at one time.

Boot condition

In order to prevent, that for an NCU restart, the previously selected MCP is activated, when calling FB1 in OB100, input parameter "MCP1BusAdr" must be set = **255** (address 1st MCP) and "MCP1Stop" must be set **= TRUE** (switch off 1st MCP).

Approvals

When one MCP is switched over to another, any active feed or axis enables will be retained.

Note

Keys actuated at the moment of switchover remain operative until the new MCP is activated (by the HMI, which is subsequently activated). The override settings for feedrate and spindle also remain valid. To deactivate actuated keys, the input image of the machine control signals must be switched to nonactuated signal level on a falling edge of DB10.DBX104.0 (MCP 1 ready). The override settings should remain unchanged. Measures for deactivating keys must be implemented in the PLC user program (see example "Override Changeover").

The call is permitted only in cyclic program OB1.

Declaration of the function

```
FUNCTION_BLOCK FB9
VAR_INPUT
   Ackn:            BOOL;           // Acknowledge interrupts
   OPMixedMode:     BOOL:= FALSE;   // Mixed operation with non-M-to-N-
                                    enabled OP // deactivated
   ActivEnable:     BOOL:=TRUE;     // Not supported
   MCPEnable:       BOOL:=TRUE;     // Activate MCP switchover
END_VAR
VAR_OUTPUT
   Alarm1:          BOOL;           // Interrupt: Error in HMI bus address,
                                    bus type!
   Alarm2:          BOOL;           // Interrupt: No confirmation HMI 1
                                    offline!
   Alarm3 :         BOOL;           // Interrupt: HMI 1 is not going offline!
   Alarm4 :         BOOL;           // Interrupt: No confirmation HMI 2
                                    offline!
   Alarm5 :         BOOL;           // Interrupt: HMI 2 is not going offline!
   Alarm6 :         BOOL;           // Interrupt: Queuing HMI is not going
                                    online!
   Report :         BOOL;           // Message: Signoflife monitoring
   ErrorMMC :       BOOL;           // Error detection HMI
END_VAR
```

Description of formal parameters

Signal	Type	Type	Value range	Description
Ackn:	I	BOOL	0 (FALSE), 1 (TRUE)	Acknowledge interrupts
OPMixedMode:	I	BOOL	0 (FALSE), 1 (TRUE)	Mixed operation deactivated for OP without M:N capability

Signal	Type	Type	Value range	Description
ActivEnable:	I	BOOL	0 (FALSE), 1 (TRUE)	Function is not supported. Operator panel switchover Interlocking using MMCx_SHIFT_LOCK in DB19
MCPEnable:	I	BOOL	0 (FALSE), 1 (TRUE)	Activate MCP switchover 1: MCP is switched over with operator panel. 0: MCP is not switched over with operator panel. This can be used to permanently link an MCP. See also MMCx_MSTT_SHIFT_LOCK in DB19.
Alarm1:	O	BOOL	0 (FALSE), 1 (TRUE)	Interrupt: Error in HMI bus address, bus type!
Alarm2:	O	BOOL	0 (FALSE), 1 (TRUE)	Interrupt: No confirmation HMI 1 offline!
Alarm3 :	O	BOOL	0 (FALSE), 1 (TRUE)	Interrupt: HMI 1 is not going offline!
Alarm4 :	O	BOOL	0 (FALSE), 1 (TRUE)	Interrupt: No confirmation HMI 2 offline!
Alarm5 :	O	BOOL	0 (FALSE), 1 (TRUE)	Interrupt: HMI 2 is not going offline!
Alarm6 :	O	BOOL	0 (FALSE), 1 (TRUE)	Interrupt: Queuing HMI is not going online!
Report :	O	BOOL	0 (FALSE), 1 (TRUE)	Message: Sign-of-life monitoring HMI
ErrorMMC :	O	BOOL	0 (FALSE), 1 (TRUE)	Error detection HMI

Example of calling FB9

```
CALL FB9, DB109(
Ackn              := Error_ack,       //e.g., MCP RESET
OPMixedMode       := FALSE,
ActivEnable       := TRUE,
MCPEnable         := TRUE);           // Enable for MCP switchover
```

Note

Input parameter “MCPEnable” must be set to TRUE to enable the MCP switchover. The default value of these parameters is set in this way and need not be specially assigned when the function is called.

Interrupts, errors

The output parameters "Alarm1" to "Alarm6" and "Report" exist as information in the PLC and are output in the event of M:N errors visualized on the HMI by the appearance of alarms 410900 - 410906.

If execution of an HMI function has failed (and an appropriate error message cannot be displayed), status parameter "ErrorMMC" is set to 'logical 1' (e.g., booting error, when no connection is made).

Example of a call for FB (call in the 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 1st MCP
    MCP1Timeout           := S5T#700MS,
    MCP1Cycl              := S5T#200MS,
    MCP1Stop              := TRUE,           // MCP switched off
    NCCyclTimeout         := S5T#200MS,
    NCRunupTimeout        := S5T#50S);
```

Example: Override switchover

```
// Auxiliary flags used M100.0, M100.1, M100.2, M100.3
//Edge positive of MCP1Ready must check the override
//and measures for activation
// Initiate MCP block
//This example applies to the feedrate override;
//The interface and input bytes must be exchanged for spindle override.
U     DB10.DBX104.0;      //MCP1Ready
EN    M   100.0;          // Edge memory bit 1
JCN smth1;
S     M   100.2;          // Set auxiliary memory bit 1
R     M   100.3;          //Reset auxiliary flag 2


// Save override
     L DB21.DBB4;         //Feed override interface
     T EB28;              //Buffer storage (freely input
                          // or flag byte)

wei1:
U    M    100.2;          //Switchover takes place
O    DB10.DBX104.0;       //MCP1Ready
JCN smth2;
U    DB10.DBX104.0;       //MCP1Ready
FP   M    100.1;          // Edge memory bit 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.DBB4;        //Route override interface
     L EB3;              //Override input byte for feed
     <>i;                //Match?
JC smth2;                //No, jump
S    M   100.3;          //Yes, set auxiliary flag 2
// When override values match, call the MCP program again
MCP: CALL "MCP_IFM"(     //FC19
     BAGNo        := B#16#1,
     ChanNo       := B#16#1,
     SpindleIFNo  := B#16#0,
     FeedHold     := M 101.0,
     SpindleHold  := M 101.1);
wei2: NOP         0:
```

14.17.8 FB10: Safety relay (SI relay)

Function

The SPL function block FB10 "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" are deactivated via the parameterized timer values (parameters "TimeValue1", "TimeValue2", "TimeValue3"). The outputs are activated again without delay if inputs "In1" to "In3" take on value 1 and a positive edge change is detected at one of the acknowledgment inputs "Ack1", "Ack2".

Call bit memory

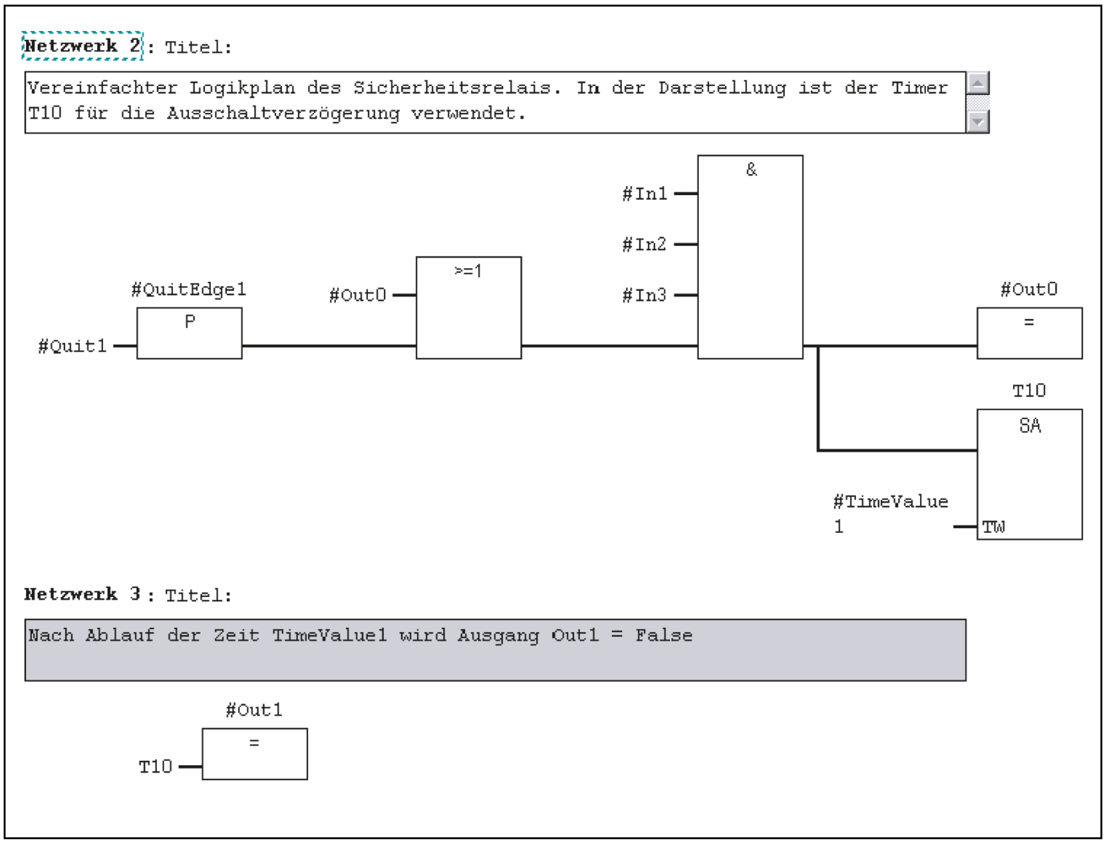
To bring the outputs to their basic setting (values = 0) after booting, the parameter "FirstRun" must be configured as follows. The parameter "FirstRun" must be switched to the value TRUE via a retentive data (memory bit, bit in data block) on the 1st run after control booting. This data can be preset, e.g., in OB100. The parameter "FirstRun" is reset to FALSE when FB10 is executed for the first time. Separate data must be used for parameter "FirstRun" for each call with its own instance.

General conditions

- FB10 has multi-instance capability
- After the start of the SPL program, FB10 is called once per SI relay in the cyclic part of the PLC basic program (OB1).
- Every call of FB10 must be assigned a separate instance DB from the user area.

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.

Netzwerk 3: Titel:

Nach Ablauf der Zeit TimeValue1 wird Ausgang Out1 = False

Declaration of the function

```
FUNCTION_BLOCK FB10
VAR_INPUT
    In1 :          BOOL;
    In2 :          BOOL;
    In3 :          BOOL;
    Ackn1 :        BOOL;
    Ackn2 :        BOOL;
    TimeValue1 :   TIME;
    TimeValue2 :   TIME;
    TimeValue3 :   TIME;
END_VAR
VAR_OUTPUT
    Out0 :         BOOL;
    Out1 :         BOOL;
    Out2 :         BOOL;
    Out3 :         BOOL;
END_VAR
VAR_INOUT
    FirstRun:      BOOL;
END_VAR
```

Description of formal parameters

Parameter	Typ e	Type	Value range	Description
In1:	I	BOOL	0 (FALSE), 1 (TRUE)	Input 1
In2:	I	BOOL	0 (FALSE), 1 (TRUE)	Input 2
In3:	I	BOOL	0 (FALSE), 1 (TRUE)	Input 3
Ackn1 :	I	BOOL	0 (FALSE), 1 (TRUE)	Acknowledgment input 1
Ackn2 :	I	BOOL	0 (FALSE), 1 (TRUE)	Acknowledgment input 2
TimeValue1 :	I	S5TIME		Time value 1 for OFF delay
TimeValue2 :	I	S5TIME		Time value 2 for OFF delay
TimeValue3 :	I	S5TIME		Time value 3 for OFF delay
Out0 :	O	BOOL	0 (FALSE), 1 (TRUE)	Output, instantaneous (no delay)
Out1 :	O	BOOL	0 (FALSE), 1 (TRUE)	Output, delayed by TimeValue1
Out2 :	O	BOOL	0 (FALSE), 1 (TRUE)	Output, delayed by TimeValue2
Out3 :	O	BOOL	0 (FALSE), 1 (TRUE)	Output, delayed by TimeValue3
FirstRun:	I/O	BOOL	0 (FALSE), 1 (TRUE)	Call bit memory

14.17.9 FB11: Brake test

Function

The braking operation check should be used for all axes, which must be prevented from moving in an uncontrolled manner by a holding brake. "Vertical axes" are the main application.

The machine manufacturer can use the PLC user program to regularly close the brake at a suitable instant (e.g. every 8 hours) and have the drive exert torque/force in addition to the weight of the axis. In errorfree operation, the brake can produce the necessary braking torque/ braking force. The axis will hardly move during this.

When an error occurs, the actual position value exits the parameterizable monitoring window. The position controller prevents sagging of the axis. The function test of the brake mechanical system is negatively acknowledged.

References

A detailed description of the parameterization of the NC and drive within the framework of the "Safety Integrated" function can be found in:

Function Manual for Safety Integrated

Starts the brake test

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. controller inhibit, feed enable). Furthermore, the signal at the axis/spindle DB31, ...DBX28.7 (PLC-controlled axis) is to be set to status 1 by the user program for the complete duration of the test.

Before activating the NC/PLC interfaces DB31, ...DBX28.7 (PLC-controlled axis), the axis is to be switched as "neutral axis", e.g. DB31, ...DBX8.0 - 8.3 (assign NC axis to channel) is to be set to channel 0, as well as DB31, ...DBX8.4 (activation signal when changing this byte) is to be set.

The return message:

- about the current status can be queried in DB31, ... DBB68.
- the Nc via the signal DB31, ...DBX63.1 (PLC controls axis) is to be awaited before the block is started. The direction in which the drive must produce its torque/force is specified by the PLC in the form of a "traversing motion" (e.g., via FC18).

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 FC18

If FC18 is called over the further course of the user program for the same axis, 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 FC18 parameters. A second option is to call the FC18 repeatedly, in which case the inactive FC18 will not be processed by the program. A multiple-use interlock must be provided.

Structure of a brake test

Step		Expected feedback	Monitoring time value
1	Start brake test	DBX71.0 = 1	TV_BTactiv
2	Close brake	Bclosed = 1	TV_Bclose
3	Output traversing command	DBX64.6 OR DBX64.7	TV_FeedCommand
4	Issue test travel command	DBX62.5 = 1	TV_FXSreached
5	Wait for the holding time	DBX62.5 = 1	TV_FXShold
6	Deselect brake test / open brake	DBX71.0 = 0	TV_BTactiv
7	Output test ok	---	---

General conditions

- FB2 has multi-instance capability.
- Every call of FB11 must be assigned a separate instance DB from the user area.

Declaration of the function

```
Function_BLOCK FB11
VAR_INPUT
    Start              BOOL;
    Ackn :             BOOL;
    Bclosed :          BOOL;
    Axis :             INT;
    TimerNo :          TIMER ;
    TV_BTactiv :       S5TIME ;
    TV_Bclose :        S5TIM;
    TV_FeedCommand :   S5TIME ;
    TV_FXSreached :    S5TIME ;
    TV_FXShold :       S5TIME ;
END_VAR
VAR_OUTPUT
    CloseBrake :  BOOL;
    MoveAxis :    BOOL;
    Done :        BOOL;
    Error :       BOOL;
    State : :     BYTE ;
END_VAR
```

Description of formal parameters

Signal	Type	Type	Description
Start	I	BOOL	Starts the brake test
Ackn	I	BOOL	Acknowledge fault
Bclosed	I	BOOL	Checkback input whether Close Brake is activated (singlechannel - PLC)
Axis	I	INT	**Axis number of axis to be tested**
TimerNo	I	TIMER	Timer from user program
TV_BTactiv	I	S5TIME	Monitoring time value → brake test active, 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 → Travel command given Check travel command after MoveAxis has been set
TV_FXSreched	I	S5TIME	Monitoring time value → fixed stop reached

Signal	Type	Type	Description
TV_FXShold	I	S5TIME	Monitoring time value → test brake
CloseBrake	O	BOOL	Request, close brake
MoveAxis	O	BOOL	Request, initiate traversing motion
Done	O	BOOL	Test successfully completed
Error	O	BOOL	An error occurred
State	O	BYTE	See paragraph "Error identifiers"

Fault IDs

State	Description
0	No error
1	Start conditions not fulfilled, e.g. the axis is not in closed-loop control / brake closed / axis inhibited
2	No NC checkback in "Brake test active" signal on selection of brake test
3	No "Brake applied" checkback by input signal Bclosed
4	No travel command output (e.g., axis motion has not been started)
5	Fixed end stop will not be reached → axis RESET was initiated.
6	Traversing inhibit/Approach too slow → fixed stop cannot be reached. TV FXSreached monitoring timeout
7	Brake is not holding at all (the end position is reached)/approach speed is too high
8	Brake opens during the holding time
9	Error when deselecting the brake test
10	Internal error
11	"PLC-controlled axis" signal not enabled in the user program

Example of calling FB11:

```
UN    M        111.1;   //Request to close brake, Z axis of FB
=     O        85.0;    //Brake control, Z axis
AUF            Axis3";  //Brake test, Z axis
O     I        73.0;    //Brake test trigger, Z axis
O     M        110.7;   //Brake test running
FP    M        110.0;
UN    M        111.4;   //Error has occurred
S     M        110.7;   //Brake test running
S     M        110.6;   //Next step
JCN   m001
L     DBB      68;
AW    W#16#F;
T     MB       115;     //flag channel state
L     B#16#10
T     DBB      8;       //Request neutral axis
```

```
m001: U     DBX      68.6;    //Checkback signal, axis is neutral
      U     M        110.6;
      FP    M        110.1;
      R     M        110.6;
      S     M        110.5;   //Next step
      S     DBX      28.7;    //Request PLC-monitored axis

      U     DBX      63.1;    //Checkback signal, axis monitored by PLC
      U     M        110.5;
      FP    M        110.2;
      R     M        110.5;
      S     M        111.0;   //Start brake test for FB

      CALL FB11, DB211 (//brake test block
            Start             :=M   111.0,   //Start brake test
            Ackn              := I  3.7,     //Acknowledge error with RESET
                                             key
            Bclosed           := I  54.0,    //Return message close brakes
                                             //controlled
            Axis              := 3,          //Axis number of axis to be
                                             tested
                                             //Z axis
            TimerNo           := T  110,     //Timer number
            TV_BTactiv        := S5T#200MS,  //Monitoring time value:
                                             //Brake test active DBX71.0
            TV_Bclose         := S5T#1S,     //Monitoring time value:
                                             //Brake closed
            TV_FeedCommand    := S5T#1S,     //Monitoring time value:
                                             //Traversing command output
            TV_FXSreache      := S5T#1S,     //Monitoring time value:
                                             //Fixed stop reached
            TV_FXShold        := S5T#2S,     //Monitoring time value:
                                             //Brake test time
            CloseBrake        :=M   111.1,   //Request to close brake
            MoveAxis          :=M   111.2,   Initiate //Request traversing
                                             motion //
            Done              :=M   111.3,   //Test successfully completed
            Error             :=M   111.4,   //Error has occurred
            State             := MB 112);    //Error status

      AUF            "Axis3"; //Brake test, Z axis

      U     M        111.2;   //Moveaxis
      FP    M        111.5;   //FC18 Start
      S     M        111.7;   //Start FC18

      O     M        111.3;   //Test successfully completed
```

```
       O     M        111.4;   //Error has occurred
       FP    M        110.3;
       R     DBX      28.7;    //Request, PLC-monitored axis

       UN    DBX      63.1;    //Checkback signal, axis monitored by PLC
       U     M        111.0;   //Start brake test for FB
       U     M        110.7;   //Brake test running
       FP    M        110.4;
       R     M        111.0;   //Start brake test for FB
       R     M        110.7;   //Brake test running

//optional begin
       JCN   m002:
       L     MB       115;     //old channel status
       OW    W#16#10;
       T     DBB      8;       //Request channel axis
m002:  NOP   0;
//optional end

       CALL "SpinCtrl" (//Traverse Z axis
             Start    :=M      111.2,    //Start traversing motion
             Stop     := FALSE,
             Funct    := B#16#5,         //Mode: Axis mode
             Mode     := B#16#1,         //Procedure: Incremental
             AxisNo   := 3,              //Axis number of axis to be
                                         traversed
                                         //axis Z-axis
             Pos      := -5.000000e+000, //Traversing distance: Minus 5 mm
             FRate    := 1.000000e+003,  //Feedrate: 1000 mm/min
             InPos    :=M      113.0,    //Position reached
             Error    :=M      113.1,    //Error has occurred
             State    := MB    114);     //Error status

       AUF            "Axis3"; //Brake test, Z axis
       U     M        113.0;   //Position reached
       O     M        113.1;   //Error has occurred
       FP    M        113.2;
       R     M        111.7;   //Start FC18
```

14.17.10 FB29: Signal recorder and data trigger diagnostics

Function

Signal recorder

FB29 "Diagnostics" allows various diagnostic routines to be performed on the PLC user program. A diagnostic routine logs signal states and signal changes. In this diagnostic routine, function number 1 is assigned to the "Func" parameter. Up to 8 signals of the parameters "Signal_1" to "Signal_8" are recorded in a ring buffer each time one of the signals changes. The current information of parameters "Var1" as BYTE value, and "Var2" and "Var3" as INTEGER values are also stored in the ring buffer.

The number of past OB1 cycles is also stored in the buffer as additional information. This information enables the graphical evaluation of signals and values in OB1 cycle grid.

Call rule

First call of FB29 in OB1 cycle: Parameter "NewCycle" = 1

All other calls of FB29 in the same OB1 cycle: Parameter "NewCycle" = 0

Ring buffer

The ring buffer, which must be defined by the user, must have an ARRAY structure specified as in the source code. The array can have any number of elements. A size of 250 elements is recommended. The "ClearBuf" parameter is used to clear the ring buffer and set the "BufAddr" pointer to the start. The instance DB related to the FB29 is a DB from the user area and is to be transferred to the FB Diagnostics with the parameter "BufDB".

Data trigger

The data trigger function is intended to allow triggering on specific values (or bits) at any permissible memory cell. The cell to be triggered is "rounded" with a bit mask ("AndMask" parameter) before the "TestVal" parameter is compared in the diagnostic block.

Note

The source code for the function is available in the source container of the basic-program library under the name "Diagnose.awl". The instance DB and the ring buffer DB are also defined in this source block. The function call is also described in the function. The DB numbers and the call must be modified.

Declaration of the function

```
FUNCTION_BLOCK FB29
VAR_INPUT
Func              : INT;
    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;
    TestVal        : WORD;
    AndMask        : WORD;
END_VAR
VAR_OUTPUT
    TestIsTrue     : BOOL;
END_VAR
VAR_IN_OUT
    BufAddr        : INT;
END_VAR
```

Structure of the ring buffer

```
TITLE =
                                    //Ring buffer DB for FB29
VERSION : 1.0

STRUCT
   Field: ARRAY [0 .. 249 ] OF      //can be any size of this struct
   STRUCT

   Cycle : INT;                     //Delta cycle to last storage in buffer
   Signal_1 : BOOL;                 //Signal names same as FB29
   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

Signal	Type	Type	Value range	Description
Func:	I	INT	0, 1, 2	Function 0: Switch off 1: Signal recorder 2: Data trigger
Parameters for function 1				
Signal_1 ... Signal_8:	I	BOOL	0 (FALSE), 1 (TRUE)	Bit signals checked for change
NewCycle:	I	BOOL	0 (FALSE), 1 (TRUE)	See paragraph "Function" above
Var1 :	I	BYTE		Additional value
Var2 :	I	INT		Additional value
Var3 :	I	INT		Additional value
BufDB:	I	INT		Ring buffer DB no.
ClearBuf:	I	BOOL	0 (FALSE), 1 (TRUE)	Delete ring buffer DB and reset pointer Bu-fAddr
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 paragraph "Function" above
TestIsTrue:	O	BOOL	0 (FALSE), 1 (TRUE)	Result of comparison

Configuration steps

- Select function of diagnostics block.
- Define suitable data for the recording as signal recorder or data triggering.
- Find a suitable point or points in the user program for calling the diagnostics FB.
- Create a data block for the ring buffer, see call example.
- Call the diagnostics FB with parameters in the user program.

In function 1, it is advisable to clear the ring buffer with the "ClearBuf" parameter. When the recording phase with function 1 is completed, read out the ring buffer DB in STEP7 with the function "opening the data block in the data view". The content of the ring buffer DB can now be analyzed.

Call example

```
FUNCTION FC99: VOID
TITLE =
VERSION : 0.0

BEGIN
NETWORK
TITLE = NETWORK

CALL FB29, DB80(
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
```

14.17.11 FC2 : GP_HP - basic program, cyclic section

Function

The NC/PLC interface is processed by the basic program in cyclic mode (OB1). To keep the runtime to a minimum, only the control and state signals are cyclically transferred. The transfer of auxiliary and G commands is processed only on request from the NC.

Furthermore, the data for handwheel selection, modes, and other operating signals are transferred from the operator panel (HMI) to the NC/PLC interface so that the modes support the selection from the MCP or HMI as required.

The transfer of HMI signals to the NC/PLC interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB1 (DB7).

Handwheel selection signals

Requirement: FB1, parameter "HWheelMMC == "TRUE"

The handwheel selection signals from the HMI are decoded and activated in the respective machine or geometry axis of the respective handwheel.

Declaration

```
FUNCTION FC2: 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 OB1.

The following example contains the standard declarations for OB1 and the calls for the basic program (FC2), the transfer of the MCP signals (FC19), and the acquisition of error and operational messages (FC10).

```
ORGANIZATION_BLOCK OB1
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 FC2;                 // Call basic program as 1st FC
//INSERT USER PROGRAM HERE
CALL FC19(                 //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
                                      // (Number of the associated machine
                                      axis)
FeedHold :=                m22.0,     //Feed stop signal
                                      //Modal
SpindleHold :=  db2.dbx151.0);        //Spindle stop modal
                                      //in message DB
CALL FC10(                            // Error and operational messages
               ToUserIF := TRUE,      //Signals transferred from DB2
                                      //to interface
               Ackn :=     I6.1);     //Acknowledgment of error messages
                                      //via I6.1
END_ORGANIZATION_BLOCK
```

14.17.12 FC3: GP_PRAL - basic program, interruptdriven section

Function

Block-synchronized transfers from the NC to the PLC (auxiliary and G commands) are processed in the alarm-driven 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 transferred to the application interface at the start of the next OB1 cycle.

Highspeed auxiliary functions programmed immediately one after the other, are not lost for the user program. This is ensured by a mechanism in the basic program.

Normal auxiliary functions are acknowledged to the NC only after one completed cycle duration. This allows the application to issue a read disable to the NC.

The **G commands** 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 OB40 ("GP_IRFromNCK") is set by the basic program, only when the FB1 parameter "UserIR" is TRUE. This data is not set on other events (process alarms through I/Os). This information makes it possible to branch into the associated interrupt routine in the user program in order to initiate the necessary action.

To be able to implement highspeed, jobdriven processing of the user program for the machine, the following NC functions are available in the interrupt processing routine (OB40 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 OB40 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 process alarms (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 FB1 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 for a specific channel in the local data "GP_AuxFunction" for the user program (if "GP_AuxFunction[1]" is set, an auxiliary function is available for the 1st channel).

In the related channel-DB the change signal and the function value are available for the user. The request signal of 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 OB40 is set for this purpose. The PLC user program can thus check the tool management DB (DB72 or DB73) for the tool change function and initiate the tool change operation.

Position reached

In the bit structure, "GP_InPosition" of the local data of OB40 is specific to the machine axis (each bit corresponds to an axis/spindle, e.g. GP_InPosition[5] corresponds to the 5th axis).

If a function has been activated via FC18 (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 FC3: VOID
// No parameters
```

Call example

As far as the time is concerned, the basic program must be executed **before** other alarm-driven user programs. It is, therefore, called first in OB40.

The following example contains the standard declarations for OB40 and the call for the basic program.

```
ORGANIZATION_BLOCK OB40
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 NC for user
GP_TM : BOOL;                          //Tool management
GP_InPosition : ARRAY [1..3] OF BOOL;  //Axis-oriented for positioning,
                                       //Indexing axes, spindles
GP_AuxFunction : ARRAY [1..10] OF BOOL;//Channel-oriented for auxiliary
                                       functions
GP_FMBlock : ARRAY [1..10] OF BOOL;    //Currently not used
//Further local user data may be defined from this point onwards
END_VAR
BEGIN
   CALL FC3;
   //INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

14.17.13 FC5: GP_DIAG - basic program, diagnostic alarm and module failure

Function

The block FC5 "GP_DIAG" is used to record assembly disruptions and failures.

A PLC stop can be triggered via the parameter "PlcStop". The PLC stop is only triggered for incoming events. The MCPs connected to the parameterized PROFIBUS (DP1) at FB1 are excluded from this.

Declaration

```
FUNCTION FC5: VOID
   VAR_INPUT
      PlcStop: BOOL:= TRUE;
   END_VAR
```

Call example

The basic program should be run through after processing of the user programs. This is recommended because a PLC stop can be triggered by the FC5.

The example contains the standard declaration for OB82 and OB86 and the call of the FC5.

```
ORGANIZATION_BLOCK OB82
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 FC5
            (PlcStop := FALSE) ;
END_ORGANIZATION_BLOCK

ORGANIZATION_BLOCK OB86
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 FC5
             (PlcStop := TRUE) ;
END_ORGANIZATION_BLOCK
```

14.17.14 FC6: TM_TRANS2 - transfer block for tool management and multitool

Function

The block FC6 "TM_TRANS2" is used for position changes of the tools, state changes, and multitool.

The FC6 block has the same functionality as the FC8 block, plus the **multitool** functionality.

The description of FC6 only contains the **multitool** functionality.

The functionality of FC8 is described in "FC8: TM_TRANS - transfer block for tool management (Page 1052)".

Declaration of the function

```
FUNCTION FC6: VOID
VAR_INPUT
   Start:           BOOL;
   TaskIdent:       BYTE ;
   TaskIdentNo:     BYTE ;
   NewToolMag:      INT;
   NewToolLoc:      INT;
   OldToolMag:      INT;
   OldToolLoc:      INT;
   Status:          INT;
   MtoolPlaceNum:   INT;
END_VAR
VAR_OUTPUT
   Ready:           BOOL;
   Error:           INT;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

Signal	Type	Type	Value range	Description
Start:	I	BOOL	0 (FALSE), 1 (TRUE)	See block description FC8
TaskIdent:	I	BYTE		See block description FC8
TaskIdentNo:	I	BYTE		See block description FC8
NewToolMag:	I	INT		See block description FC8
NewToolLoc:	I	INT		See block description FC8
OldToolMag:	I	INT		See block description FC8
OldToolLoc:	I	INT		See block description FC8
Status:	I	INT		See block description FC8
MtoolPlaceNum:	**I**	**INT**		**Multitool location No.**
Ready:	O	BOOL	0 (FALSE), 1 (TRUE)	See block description FC8
Error:	O	INT		See block description FC8

14.17.15 FC7: TM_REV - transfer block for tool change with revolver

Function

After a revolver has been changed, the user calls block FC7 "TM_REV". The revolver number corresponding to interface number in DB73 must be specified in parameter "ChgdRevNo" for this purpose. As this block is called, the associated "Interface active" bit in data block DB73.DBW0 of FC7 is reset after parameter "Ready" == TRUE is returned.

Job executed correctly

If the job was executed correctly, then "Ready" == 1. The user must then set the parameter "Start" = 0 or no longer call FC7.

Job executed with errors

If the job was executed with errors, then parameter "Ready" == 0 and parameter "Error" == 1. The job must be repeated in the next PLC cycle. Since the parameter "Start" does not need a positive edge for a subsequent job, "Start" = 1 remains, because the job has not yet been completed. See "Call example" and "Pulse diagram" below.

General conditions

- Block FC7 may only be started with parameter "Start" = 1 if an activation signal for the associated interface (DB73.DBW0) for this transfer has been supplied by the tool management function.
- A cancellation of a transfer, e.g. by a channel reset, is not permitted.
- Parameter "Start" = 1, until parameter "Ready" == 1 or "Error" == 1

References

- For detailed information about tool management, refer to the Function Manual Tool Manager.
- PI services for tool management, see:
  - FB4: PI_SERV - request PI service (Page 988)
  - FC8: TM_TRANS - transfer block for tool management (Page 1052)
  - FC22: TM_DIR - direction selection for tool management (Page 1095)

Manual revolver switching

If the revolver is rotated in manual operation, neither a tool change nor an offset selection is associated with this operation. The first step is the removal of the tool from the toolholder back to its location in the revolver. An asynchronous transfer must be performed with FC8 (alternative: FC6). The associated parameter settings are shown below:

```
TaskIdent = 4
TaskIdentNo = Channel no.
NewToolMag = Magazine no. of the revolver
NewToolLoc = Original location of the tool
OldToolMag = Magazine no. of the buffer storage (spindle) = 9998
OldToolLoc = Buffer storage no. of the spindle
Status = 1
```

If the revolver is now turned to an arbitrary position at which a tool is located, this tool must be activated. This is done easiest by the new T programming in the part program. However, if this should take place, for example, at the end of revolver switching from the PLC user program, an ASUP must be started for this purpose. The current revolver position must be transferred to the ASUP. In this way, the tool at this location is determined in the ASUP and is selected (see Jobshop example in the toolbox).

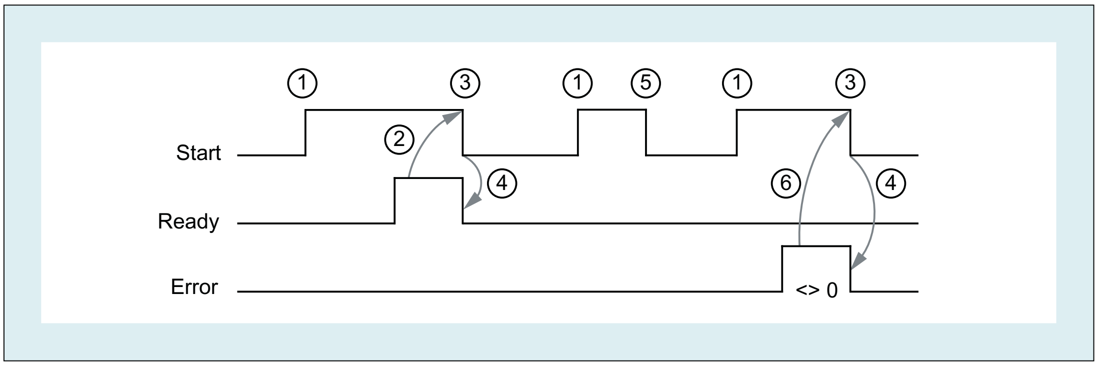
Declaration of the function

```
FUNCTION FC7:       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

Signal	Typ e	Type	Value range	Description
Start:	I	BOOL	0 (FALSE), 1 (TRUE)	1 = start transfer
ChgdRevNo:	I	BYTE	1, 2, 3, ...	Number of revolver interface
Ready:	O	BOOL	0 (FALSE), 1 (TRUE)	1 = Transfer complete
Error:	O	INT	0, 1 , 2, 3	Error checkback 0: No error has occurred 1: No revolver present 2: Illegal revolver number in parameter "ChgdRev-No" 3: Illegal job ("interface active" signal for selected revolver = "FALSE")

Pulse diagram



① User: Set request, Start = 0 → 1

② FB4: PI service successfully completed, Ready = 1
User: Reset request, IF Ready == 1 THEN Start = 0

③ User: IF Ready == 1 THEN reset request: 1 → 0

④ FB4: Reset job confirmation, Ready = 0

⑤ User: IF Ready == 0 AND Error == 0 THEN reset request Start = 1 → 0 **not permissible**

⑥ FB4: PI service completed with errors, Error = 1
User: Reset request, IF Ready == 1 OR Error == 1 THEN Start = 0, possible further error handling

Call example

```
CALL     // Tool management: Transfer block for revolver
FC7(
    Start :=           m 20.5,         // Start := "1 " => initiate the
                                       transfer
    ChgdRevNo :=       DB61.DBB1,
    Ready :=         m 20.6,
    Error :=         DB61.DBW12
};
u       m 20.6;                     // Poll ready
r       m 20.5;                     // Reset start
spb     m001;                       // Jump, if everything OK
l       db61.dbw 12;                // Error information
ow      w#16#0;                     // Evaluate error
spn     error:                      // Jump to troubleshooting, if <> 0
m001:                               // Start of another program
error:
r       m 20.5;                     // Start reset, if an error has
                                    occurred
```

14.17.16 FC8: TM_TRANS - transfer block for tool management

Function

The user calls this block FC TM-TRANS when the position of the tool or the status of the transfer operation changes. The parameter "TaskIdent" specifies the transfer job for the block FC8 at the tool management interface:

- For loading/unloading positions
- For spindle change positions
- For revolver change positions as transfer identifier
- Asynchronous transfer
- Asynchronous transfer with location reservation

The interface number is indicated in parameter "TaskIdentNo".

Example for loading point 5:

Parameter "TaskIdent":= 1 and "TaskIdentNo":= 5.

Furthermore, the **current** tool positions and status data (list of "Status" parameter in the following text) are also transferred for this transfer function.

Note

FC8 informs the NC of the current positions of the old tool.

The NC knows where the old and the new tool were located until the position change.

In the case of a transfer without a socalled "old tool" (e.g. on loading), the value 0 is assigned to parameters "OldToolMag", "OldToolLoc".

Block FC TM_TRANS may be started only with "Start" parameter = "TRUE" if an activation signal for the appropriate interface (DB71, DB72, DB73 in word 0) for this transfer has been supplied by the tool management function.

When this job is executed correctly, the output parameter "Ready" contains the value TRUE.

The user must then set the"Start" parameter to FALSE or not call the block again.

If the "Ready" parameter = FALSE, the error code in the "Error" parameter must be interpreted (see Call example FC8 and pulse 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 DB71 or DB72 or DB73, word 0 is deactivated (process completed). The appropriate bit for the interface is set to 0 by FC8.

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 the PLC to the tool management (e.g. power failure during an active command or independent changes in the position by the PLC), FC8 is called with "TaskIdent" = 4 or 5. This call does not require interface activation by the 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 location of the tool is specified in the parameters "OldToolMag" and "OldToolLoc". The current location of the tool is specified in the 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.

Supplementary conditions

- A cancellation of a transfer, e.g. by a channel reset, is not permitted.
- Parameter "Start" = 1, until parameter "Ready" == 1 or "Error" == 1

References

- For detailed information about tool management, refer to the Function Manual Tool Management.
- PI services for tool management
  - FB4: PI_SERV - request PI service (Page 988)
  - FC7: TM_REV - transfer block for tool change with revolver (Page 1049)
  - FC22: TM_DIR - direction selection for tool management (Page 1095)

Declaration of the function

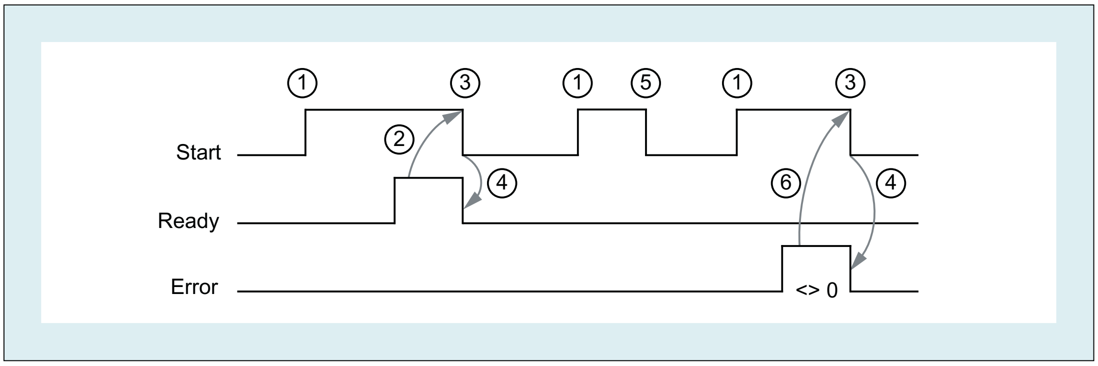
```
FUNCTION FC8: VOID
//NAME :TM_TRANS
VAR_INPUT
    Start :             BOOL;
    TaskIdent:          BYTE ;
    TaskIdentNo:        BYTE ;
    NewToolMag:         INT;
    NewToolLoc:         INT;
    OldToolMag:         INT;
    OldToolLoc:         INT;
    Status:             INT;
END_VAR
VAR_OUTPUT
    Ready               BOOL;
    Error :             INT;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

Signal	Type	Type	Value range	Meaning
Start:	I	BOOL	0 (FALSE), 1 (TRUE)	1: Start transfer
TaskIdent:	I	BYTE	1, 2, 3, 4, 5	Interface or task identifier 1: Loading/unloading location 2: Spindle change position 3: Revolver change position 4: Asynchronous transfer 5: Asynchronous transfer with location reserva- tion
TaskIdentNo:	I	BYTE	1, 2, 3 ... 10	Number of the associated interface or channel number.

Signal	Type	Type	Value range	Meaning
NewToolMag:	I	INT	-1, 0, 1, 2 ...	Current magazine number of tool to be replaced -1: Tool remains at its location NewToolLoc = any value Only permissible if TaskIdent = 2
NewToolLoc:	I	INT	0, 1, 2 … max. location no.	Current location number of new tool
OldToolMag:	I	INT	-1, 0 ...	Current magazine number of tool to be replaced -1: The 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, 2, 3 ... 7, 103, 104, 105	Status information about transfer operation
Ready:	O	BOOL	0 (FALSE), 1 (TRUE)	1: Transfer completed
Error:	O	INT	0 ... 65535	Error checkback 0: No error has occurred 1: Unknown "TaskIdent" 2: Unknown "TaskIdentNo" 3: Impermissible job, ("Interface active" signal for selected revolver == 0) Other value: The number corresponds to the error message of the tool management function in the NC caused by this transfer.

Pulse diagram



① Activation of function by means of a positive edge
② Positive acknowledgment: Tool management has been transferred
③ Reset function activation after receipt of acknowledgment
④ Signal change using FC
⑤ This signal chart is not permissible. The job must generally be terminated since the new tool positions must be conveyed to the tool management in the NC.
⑥ Negative acknowledgment: Error occured, error code in the output parameter Error

Status

Status	Description
1	**The tool management job has been completed.** The parameters "NewToolMag", "NewToolLoc", "OldToolMag", "OldToolLoc" of the FC8 block should be parameterized to the actual positions of the tools involved. Except in the case of preparing the change, they are normally the specified target positions of the tools of the associated tool management interface, see also "Explanations of the formal parameters". 1. In the case of **loading/unloading/reloading**, the tool has arrived at the required target address. If the bit in the interface in DB 71.DBX (n+0).3 "position at loading point" is enabled, status 1 cannot be used for the function termination. Status 5 must be used for correct termination. 2. In the case of "**Prepare change**", the new tool is now available. The tool may, for example, be positioned in a buffer (gripper). In some cases, the target (magazine, location) of the old tool has been moved to the toolchange position after placement of the new tool in a buffer. However, the old tool still remains in the spindle. The preparations for a tool change are thus complete. After this 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 completed.
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 NC. 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.
3	**An error has occurred** The tool positions must not have been changed. Any changes to the magazine positions of the tool that have taken place in the meantime must be notified beforehand, for example, with status = 105 via FC8 transfer block. Only then will the tool positions be taken into account by the tool management function.
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). The magazine location specified for the "old" tool must be free. Once this status has been passed to the tool management in the NC, a new preparation command is generated (Status_4 = final acknowledgment) and output to the DB72, for which the requested magazine position of the old tool is considered. No new tool search is performed explicitly, the positions for "NewToolMag" and "NewToolLoc" are taken from the original preparation command. But this is done only when this position is free. Parameters "NewToolMag" and "NewToolLoc" are not taken into account.

Status	Description
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. The status 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 NC. 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 the loading point:** If the bit in the interface in DB 71.DBX (n+0).3 "position at loading point" is enabled, then only status 5 be used for the function termination (not status 1).
6	**The tool management job has been completed** This status has the same function as status 1, but, in addition, a reservation of the source location is carried out. This status is only permitted when reloading. The command is ended and the source location of the tool is reserved if the target location is in a buffer magazine.
7	**Initiate repetition of the command "Prepare Tool"** This status is only admissible in conjunction with the "Change tool" command. This status is intended for use when the "new" tool has changed its position (e.g. via an asynchronous command of the "new" tool). After "Ready = 1" has been provided by FC8, the "Prepare Change" command is repeated automatically with the same tool. For the automatic repetition, a new tool search is carried out. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag", and "OldToolLoc" correspond to the original tool positions.
103	**The "new" tool can be inserted** This status is permitted only in the tool change preparation, when the PLC may reject the new tool (e.g. in case of MD20310 $MC_TOOL_MANAGEMENT_MASK, bit 4=1 for the possibility, request changed parameter from PLC once again). The tool positions have remained unchanged. This status is therefore necessary, when the processing is to be continued in the NC without an unnecessary stop.
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. The status enables the tool management to adjust the current position of a magazine and to improve the search strategy for subsequent commands.
105	**The specified buffer location 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 status information 1 to 7 leads to the termination of the command. If FC8 receives a status information, the "Interface active bit" of the interface specified in FC8 is reset to "0" (see also interface lists DB 71 to DB 73), thus completing the operation. The behavior is different in the case of status information 103 to 105. When the FC8 receives a 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 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 FC8(           //Tool management transfer block
   Start :=         m 20.5,        // Start := "1 " => initiate the transfer
   TaskIdent :=     DB61.DBB0,
   TaskIdentNo :=   DB61.DBB1,
   NewToolMag :=    DB61.DBW2,     // Current position of new tool
   NewToolLoc :=    DB61.DBW4,
   OldToolMag :=    DB61.DBW6,     // Current position of old tool
   OldToolLoc :=    DB61.DBW8,
   Status :=        DB61.DBW10,    // Status
   Ready :=         m 20.6,
   Error :=         DB61.DBW12);
u m 20.6;                          // Poll ready
r m 20.5;                          // Reset start
spb m001;                          // Jump if everything OK
l DB61.dbw12;                      // Error information
ow w#16#0;                         // Evaluate error
JC error;                          // Jump to troubleshooting

m001:                              // Normal branch

error:                             //Troubleshooting
r m 20.5:                          // Reset start
```

14.17.17 FC9: ASUP - start of asynchronous subprograms

Function

The block FC9 "ASUP" can be used to trigger any functions in the NC. Before an ASUP can be started from the PLC, it must have been selected and parameterized by an NC program or by FB4 (PI service ASUP). In this case, the channel and the interrupt numbers must match the parameters in FC9.

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 ASUP can be started in the same channel at a time. If several ASUPs are started in **one** PLC cycle, the ASUPs are started in this order in the NC.

Parameter "Start" = 0 must be set by the user if the ASUP has been terminated ("Done" == 1) or an error has occurred ("Error" == 1).

For processing jobs, each FC9 needs its own parameter "Ref" from the global user area. This parameter is for internal use only and must not be changed by the user. The parameter "Ref" is initialized with the value 0 in the first OB1 cycle and, for this reason, every FC9 must be called **absolutely**. Alternatively, the user can initialize parameter "Ref" with a value of 0 during startup. This option makes conditional calls possible. A conditional call requires parameter "Start" = 1 during activation of FC9 until a negative edge change has occurred at parameter "Done" (1 → 0).

General conditions

- The function block FB4 must be terminated before the block FC9 is started.
- The block FC9 cannot be started if DB10, DBX56.1 == 1 (emergency stop).
- Block FC9 must not be started if channel reset is active in the channel in which the ASUP is to be started.

Declaration of the function

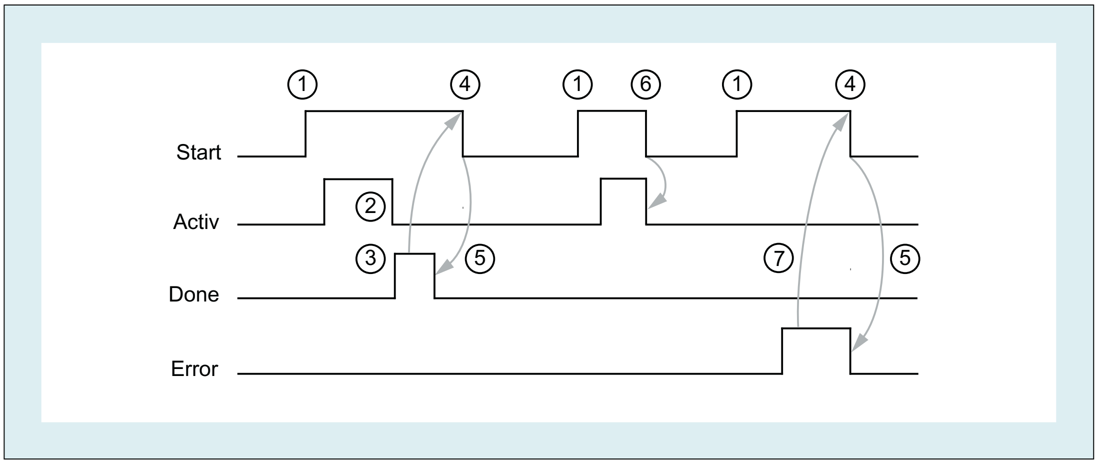
```
FUNCTION FC9: VOID
//NAME :ASUP
VAR_INPUT
    Start :      BOOL;
    ChanNo:      INT;
    IntNo:       INT;
END_VAR
VAR_OUTPUT
    Active:      BOOL;
    Done :       BOOL;
    Error :      BOOL;
    StartErr:    BOOL;
END_VAR
VAR_IN_OUT
    Ref:         WORD;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the ASUP function.

Signal	Type	Type	Value range	Description
Start:	I	BOOL	0 (FALSE), 1 (TRUE)	Job start with positive signal edge
ChanNo:	I	INT	1, 2, 3 ... 10	Channel number
IntNo:	I	INT	1, 2, 3 ... 8	Interrupt number
Active:	O	BOOL	0 (FALSE), 1 (TRUE)	1: active
Done:	O	BOOL	0 (FALSE), 1 (TRUE)	1: ASUP completed
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1: Interrupt switched off
StartErr:	O	BOOL	0 (FALSE), 1 (TRUE)	1: Interrupt number not assigned or deleted
Ref:	I/O	WORD	Global variable (MW, DBW,..)	1 word per FC9 (for internal use)

Pulse diagram



(1) Activation of function

(2) ASUP is active

(3) Positive acknowledgment: ASUP completed

(4) Reset function activation after receipt of acknowledgment

(5) Signal change using FC

(6) Not permitted If function activation is reset prior to receipt of acknowledgement, the output signals are not updated without the operational sequence of the activated function being affected.

(7) Negative acknowledgement: An error occurred

Call example

```
CALL FC9(                    //Start an asynchronous subprogram
                             //in channel 1 interrupt number 1
     Start :=      I 45.7,
     ChanNo :=     1,
     IntNo :=      1,
     Active :=     M 204.0,
     Done :=       M204.1,
     Error :=      M 204.4,
     StartErr :=   M 204.5,
     Ref :=        MW 200);
```

14.17.18 FC10: AL_MSG - error and operating messages

Function

With block FC10 "AL_MSG", the signals entered in DB2 are evaluated and displayed as incoming or outgoing error messages and operational messages on the user interface.

The incoming signals (positive edge) are displayed immediately in the case of both error and operational messages.

Outgoing signals (negative edge) are deleted immediately only for operational messages. In the event of error messages, the messages that are no longer pending are only deleted with the parameter "Quit", i.e. errors remain displayed on the user interface until they have been acknowledged by the user even if the signals are no longer pending.

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 DB2 irrespective of an alarm acknowledgment.

1. If parameter "ToUserIF" = 0, there is no transfer of the signals 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. The FB1 parameter "ExtendChanAxMsg" is evaluated and so avoids the limitation of the usable message ranges by configuring the NCK machine data.
2. If parameter "ToUserIF" = 1, 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 DB2 in conjunction with a message or alarm output. The appropriate information is overwritten in the user interface.

Alternatively to the response described in paragraph 2, the disable and hold signals can be influenced without a message being output by influencing the interface signals with a disable or stop signal state after FC10 is called.

The following program illustrates this method:

```
CALL FC10(
   ToUserIF :=     TRUE,
   Ackn :=         I 6.1);

u m 50.0;                       // Feed disable for channel 1
to DB 21;
s dbx 6.0;                      // Setting the blocking condition
                                // Resetting is done via FC AL_MSG
                                // if M 50.0 outputs the signal "0".
```

FB1 parameter "ExtendAlMsg"

With FB1 parameter "ExtendAlMsg" = TRUE, the new DB2 structure becomes effective (see "Interface PLC/HMI (Page 892)"). For the activation, bit fields for the lock and stop signals are available for 10 channels, 31 axes and maximum 64 user areas (the number of user areas is entered in the FB1 parameter "MsgUser"). The associated functionality is obtained automatically by simply setting/resetting signals in DB2.

The error and operational messages in data block DB2 must be provided in a user-specific way.

FB1 parameter "ExtendChanAxMsg"

With the activation of this parameter, a channel- or axis-number independent acquisition of alarms and messages acts. All DB2 areas are available for users Group signals cannot be transferred to the user interface. The parameter is evaluated only when the FC10 parameter "ToUserIF" is deactivated.

Display on HMI

In DB2, 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 alarms 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.

References:
List Manual for NC variables and interface signals, Section "PLC user interface" > "PLC alarms/ messages"

Declaration of the function

```
FUNCTION FC10:      VOID
    // NAME:        AL_MSG
VAR_INPUT
    ToUserIF :      BOOL;
    Ackn :          BOOL;
END_VAR
END_FUNCTION
```

Description of formal parameters

Signal	Type	Type	Value range	Meaning
ToUserIF :	I	BOOL	0 (FALSE), 1 (TRUE)	1: Transfer of the signals to the user interface in each cycle
Ackn:	I	BOOL	0 (FALSE), 1 (TRUE)	1: Acknowledgment of error messages

Call example

```
CALL FC10(                  // Error and operational messages
   ToUserIF :=   TRUE,      // Signals from DB2 are transferred to interface
   Ackn :=       E6.1       // Acknowledgment of error message via input I6.1
);
```

14.17.19 FC12: AUXFU - call interface for user with auxiliary functions

Function

The block FC12 "AUXFU" is called on an eventdriven basis in the basic program if the channel transferred in the input parameter contains new auxiliary functions. The PLC user can extend FC AUXFU with program instructions for processing his auxiliary function to avoid cyclic polling of the channel DBs. This mechanism permits auxiliary functions to be processed on a jobdriven basis. FC AUXFU is supplied as a compiled empty block in the basic program. In this case, the basic program supplies parameter "Chan" with the channel number. The PLC user knows which channel has new auxiliary functions available. The new auxiliary functions can be determined by the auxiliary-function change signals in the channel concerned.

Declaration of the function

```
FUNCTION FC12: VOID          // Event control of auxiliary functions
VAR_INPUT
    Chan:    BYTE ;
END_VAR
BEGIN
    BE;
END_FUNCTION
```

Explanation of formal parameters

Signal	Type	Type	Value range	Description
Chan:	I	BYTE	0, 1, 2 ... 9	Index of the channel = channel number -1

Example

```
FUNCTION FC12: 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
+ 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
```

14.17.20 FC13: BHGDisp - display control for handheld unit

Function

Block FC13 "BHGDisp" handles the display control for the handheld unit (HHU or HT 2). The information that is to appear on the display must be saved to a string variable. The pointer to the string is specified in the parameter "ChrArray". To do this, a fixed text assignment of 32 characters (HHU) or 64 characters (HT 2) is needed when the data block for this string is created.

16 characters are sent to the HHU per job. The assignment of the characters in the "ChrArray" for the respective line is unambiguous. For line 1, characters 1 to 16 and for line 2, characters 17 to 32 of the string data ChrArray are transferred. In addition, for HT 2 line 3 with characters 33 to 48 is displayed and line 4 with characters 49 to 64. A job takes several OB1 cycles.

Display

Block FC13 checks whether the necessary minimum length of the "ChrArray" exists for operating the handheld unit. If fewer characters exist in the string variables than should be displayed, the line is filled with blank spaces. If several variables are to be entered in the string in one or more PLC cycles without a display output, the display output can be suppressed by parameter "Row" = 0. The transfer of the characters to the rows takes several OB1 cycles. If several rows are to be updated "simultaneously" (parameter "Row" > 1), the rows are updated successively with 16 characters per row.

Variable portions

Variable components within the string can be inserted using the optional number converter functionality with the parameter "Convert" = 1. The variable to be displayed is referenced via the parameter "Addr". The format of the variables is described in the parameter "DataType".

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.

High display resolution

If, for example, the axis value is to be displayed with a higher resolution, the following must be observed:

- The variables are read as before with FB2 or FB5. Instead of anypointer BYTE 8 as criterion for output as 64-bit floating point number, REAL 2 is used (e.g.: P#M100.0 REAL 2).
- When specifying the 64-bit floating point number on the HHU/HT 2, you can select the output format with up to 14 places, distributed freely before and after the decimal point, instead of fixed, specified formats.

HHU output signals

Byte 1 is used by the HHU output signals and the character specifications are used by the block FC13. These may not be written by the PLC user program.

Relevant FB1 parameters

Handheld unit HHU

In OB100, the FB1 parameters must be set for the input and output data of the handheld unit:

- 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).

The 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.

The FB1 parameter "HHU" = 2 must be set for operating an HHU.

HT 2 handheld terminal

If HT 2 is used, FB1 parameter "HHU" = 5 must be set. The parameters of the input and output data must be set, as described in the above paragraph "Handheld Unit HHU".

The value that was configured at S2 of the DIP-Fix switch (rotary coding switch) of the connecting module of the HT 2 must be assigned to the parameters "BHGRecGDNo" and "BHGRecGBZNo".

Declaration of the function

```
DATA_BLOCK "strdat"
  STRUCT
    disp:      STRING [32]:= 'character_line1 character_line2';
  END_STRUCT;
BEGIN
END_DATA_BLOCK
```

```
FUNCTION FC13: VOID
  VAR INPUT
    Row :        BYTE ;
    ChrArray :   STRING ;
    Convert :    BOOL;
    Addr:        POINTER;
    DataType :   BYTE ;
    StringAddr : INT;
    Digits :     BYTE ;
END VAR
VAR OUTPUT
    Error :      BOOL;
END VAR
```

Description of formal parameters

Signal	Type	Type	Value range	Description
Row :	I	BYTE	0, 1, 2, ... 8 B#16#F	Display line "binary" evaluation 0: no display output 1: Line 1 2: Line 2 3: Line 1 and line 2 to be changed 4: Line 3 5: Line 1 and line 3 to be changed 8: Line 4 B#16#F automatic change of all 4 lines
ChrArray :	I	STRING	"DBName".<VarName>	Display content as pointer to string string[**32**]: not HT 2 string[**64**]: HT 2
Convert :	I	BOOL	0 (FALSE), 1 (TRUE)	Activation of numerical conversion
Addr:	I	Pointer		Pointer to the variable to be converted

Signal	Type	Type	Value range	Description
DataType:	I	BYTE	1, 2, 3 ... 8, B#16#13, B#16#30	Data type of the tag 1: BOOL, 1 character 2: BYTE, 3 characters 3: CHAR, 1 character 4: WORD, 5 characters 5: INT, 6 characters 6: DWORD, 7 characters 7: DINT, 8 characters 8: REAL, 9 characters (7 digits plus a sign and a decimal point; for places after the decimal point, refer to the Digits Parameter) B#16#13: String, up to 32/64 characters, "Addr" must be a pointer to a STRING. B#16#30: REAL64, (12 characters: 10 digits plus a sign and a decimal point; for places after the decimal point, refer to the Digits Parameter)
StringAddr :	I	INT	1 ... 32 / 64	Right-justified address within variable from "ChrArray"
Digits :	I	BYTE	1, 2, 3 ... 9	Number of places after the decimal point: 1 ... 4: DataType REAL 1 ... 9: DataType REAL64
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	Error 1: An error occurred

Ranges of values

Ranges of values of data types	
Data type	**Representable numerical range**
BOOL	0, 1
BYTE	0 ... 255
WORD	0 ... 65535
INT	- 32768 ... 32767
DWORD	0 ... 9999999
DINT	-9999999 ... 9999999
REAL (Digits := 1)	-999999.9 ... 999999.9
REAL (Digits := 2)	-99999.99 ... 99999.99
REAL (Digits := 3)	-9999.999 ... 9999.999
...	...
REAL (Digits := 9)	-0.9999999 ... 0.9999999

Call example

// DB with name strdat in the simple table, data element disp is declared as String[32] (for HT 2: // String[64]) and completely assigned with characters

```
CALL FC13(
   Row :=        MB 26,
   ChrArray :=   "strdat".disp,
   Convert :=    M 90.1,
   Addr :=       P#M 20.0,      // Number to be converted
   DataType :=   MB 28,         // Data type of the variables
   StringAddr := MW 30,
   Digits :=     B#16#3,        // 3 decimal places
   Error :=      M 90.2);
```

14.17.21 FC17: YDelta - star-delta switchover

Function

Block FC17 is used for star-delta changeover for digital main spindle drives. The changeover can be made in both directions (star > delta or delta > star).

Prerequisite

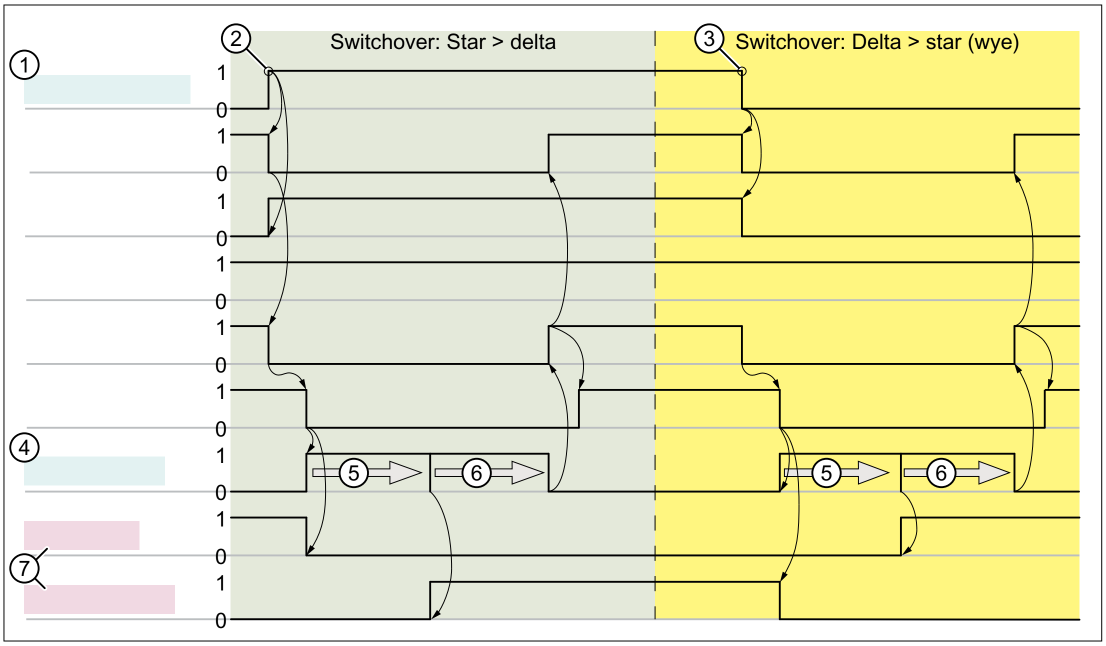
The prerequisite is two isolated contactors. The contactors are controlled via the peripheral output signals configured at the FC17 outputs: "Y" or "Delta".

Internal sequence

The internal sequence when changing over between star and delta after switching over the FC17 control signal is described in the following: "YDelta" displayed.

1. DB31, ...DBX21.5 = 0 (reset feedback signal "motor selected")
   DB31, ...DBX21.x = 1 (set request "2nd motor data set" corresponding to the interface parameterization DB31, ...DBX130.0 - 4 (Page 74))
2. DB31, ...DBX93.7 == 0 (feedback signal "pulses enabled" was reset) ⇒
   - Start FC17 timer
   - FC17: "Y" = 0 (reset output for star contactor)
3. After the FC17 timer elapses (FC17: "TimeVal") ⇒
   - FC17: "Y" = 1 (set output for delta contactor)
4. After the FC17 timer elapses again (FC17: "TimeVal") ⇒
   - The pulses are again internally set
   - DB31, ...DBX93.7 == 1 (feedback signal "pulses enabled" was set) ⇒
   - DB31, ...DBX21.5 = 0 (feedback signal "motor selected" was set)

Signal flow



①	FC17 input: Signal for star-delta changeover
②	- Changeover: Star > delta
③	- Changeover: Delta > star
④	FC17 input: Parameterizable changeover time ""TimeVal"
⑤	- Wait time until the control of the output signal: "Y" or "delta"
⑥	- Wait time until pulse enable
⑦	FC17 output: Signals for contactor control

References

For additional explanations of motor speed adjustments, see:

- Function Manual, Basic Functions; Spindles (S1); Chapter "Configurable gear adaptation"
- Functions Manual, Basic Functions; Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)

Note

Drive parameters

The following drive parameters must be considered for a star-delta changeover:

- p833 (dataset changeover configuration)
  - Bit 0 = **1** (contactor switchover via **application**)
  - Bit 1 = **0** (pulse cancellation by **drive**)
- p826 (motor changeover motor number)
- p827 (motor changeover status word bit number)

General conditions

- Block FC17 must be called absolutely and separately for each spindle.
- The drive pulses are deactivated during a star-delta changeover. The feedback signal to the PLC is realized using:
  - DB31, ...DBX93.7 == 0 (pulses enabled)
  - DB31, ...DBX61.7 == 0 (current controller active)
  - DB31, ...DBX61.6 == 0 (speed controller active)
- For a spindle which is located in an axis mode such as M70 or SPOS, a start-delta changeover is not carried out.
- For a closed-loop position controlled spindle (DB31, ...DBX61.5 == 1 (position control active)), while the spindle is moving it is not permissible to carry out a star-delta changeover. In the case of a fault, Alarm 25050 "Contour monitoring" is displayed, and the star-delta changeover is not executed.
- Once the star-delta changeover has been initiated using FC17, it cannot be delayed by the user, e.g. by waiting until the star-delta contactors change over during the course of operation. A delay such as this must be implemented by the user in the PLC user program.

Declaration of the function

```
VAR_INPUT
    YDelta:         BOOL;
    SpindleIFNo:    INT;
    TimeVal:        S5TIME ;
    TimerNo :       INT;
END_VAR
VAR_OUTPUT
    Y:              BOOL;
    Delta:          BOOL;
END_VAR
VAR_IN_OUT
    Ref:            WORD;
END_VAR
```

Description of formal parameters

Signal	Type	Type	Value range	Description
YDelta:	I	BOOL	0 (FALSE), 1 (TRUE)	Input signal for star-delta changeover The changeover is initiated by a signal change: • 0 (FALSE): Star • 1 (TRUE): Delta
SpindleIFNo:	I	INT	1 ...	Number of the spindle interface (number of the associated machine axis)

Signal	Type	Type	Value range	Description	
TimeVal:	I	S5time	0, 50 ms ...	Switchover time	
				Configured	**Internally effective**
				0	100
				< 50	50
TimerNo:	I	INT	10 ...	Number of the timer being used	
Y:	O	BOOL	0 (FALSE), 1 (TRUE)	Peripheral output for controlling the star contactor	
Delta:	O	BOOL	0 (FALSE), 1 (TRUE)	Peripheral output for controlling the delta contactor	
Ref:	I/O	WORD		Instance for status information Internal use	

Call example

```
CALL FC17 (
    YDelta :=         I 45.7,         // Accept star/delta changeover from
                                      input 45.7
    SpindleIFNo :=    4,              // Spindle interface number: 4
                                      // (Number of the associated machine
                                      axis)
    TimeVal :=        S5T#150ms,      // Changeover time: 150 ms
    TimerNo :=        10,             // Timer: 10
    Y :=              O 52.3,         // Control of the star contactor:
                                      Output 52.3
    Delta :=          O 52.4,         // Control of the delta contactor:
                                      Output 52.4
    Ref :=            MW 50           // Bit memory word 50
};
```

14.17.22 FC18: SpinCtrl - spindle control

Function

Block FC18 "SpinCtrl" can be used to control spindles and axes from the PLC. The block supports the following functions:

- Position spindle
- Rotate spindle
- Oscillate spindle
- Traverse indexing axis
- Traverse positioning axis

Each function is activated by the positive edge of the appropriate initiation signal (start, stop). This signal must remain at a logical "1" 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 has been completed.

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 block with activation of the "Start" or "Stop" parameter. When you do this, the block requests control of the spindle/axis from the NC.

The NC signals the status of the spindle/axis in the associated axis-specific interface DB31, ... DBX68.4 - 7. Once the axis/spindle is operating under PLC control, the travel command for the active status can be evaluated via the relevant axis-specific interface.

Upon completion ("InPos" is True, "Start" changes to zero), the axis/spindle check function is switched to a neutral status using block FC18.

Alternatively, the PLC user program can also request control for the PLC before calling FC18.

By calling this function several times in succession, a better response by the spindle/axis can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding spindle interface in byte 8.

After return of the check, the spindle can again be programmed by the NC program.

References

- Function Manual, Basic Functions; Spindles (S1)
- Function Manual, Extended Functions; Positioning Axes (P2)
- Function Manual Expanded Functions; Indexing Axes (T1)

⚠ WARNING

Changed response behavior of the axis/spindle

If several block calls (FC18) 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.

Note

Call note

FC18 must be called cyclically until signal "InPos" or, in the case of an error "Error", produces an edge transition of 1 to 0. An additional "Start" or "Stop" is only possible for this spindle/axis when the "InPos"/"Error" signal has supplied a value of 0. (the system must wait for at least one PLC cycle with the next "Start" or "Stop"). This also applies when the assignment in data byte 8 on the axial interface has been changed.

Abort

The function cannot be aborted by means of parameter "Start" or "Stop", but only by means of the axial interface signals (e.g. delete distance-to-go). The axial interface also returns status signals of the axis that may need to be evaluated (e.g. exact stop, traverse command).

Simultaneity

Several axes can be traversed simultaneously or subject to a delay by FC18 blocks. The upper limit is only limited by the maximum number of axes of the NC.

Axis disable

For a set axis inhibit (DB31, ...DBX1.4 == 1), the axis controlled via FC18 does not move. Only a simulated actual value is generated. Same behavior as when traversing the axis for an axis inhibit by the NC.

Functions

Function 1: Position spindle

Parameter	Meaning
Start :	0 → 1: Start the function
Funct :	1: Function number for "Position spindle"
Mode :	Positioning modes 1, 2, 3, 4 (refer to the paragraph below, "Explanation of the formal parameters"
AxisNo :	Number of machine axis
Pos :	Position
FRate :	FRate ≠ 0: Positioning velocity FRate = 0: Velocity corresponding to MD35300 $MA_SPIND_POSCTRL_VE-LO
InPos :	1: Position reached with "Exact stop fine"
Error :	1: Positioning error
State :	Error code

Function 2: Rotate spindle

Parameter	Meaning
Start :	0 → 1: Start the function
Stop :	0 → 1: Stop the function
Funct :	2: Function number for "Rotate spindle"

Parameter	Meaning
Mode :	Mode = 5: direction of rotation M4 Mode ≠ 5: direction of rotation M3
AxisNo :	Number of machine axis
FRate :	Spindle speed
InPos :	1: Setpoint speed is output, also see DB31, ... DBX83.5 (spindle in the set-point range)
Error :	1: Positioning error
State :	Error code

Function 3: Oscillate spindle

Parameter	Meaning
Start :	0 → 1: Start the function
Stop :	0 → 1: Stop the function
Funct :	3: Function number for "Oscillate spindle"
AxisNo :	Number of machine axis
Pos :	Setpoint gear stage
InPos :	1: Setpoint speed is output, also see DB31, ... DBX83.5 (spindle in the set-point range)
Error :	1: Positioning error
State :	Error code

Parameterized oscillation speed: MD35400 $MA_SPIND_OSCILL_DES_VELO

The function of the parameter "Pos" depends on the setting in MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = <value>

<Value>	Pos	Function
0	0, 1, 2, ... 5	Oscillation
1	0	Oscillation with gear stage change M40
	1	Oscillation with gear stage change M41
	2	Oscillation with gear stage change M42
	3	Oscillation with gear stage change M43
	4	Oscillation with gear stage change M44
	5	Oscillation with gear stage change M45

Function 4: Traverse indexing axes

Note

The modulo conversion can be compared with approaching the indexing position via POS[AX] = CIC (value) in the part program.

Parameter	Meaning
Start :	0 → 1: Start the function
Funct :	4: Function number for "Indexing axis"
Mode :	Positioning mode 0, 1, 2, 3, 4
AxisNo :	Number of machine axis
Pos :	Indexing position
FRate :	FRate ≠ 0: Positioning velocity FRate = 0: Velocity corresponding to MD32060 $MA_POS_AX_VEL
InPos :	1: Position reached with "Exact stop fine"
Error :	1: Positioning error
State :	Error code

Function 5, 6, 7, 8: Position axes

Parameter	Meaning
Start :	0 → 1: Start the function
Funct :	5, 6, 7, 8: Function number for "Position axes"
Mode :	Positioning mode 0, 1, 2, 3, 4
AxisNo :	Number of machine axis
Pos :	Position
FRate :	FRate ≠ 0: Positioning velocity FRate = 0: Velocity corresponding to MD32060 $MA_POS_AX_VELO
InPos :	1: Position reached with "Exact stop fine"
Error :	1: Positioning error
State :	Error code

Function 9: Rotate spindle with automatic gear stage selection:

Parameter	Meaning
Start :	0 → 1: Start the function
Stop :	0 → 1: Stop the function
Funct :	9: Function number for "Rotate spindle with gear stage selection"
Mode :	Mode = 5: direction of rotation M4 Mode ≠ 5: direction of rotation M3
AxisNo :	Number of machine axis
FRate :	Spindle speed
InPos :	1: Setpoint speed is output
Error :	1: Positioning error
State :	Error code

Function 10, 11: Rotate spindle with constant cutting rate

The "Constant cutting rate" function (G96) must be active in the NC.

Parameter	Meaning
Start :	0 → 1: Start the function
Stop :	0 → 1: Stop the function
Funct :	10: Function number for "Constant cutting rate (m/min)" 11: Function number for "Constant cutting rate (feet/min)"
Mode :	Mode = 5: direction of rotation M4 Mode ≠ 5: direction of rotation M3
AxisNo :	Number of machine axis
FRate :	Cutting rate
InPos :	1: Setpoint speed is output
Error :	1: Positioning error
State :	Error code

Declaration of the function

```
FUNCTION FC18: 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

Signal	Typ e	Type	Value range	Meaning
Start:	I	BOOL	0 (FALSE), 1 (TRUE)	0 → 1: Start the function
Stop:	I	BOOL	0 (FALSE), 1 (TRUE)	0 → 1: Stop the function

Signal	Type	Type	Value range	Meaning
Funct:	I	BYTE	1, 2, 3, ... 11	1: Position spindle 2: Rotate spindle 3: Oscillate spindle 4: Indexing axis 5: Positioning axis metric 6: Positioning axis inch 7: PosAxis metric with handwheel override 8: PosAxis inch with handwheel override 9: Rotate spindle with automatic gear stage selection 10: Rotate spindle with constant cutting rate (m/min) 11: Rotate spindle with constant cutting rate (feet/min)
Mode:	I	BYTE	0, 1, 2, ... 5	0: Positioning to absolute position 1: Positioning incremental 2: Positioning along the shortest path 3: Positioning absolute, positive approach direction 4: Positioning absolute, negative approach direction 5: Direction of rotation as for M4
AxisNo:	I	INT	1, 2, 3, ... 31	Number of the 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] Linear axes: [m/min] or [ft/min]
InPos:	O	BOOL	0 (FALSE), 1 (TRUE)	1: Position reached or function executed
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1: Error
State:	O	BYTE	0, 1, 2, ... 255	Error detection

Error identifiers

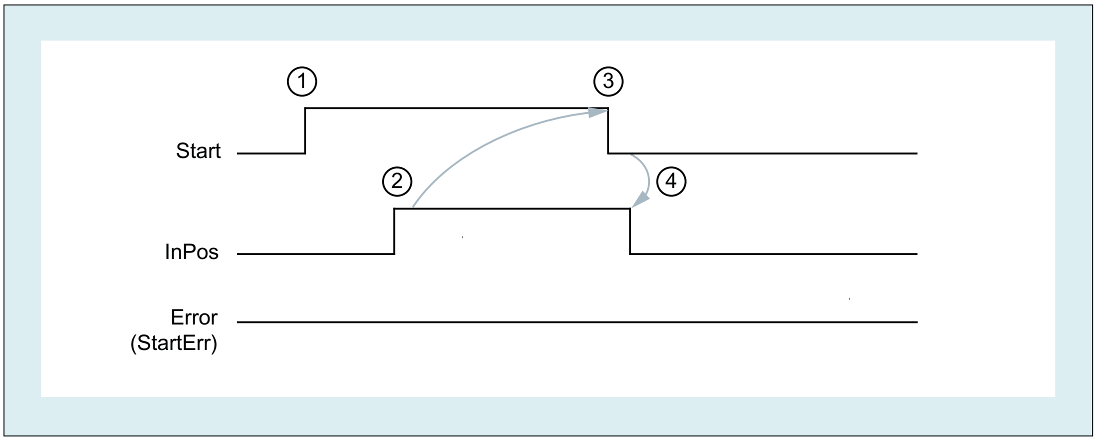
An error is active, if: Parameter "Error" == 1 (TRUE)

The cause of the error is displayed in: Parameter "State"

State	Meaning
	Cause of the error on the PLC side:
1	Several functions of the axis/spindle were activated simultaneously
20	A function was started without the position being reached
30	The axis/spindle was transferred to the NC while still in motion

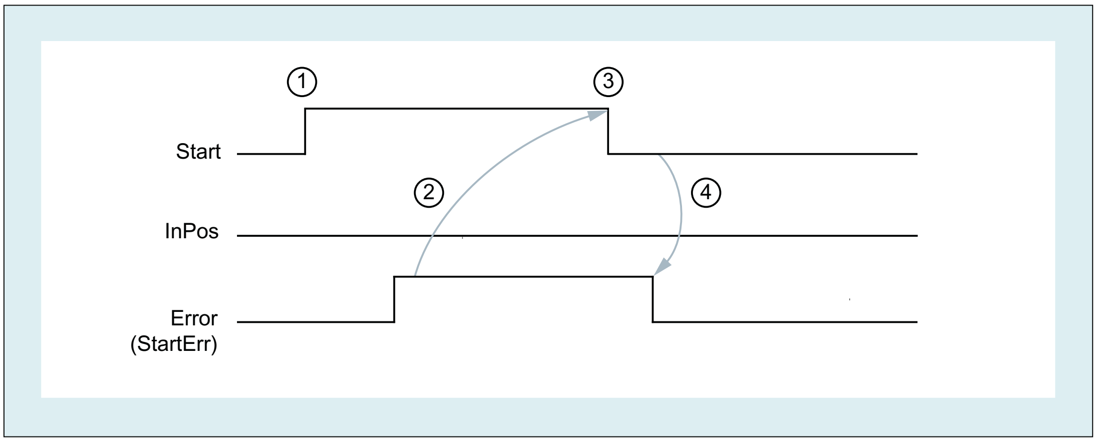
State	Meaning
40	The axis is programmed by the NC program, NC internal error
50	Permanently assigned PLC axis. Traverses (JOG) or references
60	Permanently assigned PLC axis. Channel status does not permit a start
Cause of the error on the NC side	
100	Incorrect position programmed for axis/spindle (corresponds to alarm 16830)
101	Programmed speed is too high
102	Incorrect value range for constant cutting rate (corresponds to alarm 14840)
104	Following spindle: Illegal programming (corresponds to alarm 22030)
105	No measuring system available (corresponds to alarm 16770)
106	Axis positioning still active (corresponds to alarm 22052)
107	Reference mark not found (corresponds to alarm 22051)
108	No transition from speed control to position control (corresponds to alarm 22050)
109	Reference mark not found (corresponds to alarm 22051)
110	Velocity/speed is negative
111	Setpoint speed == 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	Axis is not an indexing axis (corresponds to alarm 20072)
121	Indexing position error (corresponds to alarm 17510)
125	DC (shortest distance) not possible (corresponds to alarm 16800)
126	Absolute value minus not possible (corresponds to alarm 16820)
127	Absolute value plus not possible (corresponds to alarm 16810)
128	No transverse axis available for diameter programming (corresponds to alarm 16510)
130	Software limit switch plus (corresponds to alarm 20070)
131	Software limit switch minus (corresponds to alarm 20070)
132	Working area limit plus (corresponds to alarm 20071)
133	Working area limit minus (corresponds to alarm 20071)
134	Frame not permitted for indexing axis
135	Indexing axis with "Hirth joint" is active (corresponds to alarm 17501)
136	Indexing axis with "Hirth joint" is active and axis not referenced (corresponds to alarm 17503)
137	Spindle operation not possible for transformed spindle/axis (corresponds to alarm 22290)
138	Axis: Coordinate system-specific working area plus violated (corresponds to alarm 20082)
139	Axis: Coordinate system-specific working area minus violated (corresponds to alarm 20082)
System error	
200	corresponds to alarm 450007
Alarm numbers: **References** Diagnostics Manual	

Signal sequence: Normal case



① PLC user program: Function start using a positive edge: 0 → 1
② NC: Positive acknowledgment, function executed / position reached
③ PLC user program: Reset after detecting the positive acknowledgment
④ FC18: Reset of the positive acknowledgment

Signal sequence: Error case



① PLC user program: Function start using a positive edge: 0 → 1
② NC: Negative acknowledgment, error occurred
③ PLC user program: Reset after detecting the negative acknowledgment
④ FC18: Reset of the negative acknowledgment

Call examples

Example 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 E 6.4;                    //Key T12
R M 100.0;                  //Start
//Start with T13
U E 6.3;                    //Key T13
UN M 112.0;                 //Restart only when InPos or Error = 0
UN M 113.0;
S M 100.0;

CALL FC18(
    Start :=  M100.0,
    Stop :=  FALSE,
    Funct :=  B#16#1,       //Position spindle
    Mode :=  B#16#2,        //Shortest path
    AxisNo :=  5,
    Pos :=  MD104,
    FRate :=  MD108,
    InPos :=  M112.0,
    Error :=  M113.0,
    State :=  MB114);
```

Example 2: Start spindle rotation:

```
CALL FC18(
    Start :=  M100.0,
    Stop :=  FALSE,
    Funct :=  B#16#2,       //Rotate spindle
    Mode :=  B#16#5,        //Direction of rotation as for M4
    AxisNo :=  5,
    Pos :=  0.0,
    FRate :=  MD108,
    InPos :=  M112.0,
    Error :=  M113.0,
    State :=  MB114);
```

Example 3: Start spindle oscillation

```
CALL FC18(
    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);
```

Example 4: Traverse indexing axis

```
CALL FC18(
    Start :=  M100.0,
    Stop :=  FALSE,           //Not used
    Funct :=  B#16#4,         //Traverse indexing axis
    Mode :=  B#16#0,          //Absolute positioning
    AxisNo :=  4,
    Pos :=  MD104,            //Default setting in REAL: 1.0;2.0;..
    FRate :=  MD108,
    InPos :=  M112.0,
    Error :=  M113.0,
    State :=  MB114);
```

Example 5: Position axes

```
CALL FC18(
    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);
```

14.17.23 FC19: MCP_IFM - transfer of MCP signals to interface

Function

Block FC19 "MCP_IFM" (M version e.g. MCP 483) is used to transfer data from the machine control panel to the NC/PLC interface:

- Modes
- Axis selections
- WCS/MCS switchover
- Traversing keys

- Overrides
- Keyswitch

The following specifications apply to the **feedrate override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feedrate override**
  - The feedrate override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feedrate override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB5) interface byte if the "Feedrate 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 INC and axis travel keys**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.
  - When the system is switched between MCS and WCS, the active axes are generally deselected.

The LEDs on the machine control panel derived from the selections in the feedback.

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 FC10: AL_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

FC19 or also FC24, FC25, FC26 can be called a multiple number of times in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the mode group number is contained in the lower nibble.

"BAGNo" = 0 or B#16#10 ⇒ mode group signals are not processed.

"ChanNo" = 0 ⇒ no processing of the channel signals.

The INC selections are transferred to the mode group interface. The activation for this specification is done via the DB10.DBX57.0 (INC inputs in BAG area active) through this block once after power up.

Furthermore, two machine control panels can still be handled in parallel by the FC19 block. Whereby, the call of the block for the 2nd machine control panel in OB1 cycle must be set after the call for the 1st machine control panel. Support of two machine control panels exists to a limited extent in the machine control panel blocks. Mutual locking of the axis selections for equally assigned axes for two machine control panels is not supported.

Cartesian manual traversing

The R11 direction key on the machine control panel (at the left next to WCS/MCS) makes the "Manual traversing in tool orientation" function available. This requires activation via the FB1 input parameter "MCP_IF_TCS" in DB7.

For "MCP_IF_TCS" = TRUE, the R11 key switches to "Manual traversing in tool orientation". Whereby, the Z key (R3) is permanently selected with FC19. The direction keys act on the 3rd geometry axis of the associated channel.

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 provide this flexibility, tables for axis numbers are stored in DB10.

For the **first** machine control panel the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the **second** machine control panel starting at byte 32 (symbolic name: MCP2AxisTbl[1..22]). The machine axis numbers must be entered byte-wise here.

It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

For **FC19**, the **maximum possible number of axis selections** can also be restricted. This upper limit is set for the first machine control panel in DB10.DBW30 (symbolic name: MCP1MaxAxis) or for the second machine control panel in DB10.DBW54 (symbolic name: MCP2MaxAxis).

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 for FC19. 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 flip-flop.
- Evaluate the flip-flop output as positive and negative edge.

- For a positive edge write one set of axis numbers in the axis table (DB10) and switch on LED via this key
- For a negative 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 FC19: VOID             //Symbolic name: MCP_IFM

   VAR_INPUT
      BAGNo :          BYTE ;
      ChanNo:          BYTE ;
      SpindleIFNo:     BYTE ;
   END_VAR

   VAR_OUTPUT
      FeedHold :       BOOL;
      SpindleHold :    BOOL;
   END_VAR

BEGIN
END_FUNCTION
```

Description of formal parameters

Signal	Type	Type	Value range	Meaning
BAGNo :	I	BYTE	B#16#00 - B#16#0A	**1st MCP**: Number of mode group to which the mode signals are transferred
			B#16#10 - B#16#1A	**2nd MCP**: Number of mode group to which the mode signals are transferred
ChanNo:	I	BYTE	B#16#00 - B#16#0A	Number of the channel to which the channel signals are transferred
SpindleIFNo:	I	BYTE	0 - 31 (B#16#1F)	Number of the axis/spindle to which the spindle data is transferred (number of the associated machine axis)
FeedHold :	O	BOOL	0 (FALSE), 1 (TRUE)	Feed stop from MCP, modal
SpindleHold :	O	BOOL	0 (FALSE), 1 (TRUE)	Spindle stop from MCP, modal

MCP selection signals to the user interface

Table 14-3 Keyswitch

Source: **MCP - Switch**	**Target:** **Interface DB**
Position 0	DB10.DBX56.4
Position 1	DB10.DBX56.5
Position 2	DB10.DBX56.6
Position 3	DB10.DBX56.7

Table 14-4 Operating modes and machine functions

Source: **MCP - Key**	**Target:** **Interface DB (parameter BAGNo)** **Display for BAG 1**
AUTOMATIC	DB11.DBX0.0
MDI	DB11.DBX0.1
JOG	DB11.DBX0.2
REPOS	DB11.DBX1.1
REF	DB11.DBX1.2
TEACH IN	DB11.DBX1.0
INC 1 ... 10 000, INC Var.	DB11.DBX2.0 - 2.5

Table 14-5 Direction keys rapid traverse override

Source: **MCP - Key**	**Target:** **Interface DB (parameter ChanNo)**
Direction key +	DB21, ...DBX12.7
Direction key -	DB21, ...DBX12.6
Rapid traverse override	DB21, ...DBX12.5
Direction key +	DB21, ...DBX16.7
Direction key -	DB21, ...DBX16.6
Rapid traverse override	DB21, ...DBX16.5
Direction key +	DB21, ...DBX20.7
Direction key -	DB21, ...DBX20.6
Rapid traverse override	DB21, ...DBX20.5

Source: **MCP - Key**	**Target:** **Interface DB (all axis DBs)**
Direction key +	DB31, ...DBX4.7
Direction key -	DB31, ...DBX4.6
Rapid traverse override	DB31, ...DBX4.5

The transfer is dependent upon the selected axis. The associated interface bits are deleted for axes which are not selected.

Table 14-6 Override

Source: **MCP - Switch**	**Target:** **Interface DB (parameter ChanNo)**
Feedrate override	DB21, ...DBB4

Source: **MCP - Switch**	**Target:** **Interface DB (all axis DBs)**
Feedrate override	DB31, ...DBB0 (selected axis number) The feedrate override of the 1st MCP is applied to all axes.
Spindle override	DB31, ...DBB19 (parameter SpindleIFNo)

Table 14-7 Channel signals

Source: **MCP keys**	**Target:** **Interface DB (parameter ChanNo)**
NC start	DB21, ...DBX7.1
NC stop	DB21, ...DBX7.3
RESET	DB21, ...DBX7.7
Single block	DB21, ...DBX0.4

Table 14-8 Feedrate, spindle signals

Source: **MCP keys**	**Target:** **FC output parameters**
Feed stop Feed enable	Parameter: "FeedHold" linked with memory, LEDs are controlled
Spindle stop Spindle enable	Parameter: "SpindleHold" linked with memory, LEDs are controlled

Table 14-9 Cartesian manual traversing

Source: **MCP keys**	**Target:** **Interface DB (parameter ChanNo)**
R11 direction key	DB21, ...DBB392
Direction key +	DB21, ...DBX20.7
Direction key -	DB21, ...DBX20.6

Checkback signals from user interface for controlling displays

Table 14-10 Operating modes and machine functions

Target: **MCP - LED**	**Source:** **Interface DB (parameter BAGNo)** **Display for BAG 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

Target: **MCP - LED**	**Source:** **Interface DB (parameter BAGNo)** **Display for BAG 1**
INC 1 ... 10 000, INC Var.	DB11.DBX8.0 - 8.5

Table 14-11 Channel signals

Target: **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 key.

Call example

```
CALL FC19(     //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 this parameterization, 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, FC19 **needs not be modified or reprogrammed**. The axis number simply has to be entered in axis table DB10.DBB8 and followed as required: The axis number simply has to be entered in axis table DB10.DBB8 and followed as required:

Example

The spindle is defined as the 4th machine axis and should be selected via axis key 9.

```
Solution:
The value 4 must be entered in DB10 byte (8+(9-1)) for the 4th axis.

CALL FC19(           // Signals to interface
   BAGNo :=          B#16#1,     // Mode group no. 1
   ChanNo :=         B#16#1,     // Channel no. 1
   SpindleIFNo :=    B#16#4,     // Spindle Interface Number = 4

   FeedHold :=       m30.0,      // Feed stop signal modal
   SpindleHold :=    m30.1);     //Spindle stop modal
```

14.17.24 FC21: Transfer - data exchange NC/PLC

14.17.24.1 Function

Block FC21 is used to exchange data between the PLC and NC. The data are immediately transferred when FC21 is called – not waiting until the next basic PLC program cycle starts.

The data transfer is activated by calling the block FC21 with parameter "Enable" =1

Functions

The block provides the following functions:

- Synchronized action signals: PLC → NC channel
- Synchronized action signals: NC channel → PLC
- Fast data exchange PLC-NC (read function in NC)
- Fast data exchange PLC-NC (write function in NC)
- Update signals to the NC channel
- Update signals to axes (data byte 2 of the user interface)
- Update signals to axes (data byte 4 of the user interface)

14.17.24.2 Declaration of the function

Declaration of the function

```
VAR_INPUT
          Enable : BOOL;
          Funct : BYTE ;
          S7Var : ANY ;
          IVar1 : INT ;
          IVar2 : INT ;
END_VAR

VAR_OUTPUT
          Error : BOOL;
          ErrCode : INT ;
END_VAR
```

14.17.24.3 Explanation of formal parameters

Explanation of formal parameters

Signal	Type	Type	Value range	Description	
Enable:	I	BOOL	0 (FALSE), 1 (TRUE)	1:	Transferring data
Funct:	I	BYTE	1, 2, 3, ... 7	1:	Synchronized actions at channel
				2:	Synchronized actions from channel
				3:	Read data
				4:	Write data
				5:	Control signals to channel
				6:	Control signals to axis
				7:	Control signals to axis
S7Var :	I	ANY	S7 data storage area	Depends on "Funct"	
IVAR1:	I	INT	---	Depends on "Funct"	
IVAR2:	I	INT	---	Depends on "Funct"	
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1:	An error is active
ErrCode:	O	INT	---	Depends on "Funct"	

14.17.24.4 Function 1, 2: Signals synchronized actions to / from Channel

Synchronized actions can be disabled or enabled by the PLC.

The data area is stored on the user interface in DB21, ...DBB300 ...307 (to channel) and DB21, ...DBB308 ...315 (from channel). The parameter "S7Var" is not evaluated for this function, but must be assigned an actual parameter (see call example). The data are transferred to/from the NC as soon as FC21 is processed.

Signal	Type	Type	Value range	Description	
Enable:	I	BOOL	0 (FALSE), 1 (TRUE)	1:	Transferring data
Funct:	I	BYTE	1, 2	1: 2:	to channel from channel
S7Var :	I	ANY	S7 data storage area	Not used	
IVAR1:	I	INT	1, 2, ... max. channel number	Channel number	
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1:	An error is active
ErrCode:	O	INT	1, 10	1: 10:	"Funct" invalid Channel number invalid

Call example:

```
FUNCTION FC100: VOID
VAR_TEMP
        myAny: ANY ;
END_VAR

BEGIN
NETWORK

//Deactivate synchronized actions with ID3, ID10 and ID31 in NC channel 1 :
SYAK:   AUF   DB21;
        SET;
        S     DBX300. //ID3
              2;
        S     DBX301. //ID10
              1;
        S     DBX303. //ID31
              6;
        L     B#16#1;
        T     MB11;
        SPA   TRAN;

//Synchronized actions from NCK channel 1:
SYVK:   L B#16#2;
        T MB11;
TRAN:   CALL FC21 (
             Enable   := M 10.0,      //if TRUE, FC 21 active
             Funct    := MB 11,
             S7Var    := #myAny,      //Not used
             IVAR1    := 1,           //Channel no.
             IVAR2    := 0,
             Error    := M 10.1,
             ErrCode  := MW 12);
```

```
END_FUNCTION
```

14.17.24.5 Function 3, 4: Fast data exchange PLC-NC

General

A separate, internal data area is provided as interface to allow the fast exchange of data between the NC and the PLC. The interface encompasses 4096 bytes. PLC access operations (reading/writing) are realized via FC21. The internal structure of the interface is solely defined by the user, and must have precisely the same definition on the NC and PLC sides.

This data can be accessed by the NC program using commands $A_DBB[x], $A_DBW[x], $A_DBD[x], $A_DBR[x] (References: List Manual, System Variables).

The concrete address in the data array is specified by a byte offset (0 to 4095) in parameter "IVAR1". In this case, the alignment must be selected according to the data format, i.e. a DWORD starts at a 4byte limit and a WORD at a 2byte limit. Bytes can be located at any offset within the data field. Individual bits cannot be accessed. FC21 converts them over to a byte access. Data type information and quantity of data are taken from the ANY parameter, transferred via S7Var.

Without additional programming-related measures, data consistency is only ensured for 1 and 2 byte access operations - both from the NCU and from the PLC. For 2-byte consistency this is true only for the data type WORD or INT, but not for the data type BYTE.

In the case of longer data types or transfer of arrays which should be transferred consistently, the semaphore byte must be used in parameter "IVAR2", which can be used by FC21 to determine the validity or consistency of a block. This handling must be supported by the NC, i.e. in the NC program, by writing or deleting the semaphore byte. The semaphore byte is stored in the same data field as the user data.

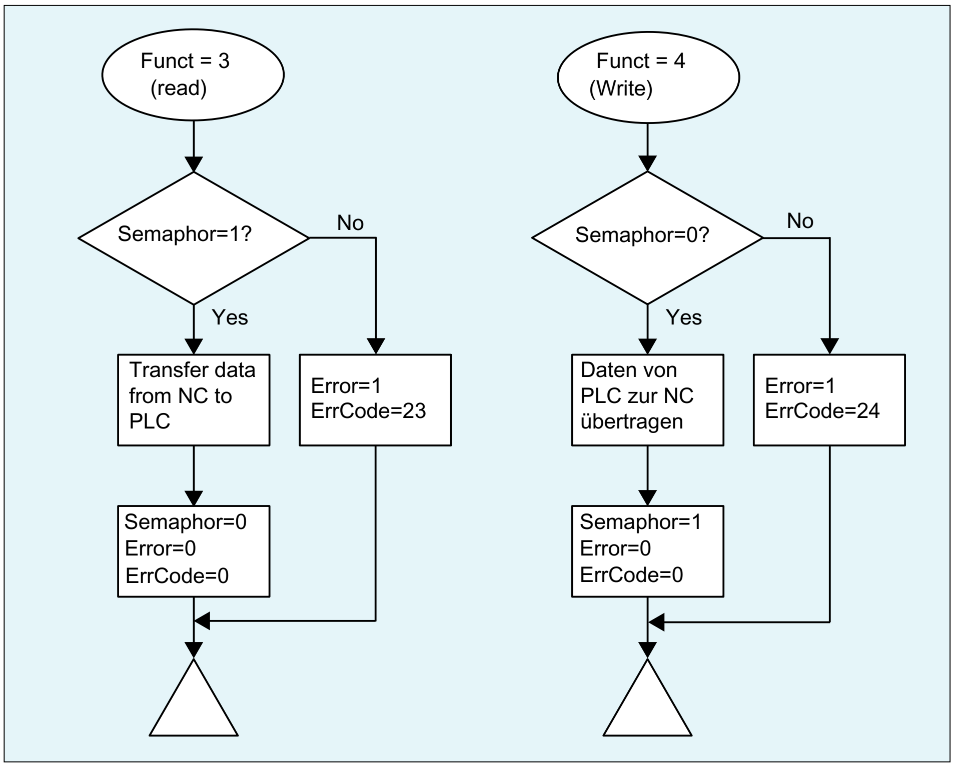
The semaphore byte is identified by a value between 0 and 4095 in "IVAR2".

The PLC reads and writes to the semaphore byte via FC21 in the same call, in which the user data should be transferred. The PLC programmer only has to define the semaphore variable in the interface. For access from the NC via the NC program, the semaphore mechanism must be programmed using individual instructions according to the flow chart shown below. The sequence is different for reading and writing variables.

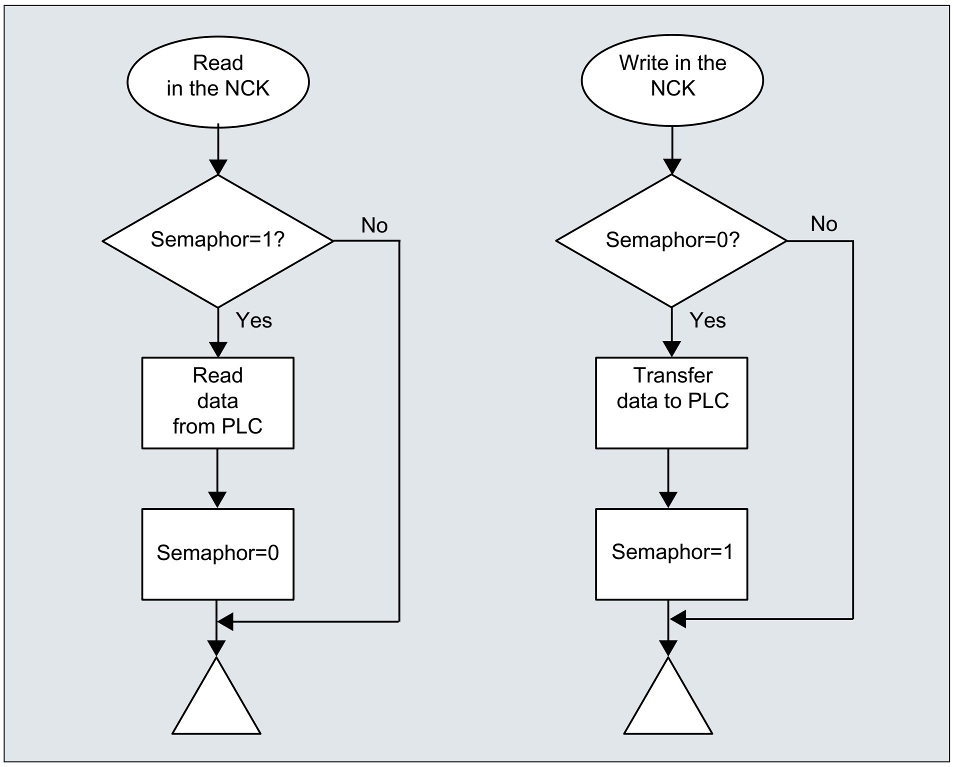
Only individual variables or arrays (fields) can be supported directly using the semaphore technique. Transferring structures must be split up into individual jobs. Here, the user must ensure data consistency of this structure by programming the appropriate semaphore mechanism.

If "IVAR2" = -1 is set, data are transferred without a semaphore.

Data exchange with semaphore in PLC (schematic of FC21)


Funct = 3 (read)
Semaphor=1?
No
Yes
Transfer data from NC to PLC
Error=1 ErrCode=23
Semaphor=0 Error=0 ErrCode=0
Funct = 4 (Write)
Semaphor=0?
No
Yes
Daten von PLC zur NC übertragen
Error=1 ErrCode=24
Semaphor=1 Error=0 ErrCode=0

Basic structure in the NC:

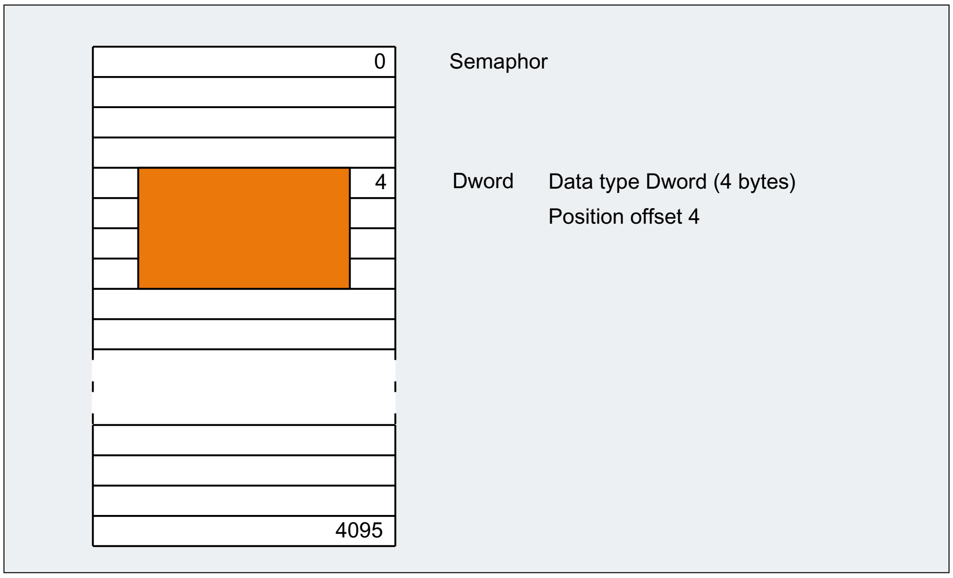


Variable value ranges

Signal	Type	Type	Value range	Description	
Enable:	I	BOOL	0 (FALSE), 1 (TRUE)	1:	Transferring data
Funct:	I	BYTE	3, 4	3:	Read data
				4:	Write data
S7Var :	I	ANY	S7 data area, except local data	Source/destination data storage area	
IVAR1:	I	INT	0 ... 4095	Position offset	
IVAR2:	I	INT	-1 ... 4095	Semaphor byte Transfer without semaphore: -1	
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1:	An error is active
ErrCode:	O	INT	20, 21, 22, 23, 24, 25	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

Example 1: Reading a DWORD from position offset 4 using a semaphore in byte 0 - and saving in memory double word 100

- Data type Dword (4 bytes)
- Position offset 4



PLC programming

```
CALL  FC21(
        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     M10.0;
```

Programming the NC with synchronized actions

- Writing the data to the PLC - byte 0 serves as semaphore:
  ID=1 WHENEVER $A_DBB[0] == 0 DO $A_DBR[4] = $AA_IM[X] $A_DBB[0] = 1
- Reading the data from the PLC - byte 1 serves as semaphore:
  ID=2 WHENEVER $A_DBB[1] == 1 DO $R1 = $A_DBR[12] $A_DBB[1] = 0

Example 2: Reading a WORD from position offset 8 without semaphore, and saving in memory word 104

```
CALL  FC21(
        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);
```

14.17.24.6 Function 5: Update control signals to channel

The purpose of this function is to transmit important control signals at high speed in between cyclic data transfers. Data bytes 6 and 7 of user interfaces DB21, ... are transferred to the NC. The channel is specified in parameter "IVAR1". This enable, for example, the feed disable, read-in disable to be transferred outside of the PLC cycle.

Signal	Type	Type	Value range	Description	
Enable:	I	BOOL	0 (FALSE), 1 (TRUE)	1:	Transferring data
Funct:	I	BYTE	5	5:	Control signals to channel
S7Var :	I	ANY	S7 data storage area	Not used	
IVAR1:	I	INT	1. Maxchannel	Channel number	
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1:	An error is active

Signal	Typ e	Type	Value range	Description	
ErrCode:	O	INT	1, 10	1:	"Funct" invalid
				10:	Channel no. invalid

14.17.24.7 Function 6: Update control signals to axes

The purpose of function 6 is to transmit important control signals at high speed in between cyclic data transfers. The **data byte 2** of application interface DB31, ... is transferred to the NC. The transfer is performed for all activated axes. This allows the controller enable to be transferred outside the PLC cycle, for example.

Signal	Typ e	Type	Value range	Description	
Enable:	I	BOOL	0 (FALSE), 1 (TRUE)	1:	Transferring data
Funct:	I	BYTE	6	6:	Control signals to axes
S7Var :	I	ANY	S7 data storage area	Not used	
IVAR1:	I	INT	0		
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1:	An error is active
ErrCode:	O	INT	1	1:	"Funct" invalid

14.17.24.8 Function 7: Update control signals to axes

The purpose of function 7 is to transmit important control signals at high speed in between cyclic data transfers. The **data byte 4** of application interface DB31, ... is transferred to the NC. The transfer is performed for all activated axes. This enables, for example, the feed stop to be transferred outside the PLC cycle.

Signal	Typ e	Type	Value range	Description	
Enable:	I	BOOL	0 (FALSE), 1 (TRUE)		
Funct:	I	BYTE	7	7:	Control signals to axes
S7Var :	I	ANY	S7 data storage area	Not used	
IVAR1:	I	INT	0		
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1:	An error is active
ErrCode:	O	INT	1	1:	"Funct" invalid

14.17.25 FC22: TM_DIR - direction selection for tool management

NOTICE

Use

Block FC22 "TM_DIR" may only be used in conjunction with the tool management.

Function

Referred to the location numbers, e.g. a tool magazine or revolver (indexing axis) block FC22 "TM_DIR" supplies the shortest path and direction of motion for positioning, based on the target and current position.

Outputs

- Input FC22: "Start" = 1 ⇒ the outputs are cyclically updated.
- Input FC22: "Start" = 0 ⇒ the outputs are undefined.

Special positioning

For direction selection with special positioning (input FC22: "Offset" > 0) a new target position is calculated from the target position, the offset for special positioning as well as the number of magazines locations:

New_target position = (target position - (special position -1)) neg. MODULO number_of_magazine locations

The new target position corresponds to the location number at which the magazine must be positioned so that the target 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.

Call

The block must be called once with the appropriate parameter settings for each magazine.

References

- Further PI services for tool management:
  - FB4: PI_SERV - request PI service (Page 988)
  - FC7: TM_REV - transfer block for tool change with revolver (Page 1049)
  - FC8: TM_TRANS - transfer block for tool management (Page 1052)
- Function Manual, Tool Management

Declaration of the function

```
FUNCTION FC22: 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

Signal	Type	Type	Value range	Description
MagNo:	I	INT	1, 2, 3, ...	Magazine number
ReqPos:	I	INT	1, 2, 3, ...	Target position (magazine location number)
ActPos:	I	INT	1, 2, 3, ...	Actual position (magazine location number)
Offset:	I	BYTE	0, 1, 2, ...	Offset for special positioning
Start:	I	BOOL	0 (FALSE), 1 (TRUE)	1 = start of calculation
Cw:	O	BOOL	0 (FALSE), 1 (TRUE)	1 = Move magazine clockwise
Ccw:	O	BOOL	0 (FALSE), 1 (TRUE)	1 = Move magazine counterclockwise
InPos:	O	BOOL	0 (FALSE), 1 (TRUE)	1 = In position
Diff:	O	INT	0, 1, 2, ...	Absolute value of the differential path (shortest path)
Error:	O	BOOL	0 (FALSE), 1 (TRUE)	1 = error

Call example

```
CALL FC22(                   // Tool management direction selection
                             // Inputs
   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
                             // Outputs
   Cw :=         m 30.0,     // Move magazine clockwise
   Ccw :=        m 30.1,     // Move magazine counterclockwise
   InPos :=      m 30.2,     // Magazine in position
   Diff :=       mw 32,      // Differential path
   Error :=      m30.3       // Error has occurred
   );
```

14.17.26 FC24: MCP_IFM2 - transferring MCP signals to the interface

Function

Block FC24 "MCP_IFM2" (M variant, e.g. MCP 310) is used for transferring data from the machine control panel to the NC/PLC interface:

- Modes
- Axis selections
- WCS/MCS switchover
- Traversing keys
- Overrides or override simulation signals
- Key-operated switch position

The following specifications apply to the **feedrate override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feedrate override:**
  - The feedrate override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feedrate override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB 5) interface byte if the "Feedrate override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.
- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.
  - When the system is switched between MCS and WCS, the active axes are generally deselected.

The associated LEDs on 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 FC10: 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, FC18 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.

FC24 or also FC19, FC25, FC26 can be called a multiple number of times in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).

"BAGNo" = 0 or B#16#10 ⇒ mode group signals are not processed.

"ChanNo" = 0 ⇒ no processing of the channel signals.

The INC selections are transferred to the mode group interface. The activation for this specification is done via the DB10.DBX57.0 (INC inputs in BAG area active) through this block once after power up.

Furthermore, two machine control panels can be handled in parallel by this block. Whereby, the call of the block for the 2nd machine control panel in OB1 cycle must be set 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).

Key-operated switch position

As of software version 4.5 SP2, the key-operated switch signals in the FC24 are also transferred to the user interface (DBX56.5 to 7). This transfer is made independent of whether a keyswitch is mounted on the MCP.

Note

For further information see "FC19: MCP_IFM - transfer of MCP signals to interface (Page 1081) ".

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 provide this flexibility, tables for axis numbers are stored in DB10.
For the **1st** machine control panel (MCP), the table starts from byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the **2nd** machine control panel (MCP), the table starts from byte 32 (symbolic name: MCP2AxisTbl[1..22]). The machine axis numbers must be entered byte by byte here. It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

For **FC24**, the **maximum possible number of axis selections** can also be restricted.
This upper limit is set for the 1st machine control panel in DB10.DBW30 (symbolic name: MCP1MaxAxis) or for the 2nd machine control panel in DB10.DBW54 (symbolic name: MCP2MaxAxis) for the respective MCP.

The default setting is 0, corresponding to the maximum number of configured axes. The axis numbers and the limit can also be adapted dynamically. Afterwards, a new axis must be selected for FC24. Axis numbers may not be switched over while the axes are traversing the relevant direction keys. The compatibility mode is preset with axis numbers **1 to 6** for both MCPs and restricted to the configured number of axes.

Declaration of the function

```
FUNCTION FC24: 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

Signal	Typ e	Type	Value range	Meaning	
BAGNo :	I	BYTE	B#16#00 - B#16#0A	**1st MCP**: Number of mode group to which the mode signals are transferred	
			B#16#10 - B#16#1A	**2nd MCP**: Number of mode group to which the mode signals are transferred	
ChanNo:	I	BYTE	B#16#00 - B#16#0A	Number of the channel to which the channel signals are transferred	
SpindleIFNo:	I	BYTE	0 - 31 (B#16#1F)	Number of the axis/spindle to which the spindle data is transferred (number of the associated machine axis)	
FeedHold :	O	BOOL	0 (FALSE), 1 (TRUE)	Feed stop from MCP, modal	
SpindleHold :	O	BOOL	0 (FALSE), 1 (TRUE)	Spindle stop from MCP, modal	
SpindleDir:	O	BOOL	0 (FALSE), 1 (TRUE)	Direction of spindle rotation	
				0:	Direction of rotation + (left)
				1:	Direction of rotation - (right)

Call example

```
CALL FC24(                         // Slimline machine control panel M variant
                                   // 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 this parameterization, 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 FC18 call.

14.17.27 FC25: MCP_IFT - transfer of MCP/OP signals to interface

Function

Block FC25 "MCP_IFT" (T variant, e.g. MCP 483) is used for transferring data from the machine control panel to the NC/PLC interface:

- Modes
- Direction keys of four axes
- WCS/MCS switchover
- Overrides
- Keyswitch

The following specifications apply to the **feedrate override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feedrate override:**
  - The feedrate override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feedrate override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB 5) interface byte if the "Feedrate override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.
- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.

The associated LEDs on the machine control panel derived from the acknowledgments of the relevant selections.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g. using the appropriate input signals in FC10: AL_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

FC25 or also FC19, FC24, FC26 can be called a multiple number of times in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).

"BAGNo" = 0 or B#16#10 ⇒ mode group signals are not processed.

"ChanNo" = 0 ⇒ no processing of the channel signals.

Flexible axis configuration

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Support is now provided by the MCP blocks for the use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups. The block call for the second machine control panel in OB1 cycle must be set 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 provide this flexibility, tables for axis numbers are stored in DB10.
For the **1st** machine control panel (MCP), the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the **2nd** machine control panel (MCP), from byte 32 (symbolic name: MCP2AxisTbl[1..22]). The machine axis numbers are entered here byte-by-byte. It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

The restriction of the **possible number of axes at FC25** is realized using the 0-values in the axis table. The axis numbers can also be adapted dynamically. During the manual traversing of axes using the direction keys, the axis numbers must not be switched over. The compatibility mode is preset with axis numbers **1 to 4** for both MCPs and restricted to the configured number of axes.

Note

For further information see "FC19: MCP_IFM - transfer of MCP signals to interface (Page 1081) ".

Declaration of the function

```
FUNCTION FC25: 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

Signal	Type	Type	Value range	Meaning
BAGNo :	I	BYTE	B#16#00 - B#16#0A	**1st MCP**: Number of mode group to which the mode signals are transferred
			B#16#10 - B#16#1A	**2nd MCP**: Number of mode group to which the mode signals are transferred
ChanNo:	I	BYTE	B#16#00 - B#16#0A	Number of the channel to which the channel signals are sent
SpindleIFNo:	I	BYTE	B#16#00 - B#16#1F	Number of the axis/spindle to which the spindle data is transferred (number of the associated machine axis)
FeedHold :	O	BOOL	0 (FALSE), 1 (TRUE)	Feed stop from MCP, modal
SpindleHold :	O	BOOL	0 (FALSE), 1 (TRUE)	Spindle stop from MCP, modal

Call example

With this parameter assignment example, the signals are sent to the 1st mode group, the 1st channel and all axes. The spindle override is transferred to the 4th axis/spindle. Feed stop is sent to bit memory 22.0 and spindle stop is sent to the data block DB2, DBX151.0.

```
CALL FC25(                        //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
```

14.17.28 FC26: HPU_MCP - transfer of HT 8 signals to the interface

Declaration of the function

```
FUNCTION FC26: VOID
// NAME:         HPU_MCP

VAR_INPUT
    BAGNo :      BYTE ;
    ChanNo:      BYTE ;
END_VAR

BEGIN
END_FUNCTION
```

Description of formal parameters

Signal	Type	Type	Value range	Meaning
BAGNo :	I	BYTE	B#16#00 - B#16#0A	**1st MCP**: Number of mode group to which the mode signals are transferred
			B#16#10 - B#16#1A	**2nd MCP**: Number of mode group to which the mode signals are transferred
ChanNo:	I	BYTE	B#16#00 - B#16#0A	Number of the channel to which the channel signals are transferred

Call examples

Calling FC26 for the first MCP, the first mode group and the first channel of the NC.

```
CALL FC26(                         //Machine control panel of the HT 8
           BAGNo :=    B#16#01,    //1.MCP, 1.BAG
           ChanNo :=   B#16#01);   //Channel 1
```

Calling FC26 for the second MCP, the second mode group and the third channel of the NC.

```
CALL FC26(                         //Machine control panel of the HT 8
           BAGNo :=    B#16#12,    //2.MCP, 2.BAG
           ChanNo :=   B#16#03 );  //Channel 3
```

General description of functions

The function FC26 "HPU_MCP (machine control panel signals of the handheld unit HT 8)" transfers the HT 8-specific signals of the following functions between the HT 8-input/output data areas parameterized in function block FB1 (Parameter: MCPxIn and MCPxOut) and the NC/PLC-interface:

- Modes
- Machine function INC
- Coordinate system WCS or MCS
- Axial traverse key
- Axis selection
- Feedrate override
- Rapid traverse override
- Keyswitch information

Note

Mode switchover through HT 8 and/or HMI

The function FC2 "GP_HP Basic program, cyclic part" transfers the signals of the mode switchover in such a way that an alternative selection from MCP of HT 8 and of the HMI is possible. The transfer of the HMI signals to the NC/PLC interface can also be deactivated in the function block FB1 with the parameter "MMCToIF" = FALSE .

Active axes

Using HT 8 a maximum of 6 axes can be addressed at the same time. The selection of the axes is to be realized by the user/machine manufacturer in the PLC user program.

Flexible axis configuration

The function FC26 enables a flexible assignment of the machine axes to the traversing keys or to the axis selection. 2 tables are available in DB10 for this purpose:

- Machine axis table, 1st MCP: DB10.DBB8 to DBB13 (Table of the machine axis number)
  Symbolic name: MCP1AxisTbl[1..22]
- Machine axis table, 2nd MCP: DB10.DBB32 to DBB37 (Table of the machine axis number)
  Symbolic name: MCP2AxisTbl[1..22]

In the tables the axis numbers n (with n = 1, 2, ...) of the active machine axis are to be entered byte-wise. The value 0 must be entered in the unused table locations.

The table length can be specified to the FC26:

- 1st MCP: DB10.DBB30 (upper limit of the machine axis table)
- 2nd MCP: DB10.DBB54 (upper limit of the machine axis table)

A value of 4, for example, means that FC26 takes into account only the first 4 table entries or machine axes. The maximum value for the FC26 is 6. For a value of 0 or values greater than 6 the maximum value is taken implicitly.

Note

Please note the following constraints:

- A check of the permissible machine axis numbers is not done. Invalid machine axis numbers can lead to a PLC stop.
- The machine axis numbers can be changed dynamically. The table may not be written, if currently a machine axis is being moved via a traversing key.

Transfer of the traversing key signals depending upon the active coordinate system

The traversing key signals for 6 axes lie in the HT 8 input data area below:

- EB n + 2, Bit 0 - Bit 5 (positive traversing direction)
- EB n + 3, Bit 0 - Bit 5 (negative traversing direction)

The switchover of the coordinate system is done via the input signal:

- EB n + 0, Bit 0 (MCS/WCS)

The input signal is evaluated in FC26 with the help of the edge trigger flag. The active coordinate system is shown in the following output signal:

- AB n + 0, Bit 0 (MCS/WCKS) with 0 = MCS, 1 = WCS

In case of active MCS the traversing key signals of the axes 1 - 6 are transferred in the axis-specific interfaces (DB31, ...DBX4.6 and DBX4.7 (traversing key +/-)) of the axes specified in the machine axis tables (DB10.DBB8 to DBB13 or DBB32 to DBB37).

In case of active WCS it is assumed that the axes 1 - 3 of the machine axis table are geometric axes. For this reason the traversing key signals:

- Of the axes **1** - **3** (EB n + 2 / 3, Bit 0 - Bit 2) are transferred in the interface of the geometric axes in DB21, ...DBB12 + (n * 4), with n = 0, 1, 2), bit 6 and bit 7 (traversing keys +/-) of the channel specified with the parameter "ChanNo" .
  The assignment of the traversing key signals of axes 1, 2 and 3 to the geometric axes 1, 2 and 3 of the channel is permanent and may not be changed.
- Of the axes **4** - **6** (EB n + 2 / 3, Bit 3 - Bit 5) are transferred in the axis-specific interface (DB31, ...DBX4.6 and DBX4.7 (traversing keys +/-)) of the axes 4 - 6 entered in the machine axis table (DB10.DBB**11** to DBB**13** or DBB**35** to DBB**37**).

No traversing of machine axes in WCS

In case of active WCS (AB n + 0, Bit 0 = 1) the traversing of the machine axes can be locked. For this, the following output signals are to be set in the PLC user program:

- AB n + 3, Bit 7 = 1 (For WCS: no machine axes)
  Requirement to the FC26, not to transfer any traversing key signals for the machine axes. The traversing key signals for the axes 1 - 3 of the machine axis table are transferred to the geometric axes 1 - 3 of the specified channel. The traversing key signals for the axes 4 - 6 of the machine axis table are not transferred.
- AB n + 2, Bit 6 (axes 7 - n selected)
  Requirement to the FC26 not to transfer any traversing key signals, since the axes 1 - 6 of the machine axis table are switched over. The axes 1 - 3 are thus not geometric axes, but instead also machine axes.

Feedrate override

The value of the HT 8 override switch is transferred as feedrate override in the channel-specific interface DB21, ...DBB4 (feedrate override) of the programmed channel (parameter: "ChanNo") and in the axis-specific interfaces DB31, ...DBB0 (feedrate override) of the axes programmed in the table DB10.DBB8 to DBB13 (machine axis number).

Rapid traverse override

If for the programmed channel (parameter: "ChanNo") the signal DB21, ...DBX25.3 = 1 (feedrate override for rapid traverse) is set, the value of the HT 8 override switch is sent as a rapid traverse override to its channel-specific interface in DB21, ...DBB5 (rapid traverse override) and the signal DB21, ...DBX6.6 = 1 (rapid traverse override active) is also set.

Machine function INC

The HT 8 signals of the machine function INC are transferred differently depending upon the active coordinate system MCS or WCS:

- Active coordinate system: MCS
  The selected machine function INC is transferred for all 6 axes in the axis-specific interfaces in DB31, ...DBX5.0 to DBX5.5 (machine function) of the axes programmed in the table in DB10.DBB8 to DBB13 (machine axis numbers) .
- Active coordinate system: WCS
  For the axes 1 to 3 the signals of the machine function INC are transferred in the channel-specific. Interface in DB21, ...DBX13.0 to DBX13.5 (machine function) of the programmed channel (Parameter: "ChanNo").
  For the axes 4 to 6 the signals of the machine function INC are transferred in the channel-specific. interfaces in DB31, ...DBX5.0 to DBX5.5 (machine function) of the axes programmed in the table in DB10.DBB11 to DBB13 (machine axis numbers).

The selection signals of the INC machine functions are transferred in the mode group-specific interface DB11.DBB 2 + (n * 20), bit 0 to bit 5 (with n = 0, 1, 2, ...). The FC26 informs the NC about the activation of the mode group interface for the INC machine functions once after power-up with DB10.DBX57.0 (INC inputs active in the mode group area).

Handwheel selection

The hand-wheel selection signals are evaluated by HMI and transferred to the corresponding NC/PLC interface signals of the machine or geometric axes:

- Geometry axes: DB21, ... DBB 12 + (n * 4), bit 0 to bit 2 (with n = 0, 1, 2)
- Machine axes: DB31, ...DBX4.0 to DBX4.2

Requirement: FB1 parameters: "HWheelMMC" = TRUE

Multiple call in one PLC cycle

FC26 can be called a multiple number of times in a single PLC cycle. Upon the first call in the PLC cycle:

- All actions of the parameterized blocks are executed
- The LED signals are written in the output area
- In case of selected WCS, the traversing key signals of the geometric axes are written
- The signals for the selection and deselection of the individual block are processed

When FC26 is called more often, then the channel and mode group interface are processed at a reduced rate.

Processing of two MCP

If the function FC26 is called twice for two MCP in the cyclic sequence of the PLC program (organization block OB1), the call for the second MCP must be made after the call for the first MCP.

Note

If an axis can be traversed from two MCP, then the implementation of a mutual interlocking is the responsibility of the user (machine manufacturer).

Failure of the MCP of the HT 8

In case of failure of the MCP of the HT 8 all the input signals are set to the value 0.

14.17.28.1 Overview of the NC/PLC interface signals of HT 8

Operating modes and machine functions

Source: MCP	Destination: Programmed mode group (Parameter BAGNo) Display for BAG 1
AUTOMATIC	DB11.DBX0.0
MDI	DB11.DBX0.1
JOG	DB11.DBX0.2

Source: MCP	Destination: Programmed mode group (Parameter BAGNo) Display for BAG 1
REPOS	DB11.DBX1.1
REF	DB11.DBX1.2
TEACH IN	DB11.DBX1.0
INC 1 ... 10 000, INC Var.	DB11.DBX2.0 - DBX 2.5

Traversing keys and rapid traverse override

Source: MCP	Destination: Geometry axis of the prog. channel (Parameter: ChanNo)
Traversing key +	DB21, ...DBX12.7
Traversing key -	DB21, ...DBX12.6
Rapid traverse override	DB21, ...DBX12.5
Traversing key +	DB21, ...DBX16.7
Traversing key -	DB21, ...DBX16.6
Rapid traverse override	DB21, ...DBX16.5
Traversing key +	DB21, ...DBX20.7
Traversing key -	DB21, ...DBX20.6
Rapid traverse override	DB21, ...DBX20.5

Source: MCP	Destination: Prog. axes corresponding to the table in DB10, DBB8 - 13 (1. MCP) or DBB32 - 37 (2. MCP)
Traversing key +	DB31, ...DBX4.7
Traversing key -	DB31, ...DBX4.6
Rapid traverse override	DB31, ...DBX4.5

Override

Source: MCP	Destination: Programmed channel (Parameter: ChanNo)
Feed override	DB21, ...DBB4

Source: MCP	Destination: Prog. axes corresponding to the table in DB10, DBB8 - 13 (1. MCP) or DBB32 - 37 (2. MCP)
Feed override	DB31, ...DBB0

Channel signals

Source: MCP	Destination: Programmed channel (Parameter: ChanNo)
NC start	DB21, ...DBX7.1
NC stop	DB21, ...DBX7.3

Source: MCP	Destination: Programmed channel (Parameter: ChanNo)
RESET	DB21, ...DBX7.7
Single BLock	DB21, ...DBX0.4

14.17.28.2 Overview of the NC/PLC interface signals of HT 8

Operating modes and machine functions

Destination: MCP	Source: Interface-DB (parameter BAGNo) Display for BAG 1
AUTOMATIC	DB11.DBX6.0
MDI	DB11.DBX6.1
JOG	DB11.DBX6.2
REPOS	DB11.DBX7.1
REF	DB11.DBX7.2
TEACH IN	DB11.DBX7.0

14.17.29 FC1005: AG_SEND - transfers data to Ethernet CP

Function

The AG_SEND function block transfers the data to be transferred via a configured connection to the Ethernet CP. Together with the AG_RECV function block, data exchange can be established with another station via the integrated "CP 840D sl." This station must be configured in STEP 7, "NetPro."

In the basic program, this function is available as a function block FC1005. This is roughly equivalent to function block FC5 in the "SIMATIC_NET_CP" library.

The TCP, UDP, and ISO-on-TCP protocols are supported.

Note

Smaller volume of transmittable data with SINUMERIK CP and UDP or ISO-on-TCP protocol

For SINUMERIK CP, only **240** bytes can be transmitted when function block FC1005 and the UDP or ISO-on-TCP protocol are used.

Description of formal parameters

Signal	Type	Type	Value range	Description
ACT:	I	BOOL	0 (FALSE), 1 (TRUE)	Job initiation [1)]
ID:	I	INT	1, 2, 3 ...16	Connection ID
LADDR:	I	WORD	-	Module start address [2)]

Signal	Type	Type	Value range	Description
SEND:	I	ANY	-	Specifies the address and length. The address of the data area alternatively refers to: • Bit memory address area • Data block area
LEN:	I	INT	TCP: 1, 2, ... 8192 UDP: 1, 2, ... **240** ISO-on-TCP: 1, 2. ... **240**	Number of bytes that are transmitted
DONE:	O	BOOL	0 (FALSE), 1 (TRUE)	0: Job running 1: Job executed
ERROR:	O	BOOL	0 (FALSE), 1 (TRUE)	0: No error 1: An error is active
STATUS:	O	WORD	-	Status display
1) Parameter ACT must be TRUE long enough until the following applies: (DONE == 1) OR (ERROR == 1) 2) For SINUMERIK 840D sl: Parameter LADDR := **W#16#8110**				

Documentation

A detailed module description can be found in:

- SINUMERIK user interfaces: Online help
- SIMATIC Programming Manual: Program blocks for SIMATIC NET S7-CPs
  Chapter: "Program blocks for Industrial Ethernet" > "Program blocks for open communication services (SEND/RECEIVE interface)" > "AG_SEND / AG_LSEND / AG_SSEND"

14.17.30 FC1006: AG_RECV - receives data from the Ethernet CP

Function

The function block AG_RECV receives data transferred via a configured connection from the Ethernet CP. Together with the AG_SEND function block, data exchange can be established with another station via the integrated "CP 840D sl." This station must be configured in STEP 7, "NetPro."

In the basic program, this function is available as a function block FC1006. This is roughly equivalent to function block FC6 in the "SIMATIC_NET_CP" library.

The TCP, UDP, and ISO-on-TCP protocols are supported.

Note

Smaller volume of transmittable data with SINUMERIK CP and UDP or ISO-on-TCP protocol

For SINUMERIK CP, only **240** bytes can be transmitted when function block FC1006 and the UDP or ISO-on-TCP protocol are used.

Formal parameters

Signal	Typ e	Type	Value range	Description
ID:	I	INT	1, 2 ...16	Connection ID
LADDR:	I	WORD	-	Module start address 1)
RECV:	I	ANY	-	Specifies the address and length. The address of the data area alternatively refers to: • Bit memory address area • Data block area
NDR :	O	BOOL	0 (FALSE), 1 (TRUE)	0: Job running 1: New data accepted
ERROR:	O	BOOL	0 (FALSE), 1 (TRUE)	0: No error 1: An error is active
STATUS:	O	WORD	-	Status display
LEN:	O	INT	TCP: 1, 2, ... 8192 UDP: 1, 2, ... **240** ISO-on-TCP: 1, 2. ... **240**	Number of bytes that are transmitted
1) For SINUMERIK 840D sl: Parameter LADDR := **W#16#8110**.				

Documentation

A detailed module description can be found in:

- SINUMERIK user interfaces: Online help
- SIMATIC Programming Manual: Program blocks for SIMATIC NET S7-CPs
  Chapter: "Program blocks for Industrial Ethernet" > "Program blocks for open communication services (SEND/RECEIVE interface)" > "AG_RECV / AG_LRECV / AG_SRECV"

14.18 Signal/data descriptions

14.18.1 Interface signals NCK/PLC, HMI/PLC, MCP/PLC

References

A complete overview of all interface signals can be found in:

"NC Variables and Interface Signals" List Manual

The NC interface signals that are evaluated by the basic PLC program and transferred in conditioned form to the user interface are presented in the following sections.

14.18.2 Decoded M signals

The M functions programmed in the part program, ASUP 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
- ...

DB21, ...	Meaning
DBX194.0 ... 7	M functions M0 ... M7
DBX195.0 ... 7	M functions M8 ... M15
DBX196.0 ... 7	M functions M16 ... M23
DBX197.0 ... 7	M functions M24 ... M31
DBX198.0 ... 7	M functions M32 ... M39
DBX199.0 ... 7	M functions M40 ... M47
DBX200.0 ... 7	M functions M48 ... M55
DBX201.0 ... 7	M functions M56 ... M63
DBX202.0 ... 7	M functions M64 ... M71
DBX203.0 ... 7	M functions M72 ... M79
DBX204.0 ... 7	M functions M80 ... M87
DBX205.0 ... 7	M functions M88 ... M95
DBX206.0 ... 3	M functions M96 ... M99

The signal for the associated M function remains pending for at least one PLC cycle in the NC/PLC interface.

Note

The following spindle-specific M functions are not transferred to the NC/PLC interface: M3, M4, M5 and M70.

M02/M30 part program end

The output of the M02/M30 M functions does not inform reliably that the part program has completed. The output of the M02/M30 M functions could arise from an asynchronous subprogram (ASUP) or a synchronized action that has nothing to do with the real end of the part program in this case. To detect reliably the end of a part program in the channel, the following interface signal must be evaluated:

DB21, ...DBX35.7 (channel state: Reset)

14.18.3 G commands

The G commands programmed in the part program, ASUP or synchronized actions are transferred channel-specific from the NC to the PLC:

- G commands from channel 1: DB21
- G commands from channel 2: DB22
- ...

DB21, ...	Meaning
DBB208	Active G command of G group 1
DBB209	Active G command of G group 2
DBB210	Active G command of G group 3
...	...
DBB271	Active G command of G group 64

The associated G command remains pending for at least one PLC cycle in the NC/PLC interface.

Address in DB21, ...

The DBBx address of a G group in DB21, ... is calculated as:

Byte number = 208 + "number of G groups" - 1

Basic setting after Power On

After Power On, the value zero, i.e. active G group undefined, is transferred for all G groups.

Part program end or abort

After part program end or abort, the last active G command is retained.

NC START

After NC START the values of the eight G groups specified in the following machine data are overwritten in accordance with the default setting set via the machine data as well as the values programmed in the part program:
MD22510 $MC_GCODE_GROUPS_TO_PLC

References

An overview of all G commands can be found in:

Programming Manual, Fundamentals; Section: "Tables" > "G commands"

14.18.4 Message signals in DB 2

DB2 allows the messages for individual signals to be output on the operator panel front. The signals are subdivided 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:

- List Manual, "NC Variable and Interface Signals"; Section "Interface signals - Overview" > "PLC alarms/messages"
- Commissioning Manual; Chapter "Alarm and message texts"

Note

The number of user areas can be parameterized using FB1.

After the configuration has been modified (FB1: MsgUser) DB2/DB3 must be deleted.

14.19 Notes on programming in STEP 7

In the following chapters, instructions are given for simplifying the programming of complex processes and functions in STEP 7:

- Copying data (Page 1115)
- Data types ANY and POINTER (Page 1116)
- Multi-instance DB (Page 1121)
- Strings (Page 1122)
- Determination of offset addresses on data block structures (Page 1123)
- FB calls (Page 1123)

References

You will find basic information on the structure of data types ANY and POINTER in:

SIMATIC STEP 7 Manual; Chapter: "Designing user programs" > "Register of CPU and saving of data"

14.19.1 Copying data

Copying variants

For the high-speed copying of data from one DB into another it is recommended

- **for larger data quantities** to use the system function SFC BLKMOV or SFC FILL, because here a high-speed copying takes place.
- the routine given below is **for smaller data quantities**, because the supply of ANY parameter to the SFCs consumes additional time.

Example

```
                                        // DB xx.[AR1] is the source
                                        // DI yy.[AR2] is the destination
AUF        DB 100;                      //Source DB
LAR1       P#20.0;                      //Source start address on data byte 20
AUF        DI 101;                      //Destination DB
LAR2       P#50.0;                      //Destination start address on data byte 50
                                        //AR1, AR2, DB, DI loaded beforehand
L          4;                           //Transfer 8 bytes
M001:
L          DBW [AR1,P#0.0];             //Copy word-oriented
T          DIW [AR2,P#0.0];
+AR1       P#2.0;
+AR2       P#2.0;
TAK;
LOOP       M001;
```

14.19.2 ANY and POINTER

The following programming examples show the programming mechanism. They demonstrate how input/output and transit variables (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT) are accessed by data types "POINTER" or "ANY" within an FC or FB. The access operations are described in such a way that a part symbolic method of programming can be used.

14.19.2.1 Use of POINTER and ANY in FC

Function

FC99 has input parameters that are defined as POINTER or ANY.

The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1, Thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example

```
FUNCTION FC99: VOID
VAR_INPUT
   Row : BYTE ;
   Convert : BOOL;                    //Activate numerical conversion
   Addr: POINTER;                     //Points to variable
   Addr1 : ANY ;
END_VAR
VAR_TEMP
   dbchr : WORD ;
   Number: WORD ;
   type : BYTE ;
END_VAR
BEGIN
NETWORK
TITLE =
                                      //POINTER
   L            P##Addr;
   LAR1 ;                             //Retrieve pointer
   L            W [AR1,P#0.0];        //Retrieve DB number
   T            #dbchr;
   L            D [AR1,P#2.0];        //Offset part of pointer
LAR1 ;
AUF DB [#dbchr];                      //Open DB of variables
L B [AR1,P#40.0];                     //Retrieve byte value using pointer with
                                      //address offset 40
                                      //ANY
   L            P##Addr1;
   LAR1 ;                             //Retrieve ANY
   L            B [AR1,P#1.0];        //Retrieve type
   T            #typ;
   L            W [AR1,P#2.0];        //Retrieve amount
   T            #Amount;
   L            W [AR1,P#4.0];        //Retrieve DB number
   T            #dbchr;
   L            D [AR1,P#6.0];        //Offset part of pointer
LAR1 ;
   AUF          DB [#dbchr];          //Open DB of variables
   L            B [AR1,P#0.0];        //Retrieve byte value using ANY
```

14.19.2.2 Use of POINTER and ANY in FB

Function

FB99 has input parameters that are defined as POINTER or ANY.

The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1, thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example

```
FUNCTIONBLOCK FB99
VAR_INPUT
   Row : BYTE ;
   Convert : BOOL;                      //Activate numerical conversion
   Addr: POINTER;                       //Points to variable
   Addr1 : ANY ;
END_VAR
VAR_TEMP
   dbchr : WORD ;
   Number: WORD ;
   type : BYTE ;
END_VAR
BEGIN
NETWORK
TITLE =
                                        //POINTER
   L              P##Addr;
   LAR1 ;                               //Retrieve pointer from instance DB
   L              DIW [AR1,P#0.0];      //Retrieve DB number
   T              #dbchr;
   L              DID [AR1,P#2.0];      //Offset part of pointer
   LAR1 ;
   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 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 ;
   AUF             DB [#dbchr];          //Open DB of variables
   L               B [AR1,P#0.0];        //Retrieve byte value using ANY
```

14.19.2.3 POINTER or ANY variable for transfer to FC or FB

POINTER or ANY variable

With version 1 or later of STEP 7 it is possible to define a pointer or ANY in VAR_TEMP.

The following two examples show how an ANY can be supplied.

Example 1: Transfer ANY parameter via a selection list to another FB (FC)

Several ANY parameters are defined in an FB (FC). A specific ANY parameter must now be chosen from a selection list for transfer to another FB (FC). This can only be done by means of an ANY in VAR_TEMP. 1 to 4 can be set in parameter "WhichAny" in order to select Addr1 to Addr4.

Note

Address register AR2 is used in the block. However, this address register AR2 is also used for multiinstance DBs. For this reason, this FB should **not** be declared as multi-instance DB.

```
FUNCTIONBLOCK FB100
CODE_VERSION1                     //starting from STEP 7 Version 2 for
                                  deactivating the
                                  //multi-instance DB
VAR_INPUT
WhichAny : INT;
   Addr1 : ANY ;                  //Observe predetermined order
   Addr2 : ANY ;
   Addr3 : ANY ;
   Addr4 : ANY ;
END_VAR
VAR_TEMP
   dbchr : WORD ;
   Number: WORD ;
   type : BYTE ;
   Temp_addr : ANY ;
END_VAR
BEGIN
NETWORK
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 FB101, DB100
         (ANYPAR := #Temp_addr);  //ANYPAR is data type ANY
```

Example 2: Transfer an ANY parameter constructed earlier to another FB (FC)

An ANY parameter that has already been compiled must be transferred to another FB (FC). This can be done only by means of an ANY stored in VAR_TEMP.

```
FUNCTIONBLOCK FB100
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 FB101, DB100
         (ANYPAR := #Temp_addr);  //ANYPAR is data type ANY
```

14.19.3 Multiinstance DB

Function

From Version 2 in STEP 7, you can provide multi-instance enabled FBs, i.e. with multi-instance DBs. The primary characteristic of multiinstance DBs is that a data module can be used for various instances of FBs (see STEP 7 documentation), The quantity structure of the DBs can be optimized this way.

Multi-instance DBs should be activated only when they are actually going to be used since they increase the runtime and code size of the FBs.

Note

When complex programs are implemented in multiinstance enabled FBs that use a pointer and address register, it is important for the programmer to observe certain rules.

With multiinstance DBs, the start address of the variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is transferred with the DI data block register and address register AR2. When variables are accessed within the multiinstance enabled FB, the compiler independently controls the access operation via address register AR2. However, when complex program sections also have to work with address registers in the same FB (e.g. to copy data), then the old contents of AR2 must be saved before the register is changed. The contents of AR2 must be restored to their original state before an instance variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is accessed. The AR2 register of the instance is to be saved most usefully in a local variable (VAR_TEMP).

The command "Load pointer to an instance variable" returns a pointer value from the start of the instance data. To be able to access this variable via a pointer, the offset stored in AR2 must be added.

Example

```
FUNCTION_BLOCK FB99
VAR_INPUT
   varin: INT;
END_VAR
VAR
   variable1: ARRAY[[0..9]
   of INT;
   variable2: INT;
END_VAR
```

```
BEGIN
L          P##variable1;     //Pointer at start of ARRAY
                             //The value 8500 0010 is now in the accumulator
                             //and a cross-area pointer is in the AR2
                             //Pointer. If one is to work across areas
                             //then, during the addition of these
                             //two pointers, an area is to be disabled.
AD         DW#16#00FF_FFFF, //Skipping of an area
LAR1                         //Load into AR1
TAR2;
+AR1 AR2;                    //AR2 instance offset to be added
                             //You can now indirectly access the ARRAY
                             //of variable 1 via AR1.
L          DIW [AR1, P#0.0];//E.g. access to the first element
END_FUNCTION_BLOCK
```

14.19.4 Strings

The STRING data type is required by certain services of the basic program. For this reason, some additional facts about the string structure and general handling procedures for parameter assignments are given below.

Structure of strings

A data of type STRING is generally stored (defined) in a data block. There are two methods of defining a string:

1. Only the data type STRING is assigned to a variable. The STEP7 compiler automatically generates a length of 254 characters.
2. Data type STRING is assigned to a variable together with a string length in square parenthesis (e.g. [32]). With this method, the STEP7 compiler generates a string length corresponding to the input.

Two bytes more than prescribed by the definition are always stored for variables of the STRING data type. The STEP 7 compiler stores the maximum possible number of characters in the 1st byte. The 2nd byte contains the number of characters actually used. Normally, the useful length of the assigned strings is stored by the STEP 7 compiler. The characters (1 byte per character) are then stored from the 3rd byte onwards.

String parameters are generally assigned to blocks of the basic program by means of a POINTER or ANY. Such assignments must generally by made using symbolic programming methods. The data block, which contains the parameterizing string, must be stored in the symbol list. The assignment to the basic program block is then made by means of the symbolic data block name followed by a full stop and the symbolic name of the string variable.

14.19.5 Determining offset addresses for data block structures

Function

Another task, which occurs frequently, is symbolic determination of an offset address within a structured DB, e.g. an ARRAY or STRUCTURE is stored somewhere within the DB. After loading the address register symbolically with the start address, you might like to access the individual elements of the ARRAY or STRUCTURE via an address register. One way of loading the address register symbolically is to use an FC whose input parameter is a pointer. The address of the ARRAY or STRUCTURE is then assigned symbolically to the input parameter of this FC in the program. The program code in the FC now determines the offset address from the input parameter, and passes the offset address in the address register (AR1) to the calling function. Symbolic addressing is thus possible even with indirect access.

Example

```
FUNCTION FC99: VOID
VAR_INPUT
    Addr: POINTER;              //Points to variable
END_VAR
BEGIN
NETWORK
TITLE =
L        P##Addr;
LAR1 ;                          //Retrieve pointer from Addr
L        D [AR1,P#2.0];         //Offset part of pointer of variable
LAR1 ;
END_FUNCTION
```

14.19.6 FB calls

Function

For optimizing the execution speeds, it is useful to call all function block calls with many static parameters, such as the blocks FB2, 3, 4, 5, and 7 provided by the basic program when starting with the related instance parameters. When starting (OB100), the preassignment of the parameters must be done, which can then no longer be changed in the cyclic part (OB1). These fixed parameter values are no longer parameterized in the cyclic call, because they have already been written in the Instance DB.

Example: Parameterizing the FB2 with instance DB110

The following example shows how a sensible distribution in OB100 and OB1 component can be implemented.

First, the usual call in the cyclic program is displayed.

```
CALL FB2, DB110(
     Req :=       M 100.0,
     NumVar :=    2,                      //Read 2 variables
     Addr1 :=     NCVAR.C1_RP_rpa0_0
     Line1 :      W#16#1
     Addr2 :=     NCVAR.C1_RP_rpa0_0
     Line2 .      W#16#2
     Error :=     M1.0,
     NDR :=       M1.1,
     State :=     MW 2,
     RD1 :=       P#M 4.0 REAL 1,
     RD2 :=       P#M 24.0 REAL 1,
```

The modified version of the program call starts from here.
Here the call in OB100 is displayed:

```
CALL FB2, DB110(
     Req :=       FALSE,
     NumVar :=    2,                      //Read 2 variables
     Addr1 :=     NCVAR.C1_RP_rpa0_0
     Line1 :      W#16#1
     Addr2 :=     NCVAR.C1_RP_rpa0_0
     Line2 .      W#16#2
     RD1 :=       P#M 4.0 REAL 1,
     RD2 :=       P#M 24.0 REAL 1,
```

Here, the call still remaining in OB1 is displayed:

```
CALL FB2, DB110(
     Req :=       M0.0,
     Error :=     M1.0,
     NDR :=       M1.1,
     State :=     MW 2,
```

Note

Owing to this measure, a shorter cycle time is achieved in OB1, because the static parameter values need not be copied in the instance DB in each OB1 cycle.

The savings of this variant:

The cyclic copying effort of 3 integer values and 4 ANY parameters with respect to the instance DB, which results from 3 time loading of a constant with a 3-time transfer in the instance data block. In case of each ANY transfer, constants are loaded in the data block 4 times with subsequent transfer.

14.20 Data lists

14.20.1 Machine data

14.20.1.1 Display machine data

Number	Identifier: $MM_	Description
9032	HMI_MONITOR	Determining the PLC data for HMI monitor information

14.20.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10100	PLC_CYCLIC_TIMEOUT	Cyclic PLC monitoring time
14504	MAXNUM_USER_DATA_INT	Number of user data (INT)
14506	MAXNUM_USER_DATA_HEX	Number of user data (HEX)
14508	MAXNUM_USER_DATA_FLOAT	Number of user data (FLOAT)
14510	USER_DATA_INT	User data (INT)
14512	USER_DATA_HEX	User data (HEX)
14514	USER_DATA_FLOAT[n]	User data (FLOAT)

Note

Machine data in integer/hex format is operated in the NC as DWORD. A machine data in floating point format is managed in the NC as FLOAT (8-byte IEEE). They are stored only in the NC/PLC interface and can be read by the PLC user program from DB20 even when the PLC boots.

14.20.1.3 Channelspecific machine data

Number	Identifier: $MC_	Description
28150	MM_NUM_VDIVAR_ELEMENTS	Number of elements for writing PLC variables

14.20.2 Signals

14.20.2.1 Signals from operator panel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Active SINUMERIK operating area	DB19.DB21	DB1900.DBB1
Current screen number	DB19.DBW24	DB1900.DBW4

P4: PLC for SINUMERIK 828D 15

15.1 Overview

15.1.1 PLC firmware

The PLC of the SINUMERIK 828D is an integrated PLC based on the SIMATIC S7-200 command set.

The PLC user program is essentially programmed using a Windows PC with the "PLC Programming Tool". In addition, the PLC can be diagnosed and edited via the operator interface of the control. A "Ladder-Add-On-Tool" is available in the control for this purpose.

Note the following special features:

- The PLC user program is completely programmed in ladder logic (LAD).
- A subset of the programming language of the S7-200 is supported.
- When loading to the CPU, in addition to the code for execution, the complete project data (including symbols and comments) is loaded into the control. This means that the control always has the project that matches the currently running PLC user program.
- When loading from the CPU, the complete project data (including symbols and comments) is loaded into the PLC Programming Tool and can be processed/edited using this.
- The user must manage the data and process information according to type. The declared data type must be used consistently each time that the data is accessed.

15.1.2 PLC user interface

The user interface is set up by the PLC firmware, which also organizes the exchange of all signals and data between the PLC on one side and the NC and HMI on the other side.

The user interface comprises the parts:

- Data interface with cyclic exchange (see "Data interface (Page 1136)")
- Function interface with function- or task-related data exchange (see "Function interface (Page 1154)").

The structured data of these interfaces (retentive and non-retentive) are made available to the user by the firmware by assigning to data blocks: The NC (NC, tool management, NC channel, axes, spindles, …) and the HMI are "Communication partners" of the PLC user program.

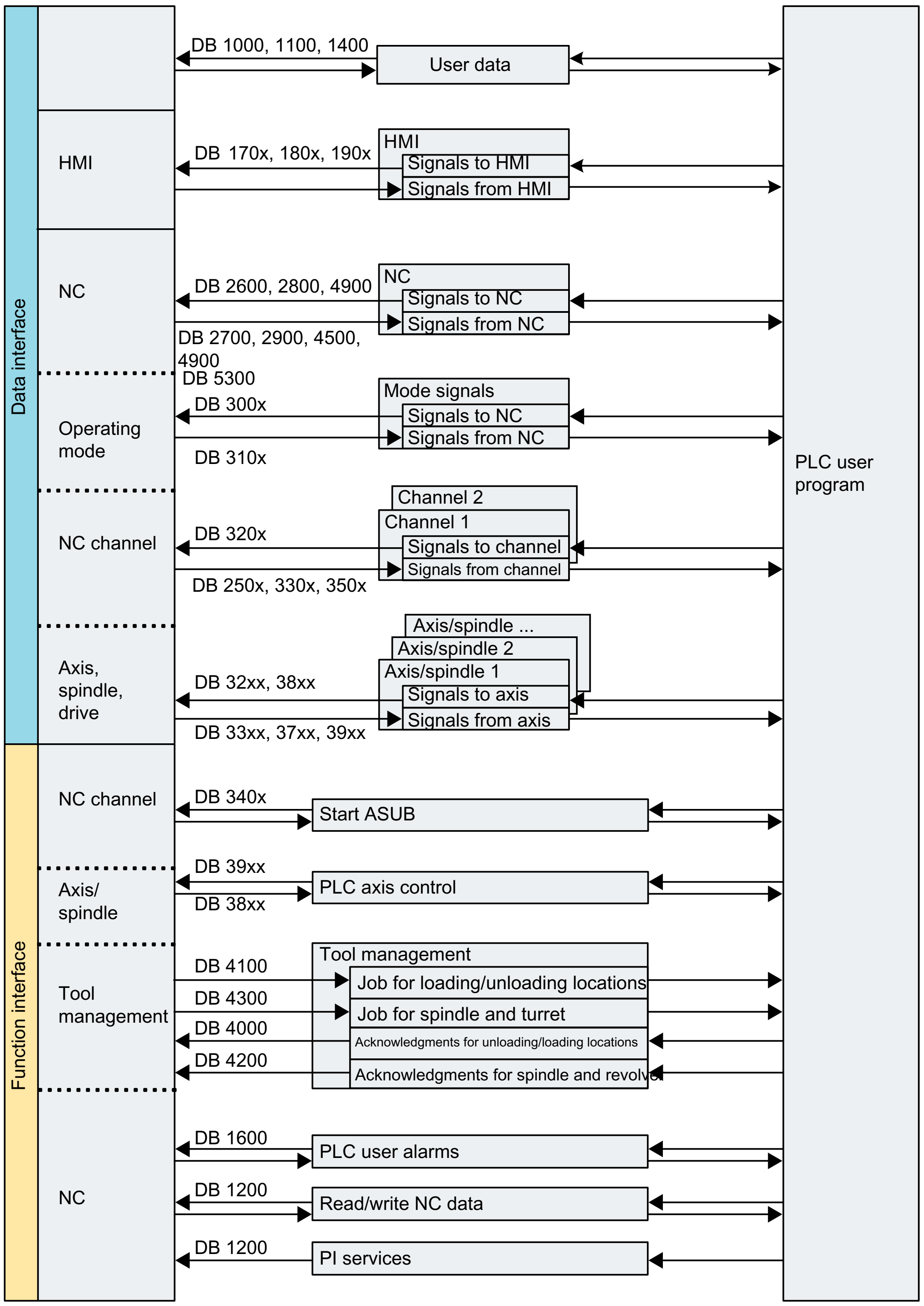


Figure 15-1 Overview of the user interface of the PLC 828D

15.1.2.1 Data that are cyclically exchanged

Data is exchanged between the PLC and NC on one side as well as between the PLC and HMI on the other side.

Data **to** the PLC is provided by the firmware at the cyclic start of the user program. This ensures, for example, that the signals from the NC remain constant throughout a cycle.

The firmware transfers data **from** the PLC to the NC or HMI at the cycle end of the user program.

PLC ↔ NC interface

The cyclic data includes status signals ("program running", "program interrupted") and control signals (start, stop), and auxiliary and G commands.

Data is structured in signals for:

- Modes
- Channels
- Axes/spindles
- General NC signals

Interface PLC ↔ HMI

These are signals for:

- Program selection via lists
- Messenger control command
- General signals from/to HMI
- Signals from/to the maintenance planner
- Signals from operator panel (retentive area)
- General selection/status signals from/to HMI (retentive area)

15.1.2.2 Alarms and messages

The user interface in DB1600 offers the option of displaying fault and operating messages on the HMI.

The firmware evaluates the signals that have been entered and sends these as coming and going alarms and messages to the HMI where they are displayed. The HMI manages the fault texts.

Note

A maximum of eight PLC alarms is displayed on the HMI.

15.1.2.3 Retentive data

For the retentive data there are the user data blocks DB9000 - DB9063 and the data area DB1400.DBW0 - DBW127. There the user can store all data that should remain valid after POWER OFF/ON. The retentive data is stored in the non-volatile memory, however, not for data backup.

15.1.2.4 Non-retentive data

Non-retentive data (e.g. bit memories, timers and counters) are deleted every time the control is booted.

15.1.2.5 PLC machine data

The PLC machine data is in the NC machine data area. At POWER ON, this data is transferred by the PLC firmware into DB4500 of the PLC user interface where it can be evaluated by the PLC user program.

References

SINUMERIK 828D Parameter Manual

15.1.3 PLC key data

The integrated PLC has a program memory of 24000 PLC statements which are completely executed in one fixed PLC cycle.

A maximum of 500 statements can be executed in the INT0 interrupt program that can be optionally used. It is executed servo-synchronous and allows the fastest possible reaction to process events. This is the reason why interrupt-capable PLC I/O modules are not required.

Data		Number	Special features
Main program (MAIN)		1	
Subprograms (SBRx)		256	
Interrupts		2	
Time-driven interrupt		1	Servo-synchronous interrupt program
Alarms/messages		1248	A maximum of 8 alarms is displayed.
Bit memory		4096	Non-retentive.
Counters		64	Non-retentive.
Timers, of which:		128	Non-retentive.
	10 ms increment interval	112	
	100 ms increment interval	16	
User data block each with a max. of 512 bytes		64	Address range DB9000 to DB9063
NC ↔ PLC data transmission			Via fixed parameterizable interface.

15.1.4 PLC I/O, fast onboard inputs/outputs

For information on the properties of the rapid onboard inputs/outputs and their response times, see Section "Fast on-board inputs and outputs (Page 1131)".

For information on the I/O modules, the machine control panels as well as the assignment of the onboard inputs/outputs to the PLC, see:
References:
Manual PPU SINUMERIK 828D

15.1.5 PLC Toolbox

15.1.5.1 Star/delta changeover

For star/delta changeover, the following block is provided in the PLC Toolbox:

- StarDelta

Note

This block can be used to perform a star/delta changeover - also for 1PH8 spindle motors with SMI connected to a SINAMICS S120.

15.2 Fast on-board inputs and outputs

For digital input and output signals that must be especially quickly processed by the control system, the PPU module has several interfaces that can be used to directly connect signals:

- Connector X242:4 input signals, 4 output signals
- Connector X252:4 input signals, 2 output signals

Interface	I/O signals		Addressing
X242	Inputs:	DIN1 ... DIN4	I256.0 ... I256.3
	Outputs:	DOUT1 ... DOUT4	Q256.0 ... Q256.3
X252	Inputs:	DIN9 ... DIN12	I256.4 ... I256.7
	Outputs:	DOUT9 ... DOUT10	Q256.4 ... Q256.5

Response times

This results in the following response times depending on the position control cycle, place of execution and digital inputs/outputs that are used:

Position control cycle clock		1.5 ms	3 ms
	Processing via:	**Response time** [1)]	
	• Subprogram SBRx in cyclic operation (MAIN, OB1) • Digital inputs/outputs: **I/O module PP 72/48**	12.5 ms	14 ms
	• Interrupt program INT0 (servo synchronous) • Digital inputs/outputs: **Onboard inputs/outputs** of the PPU	3 ms	4.5 ms
[1)] Signal at the input terminal → processing in the PLC program → signal at the output terminal			

Read/write access operations

Input/output signals are directly read and written at the module input/outputs. For the fastest possible access (servo-synchronous), use of the direct operation commands in the interrupt program **INT0** is recommended:

Command	Symbol
Direct NO contact	-| I |-
Direct NC contact	-| /I |-
Directly set bit value	-(SI)-
Directly reset bit value	-(RI)-
Directly assign bit value	-(I)-

Parameterization

Using machine data, the input/output signals can be exclusively allocated to the NC or the PLC:

- MD10366 $MN_HW_ASSIGN_DIG_FASTIN[<n>]
- MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[<n>]

with <n>: Index to address the input/output byte (0 = 1. byte, 1 = 2. byte, ...)

I/O signals		Assignment	
		NC	PLC
Inputs:	DIN1 ... DIN4	MD10366[0] = 00 01 **01** 01$_H$	MD10366[0] = 00 01 **00** 01$_H$
Outputs:	DOUT1 ... DOUT4	MD10368[0] = 00 01 **01** 01$_H$	MD10368[0] = 00 01 **00** 01$_H$
Inputs:	DIN9 ... DIN12	MD10366[1] = 00 01 **01** 01$_H$	MD10366[1] = 00 01 **00** 01$_H$
Outputs:	DOUT9 ... DOUT10	MD10368[1] = 00 01 **01** 01$_H$	MD10368[1] = 00 01 **00** 01$_H$

References

A detailed specification of the interfaces is provided in:

SINUMERIK 828D PPU Manual

15.3 Ladder Viewer, Ladder editor, and Ladder add-on tool

15.3.1 Overview

Ladder Viewer

In the Ladder Viewer, the logic operations of contacts and relays in the PLC user program are displayed as a ladder diagram (LAD).

Ladder editor

The blocks and networks of a PLC project can be edited in the Ladder editor. All the operations supported by the PLC type are available for editing.

References:
A detailed description of the functions and operation of the Ladder editor is provided in the SINUMERIK Operating Manual.

Ladder add-on tool

The Ladder add-on tool allows **limited editing of interrupt routines INT100 and INT101 when the Ladder editor is disabled**. This may be useful, for example, when reassigning inputs and outputs for service or final commissioning tasks when adapting the machine to the actual conditions at the end customer.

Restrictions

The following restrictions apply when editing with the Ladder add-on tool:

- Only empty networks can be edited. Networks, that already include statements, can only be deleted.
- A simple, single line can be edited for each network.
- You can create a maximum of 3 columns per network:

Column	Operation	
Column 1	• NO contact • NC contact	-| |- -|/|-
Column 2 (optional)	NOT Rising edge Falling edge Assign Set Reset	-|NOT|- -|P|- -|N|- -( ) -(S) -(R)
Column 3 (only possible if no assign, set or reset operations were specified in the second column)	Assign Set Reset	-( ) -(S) -(R)

Note

Logical AND (serial contact) and logical OR (parallel contact) are not possible.

The bit combinations comprise one or more logical operations and the assignment to an output / bit memory.

If the cursor is moved further to the left with the arrow key, the type of assignment or a logic operation can be selected. A further logic operation cannot be placed to the right of an assignment. A network must always be terminated with an assignment.

15.3.2 Parameterization

Display of addresses

Using NC-specific machine data, the display of addresses in the ladder Viewer can be configured according to SIMATIC S7 300 notation:

MD51230 $MN_ENABLE_LADDER_DB_ADDRESSES = <value>

<value>	Meaning
0	Address display in SIMATIC S7 200 notation (e.g. Vxxxxx)
1 (default)	Address display in SIMATIC S7 300 notation (e.g. DBxx.DBBxxxx)

Enabling / locking editing

Editing of the PLC project can be enabled or disabled in the following machine data:

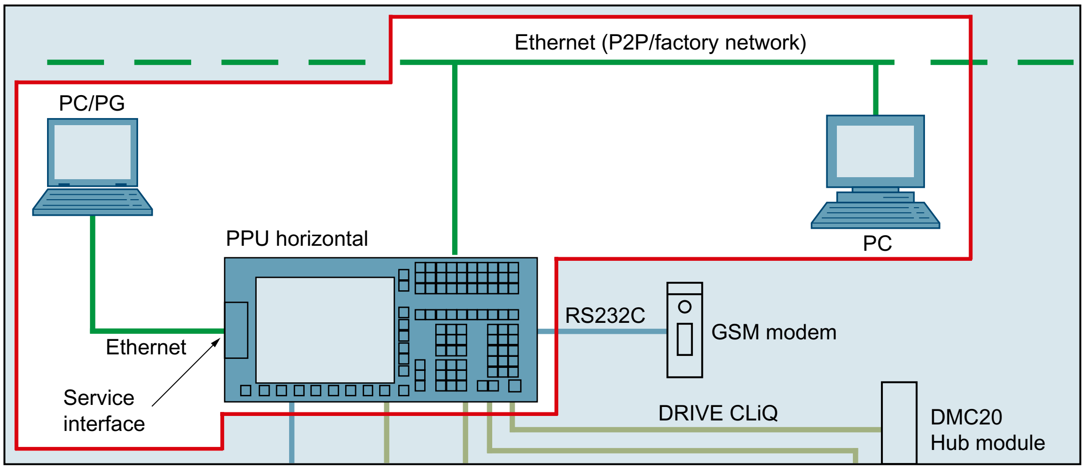
MD51231 $MN_ENABLE_LADDER_EDITOR (PLC Ladder add-on tool activated for INT100/101)

MD51232 $MN_ENABLE_LADDER_EDITOR_ADV (PLC Ladder editor activated for the entire PLC project)

In the default setting, both functions are enabled.

15.4 PLC Programming Tool

The "PLC Programming Tool" is a tool for writing PLC user programs in a user-friendly fashion. This is a Windows program and must be installed on a Windows PC. For online access to the control, the PC must be connected to the control via one of the Industrial Ethernet ports, X130 (factory network) or X127 (service interface).



When the "PLC Programming Tool" is called, without specifying an existing project, a new project is created with the default name "Project1." This project can be immediately used to generate the PLC user program, saved under any name, and loaded into the control. Existing projects can be opened in the usual Windows manner.

The online help for the "PLC Programming Tool" can also be called with the "F1" function key, just like in Windows.

References

- SINUMERIK 828D Commissioning Manual, turning and milling
  Section: "Scope of delivery and preconditions" > "Communication with the controller" > "This is how you communicate with the controller via the programming tool"
- Online help for the PLC Programming Tool

15.5 Data interface

Data is cyclically exchanged on the one hand between the PLC and NC and and on the other hand between the PLC and HMI. This especially means that the data received from HMI and destined for the NC must be marshaled by the user program in order that these become effective.

Data to the PLC is provided by the firmware at the start of the user program cycle. This ensures, for example, that the signals from the NC remain constant throughout a cycle.

Data from the PLC is transferred by the firmware to the NC or HMI at the end of the user program cycle.

All data of this interface are listed in the manual for SINUMERIK 828D, PPU.

15.5.1 PLC-NC interface

This cyclic data includes status signals ("program running", "program interrupted"), control signals (start, stop), and auxiliary and G commands.

Data is structured in signals for:

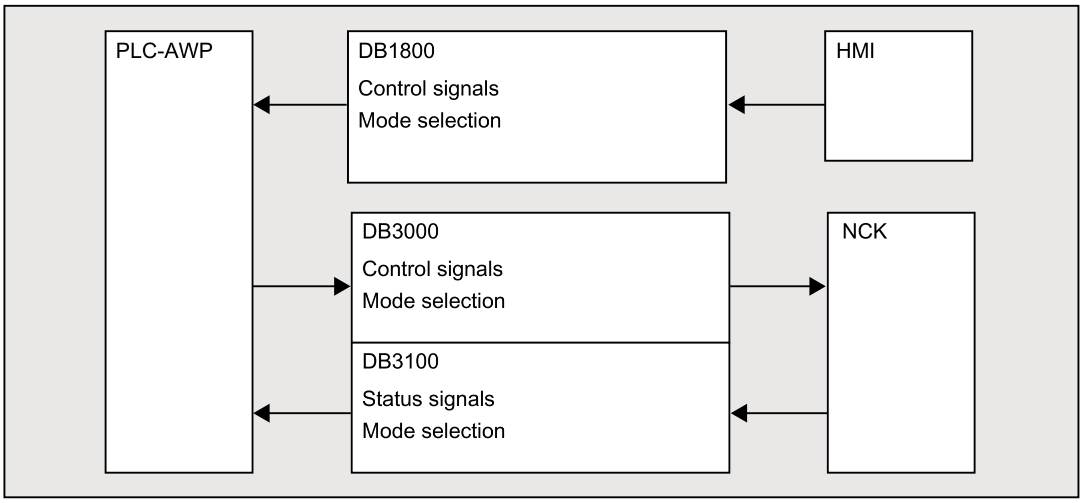
- Mode signals
- NC channel signals
- Axis and spindle signals
- General NC signals
- PLC-NC fast data exchange

15.5.1.1 Mode signals

DB3000, 3100

The mode signals specified by the machine control panel or the HMI are transferred to the NC.

Their actual states are signaled to the PLC from the NC.

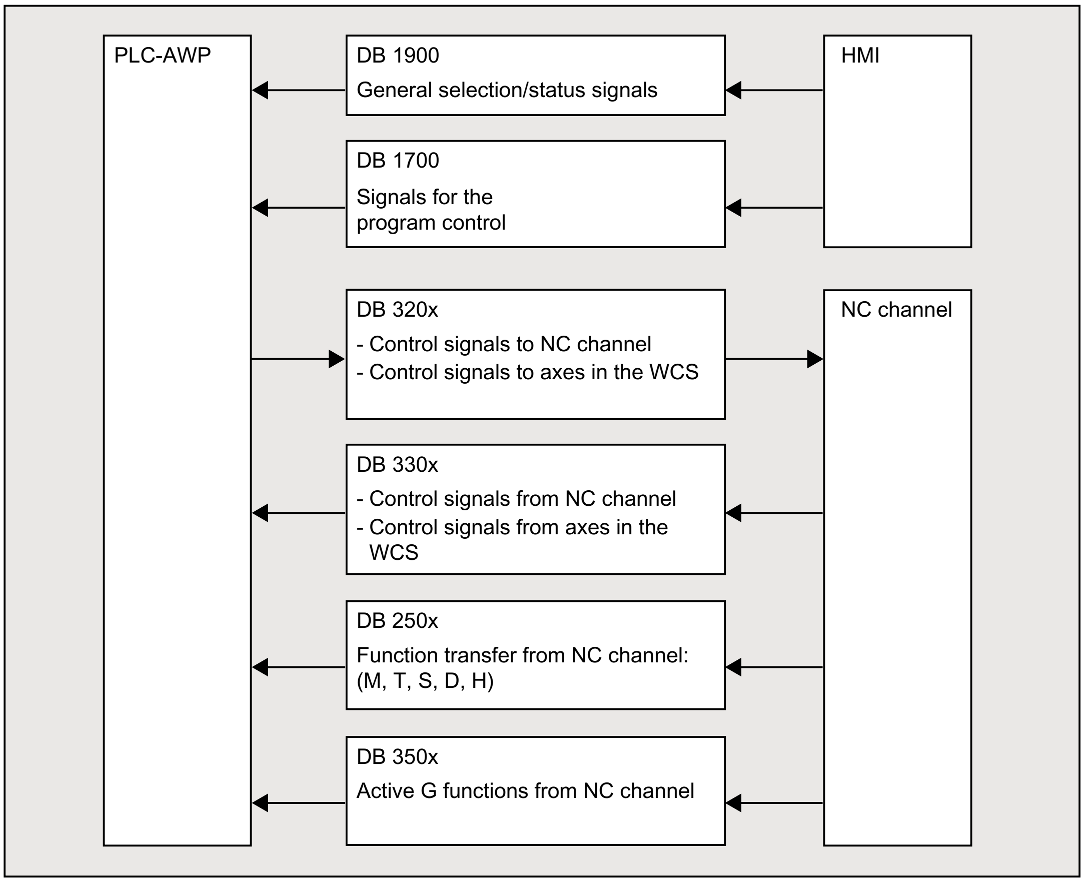

PLC-AWP
DB1800
Control signals
Mode selection
HMI
DB3000
Control signals
Mode selection
NCK
DB3100
Status signals
Mode selection

15.5.1.2 NC channel signals

DB250x, 320x, 330x, 350x

The signals are structured as follows:

- Control/status signals with normal cyclic transfer, see "Mode signals (Page 1136) ".
- Auxiliary and G commands
  These are entered in the interface DBs in two ways.
  First, they are entered with the change signals.
  The M signals M0 to M99 are additionally decoded and the associated interface bits are set for one cycle.
  For G commands, only the groups selected via machine data are entered in the interface data block.
  The S values are also entered together with the related M signals (M03, M04, M05) in the spindlespecific interface. The axisspecific feedrates are also entered in the appropriate axis-specific interface.


PLC-AWP
DB 1900
General selection/status signals
HMI
DB 1700
Signals for the program control
DB 320x
- Control signals to NC channel
- Control signals to axes in the WCS
NC channel
DB 330x
- Control signals from NC channel
- Control signals from axes in the WCS
DB 250x
Function transfer from NC channel: (M, T, S, D, H)
DB 350x
Active G functions from NC channel

15.5.1.3 Axis and spindle signals

DB370x, 380x, 390x

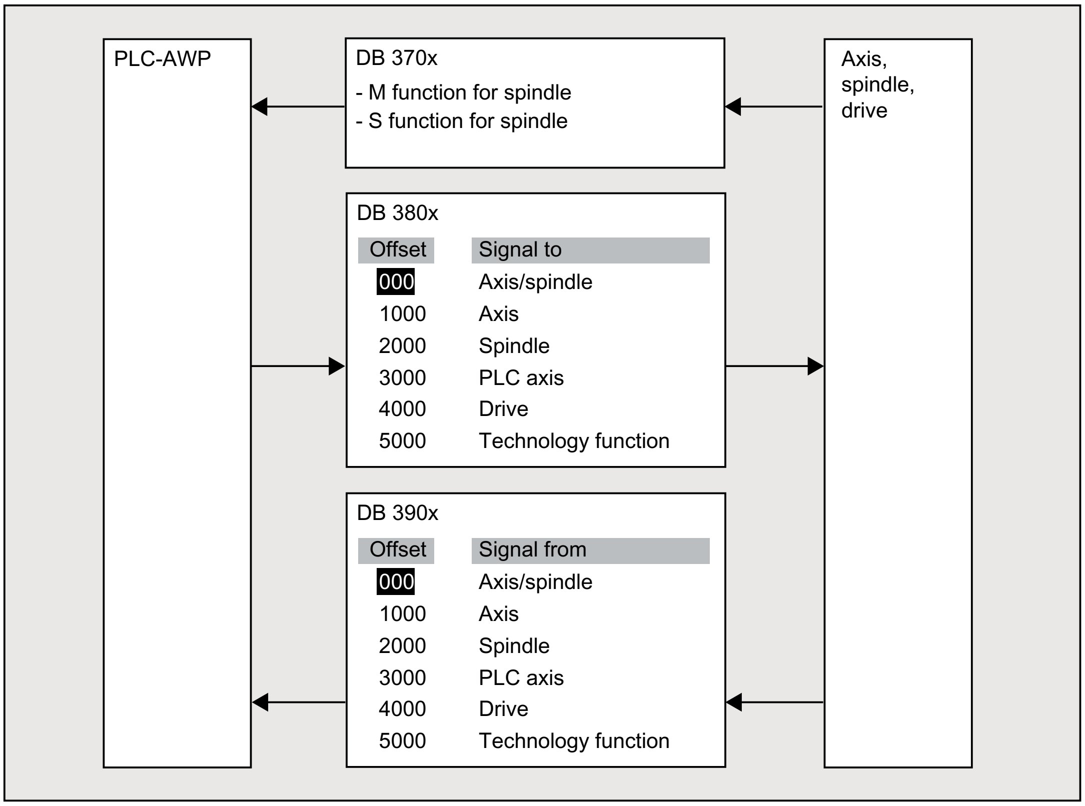
The axis-specific and spindle-specific signals are divided into the following groups:

- Shared axis/spindle signals
- Axis signals
- Spindle signals
- Drive signals

The signals are transferred cyclically with the following exceptions. The exceptions include axial F value, M and S value.

An axial F value is entered via the M, S, F distributor if it is transferred to the PLC during the NC machining process.

The M and S values are also entered via the M, S, F distributor if one or both values requires processing.



DB 380x

Offset	Signal to
000	Axis/spindle
1000	Axis
2000	Spindle
3000	PLC axis
4000	Drive
5000	Technology function

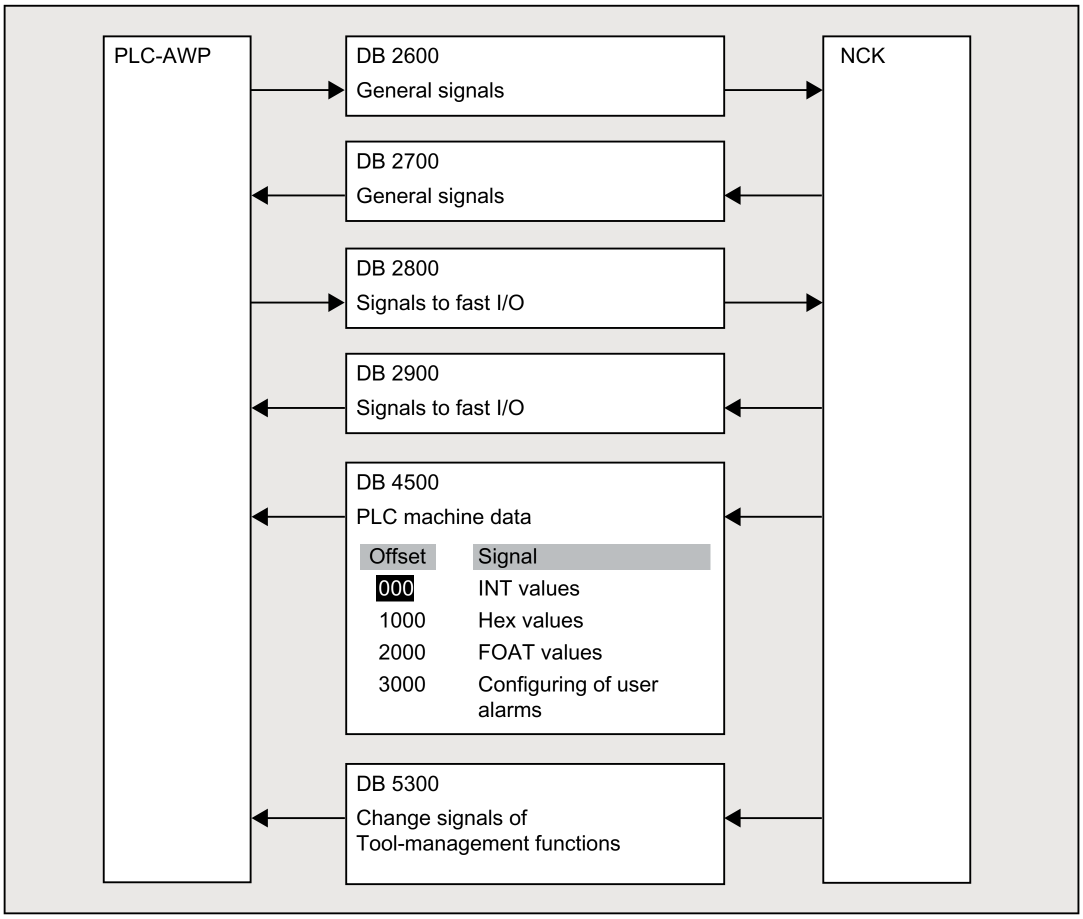
DB 390x

Offset	Signal from
000	Axis/spindle
1000	Axis
2000	Spindle
3000	PLC axis
4000	Drive
5000	Technology function

15.5.1.4 General NC signals

DB2600, 2700, 2800, 2900, 4500, 5300

- Setpoints to digital and analog inputs/outputs of the NC
- Actual values from the digital and analog inputs/outputs of the NC
- Keyswitch and Emergency Stop signals
- Ready and status signals of the NC



15.5.1.5 PLC-NC fast data exchange

DB4900

Data block DB4900 with a size of 1024 bytes is used for fast information exchange between the PLC and NC.

The assignment of the area (structure) must be identically negotiated in the NC part program and PLC user user program.

This data can be accessed from the NC part program using the commands `$A_DBB[x]`, `$A_DBW[x]`, `$A_DBD[x]` and `$A_DBR[x]; 0 ≤ x ≤ 1023` (see Parameter Manual, System variables).

In this case, the alignment of the data must be selected corresponding to its format, i.e. a Dword starts at a 4byte limit and a word at a 2byte limit. Bytes can be located at any offset within the data field.

Data consistency is guaranteed for byte, word and Dword accesses. When transferring several data, the consistency must be guaranteed on the user-side using semaphores, which can be used to detect the validity or consistency of a block.

15.5.2 PLC-HMI interface

DB1700, 1800, 1900

These signals have already been specified in the figures of Section PLC-NC interface (Page 1136).

A reference is again made to what has been stated at Data interface (Page 1136):

Data received from the HMI and destined for the NC are not automatically entered into the NC interface range. In fact, these signals and data must be marshaled by the user program.

It involves the following signals:

- Program selection via lists
- Messenger control command
- General signals from/to HMI
- Signals from/to the maintenance planner
- Signals from operator panel (retentive area)
- General selection/status signals from/to HMI (retentive area)

15.5.2.1 Program selection

Function

Preselected programs/workpieces can be selected for machining by the NC via the PLC/HMI interface.

The presetting is done by entering programs/workpiece in files, called PLC program lists (*.ppl).

Requirements

The following machine data must be set to allow the HMI to process tasks:

MD9106 $MM_SERVE_EXTCALL_PROGRAMS

In order to activate a sector-specific PLC program list, you must set the appropriate machine data and at least the protection level password:

- Area **User**
  - MD51041 $MN_ENABLE_PROGLIST_USER = 1
  - Protective level password: 3 (users)
  - Program list: /user/sinumerik/hmi/plc/programlist/plc_proglist_user.ppl
- Area **Manufacturer (OEM)**
  - MD51043 $MN_ENABLE_PROGLIST_MANUFACT = 1
  - Protective level password: 1 (manufacturer)
  - Program list: /oem/sinumerik/hmi/plc/programlist/plc_proglist_manufacturer.ppl

Structure of a program list

A program list is a text file. Each line contains the following information:

<program number> <program path><program name> [CH=<channel number>]

- Program number
  The program numbers which may be used in a program list depend on the sector:
  - user: 1 - 100
  - Individual (oem_i): 101 - 200
  - Manufacturer (OEM): 201 - 255
- Program path
  The program path must be completely specified in absolute terms.
  For specifying the program path, see:
  References
  Programming Manual, Work Planning, Section "File and Program Administration" > "Program memory" > "Addressing the files of the program memory"
- Channel number
  Specifying the channel number "CH=<channel number>" is optional. It is only required if the NC has more than one channel.

The following excerpt as example shows the structure of the user program list:

Program list: plc_proglist_user.ppl

```
1 //DEV2:/MPFDir/PROG_01.MPF CH=1
2 //DEV2:/MPFDir/PROG_01.MPF CH=2
```

Generating entries in a program list

The entries in a program list (*.ppl) can be directly edited in the file or entered in screen forms in the user interface.

- Via the user interface for the **user** area
  Operating area "Program Manager" > "ETC key (">")" > "Prog. list"
- Via the user interface for the **Manufacturer** area
  Operating area "Commissioning" > "System data" > "ETC key (">")" > "Prog. list"

Program selection: Job interface

Note

The PLC may only request a new job if the last job has been acknowledged by the HMI: DB1700.DBB2000 == 0

Program list

DB1700.DBB1001 = <number of the program list>

Number	Program list
129	/user/sinumerik/hmi/plc/programlist/**plc_proglist_user.ppl**
130	/oem_i/sinumerik/hmi/plc/programlist/**plc_proglist_individual.ppl**
131	/oem/sinumerik/hmi/plc/programlist/**plc_proglist_manufacturer.ppl**

Program number

The program number refers to the programs contained in the selected program list.

DB1700.DBB1002 = <program number>

- user area: 1 - 100
- individual area: 101 - 200
- oem area: 201 - 255

Requesting program selection

DB1700.DBX1000.7 = 1

Program selection: Acknowledgment interface

Job acknowledgment

- DB1700.DBX2000.7 == 1 (selection identified)
- DB1700.DBX2000.3 == 1 (program is selected)
- DB1700.DBX2000.2 == 1 (error when selecting the program, see error ID DB1700.DBB2001)
- DB1700.DBX2000.1 == 1 (job completed)

Error detection

DB1700.DBB2001 == <error ID>

Error detection	
Value	**Meaning**
0	No error.
1	Invalid program list number (DB1700.DBB1001).
3	User-specific program list plc_proglist_main.ppl not found (only for DB1700.DBB1001 ≠ 129, 130, 131).
4	Invalid program number (DB1700.DBB1002).
5	Job list in the selected workpiece could not be opened.
6	Error in job list (job list interpreter returns error).
7	Job list interpreter returns empty job list.

Program selection: Job processing

A job to select a program is executed as follows:

1. Checking the acknowledgment byte: DB1700.DBB2000 == 0
   If the acknowledgment byte is not 0, then the last job has still not been completed.
2. Specifying the program list: DB1700.DBB1001
3. Specifying the program number: DB1700.DBB1002
4. Setting the request to select a program: DB1700.DBX1000.7 = 1
5. Evaluating the acknowledgment and error interface: DB1700.DBB2000 and DBB2001
   The order is still not completed on the HMI side as long as: DB1700.DBX2000.3 == 1 (active)
   The order has been completed on the HMI side if one of the two signals has been set:
   - DB1700.DBX2000.1 == 1 == 1 (job completed)
   - DB1700.DBX2000.2 == 1 (error)
6. To complete the order, the program selection request must be reset: DB1700.DBX1000.7 = 0
7. The HMI signals that it is ready to accept a new order by resetting the acknowledgment byte: DB1700.DBB2000 == 0

15.5.2.2 Operating area numbers

The number of the active operating area is displayed in: DB1900.DBB1

Operating area numbers

Operating area	Number
Machine	201
Parameters	205
Programming	203
Program Manager	202

Operating area	Number
Diagnostics	204
Commissioning	206

15.5.2.3 Screen numbers

The current screen number is displayed in: DB1900.DBW4

Screen number ranges

The following screen number ranges are available:

- JOG, manual machine (Page 1145)
- Reference point approach (Page 1150)
- MDA (Page 1150)
- AUTOMATIC (Page 1150)
- Parameters operating area (Page 1151)
- Program operating area (Page 1152)
- Program manager operating area (Page 1153)
- Diagnostics operating area (Page 1153)

Screen numbers: JOG, manual machine

JOG mode

Screen	Number
Turning technology	
Cycle start screen for all screens that can be taken over	81
Milling technology	
Cycle start screen for all screens that can be taken over	3
Turning/milling	
Start screen	19
T,S,M	2
Set WO	21
Positioning	4
Face milling	18
Stock removal	80
Cycle start screen for all user screens	91
General settings	1
Multi-channel function settings	106
Collision avoidance settings	107
Measurement log settings	108
Swiveling	60

Screen	Number
All G commands	100
Actual zoom value (MCS/WCS)	101
Thread synchronizing	102
Retract	103
Handwheel	104
Action synchronization	105
Turning technology: Workpiece zero	
Workpiece zero (main menu)	30
User screen	31
User screen	34
User screen	35
User screen	36
User screen	37
User screen	38
User screen	40
Measure edge Z	5
Turning technology: Workpiece, measurement	
Measure tool (main menu)	50
Manual X or user screen	51
Manual Y	71
Manual Z or user screen	52
Zoom or user screen	53
User screen	54
User screen	55
Probe calibration X or user screen	56
Probe calibration Z or user screen	57
Automatic length in Z	58
Automatic length in Y	73
Automatic length in X	59
Milling technology: Workpiece zero	
Workpiece zero (main menu)	30
Measure edge X	5
Measure edge X	22
Measure edge Z	23
User screen	7
Align edge or user screen	31
Distance 2 edges or user screen	32
Right-angled corner	33
Any corner or user screen	8
Rectangular pocket	34
1 hole or user screen	9
2 holes	35
3 holes	36

Screen	Number
4 holes	37
Rectangular spigot	38
1 circular spigot or user screen	10
2 circular spigots	39
3 circular spigots	40
4 circular spigots	41
Set up level	42
Probe length calibration or user screen	11
Probe radius calibration	12
Milling technology: Workpiece, measurement	
Measure tool (main menu)	50
Measure length, manual (with milling tool) or measure length in X, manual (with turning tool) or user screen	16
Measure length in Y, manual (with turning tool)	74
Measure length in Z, manual (with turning tool)	24
Measure diameter, manual or user screen	17
Measure length, automatic (with milling tool) or measure length in X, automatic (with turning tool) or user screen	13
Measure length in Y, automatic (with turning tool)	75
Measure length in Z, automatic (with turning tool)	25
Measure diameter, automatic or user screen	14
User screen	51
Probe calibration or user screen	15
Fixed point calibration or user screen	52
RunMyScreens (only for set JobShopIntegration)	
User screen for the 1st horizontal softkey	96
User screen for the 2nd horizontal softkey	98
User screen for the 3rd horizontal softkey	99
User screen for the 4th horizontal softkey	94
User screen for the 5th horizontal softkey	95
User screen for the 6th horizontal softkey	92
User screen for the 7th horizontal softkey	97
User screen for the 8th horizontal softkey	90
User screen for the 9th horizontal softkey	83
User screen for the 10th horizontal softkey	82
User screen for the 11th horizontal softkey	93
User screen for the 12th horizontal softkey	84
User screen for the 13th horizontal softkey	85
User screen for the 14th horizontal softkey	86
User screen for the 15th horizontal softkey	87
User screen for the 16th horizontal softkey	88

JOG mode, manual machine

DB19.DBB24	
Screen	**Screen number**
Turning/milling	
Taper turning	61
Angle milling	62
Stop	63
Straight line	1300
Straight line all axes	1330
Straight line X alpha	1340
Straight line Z alpha	1350
Circle	1360
Drilling	1400
Center drilling	1410
Drilling, thread centered	1420
Drilling, centering	1433
Drilling, drilling	1434
Drilling, reaming	1435
Drilling, boring	1436
Drilling, deep hole drilling	1440
Drilling, deep hole drilling 2	1441
Drilling, tapping	1453
Drill thread milling	1455
Positions	1473
Position row	1474
Position grid	1477
Position frame	1478
Position circle	1475
Position pitch circle	1479
Obstacle	1476
Turning	1500
Turning, stock removal 1	1513
Turning, stock removal 2	1514
Turning, stock removal 3	1515
Turning, groove 1	1523
Turning, groove 2	1524
Turning, groove 3	1525
Turning, undercut form E	1533
Turning, undercut form F	1534
Turning, undercut thread DIN	1535
Turning, undercut thread	1536
Turning, thread, longitudinal	1543
Turning, thread, taper	1544

DB19.DBB24	
Screen	**Screen number**
Turning, thread, facing	1545
Turning, thread, chain	1546
Turning, cut-off	1550
Milling	1600
Milling, face milling	1610
Milling, rectangular pocket	1613
Milling, circular pocket	1614
Milling, rectangular spigot	1623
Milling, circular spigot	1624
Milling, longitudinal groove	1633
Milling, circumferential groove	1634
Milling, open groove	1635
Milling, multi-edge	1640
Milling, thread milling	1454
Milling, engraving	1670
Contour turning	1200
Contour turning, new contour / last contour	1210
Contour turning, stock removal along contour	1220
Contour turning, contour grooving	1230
Contour turning, contour plunge turning	1240
Contour milling	1100
Contour milling, new contour / last contour	1110
Contour milling, path milling	1120
Contour milling, centering	1130
Contour milling, rough drilling	1140
Contour milling, contour pocket	1150
Turning technology: Simulation	
Side view	1740
Front view	1750
3D view	1760
2-window view	1770
Half section	1780
Turning technology: Simultaneous recording	
Side view	1741
Front view	1751
3D view	1761
2-window view	1771
Machine space	1791
Half section	1781
Milling technology: Simulation	
Top view	1742
3D view	1760

DB19.DBB24	
Screen	Screen number
From the front	1744
From the rear	1746
From the Left	1748
From the right	1752
Half section	1780
Turning view	1782
Milling technology: Simultaneous recording	
Top view	1743
3D view	1761
From the front	1745
From the rear	1747
From the Left	1749
From the right	1753
Machine space	1791
Half section	1781
Turning view	1783

Screen numbers: Reference point approach

Screen	Number
Actual zoom value MCS/WCS	101

Screen numbers: MDA

Screen	Number
MDI	20
All G commands	100
Actual zoom value MCS/WCS	101
Handwheel	104
Action synchronization	105
Program control	210
Settings	250

Screen numbers: AUTOMATIC

Screen	Number
Automatic	200
Overstore	202
Program control	210
Block search	220
General settings	250

Screen	Number
Multi-channel function settings	106
Collision avoidance settings	107
All G commands	100
Actual zoom value MCS/WCS	101
Handwheel	104
Action synchronization	105
Turning technology: Simultaneous recording	
Side view	243
Front view	244
3D view	245
2-window view	246
Machine space	247
Half section	253
Milling technology: Simultaneous recording	
Top view	242
3D view	244
From the front	248
From the rear	249
From the Left	251
From the right	252
Machine space	247
Half section	253
Turning view	254

Screen numbers: Parameters operating area

Screen	Number
Tool list	600
Tool wear	610
User tool list	620
Magazine	630
Work offset	
Work offset, active	642
Work offset, overview	643
Work offset, basic	644
Work offset, G54 - G509	645
Details of work offset, active, overview, basic or G54 - G509	647
User variable	
R parameters	650
Global GUD 1 (SGUD)	660
Global GUD 2 (MGUD)	661
Global GUD 3 (UGUD)	662

Screen	Number
Global GUD 4	663
Global GUD 5	664
Global GUD 6	665
Global GUD 7	666
Global GUD 8	667
Global GUD 9	668
Channel GUD 1 (SGUD)	690
Channel GUD 2 (MGUD)	691
Channel GUD 3 (UGUD)	692
Channel GUD 4	693
Channel GUD 5	694
Channel GUD 6	695
Channel GUD 7	696
Channel GUD 8	697
Channel GUD 9	698
Local LUD	681
Local LUD/PUD	684
Setting data	
Working area limitation	671
Spindle data	670
Spindle chuck data	672
Ctrl-Energy	
Ctrl-Energy, main menu	6170
Ctrl-Energy, analysis	6171
Ctrl-Energy, profiles	6172
Ctrl-Energy, analysis graphic	6176
Ctrl-Energy, analysis long-term measurement	6177
Ctrl-Energy, analysis details	6179
Ctrl-Energy, compare measurements	6178

Screen numbers: Program operating area

Screen	Number
Turning technology: Simulation	
Side view	413
Front view	414
3D view	415
2-window view	416
Half section	423
Milling technology: Simulation	
Top view	412
3D view	414

Screen	Number
From the front	418
From the rear	419
From the Left	421
From the right	422
Half section	423
Turning view	424

Screen numbers: Program manager operating area

Screen	Number
NC directory	300
Local drive	325
USB / configured drive1	330
Configured drive2	340
Configured drive3	350
Configured drive4	360
Configured drive5	383
Configured drive6	384
Configured drive7	385
Configured drive8	386

Screen numbers: Diagnostics operating area

Screen	Number
Alarm list	500
Messages	501
Alarm log	502
NC/PLC variable	503

15.5.2.4 HMI monitor

Function

The HMI monitor is an 8-byte data area in a freely selectable data block, in which the HMI provides the following data for the PLC user program:

- Operating area numbers (Page 1144)
- Screen numbers (Page 1145)

Parameterization

The data area is configured using the following display machine data:

MD9032 $MM_HMI_MONITOR = "string"

with "string" = "DB<DB number>.DBB<byte address>"

Note

Even byte address

The data area must start at an even byte address.

Structure of the data area

Byte	Meaning
EB n + 0	Active SINUMERIK operating area
EB n + 1	Reserved
EB n + 2	Current screen number
EB n + 3	
EB n + 4	Reserved
...	...
EB n + 7	Reserved

Supplementary conditions

When the HMI monitoring is active, the following PLC/HMI interface signals are no longer processed:

- DB1900.DBB5003 (PLC hardkeys)
- DB1900.DBB0001 (active SINUMERIK operating area)
- DB1900.DBW0004 (current screen number)

15.6 Function interface

15.6.1 Read/write NC variables

15.6.1.1 User interface

The PLC user program can read or write a maximum of eight NC variables simultaneously via the NC/PLC interface "Read/write NC variable".

The following steps must be performed as part of a job (read/write):

1. Job specification (Page 1155)
2. Job management: Start job (Page 1156)
3. Job management: Waiting for end of job (Page 1157)
4. Job management: Job completion (Page 1157)
5. Job evaluation (Page 1158)

Flow diagram of a job: See "Job management: Flow diagram (Page 1158)"

15.6.1.2 Job specification

Variable-specific job interface

Each variable that is to be processed in a job, must be specified in the **variable-specific job interface** via its parameters. The general identifiers are discussed in more detail later for each variable that can be accessed from the interface.

DB120x 1)	Read/write NC data (PLC → NC)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB1000	Variable index							
DBB1001	Area number							
DBW1002	Line index, NC variable x							
DBW1004	Column index, NC variable x							
DBW1006	---							
DBD1008	Write: Data to NC variable x (data type of the variables: 1…4 bytes)2)							
DBD1012	---							
DBD1016	Write: Data to NC variable x (REAL) 3)							
DBD1020	Write: Data to NC variable x (DWORD/DINT) 3)							
DBW1024	Write: Data to NC variable x (WORD/INT) 3)							
DBB1026	Write: Data to NC variable x (BYTE) 3)							
DBB1027	---	---	---	---	---	---	---	Write: Data to NC variable x 3)
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								
2) Only for predefined variables of the "Read/write NC variable" user interface								
3) Only for variables from DB9910 NC_DATA								

Note

Channel-specific variables

When reading/writing channel-specific variables, only the variables of exactly **one** channel may be addressed in a job.

Drive-specific variables

When reading/writing drive-specific variables, only the variables of exactly **one** SERVO drive object may be addressed in a job. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

Error case

In the event of an error, reading/writing variables from different drive objects, or simultaneously from a channel and a drive object, an error message is output:

DB1200.DBX3000.1 == 1 (error occurred)

Example: Reading a variable of the "Location type" as the fourth variable

```
DB1203.DBB1000: 7
DB1203.DBB1001: -
DB1203.DBW1002: <Location number>
DB1203.DBW1004: <Magazine number>
DB1203.DBW1006: -
DB1203.DBD1008: -
```

Example: Writing a variable as the fourth variable

To write a data item to the NC, the value must be entered into the double word `DBD1008`:

```
DB1203.DBB1000: <Variable index>
DB1203.DBB1001: <Area number>
DB1203.DBW1002: <Column index>
DB1203.DBW1004: <Line index>
DB1203.DBW1006: -
DB1203.DBD1008: <Value>
```

15.6.1.3 Job management: Start job

The following data must be written by the user to the **global job interface**:

DB120x 1)	Read/write NC data (PLC → NC)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB0							Job type	Job: Start
DBB1	Number of variables to be processed in the job							
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								

Job type

- **Read** variable: DB1200.DBX0.1 = **0**
- **Write** variable: DB1200.DBX0.1 = **1**

Start job

The start signal must be set to start the job via a specified number of variables:

DB1200.DBX0.0 = **1**

Note

A new job can only be started if the previous job was completed. See Section "Job management: Waiting for end of job (Page 1157)".

The execution of a job may take several PLC cycles and vary depending on the utilization. Therefore, the time for this function cannot be defined.

15.6.1.4 Job management: Waiting for end of job

The end of the job is always signaled back by the NC for the **whole** job in the **global result interface**. The signals can only be read by the PLC user.

DB120x [1)]	Read/write NC data (NC → PLC)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB2000							Error in job	Job completed
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								

Job status

- End of job without error
  DB1200.DBX2000.0 == **1 AND** DB1200.DBX2000.1 == **0**
- End of job with error
  DB1200.DBX2000.0 == **1 AND** DB1200.DBX2000.1 == **1**

Possible error causes

- Number of variables (DB1200.DBB1) out of the valid range
- Variable index (DB1200.DBB1000) out of the valid range
- Simultaneous reading/writing of NC data from different servo drive objects

15.6.1.5 Job management: Job completion

Requirement

In order to complete the job, the start signal of the job must be reset from the PLC user program after detection of the end of the job:

DB1200.DBX0.0 = **0**

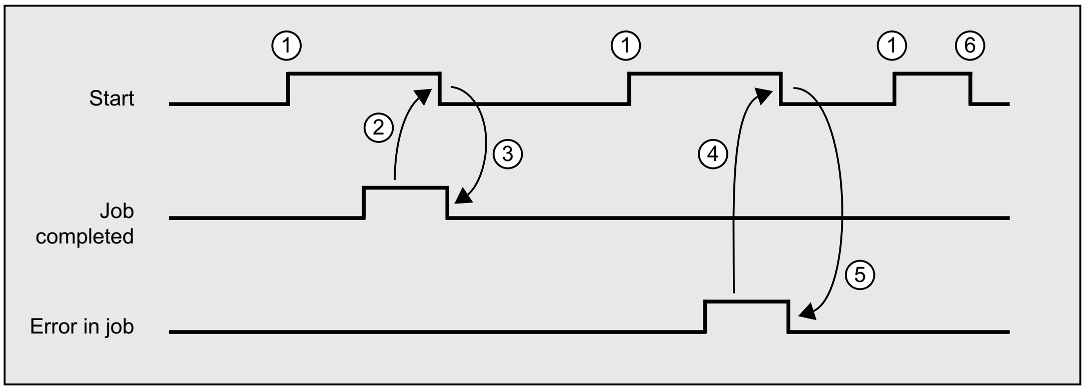
Feedback signal

As feedback, the NC resets the status signals:

- DB1200.DBX2000.0 == **0**
- DB1200.DBX2000.1 == **0**

The job is now completed.

15.6.1.6 Job management: Flow diagram



①	Start job: DB1200.DBX0.0 (start) = 1
②	Waiting for end of job: DB1200.DBX2000.0 (job completed) == **1** AND DB1200.DBX2000.1 (error in job) == **0** ⇒ Reset job request: DB1200.DBX0.0 = 0 (start)
③	With DB1200.DBX0 0 == 0 (start), the job is completed by the basic PLC program: DB1200.DBX2000.0 (job completed) = 0
④	Waiting for end of job: DB1200.DBX2000.0 (job completed) == **0** AND DB1200.DBX2000.1 (error in job) == **1** ⇒ Perform error handling ⇒ Reset job request: DB1200.DBX0.0 (start) = 0
⑤	With DB1200.DBX0.0 == 0 (start), the job is completed by the basic PLC program: DB1200.DBX2000.1 (error in job) = 0
⑥	If DB1200.DBX0.0 (start) is reset before the end of job is signaled by the basic PLC program, the job is executed without further feedback.

15.6.1.7 Job evaluation

The variable-specific result interface must be evaluated for each variable processed in the job.

DB120x 1)	**NC services (NC → PLC)**							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB3000							Error has occurred	Variable valid
DBB3001	Access result (see "Access result" below)							
DBW3002	---							
DBD3004	Read: Data from NC variable x (data type of the variables: 1…4 bytes)2)							
DBD3008	---							
DBD3012	---							
DBD3016	Read: Data from NC variable x (REAL)3							

DB120x [1]	NC services (NC → PLC)							
DBD3020	Read: Data from NC variable x (DWORD / DINT)[3]							
DBW3024	Read: Data from NCK variable x (WORD / INT)[3]							
DBB3026	Read: Data from NCK variable x (BYTE)[3]							
DBB3027	---	---	---	---	---	---	---	Read: Data from NC variable x [3]
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								
2) Only for predefined variables of the "Read/write NC variable" user interface								
3) Only for variables from DB9910 NC_DATA								

Note

Channel-specific variables

When reading/writing channel-specific variables, only the variables of exactly **one** channel may be addressed in a job.

Drive-specific variables

When reading/writing drive-specific variables, only the variables of exactly **one** SERVO drive object may be addressed in a job. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

Error case

In the event of an error, reading/writing variables from different drive objects, or simultaneously from a channel and a drive object, an error message is output:

DB1200.DBX3000.1 == 1 (error occurred)

Access result

NC variables

DBB3001	
Value	**Meaning**
0	No error
3	Access to object is not permitted
5	Invalid address
10	Object does not exist

Drive-specific variables

If an error occurs while reading/writing a drive-specific variable (DB1200.DBX3000.1 == 1), an error number is displayed in the access result which is based on the error numbers defined in the PROFIdrive profile.

DBB3001	
Value	**Meaning**
x	<error number of the PROFIdrive profile> + 20_H or 36_D

Determining the meaning of the access result:

1. Calculate the error number of the PROFIdrive profile
   <error number of the PROFIdrive profile> = access result - 20_H or 36_D.
2. Determine the meaning of the error number of the PROFIdrive profile
   The error numbers of the PROFIdrive profile are described in:
   References
   Function Manual, SINAMICS S120 Drive Functions; Section "Communication" > "Communication according to PROFIdrive" > "Acyclic communication" > "Structure of the requests and responses" > Subsection "Error values in parameter responses"

Examples: Job status

Job without error

- DB1200.DBX3000.0 == 1 (variable valid) **AND**
- DB1200.DBX3000.1 == 0 (no error occurred)

Result:

- DB1200.DBB3001 == 0 (access result: "No error")
- DB1200.DBD3004 == <read value>

Job with error

- DB1200.DBX3000.0 == 0 (variable not valid) **AND**
- DB1200.DBX3000.1 == 1 (error occurred)

Result:

- DB1200.DBB3001: For possible error causes, see "Access result" above

15.6.1.8 Operable variables

The following variables are available:

Variable	Meaning
cuttEdgeParam (Page 1161)	Compensation value parameters and cutting edge list with D numbers for a tool
numCuttEdgeParams (Page 1161)	Number of P elements of a cutting edge
linShift (Page 1161)	Translation of a settable zero-point offset
numMachAxes (Page 1162)	Number of the highest existing channel axis
rpa (Page 1162)	R parameters
actLineNumber (Page 1163)	Line number of the current NC block
$TC_MPPx (Page 1163)	Magazine location data
r0078[1] (Page 1164)	Current actual value, torque-generating
r0079[1] (Page 1165)	Torque setpoint at the output of the speed controller
r0081 (Page 1165)	Torque utilization in percent

Variable	Meaning
r0082[1] (Page 1165)	Active power actual value
TEMP_COMP_x (Page 1166)	Temperature compensation data

Variable "cuttEdgeParam"

Compensation value parameters and cutting edge list with D numbers for a tool

The meanings of the individual parameters depend on the type of the tool in question. Currently, 25 parameters are reserved for each tool edge (but only a part of them is loaded with values). To be able to remain flexible for future extensions, it is not recommended to use a fixed value of 25 parameters for calculation, but the variable value 'numCuttEdgeParams' (variable index 2).

For a detailed description of tool parameters, see Section W1: Tool offset (Page 1451).

Address	Value / meaning
DB120x.DBB1000	1
DB120x.DBB1001	-
DB120x.DBW1002	(Cutting edge No. - 1) * numCuttEdgeParams + ParameterNr (WORD)
DB120x.DBW1004	T number (1...32000) (WORD)
DB120x.DBD1008	**Write**: Data to NC variable x (data type: REAL)
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable "numCuttEdgeParams"

Number of P elements of a cutting edge

Address	Value / meaning
DB120x.DBB1000	2
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Reading**: Data from NCK variable x (data type: WORD)

Variable "linShift"

Translation of a settable work offset (channel-specific settable frames)

The variable only exists if MD18601 MM_NUM_GLOBAL_USER_FRAMES > 0.

The following frame indices are available:

Index	Meaning
0	ACTFRAME = actual resulting work offset
1	IFRAME = actual settable work offset
2	PFRAME = actual programmable work offset

Index	Meaning
3	EXTFRAME = actual external work offset
4	TOTFRAME = actual total work offset = total of ACTFRAME and EXTFRAME
5	ACTBFRAME = actual total base frame
6	SETFRAME = current 1st system frame (PRESET, scratching)
7	EXTFRAME = current 2nd system frame (PRESET, scratching)
8	PARTFRAME = current 3rd system frame (TCARR and PAROT with an orientable tool-holder)
9	TOOLFRAME = current 4th system frame (TOROT and TOFRAME)
10	MEASFRAME = result frame for the workpiece and tool measuring
11	WPFRAME = current 5th system frame (workpiece reference points)
12	CYCFRAME = current 6th system frame (cycles)

Address	Value / meaning
DB120x.DBB1000	3
DB120x.DBB1001	-
DB120x.DBW1002	Frame index * numMachAxes (Page 1162) + axis number
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable "numMachAxes"

Number of the highest existing channel axis

If there are no channel axis **gaps**, the value of the variables is also the number of available axes in the channel.

Address	Value / meaning
DB120x.DBB1000	4
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: Data from NC variable x (data type: WORD)

Variable "rpa"

R parameters

Address	Value / meaning
DB120x.DBB1000	5
DB120x.DBB1001	-
DB120x.DBW1002	R number + 1

Address	Value / meaning
DB120x.DBW1004	-
DB120x.DBD1008	**Write**: Data to NC variable x (data type: REAL)
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable "actLineNumber"

Line number of the current NC block

No.	Meaning
≥ 1	Line number of the current NC block
0	No line number available because the program has not started
-1	No line number available: Error
-2	No line number available: Suppression of block display with `DISPLOF` is active

Address	Value / meaning
DB120x.DBB1000	6
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: Data from NC variable x (data type: INT)

Tool management: Magazine location data

Location type ($TC_MPP2)

Address	Value / meaning	
DB120x.DBB1000	7	
DB120x.DBB1001	-	
DB120x.DBW1002	Location number (1 … 31999)	
DB120x.DBW1004	Magazine number (1 … 9999)	
DB120x.DBD1008	-	
DB120x.DBW3004	**Read**: Value of the NC variable (data type: WORD)	
	Value	**Meaning**
	> 0	Location type for virtual location
	0	"match all" (buffer)
	9999	undefined (no virtual location)

Location status ($TC_MPP4)

Address	Value / meaning
DB120x.DBB1000	8
DB120x.DBB1001	-

Address	Value / meaning	
DB120x.DBW1002	Location number (1 … 31999)	
DB120x.DBW1004	Magazine number (1 … 9999)	
DB120x.DBD1008	-	
DB120x.DBW3004	**Read**: Value of the NC variable (data type: WORD)	
	Value	**Meaning**
	1	Blocked
	2	free (<> occupied)
	4	reserved for tool in buffer
	8	reserved for tool to be loaded
	16	occupied in left half location
	32	occupied in right half location
	64	occupied in upper half location
	128	occupied in lower half location

T No. of tool at this location ($TC_MPP6)

Address	Value / meaning
DB120x.DBB1000	9
DB120x.DBB1001	-
DB120x.DBW1002	Location number (1 … 31999)
DB120x.DBW1004	Magazine number (1 … 9999)
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: T number of the tool at this location (data type: WORD)

Variable r0078[1]

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Current actual value, torque-generating [A_{rms}]
- Index [1]: smoothed with p0045

Address	Value / meaning
DB120x.DBB1000	10
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable r0079[1]

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Torque setpoint at the output of the speed controller (before clock cycle interpolation) [Nm]
- Index [1]: smoothed with p0045

Address	Value / meaning
DB120x.DBB1000	11
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable r0081

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Torque utilization in percent

Address	Value / meaning
DB120x.DBB1000	12
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable r0082[1]

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Active power actual value [kW]
- Index [1]: smoothed with p0045

Address	Value / meaning
DB120x.DBB1000	13
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Temperature compensation

Variable "TEMP_COMP_ABS_VALUE" (SD43900)

Position-independent temperature compensation value

Address	Value / meaning
DB120x.DBB1000	14
DB120x.DBB1001	No. of the axis (1, 2, ...)
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	Write: Data to NC variable x (data type: REAL)
DB120x.DBW3004	Read: Data from NC variable x (data type: REAL)

Variable "TEMP_COMP_SLOPE" (SD43910)

Gradient for position-dependent temperature compensation

Address	Value / meaning
DB120x.DBB1000	15
DB120x.DBB1001	No. of the axis (1, 2, ...)
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	**Write**: Data to NC variable x (data type: REAL)
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable "TEMP_COMP_REF_POSITION" (SD43920)

Reference position for position-dependent temperature compensation

Address	Value / meaning
DB120x.DBB1000	16
DB120x.DBB1001	No. of the axis (1, 2, ...)
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	**Write**: Data to NC variable x (data type: REAL)
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

Variable "TOOL_TEMP_COMP" (SD42960[...])

Temperature compensation referred to the tool

Address	Value / meaning
DB120x.DBB1000	17
DB120x.DBB1001	-
DB120x.DBW1002	Index + 1 (1, 2, 3)
DB120x.DBW1004	-
DB120x.DBD1008	**Write**: Data to NC variable x (data type: REAL)
DB120x.DBW3004	**Read**: Data from NC variable x (data type: REAL)

15.6.1.9 Specifying selected NC variables

The selected NC variables are specified via the DB9910 data block (selected NC variable). The length of the data block depends on the number of NC variables selected in the variables list. The variables list can contain maximum 42 selected NC variables. The DB9910 data block contains the data for variable addressing and data type conversion for each NC variable.

DB9910	Selected NC variable, read (PLC → NC)							
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB0	Variable index **Variable 1**							
DBB1	Syntax ID							
DBB2	Area							
DBB3	Unit							
DBW4	Column index							
DBW6	Line index							
DBB8	Block							
DBB9	Number of lines							
DBB10	Type							
DBB11	Length							
....	.....							
DBB492	Variable index **Variable 42**							
DBB493	Syntax ID							
DBB494	Area							
DBB495	Unit							
DBW496	Column index							
DBW498	Line index							
DBB500	Block							
DBB501	Number of lines							
DBB502	Type							
DBB503	Length							

Variable index

The variable index refers to the name of the NC variable. The variable index consists of a start value 100 and the offset of the NC variable in the list (0 to 41).

The variable index is entered as DBB1000 A_VarIdx in the DB120x RW_NCDx user interface.

The comment contains the following data record with space as separator:

- Area
- Block
- VariablenName
- VarType
- Column
- VarAnzByte

Extended user interface

The data test in data blocks means the value of a variable can be written to an address in the data block that has the same type, e.g. a REAL value can be written only to a REAL address (e.g. with MOV_R). The "Read/write NC variable " user interface contains currently only one address of the DWORD type for the value to be written (DBD1008). This means a REAL value can be written only via a temporary variable, flag or accumulator. The same is also true for reading (DBD3004). For this reason, the "Read/write NC variable" user interface is extended. For reading and writing, one address for each type is added: REAL, DWORD/DINT, WORD/INT, BYTE and BOOL (DBD1016 … DBB1027 or DBD3016 … DBB3027). These new addresses are used by the PLC firmware only for those variables selected with the NC variables editor – and entered into DB9910 NC_DATA when compiled (variable index ≥ 100). The NC variables currently defined in the "Read/write NC variable" user interface continue to use the old addresses (DBD1008 or DBD3004).

15.6.2 Program instance services (PI services)

15.6.2.1 Job specification

PI services are specified via the **job interface** (DB1200 from offset 4000).

DB1200			**PI service [r/w]**					
			PLC → NC interface					
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 4000	-	-	-	-	-	-	-	Start 1)
DBB 4001	PI index 2)							
DBB 4002	-							
DBB 4003	-							
DBW 4004	PI parameter 1 3)							
DBW 4006	PI parameter 2							
DBW 4008	PI parameter 3							
DBW 4010	PI parameter 4							
DBW 4012	PI parameter 5							
DBW 4014	PI parameter 6							
DBW 4016	PI parameter 7							
DBW 4018	PI parameter 8							
DBW 4020	PI parameter 9							
DBW 4022	PI parameter 10							
1) DB1200.DBX4000.1, start: DBX4000.1 = 1 ⇒ start of the PI service; DBX4000.1 = 0 ⇒ PI service completed								
2) DB1200.DBB4001, PI index: Specifies the specific PI service								
3) DB1200.DBW4004 ..., PI parameters: PI-specific parameters								

Overview of the PI services:

- PI service ASUB (Page 1169)
- PI service LOGOUT (Page 1171)

- PI service DATA_SAVE (Page 1171)
- PI service TMMVTL (Page 1172)

15.6.2.2 Job feedback

The PLC provides feedback as to whether the started PI service was successful or not successful in the **result interface** (DB1200 from offset 5000).

The job end is signaled using the following signals:

- DB1200.DBX5000.0
- DB1200.DBX5000.1

The signals are written by the PLC; therefore, they can only be read by the user.

A job has been completed as soon as the user resets the "Start" signal (DB1200.DBX4000.1). Status signals DB1200.DBX5000.0 and .1 are then set to zero.

DB1200		**PI service [r]**						
		NCK → PLC interface						
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 5000	-	-	-	-	-	-	Error in the job 2)	Job completed 1)
DBB 5001	-							
DBB 5002	-							
1) DB1200.DBX5000.0, job status: DBX5000.0 == 1 ⇒ job completed								
2) DB1200.DBX5000.1, error status: DBX5000.1 == 0 ⇒ no error; DBX5000.1 == 1 ⇒ error								

Possible error causes

- The index of the parameterized PI service (DB1200.DBB4001) is outside the valid range
- Parameter error

15.6.2.3 PI service ASUB

Interrupts

Note

Relationship between channels and interrupts

For control systems with maximum **one** channel, **two** interrupts are available; for control systems with maximum **two** channels, **four** interrupts are available.

Every interrupt number can be assigned an interrupt program (ASUB) with the "ASUB" PI service from the PLC. The interrupt program is then executed on the NC when the associated interrupt is issued. The associated interrupt programs must be present in the NC in the machine manufacturer directory (_N_CMA_DIR) with the following program names:

Interrupt	Program name
1	PLCASUP1_SPF
2	PLCASUP2_SPF
3	PLCASUP3_SPF
4	PLCASUP4_SPF

If the interrupt programs are not available, they must be created. An NC Reset (Power On) must then be issued on the NC.

The PI service "ASUP" must only be executed once per interrupt assignment after the control powers up. The assignment of the interrupt to the interrupt program is retained until the control powers up again.

Job specification

PI service: ASUP		
Address	**Meaning**	**Valid values**
DB1200.DBW4001	PI index 1)	1, 2, 13, 14
DB1200.DBW4004	Parameter 1: LIFTFAST 2)	0 (FALSE), 1 (TRUE)
DB1200.DBW4006	Parameter 2: BLSYNC 3)	0 (FALSE), 1 (TRUE)
DB1200.DBW4008	Parameter 3: Channel index 4)	0, 1
DB1200.DBW4010	Parameter 4: Interrupt priority 5)	0, 1, 2, 3, 4
1) • PI index = 1: Interrupt 1 ⇒ _N_CMA_DIR / PLCASUP1_SPF, interrupt priority set as default: 1 • PI index = 2: Interrupt 2 ⇒ _N_CMA_DIR / PLCASUP2_SPF, interrupt priority set as default: 2 • PI index = 13: Interrupt 3 ⇒ _N_CMA_DIR / PLCASUP3_SPF, interrupt priority set as default: 3 • PI index = 14: Interrupt 4 ⇒ _N_CMA_DIR / PLCASUP4_SPF, interrupt priority set as default: 4		
2) After the interrupt has been initiated, LIFTFAST first results in a fast retraction (fast lift). Only then is the interrupt routine executed. **References** Function Manual, Special Functions; Section "R3: Extended standstill and retract " > "Control-managed ESR" > "Retract"		
3) BLSYNC ensures that after the interrupt has been initiated, the system first waits until the actual block has been executed. Only then is the interrupt routine executed.		
4) 0 → channel 1, 1 → channel 2		
5) When using default interrupt priorities, the parameter should be set to a value of 0.		

Machine data

- Lowest interrupt priority
  In the following machine data, the lowest interrupt priority for the NC is defined, whose associated interrupt is processed. Interrupts with a lower priority than that specified in the machine data are not processed in the control system:
  MD11604 $MN_ASUP_START_PRIO_LEVEL

Supplementary conditions

Channel status

The PI service "ASUB" may only be executed when the channel in which it is requested is in the "reset" state.

ProgEvent "Power up"

If, for an event-driven program call (ProgEvent), "power up" is configured as initiating event (MD20108 $MC_PROG_EVENT_MASK), then the PI service "ASUP" may only be started after the ProgEvent program has been completed (PROG_EVENT_SPF or MD11620 $MN_PROG_EVENT_NAME = <user_prog_event_SPF>).

See also

Job specification (Page 1168)

Job feedback (Page 1169)

15.6.2.4 PI service LOGOUT

Function

The password last transferred to the NC is reset.

Job specification

PI service: LOGOUT		
Address	**Description**	**Valid values**
DB1200.DBW4001	PI index	3 (reset password)

15.6.2.5 PI service DATA_SAVE

Function

Save the current NC state to the system CompactFlash card.

Note

Powering up

The next time that the control powers up, in the "Start up menu" via "Reload saved user data", the saved state can be loaded to the NC.

Job specification

PI service: DATA_SAVE		
Address	**Description**	**Valid values**
DB1200.DBW4001	PI index	4

References
SINUMERIK 828D Commissioning Manual; Section "CNC commissioning" > "Scope of delivery and requirements" > "Control run-up"

15.6.2.6 PI service TMMVTL

Function

Using the PI service TMMVTL, it is possible to request a job from the PLC to relocate a tool. As a result of the PI service, the tool manager carries out an empty location search in the target magazine for the tool specified in the PI service (tool number or source location number/source magazine number) The PLC then receives a job to relocate the tool via DB41xx.DBB0.

Job specification

Address	Description	Valid values
DB1200.DBW4001	PI index	5
DB1200.DBW4004	Parameter 1: Tool number 1)	-1, 1 … 31999
DB1200.DBW4006	Parameter 2: Source location number 1)	-1, 1 … 31999
DB1200.DBW4008	Parameter 3: Source magazine number 1)	-1, 1 … 9999
DB1200.DBW4010	Parameter 4: Target location number 2)	-1, 1 … 31999
DB1200.DBW4012	Parameter 5: Target magazine number 3)	-1, 1 … 9999

1) The tool can be optionally specified using:
- Tool number (T number)
- Source location and source magazine number

For the unused parameters of the other version, a value of -1 must be entered.

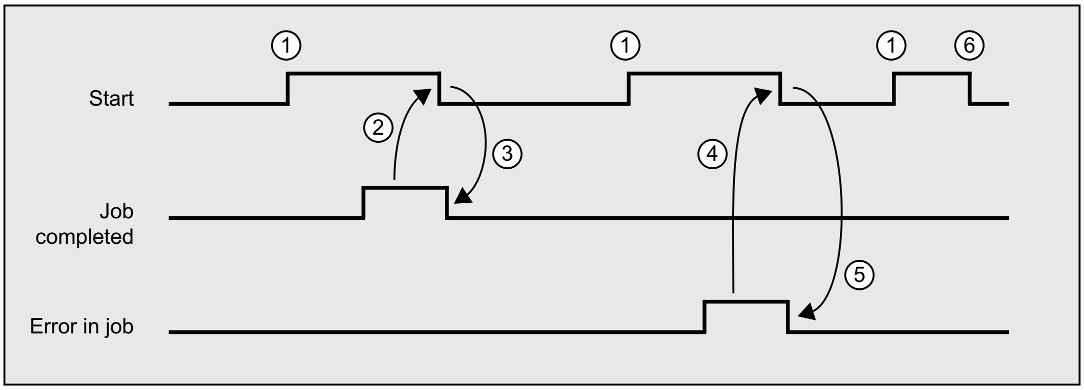
2) With the target location number = -1, a search is made in the complete magazine for an empty location for the tool according to the search strategy that has been selected. If a target location is specified, then a check is made as to whether the location with the specified target location number is free and suitable for the particular tool.

3) For a target magazine number = -1, a search is made in a buffer for the tool corresponding to the assignment obtained from $TC_MDP2.

Application examples

- When using buffers to return the tool (for example Toolboy and/or shifter), an explicit empty location search in the magazine may be needed during the asynchronous return transport. In this case, the PLC does not have to note the original location, the PI service TMMVTL searches for a suitable location.
- A tool is to be moved from a background magazine to the front magazine.

15.6.2.7 PI services: Cycle diagram



①	User sets the signal "Start", job execution starts.
②	After the PLC firmware signals "Job completed", the user resets the signal "Start".
③	By resetting the signal "Start", the PLC firmware resets the signal "Job completed".
④	After the PLC firmware signals "Error in job", the user resets the signal "Start".
⑤	By resetting the signal "Start", the PLC firmware resets the signal "Error in job".
⑥	If the user accidentally resets the signal "Start" before one of the signals "Job completed" or "Error in job" is received, then the result signals for this job are not updated. However, the job is executed.

15.6.3 PLC user alarms

15.6.3.1 User interface

Note

Although the designation user "*alarms*" is used in the following, it defines only whether a **message** or an **alarm** is involved when entering the associated **cancel criterion** (see "Configuring user alarms (Page 1176)").

The user interface in DB1600 offers the option of displaying error and operational messages on the HMI.

This includes the following deliverables:

- Activation of the user alarms 700000 to 700247 and the extended user alarms 701000 to 701999.

Note

The use of the extended PLC user alarms 701000 to 701999 is subject to the following preconditions:

- Compatibility mode must be deactivated.
- The data block DB9913 must be contained in the PLC project (i.e. DB9913 was selected in the PLC Programming Tool under "Libraries" > "Special data blocks" and transferred with Copy & Paste / double-click to the PLC project).

- The user alarms 700000 to 700247 and 701000 to 701247 can be given an additional numeric parameter.
- Deactivation and acknowledgment of the user alarms.
- Evaluation of the system responses initiated by the user alarms.

The firmware evaluates the signals that have been entered and sends these as coming and going alarms and messages to the HMI where they are displayed. The HMI manages the error texts.

15.6.3.2 Activation interface of the user alarms

Each user alarm is activated using its assigned activation bit. These bits are set in the **activation interface.**

A new user alarm is activated with a 0/1 edge of the particular bit.

Activation interface for alarms 700000 to 700247

DB1600	Alarm activation [r/w] (PLC → HMI)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Activation of alarm no.							
DBB0	700007	700006	700005	700004	700003	700002	700001	700000
	Activation of alarm no.							
DBB1	700015	700014	700013	700012	700011	700010	700009	700008
	Activation of alarm no.							
DBB2	700023	700022	700021	700020	700019	700018	700017	700016
	Activation of alarm no.							
DBB3	700031	700030	700029	700028	700027	700026	700025	700024
	Activation of alarm no.							
DBB4	700039	700038	700037	700036	700035	700034	700033	700032
	Activation of alarm no.							
DBB5	700047	700046	700045	700044	700043	700042	700041	700040
...	….							

DB1600	Alarm activation [r/w] (PLC → HMI)							
	Activation of alarm no.							
DBB30	700247	700246	700245	700244	700243	700242	700241	700240

Activation interface for alarms 701000 to 701999

DB1600	Alarm activation [r/w] (PLC → HMI)							
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Activation of alarm no.							
DBB4000	701007	701006	701005	701004	701003	701002	701001	701000
	Activation of alarm no.							
DBB4001	701015	701014	701013	701012	701011	701010	701009	701008
	Activation of alarm no.							
DBB4002	701023	701022	701021	701020	701019	701018	701017	701016
	Activation of alarm no.							
DBB4003	701031	701030	701029	701028	701027	701026	701025	701024
	Activation of alarm no.							
DBB4004	701039	701038	701037	701036	701035	701034	701033	701032
	Activation of alarm no.							
DBB4005	701047	701046	701045	701044	701043	701042	701041	701040
...	….							
	Activation of alarm no.							
DBB4124	701999	701998	701997	701996	701005	701994	701993	701992

15.6.3.3 Variables interface of the user alarms

The user alarms 700000 to 700247 and 701000 to 701247 can be given a variable as parameter. A double word is reserved for each in the **variable interface**. As a consequence, valid offsets must be divisible by 4.

Variable interface for alarms 700000 to 700247

DB1600	Variable for alarm [r32/w32] (PLC → HMI)							
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBD1000	Variable for alarm 700000							
DBD1004	Variable for alarm 700001							
DBD1008	Variable for alarm 700002							
	...							
DBD1980	Variable for alarm 700245							
DBD1984	Variable for alarm 700246							
DBD1988	Variable for alarm 700247							

Variable interface for alarms 701000 to 701247

DB1600	Variable for alarm [r32/w32] (PLC → HMI)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBD5000	Variable for alarm 701000							
DBD5004	Variable for alarm 701001							
DBD5008	Variable for alarm 701002							
	...							
DBD5980	Variable for alarm 701245							
DBD5984	Variable for alarm 701246							
DBD5988	Variable for alarm 701247							

15.6.3.4 Configuring user alarms

The following attributes can be specified for each alarm:

- Alarm response: How the control system responds when an error occurs.
- Cancel criterion: What must be done to cancel the alarm again or acknowledge it. The cancel criterion simultaneously defines the alarm type and priority.
- Channel assignment The channel to which the alarm is assigned.

User alarms 700000 to 700247

User alarms 700000 to 700247 are configured via machine data.

Alarm response and cancel criterion

The specification of the alarm reactions and the cancel criterion are coded as bits in the machine data:

MD14516 $MN_USER_DATA_PLC_ALARM [x] = <Alarm response and cancel criterion>

with x = user alarm number - 700000; value range: 0 ≤ x ≤ 247

<Alarm response and cancel criterion>	
Bit	**Meaning**
Alarm responses	
0	NC Start disable
1	Read-in disable
2	Feed disable for all axes
3	Emergency stop
4	PLC STOP
5	Alarm log
Cancel criteria	
6	Interrupt with DB1600 DBX3000.0
7	Power On
Bit x = 1: Activating the functionality Bit x = 0: Deselecting the functionality	

Channel assignment

The channel is assigned as bit coding in the machine data:

MD14518 $MN_USER_DATA_PLC_ALARM_ASSIGN [x] = <channel assignment>

with x = user alarm number - 700000; value range: 0 ≤ x ≤ 247

<Channel assignment>	
Bit	**Meaning**
0	Scope NC-channel 1
1	Scope NC-channel 2
Bit x = 1: Channel selected Bit x = 0: Channel deselected	

User alarms 701000 to 701999

User alarms 701000 to 701999 are configured in data block DB9913 (ALARM_INI). Each alarm requires two bytes of configuration data. One byte for selecting the alarm reactions and cancel criteria, and one byte for the channel assignment. These two bytes are grouped as one word.

DB9913	Configuring user alarms 701000 to 701999 [r]
DBW0	Alarm 701000
DBW2	Alarm 701001
DBW4	Alarm 701002
	...
DBW1998	Alarm 701999

Bit	Meaning
Alarm response	
0	NC Start disable
1	Read-in disable
2	Feed disable for all axes
3	Emergency stop
4	PLC STOP
5	Alarm log
Cancel criterion	
6	Interrupt with DB1600 DBX3000.0
7	Power On
Channel assignment	
8	PLC user alarm for NC-channel 1
9	PLC user alarm for NC-channel 2
10	Reserved
11	Reserved
12	Reserved
13	Reserved

Bit	Meaning
14	Reserved
15	Reserved

Cancel criterion and priority

The cancel criterion and therefore implicitly also the type and priority of a user alarm is set using bits 6 and 7:

Bit 7	Bit 6	Cancel criterion:	Type	Priority
0	0	Resetting of the activation bit	Message	Low
0	1	Acknowledgment in DB1600.DBX3000.0 (see "Acknowledgement interface of the user alarms (Page 1179)")	Alarm	Medium
1	0	Power On	Alarm	High
1	1	Reserved (internally evaluated as: Bit 7 = 1, bit 6 = 0)	-	-

Display message

If an alarm response is not activated (machine data bits 0 to 4 = 0) for one of the user alarms listed above, then this defines that it involves what is known as a "Display message" without having any effect on the system. This especially indicates that also the cancel criterion (machine data bits 6 and 7) of the corresponding machine data is not evaluated.

15.6.3.5 Export active alarm responses and cancel criteria

The present **active alarm responses** (i.e. the actual responses), the**active cancel criteria** and the **active channel assignment** can be exported globally from the interface.

DB1600	Active alarm response [r]							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB2000	Power On	Interrupt with DB1600 DBX3000.0		PLC STOP	Emergency stop	Channel 1		
						Feedrate disable for all axes	Read-in disable	NC start disable
DBB2001						Channel 2		
						Feed disable for all axes	Read-in disable	NC Start disable

One bit is set if for at least one active alarm the corresponding response or the corresponding cancel criterion is configured. It is canceled if this response/cancel criterion is no longer configured for any of the pending alarms.

15.6.3.6 Acknowledgement interface of the user alarms

Requirement to acknowledge a user alarm is that the corresponding activation bit is reset.

- Messages with cancel criterion {0,0} then disappear automatically from the display.
- Alarms with cancel criterion {0,1} are canceled by the acknowledgment bit *Ack*.
- Alarms with cancel criterion {1,0} are not influenced when the acknowledgment bit is set and can only be canceled by a Power On.

DB1600	**Alarm acknowledgment [r/w]**							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB3000								Ack
DBB3001								
DBB3002								
DBB3003								

15.6.3.7 Interface to HMI.

The PLC can transfer eight messages or alarms for display on the HMI, which are displayed in the sequence that they occur.

When additional messages/alarms occur, the first seven are kept in the HMI, and the latest message or the latest alarm is displaced from one that has just occurred according to the following rules:

- System message/alarm displaces user message/alarm.
- Messages/alarms with a higher priority displace those of a lower priority.

The first seven messages/alarms are kept in the display because it is very probable that these define the cause of the problem and the following are just of a secondary nature. However, if one or several messages/alarms are acknowledged and therefore cleared, then a corresponding number of alarms/messages that have been received move up in the HMI.

15.6.4 PLC axis control

15.6.4.1 General information

The PLC can control axes/spindles via data blocks of the user interface; the axis/spindle is specified by its DB number:

- **DB380x** PLC → NC interface (to axis/spindle)
- **DB390x** NC → PLC interface (from axis/spindle)

with axis index x: 0 ≤ x ≤ max. axis index; axis index = axis number - 1

The following functions are supported:

- Positioning axes
- Position spindle
- Rotate spindle
- Oscillate spindle
- Indexing axes

References

- Function Manual, Extended Functions; Positioning Axes (P2) and Indexing axes (T1)
- Function Manual, Basic Functions; Spindles (S1)

Requirement

The axis to be controlled must be assigned to the PLC (PLC axis). An axis can be interchanged between NC and PLC using the user interface "*Axis interchange*" (DB3800.DBB8/ DB3900.DBB8).

Function start

Each function is activated by the positive edge of the corresponding "Start" signal. This signal must remain a logical "1" until the function has been positively or negatively acknowledged (e.g. using *Position reached* = "1" or *Error* = "1"). The signal "Positioning axis active" = "1" indicates that the function is active and that the output signals are valid.

Abort

It is **not** possible to **interrupt** the function by resetting the start signal, but only via other interface signals (using the axis-specific signal *Delete distance to go/spindle reset*, DB380x DBX2.2).

The axis interface returns axis status signals that may need to be evaluated (e.g. *exact stop*, *travel command*, → DB390x).

If the axis/spindle is being traversed via the NC program when the PLC axis control is called (travel command present), then the function is only started after this traversing motion has ended. No error code is output in this situation.

Axis disable

With the **axis disable** set, an axis controlled via PLC axis control will not move. Only a simulated actual value is generated. (Behavior as with NC programming).

15.6.4.2 User interface: Preparing the NC axis as PLC axis

Request signals to axis/spindle (excerpt)

Firstly, the axis/spindle must be requested from the PLC:

DB380x	Signals to the axis/spindle (PLC → NC) [r/w]							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB8	Request PLC axis/ spindle			Activation signal when this byte is changed			Assign NC axis/spindle channel	
							B	A

A change of a request signal (DB380x.DBX8.7 or 8.0) must be notified to the NC via a 0→1 edge of the activation signal (DB380x.DBX8.4). After a PLC cycle, the activation signal must be reset again.

Status signals from an axis/spindle (excerpt)

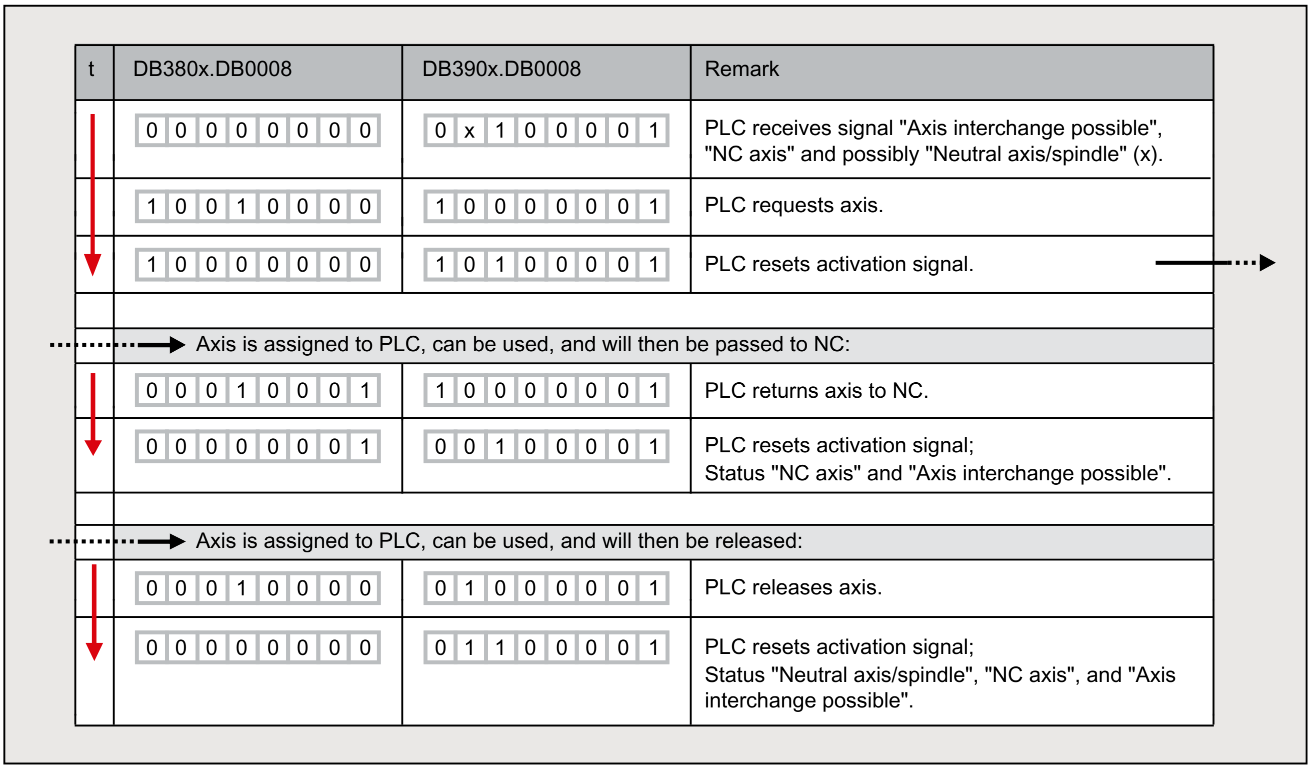
DB390x	Signals from the axis/spindle (NC → PLC) [r]							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB8	PLC axis/ spindle	Neutral axis/ spindle	Axis interchange possible	New type requested from PLC			Current assignment of the NC axis/spindle in channel	
							B	A

Note

Simulation

To activate the interface signals, the machine data MD30350 $MA_SIMU_AX_VDI_OUTPUT must be set for each required axis during the simulation.

Request and relinquish PLC axis



t	DB380x.DB0008	DB390x.DB0008	Remark
	0 0 0 0 0 0 0 0	0 x 1 0 0 0 0 1	PLC receives signal "Axis interchange possible", "NC axis" and possibly "Neutral axis/spindle" (x).
	1 0 0 1 0 0 0 0	1 0 0 0 0 0 0 1	PLC requests axis.
	1 0 0 0 0 0 0 0	1 0 1 0 0 0 0 1	PLC resets activation signal.
	Axis is assigned to PLC, can be used, and will then be passed to NC:		
	0 0 0 1 0 0 0 1	1 0 0 0 0 0 0 1	PLC returns axis to NC.
	0 0 0 0 0 0 0 1	0 0 1 0 0 0 0 1	PLC resets activation signal; Status "NC axis" and "Axis interchange possible".
	Axis is assigned to PLC, can be used, and will then be released:		
	0 0 0 1 0 0 0 0	0 1 0 0 0 0 0 1	PLC releases axis.
	0 0 0 0 0 0 0 0	0 1 1 0 0 0 0 1	PLC resets activation signal; Status "Neutral axis/spindle", "NC axis", and "Axis interchange possible".

15.6.4.3 User interface: Functionality

The two tables provide an overview of the available interface signals. The precise description of the signals and the explanation of what signals are relevant for the individual functions are explained in the following.

Signals to PLC axis

DB380x	Signals to the PLC axis (PLC → NC) [r/w]							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB3000	Start positioning axis	Start spindle positioning	Start spindle rotation	Start spindle oscillation	-	-	-	-
DBB3001	-	-	Stop spindle rotation	Stop spindle oscillation	-	-	-	-
DBB3002	Automatic gear selection	Constant cutting rate	Direction of rotation as for M4	-	Handwheel override	Traversing dimension, inches (not metric)	Distance condition, shortest distance (DC)	Distance condition, incremental (IC)

DB380x	Signals to the PLC axis (PLC → NC) [r/w]							
DBB3003	Indexing position	-	-	-	-	-	Distance condition, abs. pos. direction (ACP)	Distance condition, abs. neg. direction (ACN)
DBD3004	Position (REAL, with indexing axis: DINT)							
DBD3008	Feedrate velocity (REAL), if < 0, the value is taken from machine data POS_AX_VELO							

The bits of the distance conditions and the direction of rotation definition define the particular positioning or traversing mode, only one of the bits must be set:

Meaning	Distance condition to be set
Positioning absolute	No mode bit set
Positioning incremental	DBB3002.0 = 1
Positioning shortest distance	DBB3002.1 = 1
Positioning absolute, positive approach direction	DBB3003.1 = 1
Positioning absolute, negative approach direction	DBB3003.0 = 1
Direction of rotation as for M4	DBB3002.5 = 1

The remaining bits are used to specify and start the particular function, these function bits as well as position and velocity are explained in more detail for the individual functions.

Signals from PLC axis

DB390x	Signals from the PLC axis (NC → PLC) [r]							
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB3000	Positioning axes active	Position reached		-	-	-	Error while traversing	Axis cannot be started
DBB3001	-	-	-	-	-	-	-	-
DBB3002	-	-	-	-	-	-		-
DBB3003	Error number							

The following requirements must be satisfied in order to use the functions listed below:

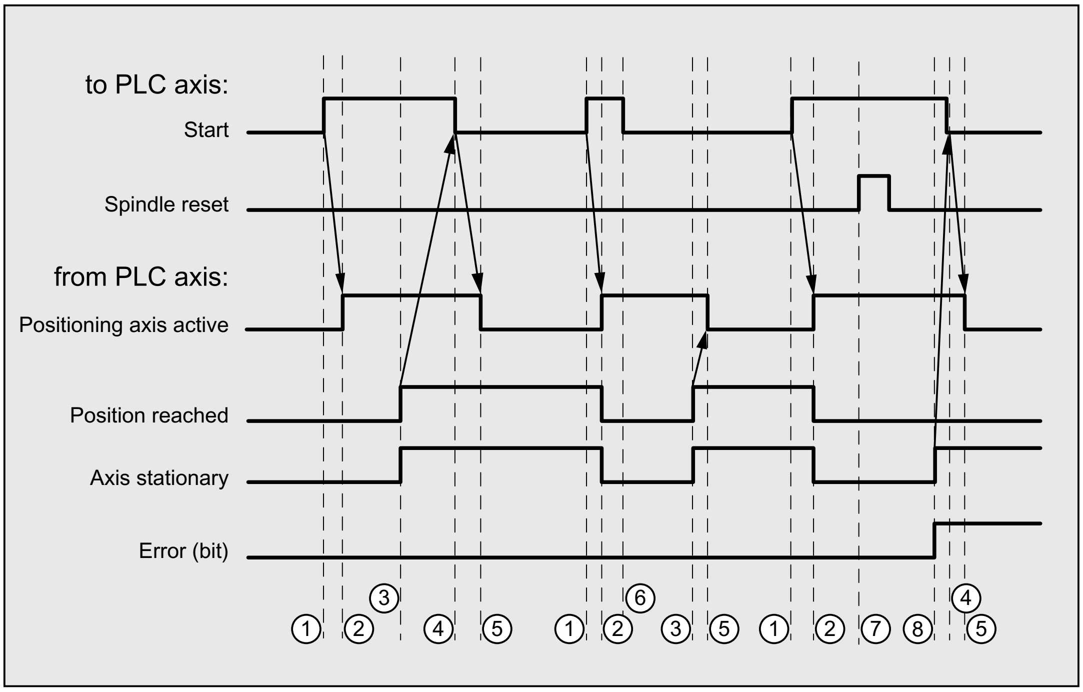
- The axis or spindle is correctly assigned to the PLC.
- Controller and pulse enable are set.
- After setting all of the control signals, only one of the start signals is set in DB380x.DBB3000.

15.6.4.4 Spindle positioning

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	Only one of the four signals may be set concurrently. All signals 0: Absolute positioning
DBX3002.1	Traverse along the shortest path	
DBX3003.0	Absolute, negative direction	
DBX3003.1	Absolute, positive direction	

DB380x	PLC → NC control signals	Remark
DBD3004	Position setpoint / distance setpoint	Distance setpoint if DBX3002.0 == 1
DBD3008	Feedrate	0: Traverse with the value from MD35300 $MA_SPIND_POSCTRL_VELO
DBX3000.6	Start	**Note** Reset of the signal does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	Interrupt signal, exits the function

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	Also 1 when override = 0 or position setpoint reached when start = 1
DBX3000.6	Position reached	1: Position setpoint reached with "Exact stop fine"
DBX3000.0	Spindle cannot be started	
DBX3000.1	Error while traversing	1: Error during traversing, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	1: if $n < n_{min}$

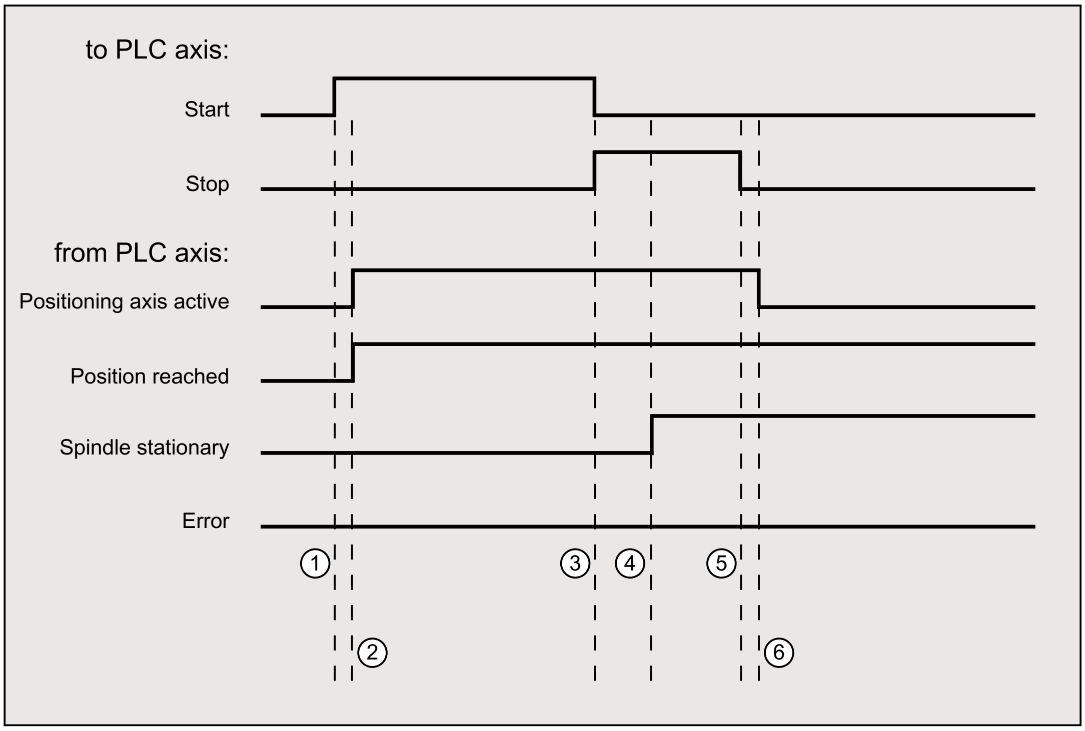


①	Function activated by user with a positive edge of *Start*.
②	*Positioning axis active* message shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* may be withdrawn. For path specification = 0, the signals are not canceled.
③	When the position is reached this is signaled (*Position reached*), *Spindle stationary* is set.
④	The user then withdraws *Start*.
⑤	The *Positioning axis active* signal is then reset.
⑥	The user immediately resets the *Start* signal with receipt of the *Positioning axis active* signal.
⑦	Positioning is aborted by setting *Spindle reset*. This signal must be present for at least one PLC cycle.
⑧	The spindle comes to a standstill (*Spindle stationary*), the *Error* signal is set. (In this case, error number 115 is output.)

15.6.4.5 Rotate spindle

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	-
DBX3002.1	Traverse along the shortest path	-
DBX3002.5	Direction of rotation as for M4	1: Direction of rotation specified by M4 0: Direction of rotation specified by M3
DBX3003.0	Absolute, negative direction	-
DBX3003.1	Absolute, positive direction	-
DBD3008	Feedrate	Spindle speed
DBX3000.5	Start spindle rotation	-
DBX3001.5	Stop spindle rotation	-

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	1: For start or stop == 1
DBX3000.6	Position reached	1: Function was started without an error
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-
DBX1.4	Axis/spindle stationary	-



① Function activated by user with a positive edge of *Start*.

② Messages *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle rotation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

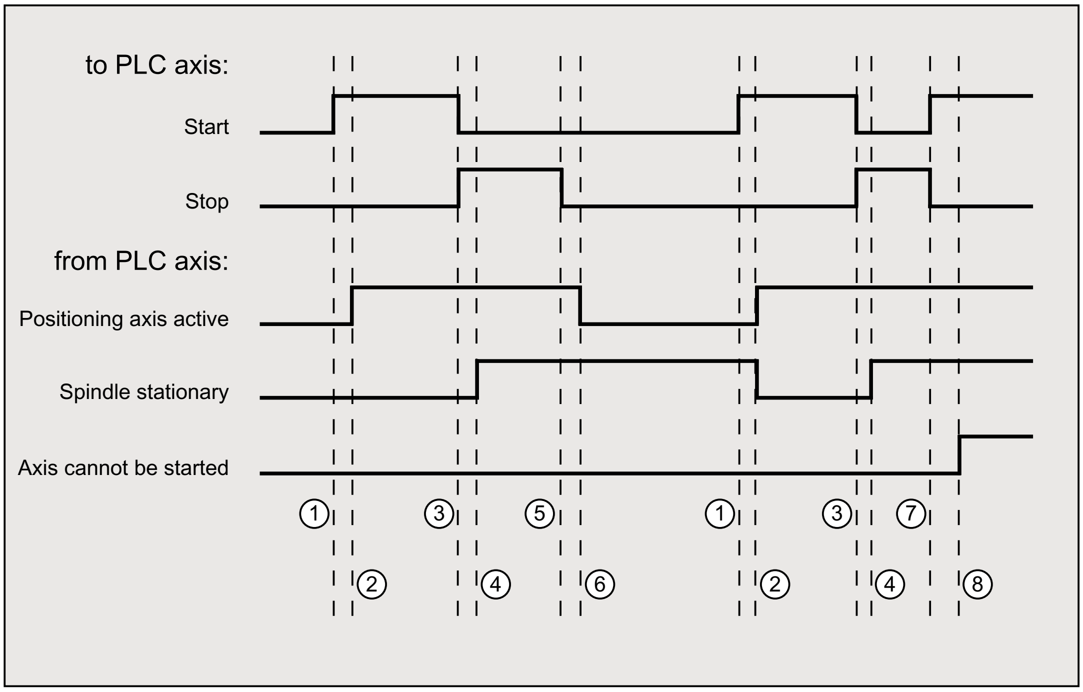
⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

15.6.4.6 Oscillate spindle

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental	It is not permissible that any of the bits are set.
DBX3002.1	Shortest path	
DBX3002.5	Direction of rotation as for M4	
DBX3003.0	Absolute, negative direction	
DBX3003.1	Absolute, positive direction	
DBD3004	Setpoint gear stage	**MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 0** 0 - 5: Oscillation **MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 1** 0: Oscillation 1: Oscillation with gear stage change M41 2: Oscillation with gear stage change M42 3: Oscillation with gear stage change M43 4: Oscillation with gear stage change M44 5: Oscillation with gear stage change M45
DBD3008	Feedrate	When oscillating, no significance! The oscillation speed is taken from machine data MD35400, $MA_SPIND_OSCILL_DES_VELO.
DBX3000.5	Start spindle oscillation	It is not permissible that the start directly follows a stop. Stop must first be reset (both 0).
DBX3001.5	Stop spindle oscillation	

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	1: For start or stop == 1
DBX3000.6	Position reached	1: after start
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBX3000.0	Axis cannot be started	1: Error when starting, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	



①	Function activated by user with a positive edge of *Start*.
Note: This is possible only when the *Positioning axis active* signal is reset!

②	Messages *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant and is therefore not shown.

③	The user stops spindle oscillation by resetting *Start* and setting *Stop*.

④	The spindle stops and the *Spindle stationary* signal is set.

⑤	The user then resets *Stop*.

⑥	Reset of *Stop* causes *Positioning axis active* to be reset.

⑦	*Stop* is reset in the user program and *Start* is again set, incorrectly, in the same PLC cycle. This means that *Positioning axis active* is not reset, but...

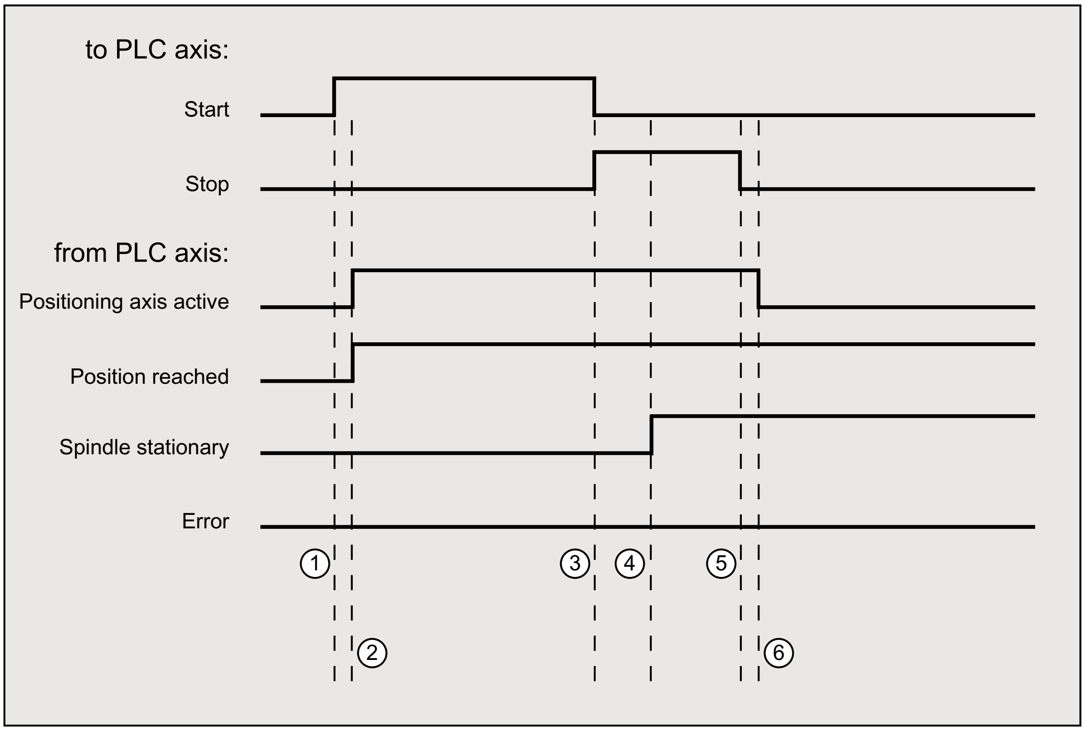
⑧	...the *Axis cannot be started* signal is set (error number 106).

15.6.4.7 Indexing axis

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	Only one of the signals may be set concurrently. All signals 0: Absolute positioning
DBX3002.1	Traverse along the shortest path	
DBX3002.5	Direction of rotation as for M4	
DBX3003.0	Absolute, negative direction	
DBX3003.1	Absolute, positive direction	
DBX3003.7	Indexing position	Indexing axis ON
DBD3004	Position setpoint / distance setpoint	Distance setpoint if DBX3002.0 == 1
DBD3008	Feedrate	0: Traverse with the value from MD32060 $MA_POS_AX_VELO

DB380x	PLC → NC control signals	Remark
DBX3000.7	Start positioning axis	**Note** Reset of the signal does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	Interrupt signal, exits the function

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	Also 1, if override = 0 or position setpoint reached.
DBX3000.6	Position reached	1: Position setpoint reached with "Exact stop fine".
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-
DBX1.4	Axis/spindle stationary	-



① Function activated by user with a positive edge of *Start*.
Note: This is possible only when the *Positioning axis active* signal is reset!

② Messages *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle oscillation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

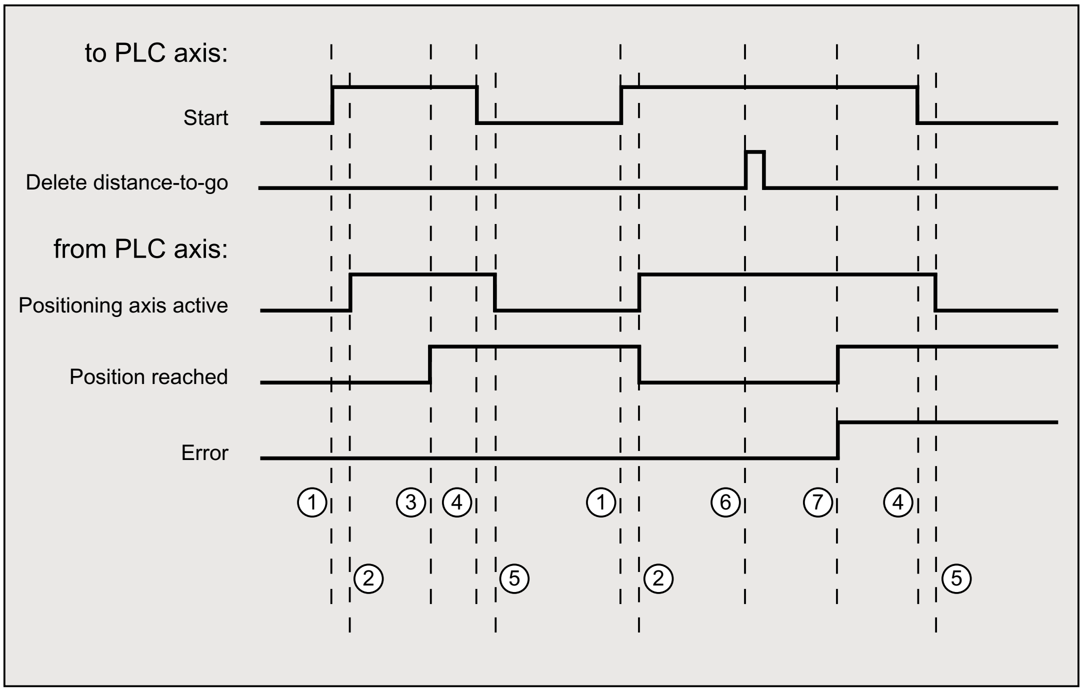
⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

15.6.4.8 Positioning axis metric

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	Only one of the signals may be set concurrently. All signals 0: Absolute positioning
DBX3002.1	Traverse along the shortest path	
DBX3002.5	Direction of rotation as for M4	
DBX3003.0	Absolute, negative direction	
DBX3003.1	Absolute, positive direction	
DBD3002.2	Traversing dimension inch	0: Traversing dimension, metric
DBX3002.3	Handwheel override	0: Override OFF
DBD3004	Position setpoint / distance setpoint	Distance setpoint if DBX3002.0 == 1
DBD3008	Feedrate	0: Traverse with the value from MD32060 $MA_POS_AX_VELO
DBX3000.7	Start positioning axis	**Note** Reset of the signal does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	Interrupt signal, exits the function

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	Also 1, if override = 0 or position setpoint reached.
DBX3000.6	Position reached	1: Axis has reached the position setpoint.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-



① First function activation using positive edge of *Start*.

② *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.

③ Positive acknowledgment *Position reached* = 1 and *Positioning axis active* = 1

④ Reset function activation after receipt of acknowledgment

⑤ Signal change via function

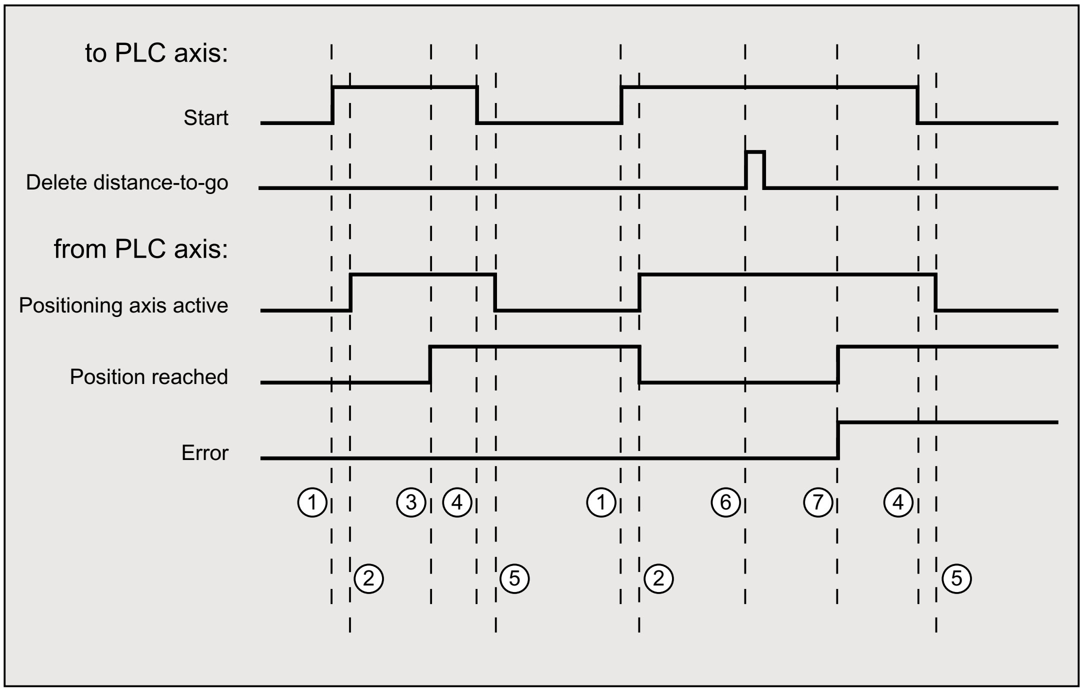
⑥ Positioning is aborted by delete distance-to-go, signal duration min. 1 PLC cycle.

⑦ The signals *Position reached* and *Error* are set, the *Error number* can be read (in this case, 30).

15.6.4.9 Positioning axis inch

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	Only one of the signals may be set concurrently. All signals 0: Absolute positioning
DBX3002.1	Traverse along the shortest path	
DBX3002.5	Direction of rotation as for M4	
DBX3003.0	Absolute, negative direction	
DBX3003.1	Absolute, positive direction	
DBX3002.2	Traversing dimension inch	0: Traversing dimension, metric
DBX3002.3	Handwheel override	0: Override OFF
DBD3004	Position setpoint / distance setpoint	Distance setpoint if DBX3002.0 == 1
DBD3008	Feedrate	0: Traverse with the value from MD32060 $MA_POS_AX_VELO
DBX3000.7	Start positioning axis	**Note** Reset of the signal does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	Interrupt signal, exits the function

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	Also 1, if override = 0 or position setpoint reached.
DBX3000.6	Position reached	1: Axis has reached the position setpoint.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-



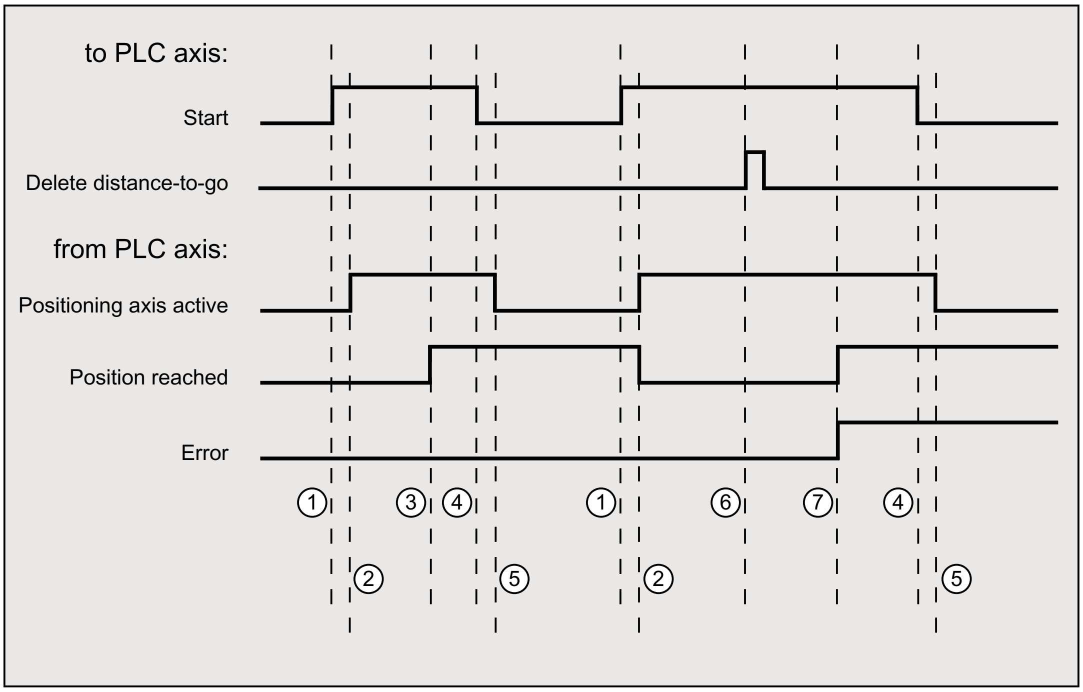
①	First function activation using positive edge of *Start*.
②	*Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.
③	Positive acknowledgment *Position reached* = 1 and *Positioning axis active* = 1
④	Reset function activation after receipt of acknowledgment
⑤	Signal change via function
⑥	Positioning is aborted by delete distance-to-go, signal duration min. 1 PLC cycle.
⑦	The signals *Position reached* and *Error* are set, the *Error number* can be read (in this case, 30).

15.6.4.10 Positioning axis metric with handwheel override

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	Only one of the signals may be set concurrently. All signals 0: Absolute positioning
DBX3002.1	Traverse along the shortest path	
DBX3002.5	Direction of rotation as for M4	
DBX3003.0	Absolute, negative direction	
DBX3003.1	Absolute, positive direction	

DB380x	PLC → NC control signals	Remark
DBX3002.2	Traversing dimension inch	0: Traversing dimension, metric
DBX3002.3	Handwheel override	0: Override OFF
DBD3004	Position setpoint / distance setpoint	Distance setpoint if DBX3002.0 == 1
DBD3008	Feedrate	0: Traverse with the value from MD32060 $MA_POS_AX_VELO
DBX3000.7	Start positioning axis	**Note** Reset of the signal does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	Interrupt signal, exits the function

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	Also 1, if override = 0 or position setpoint reached.
DBX3000.6	Position reached	1: Axis has reached the position setpoint.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-



① First function activation using positive edge of *Start*.

② *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.

③ Positive acknowledgment *Position reached* = 1 and *Positioning axis active* = 1

④ Reset function activation after receipt of acknowledgment

⑤ Signal change via function

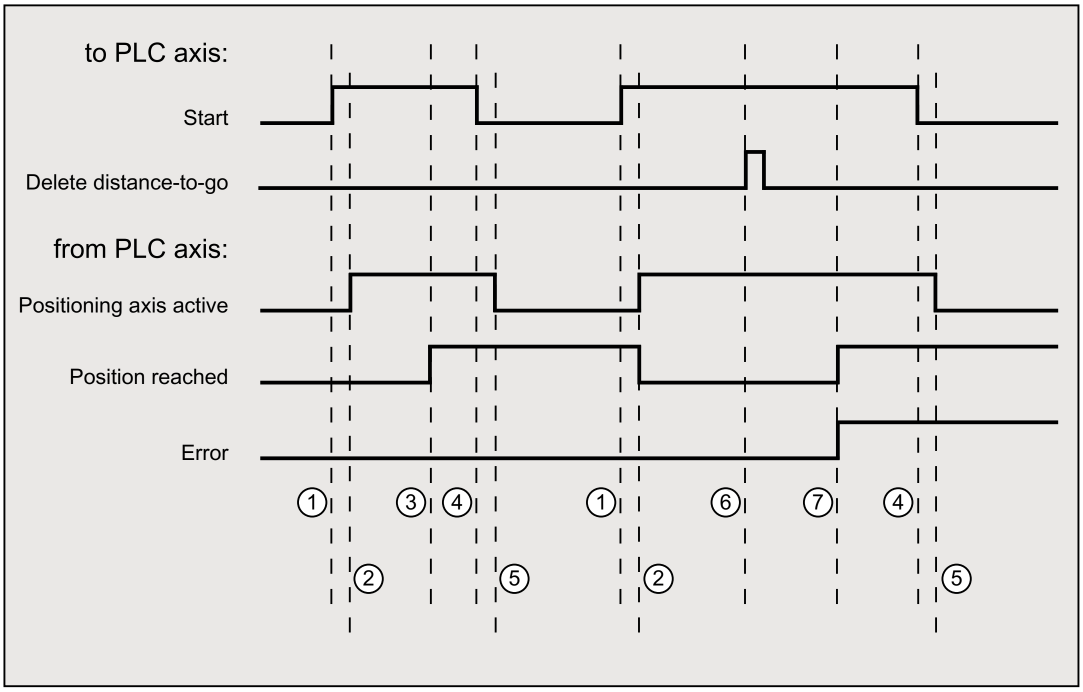
⑥ Positioning is aborted by delete distance-to-go, signal duration min. 1 PLC cycle.

⑦ The signals *Position reached* and *Error* are set, the *Error number* can be read (in this case, 30).

15.6.4.11 Positioning axis inch with handwheel override

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	Only one of the signals may be set concurrently. All signals 0: Absolute positioning
DBX3002.1	Traverse along the shortest path	
DBX3002.5	Direction of rotation as for M4	
DBX3003.0	Absolute, negative direction	
DBX3003.1	Absolute, positive direction	
DBX3002.2	Traversing dimension inch	0: Traversing dimension, metric
DBX3002.3	Handwheel override	0: Override OFF
DBD3004	Position setpoint / distance setpoint	Distance setpoint if DBX3002.0 == 1
DBD3008	Feedrate	0: Traverse with the value from MD32060 $MA_POS_AX_VELO
DBX3000.7	Start positioning axis	**Note** Reset of the signal does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	Interrupt signal, exits the function

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	Also 1, if override = 0 or position setpoint reached.
DBX3000.6	Position reached	1: Axis has reached the position setpoint.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-



① First function activation using positive edge of *Start*.

② *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.

③ Positive acknowledgment *Position reached* = 1 and *Positioning axis active* = 1

④ Reset function activation after receipt of acknowledgment

⑤ Signal change via function

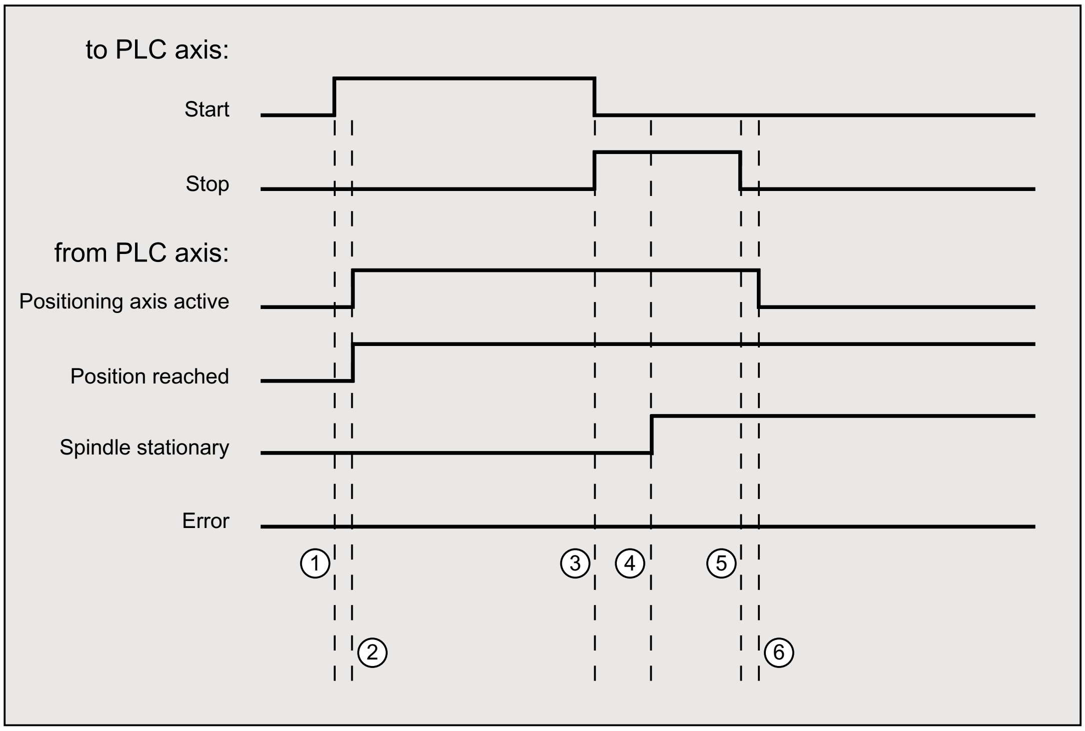
⑥ Positioning is aborted by delete distance-to-go, signal duration min. 1 PLC cycle.

⑦ The signals *Position reached* and *Error* are set, the *Error number* can be read (in this case, 30).

15.6.4.12 Rotate spindle with automatic gear stage selection

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	-
DBX3002.1	Traverse along the shortest path	-
DBX3002.5	Direction of rotation as for M4	1: Direction of rotation specified by M4 0: Direction of rotation specified by M3
DBX3003.0	Absolute, negative direction	-
DBX3003.1	Absolute, positive direction	-
DBX3002.7	Automatic gear stage selection	1: Automatic gear stage selection ON
DBD3008	Feedrate	Spindle speed
DBX3000.5	Start spindle rotation	-
DBX3001.5	Stop spindle rotation	-

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	1: For start or stop == 1
DBX3000.6	Position reached	1: Setpoint speed is output
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-
DBX1.4	Axis/spindle stationary	-



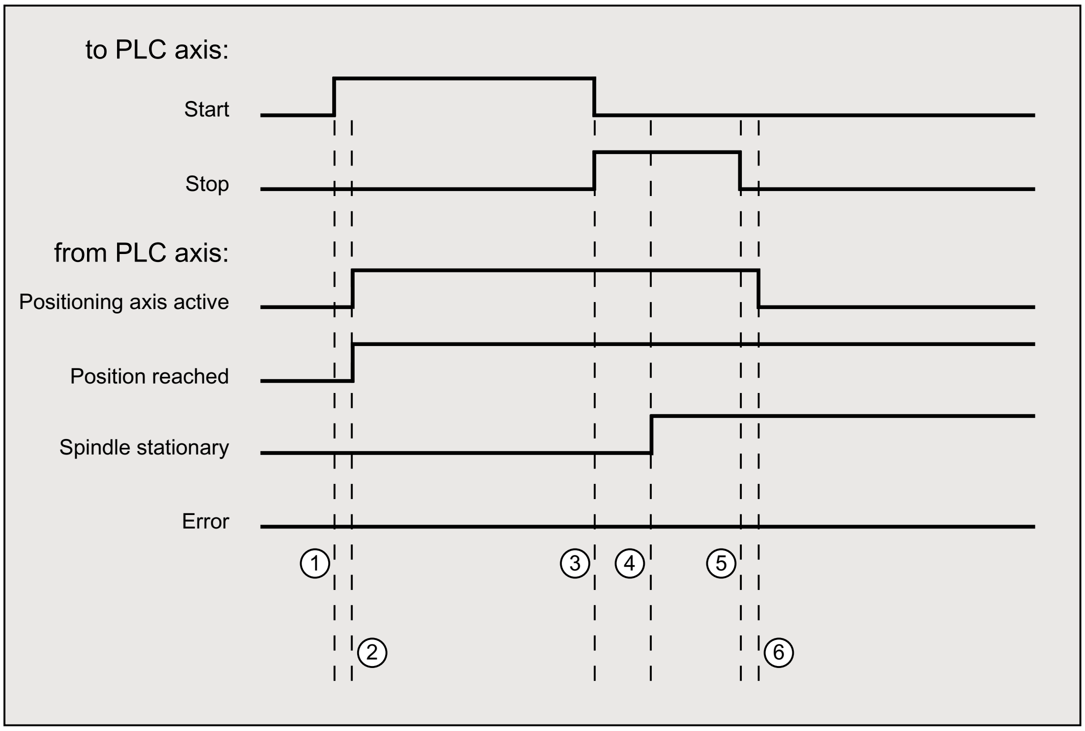
① Function activated by user with a positive edge of *Start*.
② Messages *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.
③ The user stops spindle rotation by resetting *Start* and setting *Stop*.
④ The spindle stops and the *Spindle stationary* signal is set.
⑤ The user then resets *Stop*.
⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

15.6.4.13 Rotate spindle with constant cutting rate [m/min]

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	-
DBX3002.1	Traverse along the shortest path	-
DBX3002.5	Direction of rotation as for M4	1: Direction of rotation specified by M4 0: Direction of rotation specified by M3

DB380x	PLC → NC control signals	Remark
DBX3003.0	Absolute, negative direction	-
DBX3003.1	Absolute, positive direction	-
DBX3002.2	Traversing dimension inch	**0**: Traversing dimension, metric
DBX3002.6	Const. Cutting rate	**1**: Constant cutting speed ON
DBD3008	Feedrate	Spindle speed
DBX3000.5	Start spindle rotation	-
DBX3001.5	Stop spindle rotation	-

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	1: For start or stop == 1
DBX3000.6	Position reached	1: Setpoint speed is output
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-
DBX1.4	Axis/spindle stationary	-



① Function activated by user with a positive edge of *Start*.

② Messages *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle rotation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

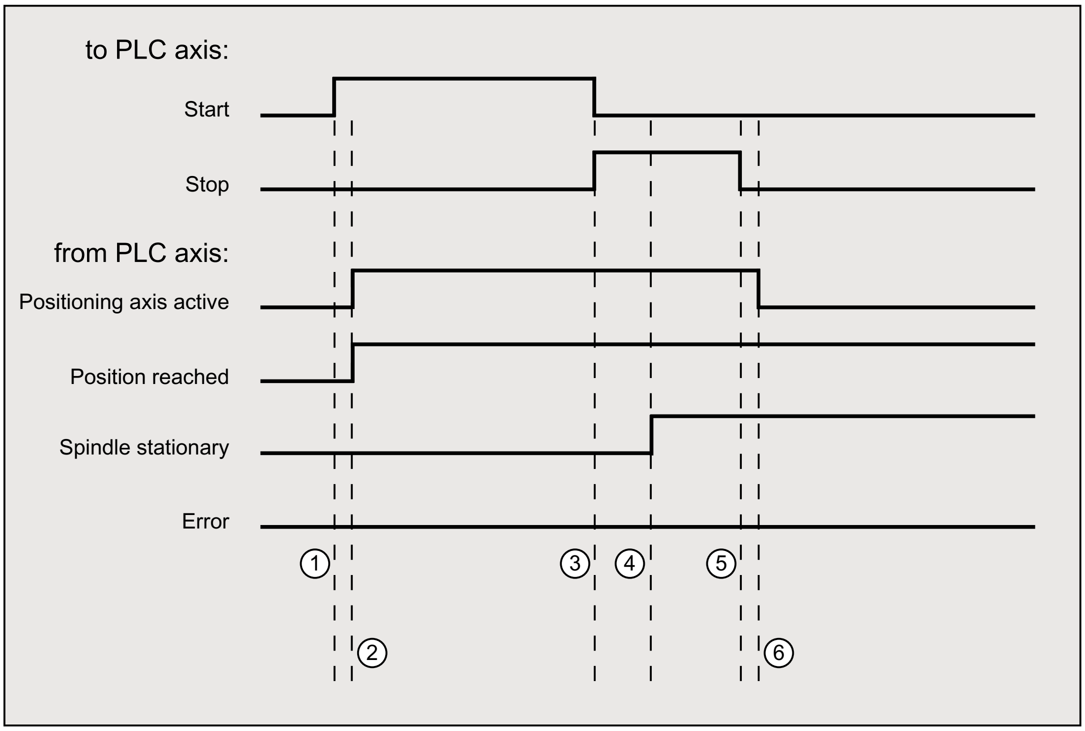
⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

15.6.4.14 Rotate spindle with constant cutting rate [feet/min]

DB380x	PLC → NC control signals	Remark
DBX3002.0	Incremental traversing	-
DBX3002.1	Traverse along the shortest path	-
DBX3002.5	Direction of rotation as for M4	1: Direction of rotation specified by M4 0: Direction of rotation specified by M3
DBX3003.0	Absolute, negative direction	-
DBX3003.1	Absolute, positive direction	-
DBX3002.2	Traversing dimension inch	1: Traversing dimension inch
DBX3002.6	Const. Cutting rate	1: Constant cutting speed ON
DBD3008	Feedrate	Spindle speed
DBX3000.5	Start spindle rotation	-
DBX3001.5	Stop spindle rotation	-

DB390x	NC → PLC status signals	Remark
DBX3000.7	Positioning axis active	1: For start or stop == 1
DBX3000.6	Position reached	1: Setpoint speed is output
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	-
DBX1.4	Axis/spindle stationary	-



① Function activated by user with a positive edge of *Start*.
② Messages *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.
③ The user stops spindle rotation by resetting *Start* and setting *Stop*.
④ The spindle stops and the *Spindle stationary* signal is set.
⑤ The user then resets *Stop*.
⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

15.6.4.15 Error messages

If a function could not be executed, the following signals are set depending on the error:

- DB390x .DBX3000.0 == 1 (axis cannot be started)
- DB390x.DBX3000.1 == 1 (error during travel)

The exact error cause is indicated as:

- DB390x.DBB3003 (error number)

Error number		Meaning
Decimal	Hex	
1	01	Several functions of the axis/spindle were activated simultaneously
20	14	A function was started without the position being reached
30	1E	The axis/spindle was transferred to the NC while still in motion
40	28	The axis is programmed by the NC program, NC internal error
50	32	Permanently assigned PLC axis: Traverses (JOG) or references

Error number		Meaning
Decimal	Hex	
60	3C	Permanently assigned PLC axis: Channel status does not permit a start
100	64	False position programmed for axis/spindle (alarm number1) 16830)
101	65	Programmed speed is too high
102	66	Incorrect value range for constant cutting rate (alarm number1) 14840)
104	68	Following spindle: Illegal programming (alarm number1) 22030)
105	69	No measuring system available (alarm number1) 16770)
106	6A	Positioning process of the axis still active (alarm number1) 22052)
107	6B	Reference mark not found (alarm number1) 22051)
108	6C	No transition from speed control to position control (alarm number1) 22050)
109	6D	Reference mark not found (alarm number1) 22051)
110	6E	Velocity/speed is negative
111	6F	Speed setpoint is zero
112	70	Invalid gear stage
115	73	Programmed position has not been reached
117	75	G96/G961 is not active in the NC
118	76	G96/G961 is still active in the NC
120	78	Axis is not an indexing axis (alarm number1) 20072)
121	79	Indexing position error (alarm number1) 17510)
125	7D	DC (shortest distance) not possible (alarm number1) 16800)
126	7E	Absolute value minus not possible (alarm number1) 16820)
127	7F	Absolute value plus not possible (alarm number1) 16810)
128	80	No transverse axis available for diameter programming (alarm number1) 16510)
130	82	Software limit switch plus (alarm number1) 20070)
131	83	Software limit switch minus (alarm number1) 20070)
132	84	Working area limitation plus (alarm number1) 20071)
133	85	Working area limitation minus (alarm number1) 20071)
134	85	Frame not permitted for indexing axis
135	87	Indexing axis with "Hirth joint" is active (alarm number1) 17501)
136	88	Indexing axis with "Hirth joint" is active and axis not referenced (alarm number1) 17503)
137	89	Spindle operation not possible for transformed spindle/axis (alarm number1) 22290)
138	8A	The corresponding effective coordinate-system-specific working area limit plus violated for the axis (alarm number1) 20082)
139	8B	The corresponding effective coordinate-system-specific working area limit minus violated for the axis (alarm number1) 20082)
200	C8	System alarm number1) 450007
1) The detailed alarm description is contained in: Alarms diagnostics manual; SINUMERIK 828D, SINAMICS S120		

15.6.5 Start ASUB

15.6.5.1 Job start

DB340x	ASUP: Job [r/w] (PLC → NC)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB0	-	-	-	-	-	-	-	Start INT1
DBB1	-	-	-	-	-	-	-	Start INT2
DBB2	-	-	-	-	-	-	-	Start INT3
DBB3	-	-	-	-	-	-	-	Start INT4

Interrupts

"Start INT1":

- DBX0.0 = 1: Request to start the interrupt program (ASUP) assigned to INT1.
- DBX0.0 = 0: Reset the ASUP request after acknowledgment in the result interface DB340x.DBB1000.0 - 3.

"Start INT2":

- DBX1.0 = 1: Request to start the interrupt program (ASUP) assigned to INT2.
- DBX1.0 = 0: Reset the ASUP request after acknowledgment in the result interface DB340x.DBB1001.0 - 3.

"Start INT3":

- DBX2.0 = 1: Request to start the interrupt program (ASUP) assigned to INT3.
- DBX2.0 = 0: Reset the ASUP request after acknowledgment in the result interface DB340x.DBB1002.0 - 3.

"Start INT4":

- DBX3.0 = 1: Request to start the interrupt program (ASUP) assigned to INT4.
- DBX3.0 = 0: Reset the ASUP request after acknowledgment in the result interface DB340x.DBB1003.0 - 3.

15.6.5.2 Job result

DB340x			ASUP: Result [r] (NC → PLC)					
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB1000	Signals to start INT1 (DB340x.DBX0.0)							
					ASUP cannot be executed [1)]	Interrupt no. not assigned [2)]	ASUP is being executed	ASUP completed [3)]

DB340x	**ASUP: Result [r] (NC → PLC)**							
DBB1001	Signals to start INT2 (DB340x.DBX1.0)							
					ASUP cannot be executed [1)]	Interrupt no. not assigned [2)]	ASUP is being executed	ASUP completed [3)]
DBB1002	Signals to start INT3 (DB340x.DBX2.0)							
					ASUP cannot be executed [1)]	Interrupt no. not assigned [2)]	ASUP is being executed	ASUP completed [3)]
DBB1003	Signals to start INT4 (DB340x.DBX3.0)							
					ASUP cannot be executed [1)]	Interrupt no. not assigned [2)]	ASUP is being executed	ASUP completed [3)]

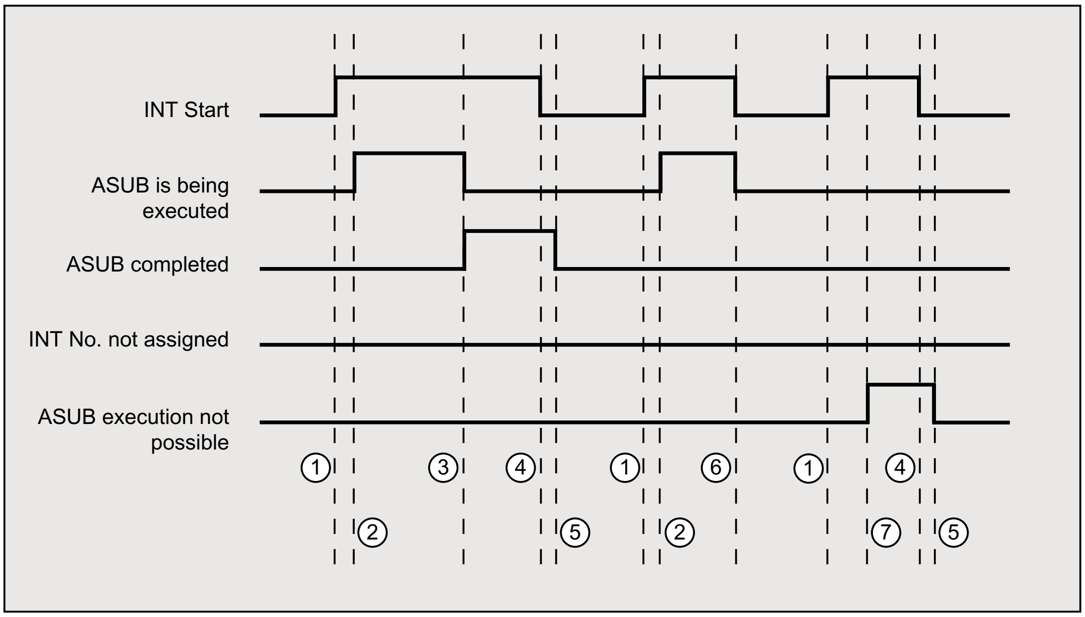
1) Negative acknowledgment: E.g. for emergency stop or channel reset request.

2) Negative acknowledgment: Number has not been assigned yet. Remedy: Execute PI service "ASUP".

3) Positive acknowledgment: ASUP successfully completed. Reset start signal: DB340x.:DBXn.0 = 0, with n = 0, 1, 2, 3

15.6.5.3 Signal flow

Signal flow



① Function activated by user with a positive edge of *Start*.

② *ASUP is being executed* is signaled back.

③ The acknowledgement *ASUP completed* indicates the successful execution, *ASUP is being executed* is withdrawn.

④ The signal to initiate the function is reset after receiving the acknowledgement from the user.

⑤ Signal change by the firmware.

⑥ Not permitted! If function activation is reset prior to receipt of acknowledgement, the output signals are not updated – without the operational sequence of the activated function being affected.

⑦ *ASUP cannot be executed*: Negative acknowledgement, error occurred.

Figure 15-2 Example: Signal flow

15.6.6 Channel selection on the HMI

Function

The channel displayed on the HMI, e.g. in the machine start screen, can be selected from the PLC user program via the HMI/PLC interface.

Requirement

More than one channel is parameterized in the NC.

Job and acknowledgment interface

DB1900	Channel selection [r/w] (PLC ↔ HMI)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB5011	Error identification (see subsection below) (HMI → PLC)							
...	...							
DBB5021	Status (HMI → PLC)	Requirement (PLC ↔ HMI)	Function number (PLC → HMI)					
DBB5022	Channel number (see subsection below) (PLC → HMI)							

Channel number

- Channel number: 1, 2, ... max. number of channels
- Next channel: FF_H

Error identification

- 0: No error
- 1: Invalid function number (DBX32.0 - .5)
- 2: Invalid parameter (DBB33 - DBB35)
- 3: Error when writing to HMI-internal variable
- 10: Channel not present (DBB33)

Functional sequence

PLC → HMI

The PLC user program must maintain the following execution sequence:

1. Check whether the interface is free for a new job:
   - `DB1900.DBX`**5021**`.6 == 0` (function request)
   - `DB1900.DBX`**5021**`.7 == 0` (status)
2. If the interface is free, the job data must be entered and the function request set:
   - `DB1900.DBB`**5022** `= <channel number>`
   - `DB1900.DBX`**5021**`.0 - .5 = 1` (function number: channel selection)
   - `DB1900.DBX`**5021**`.6 = 1` (function request)

HMI → PLC

The HMI makes the following responses for **error-free** parameterization:

1. Once the HMI has recognized the function request for channel selection, the status is set to "function being performed" and the function request reset:
   - `DB1900.DBX`**5021**`.7 = 1` (status)
   - `DB1900.DBX`**5021**`.6 = 0` (function request)
2. Once the channel selection has been performed, the status is reset again and value 0 is set as error identification:
   - `DB1900.DBX`**5021**`.7 = 0` (status)
   - `DB1900.DBX`**5011** `= 0` (error identification)

The HMI makes the following responses for **faulty** parameterization:

- The function request is reset and the appropriate error identification is set:
  - `DB1900.DBX`**5021**`.6 = 0` (function request)
  - `DB1900.DBX`**5021**`.7 = 0` (status)
  - `DB1900.DBX`**5011** `= <error identification>`

15.7 CNC lock function (option)

15.7.1 Function

Note

The "CNC lock function" is a licensed option (article number: 6FC5800-0AP76-0YB0).

The use of the CNC lock function requires purchase of the appropriate license from SIEMENS. The use of the lock function with a trial license is **not** possible.

NOTICE

License certificate

The company that created the CNC lock function (machine manufacturer or dealer) must retain the license certificate for this option (CoL)!

This certificate can be used as legitimation for SIEMENS should the PIN be forgotten. The owner of the certificate (CoL) can have the machine unlocked.

The machine manufacturer can use the "CNC lock function" and the encrypted file that created with SINUMERIK Access MyMachine (AMM) Integrate application to activate a lock date in the control system. This allows the use of the machine to be limited to the time until the lock date is reached. The NC Start function of the control system is locked when the lock date is exceeded.

The CNC lock function can be lengthened or deactivated with an additional encrypted file. The machine manufacturer sends this file to the end user if the user has fulfilled the agreed obligations.

15.7.2 Requirements

The following preconditions must be met for using the "CNC lock function":

- The "CNC lock function" option must be set.
- A PLC project of CPU type "828D Step 2 x.yy" must be used.
  Compatibility mode must be deactivated.
- The SINUMERIK Integrate Access MyMachine/ P2P (PC) application must be installed.

15.7.3 Restrictions

The CNC lock function supports the business model with time-limited use. This protects against unauthorized use beyond the set interval. The direct access to the CNC, however, makes it possible to circumvent the function. The CNC lock function does not offer an absolute protection against manipulation. Unauthorized use of the machine is precluded by locking the automatic mode of the CNC. Because a running automatic program cannot be interrupted, this can extend the runtime beyond the lock date. All other functions of the SINUMERIK control system remain available.

To permit the CNC lock function to act, support of the machine manufacturer is required. Consequently, the following note and supplementary conditions must be observed when the CNC lock function is used!

Note

The CNC lock function is based on a connection of the PLC project to the associated SINUMERIK control system. The SINUMERIK control system consists of a combination of Panel Processing Unit (PPU) and CF card with system software.

Supplementary conditions

- Manipulation attempts and/or inconsistencies can lead to the CNC lock function causing a machine standstill.
- The use of the CNC lock function may require additional service calls of the machine manufacturer or dealer at the customer site.
- The PLC project should never be given to the customer without saved OEM PIN. The use of a "free" PLC project allows circumvention of the CNC lock function!
- A reimplementation of the PLC project of the associated machine tool allows circumvention of the CNC lock function!
- The protection of the program organization units in the PLC of the SINUMERIK must be used. Activation in the PLC Programming Tool is possible. The coding of the PLC project must be kept secret.

- To provide better protection, each machine must be given its own OEM PIN.
- Before setting up the lock function for the first time (see Chapter "Initial creation of the CNC lock function (Page 1207)"), the setting up engineer must correctly set the date and the time in the SINUMERIK control. If the date lies in the past, then the operating time of the machine extends corresponding to the difference to the real date.
- The CNC lock function is built on the real-time clock of the SINUMERIK 828D. The maintenance-free design of the SINUMERIK 828D can cause the time of day to be lost. The CNC lock function performs a computerized numerical plausibility check of the time of day. This check can be impaired by power loss to the real-time clock. In this case, the time without power supply is ignored.
- The company that created the CNC lock function retains the associated license certificate (CoL). This certificate serves as proof as authorized function creator to SIEMENS and so implicitly as authorized user of the machine.
- A software malfunction can cause unintentional locking of the control system.

15.7.4 Protection from manipulation

The CNC lock function as part of the currently expected use and misuse of the CNC serves to permit use only within the set time period. Despite the available protective mechanisms against an impermissible manipulation of the CNC lock function, a residual risk that the protective mechanisms can be circumvented remains. The CNC lock function cyclically checks the installed combination of PPU, CF card and PLC project. The lock function is not applicable when replacing all three components. To protect against manipulation, it is absolutely essential that the secondary conditions listed in Chapter "Restrictions (Page 1206)" are carefully complied with.

Note

The CNC lock function uses a cryptographic technique. When first marketed, the technique used corresponds to state-of-the-art technology. As time goes by, the probability that this technique will be able to be bypassed (manipulated) increases.

15.7.5 Initial creation of the CNC lock function

The initial creation of the CNC lock function couples the control system hardware, the Panel Processing Unit (PPU), together with the system software on the CF card and the PLC project that belongs to the machine. An initial lock date is also set at the same time.

To create the lock function, an encrypted lockset file (lockset.clc) appropriate for the hardware must be generated. The file is created with the SINUMERIK Integrate Access MyMachine (AMM) application.

Generating the lockset file

The following data is required to generate the lockset file:

- Serial number of the CF card
- Serial number of the control system (Panel Processing Unit, PPU)
- OEM PIN
- Lock date

Note

The **serial number** of the CF card and of the control system (PPU) can be found on the SINUMERIK Operate user interface in the "Diagnostics" > "Version" > "Hardware" > "NCU/PLC" > "ncu1" operating area:

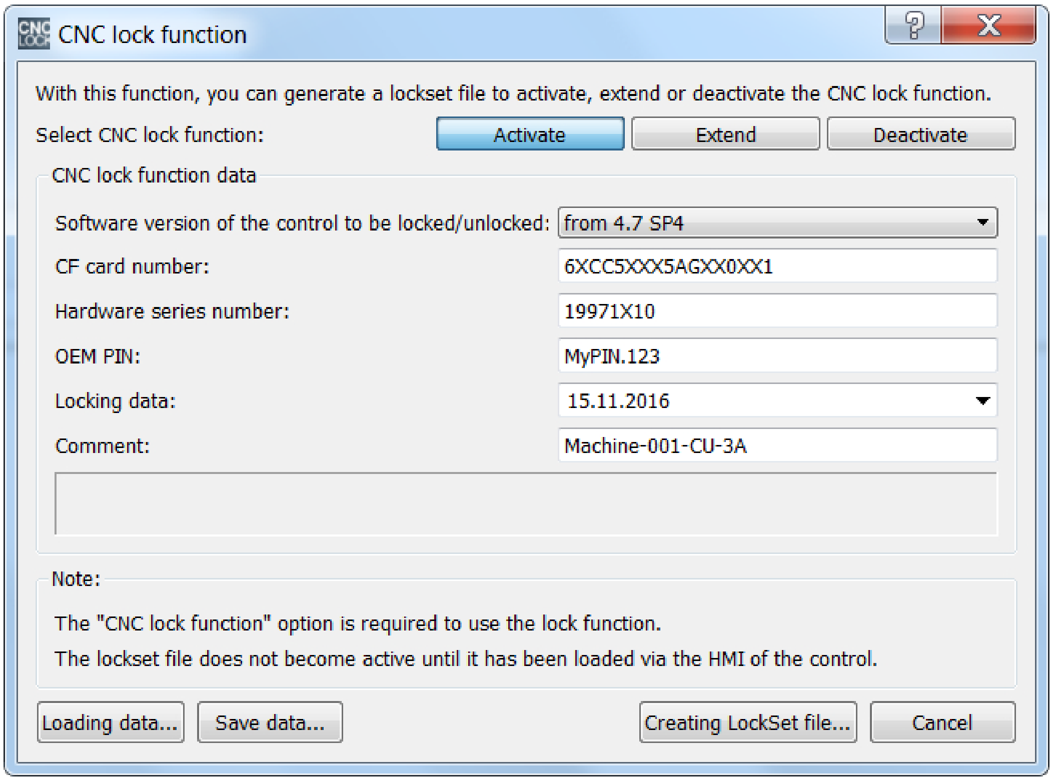
- **CF card:** "CF card" > "SerialNo." area
- **PPU:** "SINUMERIK 828D PPU" > "SerialNo." area

Note

The **OEM PIN** increases the protection against manipulation of the CNC lock function.

The OEM PIN is stored by the system when activating the CNC lock function in the PLC user program. The OEM PIN cannot be viewed, changed or deleted by the user in the PLC user program.

The data is entered via Access MyMachine in the "CNC lock function" dialog window (called from the main menu: "Tools" > "CNC lock function"). The "**Activate**" button must be selected:



With this function, you can generate a lockset file to activate, extend or deactivate the CNC lock function.

Select CNC lock function: Activate

CNC lock function data

Software version of the control to be locked/unlocked:	from 4.7 SP4
CF card number:	6XCC5XXX5AGXX0XX1
Hardware series number:	19971X10
OEM PIN:	MyPIN.123
Locking data:	15.11.2016
Comment:	Machine-001-CU-3A

Note:

The "CNC lock function" option is required to use the lock function.

The lockset file does not become active until it has been loaded via the HMI of the control.

After entering all of the data required, the lockset file to activate is generated using the "Creating lockset file...".

Transferring the lockset file

The lockset file must then be transferred to the control system either directly via an Ethernet connection or alternatively via a storage medium, e.g. USB flash drive.

The file is located at: /System-CF-Card/User/sinumerik/data/license.

Reading in the lockset file

Before the lockset file is imported into the control system, the machine manufacturer must set the time of day of the control system correctly, because the time of day at the point in time of activating the CNC lock function is saved as the start value for monitoring. The import can then be started from the user interface: "Commissioning" > "Licenses" > "Import license key" operating area.

No access level is required for the import.

If no error occurs when importing the lockset file, the CNC lock function is active in the control system.

Note

If an error occurs when importing the lockset file, an error-specific alarm will be issued. The state of the CNC lock function remains unchanged.

Note

We recommend that the machine manufacturer after commissioning the machine and activating the CNC lock function creates a complete **commissioning archive** over all control system components. This ensures data consistency for the CNC lock function. If necessary, this commissioning archive can be used to recommission the control system without requiring a service call to reactivate the CNC lock function.

Checking the lock date

The activation of the CNC lock function is apparent with the lock date entered in the lockset file being displayed on the user interface:

SINUMERIK Operate user interface: "Commissioning" > "Licenses" operating area

Setting the prewarning time

The prewarning time is the time range before reaching the lock date above which alarm 8063 is displayed once daily. The alarm indicates that the lock date has initially been reached and NC Start is locked for the control system. The prewarning time is set via the machine data:

MD17300 $MN_CNC_LOCK_WARNING_TIME

Important notes

The following information must be heeded for the correct and reliable functioning of the CNC lock function!

Note

Before activating the CNC lock function, the **time of day** must be set correctly on the control system.

Note

The PLC user program should always be given the **POU password protection**. This prevents users from copying the machine-specific know-how and using it in their own PLC user program, and then replacing the PLC user program with the PLC user program of the machine manufacturer that contains the PLC key of the CNC lock function.

Note

The **commissioning archive** for the end user may be exported only **after activation** of the CNC lock function.

Note

After deactivation of the CNC lock function (see "Deactivating the CNC lock function (Page 1213)"), a **new commissioning archive** must be exported and replaced with the original archive. Only this guarantees that no inadvertent reactivation of the CNC lock function occurs. This would cause an alarm and disable of the NC Start. To revoke this, the lock set file for deactivating the CNC lock function must be reimported.

Note

The machine manufacturer has sole responsibility for the correct and reliable functioning of the CNC lock function.

15.7.6 Extending the CNC lock function

To extend the CNC lock function, the machine manufacturer must use Access MyMachine (AMM) to create a new lockset file (lockset.clc) **with new lock date** for the CNC lock function.

Generating the lockset file

The following data is required to generate the lockset file:

- Serial number of the CF card
- Serial number of the control system (Panel Processing Unit, PPU)
- OEM PIN
- **New lock date**

Note

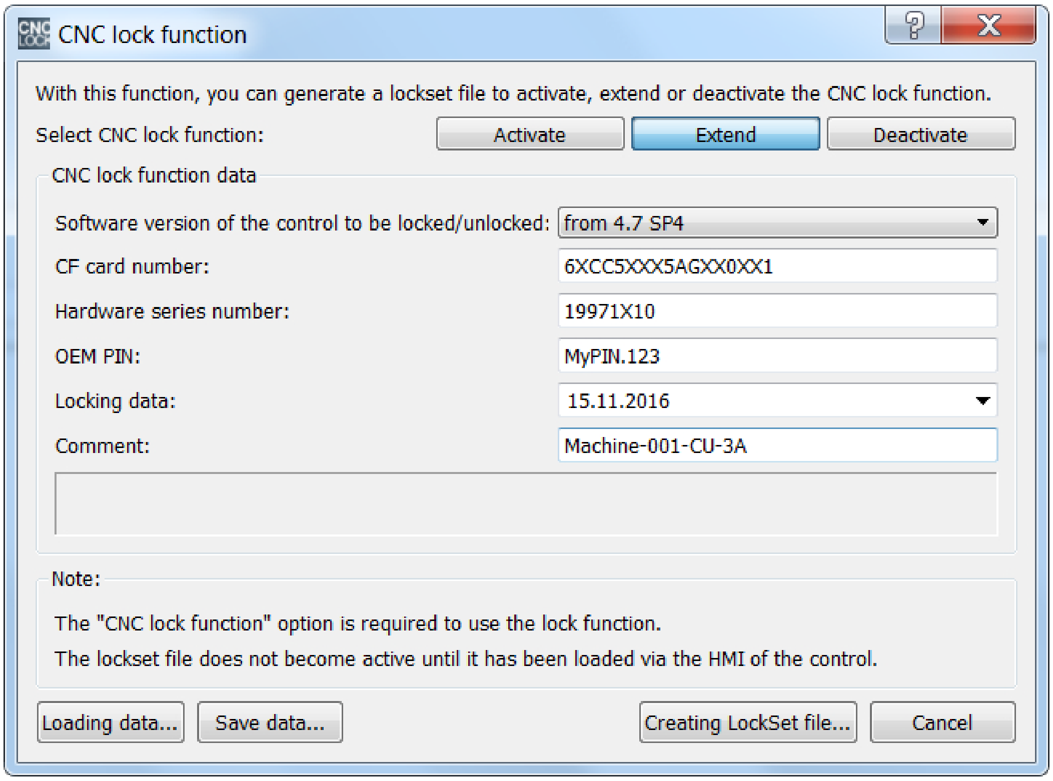
The serial number of the CF card and the control system (Panel Processing unit, PPU) as well as the OEM PIN must match the values used when the CNC lock functions was activated initially.

Note

The **serial number** of the CF card and of the control system (PPU) can be found on the SINUMERIK Operate user interface in the "Diagnostics" > "Version" > "Hardware" > "NCU/PLC" > "ncu1" operating area:

- **CF card:** "CF card" > "SerialNo." area
- **PPU:** "SINUMERIK 828D PPU" > "SerialNo." area

The data is entered via Access MyMachine in the "CNC lock function" dialog window (called from the main menu: "Tools" > "CNC lock function"). The "**Extend**" button must be selected:


CNC lock function

With this function, you can generate a lockset file to activate, extend or deactivate the CNC lock function.

Select CNC lock function: Activate | Extend | Deactivate

CNC lock function data

Software version of the control to be locked/unlocked:	from 4.7 SP4
CF card number:	6XCC5XXX5AGXX0XX1
Hardware series number:	19971X10
OEM PIN:	MyPIN.123
Locking data:	15.11.2016
Comment:	Machine-001-CU-3A

Note:

The "CNC lock function" option is required to use the lock function.

The lockset file does not become active until it has been loaded via the HMI of the control.

After entering all of the data required, the lockset file to extend is generated using the "Creating lockset file...".

Transferring the lockset file

The new lockset file must be transferred to the control system. The transfer can be made by the machine manufacturer directly via an Ethernet connection to the control system.

The file is located at: /System-CF-Card/User/sinumerik/data/license.

Or the machine manufacturer sends the new lockset file to the end user who transfers the file to the control system.

Reading in the lockset file

The lockset file import is started from the user interface: "Commissioning" > "Licenses" > "Import license key" operating area.

No access level is required for the import.

If no error occurs when importing the lockset file, the CNC lock function with the new lock date is active in the control system.

Note

If an error occurs when importing the lockset file, an error-specific alarm will be issued. The state of the CNC lock function remains unchanged.

Checking the changed lock date

The successful change of the lock date in the control system can be checked on the user interface:

SINUMERIK Operate user interface: "Commissioning" > "Licenses" operating area

15.7.7 Deactivating the CNC lock function

To deactivate the CNC lock function, the machine manufacturer must use Access MyMachine (AMM) to create a lockset file (lockset.clc) **without lock date**.

Generating the lockset file

The following data is required to generate the lockset file:

- Serial number of the CF card
- Serial number of the control system (Panel Processing Unit, PPU)
- OEM PIN

Note

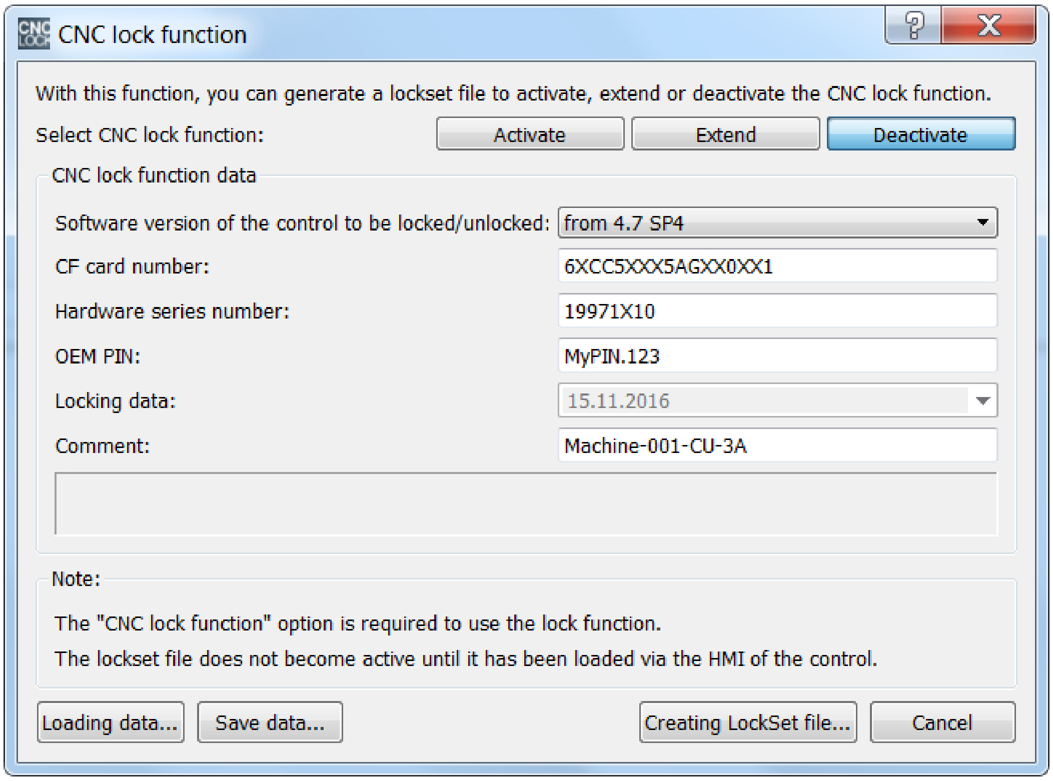
The serial number of the CF card and of the control system (Panel Processing unit, PPU) as well as the OEM PIN must match the values used when the CNC lock function was activated initially.

Note

The **serial number** of the CF card and of the control system (PPU) can be found on the SINUMERIK Operate user interface in the "Diagnostics" > "Version" > "Hardware" > "NCU/ PLC" > "ncu1" operating area:

- **CF card:** "CF card" > "SerialNo." area
- **PPU:** "SINUMERIK 828D PPU" > "SerialNo." area

The data is entered via Access MyMachine in the "CNC lock function" dialog window (called from the main menu: "Tools" > "CNC lock function"). The "**Deactivate**" button must be selected:



With this function, you can generate a lockset file to activate, extend or deactivate the CNC lock function.

Select CNC lock function: Deactivate

CNC lock function data

Software version of the control to be locked/unlocked:	from 4.7 SP4
CF card number:	6XCC5XXX5AGXX0XX1
Hardware series number:	19971X10
OEM PIN:	MyPIN.123
Locking data:	15.11.2016
Comment:	Machine-001-CU-3A

Note:

The "CNC lock function" option is required to use the lock function.

The lockset file does not become active until it has been loaded via the HMI of the control.

After entering all of the data required, the lockset file to deactivate is generated using the "Creating lockset file...".

Transferring the lockset file

The new lockset file must be transferred to the control system. The transfer can be made by the machine manufacturer directly via an Ethernet connection to the control system.

The file is located at: /System-CF-Card/User/sinumerik/data/license.

Or the machine manufacturer sends the new lockset file to the end user who transfers the file to the control system.

Reading in the lockset file

The lockset file import is started from the user interface: "Commissioning" > "Licenses" > "Import license key" operating area.

No access level is required for the import.

If no error occurs when importing the lockset file, the CNC lock function is deactivated.

Note

If an error occurs when importing the lockset file, an error-specific alarm will be issued. The state of the CNC lock function remains unchanged.

Note

We recommend that the end user **after deactivating** the lock date creates a complete **commissioning archive** over all control system components. If necessary, this commissioning archive can be used to recommission the control system without reactivating the CNC lock function.

Check

The successful deactivation of the lock date in the control system can be checked by the lock date no longer being displayed on the user interface:

SINUMERIK Operate user interface: "Commissioning" > "Licenses" operating area

15.7.8 Replacing a defective control system hardware (PPU)

During the replacement of a defective control system hardware (PPU), the system CF card must remain in the machine so it can be deployed in the new control system hardware.

After the initial power up of the control system with the new hardware, an existing commissioning archive created with the previous control system hardware must be imported. After the next power up of the control system, alarm 8062: "CNC lock function: The execution of the function failed: Cause 2" (the hardware has been replaced) is displayed and NC Start disabled. The cause of the alarm is the new serial number of the new control system hardware.

Requesting a new lockset file

To unlock the control system again, the end user must request a new lockset file (lockset.clc) appropriate for the control system from the company that created the CNC lock function (machine manufacturer or dealer).

Generating the lockset file

The following data is required to generate the lockset file:

- Serial number of the CF card remaining with the machine
- Serial number of the **new** control system (PPU)
- The original assigned OEM PIN
- Last lock date or new lock date

Note

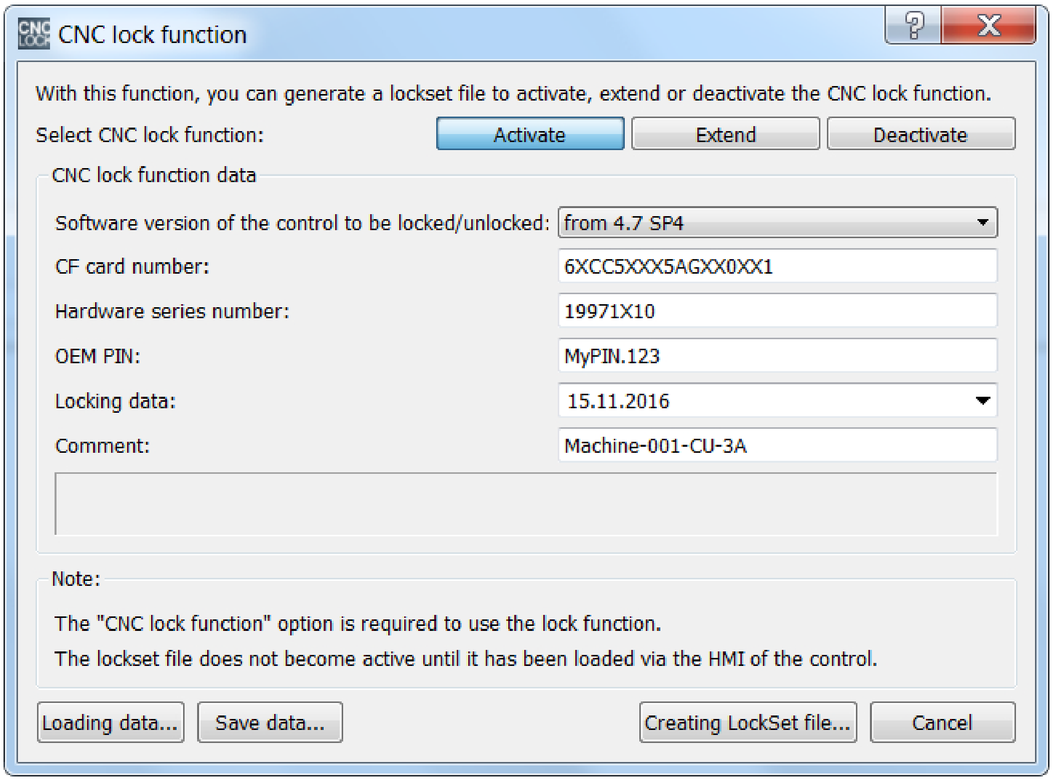
The original assigned **OEM PIN** must be known to the company that created the CNC lock function for the replacement of defective control system hardware (PPU)!

Note

The **serial number** of the CF card and of the control system (PPU) can be found on the SINUMERIK Operate user interface in the "Diagnostics" > "Version" > "Hardware" > "NCU/PLC" > "ncu1" operating area:

- **CF card:** "CF card" > "SerialNo." area
- **PPU:** "SINUMERIK 828D PPU" > "SerialNo." area

The data is entered via Access MyMachine in the "CNC lock function" dialog window (called from the main menu: "Tools" > "CNC lock function"). The "**Activate**" button must be selected:



With this function, you can generate a lockset file to activate, extend or deactivate the CNC lock function.

Select CNC lock function: Activate

CNC lock function data

Software version of the control to be locked/unlocked:	from 4.7 SP4
CF card number:	6XCC5XXX5AGXX0XX1
Hardware series number:	19971X10
OEM PIN:	MyPIN.123
Locking data:	15.11.2016
Comment:	Machine-001-CU-3A

Note:

The "CNC lock function" option is required to use the lock function.

The lockset file does not become active until it has been loaded via the HMI of the control.

After entering all of the data required, the lockset file to unlock is generated using the "Create lockset file... button".

Transferring the lockset file

The new lockset file must be transferred to the control system. The transfer can be made by the machine manufacturer directly via an Ethernet connection to the control system.

The file is located at: /System-CF-Card/User/sinumerik/data/license.

Or the machine manufacturer sends the new lockset file to the end user who transfers the file to the control system.

Reading in the lockset file

Note

Before the lockset file **is imported**, the **time of day** of the new control system hardware must be set correctly, because the time of day at the time of the activation of the CNC lock function is saved as the start value for monitoring. The company that created the CNC lock function can protect the action by the replacement and the setting of the time of day being made by the service personnel.

The lockset file import is started from the user interface: "Commissioning" > "Licenses" > "Import license key" operating area.

No access level is required for the import.

If no error occurs when importing the lockset file, the CNC lock function is active in the new control system.

15.7.9 Replacing a defective CF card

After replacing a defective system CF card, it must be sent to SIEMENS. In return, SIEMENS supplies a new CF card.

After the initial power up of the control system with the new CF CF card, an existing commissioning archive created with the previous control system hardware must be imported. After the next power up of the control system, alarm 8062: "CNC lock function: The execution of the function failed: Cause 1" (the CF card has been replaced) is displayed and the NC Start locked. The cause of the alarm is the new serial number of the new CF card.

Requesting a new lockset file

To unlock the control system again, the end user must request a new lockset file (lockset.clc) appropriate for the control system from the company that created the CNC lock function (machine manufacturer or dealer).

Generating the lockset file

The following data is required to generate the lockset file:

- Serial number of the new **CF card**
- Serial number of the control system (PPU)
- The original assigned OEM PIN
- The lock date in accordance with the status of the CNC lock function (last lock date, new lock date, no lock date)

Note

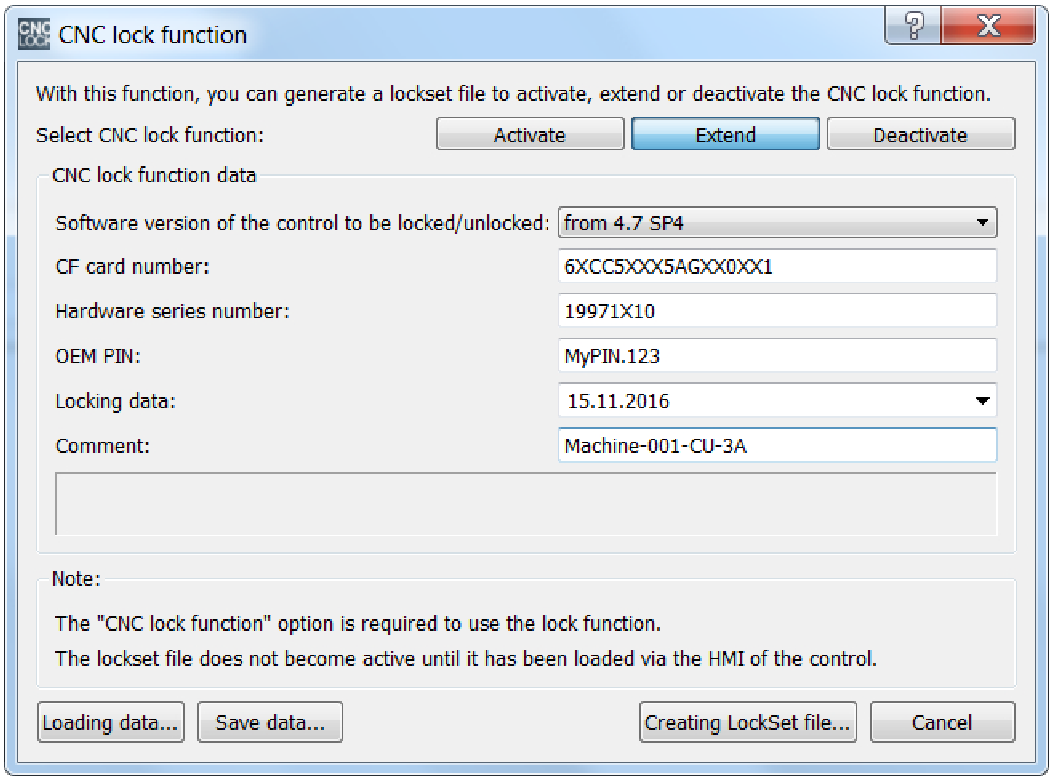
The original assigned **OEM PIN** must be known to the company that created the CNC lock function for the replacement of defective CF card!

Note

The **serial number** of the CF card and of the control system (PPU) can be found on the SINUMERIK Operate user interface in the "Diagnostics" > "Version" > "Hardware" > "NCU/PLC" > "ncu1" operating area:

- **CF card:** "CF card" > "SerialNo." area
- **PPU:** "SINUMERIK 828D PPU" > "SerialNo." area

The data is entered via Access MyMachine in the "CNC lock function" dialog window (called from the main menu: "Tools" > "CNC lock function"). The "**Extend**" or "**Deactivate**" button must be selected:



With this function, you can generate a lockset file to activate, extend or deactivate the CNC lock function.

Select CNC lock function: Extend

CNC lock function data

Software version of the control to be locked/unlocked:	from 4.7 SP4
CF card number:	6XCC5XXX5AGXX0XX1
Hardware series number:	19971X10
OEM PIN:	MyPIN.123
Locking data:	15.11.2016
Comment:	Machine-001-CU-3A

Note:

The "CNC lock function" option is required to use the lock function.
The lockset file does not become active until it has been loaded via the HMI of the control.

Figure 15-3 Update CNC lock function

After entering all of the data required, the lockset file to unlock is generated using the "Create lockset file... button".

Transferring the lockset file

The new lockset file must be transferred to the control system. The transfer can be made by the machine manufacturer directly via an Ethernet connection to the control system.

The file is located at: /System-CF-Card/User/sinumerik/data/license.

Or the machine manufacturer sends the new lockset file to the end user who transfers the file to the control system.

Reading in the lockset file

Note

Before the lockset file **is imported**, the **time of day** of the new control system hardware must be set correctly, because the time of day at the time of the activation of the CNC lock function is saved as the start value for monitoring. The company that created the CNC lock function can protect the action by the replacement and the setting of the time of day being made by the service personnel.

The lockset file import is started from the user interface: "Commissioning" > "Licenses" > "Import license key" operating area.

No access level is required for the import.

If no error occurs when importing the lockset file, the CNC lock function is active in the new control system.

15.7.10 OEM PIN forgotten

The company that created the CNC lock function (machine manufacturer or dealer) has forgotten the OEM PIN that was assigned during the initial creation and so can no longer create a valid lockset file for the associated control system.

Unlocking the machine

To allow the company that created the CNC lock function to operate the machine, the technician must contact the SIEMENS Hotline and provide the following information:

- License certificate (CoL) for the "CNC lock function" option
- Serial number of the CF card
- Serial number of the control system (PPU)
- Software version of the CNC software

Note

The **serial number** of the CF card and of the control system (PPU) can be found on the SINUMERIK Operate user interface in the "Diagnostics" > "Version" > "Hardware" > "NCU/PLC" > "ncu1" operating area:

- CF card: "CF card" > "SerialNo." area
- PPU: "SINUMERIK 828D PPU" > "SerialNo." area

The **software version of the CNC software** can be found on the SINUMERIK Operate user interface in the "Diagnostics" > "Version" > "Actual version" operating area.

The original license certificate (CoL) must be sent to SIEMENS. Unlocking by SIEMENS involves a fee.

The company that created the CNC lock function receives from the Hotline the lockset file to unlock the machine. The unlocking acts, however, only on the hardware of the control system

(PPU). The PLC project is not unlocked. For this reason, the original PLC project appropriate for the machine must be available.

The unlocking of the machine requires the on-site presence of a service technician of the creator of the CNC lock function and the end user (operating company) of the machine.

Further information concerning the procedure can be obtained from the SIEMENS Hotline.

15.7.11 Other information

Project file

Note

The machine manufacturer is responsible for documenting the assignments of the serial numbers and OEM PINs.

Access MyMachine can be used to create an unencrypted project file ("ucls" for "**U**ser-**C**NC-**L**ock-**S**et file") that contains the following data:

- Serial number of the CF card
- Serial number of the control system (PPU)
- OEM PIN
- Creation date
- Lock date

This function is called using the "Save data..." button in the "CNC lock function" dialog window.

Clicking the "Load data..." button reimports the unencrypted data stored in the project file.

Faulty settings for date or time of day

If incorrect settings for date or time of day are determined for activated CNC lock function, then the following actions are initiated:

- Alarm 8065 is output: "CNC lock function: Please correctly set the date/time of day!"
- Blocking an NC start

The incorrect settings must be corrected before switching off the control in order to withdraw the lock.

NOTICE
Permanent lock
The incorrect date or time of day setting must be corrected before switching off the control. Otherwise there is a risk of a permanent lock through alarm 8064: "CNC lock function: The lock date has been reached, the NC cannot be started!"
Remedy: Correctly set the date/time of day **before** switching off the control.

If, with the CNC lock function activated, a future date is set, then the following alarm is output:

Alarm 8066 "CNC lock function: Changing the date would reduce the remaining runtime!"

The date can still be corrected as long as the control system was not switched off.

NOTICE
Shorter service life
After switching off the control system, a date set in the future is considered as being an actual date and can no longer be reset. This reduces the service life until the lock date.
Remedy: Correctly set the date **before** switching off the control.

Further information

See:

- Online help for SINUMERIK Integrate Access MyMachine / P2P (PC)
- Online help for the PLC Programming Tool

R1: Referencing 16

16.1 Brief Description

Function

When referencing a machine axis, the coordinate system of the machine axis is synchronized with the coordinate system of the machine. The machine axis is traversed to the machine zero and then the actual position of the machine axis is set to zero.

If the machine zero cannot be directly approached as a result of the machine design, then a reference point is defined in the traversing range of the machine axis, which then used to synchronize the machine axis. Its position with reference to the machine zero must be known. When referencing, the actual machine axis position is set to this value

Measuring systems and referencing methods

When referencing, machine axes can be synchronized with the following measuring systems and referencing types:

- 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 proximity switch and one-edge and two-edge detection
  - Referencing with incremental measuring systems with replacement of homing cam with proximity switch
  - Referencing with incremental measuring systems with proximity switch 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

Referencing the machine axis can be manually started, or from the part program:

- Manual: Operation mode JOG and MDI, machine function REF
- Part program: Command `G74`

16.2 Axisspecific referencing

For axis-specific referencing using the reference point approach, the operation must be individually started for each machine axis to be referenced.

Selecting mode and machine function

Before starting reference point approach of the machine axes, you must first place the relevant mode group in JOG or MDI mode:

DB11, ... DBX0.2 (active JOG mode)

DB11, ... DBX0.1 (active MDI mode)

Then machine function REF (reference point approach) must be selected:

DB11, ... DBX1.2 (REF machine function)

Start of reference point approach

In axis-specific reference point approach, each machine axis must be started individually.

Reference point approach is started with the axis-specific traversing keys:

DB31, ... DBX4.6 (Traversing key minus)

DB31, ... DBX4.7 (Traversing key minus)

Direction enable

To avoid operator errors, the direction release must be parameterized. The direction enable specifies which traversing key starts the reference point approach:

MD34010 $MA_REFP_CAM_DIR_IS_MINUS = <value>

<Value>	Meaning
0	Reference point approach in plus direction
1	Reference point approach in minus direction

Jog mode

The following machine data element can be used to specify whether reference point approach is completed when the direction key is pressed once or whether the operator is required to keep the direction key pressed (jogging) for safety reasons:

MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD (INC and REF in jog mode)

If the machine operator releases the direction key, the machine axis is decelerated to zero speed. Reference point approach is not aborted. Reference point approach is continued the next time the direction key is pressed.

Referencing status

The referencing status of the machine axis is reset with the start of the reference point approach:

DB31, ... DBX60.4 (referenced/synchronized 1)

DB31, ... DBX60.5 (referenced/synchronized 2)

DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)

Distance-coded measuring systems

In distance-coded measuring systems, reference point approach can be started with any traversing key.

Sequence

The machine operator or machine manufacturer (via the PLC user program) is responsible for ensuring that the machine axes are referenced in the proper order.

- Machine operator
  The machine axes must be started by the machine operator in the specified order.
- Machine manufacturer
  The PLC user program of the machine manufacturer allows machine axes to be started only in the proper order.

Simultaneous reference point approach of several machine axes

Several machine axes can be referenced simultaneously, depending on the control:

Completion of reference point approach

Acknowledgment that reference point approach of a machine axis has been successfully completed is given by setting the referencing status:

DB31, ... DBX60.4 (referenced/synchronized 1)

DB31, ... DBX60.5 (referenced/synchronized 2)

Cancellation of reference point approach

In axis-specific reference point approach, the machine axis is traversed in the channel that was assigned as the master channel of the machine axis.

MD30550 $MA_ AXCONF_ASSIGN_MASTER_CHAN

For aborting the reference point approach, either mode group reset or channel reset for the master channel of the machine axis must be activated:

DB11, ... DBX0.7 (mode group reset)

DB21, ... DBX7.7 (channel reset)

All machine axes that have not yet successfully completed reference point approach when the action is cancelled remain in status "Not referenced":

DB31, ... DBX60.4 (referenced/synchronized 1)

DB31, ... DBX60.5 (referenced/synchronized 2)

16.3 Channelspecific referencing

For channel-specific referencing, all machine axes of the channel are referenced in the parameterized sequence when reference point approach is initiated.

Selecting mode and machine function

Before starting the reference point approach of the machine axes, the associated mode group must be switched into the JOG or MDI operating mode:

DB11, ... DBX0.2 (active JOG mode)

DB11, ... DBX0.1 (active MDI mode)

Then machine function REF (reference point approach) must be selected:

DB11, ... DBX1.2 (REF machine function)

Parameterizing the axis sequence

The following machine data element is used to specify the sequence in which the machine axes of the channel are referenced:

MD34110 $MA_REFP_CYCLE_NR = <number>

<number>	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 will be referenced simultaneously.

Simultaneous reference point approach of several machine axes

Several machine axes can be referenced simultaneously, depending on the control:

Start of reference point approach

Channel-specific reference point approach is started with:

DB21, ... DBX1.0 (activate referencing)

The status of channel-specific reference point approach is indicated by the channel with:

DB21, ... DBX33.0 (referencing active)

Referencing status

The referencing status of the machine axis is reset with the start of the reference point approach:

DB31, ... DBX60.4 (referenced/synchronized 1)

DB31, ... DBX60.5 (referenced/synchronized 2)

Completion of reference point approach

As soon as channel-specific reference approach has been successfully completed for all machine axes involved, this is acknowleged with:

DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)

Cancellation of reference point approach

In channel-specific reference point approach the machine axis is traversed in the channel to which that axis is currently assigned as channel axis.

For aborting the reference point approach either mode group reset or channel reset for the corresponding channel must be activated:

DB11, ... DBX0.7 (mode group reset)

DB21, ... DBX7.7 (channel reset)

All machine axes for which the reference point approach is not yet successfully completed when the action is cancelled remain in status "Not referenced":

DB31, ... DBX60.4 (referenced/synchronized 1)

DB31, ... DBX60.5 (referenced/synchronized 2)

16.4 Reference point appraoch from part program (G74)

Function

With command `G74`, machine axes can be referenced from a part program or synchronized action either for the first time or again.

Referencing must be repeated, for example, after:

- Actual value offset `PRESETON`
  References: Programming Guide Advanced, section "Coordinate transformations (Frames)" > "Preset offset with PRESETON"
- Parking a machine axis (Page 125)
- Exceeding the encoder limit frequency of the position measuring system

Programming

Syntax

```
G74 <machine axis> {<machine axis>}
```

Meaning

`G74`:	Command for referencing machine axes	
	Alone in the block:	Yes
	Effectiveness:	Non-modal
`<machine axis>`:	The name of a machine axis must be specified (MD10000 $MN_AXCONF_MACHAX_NAME_TAB). The machine axis must be a channel axis of the channel in which the part program or the synchronized action is executed.	

Reset response

Mode group reset or channel reset aborts the reference point approach for all programmed machine axes:

- DB11, ... DBX0.7 (mode group reset)
- DB21, ... DBX7.7 (channel reset)

All machine axes for which the reference point approach is not yet successfully completed when the action is canceled remain in status "Not referenced":

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

16.5 Referencing with incremental measurement systems

16.5.1 Hardware signals

Depending on the machine design and the properties of the incremental measuring system used, different hardware signals must be connected.

Reference cam

- Connection
  The reference cam signal can be connected to a digital input of an external PLC I/O module or to a fast input on the NCU X142 interface.
- NC/PLC interface signal
  The reference cam signal must be transferred from the PLC user program to the axial NC/PLC interface:
  DB31, ... DBX12.7 (deceleration of reference point approach)

Zero mark selection

If during the reference point approach of the axis or spindle several zero marks of the measuring system are detected (e.g. measuring gear between the motor and encoder), then the specific zero mark must be selected with an additional proximity switch signal.

- Connection
  The proximity switch must be connected to a fast digital input on the NCU X122 or X132 interface.
- Activation
  In order that the proximity switch signal is evaluated, the digital input to which the proximity switch is connected must be selected in drive parameter p0493 for the axis/spindle.

Equivalent zero mark

If the used measuring system does not provide a zero mark signal, an equivalent zero mark can be created via a proximity switch signal.

- Connection
  The proximity switch must be connected to a fast digital input on the NCU X122 or X132 interface.
- Activation
  In order that the proximity switch signal is evaluated, the digital input to which the proximity switch is connected must be selected in the p0494 or p0495 drive parameter for the axis/spindle.

Overview

Signal	Connection: Digital input via	Set
Reference cam	Ext. PLC I/O module or NCU: X142	PLC user program: DB31, ... DBX12.7
Zero mark selection	NCU: X122 or X132	Drive parameter: p0493
External zero mark or equivalent zero mark	NCU: X122 or X132	Drive parameter: p0494 or p0495

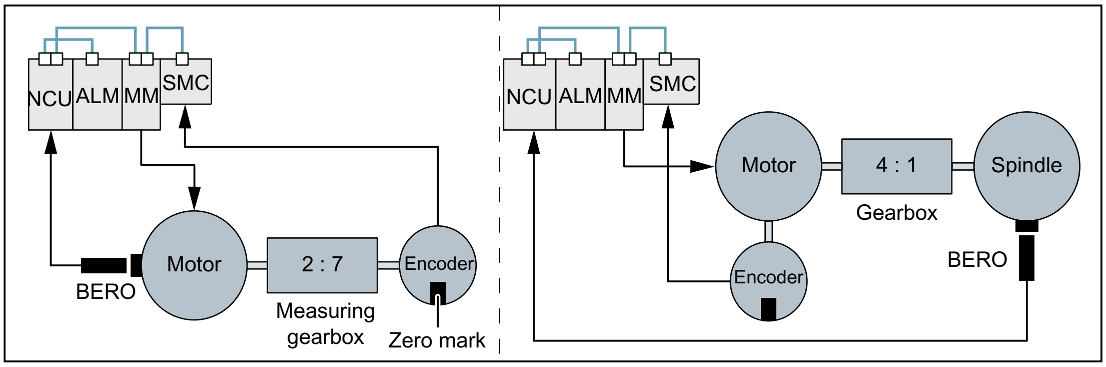
References

- NCU interfaces: SINUMERIK 840D sl Manual, NCU7x0.3 PN,
  Section "Connecting" > "Digital I/Os"
- Drive parameter: SINAMICS S120/S150 List Manual

16.5.2 Zero mark selection

Function

Referencing of incremental measuring systems is based on the unique position of the encoder zero mark relative to the overall traversing range of the machine axis. If because of machine-specific conditions, several encoder zero marks are detected in the traversing range of the machine axis (for examples, see figure below), a proximity switch must be mounted on the machine for clear determination of the reference point. The position of the reference point is then derived from the combination of contact-less proximity switch signal and encoder zero mark.



BER O Contact-less proximity switch

Figure 16-1 Measuring gear between the motor and encoder or reduction gear between the motor and spindle

Parameterization

NC: Referencing mode

"Referencing of incremental, rotary or linear measuring systems: Zero pulse on the encoder track" should be parameterized as referencing mode:

MD34200 $MA_ENC_REFP_MODE[<axis>] = 1

Drive: Zero mark selection

The digital input on the NCU interface to which the proximity switch is connected must be set in parameter p0493.

Note

Zero mark selection

The processing of the contact-less proximity switch signal is performed exclusively in the drive. Connection and parameterization, see Section "Hardware signals (Page 1228)".

16.5.3 Time sequence

Reference point approach with incremental measuring systems can be divided into three phases:

- Phase 1: "Phase 1: Traversing to the reference cam (Page 1232)"
- Phase 2: "Phase 2: Synchronization with the zero mark (Page 1234)"
- Phase 3: "Phase 3: Traversing to the reference point (Page 1239)"

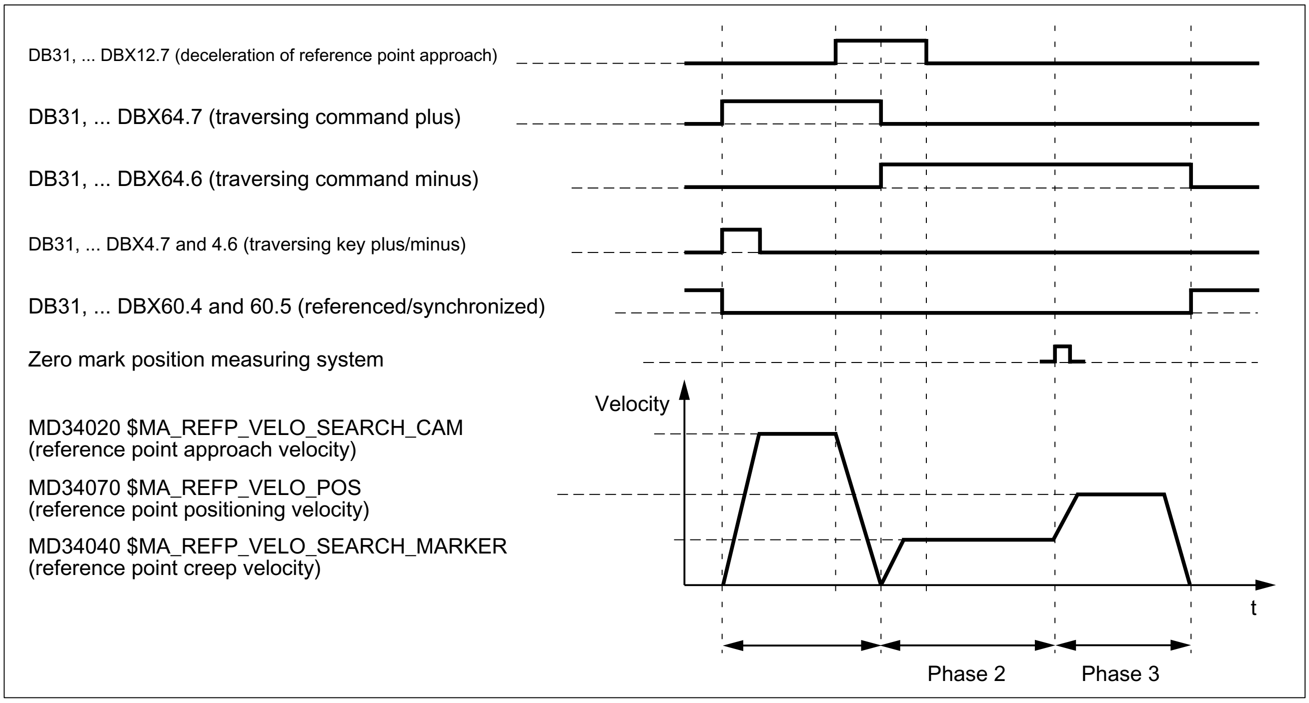

DB31, ... DBX12.7 (deceleration of reference point approach)
DB31, ... DBX64.7 (traversing command plus)
DB31, ... DBX64.6 (traversing command minus)
DB31, ... DBX4.7 and 4.6 (traversing key plus/minus)
DB31, ... DBX60.4 and 60.5 (referenced/synchronized)
Zero mark position measuring system
Velocity
MD34020 $MA_REFP_VELO_SEARCH_CAM
(reference point approach velocity)
MD34070 $MA_REFP_VELO_POS
(reference point positioning velocity)
MD34040 $MA_REFP_VELO_SEARCH_MARKER
(reference point creep velocity)
t
Phase 2
Phase 3

Figure 16-2 Time sequence when referencing with incremental measuring systems (example)

16.5.4 Phase 1: Traversing to the reference cam

Phase 1: Graphic representation

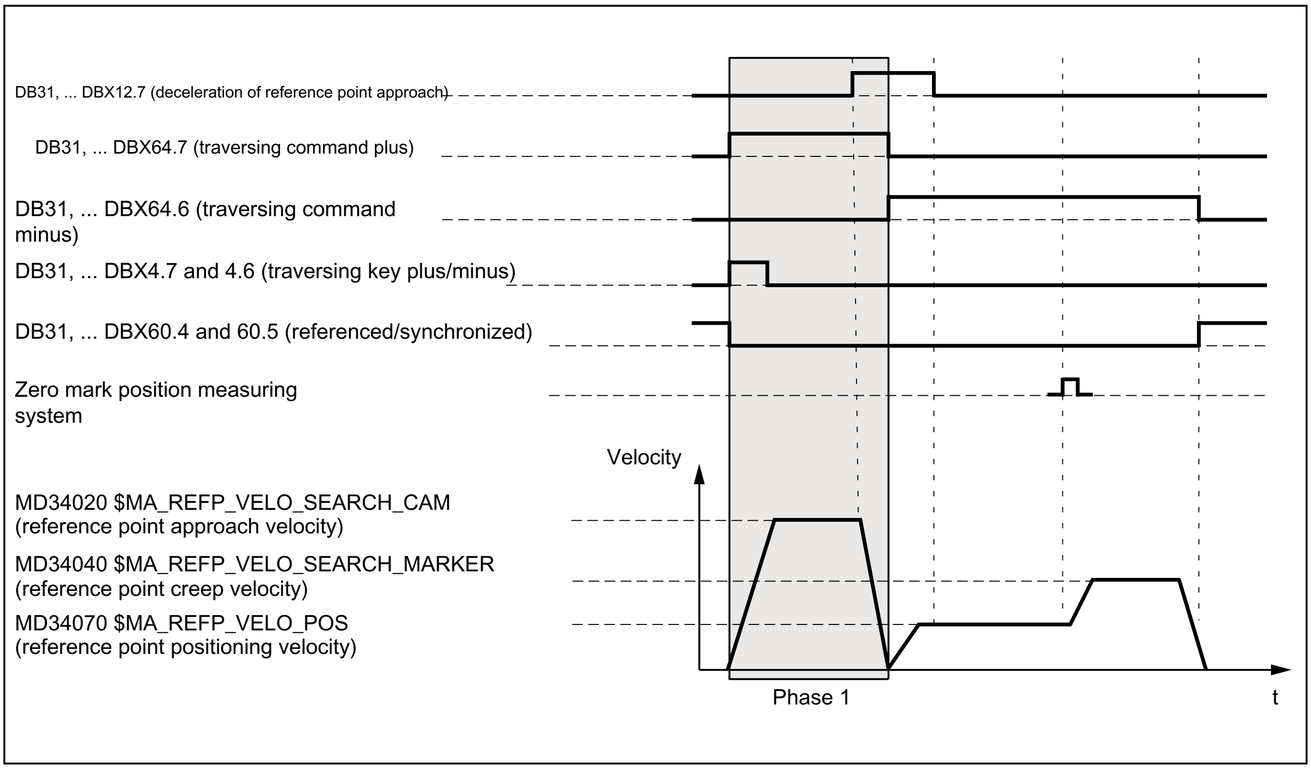

DB31, ... DBX12.7 (deceleration of reference point approach)
DB31, ... DBX64.7 (traversing command plus)
DB31, ... DBX64.6 (traversing command minus)
DB31, ... DBX4.7 and 4.6 (traversing key plus/minus)
DB31, ... DBX60.4 and 60.5 (referenced/synchronized)
Zero mark position measuring system
Velocity
MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)
MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)
MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)
Phase 1
t

Figure 16-3 Phase 1: Traversing to the reference cam

Phase 1: Start

To start the reference point approach, see Sections "Axisspecific referencing (Page 1224)" and "Axisspecific referencing (Page 1224)".

Phase 1: Sequence

In Phase 1, depending on the position of the machine axis with reference to the reference cam, we distinguish between three cases:

1. The machine axis is positioned before the reference cam
2. The machine axis is positioned on the reference cam
3. The machine axis has no reference cam

Case 1: The machine axis is positioned before the reference cam

After the start of reference point approach, the machine axis is accelerated in the parameterized direction and to the parameterized reference point approach velocity :

- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (reference point approach in minus direction)
- MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)

The reaching of the reference cam must detected by querying a digital input in the PLC user program and communicated to the NC via the following interface signal:

DB31, ... DBX12.7 = 1 (reference point approach deceleration)

With detection of the NC/PLC interface signal, the machine axis is decelerated to zero speed. Whereby at least the distance s_{min} is traversed. This ensures that the machine axis leaves the reference cam in Phase 2 with the parameterized reference point creep velocity.

$$s_{min} = \frac{\text{(MD34040 \$MA\_REFP\_VELO\_SEARCH\_MARKER)2}}{\text{2 * MD32300 \$MA\_MAX\_AX\_ACCEL}}$$

Phase 1 is now complete. Reference point approach is continued with Phase 2.

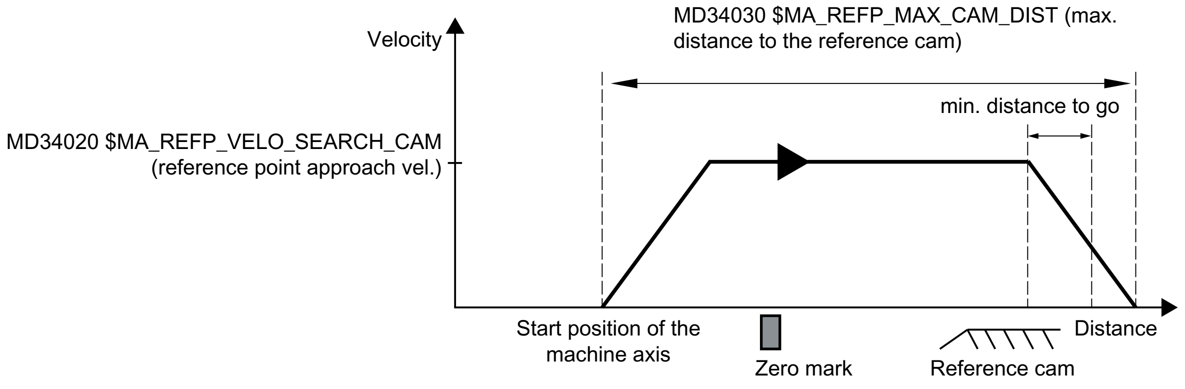

Velocity
MD34030 $MA_REFP_MAX_CAM_DIST (max. distance to the reference cam)
min. distance to go
MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach vel.)
Start position of the machine axis
Zero mark
Reference cam
Distance

Figure 16-4 Minimum distance for deceleration

Case 2: The machine axis is positioned on the reference cam

The machine axis remains at its starting position.
Phase 1 is now complete. Reference point approach is continued with Phase 2.

Case 3: The machine axis has no reference cam

Machine axes without reference cams remain at their starting position.

These include, for example:

- Machine axes that only have one zero mark along their entire traversing range
- Rotary axes that only have one zero mark per revolution

Zero must be entered in the following machine data for machine axes without a reference cam:

MD34000 $MA_REFP_CAM_IS_ACTIVE = 0 (axis with reference cam)

Phase 1 is now complete. Reference point approach is continued with Phase 2.

Phase 1: Properties

- Feedrate override is active.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.
- The machine axis is stopped if the reference cam is not reached within the parameterized maximum distance:
  MD34030 $MA_REFP_MAX_CAM_DIST (max. distance to the reference cam)

See also

Channelspecific referencing (Page 1226)

16.5.5 Phase 2: Synchronization with the zero mark

Phase 2: Graphic representation

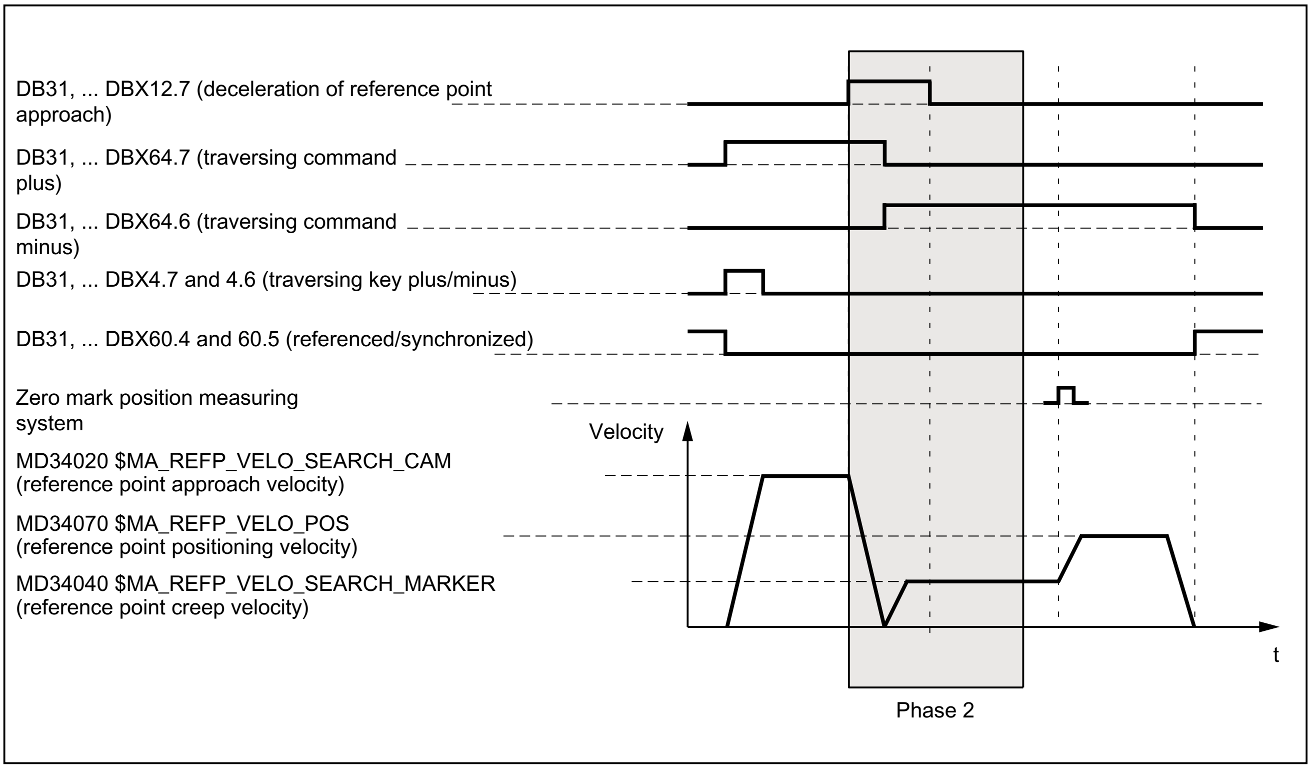

DB31, ... DBX12.7 (deceleration of reference point approach)
DB31, ... DBX64.7 (traversing command plus)
DB31, ... DBX64.6 (traversing command minus)
DB31, ... DBX4.7 and 4.6 (traversing key plus/minus)
DB31, ... DBX60.4 and 60.5 (referenced/synchronized)
Zero mark position measuring system
Velocity
MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)
MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)
MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)
t
Phase 2

Figure 16-5 Phase 2: Synchronization with the zero mark

Phase 2: Start

Phase 2 is automatically started when Phase 1 has been completed without an alarm.

Initial situation:

The machine axis is positioned on the reference cam.

Zero mark search direction:

The direction of the zero mark search results from the settings in the machine data:

- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (reference point approach in minus direction)
- MD34050 $MA_REFP_SEARCH_MARKER_REVERSE (direction reversal on reference cam)

Phase 2: Sequence

The synchronization in Phase 2 can be performed via the falling or rising edge of the reference cam. The parameterization is performed via:

MD34050 $MA_REFP_SEARCH_MARKER_REVERSE[<axis>] = <value>

Value	Meaning
0	Synchronization with falling reference cam edge
1	Synchronization with rising reference cam edge

Note

If the actual velocity of the machine axis at approach of the reference cam has not yet reached the target velocity of Phase 2 within the parameterized tolerance limits, Phase 1 will be restarted. This will be the case, for example, if the machine axis is positioned on the reference cam when reference point approach is started.

MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)

Case 1: Synchronization with falling reference cam edge

During synchronization with falling reference cam edge, the machine axis accelerates to the parameterized reference point creep velocity opposite to the parameterized reference point approach direction (traversing direction of Phase 1)

After leaving the reference cam, the machine axis waits for the next encoder zero mark:
DB31, ... DBX12.7 == 0

As soon as the encoder zero mark is detected, Phase 2 comes to an end. The machine axis continues at constant velocity and reference point approach is continued with Phase 3.

- MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

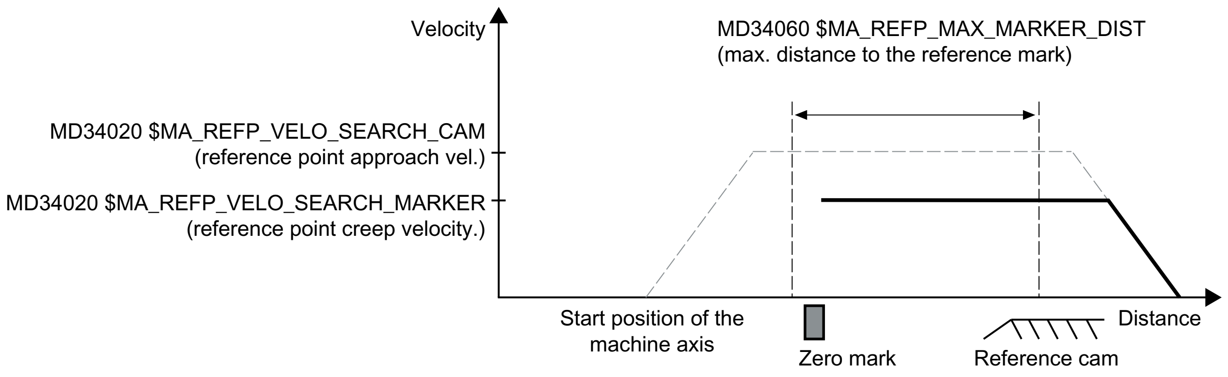

Velocity
MD34060 $MA_REFP_MAX_MARKER_DIST
(max. distance to the reference mark)
MD34020 $MA_REFP_VELO_SEARCH_CAM
(reference point approach vel.)
MD34020 $MA_REFP_VELO_SEARCH_MARKER
(reference point creep velocity.)
Start position of the machine axis
Zero mark
Reference cam
Distance

Figure 16-6 Synchronization with falling reference cam edge

Case 2: Synchronization with rising reference cam edge

During synchronization with rising reference cam signal edge, the machine axis accelerates to the parameterized reference point approach velocity against the parameterized reference point approach direction (traversing direction of the Phase 1):

- MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)
- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (reference point approach in minus direction)

After leaving the reference cam, the machine axis decelerated to standstill:
DB31, ... DBX12.7 == 0

The machine axis then travels back to the reference cam at the parameterized reference point creep velocity:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

After reaching the reference cam (DB31, ... DBX12.7 = 1), the machine axis waits for the next encoder zero mark.

As soon as the encoder zero mark is detected, Phase 2 comes to an end. The machine axis continues at constant velocity and reference point approach is continued with Phase 3.

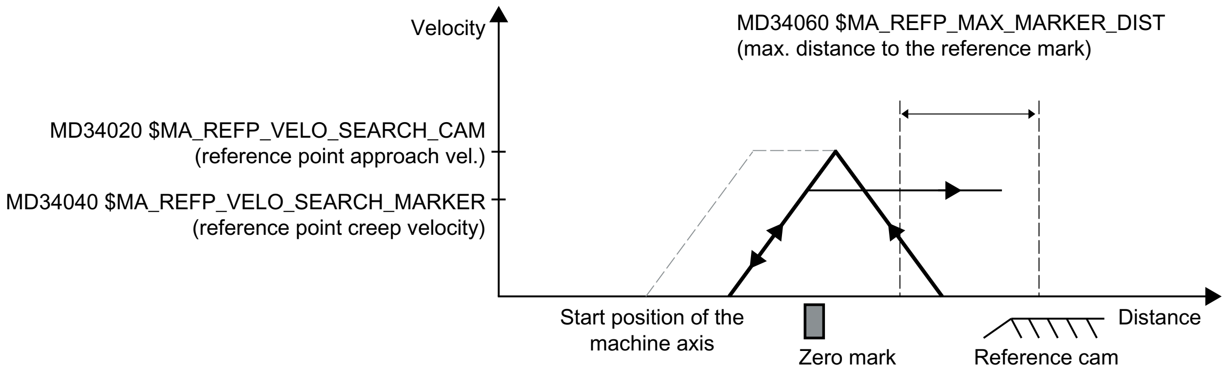

Velocity
MD34060 $MA_REFP_MAX_MARKER_DIST
(max. distance to the reference mark)
MD34020 $MA_REFP_VELO_SEARCH_CAM
(reference point approach vel.)
MD34040 $MA_REFP_VELO_SEARCH_MARKER
(reference point creep velocity)
Start position of the machine axis
Zero mark
Reference cam
Distance

Figure 16-7 Synchronization with rising reference cam edge

Electronic reference cam shift

The electronic reference cam shift is used to compensate for expansions of the reference cam caused by temperature so that synchronization is always to the same encoder zero mark:

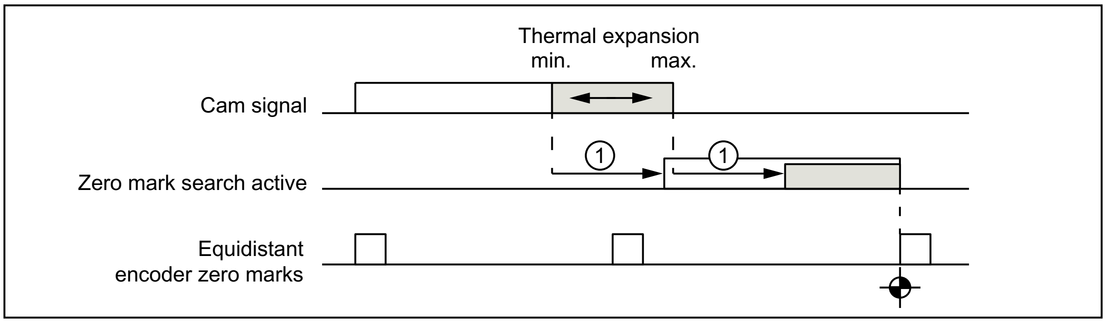
MD34092 $MA_ REFP_CAM_SHIFT (electronic reference cam shift for incremental measuring systems with equidistant zero marks)

With the electronic reference cam shift, synchronization is not performed immediately to the next encoder zero mark after detection of the reference cam edge, but only after the parameterized offset distance has been traversed.

Due to the determination of the distance traversed in the interpolator clock cycle since the detection of the reference cam edge, the effective shift distance is s_{shift}:

s_{shift_min} =	MD34092 $MA_ REFP_CAM_SHIFT
s_{shift_max} =	MD34092 $MA_ REFP_CAM_SHIFT + MD34040 $MA_REFP_VELO_SEARCH_MARKER * interpolator clock cycle

The electronic reference cam shift acts in the direction of zero mark search.



① Reference cam shift

Figure 16-8 Electronic reference cam shift

Requirement

The electronic reference cam shift is only active for machine axes with reference cam:

MD34000 $MA_REFP_CAM_IS_ACTIVE == 1

Reference cam adjustment

Encoder with equidistant zero marks

Always ensure that the reference cam of encoders that supply zero marks at equidistances is accurately adjusted so that the correct zero mark is always detected during reference point approach.

Dynamic response

The following factors influence the dynamic response from the arrival of the reference cam to the machine up to the detection of reference cam signals transferred from the PLC user program to the NC:

- Switching accuracy of the reference cam switch
- Delay of the reference cam switch (NC contact)
- Delay at the PLC input
- PLC cycle time

- Cycle time for updating the NC/PLC interface
- Interpolator clock cycle
- Position control cycle

Notes on setting

- Reference cam
  Aligning the signal edge of the reference cam directly between two zero marks has proven to be the most practical method.
- Electronic reference cam shift

  ⚠ **WARNING**

  Risk of collision

  If the reference cam adjustment is faulty or inaccurate, an incorrect zero mark can be evaluated. The controller then calculates an incorrect machine zero. As a result, the machine axis will approach the wrong positions. Software limit switches, protected areas and working area limitations act on incorrect positions and are therefore incapable of protecting the machine. The path difference is +/- of the path covered by the machine axis between two zero marks.

  Information needed for parameterizing the electronic reference cam shift is to be found in the read-only machine data:
  MD34093 $MA_REFP_CAM_MARKER_DIST (distance between reference cam/reference mark)
  The indicated value is equivalent to the distance between departure from the reference cam and detection of the reference mark. If the values are too small, there is a risk that the determination of the reference point will be non-deterministic, due to temperature effects or fluctuations in the operating time of the cam signal.

Phase 2: Properties

- Feedrate override is **not** active.
  Traversing is performed internally with feedrate override = 100%.
  If a feedrate override of 0% is specified, an abort occurs.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are **not** active.
- If the machine axis does not arrive at Phase 2 within the parameterized distance of the reference mark (encoder zero mark), the machine axis will be stopped:
  MD34060 $MA_REFP_MAX_ MARKER_DIST (max. distance to the reference mark)

16.5.6 Phase 3: Traversing to the reference point

Phase 3: Graphic representation

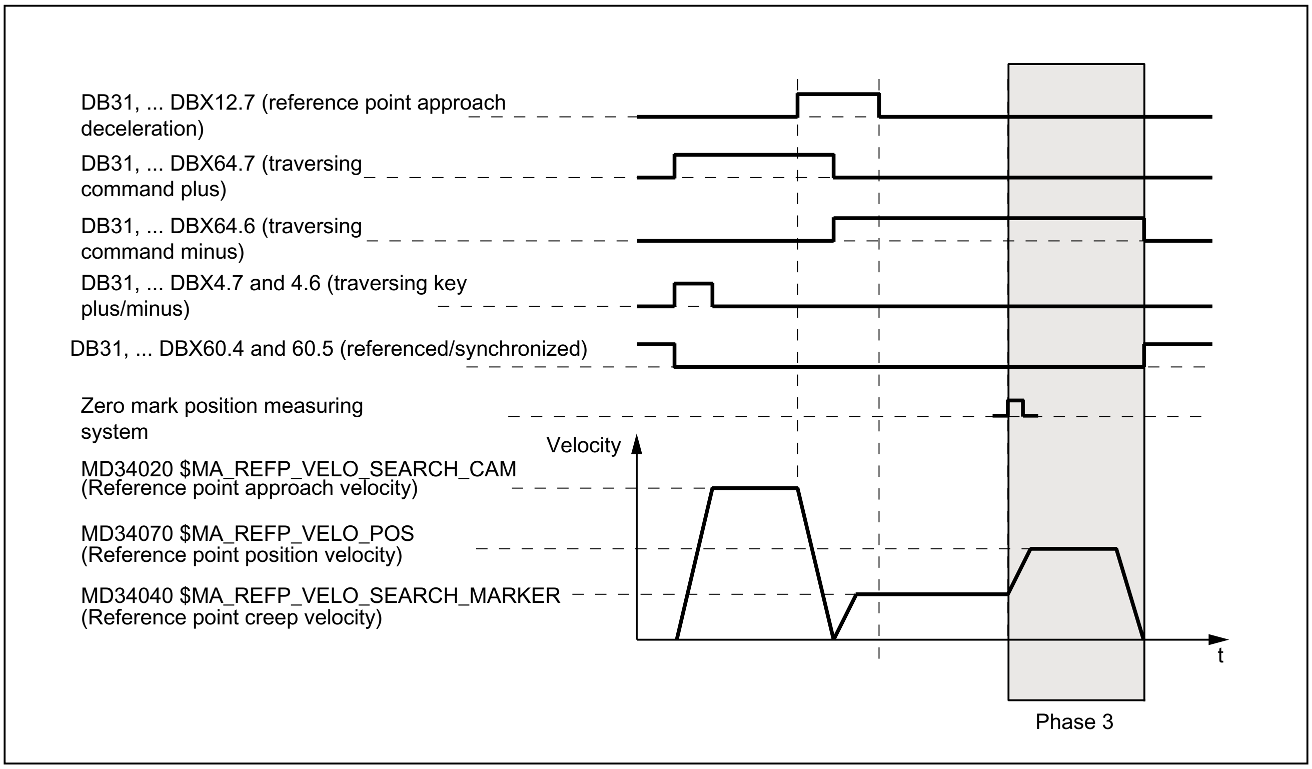

DB31, ... DBX12.7 (reference point approach deceleration)
DB31, ... DBX64.7 (traversing command plus)
DB31, ... DBX64.6 (traversing command minus)
DB31, ... DBX4.7 and 4.6 (traversing key plus/minus)
DB31, ... DBX60.4 and 60.5 (referenced/synchronized)
Zero mark position measuring system
Velocity
MD34020 $MA_REFP_VELO_SEARCH_CAM (Reference point approach velocity)
MD34070 $MA_REFP_VELO_POS (Reference point position velocity)
MD34040 $MA_REFP_VELO_SEARCH_MARKER (Reference point creep velocity)
t
Phase 3

Figure 16-9 Phase 3: Traversing to the reference point

Phase 3: Start

At the end of Phase 2 the machine axis travels at reference point creep velocity. Therefore, as soon as Phase 2 is completed successfully without an alarm, Phase 3 is started without interruption.

Initial situation

The encoder zero mark has been detected.

Phase 3: Sequence

The machine axis moves at the assigned reference point positioning velocity:
MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)
from the encoder zero mark detected in Phase 2 to the reference point.

The path s_{ref} to be covered is calculated from the sum of the reference point distance plus reference point offset:

MD34080 $MA_REFP_MOVE_DIST (reference point distance)

MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)

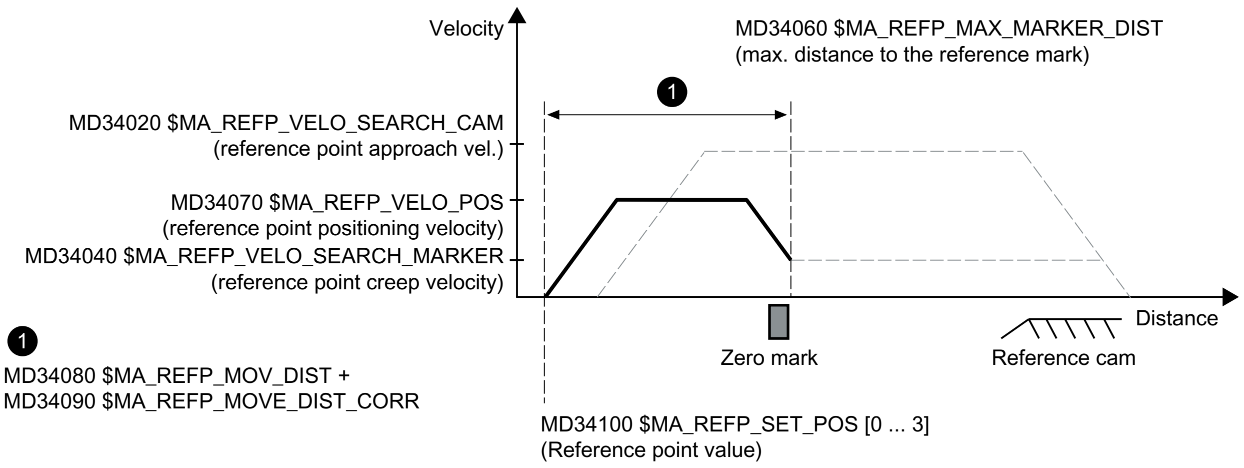

Velocity
MD34060 $MA_REFP_MAX_MARKER_DIST
(max. distance to the reference mark)
1
MD34020 $MA_REFP_VELO_SEARCH_CAM
(reference point approach vel.)
MD34070 $MA_REFP_VELO_POS
(reference point positioning velocity)
MD34040 $MA_REFP_VELO_SEARCH_MARKER
(reference point creep velocity)
Distance
Zero mark
Reference cam
1
MD34080 $MA_REFP_MOV_DIST +
MD34090 $MA_REFP_MOVE_DIST_CORR
MD34100 $MA_REFP_SET_POS [0 ... 3]
(Reference point value)

Figure 16-10 Reference point position

When the reference point is reached, the machine axis is stopped and the actual-value system is synchronized with the reference point value n specified by the NC/PLC interface.

MD34100 $MA_ REFP_SET_POS[<n>] (reference point value)

The selection of the reference point value is performed via the NC/PLC interface:

DB31, ... DBX2.4 ... 7 (reference point value 1 ... 4)

The actual-value system is synchronized to the reference point value that was selected at the time the reference cam was reached in Phase 1 (DB31, ... DBX12.7 == 1).

The machine axis is now referenced. The interface signal is set as feedback to the PLC user program, depending on the active measuring system:

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

Features of Phase 3

- Feedrate override is active.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.

Special feature of Phase 3

In the following cases, the machine axis stops first after detection of the zero mark and then traverses back to the reference point:

- Because of the reference point positioning velocity, the sum of reference point distance and reference point offset is less than the required braking distance:
  MD34080 + MD34090 < "required braking distance due to MD34070"
- The reference point is located, opposite to the current travel direction, "behind" the reference cam.

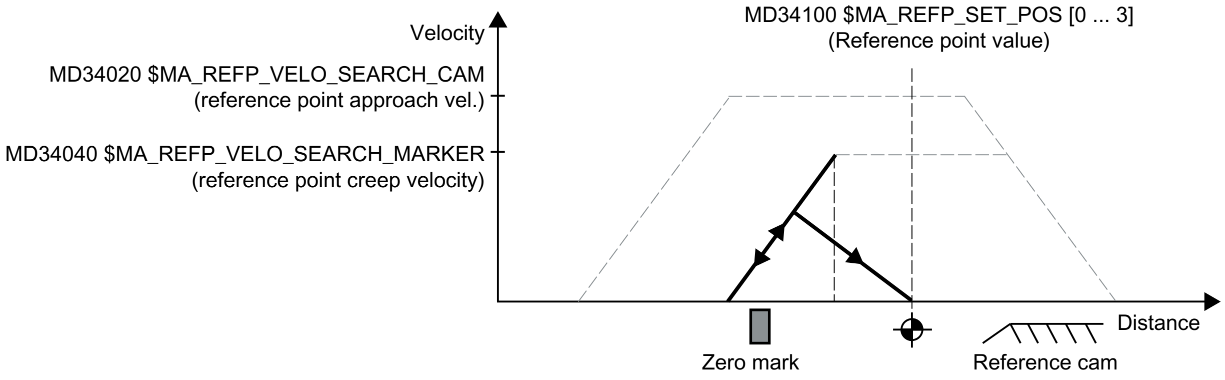

Velocity
MD34020 $MA_REFP_VELO_SEARCH_CAM
(reference point approach vel.)
MD34040 $MA_REFP_VELO_SEARCH_MARKER
(reference point creep velocity)
MD34100 $MA_REFP_SET_POS [0 ... 3]
(Reference point value)
Distance
Zero mark
Reference cam

Figure 16-11 Reference point distance plus reference point offset less than braking distance

16.6 Referencing with distance-coded reference marks

16.6.1 General overview

Distancecoded reference marks

Measuring systems with distance-coded reference marks consist of two parallel scale tracks:

- Incremental grating
- Reference mark track

The distance between any two consecutive reference marks is defined. This makes it possible to determine the absolute position of the machine axis when two consecutive reference marks are crossed. For example, if the distance between the reference marks is approx. 10 mm, a traverse path of approx. 20 mm is all that is required to reference the machine axis.

Referencing can be performed from any axis position in the positive or negative direction (exception: end of travel range).

16.6.2 Basic parameter assignment

Linear measuring systems.

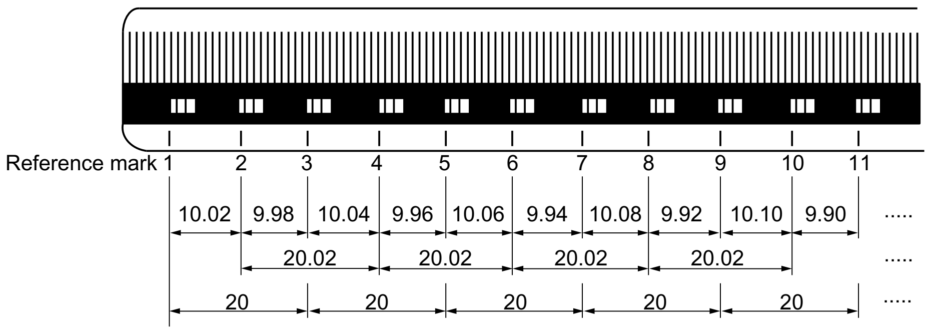


Figure 16-12 Glass measuring scale with distance-coded reference marks, grid spacing: 20 mm

The following data must be set to parameterize linear measuring systems:

- The absolute offset between the machine zero point and the position of the first reference mark of the linear measuring system:
  MD34090 $MA_REFP_MOVE_DIST_CORR (reference point/absolute offset)
  See also below: Determining the absolute offset
- Orientation of the length measuring system (equidirectional or inverse) relative to the machine system coordinate system:
  MD34320 $MA_ENC_INVERS (length measuring system inverse to the machine system)

Rotary measuring system

For rotary measuring systems, the same applies as for linear measuring systems (see above).

Determining the absolute offset

The following procedure is recommended for the determination of the absolute offset between the machine zero point and the position of the first reference mark of a machine axis:

1. Enter the value zero for the absolute offset:
   MD34090 $MA_REFP_MOVE_DIST_CORR = 0
2. Perform reference point approach.
   Note: The reference point should be approached at a point in the machine where the exact position of the machine axis relative to machine zero can be determined easily (using a laser interferometer, for example).
3. Read the displayed actual position of the machine axis in the machine coordinate system MCS from the user interface, e.g. SINUMERIK Integrate.

4. Measure the actual position of the machine axes referred to the machine zero point.
5. Calculate the absolute offset and enter in MD34090.
   The absolute offset is calculated with respect to the machine coordinate system and depending on the orientation of the measuring system as:

Orientation of the measuring system	Absolute offset
Equidirectional	(Measured position) **+** (displayed actual position)
Opposite direction	(Measured position) **-** (displayed actual position)

⚠ WARNING

Reference point deviation

After determining the absolute offset and the entry in MD34090, the reference point traversing for the machine axis must be carried out once more.

Referencing methods

Referencing with distance-coded reference marks can be performed in one of two ways:

- Evaluation of **two** consecutive reference marks:
  MD34200 $MA_ENC_REFP_MODE = 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

16.6.3 Time sequence

Time sequence

Referencing with distance-coded reference marks can be divided into two phases:

- Phase 1: Travel across the reference marks with synchronization
- Phase 2: Travel to a fixed destination point

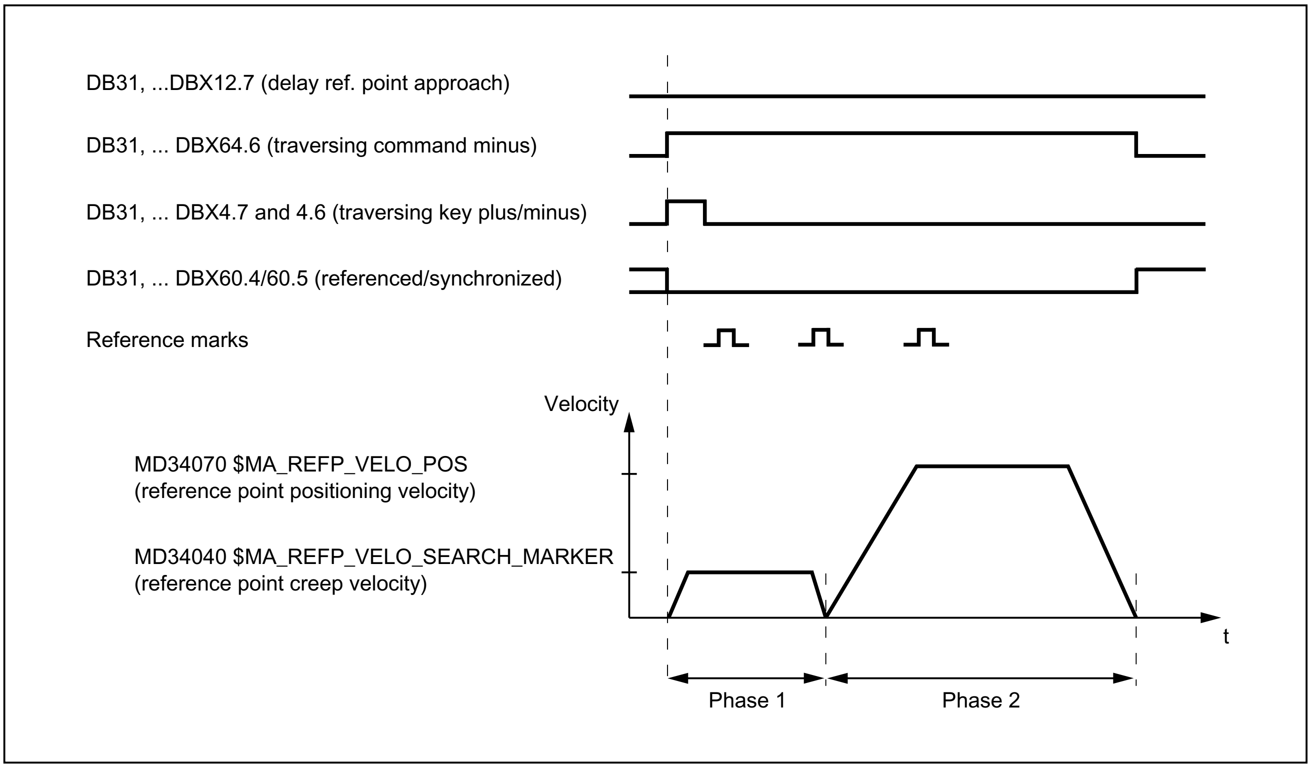

DB31, ...DBX12.7 (delay ref. point approach)
DB31, ... DBX64.6 (traversing command minus)
DB31, ... DBX4.7 and 4.6 (traversing key plus/minus)
DB31, ... DBX60.4/60.5 (referenced/synchronized)
Reference marks
Velocity
MD34070 $MA_REFP_VELO_POS
(reference point positioning velocity)
MD34040 $MA_REFP_VELO_SEARCH_MARKER
(reference point creep velocity)
t
Phase 1
Phase 2

Figure 16-13 Distance-coded reference marks

16.6.4 Phase 1: Travel across the reference marks with synchronization

Phase 1: Start

To start the reference point approach, see Sections "Axisspecific referencing (Page 1224)" and "Channelspecific referencing (Page 1226)".

Reference cam

In measuring systems with distance-coded reference marks, reference cams are not required for the actual referencing action. For functional reasons, however, a reference cam is required for channel-specific reference point approach and reference point approach from the part program (`G74`) before the traversing range end of the machine axis.

Phase 1: Sequence

Sequence without contact witha reference cam

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Sequence when starting from the reference cam

If the machine axis is at the reference cam at the start of the reference point traversing, it accelerates to the parameterized reference point creep velocity against the parameterized reference point approach direction:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

MD34010 $MA_CAM_DIR_IS_MINUS (reference point approach in minus direction)

That ensures that the machine axis does not reach the travel range limit before it has crossed the parameterized number of reference marks.

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Sequence when contact is made with reference cam during referencing

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Before the machine axis travels over the parameterized number of reference marks, it makes contact with the reference cam. It is then reversed and reference mark search is restarted in the opposite direction.

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Plausiblity check of the reference mark distance

An error occurs if, during reference point traversing for two subsequent reference marks, the NC determines a distance greater than twice the parameterized reference mark distance.

MD34300 $MA_ENC_REFP_MARKER_DIST (reference mark distance)

The machine axis will then traverse in opposite direction at half the parameterized reference point creep velocity (MD34040) and the search for reference mark is restarted.

If a faulty reference mark distance is detected again, the machine axis is stopped and the reference point traversing is aborted (alarm 20003 "fault in the measuring system").

Abort criterion

If the parameterized number of reference marks is not detected within the parameterized distance, the machine axis is stopped and reference point traversing is aborted.

MD34060 $MA_REFP_MAX_ MARKER_DIST (max. distance to the reference mark)

Features of Phase 1

After Phase 1 is successfully completed, the actual value system of the machine axis is synchronized.

16.6.5 Phase 2: Traversing to the target point

Phase 2: Start

Phase 2 is automatically started when Phase 1 has been completed without an alarm.

Initial situation:

- The machine axis is positioned directly behind the last of the parameterized number of reference marks.
- The actual value system of the machine axis is synchronized.

Phase 2: Sequence

In Phase 2, the machine axis completes reference point approach by traversing to a defined target position (reference point). This action can be suppressed in order to shorten the reference point approach:

MD34330 $MA_STOP_AT_ABS_MARKER = <value>

Value	Meaning
0	Travel to target position
1	No travel to target position

Travel to target position (normal case)

The machine axis accelerates to the parameterized reference point position velocity and travels to the parameterized target point (reference point):

MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)

MD34100 $MA_REFP_SET_POS (reference point value)

The machine axis is referenced. To identify this, the NC sets an interface signal for the measuring system that is currently active:

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

No travel to target position

The machine axis is now referenced. To identify this, the NC sets an interface signal for the measuring system that is currently active:

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

Features of Phase 2

Phase 2 will display different characteristics, depending on whether a reference cam is parameterized for the machine axis.

Machine axis without reference cam

MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) = 0

Properties:

- Feedrate override is active.
- The feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.

Machine axis with reference cam

MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) = 1

Properties:

- Feedrate override is **not** active.
  Traversing is performed internally with feedrate override = 100%.
  If a feedrate override of 0% is specified, an abort occurs.
- The feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are **not** active.
- If the parameterized number of reference marks is not detected within the parameterized distance after the exit of the reference cam, the machine axis will be stopped.
  MD34060 $MA_REFP_MAX_ MARKER_DIST (max. distance to the reference mark)

Special features of rotary measuring systems

On rotary distance-coded measuring systems, the absolute position can only be determined uniquely within one revolution. Depending on the mechanical mounting of the encoder, the overtravel of the absolute position in the hardware does not always coincide with the traversing range of the rotary axis.

Special features of modulo rotary axes

With module rotary axes, the reference point position is mapped on the parameterized modulo range:

MD30330 $MA_MODULO_RANGE (size of the modulo range)

MD30340 $MA_MODULO_RANGE_START (start position of the modulo range)

Note

The reference point position is mapped on the parameterized (fictive) 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

16.7 Referencing by means of actual value adjustment

16.7.1 Actual value adjustment to the referencing measurement system

Function

When actual value adjustment to the referencing measuring system is performed, the resulting absolute actual position after successful referencing of the measuring system of a machine axis is transferred directly to all other measuring systems of the machine axis, and the machine axis is designated as referenced:

DB31, ... DBB60.4 / 60.5 (referenced/synchronized 1/2) = 1

Advantage

When the machine axis switches from an explicitly referenced measuring system to the measuring system referenced by actual value adjustment, continuous servo control is assured (servo enable active) because the matched actual position prevents a sudden change in actual value.

Note

In order to improve positioning precision by determining the measuring-system-specific encoder fine information, we recommend explicitly re-referencing the measuring system previously referenced by actual value adjustment after switching over.

Activation

The activation of the actual value adjustment to the referencing measuring system is machine-specifically carried out via:

MD34102 $MA_REFP_SYNC_ENCS = 1

16.7.2 Actual value adjustment for measuring systems with distance-coded reference marks

Function

In order to improve positioning precision by determining the measuring-system-specific encoder fine information, we recommend explicitly re-referencing the measuring system previously referenced by actual value adjustment after switching over the measuring system.

If an encoder with distance-coded reference marks is used for the passive measuring system, referencing can be avoided under the following conditions:

1. Active measuring system: Indirect measuring system (motor measuring system) with absolute encoder, for example
2. Passive measuring system: Direct measuring system with distance-coded reference marks
3. Traversing motion 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. Whereby, the passive direct measuring system is referenced automatically.

Parameterization

In addition to the specific machine data required to reference the individual measuring systems, the following machine data must be set:

- Enable the actual value adjustment:
  MD34102 $MA_REFP_SYNC_ENCS = 1
- Direct measuring system with distance-coded reference marks:
  - MD34200 $MA_ENC_REFP_MODE[*measuring system*] = 3
    Distance-coded reference marks
  - MD30242 $MA_ENC_IS_INDEPENDENT[*measuring system*] = 2
    In the actual value adjustment, the passive direct measuring system is aligned to the actual position of the active indirect measuring system, but not marked as referenced. After the parameterized number of reference marks is crossed, the passive direct measuring system is referenced automatically. Referencing is performed in every operating mode.

Sequence

1. Initial situation: Both measuring systems are not referenced:
   DB31, ... DBX60.4 = 0 (referenced/synchronized 1)
   DB31, ... DBX60.5 = 0 (referenced/synchronized 2)
2. Referencing of 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. Traversing of 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)

16.8 Referencing in follow-up mode

Function

Incremental measuring systems and measuring systems with distance-coded reference marks can be referenced even when the machine axis is in follow-up mode. Prerequisite for this is the correct parameterization of the reference point approach according to the used measuring system (see Section "Referencing with incremental measurement systems (Page 1228)" and "Referencing with distance-coded reference marks (Page 1241)").

When referencing in follow-up mode the machine axis is moved not by the NC but by means of an external travel motion over the encoder zero mark and the parameterized number of distance-coded reference marks. The measuring system is referenced when the encoder zero mark or parameterized number of distance-coded reference marks are detected.

Note

Reproducibility of the referencing result

In NC-guided reference point approach, reproducibility of the referencing result is ensured through adherence to the assigned traverse velocities during the referencing operation. During referencing in follow-up mode, responsibility for achieving reproducibility of the referencing results lies with the machine manufacturer / user.

Unique zero mark

Referencing of an incremental measuring system is based on the explicit position of the encoder zero mark relative to the overall traversing range of the machine axis.

Because the reference cam signal is not evaluated by the NC during referencing in follow-up mode, unique identification of the reference point when referencing in follow-up mode will only result with:

- Only one encoder zero mark in the traversing range of the machine axis.
- Linear measuring systems with distance-coded reference marks.
- Modulo rotary axes (absolute position within one revolution).

Zero mark selection when several zero mark signals occur

If several encoder zero marks are detected in the traversing range of the machine axis due to machine-specific factors, e.g. reduction gear between encoder and load, a proximity switch must be mounted on the machine and connected via a digital input of the NCU interface in order to clearly determine the reference point.

Note

Contact-less proximity switch signal: Zero mark selection

The processing of the proximity switch signal is performed exclusively in the drive. Connection and parameterization, see Section "Hardware signals (Page 1228)".

Enable

The "Referencing in follow-up mode" function is enabled with:

MD34104 $MA_REFP_PERMITTED_IN_FOLLOWUP = TRUE

Starting the referencing operation

If the machine axis is in follow-up mode (DB31, ... DBX61.3 == TRUE) at the start of reference point approach, the measuring system is referenced in follow-up mode.

If the machine axis is not operating in the follow-up mode at the start of reference point traversing, the "normal" from the NC-controlled reference point travels is carried out.

Referencing in follow-up mode can be started in the following modes:

- JOG-REF: Traversing keys
- AUTOMATIC: Part program command `G74`

Sequence of the referencing operation (JOG-REF mode)

1. Activate follow-up mode of machine axis:
   DB31, ... DBX1.4 (follow-up mode) = 1
   DB31, ... DBX2.1 (controller enable) = 0
2. Wait for activation of follow-up mode:
   DB31, ... DBX61.3 (follow-up active) == 1
3. Switch to JOG mode, REF machine function
4. External motion of machine axis across encoder zero mark or parameterized number of distance-coded reference marks. The referencing operation is started internally in the NC as soon as the machine axis is moved: The following NC/PLC interface signal is reset as feedback:
   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: The following NC/PLC interface signal is set as feedback:
   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 command `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) = 1
   DB31, ... DBX2.1 (controller enable) = 0
4. Wait for activation of follow-up mode:
   DB31, ... DBX61.3 (follow-up active) == 1
5. The referencing operation is started internally in the NC as soon as `G74` command has been processed.
6. External motion of machine axis across encoder zero mark or parameterized number of distance-coded reference marks.
7. The measuring system is referenced after the encoder zero mark or the assigned number of distance-coded reference marks have been successfully detected: The following NC/PLC interface signal is set as feedback:
   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 re-referenced.

16.9 Referencing with absolute encoders

16.9.1 Information about the adjustment

Machine axes with absolute encoder

The advantage of machine axes with absolute encoder is that after a one time adjustment procedure, the necessary reference point traversing with incremental measuring systems (e.g. build-up of control, de-selection of "Parking" of machine axes etc.) can be skipped and the actual value system of the machine axis can be immediately synchronized to the determined absolute position.

Adjustment

Adjustment of an absolute encoder involves matching the actual value of the encoder with the machine zero once and then setting it to valid.

The current adjustment status of an absolute encoder is displayed in the following axis-specific machine data of the machine axis, to which it is connected:

MD34210 $MA_ENC_REFP_STATE (status of absolute encoder)

Value	Meaning
0	Encoder not calibrated
1	Encoder adjustment enabled
2	Encoder is calibrated

Adjustment methods

The following adjustment methods are supported:

- Adjustment by entering a reference point offset
- Adjustment by entering a reference point value
- Automatic adjustment with probe
- Adjustment by means of the proximity switch

Readjustment

Readjustment of the absolute encoder is required after:

- Gear change between load and absolute encoder
- Removal/installation of the absolute encoder
- Removal/installation of the motor with the absolute encoder
- Data loss in the static NC memory

- Battery failure
- Set actual value (`PRESETON`)

⚠ WARNING

Data backup

During the back-up of machine data of a machine A, the encoder status of the machine axis (MD34210) is also backed up.

During loading of this data record into a machine B of the same type, e.g. in the context of a serial start-up or after a case of maintenance, the referenced machine axes will be automatically regarded as adjusted / referenced by the NC. It is the special responsibility of the machine manufacturer / user to undertake a readjustment in such cases.

See also explanations regarding machine data:

MD30250 $MA_ACT_POS_ABS (absolute encoder position at the time of switch-off)

Note

The controller can detect a required readjustment of the absolute encoder only during the following events:

- Gear change with change of gear ratio
- Addressing the zero-mark monitoring
- New encoder serial number after change of the absolute encoder

The controller then sets the status of the absolute encoder to "0":

MD34210 $MA_ENC_REFP_STATE = 0 (encoder not adjusted)

The following alarm is displayed:
Alarm 25022 "Axis <axis name> encoder <number> warning 0"
If the zero-mark monitoring responds, the following alarm is also displayed:
Alarm 25020 "Axis <axis name> zero-mark monitoring of active encoder"

In all other cases (e.g. `PRESETON`) it is the sole responsibility of the user to display the misalignment of the absolute encoder by **manually** setting the status to "0" and to carry out a readjustment.

16.9.2 Calibration by entering a reference point offset

Function

During adjustment by entering the reference point offset, the difference between the position displayed on the operator interface and the true actual position in the machine is determined and made known to the NC as reference point offset.

Procedure

1. Determining the position of the machine axis with reference to the machine zero point via e.g.:
   - position measurement (e.g. laser interferometer)
   - Moving the machine axis to a known position (e.g., fixed stop)
2. Reading the displayed actual position of the machine axis on the operator interface.
3. Calculating the reference point offset (difference between the actual positions determined under point 1 and 2) and entering in machine data:
   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
4. Marking the absolute value encoder as adjusted:
   MD34210 $MA_ENC_REFP_STATE = 2

   Note

   The encoder adjustment does not become active until the next time the encoder is activated (e.g., when the controller is powered up).

5. Initiate POWER ON reset.
6. Controlling the position of the machine axis displayed on the operator interface.

   Note

   Backlash compensation

   If backlash compensation is parameterized for a measuring system with absolute value encoder, the following must be observed:

   No backlash is permitted during machine axis travel to the adjusted machine position.

   Activate reference point offset permanently

   The entered reference point offset (MD34090) will be permanently active only after initial POWER ON - Reset. If the machine axis is moved after the absolute encoder adjustment without an interim POWER ON - Reset, the reference point offset entered in the machine data can be overwritten, for example, as part of internal overrun offset.

   Checking the actual position

   Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (POWER ON).

16.9.3 Adjustment by entering a reference point value

Function

During adjustment by entering the reference point value, the absolute position of the machine axis with reference to the machine zero point is determined by e.g.:

- Position measurement (e.g. laser interferometer)
- Moving the machine axis to a known position (e.g. fixed stop)

This determined position value will be made known to the NC as the reference point value. The NC then calculates the reference point offset from the difference between the encoder absolute value and the reference point value.

Procedure

1. Check whether the referencing mode is set to "Assume the reference point value":
   MD34200 $MA_ENC_REFP_MODE == 0
   If not, enter the value 0 in the machine and trigger a Power On reset.
2. Traverse the machine axis in JOG mode to the position to be measured (e.g. with a laser interferometer) 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. Enter the position of the machine axis relative to machine zero as the reference point value in the machine data:
   MD34100 $MA_REFP_SET_POS = *Position*
4. Enable encoder adjustment:
   MD34210 $MA_ENC_REFP_STATE = 1
5. Trigger a Power On reset for acceptance of the entered machine data values.
6. Switch to JOG-REF mode.

7. Operate the travel key used for referencing in step 2.
   The machine axis does not move when the traversing key is actuated!
   The NC calculates the reference point offset from the entered reference point value and that given by the absolute encoder. The result is entered into the machine data:
   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
   The status of the absolute encoder is set to "Encoder is adjusted":
   MD34210 $MA_ ENC_REFP_STATE = 2
   The actual value system of the machine axis is synchronized.
   The machine axis is now referenced. As identification, the NC sets the appropriate interface signal based on which measuring system is currently active:
   DB31, ... DBB60.4 / 60.5 (referenced/synchronized 1 / 2) = 1
8. Initiate Power On reset.

Note

Activate reference point offset permanently

The entered reference point offset (MD34090) will only be permanently active after Power On reset.

If the machine axis is moved after the absolute encoder adjustment without an interim Power On reset, the reference point offset entered in the machine data can be overwritten, for example, within internal overrun corrections.

Checking the actual position

Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (Power On).

16.9.4 Automatic calibration with probe

Function

In automatic adjustment with a probe, a known position in the machine is approached with the machine axis from a part program. The position value is stored in the NC as a reference point value. The position is reached when the probe switches, and the NC then calculates the reference point offset from the difference between the encoder value and reference point value.

Note

Part program for automatic adjustment

The part program for automatic adjustment using a probe must be created by the machine manufacturer / user for the specific requirements of the machine.

Freedom from collision

Because actual-value-related monitoring is not active for the machine axes being referenced, the machine operator must take special care to ensure that collisions do not occur in the machine while the machine axes are being moved!

Part program

The part program for automatic adjustment of absolute encoders with probe must perform the points listed below for each axis in the order indicated:

1. Approach the adjustment position of machine axis, which is detected from the probe response.
   The position must be approached several times from the same direction, but at a velocity which is gradually reduced on each approach, to ensure that the measured value obtained is as accurate as possible. The measured value is stored in system variable $AA_IM.
2. Calculating and writing the reference point offset:

```
MD34090 $MA_REFP_MOVE_DIST_CORR = MD34100 $MA_REFP_SET_POS – $AA_IM
```

3. Set the absolute encoder status to "Encoder is adjusted":
   MD34210 $MA_ ENC_REFP_STATE = 2

Sequence

Proceed as follows for automatic adjustment with probe:

1. Enable part program start even for non-referenced machine axes:
   MD20700 $MC_REFP_NC_START_LOCK = 0
2. Enter the machine axis position relative to machine zero when probe is switched as the reference point value for all relevant machine axes:
   MD34100 $MA_REFP_SET_POS = reference point value
3. Activate NCK-Reset for the acceptance of the entered machine data values.
4. Start part program.
5. After completion of the part program, re-secure the partial program start for machine axes which are not referenced:
   MD20700 $MC_REFP_NC_START_LOCK = 1
6. Initiate POWER ON - Reset so that the reference point offset written by the part program is permanently active:
   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)

Note

Activate reference point offset permanently

The entered reference point offset (MD34090) will only be permanently active after POWER ON - Reset.

If the machine axis is moved after the absolute encoder adjustment without an interim POWER ON - Reset, the reference point offset entered in the machine data can be overwritten, for example, as part of internal overrun offset.

Checking the actual position

Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (POWER ON).

16.9.5 Adjustment with BERO

Function

For adjustment using proximity switch, a reference point approach to a defined machine position is performed the same as for incremental measuring systems. In this case, the proximity switch replaces the encoder zero mark that the absolute encoder does not have. After successful completion of reference point approach, the NC automatically calculates the reference point offset from the difference between the encoder absolute value and the parameterized reference point value.

Parameterization

NC: Referencing mode

The referencing mode should be set to "Referencing of incremental, rotary or linear measuring systems: Zero pulse on the encoder track":

MD34200 $MA_ENC_REFP_MODE[<axis>] = 1

NC: Reference point value

The reference point value is parameterized via:

MD34100 $MA_REFP_SET_POS[<axis>] = <reference point value>

Drive: Equivalent zero mark

The digital input on the NCU interface to which the proximity switch is connected must be set in parameter p0494 or p0495.

Execution

Reference point approach can be started manually in JOG-REF mode or in AUTOMATIC or MDI mode via a part program (`G74`).

After successful completion of the reference point approach, the absolute encoder is adjusted and the actual-value system of the machine axis is synchronized.

As feedback for the PLC user program, the NC sets the NC/PLC interface signal for the machine axis, depending on the active measuring system:

DB31, ... DBB60.4/60.5 (referenced/synchronized 1/2) = 1

Note

If the proximity switch is removed after adjustment of the absolute encoder, the referencing mode must be re-parameterized to "Referencing with absolute encoder".

MD34200 $MA_ENC_REFP_MODE[<axis>] = 0

16.9.6 Reference point approach with absolute encoders

Parameter assignment

Traversing movement release

If for a machine axis with adjusted absolute value encoder as active measuring system, reference point traversing is activated (manually in the JOG-REF mode or automatically via the part program instruction `G74`), the machine axis travels depending on the parameterized traversing movement release.

MD34330 $MA_REFP_STOP_AT_ABS_MARKER = <value>

Value	Meaning
0	Traversing is **enabled**. When reference point approach is initiated, the machine axis moves to the reference point position. When reaching the reference point position, the reference point approach is completed.
1	Traversing is **not** enabled. After the activation of the reference point travel, the machine axis does not travel and the reference point travel is immediately completed.

16.9.7 Reference point approach for rotary absolute encoders with equivalent zero mark

Function

To ensure that the reference point approach via zero mark (see Section "Referencing with incremental measurement systems (Page 1228)") can also be used with absolute encoders, the zero mark, which is not provided in the hardware of absolute encoders, is simulated. For this, the controller generates the signal for the equivalent zero mark once per encoder revolution, always at the same position within the revolution.

Difference compared to referencing with incremental encoders

An absolute encoder with replacement zero mark should not be considerd as a complete equivalent of an incremental encoder. All the properties of the absolute encoder are retained. The following table lists the different properties of incremental and absolute encoders:

Table 16-1 Properties of incremental and absolute encoders

Property	Incremental encoder	Absolute encoder
Encoder type	MD30240 $MA_ENC_TYPE =	
	1	4
Internal encoder position	MD30250 $MA_ACT_POS_ABS =	
	Value is updated only in MD34210 ≥ 1	Value is updated only in MD30270 = 0
Traversing range extension	MD30270 $MA_ENC_ABS_BUFFERING =	
	No effect	= 0 (default): Active

Property	Incremental encoder	Absolute encoder
Reference point offset	MD34090 $MA_REFP_MOVE_DIST_CORR =	
	Value input allowed	Value is updated exclusively via control
Supported referencing types	MD34200 $MA_ENC_REFP_MODE =	
	1, 3, 4, 8	0, 1
Adjustment status	MD34210 $MA_ENC_REFP_STATE = 0, 1, 2	
	Automatic encoder misalignment during shut down while in motion.	Automatic encoder misalignment during parameter set change with position jump or during serial number change.
Absolute position modulo range	MD34220 $MA_ENC_ABS_TURNS_MODULO =	
	0	1 - 4096
Encoder serial no.	MD34230 $MA_ENC_SERIAL_NUMBER =	
	0	The value must be updated from the PLC during each encoder change, otherwise loss of adjustment plus alarm.
Transfer of series startup files	Without any restrictions.	Due to encoder characteristics (MD30250, MD30270, MD34090, MD34210, MD34220, MD34230) only possible with certain restrictions.
Activation time	0 seconds	several seconds
Zero mark	1 per encoder revolution	None
Zero mark monitoring	Hardware	Software
Position after POWER ON without actual value buffering	0.0	Last position within MD34220.
	MD34210 = 0	MD30270 = 1
Position after POWER ON with actual value buffering	Last standstill position before deactivation.	Last position including small movements during POWER OFF.
	MD34210 = 1	MD30270 = 0
Referenced after POWER ON	depends on adjustment status	

Requirement

The function can be used only with rotary absolute encoders:

- MD31000 $MA_ENC_IS_LINEAR == 0
- MD30240 $MA_ENC_TYPE == 4

Parameterization

Reference point approach with equivalent zero mark:

MD34200 $MA_ENC_REFP_MODE = 1

Supplementary conditions

- A reference point offset (MD34090 $MA_REFP_MOVE_DIST_CORR) may not be parameterized.
  This MD describes, in connection with absolute encoders, the offset between machine and absolute encoder zero, and it therefore has a different meaning.
- The load-side zero mark search rate (MD34040 $MA_REFP_VELO_SEARCH_MARKER) should not exceed the limiting frequency of the absolute trace of the encoder (MD36302 $MA_ENC_FREQ_LIMIT_LOW)
  . If the speed is too high, absolute information cannot be read any more, and thus, no equivalent zero marks are generated.
- If no zero mark is found within the parameterized path (MD34060 $MA_REFP_MAX_MARKER_DIST), an alarm is output:
- The following MD must be set if the absolute encoder retains even the referenced status through POWER OFF, besides the last position:
  MD34210 $MA_ENC_REFP_STATE = 2

Data backup and standard commissioning

Some properties of an absolute encoder restrict the transfer of a commissioning archive to other machines. The following machine data must be checked and possibly corrected after loading a commissioning archive to the controller:

- MD30250 $MA_ACT_POS_ABS (internal encoder position)
- MD30270 $MA_ENC_ABS_BUFFERING (traversing range extension)
- MD34090 $MA_REFP_MOVE_DIST_CORR (absolute offset)
- MD34210 $MA_ENC_REFP_STATE (adjustment status)
- MD34220 $MA_ENC_ABS_TURNS_MODULO (modulo range)
- MD34230 $MA_ENC_SERIAL_NUMBER (encoder serial number)

16.9.8 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, ... DBB1.5 / 1.6 (position measuring system 1/2)
  DB31, ... DBB2.1 (servo enable)
- Violation of the assigned encoder limit frequency (spindles):
  MD36300 $MA_ENC_FREQ_LIMIT

When the measuring system is activated, the NC synchronizes the actual value system of the machine axis with the current absolute value. Traversing is disabled during synchronization for axes but not for spindles.

Parameterizing the encoder limit frequency (spindles)

The EQN 1325 absolute encoder made by Heidenhain has an incremental track and an absolute track.

If a spindle is driven at a speed above the encoder limit frequency of the incremental track, the substantially lower limit frequency of absolute track must be parameterized as the encoder limit frequency.

MD36300 $MA_ENC_FREQ_LIMIT

Otherwise an incorrect absolute position would be read because the parameterized encoder limit frequency is not reached when the measuring system is activated. This would cause a position offset in the actual value system of the machine axis.

Determining the encoder limit frequency

The encoder limit frequency to be parameterized is derived from the smaller of the two following limit speeds:

- Encoder
  The limit speed or encoder limit frequency is listed in the data sheet of the encoder (e.g., limit speed = 2000 [rpm])
- NC
  Due to the NC-internal evaluation process, the maximum limit speed for which error-free calculation of the absolute value by the NC is possible is 4 encoder revolutions per interpolator clock cycle.
  For an interpolator clock cycle of, for example, 12 ms: Limit speed = 4 / 12 ms = 20,000 rpm
  The limiting frequency corresponding to the limiting speed is calculated to be:

$$\text{MD36300} = \frac{4 * \text{MD31020}}{\text{MD10050} * \text{MD10070}}$$

MD31020 $MA_ENC_RESOL (encoder pulses per revolution)
MD10050 $MN_SYSCLOCK_CYCLE_TIME (basic system clock cycle)
MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO (factor for the interpolation cycle)

Note

The position control switching speed relevant for spindles is set according to the encoder limiting frequency of the absolute value encoder of the spindle:

MD35300 $MA_SPIND_POSCTRL_VELO (position control switching speed)

MD36300 $MA_ENC_FREQ_LIMIT (Encoder limit frequency)

16.9.9 Referencing variants not supported

The following referencing variants are not supported when used with absolute encoders:

- Referencing/calibrating with encoder zero mark
- Distance-coded reference marks
- Proximity switch with two-edge evaluation

16.10 Automatic restoration of the machine reference

Without a defined machine reference when traversing machine axes, no position-dependent functions such as transformations or tool frames can be executed. In various machine situations, these functions must be available immediately with the encoder activation, e.g. after the control is switched on or after terminating "Parking a machine axis (Page 125)" to traverse the axes. However, the machine axes should not or cannot be traversed again for referencing.

Absolute encoders

For measuring systems with adjusted absolute encoders, the machine reference is restored immediately without any additional measures when the encoder value is read.

Incremental encoders

With incremental measuring systems, the machine reference can be restored without traversing the axes through "Automatic referencing" or "Restoration of the actual position".

Supplementary conditions

⚠ WARNING

Incorrect synchronization of the position measuring system caused by offset of the actual machine axis position

During the time in which the measuring system of the machine axis is switched off, it is **not** permissible that the axis is mechanically moved. Otherwise this results in an offset between the last buffered actual position and the real actual position of the machine axis. This would lead to an incorrect synchronization of the measuring system resulting in danger to personnel and machine.

The machine manufacturer must provide such measures as holding brakes, etc. on the machine so that the actual position is not changed, and this must be ensured by the user. The responsibility for this rests exclusively with the machine manufacturer / user.

If axis motion cannot be prevented mechanically in the shutdown state, either an absolute encoder must be used or the axis must be referenced again with reference point approach after switching on.

Note

SMExx sensor modules

Automatic referencing or restoration of the actual position to the last buffered position after restarting the control is only possible in conjunction with SM**E**xx (externally mounted) sensor modules. When using SM**C**xx (cabinet) or SM**I**xx (integrated) sensor modules, the actual position **cannot** be restored after restarting the control (power on). The measuring system of the machine axis must be referenced again.

16.10.1 Automatic referencing

Function

During automatic referencing, the actual position of the active measuring system of the machine axis is set to the last buffered position and "referenced" set as encoder state after switching on the control. This makes it possible to start programs in the AUTOMATIC and MDI modes directly after run-up of the control.

Requirements

- The active measuring systems when the control is switched on must already have been referenced once before switching off.
- At the time the control is switched off, the machine axis must be at standstill with "Exact stop fine" (DB31, ... DBX60.7 == 1) are located.

  Note

  If the machine axis is **not** at standstill with "Exact stop fine" when switching off, the actual position will be initialized with " 0" when switching on. "Not referenced" is displayed as the encoder state.

Parameterization

The automatic referencing is enabled by setting the encoder state to "Automatic referencing is enabled, but the encoder has not been referenced":

MD34210 $MA_ENC_REFP_STATE[<encoder>] = **1**

After the measuring system has been referenced, the encoder state displays that automatic referencing will be executed the next time the encoder is activated:

MD34210 $MA_ENC_REFP_STATE[<encoder>] == **2**

NC/PLC interface signals

After automatic referencing, the encoder state "Referenced" is displayed for the active measuring system:

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

Supplementary conditions

Encoder activation with MD34210 $MA_ENC_REFP_STATE[<encoder>] == 1

An encoder state equal to "1" at the time of the encoder activation means that "Automatic referencing" has been enabled. However, the measuring system has either not been referenced yet or the machine axis was not switched off at standstill in the "Exact stop fine" state. The following is set for the machine axis or the active measuring system:

- Actual position = 0
- Active measuring system, encoder state = "Not referenced":
  DB31, ... DBX60.4 / .5 = **0** (referenced/synchronized 1/2)

References

Description of Functions, Basic functions, Section "R1 Referencing" > "Referencing with incremental measurement systems (Page 1228)" > ""

16.10.2 Restoration of the actual position

Function

When restoring the actual position to the last buffered position, the encoder state of the active measuring system is set to "**Restored**". The axis can only be traversed manually.

AUTOMATIC mode

To enable NC start for the automatic execution of programs in the AUTOMATIC mode, the measuring system of the machine axis must be re-referenced.

MDI mode and overstore

In the MDI mode and for the overstore function, machining can also be performed, without referencing the axes, with restored positions. To do this, NC start with restored positions must be enabled explicitly for a specific channel:

MD20700 $MC_REFP_NC_START_LOCK = 2

Requirement

The measuring system that is active when the control is switched on must already have been referenced once before switching off.

Parameterization

Release: Restoration of the actual position

The enable to restore the actual position is performed by setting the encoder state to "The last buffered axis position before switching off will be restored, no automatic referencing":

MD34210 $MA_ENC_REFP_STATE[<encoder>] = **3**

Release: NC START for "MDI" and "Overstore" modes

The enable of NC START for execution of part programs or part program blocks in the "MDI" and "Overstore" modes with the state "Position restored" is performed via:

MD34110 $MA_REFP_CYCLE_NR **≠ -1** (axis sequence for channel-specific referencing)

MD20700 $MC_REFP_NC_START_LOCK = **2** (NC START lock without reference point)

NC/PLC interface signals

The restored actual position is not considered to be equivalent to an actual position after reference point approach. Therefore, the state "Position restored" and not "Referenced/ synchronized" is displayed for the measuring system of the machine axis.

Actual position restored:

- DB31, ... DBX60.4/.5 = **0** (referenced/synchronized 1/2)
- DB31, ... DBX71.4/.5 = **1** (**position restored**, encoder 1/2)

Measuring system referenced:

- DB31, ... DBX60.4/.5 = 0 → **1** (**referenced/synchronized** 1/2)
- DB31, ... DBX71.4/.5 = 1 → **0** (position restored, encoder 1/2)

Note

The monitoring of the traversing range limits (software limit switches, working area limitation, etc.) is already active in the "Position restored" state.

Supplementary conditions

Spindles

If the encoder limit frequency is exceeded, a spindle is reset to the "Not referenced/ synchronized" state:

- DB31, ... DBX60.4/.5 = 1 → **0** (referenced/synchronized 1/2)
- DB31, ... DBX71.4/.5 = 1 → **0** (position restored, encoder 1/2)

References

Description of Functions, Basic functions, Section "R1 Referencing" > "Referencing with incremental measurement systems (Page 1228)" > ""

16.11 Supplementary conditions

16.11.1 Large traverse range

Linear axes with a traversing range > 4096 encoder revolutions, rotatory absolute encoder EQN 1325 and a parameterized absolute encoder range of MD34220 $MA_ENC_ABS_TURNS_MODULO = 4096:

The maximum possible travel range corresponds to that of incremental encoders.

Endlessly turning rotary axes with absolute encoders:

- Any number of integer transmission ratios are permitted.
- We recommend that you parameterize endlessly turning rotary axes with absolute encoders as modulo rotary axes (traversing range 0...360 degrees):
  MD30310 $MA_ROT_IS_MODULO = 1
  Otherwise, the rotary axis may require a very large traversing path to reach absolute zero when the measuring system is activated.

Machine axes with absolute encoders:

In order that the controller correctly determines the current actual position after the restart of the measuring system, the machine axis may only be moved less than half the absolute encoder range when the measuring system is switched off:

MD34220 $MA_ENC_ABS_TURNS_MODULO

Notes on uniqueness of encoder positions

Note

Linear absolute encoders

The absolute value of linear position encoders, e.g. Heidenhain LC181, is always unique for the scale lengths available.

Rotary absolute encoders

The absolute value of rotary absolute encoders is only unique within the range of the specific maximum encoder revolutions.

For example, the EQN 1325 rotary absolute encoder by Heidenhain supplies a unique absolute value in the range of 0 to 4,096 encoder revolutions.

Depending on how the encoder is connected that will result in:

- Rotary axis with encoder on load: 4096 load revolutions
- Rotary axis with encoder on motor: 4096 motor revolutions
- Linear axis with encoder on motor: 4096 motor revolutions

Example:
An EQN 1325 rotary absolute encoder is mounted on the motor of a linear axis. For an effective leadscrew pitch of 10 mm this will result in a unique absolute value within the travel range -20.48 to +20.48 m.

16.12 Data lists

16.12.1 Machine data

16.12.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
11300	JOG_INC_MODE_LEVELTRIGGRD	INC/REF in jog/continuous mode

16.12.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20700	REFP_NC_START_LOCK	NC start disable without reference point

16.12.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30200	NUM_ENCS	Number of encoders
30240	ENC_TYP	Actual value encoder type
30242	ENC_IS_INDEPENDENT	Encoder is independent
30250	ACT_POS_ABS	Absolute encoder position at time of deactivation
30270	ENC_ABS_BUFFERING	Absolute encoder: Traversing range extension
30300	IS_ROT_AX	Rotary axis / spindle
30310	ROT_IS_MODULO	Modulo conversion for rotary axis / spindle
30330	MODULO_RANGE	Size of the modulo range
30340	MODULO_RANGE_START	Starting position of modulo range
30355	MISC_FUNCTION_MASK	Axis functions
31122	BERO_DELAY_TIME_PLUS	BERO delay time in plus direction
31123	BERO_DELAY_TIME_MINUS	BERO delay time in minus direction
34000	REFP_CAM_IS_ACTIVE	Axis with reference cam
34010	REFP_CAM_DIR_IS_MINUS	Reference point approach in minus direction
34020	REFP_VELO_SEARCH_CAM	Reference point 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 on reference cam
34060	REFP_MAX_MARKER_DIST	Maximum distance to reference mark; maximum distance to two reference marks with dis-tance-coded scales
34070	REFP_VELO_POS	Reference point positioning velocity

Number	Identifier: $MA_	Description
34080	REFP_MOVE_DIST	Reference point distance / destination point for distance-coded system
34090	REFP_MOVE_DIST_CORR	Reference point offset / absolute offset, distance-coded
34092	REFP_CAM_SHIFT	Electronic reference cam shift for incremental measuring 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 referencing
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	Range of the encoder serial number
34300	ENC_REFP_MARKER_DIST	Basic reference mark distance 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

16.12.2 Signals

16.12.2.1 Signals to BAG

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group RESET	DB11.DBX0.7	DB3000.DBX0.7
Machine function REF	DB11.DBX1.2	DB3000.DBX1.2

16.12.2.2 Signals from BAG

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Active machine function REF	DB11.DBX5.2	DB3100.DBX1.2

16.12.2.3 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate referencing	DB21, ...DBX1.0	DB320x.DBX1.0
OEM channel signal (HMI → PLC) REF	DB21, ...DBX28.7	-

16.12.2.4 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referencing active	DB21, ...DBX33.0	DB330x.DBX1.0
Reset	DB21, ...DBX35.7	DB330x.DBX3.7
All axes referenced	DB21, ...DBX36.2	DB330x.DBX4.2

16.12.2.5 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Follow-up mode (request)	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1 / position measuring system 2	DB31, ...DBX1.5-6	DB380x.DBX1.5-6
Reference point value 1 to 4	DB31, ...DBX2.4-7	DB380x.DBX2.4-7
Traversing keys minus/plus	DB31, ...DBX4.6-7	DB380x.DBX4.6-7
Deceleration of reference point approach	DB31, ...DBX12.7	DB380x.DBX1000.7

16.12.2.6 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced, synchronized 1/2	DB31, ...DBX60.4-5	DB390x.DBX0.4-5
Follow up active	DB31, ...DBX61.3	DB390x.DBX1.3
Traversing command minus/plus	DB31, ...DBX64.6-7	DB390x.DBX4.6-7
Restored 1/2	DB31, ...DBX71.4-5	DB390x.DBX11.4-5

S1: Spindles 17

17.1 Brief Description

The primary function of a spindle is to set a tool or workpiece in rotary motion in order to facilitate machining.

Depending on the type of machine, the spindle must support the following functions in order to achieve this:

- Input of a direction of rotation for the spindle (M3, M4)
- Input of a spindle speed (S, SVC)
- Spindle stop, without orientation (M5)
- Spindle stop with orientation / Spindle positioning (SPOS, M19 and SPOSA)
- Gear change (M40 to M45)
- Spindleaxis functionality (spindle becomes rotary axis and vice versa)
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck （G331, G332)
- Tapping with compensating chuck （G63)
- Revolutional feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- Programmable spindle speed limits (G25, G26, LIMS=)
- Position encoder assembly on the spindle or on the spindle motor
- Spindle monitoring for min. and max. speed as well as max. encoder limit frequency and end point monitoring of spindle
- Switching the position control (SPCON, SPCOF, M70) on/off
- Programming of spindle functions:
  - From the part program
  - Via synchronized actions
  - Via PLC with FC18 or via special spindle interfaces for simple spindle activation

17.2 Modes

17.2.1 Overview

A spindle is an endlessly rotating axis, which can be operated in either the closed-loop position control or closed-loop speed control modes.

Operating modes

- Control operation (closed-loop speed control)
- Oscillating operation (closed-loop speed control)
- Positioning operation (closed-loop position control)
- Synchronous mode
  References:
  Function Manual Extension Functions, Synchronous Spindle (S3)
- Tapping without compensating chuck (G331, G332)
  References:
  Programming Manual, Fundamentals, Chapter: "Position commands" > "Tapping" > "Tapping without compensating check (G331 / G332)"
- Axis mode (closed-loop position control)

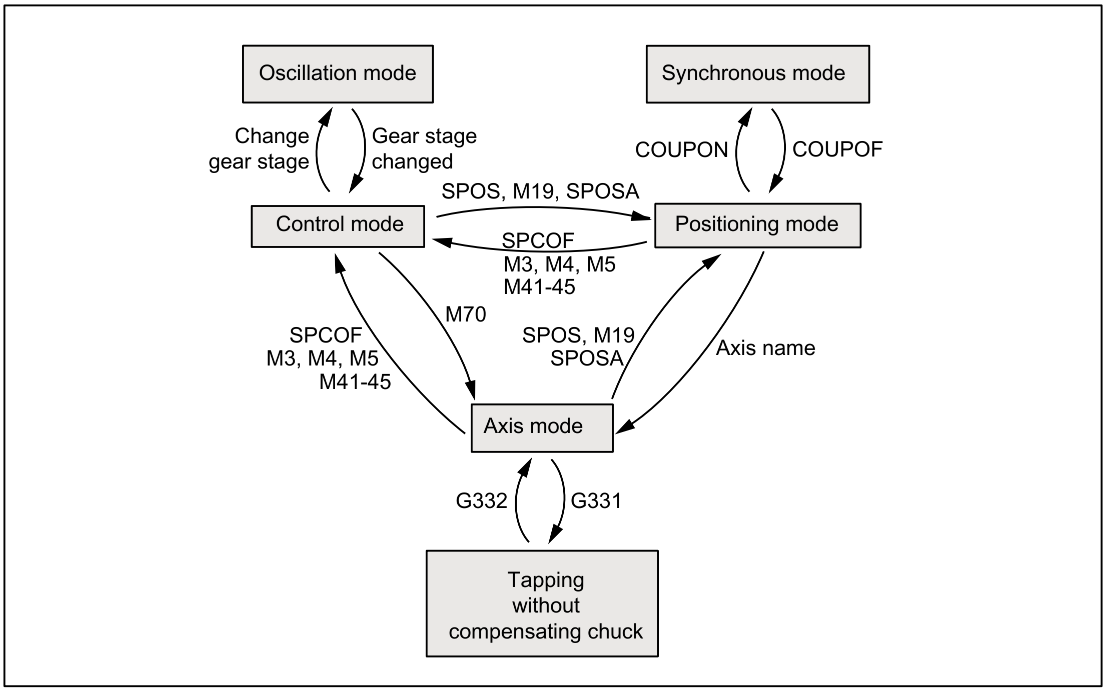
Note

Switchover, axis/spindle mode

A spindle can be switched over between the axis and spindle mode if the same motor is used for axis and spindle operation.

17.2.2 Mode change

Switching between spindle and axis operation can be done as follows:



- Control mode → Oscillation mode
  The spindle changes to oscillation mode if a new gear stage has been specified using automatic gear step selection (M40) in conjunction with a new S value or by M41 to M45. The spindle only changes to oscillation mode if the new gear step is not equal to the current actual gear step.
- Oscillation mode → Control mode
  When the new gear is engaged, the interface signal:
  DB31, ... DBX84.6 (Oscillation mode)
  is reset and the interface signal:
  DB31, ... DBX16.3 (Gear changed)
  is used to go to control mode.
  The last programmed spindle speed (S value) is reactivated.
- Control mode → Positioning mode
  To stop the spindle from rotation (M3 or M4) with orientation or to reorient it from standstill (M5), SPOS, M19 or SPOSA are used to switch to positioning mode.
- Positioning mode → Control mode
  M3, M4 or M5 are used to change to control mode if the orientation of the spindle is to be terminated. The last programmed spindle speed (S value) is reactivated.
- Positioning mode → Oscillation mode
  If the orientation of the spindle is to be terminated, M41 to M45 can be used to change to oscillation mode. When the gear change is complete, the last programmed spindle speed (S value) and M5 (control mode) are reactivated.

- Positioning mode → Axis mode
  If a spindle was stopped with orientation, the assigned axis name is used to program a change to axis mode. The gear step is retained.
- Control mode → Axis mode
  Switching from control mode to axis mode can be also achieved by the programming of M70. In this case, a rotating spindle is decelerated in the same way as for M5, position control activated and the zero parameter set selected.
- Axis mode → Control mode
  To terminate axis mode, M3, M4 or M5 can be used to change to control mode. The last programmed spindle speed (S value) is reactivated.
- Axis mode → Oscillation mode
  To terminate axis mode, M41 to M45 can be used to change to oscillation mode (only if the programmed gear step is not the same as the actual gear step). When the gear change is complete, the last programmed spindle speed (S value) and M5 (control mode) are reactivated.

17.2.3 Control mode

When open-loop control mode?

The spindle is in open-loop control mode with the following functions:

- Constant spindle speed:
  - S... M3/M4/M5 and G93, G94, G95, G97, G971
  - S... M3/M4/M5 and G33, G34, G35
  - S... M3/M4/M5 and G63
- Constant cutting speed:
  - G96/G961 S... M3/M4/M5

The spindle need not be synchronized.

Requirements

A spindle position actual value encoder is absolutely essential for M3/M4/M5 in connection with:

- Revolutional feedrate (G95)
- Constant cutting speed (G96, G961, G97, G971)
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Activate position control (SPCON, M70)

A spindle position actual value encoder is **not** required for M3/M4/M5 in connection with:

- Inverse-time feedrate coding (G93)
- Feedrate in mm/min or inch/min (G94)
- Tapping with compensating chuck (G63)

Speed control mode

Speed control mode is particularly suitable if a constant spindle speed is required, but the position of the spindle is not important (e.g. constant milling speed for even appearance of the workpiece surface).

- Speed control mode is activated in the part program with `M3`, `M4`, `M5` or with `SPCOF`.
- The following NC/PLC interface signal is set:
  DB31, ... DBX84.7 (control mode)
- NC/PLC IS:
  DB31, ... DBX61.5 (position controller active)
  is reset if position control is not used.
- Acceleration in speed control mode is defined independently of the gear stage in the machine data:
  MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL
  The value should reflect the physical circumstances, if possible.

Position control mode

Position control is particularly suitable if the position of the spindle needs to be tracked over a longer period or if synchronous spindle setpoint value linkage is to be activated.

- Position control mode is switched on in the part program with: `SPCON(<spindle number>)`
- The following NC/PLC interface signal is set:
  DB31, ... DBX61.5 (position controller active)
- Acceleration in position control mode is defined independent of the gear stage in the machine data:
  MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Independent spindle reset

The spindle response after a reset or the end of the program (M2, M30) is set with the machine data:

MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (individual spindle reset)

Value	Meaning
0	When the spindle is reset or at the end of the program, the spindle immediately decelerates to a stop at the active acceleration rate. The last programmed spindle speed and direction of rotation are deleted.
1	Upon reset or at the end of the program, the last programmed spindle speed (S-value) and the last programmed direction of spindle rotation (M3, M4, M5) are retained. The spindle is not braked.

If prior to reset or end of program the constant cutting speed (G96, G961) is active, the current spindle speed (in relation to 100% spindle override) is internally accepted as the spindle speed last programmed.

The spindle can only be stopped with the NC/PLC interface signal:

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

The direction of rotation is deleted in the event of all alarms generating a spindle quick stop. The last programmed spindle speed (S value) is retained. Once the source of the alarm has been eliminated, the spindle must be restarted.

Spindle actual speed display and spindle behavior with G96, G961

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

The speed at which the spindle is deemed to be "stationary" is set with the machine data:

MD36060 $MA_STANDSTILL_VELO_TOL

The value should be measured in such a way that the following NC/PLC interface signal is reliably present at a standstill:

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

If DB31,... DBX61.4 (axis/spindle stationary) is signaled and there is no closed-loop position control active for the spindle, an actual speed of zero is displayed at the operator interface, and zero is read with the system variable $AA_S[<n>].

Spindle response at constant cutting speed G96, G961

- At the start of machining (transition from G0 to Gx) and after NC stop, G60 (exact stop, modal) and G9 (exact stop, non-modal) the system waits until the actual speed has reached the speed setpoint tolerance range before starting the path.
  DB31, ... DBX83.5 (nact = nset)
- The NC/PLC IS:
  DB31, ... DBX83.5 (nact = nset)
  and
  DB31, ... DBX83.1 (setpoint speed limited) are also set to defined values
  even if significant speed changes are specified (transverse axis travels towards position 0).

- If the speed drops below the minimum speed
  or when NC/PLC IS:
  DB31, ... DBX61.4 (axis/spindle stationary)
  is detected, NC/PLC IS:
  DB31, ... DBX83.5 (nact = nset)
  is reset (e.g. for an emergency machine strategy).
- A path operation which has started (G64, rounding), is not interrupted.

In addition, the spindle response is influenced by the following machine data:

MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START (feed enable with spindle in setpoint range).

Spindle behavior at the end of gear stage change

- NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  informs the NC that the new gear stage
  (NC/PLC IS DB31, ... DBX16.0-16.2 (actual gear stage A to C))
  applies and oscillation mode is terminated.
  In this case, it does not matter whether NC/PLC IS:
  DB31, ... DBX18.5 (oscillation mode)
  is still set.
  The actual gear stage should correspond to the set gear stage.
  The actual gear stage signaled is relevant for selection of the parameter set.
- Once the gear stage change (GSW) has been acknowledged via the PLC (DB31, ... DBX16.3), the spindle is in speed control mode (DB31, ... DBX84.7 = 1).
  If a direction of rotation (M3, M4, M5 or FC18: "Start spindle rotation") or a spindle speed (S value) was programmed before the gear stage change, then the last speed and direction of rotation will be reactivated after the gear stage change.

17.2.4 Oscillation mode

Oscillation mode is activated for the spindle during the gear step change.

The mode of operation is described in detail in the topic "Gear step change with oscillation mode (Page 1334)".

17.2.5 Positioning mode

17.2.5.1 General functionality

When is positioning mode used?

The spindle positioning mode stops the spindle at the defined position and activates the position control, which remains active until it is de-activated.

For the following functions the spindle is in positioning mode:

- SPOS[<n>]=...
- SPOS[<n>]=ACP(...)
- SPOS[<n>]=ACN(...)
- SPOS[<n>]=AC(...)
- SPOS[<n>]=IC(...)
- SPOS[<n>]=DC(...)
- SPOSA[<n>]=ACP(...)
- SPOSA[<n>]=ACN(...)
- SPOSA[<n>]=AC(...)
- SPOSA[<n>]=IC(...)
- SPOSA[<n>]=DC(...) equal to SPOSA[<n>]=...
- M19 or M[<n>]=19

The address extension [<n>] with <n> = spindle number may not apply for the main spindle.

SPOS[<n>]=AC(...)

Spindle positioning to an absolute position (0 to 359.999 degrees). The positioning direction is determined either by the current direction of spindle rotation (spindle rotating) or the distance-to-go.

SPOS[<n>]=IC(...)

Spindle positioning to an incremental position (+/- 999999.99 degrees) in relation to the last programmed position. The positioning direction is defined by the sign of the path to be traversed.

SPOS[<n>]=DC(...)

Spindle positioning across the shortest path to an absolute position (0 to 359.999 degrees). The positioning direction is determined either by the current direction of spindle rotation (spindle rotating) or automatically by the control (spindle stationary).

SPOS[<n>]=...

Same functional sequence as SPOS [<n>]=DC(...).

SPOS[<n>]=ACP(...)

Approaches the position from the positive direction.

When positioning from a negative direction of rotation, the speed is decelerated to zero and accelerated in the opposite direction to execute the positive approach.

SPOS[<n>]=ACN(...)

Approaches the position from the negative direction.

When positioning from a positive direction of rotation, the speed is decelerated to zero and accelerated in the opposite direction to execute the negative approach.

M19 (DIN 66025)

M19 can be used to position the spindle. The position and the position approach mode are read here from the following setting data:

SD43240 $SA_M19_SPOS[<n>] (spindle position for spindle positioning with M19)

SD43250 $SA_M19_SPOSMODE[<n>] (spindle position for spindle positioning with M19)

The positioning options of M19 are identical to those of:

SPOS = <approach mode> <position/path>

M19 is output as an auxiliary function to the NC/PLC interface as an alternative to M3, M4, M5, and M70. The M19 block remains active in the interpolator for the duration of positioning (like SPOS).

Part programs using M19 as a macro (e.g. `DEFINE M19 AS SPOS = 0`) or as a subprogram, continue to remain executable. For the sake of compatibility with previous controls, the internal processing of M19 (NC positions the spindle) can be disabled as shown in the following example:

MD22000 $MC_AUXFU_ASSIGN_GROUP[0] = 4	; Auxiliary function group: 4
MD22010 $MC_AUXFU_ASSIGN_TYPE[0] = "M"	; Auxiliary function type: "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[0] = 0	; Auxiliary functions Extension: 0
MD22030 $MC_AUXFU_ASSIGN_VALUE[0] = 19	; Auxiliary function value: 19

Implicitly generated auxiliary function M19

To achieve uniformity in terms of how M19 and SPOS or SPOSA behave at the NC/PLC interface, auxiliary function M19 can be output to the NC/PLC interface in the event of SPOS and SPOSA.

The two following options are available for activating this function:

- Channel-specific activation for all the spindles in the channel via the machine data: MD20850 $MC_SPOS_TO_VDI (Output of M19 to the PLC with SPOS/SPOSA)

Bit	Value	Meaning
0	0	If bit 19 is also set to "0" in the MD35035 $MA_SPIND_FUNCTION_MASK, no auxiliary function M19 is generated in SPOS and SPOSA. This therefore eliminates the acknowledgment time for the auxiliary function.
	1	The auxiliary function M19 is generated and output to the PLC during the programming of SPOS and SPOSA in the part program. The address extension corresponds to the spindle number.

- Spindle-specific and cross-channel activation via the machine data:
  MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)

Bit	Value	Meaning
19	0	If bit 0 is also set to "0" in the MD20850 $MC_SPOS_TO_VDI, no auxiliary function M19 is generated in SPOS and SPOSA. This therefore eliminates the acknowledgment time for the auxiliary function.
	1	The implicit auxiliary function M19 is generated and output to the PLC during the programming of SPOS and SPOSA. The address extension corresponds to the spindle number.

Note

Activation via MD35035 should be preferred when using a spindle in multiple channels (axis/spindle exchange).

The auxiliary function M19 is implicitly generated if either of the MD configurations = 1.

After activation, the minimum duration of an SPOS/SPOSA block is increased to the time for output and acknowledgment of the auxiliary functions by the PLC.

The properties of the implicitly generated auxiliary function output M19 are "Quick" and "Output during motion". These properties are fixed settings and are independent of the M19 configuration in the auxiliary function-specific machine data (MD..._$M..._**AUXFU**_...).

There is **no** auxiliary function M19 implicitly generated in the case of spindle positioning commands via FC 18.

End of positioning

The positioning can be programmed with:

`FINEA[S<n>]:`	End of motion on reaching "Exact stop fine" (DB31, ... DBX60.7)
`COARSEA[S<n>]:`	End of motion on reaching "Exact stop coarse" (DB31, ... DBX60.6)
`IPOENDA[S<n>]:`	End of motion on reaching "IPO stop"

In addition, an end-of-motion criterion for block changes can be set in the braking ramp (100-0%) with IPOBRKA for single-axis interpolation.

References:
Function Manual, Extended Functions; Positioning Axes (P2)

Block change

The program advances to the next block if the end-of-motion criteria for all spindles or axes programmed in the current block, plus the block change criterion for path interpolation, are fulfilled. This applies to both part-program and technology-cycle blocks.

SPOS, M19 and SPOSA have the same functionality but differ in their block change behavior:

- SPOS and M19
  The block change is carried out if all functions programmed in the block have reached their end-of-block criterion (e.g. all auxiliary functions acknowledged by the PLC, all axes have reached their end points) and the spindle has completed its positioning motion.
- SPOSA
  The program moves to the next block if all the functions (except for spindle) programmed in the current block have reached their end-of-block criterion. If `SPOSA` is the only entry in the block, block change is performed immediately. The spindle positioning operation may be programmed over several blocks (see WAITS).

Coordination

A coordination of the sequence of motions can be achieved with:

- PLC
- MD configuration
- Programming in the part program

PLC

If the NC/PLC interface signal:
DB31, ... DBX83.5 (spindle in the setpoint range)
is not available, then the channel-specific NC/PLC interface signal:
DB21, ... DBX 6.1 (read-in inhibit)
can be set in order to wait for a spindle to reach a certain position.

MD configuration

Setting:
MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 1
is used to perform path interpolation taking the tolerance:
MD35150 $MA_SPIND_DES_VELO_TOL
into account only if the spindle has rotated up to the preselected speed.

The setting
MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 2
is used to stop traveling path axes before the start of machining at the last G0.

Machining continues:

- With the next traversing command.
- If the spindle speed is reached.
- When MD35510 $MA_SPIND_STOPPED_AT_IPO_START = 1
  (path feed enable, if spindle stationary).

Programming in the part program

Coordination actions in the part program have the following advantages:

- The part program author can decide at what point in the program the spindle needs to be up to speed, e.g. in order to start machining a workpiece.
- This avoids unnecessary delays.

Coordination in the part program involves programming the `WAITS` command:

`WAITS`:	for main spindle (master spindle)
`WAITS[<n>]`:	for spindles with number <n>
`WAITS[<n>,<m>,...]`:	for several spindles up to the maximum number of spindles

⚠ CAUTION

Coordination error

The part program author must ensure that one of the following maintenance conditions occurs for WAITS.

- Position reached
- Spindle stationary
- Spindle up to programmed speed

In cases where one spindle is used in several channels, the part program author must ensure that WAITS starts at the earliest in the phase in which the spindle from another channel has already started to accelerate or decelerate towards the required new speed or direction of rotation.

The control waits before executing subsequent blocks until:

- Position(s) programmed with `SPOSA` are reached.
- Spindle standstill is reached with M5:
  DB31, ..., DBX 61.4 (spindle stationary)
  taking into account the tolerance:
  MD36060 $MA_STANDSTILL_VELO_TOL
  WAITS is terminated and the next block loaded when the first occurrence of the signal is detected.
- In M3/M4 (speed control mode), the speed in the setpoint range is:
  DB31, ..., DBX83.5 (spindle in setpoint range)
  taking into account the tolerance:
  MD35150 $MA_SPIND_DES_VELO_TOL
  WAITS is terminated and the next block loaded when the first occurrence of the signal is detected.
  This WAITS function applies in the programmed channel.
  WAITS can be used to wait for all spindles known to this channel, although spindles may also have been started in other channels.

Special cases

- **Tolerance for spindle speed:**
  If the machine data setting is:
  MD35150 $MA_SPIND_DES_VELO_TOL = 0
  the NC/PLC interface signal:
  DB31, ... DBX83.5 (spindle in setpoint range)
  is always set to 1.
  WAITS is terminated as soon as the spindle has reached the setpoint-side target after a change in speed or direction (M3/M4).
- **Missing enable signals:**
  If the WAITS function waits for the "Spindle in setpoint range" signal in speed control mode and the spindle stops or fails to rotate because an enable signal (axial feed enable, controller, pulse enable, etc.) is missing, the block is not terminated until the "Spindle in setpoint range" signal is active, once enable signals are being received again.
- **Response to NC and mode-group stop:**
  If an NC or mode-group stop is triggered during WAITS, the wait operation is resumed after the NC start with all the above conditions.

Note

In particular when using spindles across different channels, care should be taken when programming not to start WAITS too early in one channel, i.e. at a time when the spindle in the other channel is still rotating at its "old" speed.
In such cases, the "Spindle in setpoint range" signal is activated and WAITS is stopped too soon.
To prevent this happening, it is strongly recommended to set a WAITM before WAITS.

Feedrate

The positioning speed is configured in the machine data:

MD35300 $MA_SPIND_POSCTRL_VELO (position control switching speed)

The configured positioning speed can be modified by programming or by synchronized actions:
`FA[S<n>]=<value>`

where:	`<n>`:	Spindle number
	`<value>`:	Positioning speed in degrees/min

With `FA[S<n>]=0`, the configured speed takes effect.

Acceleration

The accelerations are configured in the machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL (acceleration in position control mode)

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL (acceleration in the speed control mode)

The configured dynamic response during positioning can be modified by programming or by synchronized actions:
`ACC[S<n>]=<value>`

where: `<n>`:	Spindle number
`<value>`:	Acceleration as a percentage of the configured acceleration

With `ACC[S<n>]=0`, the configured acceleration takes effect.

Aborting the positioning process

The positioning action is aborted:

- By the NC/PLC interface signal:
  DB31, ... DBX2.2 (delete distance-to-go / spindle reset).
- With every reset (e.g. operator panel front reset).
- Through NC stop.

The abort response in independent of the machine data:

MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (individual spindle reset)

Special features

The spindle override switch is valid.

17.2.5.2 Positioning from rotation

Initial situation

The spindle can be in speed control mode or in position control mode when positioning starts (`SPOS`, `M19` or `SPOSA` command in the program).

One must distinguish between the following cases:

Case 1:	Spindle in speed control mode, encoder limit frequency exceeded
Case 2:	Spindle in speed control mode, encoder limit frequency not exceeded
Case 3:	Spindle in position control mode
Case 4:	Spindle speed ‹ Position-control activation speed

Procedure

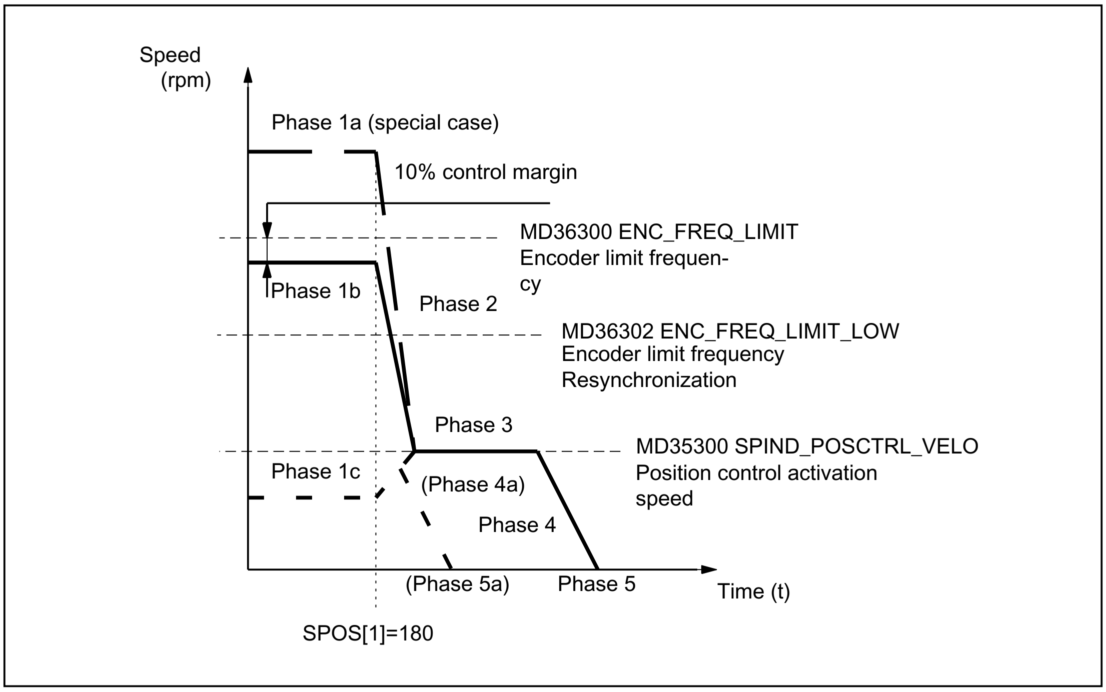

Speed (rpm)
Phase 1a (special case)
10% control margin
MD36300 ENC_FREQ_LIMIT Encoder limit frequen-cy
Phase 1b
Phase 2
MD36302 ENC_FREQ_LIMIT_LOW Encoder limit frequency Resynchronization
Phase 3
MD35300 SPIND_POSCTRL_VELO Position control activation speed
Phase 1c
(Phase 4a)
Phase 4
(Phase 5a)
Phase 5
Time (t)
SPOS[1]=180

Figure 17-1 Positioning from rotation

Note

The speed arising from the configuration of the encoder limit frequency for the resynchronization of the encoder (MD36302 $MA_ENC_FREQ_LOW) must be greater than the position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO).

Phase 1

Positioning from phase 1a:

The spindle is rotating at a higher speed than the encoder limit frequency. The spindle is not synchronized.

Positioning from phase 1b:

The spindle is rotating at a lower speed than the encoder limit frequency. The spindle is synchronized.

Note

If the position control is active, the speed can only amount to 90% of the maximum speed of the spindle or the encoder limit frequency (10% control reserve required).

Positioning from phase 1c:

The spindle rotates at the programmed spindle speed whereby the speed is lower than the configured position-control activation speed:

MD35300 $MA_SPIND_POSCTRL_VELO

The spindle is synchronized.

Phase 2

Spindle speed > Position-control activation speed

When the `SPOS`, `M19` or `SPOSA` command is activated, the spindle begins to slow down to the position-control activation speed with the configured acceleration:

MD35200 $MA_GEAR_STEP_SPEEDCTL_ACCEL

The spindle is synchronized once the encoder limit frequency threshold is crossed.

Spindle speed < Position-control activation speed

`SPOS`, `M19` or `SPOSA` are programmed to switch the spindle to position control mode (if it is not already in that mode).

The configured acceleration in position control mode is activated:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

The travel path to the target point is calculated.

The spindle travels to the programmed end point optimally in terms of time. This means that the end point is approached at the highest possible speed (maximum MD35300 $MA_SPIND_POSCTRL_VELO). Depending on the appropriate secondary conditions, phases 2 - 3 - 4 - 5 or 4a - 5a are executed.

Phase 3

Spindle speed > Position-control activation speed

When the configured position-control activation speed
(MD35300 $MA_SPIND_POSCTRL_VELO) is reached:

- Position control is activated (if not already active).
- The distancetogo (to the target point) is calculated.
- There is a switch to the configured configured acceleration in position control mode (or this acceleration is retained):
  MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Spindle speed < Position-control activation speed

The spindle was accelerated up to the configured position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO) to reach the end point. This is not exceeded.

The braking start point calculation detects when the programmed spindle position can be approached accurately at the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

Phase 4

Spindle speed > Position-control activation speed

The spindle brakes from the calculated "braking point" with machine data:
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL
to the target position.

Spindle speed < Position-control activation speed

At the time identified by the braking start point calculation in phase 3, the spindle brakes to a standstill with the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

Phase 4a:

When the `SPOS` command is activated the proximity of the end point is such that the spindle can no longer be accelerated to the configured position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO).

The spindle brakes to a standstill with the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

Phase 5

Spindle speed > Position-control activation speed

Position control remains active and holds the spindle in the programmed position.

Note

The maximum encoder limit frequency of the spindle position actual-value encoder is monitored by the control (it may be exceeded); in position control mode, the setpoint speed is reduced to 90% of the measuring system limit speed.

The following NC/PLC interface signal is set:

DB31, ... DBX83.1 (programmed speed too high)

If "MS limit frequency exceeded" is still pending following a reduction in the setpoint speed, an alarm is output.

Spindle speed < Position-control activation speed (Phase 5, 5a)

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

If the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the configured exact stop fine and coarse limits, the following NC/PLC interface signals are set:

DB31, ... DBX60.7 (Position reached with coarse exact stop)

DB31, ... DBX60.7 (Position reached with fine exact stop)

The exact stop limits are defined with the machine data:

MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)

MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)

Note

The positioning procedure is considered complete when the end-of-positioning criterion is reached and signaled.

The condition is "Exact stop fine". This applies to `SPOS`, `M19` or `SPOSA` from the part program, synchronized actions and spindle positioning by the PLC using FC 18.

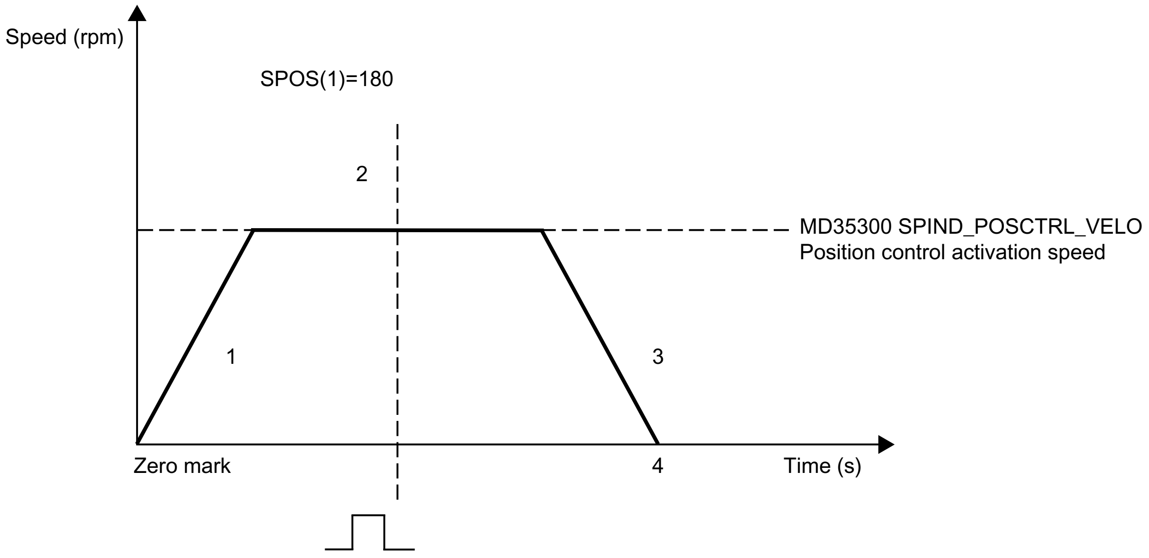
17.2.5.3 Positioning from standstill

Procedure

A distinction is made between two cases with regard to positioning from standstill:

- Case 1: The spindle is not synchronized.
  This is the case if the spindle is to be positioned after switching on the control and drive or after a gear step change (e.g. for a tool change).
  MD31040 $MA_ENC_IS_DIRECT = 0
- Case 2: The spindle is synchronized.
  This is the case if, after switching on the control and drive, the spindle is to be rotated through a minimum of one revolution with M3 or M4 and then stopped with M5 (synchronization with the zero mark) before the first positioning action.

Case 1: Spindle not synchronized



Case 1: Spindle synchronized

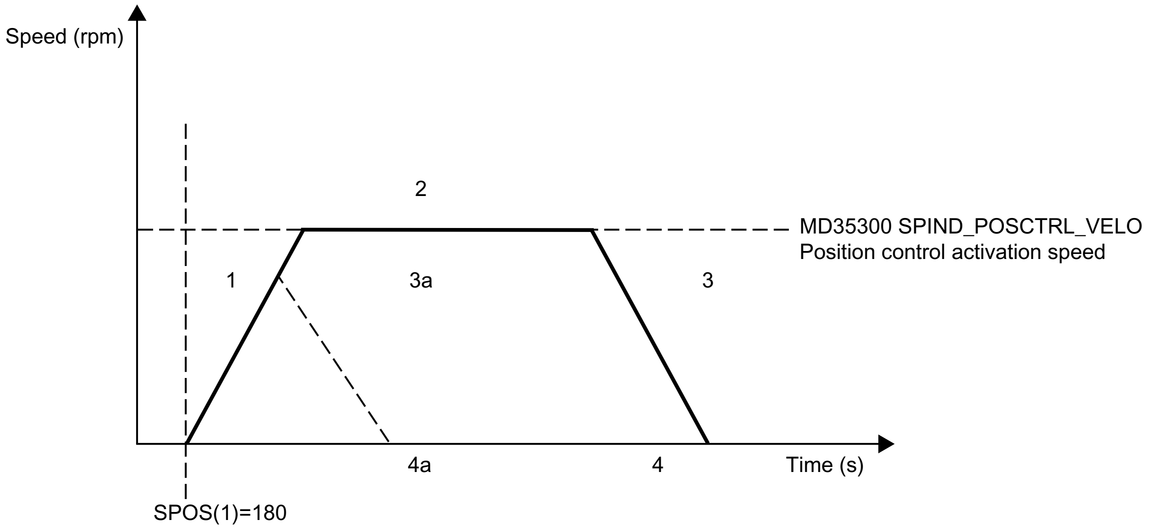


Figure 17-2 Positioning with stationary spindle

Phase 1

Case 1: Spindle not synchronized

With the programming of `SPOS`, `M19` or `SPOSA` the spindle accelerates with the acceleration from the machine data:

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL (Acceleration in the speed control mode)

This direction of rotation is defined by the machine data:

MD35350 $MA_SPIND_POSITIONING_DIR (Direction of rotation while positioning to standstill)

Exception:

If `ACN`, `ACP`, `IC` is used for positioning, the programmed direction of travel is activated.

The spindle is synchronized at the next zero mark of the spindle position actual-value encoder and switches to the position control mode.

Whether the zero mark is found in the traversed path (except for IC), is monitored:

MD34060 $MA_REFP_MAX_MARKER_DIST (maximum distance to the reference mark)

If the speed defined in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO (Positioning speed)
is reached before the spindle is synchronized, the spindle will continue to rotate at the positioning activation speed (it is not accelerated further).

Case 2: Spindle synchronized

`SPOS`, `M19` or `SPOSA` will switch the spindle to position control mode.

The acceleration from the following machine data is active:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL (acceleration in position control mode)

The direction of rotation is defined by the programmed motion (`ACP`, `ACN`, `IC`, `DC`) or via the pending distance-to-go.

The speed entered in
MD35300 $MA_SPIND_POSCTRL_VELO (position control activation speed)
is not exceeded in the machine data.

The travel path to the end position is calculated.

The spindle travels to the programmed end point optimally in terms of time. This means that the end point is approached at the highest possible speed (maximum MD35300 $MA_SPIND_POSCTRL_VELO). Depending on the appropriate secondary conditions, the phases 1 - 2 - 3 - 4 or 1- 3a - 4a are executed.

Phase 2

Case 1: Spindle not synchronized

When the spindle is synchronized, position control is activated.

The spindle rotates at the maximum speed stored in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO
until the braking start point calculation identifies the point at which the programmed spindle position can be approached accurately with the defined acceleration.

Case 2: Spindle synchronized

To reach the end point, the spindle is accelerated up to the speed defined in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO.

This is not exceeded.

The braking start point calculation identifies when the programmed spindle position can be approached accurately at the acceleration defined in machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL.

At the point, which is determined by the braking start point calculation in Phase 1, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Phase 3

At the point, which is determined by the braking start point calculation in Phase 2, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Phase 3a:

When the `SPOS` command is activated the proximity of the end point is such that the spindle can no longer be accelerated up to machine data:
MD35300 $MA_SPIND_POSCTRL_VELO.

The spindle is braked to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Phase 4, 4a

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop coarse)
and
DB31, ... DBX60.7 (Position reached with exact stop fine)
are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits.

This is defined in the machine data:

MD36010 $MA_STOP_LIMIT_FINE

MD36000 $MA_STOP_LIMIT_COARSE

Phase 3:

At the point, which is determined by the braking start point calculation in Phase 2, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_ POSCTRL_ACCEL

Phase 4:

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop coarse)
and
DB31, ... DBX60.7 (Position reached with exact stop fine)

are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits.

This is defined in the machine data:

MD36010 $MA_STOP_LIMIT_FINE

MD36000 $MA_STOP_LIMIT_COARSE

17.2.5.4 "Spindle in position" signal for tool change

Function

The motion sequence for a tool change, especially for milling machines, mainly comprises positioning the spindle and then the subsequent (for optimization runs, also at the same time) approach to the tool change position with the path axes. In this case, the mandatory requirement is that the spindle is reached before approaching the tool change position.

If the tool change cycle is interrupted by the machine operator (e.g. with an NC stop, NC stop axes plus spindles, mode group stop, etc.), then it must be completely ruled out that the spindle moves into the tool changer at an incorrect position.

This is the reason that for spindle positioning, when the last programmed spindle position is reached with "Exact stop fine" the following NC/PLC interface signal is output to check the position:

DB31, ... DBX85.5 (spindle in position)

Note

The signal is only output for the "Spindle positioning" function.

This includes:

- SPOS, SPOSA and M19 in the part program
- SPOS and M19 in synchronized actions
- Spindle positioning, using FC18
- Spindle positioning via PLC interface (DB31, ... DBX30.4)

Setting the signal

Requirements for output of signal DB31, ... DBX85.5 (spindle in position) are as follows:

- The referenced state of the spindle:
  DB31, ... DBX60.4/5 (referenced/synchronized 1/2) = 1

  Note

  When positioning the spindle, the zero mark is automatically searched for. This is the reason that for an error-free sequence, the referenced signal is always available at the end of positioning movement.

- "Exact stop fine" must have been reached:
  DB31, ... DBX60.7 (exact stop fine) = 1
  Additionally, the last programmed spindle position must have been reached on the setpoint side.

Deleting the signal

When signal DB31, ... DBX60.7 is withdrawn (exact stop fine), then signal DB31, ... DBX85.5 (spindle in position) is also always reset.

Additional properties

- If the spindle is already at the programmed position after a positioning, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) remains set.
- If, after a positioning ("Spindle in position" signal was output) the spindle is traversed, e.g. in the JOG mode, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) is deleted.
  If the spindle is returned to its original position in this mode, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) is set again. The last position selection is kept.

17.2.6 Axis mode

17.2.6.1 General functionality

Functionality

If for certain machining tasks, e.g. on lathes with end-face machining, it is not sufficient to traverse the spindle exclusively under speed control via `M3`, `M4`, `M5` or to position with `SPOS`, `M19` or `SPOSA`, the spindle can be switched to position-controlled axis mode and traversed as a rotary axis.

Examples of rotary axis functions:

- Programming with axis name
- Zero offsets (`G54, G55`, `TRANS`, etc.)

- `G90`, `G91`, `IC`, `AC`, `DC`, `ACP`, `ACN`
- Kinematic transformations (e.g. TRANSMIT)
- Path interpolation
- Traversing as positioning axis

References:
Function Manual, Extended Functions; Section "Rotary axes (R2)"

Requirements

- The same spindle motor is used for spindle mode and axis mode.
- The same position measurement system or separate position measurement systems can be used for spindle mode and axis mode.
- An actual position value encoder is a mandatory requirement for axis mode.
- The spindle must be referenced for use of the axis mode, e.g. referenced with `G74`.
  Example:

Program code	Comment
`M70`	`; Switch spindle over to axis mode`
`G74 C1=0 Z100`	`; Reference axis`
`G0 C180 X50`	`; Traverse axis position-controlled`

Configurable M function

The M function used to switch the spindle to axis mode can be configured channel-specifically via the following machine data:

MD20094 $MC_SPIND_RIGID_TAPPING_M_NR

Note

The control detects the transition to axis mode automatically from the program sequence (see "Implicit transition to axis mode (Page 1299)"). The explicit programming of the configured M function for switching the spindle to axis mode in the part program is therefore not necessary. However, the M function can continue to be programmed, e.g. to increase the readability of the part program.

Special features

- The feedrate override switch is valid.
- The NC/PLC interface signal does not terminate the axis mode per default:
  DB21, ... DBX7.7 (reset).
- The NC/PLC interface signals:
  DB31, ... DBB16 to DBB19 and DBB82 to DBB91
  are not important if:
  DB31, ... DBX60.0 == 0 (spindle / rotary axis)

- Axis mode can be activated in all gear stages.
  If the actual position value encoder is installed on the motor (indirect measuring system), the positioning and contouring accuracy can vary for the different gear stages.
- The gear stage cannot be changed when the axis mode is active. For this purpose, the spindle must first be switched to control mode with `M41` ... `M45` or `M5`, `SPCOF`.
- In axis mode, the first parameter set is effective (machine data index = zero).
  References
  Function Manual, Basic Functions; Chapter "Velocities, setpoint / actual value systems, closed-loop control (G2)" > "Closed-loop control" > "Parameter sets of the position controller"

Dynamic response

The dynamic limits of the axis apply in axis mode. For example:

- MD32000 $MA_MAX_AX_VELO[<axis>] (maximum axis velocity)
- MD32300 $MA_MAX_AX_ACCEL[<axis>] (maximum axis acceleration)
- MD32431 $MA_MAX_AX_JERK[<axis>] (maximum axial jerk for path motion)

Feedforward control

The feedforward control mode active for the axis is retained.

A detailed description of the "Dynamic feedforward control" function can be found in:

References
Function Manual, Extended Functions; Section "Compensations (K3)" > "Dynamic feedforward control (following error compensation)"

Example: Resolution switchover for analog actuator

Switching to axis mode

Programming	**Comment**
`SPOS=...`	

`M5`	`; Control enable off (from PLC)`
	`→ is output on PLC`

`M70`	`; Switch actuator (from PLC on account of M70)`
	`Control enable on (from PLC)`

`C=...`	`; NC traverses with axis parameter set`

Switching to spindle mode

Programming	**Comment**
`C=...`	
`M71`	`; → is output to PLC`
	`Closed-loop controller enable off (from PLC)`
	`Switch actuator (from PLC)`
	`Switched to spindle parameter set (1-5) internally in the NC, controller enable on (from PLC)`

Programming	Comment
`M3/4/5 or SPOS=...`	`; NC traverses with spindle parameter set`

Change to spindle mode

The parameter set 1 ... 5 is selected for the active gear stage.

The feedforward control is activated, except for tapping with compensating chuck, if the following applies:

MD32620 $MA_FFW_MODE (feedforward control mode) ≠ 0

Parameter set	Axis mode	Spindle mode
1	Valid	-
2	-	Valid
3	-	Valid
4	-	Valid
5	-	Valid
6	-	Valid
Spindle mode: Parameter set according to the gear stage		

17.2.6.2 Implicit transition to axis mode

Function

The control detects the transition to axis mode automatically from the program sequence and generates the requisite M70 sequence in the control. The situation will dictate which steps are performed. At most, these will include:

1. Stopping the spindle
2. Switching on of the position control, treatment of feedforward control and parameter block changeover
3. Position synchronization of the block preparation (internal preprocessing stop, if necessary)

This function is always active. Explicit programming of M70 in the part program is therefore basically not necessary.

Sequence

Sequence of the implicit transition to axis mode (M70 is not programmed in the part program):

- Transition from speed control mode (M3, M4, M5, SPCOF, ...) to axis mode:
  The transition is detected internally by the control, and an intermediate block is inserted in front of the block which requests the axis mode. The block created contains the M70 functionality. The execution duration for this block is more or less the same as the time required to execute a programmed M70 block. Differences may arise in the event of short switchovers when the spindle is stationary (no braking time) if the implicit generation and output of the auxiliary function M70 to the PLC is dispensed with (see MD35035).
- Transition from positioning mode (M19, SPOS, SPOSA) to axis mode:
  The transition is executed immediately and without the generation of an intermediate block. Configured accordingly (see MD35035), the auxiliary function M70, which is generated implicitly, is output to the PLC when the block in which the spindle has its axis mode is loaded.

Output of auxiliary functions to PLC

The implicit transition to axis mode can be notified to the PLC in the form of an auxiliary function output.

Activation/deactivation

The activation/deactivation of this functionality is performed via the machine data:

MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)

Bit	Value	Meaning
20	0	No auxiliary function output to the PLC in the case of M70 functionality which is generated in the control.
	1	In the case of M70 functionality which is generated inside the control, the auxiliary function M70 is generated and output to the PLC. The address extension corresponds to the spindle number.

Note

An auxiliary function M70 which is programmed in the part program is always output to the PLC.

Properties

The properties of the implicitly generated auxiliary function output M70 are "Quick" and "Output during motion". These properties are fixed settings and are independent of the M70 configuration in the auxiliary-function-specific machine data (MD..._$M..._**AUXFU**_...).

M70 is only generated once during transition to axis mode. No further M70 auxiliary functions are generated and output in adjacent blocks in which the spindle is operated as an axis. M70 is not implicitly generated and output again until axis mode is exited via, for example, SPOS, M3, M4, M5, SPCOF, etc. and following a renewed transition to axis mode.

Supplementary conditions

Synchronized actions

When the spindle is programmed as an axis in synchronized actions, it is essential to continue making provisions in the application to ensure there are criteria for the transition to axis mode.

If the spindle is in speed control mode, the instruction `M70` or `SPOS` must be programmed prior to programming as an axis. Otherwise alarm signals occur during axis programming.

FC 18

As with synchronized actions, transition to axis mode must also be performed on the application side in FC 18, e.g. through a preparatory positioning instruction. Otherwise, the FC 18 call is acknowledged with an error bit in the FC 18 status word.

No auxiliary function M70 is implicitly generated in the event of transition to axis mode through programming via FC 18.

Examples

Example 1:

Part program: Transition from rotating spindle to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

Program code	Comment
`N05 M3 S1000`	
`N10 ...`	
`N15 POS[C]=77`	`: Before loading N15, an M70 intermediate block is generated in which the spindle is stopped, and M70 is output to the PLC.`
…	

Example 2:

Part program: Transition from positioning mode to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

Program code	Comment
`N05 SPOS=0`	
`N10 ...`	
`N15 C77`	`; Output of the implicit M70 to the PLC, no intermediate block.`
…	

Example 3:

Synchronized actions: Transition from spindle positioning mode to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

Program code	Comment
`WHEN COND1==TRUE DO SPOS=180`	
`WHEN COND2==TRUE DO POS[C]=270`	`; Output of the implicit M70 to the PLC.`

Example 4:

Synchronized actions: Transition from speed control mode to axis mode with M70

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

Program code	Comment
WHEN COND11==TRUE DO M3 S1000	
WHEN COND12==TRUE DO M70	; Output of M70 to the PLC.
WHEN COND13==TRUE DO POS[C]=270	; No generation of an implicit M70 because axis mode already exists.

Example 5:

Synchronized actions: Invalid transition from speed control mode to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

Program code	Comment
WHEN COND21==TRUE DO M3 S1000	
WHEN COND22==TRUE DO POS[C]=270	; Alarm 20141!

17.2.7 Initial spindle state

Spindle basic setting

The following machine data is used to specify a spindle mode as basic setting:

MD35020 $MA_SPIND_DEFAULT_MODE

Value	Spindle basic setting
0	Speed control mode, position control deselected
1	Speed control mode, position control activated
2	Positioning mode
3	Axis mode

Time when the spindle basic setting takes effect

The time when the spindle basic setting takes effect is set in the machine data:

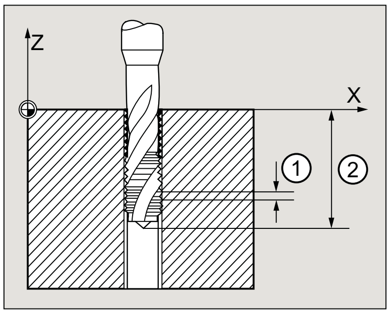
MD35030 $MA_SPIND_DEFAULT_ACT_MASK

Value	Effective time
0	POWER ON
1	POWER ON and program start
2	POWER ON and RESET (M2 / M30)

17.2.8 Tapping without compensating chuck

17.2.8.1 Function

For tapping without compensating chuck, the traversing motion of the linear axis and the spindle are interpolated, closed-loop position controlled. This requires a **position-controlled spindle** with **position measuring system**.



①	Thread pitch
②	Thread depth

Thread direction of rotation

The thread direction of rotation (right or left-handed thread) is defined by the sign of the pitch:

- Positive pitch → right-handed thread (direction of rotation corresponding to `M3`)
- Negative pitch → left-handed thread (direction of rotation corresponding to `M4`)

Defined spindle start position

When tapping without compensating chuck, a defined spindle start position (`SPOS`) must be programmed in the following cases:

- Threads produced using multiple machining operations
- Threads, for which a defined thread start position is specified

```
SPOS = <start position>

G331 ...
```

17.2.8.2 Programming

For tapping without compensating chuck, using the `G331` and `G332`commands, the following traversing motion is executed:

- `G331`: Tapping in the tapping direction up to the end of thread point
- `G332`: Retraction motion up to the tapping block `G331` with automatic spindle direction of rotation reversal

Syntax

```
G331 <axis> <thread pitch>
G331 <axis> <thread pitch> S...
```

```
G332 <axis> <thread pitch>
```

Meaning

`G331`:	Tapping The tapped hole is defined by the traversing motion of the axis (drilling depth) and the thread pitch.	
	Effective:	Modal
`G332`:	Retraction motion when tapping Retraction motion must have the same pitch as when tapping (`G331`). The direction of rotation of the spindle is reversed automatically.	
	Effective:	Modal
`<Axis>`:	Traversing distance/position of the geometry axis (X, Y or Z) at the end of the thread, e.g. Z50	
`<Thread pitch>`:	Thread pitch I (X), J (Y) or K (Z): • Positive pitch: Right-handed thread, e.g. K1.25 • Negative pitch: Left-handed thread, e.g. K-1.25	
	Value range:	±0.001 to ±2000.00 mm/revolution
`S...`:	Spindle speed The last active spindle speed is used if a spindle speed is not specified.	

Note

Second gear-stage data block

To achieve effective adaptation of spindle speed and motor torque and be able to accelerate faster, a second gear-stage data block for two further configurable switching thresholds (maximum speed and minimum speed) can be preset in axis-specific machine data deviating from the first gear step data block and also independent of these speed switching thresholds. The specifications of the machine manufacturer must be observed.

References:
Function Manual, Basic Functions; Spindles (S1), Section: " Configurable gear adaptation"

Examples

- Example: Tapping with G331 / G332 (Page 1305)
- Example: Output the programmed drilling speed in the current gear stage (Page 1305)
- Example: Application of the second gear-stage data block (Page 1306)
- Example: Speed is not programmed, the gearbox stage is monitored (Page 1306)
- Example: Gearbox stage cannot be changed, gearbox stage monitoring (Page 1306)
- Example: Programming without SPOS (Page 1307)

17.2.8.3 Example: Tapping with G331 / G332

Program code	Comment
N10 SPOS[n]=0	; Spindle: Position control mode
	; Start position 0 degrees
N20 G0 X0 Y0 Z2	; Axes: Approach starting position
N30 G331 Z-50 K-4 S200	; Tapping in Z,
	; Pitch K-4 negative =>
	; Direction of spindle rotation: CCW rotation,
	; Spindle speed 200 rpm
N40 G332 Z3 K-4	; Retraction motion in Z,
	; Pitch K-4 negative (counterclock-wise),
	; autom. direction of rotation reversal =>
	; Clockwise spindle direction of rota-tion
N50 G1 F1000 X100 Y100 Z100 S300 M3	; Spindle in spindle operation

17.2.8.4 Example: Output the programmed drilling speed in the current gear stage

Program code	Comment
N05 M40 S500	; Programmed spindle speed: 500 rpm =>
	; Gearbox stage 1 (20 to 1028 rpm)
...	
N55 SPOS=0	; Position the spindle
N60 G331 Z-10 K5 S800	; Tapping
	; Spindle speed 800 rpm => gearbox stage 1

The appropriate gear stage for the programmed spindle speed `S500` with `M40` is determined on the basis of the first gear-stage data block. The programmed drilling speed `S800` is output in the current gear stage and, if necessary, is limited to the maximum speed of the gear stage. No automatic gear-stage change is possible following an `SPOS` operation. In order for an automatic change in gear stage to be performed, the spindle must be in speed-control mode.

Note

If gearbox stage 2 is selected at a spindle speed of 800 rpm, then the switching thresholds for the maximum and minimum speed must be configured in the relevant machine data of the second gear-stage data block (see the examples below).

17.2.8.5 Example: Application of the second gear-stage data block

The switching thresholds of the second gear-stage data block for the maximum and minimum speed are evaluated for `G331`/`G332` and when programming an `S` value for the active master spindle. Automatic `M40` gear-stage change must be active. The gear stage as determined in the manner described above is compared with the active gear stage. If they are found to be different, then the gearbox stage is changed.

Program code	Comment
N05 M40 S500	; Programmed spindle speed: 500 rpm
...	
N50 G331 S800	; Master spindle: Gearbox stage 2 is selected
N55 SPOS=0	; Position the spindle
N60 G331 Z-10 K5	; Tapping
	; Spindle acceleration from second gearbox stage data block 2

17.2.8.6 Example: Speed is not programmed, the gearbox stage is monitored

If no speed is programmed when using the second gearbox stage data block with `G331`, then the last speed programmed will be used to produce the thread. The gear stage does not change. However, monitoring is performed in this case to check that the last speed programmed is within the preset speed range (defined by the maximum and minimum speed thresholds) for the active gear stage. Otherwise, alarm 16748 is output.

Program code	Comment
N05 M40 S800	; Programmed spindle speed: 800 rpm
...	
N55 SPOS=0	; Position the spindle
N60 G331 Z-10 K5	; Tapping
	; Monitoring the spindle speed, 800 rpm
	; Gearbox stage 1 is active
	; Gearbox stage 2 should be active => Alarm 16748

17.2.8.7 Example: Gearbox stage cannot be changed, gearbox stage monitoring

If the spindle speed is programmed in addition to the geometry in the `G331` block when using the second gear-stage data block, if the speed is not within the preset speed range (defined by the maximum and minimum speed thresholds) of the active gear stage, it will not be possible to change gear stages, because the path motion of the spindle and the infeed axis (axes) would not be retained.

As in the example above, the speed and gearbox stage are monitored in the `G331` block and alarm 16748 is signaled if necessary.

Program code	Comment
N05 M40 S500	; Programmed spindle speed: 500 rpm =>
	; Gearbox stage 1
...	
N55 SPOS=0	; Position the spindle

Program code	Comment
`N60 G331 Z-10 K5 S800`	`; Tapping`
	`; Gearbox stage cannot be changed,`
	`; Monitoring the spindle speed, 800 rpm`
	`; with gearbox stage data set 1: Gearbox stage 2`
	`; should be active => Alarm 16748`

17.2.8.8 Example: Programming without SPOS

Program code	Comment
`N05 M40 S500`	`; Programmed spindle speed: 500 rpm =>`
	`; Gearbox stage 1 (20 to 1028 rpm)`
`...`	
`N50 G331 S800`	`; Master spindle: Gearbox stage 2 is selected`
`N60 G331 Z-10 K5`	`; Tapping`
	`; Spindle acceleration from second gearbox stage data block 2`

Thread interpolation for the spindle starts from the current position, which is determined by the previously processed section of the part program, e.g. if the gear stage was changed. Therefore, it might not be possible to remachine the thread.

Note

Please note that when machining with multiple spindles, the drill spindle also has to be the master spindle. `SETMS(<spindle number>)` can be programmed to set the drill spindle as the master spindle.

17.2.8.9 Special case: Direction of rotation reversal via NC/PLC interface signal in the NC program

Normally, a request to invert the direction of rotation of the spindle via the NC/PLC interface signal "Invert M3 / M4" (DB31, ... DBX17.6) is realized by the PLC user program **before** the NC program is started. Only when starting does the NC automatically evaluate the status of the interface signal, and if necessary, change the spindle direction of rotation.

If, while the program is being executed, the spindle direction of rotation must again be changed as a result of the NC/PLC interface signal "Invert M3 / M4" (DB31, ... DBX17.6), then within an NC program, this can only be achieved using the subsequently described principle sequence:

`N110 SPCOF` or `M5`	Stop the spindle and switch into the closed-loop speed controlled mode
`N111 Mxx` or `Hxx`	Output a user-specific help function to initiate the following actions in the PLC user program: ● Set the read-in inhibit in the channel: DB21, ... DBX6.1 = 1 ● Invert the interface signal "Invert M3/M4" of the spindle: DB31, ... DBX17.6 ● Reset the read-in inhibit in the channel: DB21, ... DBX6.1 = 0

`N112 STOPRE`	Preprocessing stop
`N113 SPOS=IC(0.001)`	Incrementally traverse the spindle in the closed-loop position controlled mode. By incrementally traversing the spindle in the closed-loop position controlled mode, the NC again evaluates interface signal DB31, ... DBX17.6 (invert M3/M4). From this block onward, when traversing the spindle, the direction of reversal based on the interface signal is active.
`N114 SPOS=0`	**Optional**: Position the spindle if, technologically, a defined output position is required, e.g. 0°

Example

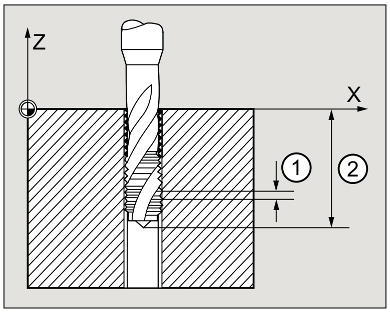
Program code	Comment
N010 G0 X0 Z0	; Initial position of the axes
N020 G1 G94 F1000 Z10	; Traverse the Z axis to the initial position
N030 M3 S100	; Traverse the spindle in the closed-loop speed controlled mode
	; 100 rpm with the spindle direction of rotation that was determined
	; at the start of the program
...	; Any arbitrary machining
N040 M5	; Stop the spindle and switch into the closed-loop speed controlled mode
N050 H10	; Auxiliary function output: H0=10
	; In the PLC user program:
	; - set the read-in inhibit: DB21, ... DBX6.1=1
	; - invert the direction of rotation: DB31, ... DBX17.6
	; - reset the read-in inhibit: DB21, ... DBX6.1=0
N060 STOPRE	; Preprocessing stop
N070 SPOS=IC(0.001)	; Incrementally traverse the spindle in the closed-loop position controlled mode.
N080 SPOS=0	; Position the spindle
N090 G331 Z-10 K1 S500	; Tapping without compensating chuck
N100 G332 Z2 K1	; Tapping without compensating chuck Retraction movement
N110 G1 G94 F1000 Z10	; Retract to the initial position of the Z axis
N120 SPCOF	; Switch the spindle into the closed-loop speed controlled mode
N130 H10	; See above
N140 STOPRE	; Preprocessing stop
N150 SPOS=IC(.001)	; Incrementally traverse the spindle in the closed-loop position controlled mode.
N160 SPOS=0	; Position the spindle
N170 G331 Z-10 K1 S500	; Tapping without compensating chuck
N180 G332 Z2 K1	; Tapping without compensating chuck Retraction movement
N190 G1 G94 F1000 Z10	; Traverse the Z axis to the initial position

Program code	Comment
...	

17.2.9 Tapping with compensating chuck

17.2.9.1 Function

When tapping with compensating chuck (`G63`) the spindle is not interpolated with the linear axis, but is operated in the closed-loop speed controlled mode. The feedrate of the linear axis dependent on the spindle speed and the thread pitch must be calculated and explicitly programmed. As the spindle and linear axis are not interpolated together, to compensate spindle speed fluctuations for example, a tapping drill in a length-compensating chuck is required.



Retraction movement

The retraction motion is also programmed with `G63` - but with the opposite spindle direction of rotation.

Axis and spindle override value

While the "Tapping **with** compensating chuck" function is selected, 100% is active as axis and spindle override value.

Feedrate

Feedrate F of the linear axis to be programmed is obtained from the product of the spindle speed and the thread pitch of the tapping drill:

F [mm/min] = spindle speed S [rpm] *thread pitch [mm/U]

17.2.9.2 Programming

For tapping with compensating chuck, using the `G63` command, the following traversing motion is executed:

- `G63`: Tapping in the tapping direction up to the end of thread point
- `G63`: Retraction motion with programmed spindle direction of rotation reversal

Note

After a `G63`block, the last effective interpolation type `G0`, `G1`, `G2` is active.

Syntax

```
G63 <axis> <direction of rotation> <speed <feedrate>
```

Meaning

`G63`:	Tapping with compensating chuck	
	Effective:	Non-modal
`<Axis>`:	Traversing distance/position of the geometry axis (X, Y or Z) at the end of the thread, e.g. Z50	
`<Direction of rotation>`:	Direction of spindle rotation: ● `M3`: Clockwise rotation, right-hand thread ● `M4`: Counterclockwise rotation, left-hand thread	
`<Speed>`:	Maximum permissible spindle speed while tapping, e.g. S100	
`<Feedrate>`:	Feedrate of the tapping axis F, with F = spindle speed * thread pitch	

Example

Tapping an M5 thread:

- Spindle pitch according to the standard: 0.8 mm/rev
- Spindle speed S: 200 rpm
- Feedrate F = 200 rpm * 0.8 mm/rev = 160 mm/min.

Program code	Comment
`N10 G1 X0 Y0 Z2 F1000 S200 M3`	`; Approach starting point`
	`; Spindle clockwise direction of rotation, 200 rpm`
`N20 G63 Z-50 F160`	`; Tapping with compensating chuck`
	`; Drilling depth: absolute Z=50mm`
	`; Feedrate: 160 mm/min`
`N30 G63 Z3 M4`	`; Retraction movement: absolute Z=3mm`
	`; Direction of rotation reversal`
	`; Spindle with counterclockwise direction of rotation, 200 rpm`

17.3 Reference / synchronize

Why synchronize?

In order to ensure that the controller detects the exact position of the spindle when it is switched on, the controller must be synchronized with the position measuring system of the spindle.

The following functions are possible only with a synchronized spindle:

- Thread cutting
- Tapping without compensating chuck
- Axis programming

For further information about synchronizing the spindle, see Section "R1: Referencing (Page 1223)".

Why reference?

In order to ensure that the controller detects the exact machine zero when it is switched on, the controller must be synchronized with the position measurement system of the rotary axis. This process is known as referencing. The sequence of operations required to reference an axis is known as search for reference.

Only a referenced axis can approach a programmed position accurately on the machine.

For further information about referencing the rotary axis, see Section "R1: Referencing (Page 1223)".

Installation position of the position measurement system

The position measurement systems can be installed as follows:

- directly at the motor in combination with a contactless proximity switch on the spindle as a zero mark encoder
- at the motor using a measuring gearbox in combination with a contactless 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 contactless proximity switch on the spindle as a zero mark encoder (only with ratios not equal to 1:1)

Where two position measuring systems are provided, they can be installed either in the same location or separately.

Synchronization procedure

When the spindle is switched on, it can be synchronized as follows:

- The spindle is started with a spindle speed (`S` value) and a spindle rotation (`M3` or `M4`) and synchronized with the next zero mark of the position measurement system or with the next signal from the contactless proximity switch.
- 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 proximity switch signal. It is then positioned to the programmed position.
- The spindle can be synchronized from the motion (after `M3` or `M4`) using `SPOS`, `M19` or `SPOSA`.
  The responses are as follows:
  - With `SPOS=<Pos>`, `SPOS=DC(<Pos>)` and `SPOS=AC(<Pos>)`, the direction of motion is retained and the position is approached.
  - With `SPOS = ACN(<Pos>)` or `SPOS = ACP(<Pos>)`, the position is always approached with a negative or positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.
- Crossing the zero mark in JOG mode by means of direction keys in speed control mode.

Note

It does not make any difference whether the synchronization procedure is initiated from the part program, FC 18 or synchronized actions.

Note

During synchronization of the spindle, all four possible reference point values are effective depending on the measuring system selected. The measurement system offset has the same effect.

The following machine data must be observed:

- MD34080 $MA_REFP_MOVE_DIST
  (Reference point distance / destination point for a distance-coded system)
- MD34090 $MA_REFP_MOVE_DIST_CORR
  (Reference point offset / absolute offset, distance-coded)
- MD34100 $MA_REFP_SET_POS
  (Reference point value, with distance-coded system without any significance)

If a non-referenced spindle with `SPOS=IC(...)` and a path < 360 degrees is positioned, it may be the case that the zero mark is not crossed and the spindle position is still not synchronized with the zero mark. This can happen:

- After POWER ON
- By setting the axial NC/PLC interface signals:
  DB31, ... DBX17.5 (resynchronize spindle when positioning 2)
  DB31, ... DBX17.4 (resynchronize spindle when positioning 1)

Special features when synchronizing using a contactless proximity switch

The position error caused by the signal delay when using a contactless proximity switch can be corrected internally in the NC by entering a signal runtime compensation.

The signal runtime compensation is set by means of the machine data:

- MD31122 $MA_BERO_DELAY_TIME_PLUS
  (delay time of the contactless proximity switch, positive motion direction)
- MD31123 $MA_BERO_DELAY_TIME_MINUS
  (delay time of the contactless proximity switch, negative motion direction)

The effect depends on the selected referencing mode:

MD34200 $MA_ENC_REFP_MODE = <value>

<Value>	Meaning
2	The position is synchronized without entering a specific starting velocity/speed.
7	As starting velocity to synchronize the position, the velocity from the following machine data is used: MD34040 $MA_REFP_VELO_SEARCH_MARKER (reduced velocity) The zero mark is not automatically looked for, it has to be requested explicitly with the 0/1 edge of the NC/PLC interface signal: DB31, ... DBX16.4 / 5 (resynchronize spindle, measuring system 1 / 2)

Note

Generally, no change is required as signal runtimes and signal runtime compensation are preset when the system is delivered.

Referencing sequence

If the spindle is to be programmed in axis mode directly after controller power-up, it must be ensured that the axis is referenced.

When the controller is switched on, the spindle can be referenced (condition is one zero mark per revolution).

For information about the referencing sequence, see Section "R1: Referencing (Page 1223)".

The rotary axis is referenced at the same time as the spindle is synchronized (see section "Synchronization procedure") if the position measuring system used for the spindle is also used for the rotary axis.

Position measurement systems, spindle

The spindle can be switched from spindle mode to axis mode (rotary axis) if a single motor is used for spindle mode and axis mode.

The spindle (spindle mode and axis mode) can be equipped with one or two position measurement systems. With two position measurement systems, it is possible to assign one position measurement system to the spindle and the other to the rotary axis, or to assign two position measurement systems to the spindle. Where two position measurement systems are provided, both are updated by the controller, but only one can be active.

The active position measuring system is selected with the NC/PLC interface signal:

DB31, ... DBX1.5 / 6 (position measuring system 1 / 2)

The active position measurement system is required for the following functions:

- Position control of the spindle (`SPCON`)
- Spindle positioning (`SPOS`, `M19` and `SPOSA`)
- Thread cutting (`G33`, `G34`, `G35`)
- Tapping without compensating chuck (`G331`, `G332`)
- Revolutional feedrate (`G95`)
- Constant cutting rate (`G96`, `G961`, `G97`, `G971`)
- Spindle actual speed display
- Axis mode
- Synchronous spindle setpoint coupling

Resynchronizing the position measuring system for the spindle

In the following cases, the spindle position measurement system must be resynchronized:

- The position encoder is on the motor, a contactless proximity switch is mounted on the spindle and the gear stage is changed. Synchronization is triggered internally once the spindle is rotating in the new gear stage (see Synchronization procedure).
- The machine has a selector switch for a vertical and horizontal spindle. Two different position encoders are used (one for the vertical spindle and one for the horizontal spindle), but only one actual value input is used on the controller. When the system switches from the vertical to the horizontal spindle, the spindle must be resynchronized.
  This synchronization is initiated with the NC/PLC interface signal:
  DB31, ... DBX16.4 (resynchronize spindle 1)
  or
  DB31, ... DBX16.5 (resynchronize spindle 2)
  The spindle must be in open-loop control mode.

Restoring the position after a warm restart

For spindles with incremental position measuring systems, it is possible to buffer the actual values after a POWER OFF and after POWER ON, to restore the position last buffered before switching-off, in order that position-dependent functions, e.g. transformations can be restored (see Section "Automatic restoration of the machine reference (Page 1264)"). One application is, e.g. tool retraction after a warm restart when machining with tool orientation (see Section "Tool withdrawal after POWER ON with orientation transformation (Page 656)").

The following NC/PLC interface signals display the state of the position measuring system after the position has been restored:

DB31, ... DBX71.4 ("Restored 1") for position measuring system 1

DB31, ... DBX71.5 ("Restored 2") for position measuring system 2

Once the tool has been retracted in JOG mode, axes whose positions have been restored are referenced. As a consequence, signals DB31, ... DBX71.4/5 ("Restored 1/2") are deleted and signals DB31, ... DBX60.4/5 ("Referenced/synchronized 1/2") are set.

Note

If machine data MD20700 $MC_REFP_NC_START_LOCK is set to a value of "2", then an NC start is also possible with "restored" axis positions (in the MDI mode or when overstoring).

17.4 Configurable gear adaptation

17.4.1 Gear stages for spindles and gear change change

Why do we need gear stages?

Gear stages are used on spindles to step down the speed of the motor in order to generate a high torque at low spindle speeds or to step up in order to maintain a high speed.

No. of gear stages

Five gear stages can be configured for each spindle.

The number of used gear stages is defined in machine data:

MD35090 $MA_NUM_GEAR_STEPS

Parameterization of the gear stages

The gear stages 1 to 5 can be parameterized via the following machine data:

Machine data	Meaning
MD35012 $MA_GEAR_STEP_CHANGE_POSITION[<n>]	Gear stage change position
MD35110 $MA_GEAR_STEP_MAX_VELO[<n>]	Maximum speed for automatic gear stage change
MD35120 $MA_GEAR_STEP_MIN_VELO[<n>]	Minimum speed for automatic gear stage change
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]	Maximum speed of gear stage
MD35135 $MA_GEAR_STEP_PC_MAX_VELO_LIMIT[<n>]	Maximum speed of gear stage in position control
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[<n>]	Minimum speed of gear stage
MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL[<n>]	Acceleration in speed control mode
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL[<n>]	Acceleration in position control mode
MD35300 $MA_SPIND_POSCTRL_VELO[<n>]	Position control activation speed

Machine data	Meaning
MD35310 $MA_SPIND_POSIT_DELAY_TIME[<n>]	Positioning delay time
MD35550 $MA_DRILL_VELO_LIMIT[<n>]	Maximum speed for tapping without compensating chuck

Type of gear stage change

The type of gear stage change is set in machine data:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE

Bit	Value	Meaning
0	0	The spindle motor is attached to the spindle directly (1:1) or with a non-variable transmission ratio (basic setting). The machine data of the first gear stage is effective.
	1	Spindle motor with up to five gear stages. The gear stage change takes place: • In oscillation mode • At indefinite change position
1	0	Meaning as in Bit 0 = 0.
	1	Meaning as in Bit 0 = 1, however, the gear stage change takes place at the configured spindle position. The change position is set in machine data: MD35012 $MA_GEAR_STEP_CHANGE_POSITION The position is approached in the current gear stage before the gear stage change. If Bit 1 is set, then Bit 0 is ignored!
3	1	The gear stage change dialog between NC and PLC is simulated.
5	1	The second gear stage data set is used while tapping with G331/G332 (see the following paragraph "Second gear stage data set"). The bit must be set for the master spindle used during the tapping.

Requirement for a gear stage change

In principle, the gear stage change is only performed if the requested gear stage is not the same as the active gear stage.

Parameter set selection during gear stage change

The servo parameter set is also changed over with the gear stage if:

MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0 or 1

For further information, see Section "Parameter set selection during gear step change (Page 1329)".

Request gear stage change

A gear stage change can be requested:

- In the part program using:
  - `M40 S...`
    Automatic gear stage selection to the programmed speed `S...`
  - `M41` ... `M45`
    Direct selection of gear stages 1 ... 5
  - `M70`
    For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5
    (see "Configurable gear step in M70 (Page 1345)")
  - `G331 S...`
    For MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, Bit 5 = 1
- In synchronized actions using:
  - `DO M40 S...`
    Automatic gear stage selection to the programmed speed `S...`
  - `DO M41... M45`
    Direct selection of gear stages 1 ... 5
  - `DO M70`
    For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5
- Through the PLC using the FC18 function block
- In the reset state through description of NC/PLC interface:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  The mechanically active gear stage can be communicated to the NC especially after a POWER ON.

Note

If the spindle motor is attached to the spindle directly (1:1) or with a non-variable transmission ratio (MD35010 = 0), then the M40 and M41 ... M45 auxiliary functions are not relevant to this spindle.

Gear stage change

Gear stage selection between two gear stages with specification of a maximum spindle speed is shown in the example below:

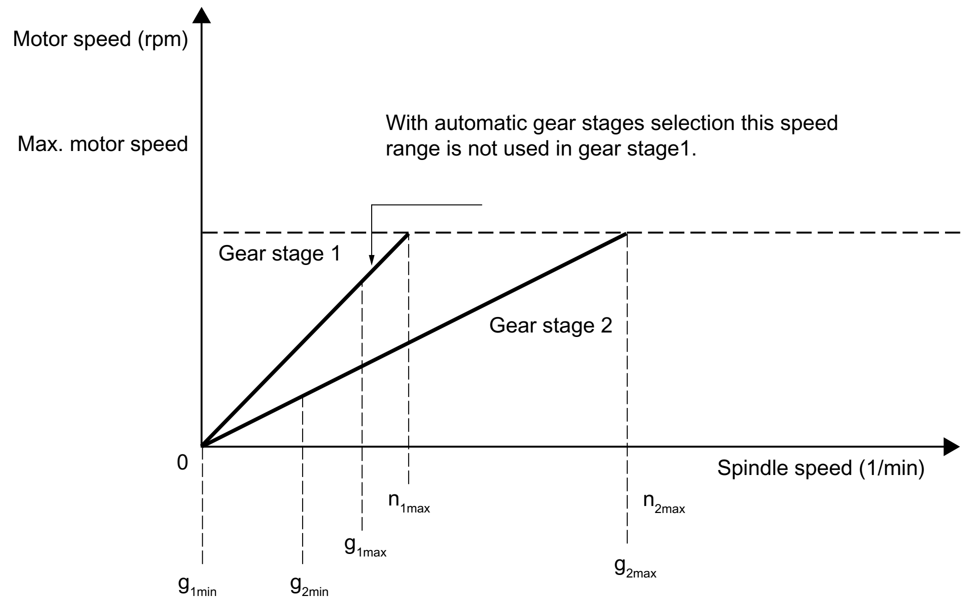


Figure 17-3 Gear stage change with selection between two gear stages

Process sequence of the gear stage change

If the new gear stage is preselected, the following sequence is implemented:

1. Changeover sequence
   The two following NC/PLC interface signals are set:
   DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)
   DB31, ... DBX82.3 (change over gear stage)
   In accordance with the point at which NC/PLC IS:
   DB31, ... DBX18.5 (oscillation speed)
   is set, the spindle decelerates to a standstill at the acceleration for oscillation or at the acceleration for speed control / position control.
   Oscillation can be activated at the latest when the spindle reaches a standstill:
   DB31, ... DBX61.4 (axis/spindle stationary)
   with NC/PLC IS:
   DB31, ... DBX18.5 (oscillation speed).

   In principle, the new gear stage can also be engaged without oscillation
   When the new gear stage is engaged, the following NC/PLC interface signals are set by the PLC program:
   DB31, ... DBX16.0-16.2 (actual gear stage A to C)
   DB31, ... DBX16.3 (gear is changed)
2. End of gear stage change
   The gear stage change is treated as completed (spindle operation type "oscillation mode" is deselected), if the following NC/PLC interface signal is set:
   DB31, ... DBX16.3 (gear is changed)
   The new actual gear stage is changed to the servo and interpolation parameter set when the motor is stationary.
   With NC/PLC interface signal:
   DB31, ... DBX16.3 (gear is changed)
   is used to communicate to the NC that the new gear stage is valid and the oscillation mode can be completed.
   The NC/PLC interface signal:
   DB31, ... DBX82.3 (change gear)
   is reset by the NC,
   which causes the PLC program to reset NC/PLC IS:
   DB31, ... DBX16.3 (gear changed).

   In this case, it does not matter whether NC/PLC IS:
   DB31, ... DBX18.5 (oscillation mode)
   is still set.
   The actual gear stage, which should correspond to the set gear stage, is relevant
   for selecting the parameter set.
   If this is not the case, then Alarm 22010 :
   MD11410 $MN_SUPPRESS_ALARM_MASK, Bit 3 = 0
   is output.
   Following acknowledgment of gear stage change via the PLC
   with NC/PLC IS:
   DB31, ... DBX16.3 (gear changed)
   the spindle is in speed control mode (DB31, ... DBX84.7).
   For further information on the signal exchange between PLC and NC, see Section "A2: Various NC/PLC interface signals and functions (Page 41)".

Second gear stage data set

The automatic gear stage change M40 can be extended by a second configurable gear stage data set.

The second gear stage data set is used **exclusively** in connection with tapping without compensation chuck (G331, G332) so that an effective adjustment of spindle speed and motor torque can be achieved.

The activation is undertaken by setting the following bit for the master spindle:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, Bit 5 = 1

The number of used gear stages of the second gear stage data set is defined with the machine data:

MD35092 $MA_NUM_GEAR_STEPS2

The second gear stage block data set is deactivated if:

MD35092 $MA_NUM_GEAR_STEPS2 = 0 (basic setting)

The first gear stage data set then selects the gear stage when M40 is active.

Note

The number of gear stages in the second data set can vary from the first. If no appropriate gear stage is found for a programmed speed for M40, then no gear stage change is carried out (exceptions, see "M40: Automatic gear stage selection for speeds outside the configured switching thresholds (Page 1375)").

For more information about a typical program sequence in thread cutting without compensating chuck G331/G332 see:
References:
Programming Manual - Fundamentals; Motion Commands

The gear stages 1 to 5 of the second gear stage data set can be parameterized via the following machine data:

Machine data	Meaning
MD35112 $MA_GEAR_STEP_MAX_VELO2[n]	Maximum speed for automatic gear stage change
MD35122 $MA_GEAR_STEP_MIN_VELO2[n]	Minimum speed for automatic gear stage change
MD35212 $MA_GEAR_STEP_POSCTRL_ACCEL2[n]	Acceleration in position control mode

---

Note

The number of servo parameter sets concerning the mechanical factors remain unchanged. Furthermore, five mechanical gear stages for the spindle and one for the axis operation can be configured.

The speed limitations are configured only once for each gear stage with the following machine data, independently of the different switching thresholds:

Machine data	Meaning
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum speed of gear stage
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[n]	Minimum speed of gear stage

For tapping without compensating chuck (G331, G332) the speed can be limited to the linear acceleration range of the motor additionally. For this, the maximum speed of the linear motor characteristics range is specified in the following machine data as a function of the gear stage:

MD35550 $MA_DRILL_VELO_LIMIT[n]

Specify gear stage in the part program

Automatic selection with active M40

The gear stage is automatically selected by the control. The gear stage in which the programmed spindle speed (`S...`) is possible is checked in this context. If a gear stage results from this that is not equal to the current (actual) gear stage, then the following NC/PLC interface signals are set:

DB31, ... DBX82.3 (change over gear stage)

DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)

While the appropriate gear stage is being determined, a gear stage change is only requested if the new speed is not within the permissible speed range of the active gear stage.

The speed is limited to the maximum speed of the current gear stage or raised to the minimum speed of the current gear stage and the appropriate NC/PLC interface signal is set:

DB31, ... DBX83.1 (speed setpoint limited)

DB31, ... DBX83.2 (speed setpoint increased)

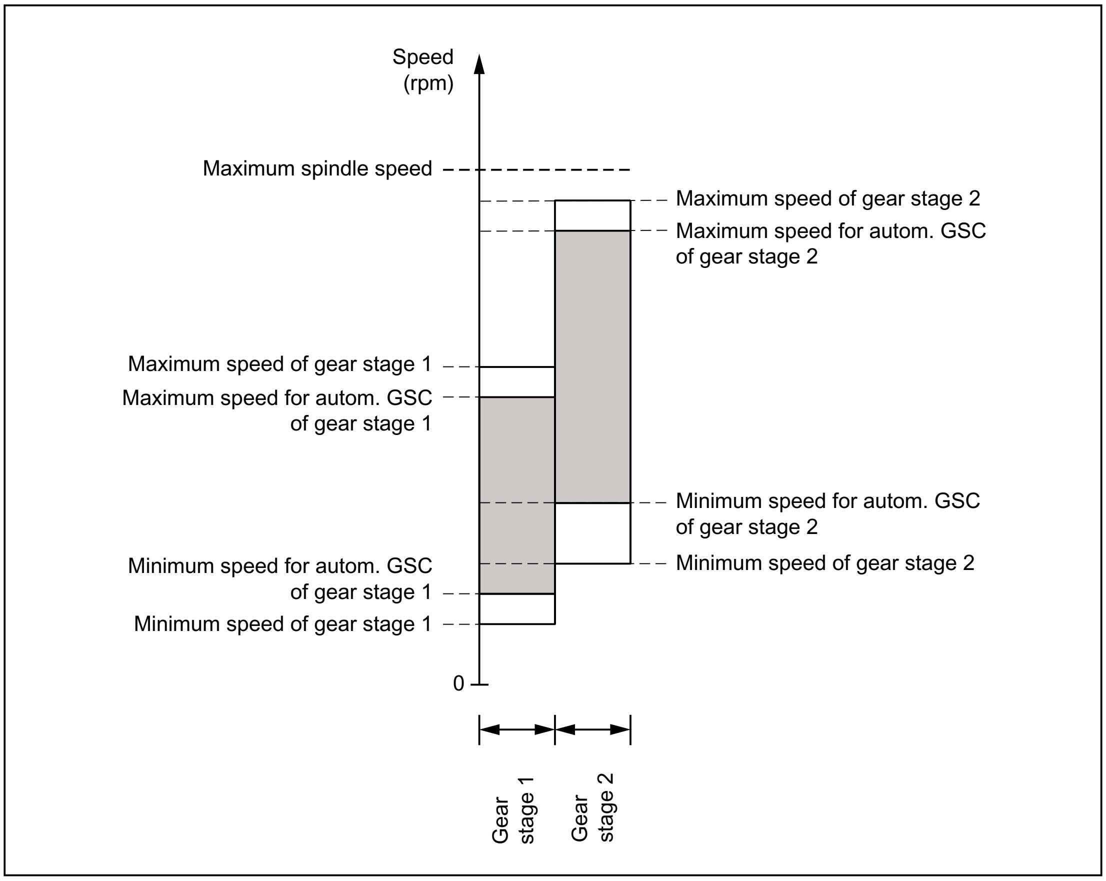

Speed (rpm)
Maximum spindle speed
Maximum speed of gear stage 2
Maximum speed for autom. GSC of gear stage 2
Maximum speed of gear stage 1
Maximum speed for autom. GSC of gear stage 1
Minimum speed for autom. GSC of gear stage 2
Minimum speed of gear stage 2
Minimum speed for autom. GSC of gear stage 1
Minimum speed of gear stage 1
0
Gear stage 1
Gear stage 2

Figure 17-4 Example for two gear stages with overlapping speed ranges for automatic gear stage change (M40)

Note

In the case of M40, the spindle must be in open-loop control mode for automatic gear stage selection with an `S` word. Otherwise the gear stage change is rejected and the following alarm is set:

Alarm 22000 "gear stage change is not possible"

Note

An active reduction gear is not considered in the selection for the automatic gear stage change.

Permanently defining the gear stage with M41 to M45

The gear stage can be permanently defined in the part program with `M41` to `M45`.

If a gear stage is specified via `M41` to `M45` that is not equal to the current (actual) gear stage, then the following NC/PLC interface signals are set:

DB31, ... DBX82.3 (change over gear stage)

DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)

The programmed spindle speed (`S...`) then refers to this permanently defined gear stage:

- If a spindle speed is programmed and it is higher than the maximum speed of the permanently defined gear stage (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT), then the speed is decreased to this limit and the following NC/PLC interface signal is set:
  DB31, ... DBX83.1 (speed setpoint limited)
- If a spindle speed is programmed and it is lower than the minimum speed of the permanently defined gear stage (MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT), then the speed is increased to this minimum and the following NC/PLC interface signal is set:
  DB31, ... DBX83.2 (speed setpoint increased)

Block change

When programming the gear stage change in the part program, the gear stage change set remains active until it is aborted by PLC.

This corresponds to the effect as if the following NC/PLC interface signal were set:

DB21, ... DBX6.1 (read-in disable)

Specification of gear stage via PLC with FC18

The gear stage change can also be performed by function block FC18 during a part program, in the reset state or in all operating modes.

If the speed and direction of rotation is specified with FC18, the NC can be requested to select the gear stage as appropriate for the speed. This corresponds to an automatic gear stage change with M40.

The gear stage is not changed if:

- The spindle is positioned via FC18.
- The spindle is traversed in the axis mode.

For further information on the FC18 function block, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)".

Specification of a gear stage in synchronized actions

The gear stage change can be requested by synchronized actions using:

- `DO M40 S...`
  Automatic gear stage selection to the programmed speed `S...`
- `DO M41... M45`
  Direct selection of gear stages 1 ... 5
- `DO M70`
  For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5
  (see "Configurable gear step in M70 (Page 1345)")

The gear stage is **not** changed if:

- The spindle is positioned via synchronized actions.
- The spindle is traversed in the axis mode.

Note

For further details, please refer to the section "Specification of a gear stage in part program".

Exception:

The block change is not affected by the specification of a gear stage in synchronized actions.

Manual specification of a gear stage

Outside a part program that is running, the gear stage can also be changed without a request from the NC or the machine. This is the case, for example, when a gear stage is changed manually.

To select the appropriate parameter set, the NC must be informed of the current gear stage. To enable this, the control or the part program must be in the reset state.

Supplementary conditions

Transfer of the gear stage to the NC is initiated when
NC/PLC IS:
DB31, ... DBX16.0-16.2 (actual gear stage A to C) changes.

These three bits must be set continuously during operation.

Successful transfer is acknowledged with NC/PLC IS:
DB31, ... DBX82.0-82.2 (set gear stage A to C)
to the PLC.

NC/PLC IS:
DB31, ... DBX16.3 (gear changed)
must not be set.

If position control is active when a new gear stage is specified by the PLC with
DB31, ... DBX16.0-16.2, then it is switched off for the duration of this changeover sequence.

NC stop during gear stage change

The spindle cannot be stopped with NC/PLC IS:
DB21, ... DBX7.4 (NC stop)
if:

- The spindle is not yet in oscillation mode for the gear stage change.
- NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  is not set.

Note

Options for aborting:

DB31, ... DBX2.2 (delete distance-to-go / spindle reset)
or
DB31, ... DBX16.3 (gear changed)
with corresponding acknowledgment from actual gear stage:
DB31, ... DBX16.0-16.2 (actual gear stage).

Spindle response after a gear stage change

How the spindle behaves once the gear stage has been changed depends on the following initial conditions:

- If the spindle was in the stop state before the gear stage change (M5, FC18: "Stop rotate spindle"), in positioning or axis mode, M5 (spindle stop) is active after completion of the gear stage change.
- If a direction of rotation
  (M3, M4, FC18: "Start spindle rotation"), then the last speed and direction of rotation will become active again after the gear stage change. In the new gear stage, the spindle accelerates to the last spindle speed programmed (`S...`).
- If position control was active before the gear stage change (`SPCON`), then it is reactivated after the gear stage change.
  The next block in the part program can be executed.

Special features

The following points must be observed on gear stage change:

- The gear stage change is not terminated by selecting
  NC/PLC IS:
  DB31, ... DBX20.1 (run-up switchover to V/f mode).

  Setpoint 0 is output.
  The gear stage change is acknowledged as usual
  via the NC/PLC interface signal:
  DB31, ... DBX16.3 (gear is changed)
- The "Ramp-function generator rapid stop" signal must be reset by the PLC before the gear stage change is completed by the PLC.
- The process sequence of the gear stage change is ended during NC reset without any alarm output.
  The gear stage output with NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  is applied by the NC.

Star/delta switchover with FC17

Digital main spindle drives can be switched in both directions between star and delta using FC17, even when the spindle is running. This automatic switchover is controlled by a defined logic circuit in FC17 which provides the user with a configurable switchover time for the relevant spindle.

For further information on the FC17 function block, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 869)".

17.4.2 Spindle gear stage 0

Technical background

For machine's where the spindle load gear can be changed over, situations can occur where the gear train between the motor and load (workpiece/tool) is interrupted. This state can occur, e.g. when pressing Reset or Emergency Stop while performing a gear stage change or when the machine is commissioned for the first time while it is being installed. The control must identify this state where the gear train is open and the next gear stage change request must be unconditionally executed.

Function

When the gear is disengaged, the binary-coded value "0" (≙ gear stage 0) is transferred to the NC from the PLC using the interface signal bits DB31, ... DBX16.0-2 (actual gear stage A to C):

DB31, ... DBX16.0-2 = 0

The value is used by the control to identify the state where the gear train is open.

Effects on the gear stage change

Gear stage change in the part program

The actual gear stage signaled from the PLC is read by the NC when starting a part program. If, at this instant in time, a value of "0" is read for the actual gear stage, then the next gear stage change is executed and the gear stage change dialog is performed by the PLC. If a value greater than "0" is read, then already in the program a comparison is made between the requested and active gear stage. If both gear stages are the same, the gear stage is not changed and a possibly programmed path motion is not interrupted.

Gear stage change in synchronized actions, FC18 and DBB30

The actual gear stage signaled from the PLC is always evaluated by the NC when the gear stage is changed. The gear stage is always changed if a value of "0" is read from the NC. When reading a value greater than "0", a comparison is made between the requested and active gear stage. The gear stage is only changed with the PLC if the two values are not equal and the NC/PLC interface signal DB31, ... DBX82.3 (change over gear) is then output.

Boundary conditions

- **Output of DB31, ... DBX16.0-2 = 0**
  When the gear is disengaged, the PLC must enter gear stage 0 in the NC/PLC interface DB31, ... DBX16.0-2 (actual gear stage A to C).
- **Enabling the gear stage change**
  The precondition for a gear stage change after reaching gear stage 0 is the general enable of the gear stage change via via machine data:
  MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear stage change)
  MD35090 $MA_NUM_GEAR_STEPS (number of gear stages set up)
  MD35092 $MA_NUM_GEAR_STEPS2 (2nd gear stage data set: Number of gear stages that have been created) if MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, bit 5 = 1 (tapping without compensating chuck)
- **PLC user program/ POWER ON ASUB**
  The PLC user program or POWER ON ASUB should ensure that when the gear is disengaged (gear stage 0) before spindle motion, a gear stage change request is programmed. For instance, this can be realized with `M41` in the ASUB. Spindle motion such as e.g. in JOG or in axis operation does not generate any gear stage change itself.

Example

Example for the sequence to select the first gear stage after POWER ON

1. POWER ON.
2. The PLC user program determines, in the mechanical environment, the "Gear is disengaged" state.
3. The PLC transfers the "Gear is disengaged" state to the NC by setting:
   DB31, ... DBX16.0-2 = 0
4. Part program start or POWER ON ASUB.

5. N05 (part program, refer below) is executed:
   The gear stage is changed to gear stage 1.
   From the NC:
   - the following NC/PLC-interface signal is set:
     DB31, ... DBX82.3 (change over gear stage)
   - the setpoint gear stage 1 is signaled to the PLC:
     DB31, ... DBX82.0 = 1
     DB31, ... DBX82.1 = 0
     DB31, ... DBX82.2 = 0
6. Mechanical gear stage change, acknowledgement
   If the gear stage is selected, then from the PLC:
   - the following NC/PLC-interface signal is set:
     DB31, ... DBX16.3 (gear is changed)
   - Actual gear stage 1 signaled to the NC:
     DB31, ... DBX16.0 = 1
     DB31, ... DBX16.1 = 0
     DB31, ... DBX16.2 = 0
7. N80 is executed:
   Due to the optimization of the gear stage change frequency in the part program, the gear stage is not changed.

Part program:

Program code	**Comment**
`N05 M41`	`; Select 1st gear stage`
`...`	
`N80 M41`	`; No gear stage change, if the 1st gear stage is selected.`

Configuring data for spindle 1 (AX5):

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE[AX5] = 1 (enable gear stage change)

17.4.3 Determining the spindle gear stage

The actual stage of a spindle can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

$VC_SGEAR[<n>]	Currently selected spindle gear stage $VC_SGEAR reads the actual gear stage signaled from the PLC. Value range: 0 ... 5

$AC_SGEAR[<n>]	Active spindle gear stage $AC_SGEAR reads the setpoint gear stage in the main run. Value range: 1 ... 5 The data set for the spindle is activated corresponding to this gear stage.

Note

For a block search, the actual gear stage ($VC_SGEAR[<n>]) can differ from the setpoint gear stage ($AC_SGEAR[<n>]) as, during the block search, no gear stage change takes place. Therefore, using $VC_SGEAR[<n>] and $AC_SGEAR[<n>], it can be interrogated whether a gear stage change should be made after a block search.

- Without preprocessing stop in the part program via system variables:

$P_SGEAR[<n>]	Setpoint gear stage $P_SGEAR reads the gear stage programmed in the part program (`M41` ... `M45`), for `M40` selected, or for `M70`, the configured gear stage.
$P_SEARCH_SGEAR[<n>]	Block search: Gear-specific M function $P_SEARCH_SGEAR contains the last programmed gear stage M function collected with the block search.

17.4.4 Parameter set selection during gear step change

Servo parameter sets

The servo parameter sets 1 to 6 adapt the position controller to the changed properties of the machine during a gear change of the spindle.

Parameter set selection during gear stage change

The gear stage parameter set (interpolation parameters) and, depending on the setting in the following machine data, the servo parameter set are also modified during gear stage change.

MD35590 $MA_PARAMSET_CHANGE_ENABLE (parameter set change possible)

Value	Meaning
0	In-system parameter set selection The parameter sets of the servo are assigned permanently. The following applies: • For axes and spindles in the axis mode, the first parameter set is active in principle. Exception: For G33, G34, G35, G331 and G332, for the axes involved, the parameter set with the following number is activated: Master spindle gear stage + 1 (corresponds to parameter set No. 2 ... 6) • For spindles in the spindle mode, the parameter set is set matching the gear stage.
1	Besides the in-system parameter set selection, there is also the option of an "external" parameter set selection. • By the PLC (DB31, ... DBX 9.0 - 9.2) • Via programming of `SCPARA` in the part program or in synchronized actions However, the in-system parameter set selection has priority. Note: Value 1 is relevant only to axes.
2	The servo parameter set is specified **exclusively** by the PLC (DB31, ... DBX 9.0 - 9.2) or through the programming of `SCPARA` in the part program or in synchronized actions (for axes and spindles). The 1st parameter set is selected after POWER ON.

Spindle mode

MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0 or 1

The parameter set is selected according to the gear stage + 1.

The active gear stage is located in:

DB31, ... DBX16.0-16.2 (actual gear stage A to C)

The active parameter set is output in:

DB31, ... DBX69.0-69.2 (controller parameter set A to C)

One set of parameters, with the following assignment, is provided by the NC for each of the five gear stages:

Data set for ...	NC/PLC interface	Parameter set	Parameter set
	DBX 69.2 / 69.1 / 69.0	Number	Index [n]
Axis mode	Last active gear stage	1	0
Gear stage 1	001	2	1
Gear stage 2	010	3	2
Gear stage 3	011	4	3
Gear stage 4	100	5	4
Gear stage 5	101 110 111	6	5

Spindle in axis mode

If the spindle is in axis mode, the parameter set index "0" is selected in the servo (note MD35590 $MA_PARAMSET_CHANGE_ENABLE!).

The gear stage change behavior depends on the setting in the machine data:

MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE (gear stage for axis mode in M70)

If there is no gear stage configured for axis mode (MD35014 = 0), no implicit gear stage change takes place in M70 (default setting!). The last gear stage is saved internally and is reactivated with the associated parameter set during the next spindle programming.

If, however, a gear stage is configured for axis mode (MD35014 = 1 ... 5), a gear stage change to gears 1 ... 5 takes place during the execution of M70. When changing from axis mode to spindle mode, the gear stage loaded with M70 remains activated. The gear stage which is activated in spindle mode prior to M70 is not automatically loaded again.

See also "Configurable gear step in M70 (Page 1345)."

Load gearbox transmission ratio

It is possible to configure positive or negative **load gearbox factors** for each gear stage and in axis mode.

The setting is undertaken separately for numerator and denominator via the machine data:

MD31050 $MA_DRIVE_AX_RATIO_DENOM[n] (load gearbox denominator)

MD31060 $MA_DRIVE_AX_RATIO_DENOM[n] (load gearbox numerator)

The setting range is the same size for positive and negative load gearbox factors.

It is not possible to enter the value "0".

Note

If an indirect encoder is configured, and the load gearbox transmission ratio changes, then the reference is lost and the NC/PLC interface signal:
DB31, ... DBX60.4/60.5 (referenced/synchronized 1 or 2)
is reset for the relevant measuring system.

References

For further information about control and servo parameter set, please refer to:

- Functions Manual - Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)
- Programming Manual, Job Planning; Section: Programmable servo parameter set

17.4.5 Intermediate gear

Application and functions

A configured intermediate gear can be used to adapt a variety of rotating tools. The intermediate gear on the tool side has a multiplicative effect on the motor/load gearbox.

It is set via machine data:

MD31066 $MA_DRIVE_AX_RATIO2_NUMERA (intermediate gear numerator)

MD31064 $MA_DRIVE_AX_RATIO2_DENOM (intermediate gear denominator)

An encoder on the tool for the intermediate gear
is configured with machine data:
MD31044 $MA_ENC_IS_DIRECT2 (encoder on intermediate gear)
.

A changed parameterization of this machine data can be activated with the "Activate machine data" function either with the aid of 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:

⚠ CAUTION

Engineering error

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 the "Activate machine data" function.

In the case of an installed tool with a transmission ratio of 2:1, a suitable intermediate gear is configured and is activated immediately in the part program with the command `NEWCONF`.

Program code

```
N05 $MA_DRIVE_AX_RATIO2-NUMERA[AX5] = 2
M10 $MA_DRIVE_AX_RATIO2-DENOM[AX5] = 1
N15 NEWCONF
```

Changeover

Changeover to a new transmission ratio is performed immediately with the aid of the "Activate machine data" function. From a technological viewpoint, the associated mechanical changeover process takes some time, since, in mechanical terms, a different intermediate gear with rotating tool is being installed.

Note

At zero speed, changeover is jerk-free. The user is therefore responsible for taking appropriate precautions.

Applications in which changeover takes place during motion and which require smoothed or soft speed transition can be handled using existing setpoint speed filters.

For further explanations regarding control engineering dependencies, see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 343)".

17.4.6 Nonacknowledged gear step change

Mode change

A gear stage change that has not been acknowledged cannot be interrupted by a change in operating mode (e.g. switchover to JOG).

The switchover is delayed by the maximum period entered in machine data:
MD10192 $MN_GEAR_CHANGE_WAIT_TIME
.

If the gear stage change is not acknowledged within this time, the NC will output an alarm:

Further events

Events that initiate reorganization will also wait until a gear stage change is completed.

The time entered in machine data:
MD10192 $MN_GEAR_CHANGE_WAIT_TIME
determines how long the control waits before executing the gear stage change.
If this time elapses without the gear stage change being completed, the NC responds with an alarm.

The following events have an analog response:

- User ASUP
- 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`
- Subprogram level abort, subprogram abort

Response after POWER ON

The active gear stage on the machine can be specified by the PLC after POWER ON and in the RESET state of the NC.

The NC will then select the appropriate parameter set
and check back the NC/PLC interface signals:
DB31, ... DBX82.0-82.2 (set gear stage A to C)
to the PLC.

17.4.7 Gear step change with oscillation mode

The spindle is in the oscillation mode if a new gear stage was defined using `M40` (automatic gear stage selection) - or `M41` to `M45`.

When oscillating, the direction of rotation of the spindle motor is continually reversed in short intervals. The oscillating motion of the motor helps to engage a new gear stage, for example. In principle, a new gear stage can also be engaged without oscillation.

The following oscillation types are possible:

- Oscillation controlled by the NC
- Oscillation via PLC
- Oscillation with FC 18
  References:
  Function Manual, Basic Functions; PLC Basic Program (P3)

Commissioning: Machine data

- MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear step change).
- MD35410 $MA_SPIND_OSCILL_ACCEL (acceleration when oscillating)
- MD35430 $MA_SPIND_OSCILL_START_DIR (start direction when oscillating)
- MD35400 $MA_SPIND_OSCILL_DES_VELO (oscillation speed)
- MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction)
- MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction)

Commissioning: NC/PLC interface signals

- DB31, ... DBX16.0 - 2 (actual gear stage)
- DB31, ... DBX16.3 (gear stage has been changed)
- DB31, ... DBX18.4 (oscillation controlled by the PLC)
- DB31, ... DBX18.5 (oscillation enable)
- DB31, ... DBX18.6 (setpoint direction of rotation, clockwise)
- DB31, ... DBX18.7 (setpoint direction of rotation, counterclockwise)
- DB31, ... DBX61.4 (spindle stationary)
- DB31, ... DBX82.0 - 2 (set gear stage)
- DB31, ... DBX82.3 (change over gear stage)
- DB31, ... DBX83.5 (spindle in setpoint range)
- DB31, ... DBX84.6 (active spindle mode: oscillating mode)

Description of the sequence: Gear stage change

Parameterization: MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, bit 0 = 1

Sequence:

- Deceleration of the spindle.
  The braking action corresponds to an M5 movement.
- NC/PLC interface signals NC → PLC:
  - DB31, ... DBX84.6 == 1 (oscillation mode)
  - DB31, ... DBX82.3 == 1 (changeover gear stage)
  - DB31, ... DBX82.0 - 2 == <set gear stage>
  - DB31, ... DBX61.5 == 0 (position controller active)
- The load gear can now "disengage".
- Oscillation can now be executed from the NC or the PLC (refer below).
- Program execution continues after the feedback signal from the PLC to the NC that the gear stage has been switched over and the new gear stage is active:
- DB31, ... DBX16.3 = 1 (gear stage has been switched over)
- DB31, ... DBX16.0 - 2 (actual gear stage)
- For measuring systems with indirect encoder (motor encoder) the referencing status is deleted:
  DB31, ... DBX60.4 / 5 = 0

Oscillation controlled by the NC

Requirement

- DB31, ... DBX18.5 = 1 (oscillation enable)

Status feedback signals

- DB31, ... DBX84.6 == 1 (oscillation mode)
- DB31, ... DBX82.3 == 1 (changeover gear stage)
- DB31, ... DBX82.0 - 2 == <set gear stage>
- DB31, ... DBX61.5 == 0 (position controller active)

The load gear can now "disengage".

Start oscillation

The PLC user program can now set the oscillation enable signal.

- DB31, ... DBX18.5 = 1 (oscillation enable)

The motor accelerates with the oscillation acceleration to the oscillation speed in the start direction for the direction of rotation dependent oscillation time:

- MD35410 $MA_SPIND_OSCILL_ACCEL (acceleration when oscillating)
- MD35430 $MA_SPIND_OSCILL_START_DIR (start direction when oscillating)
- MD35400 $MA_SPIND_OSCILL_DES_VELO (oscillation speed)
- MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction)
- MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction)

Reversing the direction of rotation

After the direction-dependent oscillation time has expired, the motor is braked with the oscillation deceleration and is accelerated in the opposite direction of rotation to the oscillation speed for the other oscillation time.

Oscillation controlled by the PLC

Requirement

- DB31, ... DBX18.4 = 1 (oscillation controlled by the PLC)
- DB31, ... DBX18.5 = 1 (oscillation enable)

Status feedback signals

- DB31, ... DBX84.6 == 1 (oscillation mode)
- DB31, ... DBX82.3 == 1 (changeover gear stage)
- DB31, ... DBX82.0 - 2 == <set gear stage>
- DB31, ... DBX61.5 == 0 (position controller active)

The load gear can now "disengage".

Start oscillation

The PLC user program can now set the oscillation enable signal.

- DB31, ... DBX18.7 = 1 (setpoint direction of rotation, counterclockwise) or DB31, ... DBX18.6 = 1 (setpoint direction of rotation, clockwise)

The motor accelerates with the oscillation acceleration to the oscillation speed in the specified setpoint direction of rotation:

- MD35410 $MA_SPIND_OSCILL_ACCEL (acceleration when oscillating)
- MD35400 $MA_SPIND_OSCILL_DES_VELO (oscillation speed)

Reversing the direction of rotation

The oscillation times and switching over the direction of rotation must be realized in the PLC user program.

End of oscillation mode

On termination of oscillation mode, the spindle returns to the 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 parameterized values of the actual gear stage.

Block change

Processing of the active NC program is stopped if the spindle was switched into the oscillation mode (DB31, ... DBX82.3 = 1 (switchover gear stage)).

Processing of the NC program is continued if the gear stage has been switched over (DB31, ... DBX16.3 == 1 (gear stage has been switched over).

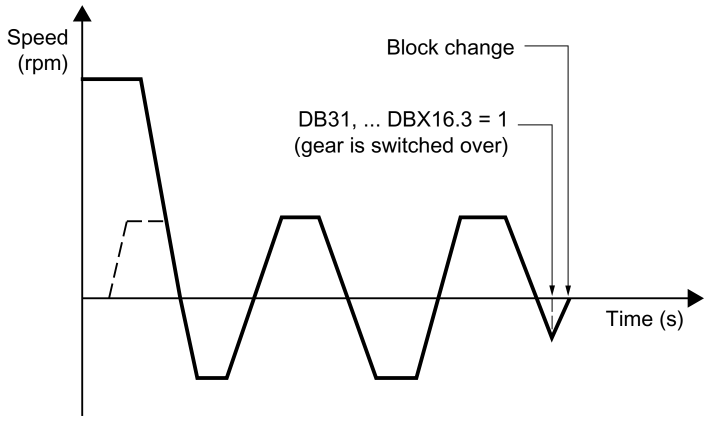
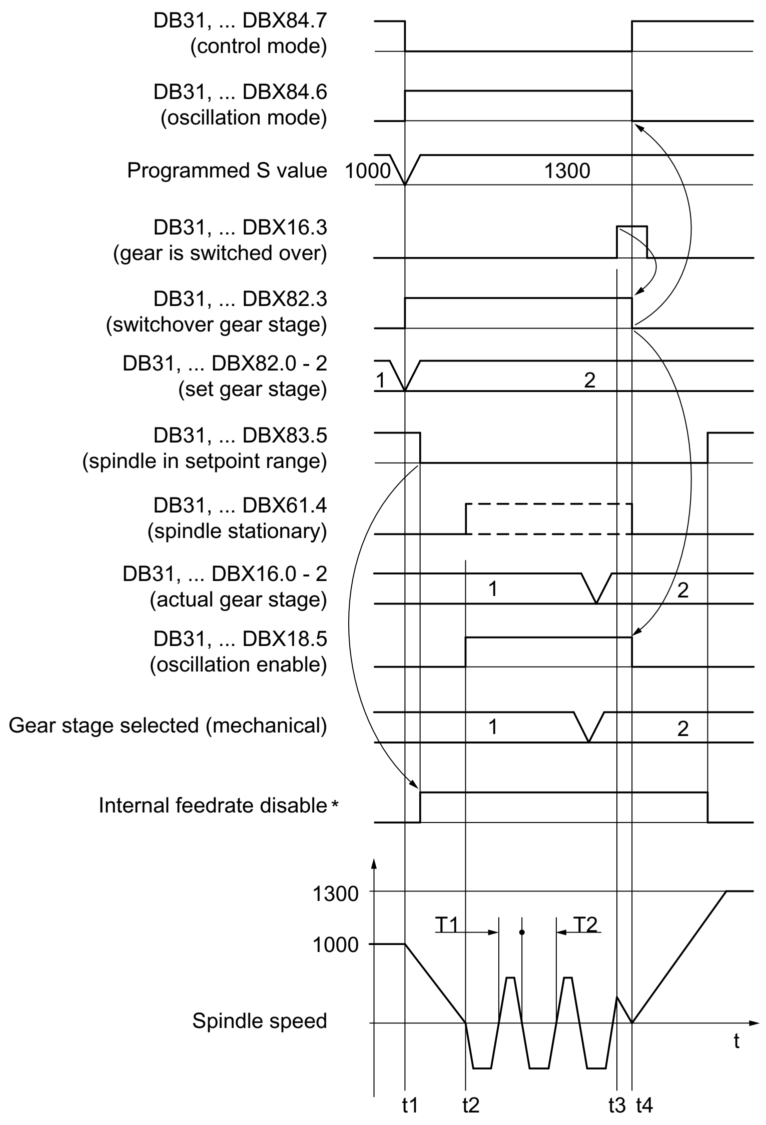


Figure 17-5 Block change following oscillation mode

Example of the signal flow



t_1: As a result of the newly programmed **S value S1300** in the NC program, the NC identifies the necessity to switch over the gear stage (1st → 2nd gear stage), and requests the enable from the PLC:

- NC → PLC: DB31, ... DBX82.3 = 1 (switchover gear stage)

The NC locks program processing (internal feed rate disable *)

t_2: The NC signals **spindle speed == 0** to the PLC:

- DB31, ... DBX61.4 = 1 (spindle stationary)

The PLC user program enables the oscillation mode.

- PLC → NC: DB31, ... DBX18.5 = 1 (oscillation enable)
  Note
  The oscillation enable must be set at the latest at time t_2.

The NC starts to oscillate.

t_3: The gear stage at the machine was switched over.

The PLC user program signals the switchover to the NC and deletes the oscillation mode enable:

- PLC → NC: DB31, ... DBX16.0 - 2 = 2 (actual gear stage)
- PLC → NC: DB31, ... DBX16.3 = 1 (gear stage has been switched over)
- PLC → NC: DB31, ... DBX18.5 = 0 (oscillation enable)

t_4: The NC exits the gear stage switchover by resetting the request to the PLC and the oscillation by switching over the spindle mode from oscillation mode to control mode:

- NC → PLC: DB31, ... DBX82.3 = 0 (switchover gear stage)
- DB31, ... DBX84.6 = 0 (active spindle mode: oscillating mode)
- DB31, ... DBX84.7 = 1 (active spindle mode: control mode)

The NC enables the next block for machining, and accelerates the spindle to the new S value S1300.

*: The internal feed disable is set if:

- Gear stage switchover as a result of programming the S value in the NC program **AND**
- A **machining**block is active (not rapid traverse `G0`)

Note

The internal feed disable is **not** set during a gear stage change from synchronized actions or in the case of specifications via the PLC with FC 18.

Figure 17-6 Gear stage change with stationary spindle

Supplementary conditions

- The NC/PLC interface signal does not have to be set in order to brake the spindle:
  DB31, ... DBX4.3 (spindle stop)
  The spindle is brought to a standstill in the control system when a gear stage switchover is requested.
- The gear stage switchover must always be exited by setting the NC/PLC interface signal:
  DB31, ... DBX16.3 = 1 (gear stage has been switched over)
- Oscillation motion in the NC is stopped when the NC/PLC interface signal is reset.
  DB31, ... DBX18.5 (oscillation enable)
  However, the spindle remains in the "Oscillation mode" mode.
- Spindle synchronization is lost for gear stage switchover and indirect measuring system (MD31050 $MA_ENC_IS_DIRECT = 0 motor encoder).
  Gear stage switchover ⇒ DB31, ... DBX60.4 / 5 = 0 (referenced/synchronized)
  The spindle is again automatically synchronized when the zero mark is passed the next time after a gear stage switchover.

17.4.8 Gear stage change at fixed position

Application and advantages

Machine tools increasingly use standardized spindle drives, firstly to save technological dead time on a gear stage change and secondly to gain the cost benefits of using standardized components.

The "Gear stage change at fixed position" function supports the "directed gear stage change" of load gearboxes that need to be activated in a different way than the NC. The gear stage change can in this case only be performed at a defined spindle position. An oscillation movement as required by conventional load gearboxes is thus no longer necessary.

Sequence for gear stage change at fixed position

The gear stage change at fixed position

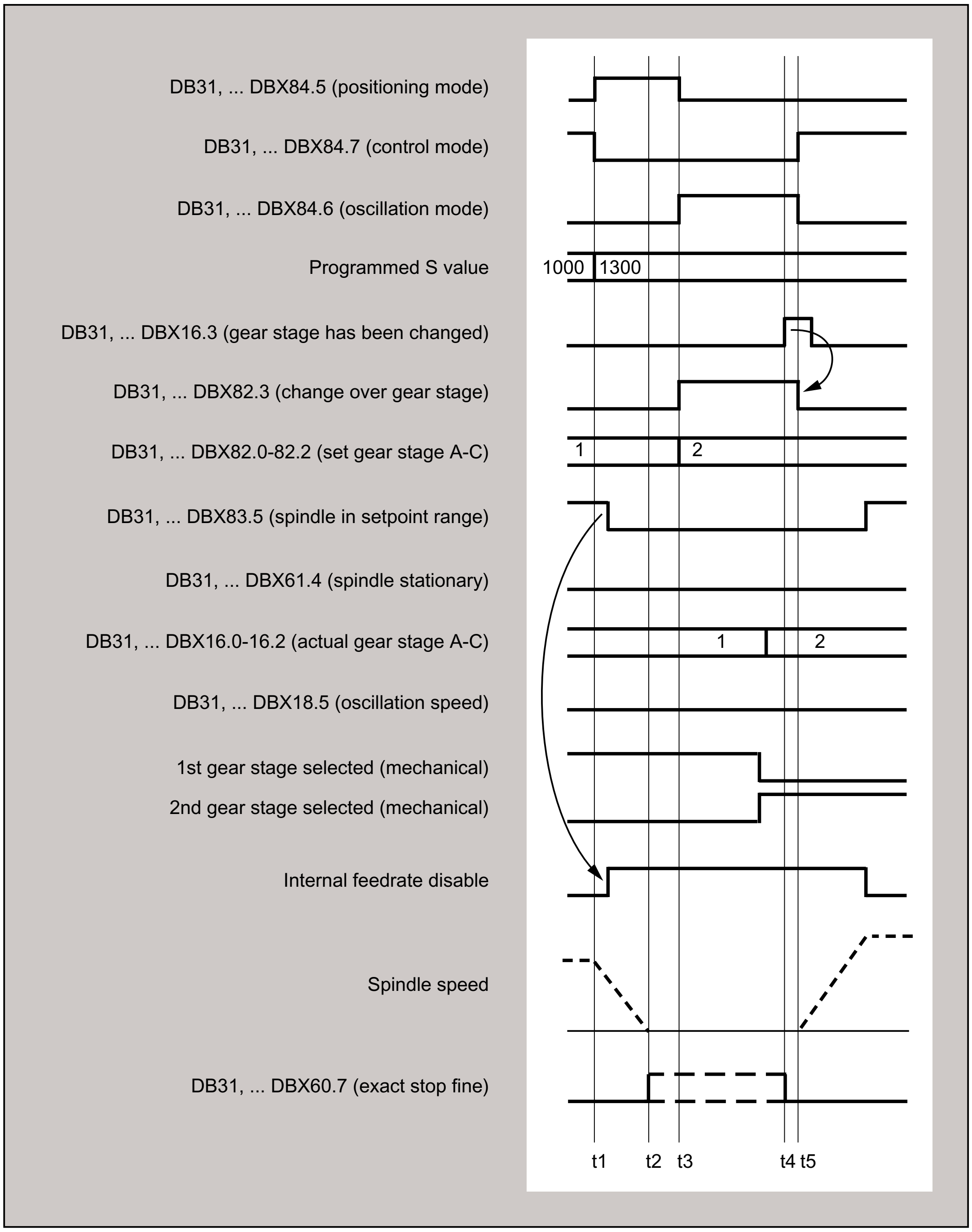
Machine data configuration:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2
runs the following sequence:

- Positioning of the spindle from standstill or movement to the position configured in machine data:
  MD35012 $MA_GEAR_STEP_CHANGE_POSITION.
  If the gear stage change is performed out of a movement, then the current direction of rotation is maintained. The spindle is in positioning mode during the positioning action.
  NC/PLC IS:
  DB31, ... DBX84.5 (positioning mode)
  is output.
  If no reference is available:
  DB31, ... DBX60.4/5 = 0
  or NC/PLC IS:
  DB31, ... DBX17.4/5 (resynchronize on positioning MS 1/2)
  is set, the positioning action is extended by the time it takes to find the zero mark.
- After reaching the gear stage change position configured in machine data:
  MD35012$MA_GEAR_STEP_CHANGE_POSITION
  the machine waits for the time in machine data:
  MD35310 $MA_SPIND_POSIT_DELAY_TIME
  before switching to oscillation mode,
  and the known gear stage change dialog starts.
- Output of NC/PLC interface signals:
  DB31, ... DBX84.6 (oscillation mode)
  DB31, ... DBX82.3 (change gear)
  DB31, ... DBX82.0-82.2 (set gear stage A to C).
- Position control is not disabled when an active measuring system with indirect encoder (motor encoder) is used:
  MD31040 $MA_ENC_IS_DIRECT = 0
  If a measuring system with a direct encoder (load encoder) is active, position control is deactivated:
  DB31, ... DBX61.5 = 0,
  because the induction flux to the load is interrupted and closed-loop position control is no longer possible.
- If position-controlled operation is not possible, it can be disabled by resetting "Controller enable":
  DB31, ... DBX2.1 = 0
  .

- Mechanical switchover of the gear stage on the machine.
  No oscillation movement is required from the drive.
  NC/PLC IS:
  DB31, ... DBX18.5 (oscillation enable)
  and
  DB31, ... DBX18.4 (oscillation via PLC)
  should **not** be set.
  In principle, oscillation movement is still possible at this point.
- Writing of NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  by the PLC.
- After signal:
  DB31, ... DBX16.3 (gear stage changed),
  the last movement to be active is continued, if available.
  For indirect encoders (motor encoders), the referencing status is cleared:
  DB31, ... DBX60.4/5 = 0.
  The spindle is in speed control mode and NC/PLC IS:
  DB31, ... DBX84.7 (open-loop control mode)
  is output.

GSC at fixed position

Typical time sequence for the gear stage change at fixed position:


DB31, ... DBX84.5 (positioning mode)
DB31, ... DBX84.7 (control mode)
DB31, ... DBX84.6 (oscillation mode)
Programmed S value
1000
1300
DB31, ... DBX16.3 (gear stage has been changed)
DB31, ... DBX82.3 (change over gear stage)
DB31, ... DBX82.0-82.2 (set gear stage A-C)
1
2
DB31, ... DBX83.5 (spindle in setpoint range)
DB31, ... DBX61.4 (spindle stationary)
DB31, ... DBX16.0-16.2 (actual gear stage A-C)
1
2
DB31, ... DBX18.5 (oscillation speed)
1st gear stage selected (mechanical)
2nd gear stage selected (mechanical)
Internal feedrate disable
Spindle speed
DB31, ... DBX60.7 (exact stop fine)
t1
t2
t3
t4
t5

t_1: With the programming of S1300, NC detects a new gear stage (second gear stage), sets IS DB31, ... DBX84.5 (positioning mode) and blocks processing for the next part program block (= internal feed disable*).

t_2: The spindle is stationary, and exact stop is signaled.

t_3: Gear stage change - wait time

t_4: The new gear stage is engaged. The PLC user transfers the new (actual) gear stage to the NC and sets IS DB31, ... DBX16.3 (gear is changed).

t_5: The NC then retracts IS DB31, ... DBX82.3 (change gear), releases the next part program block for processing, and accelerates the spindle to the new S value (S1300).

* : The internal feed disable is set if:

- The spindle gear stage change has been programmed via the part program **and**
- A processing block is activated (i.e. G0 is not active)

The internal feed disable is **not** set during a gear stage change from synchronized actions or in the case of specifications via the PLC with FC 18.

Figure 17-7 Gear stage change with stationary spindle

gear stage change position MD35012

The gear stage change position is defined in machine data:
MD35012 $MA_GEAR_STEP_CHANGE_POSITION
for each gear stage.

Gear stage change wait time MD35310

After the positioning action the machine waits for the time configured in machine data:
MD35310 $MA_SPIND_POSIT_DELAY_TIME
until gear change request:
DB31, ... DBX84.6 (oscillation mode)
DB31, ... DBX82.3 (change gear)
and
DB31, ... DBX82.0-82.2 (set gear stage A to C)
are output.

Position identifiers / position

The position is always approached via the shortest path (corresponds to DC).

If no reference is available and the spindle is in standstill
(e.g. after Power On), then the direction of travel is determined by the following machine data:
MD35350 $MA_SPIND_POSITIONING_DIR

If an adjustable gear stage change position is required, then this can be achieved by writing the machine data and by a subsequent "Activate machine data".
The change of the MD value can be achieved by the part program or HMI.

If the system is unable to reach the preset position, then alarm 22020 is signaled and the gear stage change dialog between NC 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 velocity is taken from the machine data which is configured depending on the gear stage:

MD35300 $MA_SPIND_POSCTRL_VELO

The NC/PLC interface signals "Spindle override"/"Feedrate override" at
DB31, ... DBX17.0=0: DB31, ... DBB19)
as well as:
DB31, ... DBX17.0=1: DB31, ... DBB0
are effective as normal for positioning.
The positioning speed can be changed proportionally through the program statement `OVRA[Sn]`.

Note

`OVRA[Sn]` is valid modally. After the gear stage change, a value appropriate for the machining should be reset.

The part program statement `FA[Sn]` does not change the positioning speed during gear stage change.

Acceleration

The acceleration values are determined by the machine data which is configured depending on the gear stage:

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL

and

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

The acceleration can be changed proportionally by programming `ACC[Sn]`.

Note

`ACC[Sn]` is valid modally. After the gear stage change, a value appropriate for the machining should be reset.

Speed-dependent acceleration

The "knee-shaped acceleration characteristic" is effective as in positioning with `SPOS` or FC18.

Jerk

It is currently not possible to limit the change in acceleration.

End of positioning

The transition between the end of the positioning action (DB31, ... DBX84.5)
and the start of oscillation mode (DB31, ... DBX84.6) is defined on
reaching "Exact stop fine" (DB31, ... DB60.7) and the time value entered in machine data:
MD3510 $MA_SPIND_POSIT_DELAY_TIME
.

The determination of the transition condition has an effect firstly on the gear stage change time and secondly on the accuracy of the approach to the preset gear stage change position.

Block change

The block change is stopped and the machining blocks are not started until the gear stage has been changed by the PLC (DB31, ... DBX16.3).

End of gear stage change

Once the gear stage change has been completed, the spindle returns to open-loop control mode and will automatically change to the closed-loop control mode defined by `SPCON` or `SPCOF`.

All gear-specific limit values (min./max. speed of gear stage, etc.) correspond to the check-back values of the actual gear stage.

Supplementary conditions

- The spindle must have at least one measuring system.
- Position-controlled operation must be possible and must have been activated.
- Generally, it must be possible to execute `SPOS` from the part program, from a synchronized action or via FC18: "Start spindle positioning" without errors.

Unless all requirements can be met, the function described cannot be used successfully.

Activation

The function of gear stage change at fixed position
is activated by the configuration:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2

17.4.9 Configurable gear step in M70

Technical background

In some machines the spindle needs to be in a particular gear stage during axis mode.

Possible reasons:

- Only one optimization (K_V, feedforward control, filter) to suit a gear stage can be found in the servo parameter set for axis mode (index 0). The machine data for this parameter set should not be rewritten.
- There is only one mechanical transmission ratio which, unlike the others, possesses little or no backlash compensation. The spindle can only follow a path motion or transformations (e.g. TRANSMIT) together with other axes in this gear stage.

Function

If the function is activated, a predefined gear stage is loaded automatically during transition to axis mode.

The gear stage change is integrated into the M70 process and occurs after spindle deceleration and before the loading of the servo parameter set with index 0 (note MD35590 $MA_PARAMSET_CHANGE_ENABLE!).

The typical dialog between NC and PLC which occurs during gear stage changes is executed in a similar way to programmed gear stage changes (M41 ... M45) performed.

Requirements

Gear stage changes during transition to axis mode require general enabling of the gear stage change via the machine data:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear stage change).

MD35090 $MA_NUM_GEAR_STEPS (number of gear stages set up)

Activation/deactivation

The function is activated/deactivated via the machine data:

MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE (gear stage for axis mode in M70)

Value	Meaning
0	No implicit gear stage change occurs in M70. The current gear stage is retained (default setting!).
1 ... 5	A gear stage change to gear stage 1 ... 5 occurs during the processing of M70.

Supplementary conditions

Gear stage change at fixed position (MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2)

The "gear stage change at fixed position" function is supported. The sequence in M70 is then extended by the time it takes to position the spindle. The position is approached at the current gear stage.

Transition to axis mode without programming M70

The control system detects the transition to axis mode automatically from the program sequence (see "Implicit transition to axis mode (Page 1299)") and generates the requisite M70 sequence, including the gear stage change, within the control system.

Transition to axis mode with FC 18

Implicit gear stage change is not supported in transition to axis mode with the FC 18 ("Start axis"). This requires the right gear stage to be engaged by the PLC application before switching to axis mode. The gear stage change is also possible with the FC 18 ("Start gear stage change").

Change from axis mode to spindle mode

When changing from axis mode to spindle mode, the gear stage loaded with M70 remains activated. The gear stage which is activated in spindle mode prior to M70 is not automatically loaded again. The servo parameter set is changed from parameter set 1 (index 0) to parameter sets 2 ... 6 (index 1 ... 5) to suit the gear stage (with MD35590 $MA_PARAMSET_CHANGE_ENABLE < 2).

Example

Gear stage 4 should be loaded in the case of spindle transition to axis mode.

Configuration: MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE[<spindle name>] = 4

Program code	Comment
`N05 M3 S1000`	
`N10 G1 X100 F1000`	
`N15 M70`	`; Gear stage 4 is loaded.`
`N20 POS[C]=77`	
`N25 ...`	

Note

MD35014 can be changed by the "Activate machine data" function. Thus, the gear stage being loaded can still be changed in the part program before transition to axis mode, if necessary.

17.4.10 Suppression of the gear stage change for DryRun, program test and SERUPRO

Function

For test feed rate (DryRun), program test and SERUPRO, normally, a gear stage change is not required. This is the reason that it can be suppressed for these functions. The corresponding configuration is realized with bits 0 ... 2 in machine data:

MD35035 $MA_SPIND_FUNCTION_MASK

Dry run feedrate (DryRun)	
Bit 0 = 0	When the DryRun function is active - for part program blocks - gear stages are changed with `M40`, `M41` to `M45`, or programming via FC18 and synchronized actions.
Bit 0 = 1	When the DryRun function is active - for part program blocks - a gear stage change is suppressed with `M40`, `M41` to `M45`, programming via FC18 and synchronized actions.
Program test and SERUPRO	
Bit 1 = 0	For active program test / SERUPRO function - for part program blocks - gear stages are selected with `M40`, `M41` to `M45`, programming via FC18 and synchronized actions.

Bit 1 = 1	For active program test / SERUPRO function - for part program blocks - a gear stage change is suppressed with `M40`, `M41` to `M45`, programming via FC18 and synchronized actions.
DryRun, program testing and SERUPRO	
Bit 2 = 0	Gear stage change for programmed gear stage is **not** performed subsequently on REPOS after deselection of functions DryRun, Program Test and SERUPRO.
Bit 2 = 1	Gear stage change for programmed gear stage is performed after deselection of functions DryRun and SERUPRO if possible.

Sequence

If a gear stage change is suppressed, if necessary, the interpolator will limit the programmed spindle speed to the permissible speed range of the active gear stage.

NC/PLC interface signals DB31, ... DBX83.2 (setpoint speed increased) and DB31, ... DBX83.1 (setpoint speed limited) generated as a result of this limit are suppressed.

Monitoring by the PLC program is not necessary during DryRun and in dry run feedrate.

When the gear stage change is suppressed, no new gear stage setpoint
(DB31,... DBX82.0-82.2) is output to the PLC.

The gear stage change request DB31, ... DBX82.3 (change gear) is also suppressed.

This ensures that no gear stage change information has to be processed by the PLC program.

Determining the last active gear stage

System variable $P_GEAR returns the gear stage programmed in the part program (which may not have been output to the PLC).

System variable $AC_SGEAR can be used to read the last active gear stage from the part program, synchronized actions and at the user interface.

Behavior after deselection

The DryRun function can be deselected within a running part program. Once it has been deselected, the correct gear stage requested by the part program must be identified and selected.

It cannot be assured that the remainder of the part program will run without errors until the correct gear stage has been activated. Any necessary gear stage change is performed in the system REPOS started on deselection if the spindle is in speed control mode. A complete gear stage change dialog takes place with the PLC and the last programmed gear stage is activated.

If, for REPOS, there is a mismatch between the gear stage programmed in the part program and the actual gear stage supplied via the NC/PLC interface, then no gear stage change takes place.

The same applies to the SERUPRO function.

Further explanations regarding the block search function SERUPRO, see:
References:

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

Supplementary conditions

If the gear stage change is suppressed, the output spindle speed moves within the speed range specified by the current gear stage.

The following restrictions apply to the subsequent activation of a gear stage change with REPOS:

- The gear stage change is not activated subsequently if the spindle in the deselection or target block is a command spindle (synchronized action) or PLC spindle (FC18).
- If the gear stage cannot be activated because the spindle is in position or axis mode or a link is active, alarm 22011"Channel%1 block%3 spindle2% Change to programmed gear stage not possible" is signaled.

Example

Gear stage change in DryRun

```
; Activate 1st gear stage for output state
N00 M3 S1000 M41                    ; 1st gear stage is selected
M0                                  ; Part program stop

; PI service: Activate dry run feedrate (DryRun)
                                    ; (Configuration)
N10 M42                             ; 2nd gear stage requested, no gear stage change
                                    takes place
N11 G0 X0 Y0 Z0                     ; Position axes
N12 M0                              ; Part program stop

; PI service: Deactivate dry run feedrate (DryRun)
                                    ; REORG and REPOS are performed
                                    ; now the gear stage change to the 2nd gear
                                    stage takes place
N20 G1 Z100 F1000
...
N99 M30                             ; Part program end
```

17.5 Additional adaptations to the spindle functionality that can be configured

The following spindle functions can be activated using the machine data:

MD35035 $MA_SPIND_FUNCTION_MASK, <bit> = <value>

<Bit>	<Value>	Meaning
0 ... 2	1	Activation: Gear stage change behavior for test feedrate (DryRun), program test and SERUPRO See "Suppression of the gear stage change for DryRun, program test and SERUPRO (Page 1347)".
4	1	Spindle speed `S...`, programmed in the NC program, is taken after SD43200 $SA_SPIND_S. This also applies to speed setpoint signals via the PLC user program with FC18 and synchronized actions. See "Function (Page 1360)".
5	0	When traversing the spindle in the JOG mode, the currently valid speed setpoint applies. See SD41200 $SN_JOGSPIND_SET_VELO
	1	When traversing the spindle in the JOG mode, the speed parameterized in SD43200 $SA_SPIND_S is effective.
8	1	Cutting velocity `S...`, programmed in the NC program is taken after SD43202 $SA_SPIND_CONSTCUT_S. This also applies to cutting velocity setpoint signals via the PLC user program with FC18. See "Function (Page 1360)".
10	0	SD 43206 $SA_SPIND_SPEED_TYPE is not changed by the NC program or channel settings.
	1	If the spindle is a master spindle, then the feedrate type (active value of the 15th G group) is taken after SD43206 $SA_SPIND_SPEED_TYPE. If the spindle is not a master spindle, then SD43206 $SA_SPIND_SPEED_TYPE remains unchanged. See "Function (Page 1360)".
12	Effectiveness of the spindle override for a zero mark search, M19, SPOS, SPOSA = 0:	
	0	The spindle override is**not** active.
	1	The spindle override is active.
19	Response regarding an implicit auxiliary function output M19:	
	0	**AND** MD20850 $MC_SPOS_TO_VDI == 0: For `SPOS` and `SPOSA`, auxiliary function M19 is not output to the PLC.
	1	For `SPOS` and `SPOSA`, auxiliary function M19 is output to the PLC. The address extension corresponds to the spindle number. See "Positioning mode (Page 1279)"
20	Response regarding an implicit auxiliary function output M70:	
	0	An implicit auxiliary function M70 is not output to the PLC. **Note**: A programmed auxiliary function M70 is always output to the PLC.
	1	When transitioning from spindle to axis operation, auxiliary function M70 is output to the PLC. The address extension corresponds to the spindle number. See "Implicit transition to axis mode (Page 1299)".

<Bit>	<Val-ue>	Meaning
22	**Deactivating** the axis-specific NC/PLC interface signal **DB31, ... DBX17.6** (invert M3/M4) to reverse the direction of rotation of the spindle for tapping without compensating check (**G331**, **G332**).	
	0	Interface signal DB31, ... DBX17.6 **active**. **Note** • **Standard case**: The value of the interface signal is determined when the NC program starts. When the interface signal subsequently changes, this has no effect within the NC program. • **Special case**: If a change to the interface signal must be taken into account while the NC program is being executed, then this must be implemented as part of the application engineering. See Chapter "Special case: Direction of rotation reversal via NC/PLC interface signal in the NC program (Page 1307)".
	1	Interface signal DB31, ... DBX17.6 **not active**.

17.6 Selectable spindles

Function

Regarding channel spindles, the "Switchable spindles" function allows general NC programs to be written, which can be used in different channels. The logical spindle number used in the NC program is converted to the physical spindle number using a spindle number converter. This means for the same spindle number programmed in the NC program, different physical spindles (machine axes) can be traversed in different channels.

The physical spindles loaded or unloaded by "axis replacement" no longer have to be specified explicitly in the part program.

Machine data

Physical spindle number

Each spindle must be assigned to a machine axes using a unique configurable number, the physical spindle number:

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[<machine axis>] = <physical spindle number>

Activation of the spindle number converter

The spindle number converter is activated on a channel-for-channel basis using machine data:

MD20092 $MC_SPIND_ASSIGN_TAB_ENABLE = TRUE

Setting data

Spindle number converter

To parameterize the channel-specific spindle number converter, each logical spindle number or programmed address extension - with which a spindle (channel spindle) is addressed in the NC program - is assigned a physical spindle number (machine axis).

SD42800 $SC_SPIND_ASSIGN_TAB[<logical spindle number>] = <physical spindle number>

Additional information:

- The logical spindle number of the current master spindle can be read from SD42800 $SC_SPIND_ASSIGN_TAB[**0**] It is not permissible to write to this data. The data is updated when programming `SETMS`.
  Unused spindles are assigned the value 0 in SD42800.
- When an auxiliary function is output, the physical spindle number is output as address extension.
- The spindle number converter is active when programming spindles in NC programs and synchronized actions.
- The spindle number converter is **not** active when spindles are entered at the NC/PLC interface using function block FC18, as in this case the physical spindles are directly addressed.
- System variables affected by the spindle conversion are: $P_S, $P_SDIR, $P_SMODE, $P_GWPS, $AC_SDIR, §AC_SMODE, $AC_MSNUM, $AA_S.

Supplementary conditions

- Switchable channel spindles are **not** a substitute for the Axis replacement function.
- Spindle conversion can be modified by writing to setting data (SD42800 $SC_SPIND_ASSIGN_TAB) via the NC program, PLC user program or PI service. The change takes effect immediately.
- In the spindle number converter, only physical spindles can be used that are assigned to the channel (MD20070 $MC_AXCONF_MACHAX_USED).
  If physical spindles are specified in the spindle number converter, which are presently active in another channel, then depending on the setting in MD30552 $MA_AUTO_GET_TYPE, either the physical spindle is requested for the channel, or alarm Alarm 16105 "Assigned spindle does not exist" is displayed.
- If SD42800 $SC_SPIND_ASSIGN_TAB[<n>] is specified by the PLC or from HMI, then the channel whose table is changed should be in Reset status or the spindle to be changed should not be used in the running part program.
  A synchronized response can be achieved by means of a `STOPRE` preprocessor stop.
- Converting a logical to several physical spindles is not locked. However, with the display of the logical spindle in the user interface, there are ambiguities corresponding to the conversion table.

Example

Assignment, spindle number and machine axis:

```
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX4] = 1
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX5] = 2
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX6] = 3
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX7] = 5
```

Accepting a machine axis in a channel:

```
MD20070 $MC_AXCONF_MACHAX_USED[0] = 4
MD20070 $MC_AXCONF_MACHAX_USED[1] = 5
MD20070 $MC_AXCONF_MACHAX_USED[2] = 6
MD20070 $MC_AXCONF_MACHAX_USED[3] = 7
```

Specifying the master spindle:

```
MD20090 $MC_SPIND_DEF_MASTER_SPIND = 1
```

Spindle number converter

`MD20092 $MC_SPIND_ASSIGN_TAB_EN-ABLE=1`	`; Activate spindle number converter`
`SD42800 $SC_SPIND_AS-SIGN_TAB[0]=1`	`; Master spindle as configured`
`SD42800 $SC_SPIND_AS-SIGN_TAB[1]=1`	`; Basic setting of the table`
`SD42800 $SC_SPIND_AS-SIGN_TAB[2]=2`	
`SD42800 $SC_SPIND_AS-SIGN_TAB[3]=3`	
`SD42800 $SC_SPIND_AS-SIGN_TAB[4]=0`	`; Logical spindle not assigned`
`; Address extension is 1 output: M1=3 S1=1000`	
`; Spindle 1 (physical master spindle) rotates`	
`M3 S1000`	
`...`	
`; Assignment of logical spindle 1 to physical spindle 5`	
`SD42800 $SC_SPIND_AS-SIGN_TAB[1]=5`	
`; Assignment of logical spindle 2 to physical spindle 3`	
`SD42800 $SC_SPIND_AS-SIGN_TAB[2]=3`	

```
; Notice: physical spindle 3 has now been assigned twice.
; When programming logical spindles 2 and 3, physical spindle 3
; is always addressed.
; In the basic machine displays, both spindles rotate.

SETMS(2)                          ; SD42800 $SC_SPIND_ASSIGN_TAB[0] = 2
; Master spindle = address extension = 2, converted spindle number M3 = 5 is output
; The physical spindle configured with number "3" stops.
M5
GET(S4)                           ; Alarm 16105: logical spindle 4
                                  ; cannot be converted
RELEASE(S1)                       ; Channel spindle 1 = physical spindle 5
                                  ; is released.
```

17.7 Programming

17.7.1 Programming from the part program

Programming statements

Statement	Description	
`SETMS`:	Master spindle is the spindle specified in the following machine data: MD20090 $MC_SPIND_DEF_MASTER_SPIND (position of deletion of the master spindle in the channel)	
`SETMS(<n>)`:	The spindle with the number <n> is the master spindle (may differ from the initial setting: MD20090 $MC_SPIND_DEF_MASTER_SPIND). The master spindle must be defined for the following functions:	
	• `G95`:	Revolutional feedrate
	• `G96 S.../G961 S...`:	Constant cutting rate in m/min or feet/min
	• `G97/G971`:	Cancel `G96/G961` and freeze last spindle speed
	• `G63`:	Tapping with compensating chuck
	• `G33/G34/G35`:	Thread cutting
	• `G331/G332`:	Tapping without compensating chuck
	• `G4 S...`:	Dwell time in spindle revolutions
	• Programming `M3`, `M4`, `M5`, `S`, `SVC`, `SPOS`, `M19`, `SPOSA`, `M40`, `M41` to `M45`, and `WAITS` without entering the spindle number	
	The current master spindle setting can be retained via RESET / part program end and part program start. The setting is done via the machine data: • MD20110 $MC_RESET_MODE_MASK • MD20112 $MC_START_MODE_MASK	
`M3`: `M<n>=3`:	Clockwise spindle rotation for the master spindle Clockwise spindle rotation for spindle number <n>	
`M4`: `M<n>=4`:	Counter-clockwise spindle rotation for the master spindle Counter-clockwise spindle rotation for spindle number <n>	
`M5`: `M<n>=5`:	Spindle stop without orientation for the master spindle Spindle stop without orientation for spindle number <n>	
`S...`: `S<n>=...`:	Spindle speed in rpm for the master spindle Spindle speed in rpm for spindle number <n>	
`SVC=...`: `SVC[<n>]=...`:	Cutting rate in m/min or feet/min for the master spindle Cutting rate in m/min or feet/min for the spindle <n>	
`SPOS=...`: `SPOS[<n>]=...`:	Spindle positioning for the master spindle Spindle positioning for spindle number <n> The block change is only performed when the spindle is in position.	

Statement	Description

`SPOSA=...:`
`SPOSA[<n>]=...:`
	Spindle positioning for the master spindle
	Spindle positioning for spindle number <n>
	The block change is executed immediately. Spindle positioning continues, regardless of further part program processing, until the spindle has reached its position.

`SPOS=DC(...):`
`SPOS[<n>]=DC(...):`
`SPOSA=DC(...):`
`SPOSA[<n>]=DC(...):`
	The direction of motion is retained for positioning while in motion and the position approached. When positioning from standstill, the position is approached via the shortest path.

`SPOS=ACN(...):`
`SPOS[<n>]=ACN(...):`
`SPOSA=ACN(...):`
`SPOSA[<n>]=ACN(...):`
	The position is always approached with negative direction of motion. If necessary, the direction of motion is inverted prior to positioning.

`SPOS=ACP(...):`
`SPOS[<n>]=ACP(...):`
`SPOSA=ACP(...):`
`SPOSA[<n>]=ACP(...):`
	The position is always approached with positive direction of motion.
	If necessary, the direction of motion is inverted prior to positioning.

`SPOS=IC(...):`
`SPOS[<n>]=IC(...):`
`SPOSA=IC(...):`
`SPOSA[<n>]=IC(...):`
	The travel path is specified. The direction of travel is determined from the sign in front of the travel path. If the spindle is in motion, the direction of travel is inverted as necessary to allow traversing in the programmed direction.
	If the zero mark is crossed during traversing, the spindle is automatically synchronized with the zero mark if no reference is available or if a new one has been requested via an interface signal.

`M19:`
`M[<n>]=19:`
	Positioning the master spindle to the position in SD43240
	Positioning spindle number <n> to the position in SD43240
	The block change is only performed when the spindle is in position.

`M70:`
`M<n>=70:`
	Bring spindle to standstill and activate position control, select zero parameter set, activate axis mode
	for the master spindle
	for spindle number <n>

`SPCON:`
`SPCON(<n>):`
`SPCON(<n>,<m>):`
	Spindle position control ON
	for the master spindle
	for spindle number <n>
	for spindle numbers <n> and <m>

`PCOF:`
`SPCOF(<n>):`
`SPCOF(<n>,<m>):`
	Spindle position control OFF, activate speed control mode
	for the master spindle
	for spindle number <n>
	for spindle numbers <n> and <m>

`FPRAON(S<n>):`	Revolutional feedrate for spindle <n> ON, derived from the master spindle
`FPRAON(S<n>,S<m>):`	Revolutional feedrate for spindle <n> ON, derived from spindle <m> The revolutional feedrate value must be specified with `FA[S<m>]`.
`FPRAOF(S<n>):`	Revolutional feedrate for spindle <n> OFF
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

Statement	Description
	Programmable minimum spindle speed limitation
G25 S...:	for the master spindle
G25 S<n>:	for spindle number <n>
	Programmable maximum spindle speed limitation
G26 S...:	for the master spindle
G26 S<n>:	for spindle number <n>
	Programmable maximum spindle speed limitation with G96, G961, G97
LIMS=...:	for the master spindle
LIMS[<n>]=...:	for spindle number <n>
VELOLIM[<spindle>]=...:	Programmable limiting of the configured gear stage dependent maximum speed Using machine data (MD30455 $MA_MISC_FUNCTION_MASK, bit 6), when programming in the part program, it can be set as to whether VELOLIM is effective independent of whether used as spindle or axis (bit 6 = 1) - or is able to be programmed separately for each operating mode (bit 6 = 0). If they are to be separately effective, then the selection is made using the name when programming: • Spindle name S<n> for spindle operating modes • Axis name, e.g. "C", for axis operation The correction value refers to: • Spindles in axis operation (if MD30455 Bit 6 = 0): To the configured maximum axis velocity (MD32000 $MA_MAX_AX_VELO). • Spindles in spindle or axis operation (if MD30455 bit 6 = 1): To the maximum speed of the active gear unit stage (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]) For further explanations about the programming of VELOIM, see: **References:** Programming Manual, Work Preparation
WAITS:	Synchronization command for master spindle The subsequent blocks are not processed until the spindle programmed in a previous NC block with SPOSA has reached its position (with exact stop fine). WAITS after M5: Wait until the spindle is stationary. WAITS after M3/M4: Wait until the spindle reaches its setpoint speed.
WAITS(<n>):	Synchronization command for spindle number <n>
WAITS(<n>,<m>):	Synchronization command for spindle numbers <n> and <m>
FA[S<n>]:	Programming of positioning speed (axial feed) for spindle <n> in [deg/min] With FA[S<n>]=0, the configured value takes effect once more: MD35300 $MA_SPIND_POSCTRL_VELO
OVRA[S<n>]:	Programming of the axial override value for spindle <n> in [%]
ACC[S<n>]:	Programming of the axial acceleration capacity for spindle <n> in [%]

Statement	Description
`SPI(<n>)`:	With `SPI(<n>)` a spindle number is converted into the data type AXIS according to machine data MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[ ] . `SPI` is used if axis functions are to be programmed with the spindle. The following commands are possible with `SPI`: • Frame commands: – `CTRANS()` – `CFINE()` – `CMIRROR()` – `CSCALE()` • Velocity and acceleration values for following spindles: – `FA[SPI(<n>)]` – `ACC[SPI(<n>)]` – `OVRA[SPI(<n>)]` • System variables: – `$P_PFRAME[SPI(<n>),TR]=<value>` – `$P_PFRAME= CTRANS(X,<axis value>,Y,<axis value>,SPI(<n>),<axis value>)` – `$P_PFRAME= CSCALE(X,<scale>,Y,<scale>,SPI(<n>),<scale>)` – `$P_PFRAME=CMIRROR(S<n>,Y,Z)` – `$P_UBFR=CTRANS(A,10) : CFINE (19,0.1)` For further explanations about the programming of `SPI`, see: **References:** Programming Manual, Work Preparation.
`M40`: `M<n>=40`:	Automatic gear stage selection for the master spindle Automatic gear stage selection for spindle number **<n>**
`M41` to `M45`: `M<n>=41` to `M<n>=45`:	Select gear stage 1 to 5 for the master spindle Select gear stage 1 to 5 for spindle number **<n>**.

Note

M functions M3, M4, M5, and M70 are not output in DB21, ... DBB194 and DBB202 if a spindle is configured in a channel. These M functions are offered as extended M functions in DB21, ... DBB68 ff. and in the relevant axis DBs, DB31, ... DBB86 ff.

References

More detailed explanations for programming the spindle can be found in:

- Programming Manual, Fundamentals

17.7.2 Programming via synchronized actions

M functions M40 to M45 can also be programmed in synchronized actions.

Please note:

- The programming of M40 ... M45 in the part program has no effect on the current status of the automatic gear step change of synchronized actions, and vice versa.
- When programming S values with M40, automatic gear step change is effective separately for synchronized actions and the part program.
- M40 is deactivated after POWER ON.
  The gear step is not adjusted if an S value is specified from a synchronized action.
- An M40 command programmed using synchronized actions always remains active for synchronized actions (modal) and is not reset on reset.
- M41 ... M45 selects first to fifth gear steps in accordance with the programming in the part program.
  An axis replacement is necessary in order to run the function.
  Once the gear step change has been performed, the spindle status is neutral (same response to M3, M4, M5 programming).

References

For further explanations regarding the programming of the spindle as well as spindle movements from synchronized actions, refer to:

- Programming Manual, Job Planning
- Function Manual, Synchronized Actions

17.7.3 Programming spindle controls via PLC with FC18 - only 840D sl

When the PLC specifies the direction of rotation and speed using FC18, the NC can determine and select a gear stage 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 stage selection.

References

More detailed explanations regarding the programming of spindle controls by PLC with FC18 can be found in:

- Function Manual, Basic Functions, Basic PLC Program (P3)

17.7.4 Programming using NC/PLC interface signals

17.7.4.1 Function

Note

The function is only available when using SINUMERIK Operate!

The following jobs/tasks can be issued to a spindle from the PLC user program using the job interface DB31, … DBX30.0 - 4:

- Spindle stop
- Spindle start, clockwise rotation
- Spindle start, counter-clockwise rotation
- Select gear stage
- Spindle-start positioning

Job ID

At the NC side, a spindle job is identified as a result of a positive **edge change** of the appropriate interface signal (0 → 1).

Requirements

At the instant of the spindle job, the following preconditions must be satisfied regarding the channel, program and spindle states.

- Channel state:
  – DB21, ... DBX35.6 == 1 (channel state "interrupted") **OR**
  – DB21, … DBX35.7 == 1 (channel state "reset")
- Program state:
  – DB21, ... DBX35.3 = 1 (program state "interrupted") **OR**
  – DB21, ... DBX35.4 = 1 (program state "interrupted")
- Spindle state
  – ("Channel axis" **OR** "Neutral axis") **AND NO**
  – (positioning motion using FC18 **OR** synchronized action)

Note

Response in the event of an error

In the event of an error, i.e. if a job signal is set during an invalid state, then the corresponding job is ignored **without any feedback signal**.

Channel assignment

A spindle job is processed in that particular channel, which is assigned to the spindle at the instant of the job.

The channel, which is assigned to the spindle, can be determined using the following NC/PLC interface signal:

DB31, ... DBX68.0 - 3 (channel assignment of the NC axis/spindle)

Mode change

The definitions made within the context of the spindle jobs are kept beyond any mode change.

Program start

When an NC program starts in the channel, which is assigned to the spindle, then the actual spindle definitions are kept. Spindle definitions can be changed using NC program commands or using synchronized actions.

See also

M40: Automatic gear stage selection for speeds outside the configured switching thresholds (Page 1375)

G2: Velocities, setpoint / actual value systems, closed-loop control (Page 343)

17.7.4.2 Commissioning: Machine data

Automatically accepting spindle definitions in setting data

Using the following machine data settings, spindle definitions are transferred from NC programs, synchronized actions and the spindle control by the PLC (FC18) into the corresponding setting data:
MD35035 $MA_SPIND_FUNCTION_MASK, bit x = <value>

Bit	Value	Meaning
4	1	A **speed setpoint** is entered via NC program, synchronized actions or FC18 is transferred into the following spindle-specific setting data: **SD43200** $SA_SPIND_S[<spindle>] **Note** S values that are not programmed speed values are not transferred. e.g.: • S value at constant cutting speed (G96, G961) • S value for rotation-related dwell time (G4)
	0	**No** speed setpoints are transferred into the setting data.

Bit	Value	Meaning
8	1	A **cutting rate** is entered as setpoint in the following spindle-specific setting data via the NC program, synchronized actions or FC18 if the "Constant cutting rate" function is active (G96, G961, G962): **SD43202** $SA_SPIND_CONSTCUT_S[<spindle>] **Note** S values that are not programmed cutting rate values are not transferred. e.g.: • S value for rotation-related dwell time (G4)
	0	**No** cutting rate setpoints are transferred into the setting data.
10	1	For the **master spindle** the **feedrate type** is transferred into the following setting data: **SD43206** $SA_SPIND_SPEED_TYPE[<spindle>]
	0	**No** feedrate type definitions for the master spindle are transferred into the setting data.

With the spindle definitions transferred into the setting data, the spindle is traversed to execute the spindle jobs.

17.7.4.3 Commissioning: NC/PLC interface signals

Job interface

- DB31, … DBX30.0 (spindle stop)
- DB31, … DBX30.1 (spindle start, clockwise)
- DB31, … DBX30.2 (spindle start, counter-clockwise)
- DB31, ... DBX30.3 (select gear stage)
- DB31, … DBX30.4 (spindle positioning start)

Relevant state signals

- DB21, ... DBX35.3 (program state "interrupted")
- DB21, ... DBX35.4 (program state "canceled")
- DB21, ... DBX35.7 (channel state "reset")
- DB21, ... DBX35.6 (channel state "interrupted")
- DB31, ... DBX68.0 - 3 (channel assignment of the NC axis/spindle)
- DB31, ... DBX 84.0 (constant cutting rate)

References:

A detailed description of the NC/PLC interface signals can be found in the NC Variables and Interface Signals List Manual.

17.7.4.4 Speed setpoint (SD43200)

When starting a spindle using a spindle job, then the spindle-specific value entered in the setting data becomes active as spindle speed:

SD43200 $SA_SPIND_S[<spindle>] (speed when the spindle is started via the PLC job interface)

Writing a new speed value

In the following situations, a new speed value is written to the setting data:

- A speed setpoint is entered via NC program, synchronized action or FC18
  Supplementary conditions:
  - MD35035 $MA_SPIND_FUNCTION_MASK (Page 1361), bit 4 == 1
  - DB31, ... DBX84.0 == 0 (constant cutting rate)
- Writing the setting data to the NC program or synchronized action
- Writing the setting data via HMI (OPI) @@@

Supplementary conditions

Gear stage change

A gear stage change is not initiated if the setpoint speed is out of the speed range of the actual gear stage.

Exceptions, see Chapter "M40: Automatic gear stage selection for speeds outside the configured switching thresholds (Page 1375)".

17.7.4.5 Entering a constant cutting rate (SD43202)

When starting a master spindle using a spindle job - and the "Constant cutting rate" spindle speed type is active, then the spindle-specific value entered in the setting data becomes active as constant cutting rate:

SD43202 $SA_SPIND_CONSTCUT_S[<spindle>] (cutting rate when the spindle starts via the PLC job interface)

Writing a new cutting rate of value

In the following situations, a new constant cutting rate of the master spindle is written to the spindle-specific setting data:

- A constant cutting rate is entered via NC program, synchronized action or FC18
  Supplementary conditions:
  - MD35035 $MA_SPIND_FUNCTION_MASK (Page 1361), bit 8 == 1
  - DB31, ... DBX84.0 == 1 (constant cutting rate active)
- Writing the setting data to the NC program or synchronized action
- Writing the setting data via HMI (OPI) @@@

System of units

When writing the setting data, the value is interpreted corresponding to the following secondary conditions:

- Writing via NC program or synchronized action:
  - `G700` active: feet/min
  - `G710` active: m/min
  - `G70`, `G71` active: Depending on the setting in MD10240 $MN_SCALING_SYSTEM_IS_METRIC
- Writing via SINUMERIK Operate:
  - Depending on the setting in MD10240 $MN_SCALING_SYSTEM_IS_METRIC
- Writing via FC18:
  - Function number 10: "constant cutting rate (m/min)"
  - Function number 11: "constant cutting rate (feet/min)"

Reading via system or OPI variable

In the NC program, synchronized action or HMI, the currently entered setpoint for the constant cutting rate can be read using the following system data:

System variable	OPI variable	Meaning
$P_CONSTCUT_S	---	Last programmed cutting rate
$AC_CONSTCUT_S	acConstCutS	Actual constant cutting rate

17.7.4.6 Entering the spindle speed type for the master spindle (SD43206)

Definition of the spindle speed type **for the master spindle** from part program, FC18 or synchronized actions are written to the following setting data from all the usual sources:

SD43206 $SA_SPIND_SPEED_TYPE (spindle speed type for spindle start via PLC interface)

The value range and functionality correspond to the 15th G group (feedrate type).

Permissible values are G values: 93, 94, 95, 96, 961, 97 and 971.

Depending on the setting, for DB31, … DBX30.1/2 (spindle start, clockwise/counter-clockwise) either the speed from SD43200 $SA_SPIND_S or the cutting speed from SD43202 $SA_SPIND_CONSTCUT_S is active:

93, 94, 95, 97 and 971:	The master spindle is started with the speed from SD43200.
96 and 961:	The speed of the master spindle is obtained from the specified cutting rate (SD43202) and the radius of the transverse axis.

17.7.5 External programming (PLC, HMI)

SD43300 and SD42600

The revolutional feedrate behaviour can be selected externally via the axial setting data:
SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (Rotational feedrate for spindles)
in JOG operating mode using the channel-specific setting data
SD42600 $SC_JOG_FEED_PER_REV_SOURCE (Revolutional fedrate control in JOG mode)
.

The following settings can be made via the setting data:

>0:	The machine axis number of the rotary axis/spindle from which the revolutional feedrate shall be derived.
-1:	The revolutional feedrate is derived from the master spindle of the channel in which the axis/ spindle is active in each case.
0:	Function is deselected.

FPRAON (S2)

Revolutional feedrate for spindle S2 ON, derived from the master spindle

FPRAON (S2, A)

Revolutional feedrate for spindle S2 ON, derived from axis A.
The revolutional feedrate value must be specified with `FA[Sn]`.

FPRAOF (S2)

Revolutional feedrate for spindle S2 OFF.

SPI(n)

It is also possible to program `SPI(n)` instead of `SPI(Sn)`.

17.8 Spindle monitoring

17.8.1 Permissible speed ranges

The permissible speed range of a spindle results from the parameterized or programmed speed limit values and the active spindle function (G94, G95, G96, G961, G97, G971, G33, G34, G35, G331, G332, etc.).

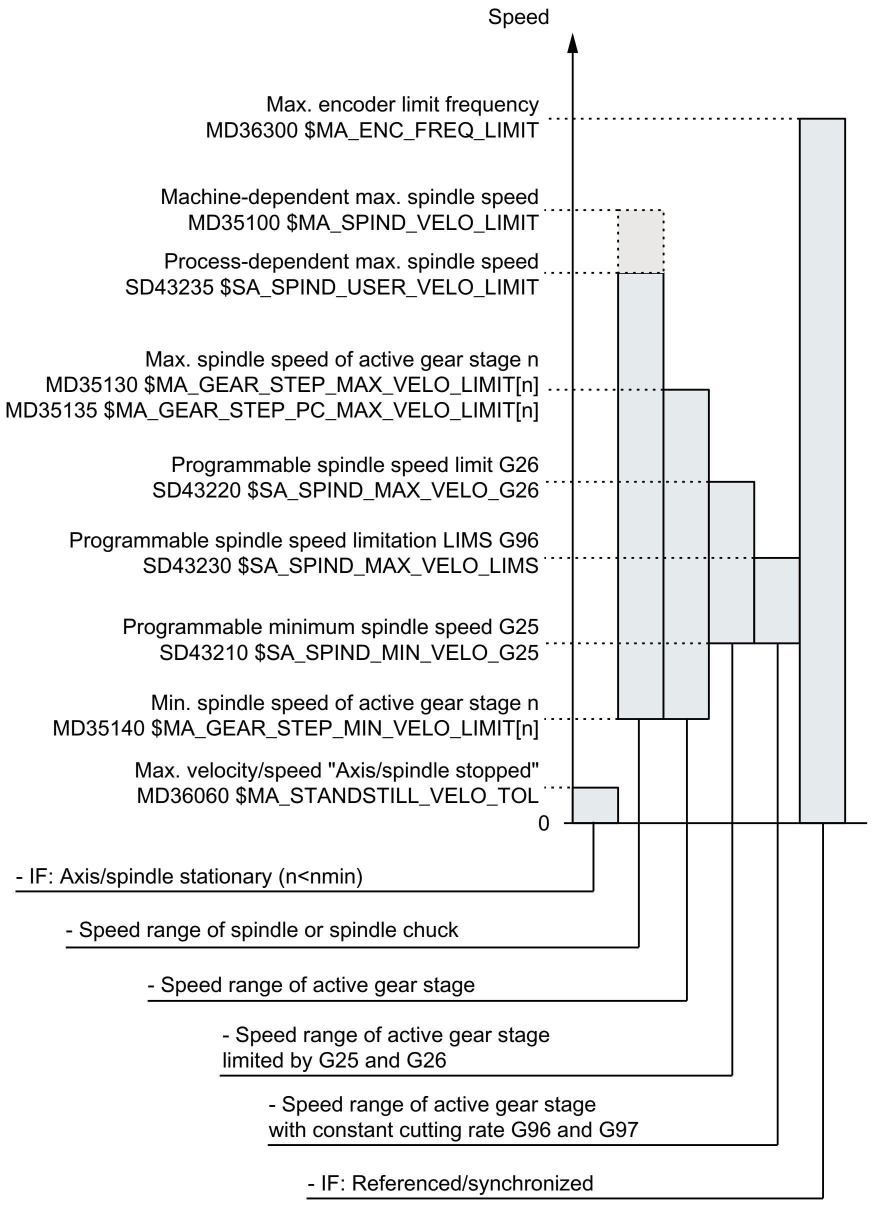

Speed
Max. encoder limit frequency
MD36300 $MA_ENC_FREQ_LIMIT
Machine-dependent max. spindle speed
MD35100 $MA_SPIND_VELO_LIMIT
Process-dependent max. spindle speed
SD43235 $SA_SPIND_USER_VELO_LIMIT
Max. spindle speed of active gear stage n
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[n]
MD35135 $MA_GEAR_STEP_PC_MAX_VELO_LIMIT[n]
Programmable spindle speed limit G26
SD43220 $SA_SPIND_MAX_VELO_G26
Programmable spindle speed limitation LIMS G96
SD43230 $SA_SPIND_MAX_VELO_LIMS
Programmable minimum spindle speed G25
SD43210 $SA_SPIND_MIN_VELO_G25
Min. spindle speed of active gear stage n
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[n]
Max. velocity/speed "Axis/spindle stopped"
MD36060 $MA_STANDSTILL_VELO_TOL
0
- IF: Axis/spindle stationary (n<nmin)
- Speed range of spindle or spindle chuck
- Speed range of active gear stage
- Speed range of active gear stage limited by G25 and G26
- Speed range of active gear stage with constant cutting rate G96 and G97
- IF: Referenced/synchronized

Figure 17-8 Ranges of spindle monitoring functions / speeds

17.8.2 Axis/spindle stationary

Functions such as tool change, open machine doors, path feedrate enable, etc. are only possible at the machine when the spindle is stationary.

Function

The "axis/spindle stationary" state is reached if a setpoint is no longer generated and the spindle actual speed falls below the configured threshold value for "axis/spindle stationary":

MD36060 $MA_STANDSTILL_VELO_TOL (max. velocity/speed for "axis/spindle stationary")

If the spindle has come to a standstill, the following NC/PLC interface signal is set:

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

Effectiveness

Monitoring for spindle stop is effective in all spindle modes and in axis mode.

Deactivate path feed

If a spindle is stopped in the open-loop control mode (M5), then path feed is deactivated if the following machine data is set:

MD35510 $MA_SPIND_STOPPED_AT_IPO_START (feedrate enable for spindle stopped)

The path feed is re-enabled if the spindle comes to a standstill.

17.8.3 Spindle in setpoint range

Function

"Spindle in setpoint range" spindle monitoring checks whether:

- The programmed spindle speed is reached.
- The spindle is at a standstill:
  DB31, ... DBX61.4 (axis/spindle stationary) = 1
- The spindle is still in the acceleration or deceleration phase.

In the spindle mode, open-loop control mode, the setpoint speed is compared with the actual speed. If the actual speed deviates by more than the spindle tolerance that can be entered via MD (refer below) then:

- The following axial NC/PLC interface signal is set to "0":
  DB31, ... DBX83.5 (spindle in setpoint range) = 0
- The next machining block is not enabled (depending on the setting in MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START, see "Axis/spindle stationary (Page 1367)").

Spindle setpoint speed

The spindle speed setpoint is derived from the programmed speed taking into account the spindle override and the active limits.

If the programmed speed is limited or increased, this is displayed using DB31, ... DBX83.1 (speed setpoint limited) or DB31, ... DBX83.2 (speed setpoint increased) (see also "Minimum / maximum speed of the gear stage (Page 1368)"). The means that reaching the tolerance range of the setpoint speed is **not** prevented.

Tolerance range for setpoint speed

The tolerance range of the setpoint speed is defined by the spindle speed tolerance factor:

MD35150 $MA_SPIND_DES_VELO_TOL

Example:

MD35150 $MA_SPIND_DES_VELO_TOL = 0.1

⇒ The spindle actual speed may deviate ±10% from the setpoint speed.

The following NC/PLC interface signal is set to "1" if the spindle actual speed lies within the tolerance range:

DB31, ... DBX83.5 (spindle in setpoint range) = 1

Special case:

If the spindle speed tolerance is set to "0", then DB31, ... DBX83.5 (spindle in the setpoint range) is permanently set to "1" and no path control is performed.

Speed change

Path control only takes place at the start of the traverse block and only if a speed change has been programmed. If the speed tolerance range is exited, e.g. due to an overload, the path movement is not automatically brought to a standstill.

17.8.4 Minimum / maximum speed of the gear stage

Minimum speed

The minimum speed of the gear stage of a spindle is configured in the machine data:

MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[<n>]

The speed setpoints, generated taking into account the override, do not fall below the minimum speed.

If an S value is programmed, which is less than the minimum speed, the setpoint speed is increased to the minimum speed and the following NC/PLC interface signal is set:

DB31, ... DBX83.2 (speed setpoint increased)

The minimum gear stage speed is effective only in speed mode and can only be undershot by:

- Spindle override 0%
- M5
- S0
- DB31, ... DBX4.3 (spindle stop)
- DB31, ... DBX2.1 (withdraw controller enable)
- DB21, ... DBX7.7 (reset)
- DB31, ... DBX2.2 (delete distance-to-go / spindle reset)
- DB31, ... DBX18.5 (oscillation speed)
- DB21, ... DBX7.4 (NC stop axes plus spindles)
- DB31, ... DBX1.3 (axis/spindle disable)
- DB31, ... DBX16.7 (delete S value)

Maximum speed

The maximum speed of the gear stage of a spindle is configured in machine data:

MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]

The speed setpoints, generated taking into account the override, are limited to this speed.

The following NC/PLC interface signal is set in the case that the speed is limited:

DB31, ... DBX83.1 (speed setpoint limited)

17.8.5 Diagnosis of spindle speed limitation

System variables

The effective/limiting spindle parameters can be read via the following system variables: The system variables must be indexed with the spindle number and they return values only relevant in the speed control and spindle position modes.

System variable	Meaning
$AC_SMAXVELO[<n>]	Maximum possible spindle speed [RPM].
$AC_SMAXVELO_INFO[<n>]	Identifier for the speed-limiting data element.
$AC_SMINVELO[<n>]	Minimum possible spindle speed [RPM].
$AC_SMINVELO_INFO[<n>]	Identifier for the speed-increasing data element.
$AC_SMAXACC[<n>]	Maximum possible spindle acceleration [rev/s^2].
$AC_SMAXACC_INFO[<n>]	Identifier for the acceleration-limiting data element.
$AC_SPIND_STATE[<n>]	Spindle status in speed-control mode.
<n>: 0, 1, 2, ... maximum spindle number (0: current master spindle of the channel)	

References

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

List Manual, System Variables

NC/PLC interface signals

The limit or increase of the spindle speed is signaled with the following NC/PLC interface signals:

- DB31, ... DBX83.1 (speed setpoint limited)
- DB31, ... DBX83.2 (speed setpoint increased)

Boundary conditions

Spindle mode

The values delivered by the system variables depend on the spindle mode:

- Speed control mode:
  All system variables deliver current values.
- Positioning mode:
  The system variables $AC_SMAXVELO, $AC_SMAXACC and $AC_SPIND_STATE deliver valid values. The system variables $AC_SMINVELO and $AC_SMINVELO_INFO deliver the data that becomes effective on changing to the speed control mode.
- Axis mode (e.g. if the spindle is used by a transformation TRANSMIT, TRACYL,... or follows a path motion as a special axis):
  The system variable $AC_SPIND_STATE can also be used in the axis mode. Separate system variables are available in the axis mode for dynamic data:
  $AA_VMAXM, $AA_VMAXB and $AA_VLFCT.

SERUPRO block search

The following control response is obtained for a SERUPRO block search:

- The system variable $AC_SMAXVELO / $AC_SMAXACC delivers the maximum representable speed / acceleration.
- $AC_SMAXVELO_INFO and $AC_SMAXACC_INFO deliver the VALUE "0" (no limitation active).
- $AC_SMINVELO and $AC_SMINVELO_INFO deliver data as for normal part program processing.
- $AC_SPIND_STATE returns the states as set for SERUPRO.

Example

The effective/limiting spindle parameters for spindle 1 are written cyclically in the R parameter and displayed as R parameter in the "Parameters" > "User variable" operating area on the HMI.

Program code

```
N05 IDS=1 WHENEVER TRUE DO $R10=$AC_SMAXVELO[1]
```

Program code

```
N10 IDS=2 WHENEVER TRUE DO $R11=$AC_SMAXVELO_INFO[1]
N15 IDS=3 WHENEVER TRUE DO $R12=$AC_SMINVELO[1]
N20 IDS=4 WHENEVER TRUE DO $R13=$AC_SMINVELO_INFO[1]
N25 IDS=5 WHENEVER TRUE DO $R14=$AC_SPIND_STATE[1]
```

See also

Spindle in setpoint range (Page 1367)

17.8.6 Maximum spindle speed

Maximum spindle speed: parameterizable, machine-dependent limit value

The maximum machine-dependent spindle speed – for protecting the spindle chuck or the tool, for example – is parameterized using the following machine data:

MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

The spindle speed is monitored by the NC in the actual values, i.e. taking into account the current gear stage.

Note

Machine manufacturer

It is recommended that changes to the maximum spindle speed in machine data MD35100 only be made when the spindle is stationary. This is particularly important for spindles that are active after NC reset (see also MD35040 SPIND_ACTIVE_AFTER_RESET). Otherwise alarm 22100 may be output.

References

- Manufacturer documentation: List Manual, "Detailed Machine Data Description"
- User documentation: Diagnostics Manual

Responses to violation

If the actual speed of a spindle exceeds the parameterized maximum spindle speed by more than the tolerance value,

```
"Actual spindle speed" > MD35100 $MA_SPIND_VELO_LIMIT (maximum
spindle speed) +
MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance value)
```

the following effects can be seen on the NC side.

- DB31, ... DBX83.0 = 1 (speed limit exceeded)
- Alarm 22100 "Chuck speed exceeded" is displayed
- All axes and spindles of the channel are stopped

Note

Rotary axis / spindle

If a spindle is also temporarily operated as a rotary axis, alarm 22100 "Chuck speed exceeded" is displayed if:

```
"Actual speed of the rotary axis" > MD35100 $MA_SPIND_VELO_LIMIT
(maximum spindle speed) +
MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance value)
```

Remedy: Adjust the maximum spindle speed to the maximum rotary axis speed during rotary operation:

```
MD35100 $MA_SPIND_VELO_LIMIT = MD32000 $MA_MAX_AX_VELO (maximum
speed of the rotary axis)
```

For the change to take effect a reset must be triggered.

Maximum spindle speed: Configurable process-dependent limit value

The maximum process-related spindle speed is configured through the following setting data:

SD43235 $SA_SPIND_USER_VELO_LIMIT (maximum spindle speed)

Changes can be made by writing the setting data through:

- Programming in an NC program
- Manually through the user interface by the machine operator

Changes become effective immediately.

Limitation of the speed setpoint

Where appropriate, the controller limits programmed spindle speed setpoints to the values specified in the setting data. The limitation is then displayed through the following system variables:

$AC_SMAXVELO_INFO[<Spindle number>] == **21**

17.8.7 Maximum encoder limit frequency

⚠ CAUTION

Limit violation

The maximum encoder frequency limit of the actual spindle position encoder is monitored by the control (the limit can be exceeded). It is the responsibility of the machine tool manufacturer to ensure that the configuration of the spindle motor, gearbox, measuring gearbox, encoder and machine data prevents the maximum speed of the actual spindle position encoder being exceeded.

Maximum encoder frequency exceeded.

If the spindle speed reaches a speed (large S value programmed), which exceeds the maximum encoder limit frequency (the maximum mechanical speed limit of the encoder must not be exceeded), the synchronization is lost. The spindle continues to rotate, but with reduced functionality.

With the following functions, the spindle speed is reduced until the active measurement system is operating below the encoder limit frequency again:

- Thread cutting (`G33, G34, G35`)
- Tapping without compensating chuck (`G331, G332`)
- Revolutional feedrate (`G95`)
- Constant cutting rate (`G96, G961, G97, G971`)
- `SPCON` (position-controlled spindle operation)

When the encoder limit frequency is exceeded
NC/PLC IS:
DB31, ... DBX60.4 (referenced/synchronized 1)
or
DB31, ... DBX60.5 (referenced/synchronized 2)
are reset for the measurement system in question and NC/PLC IS:
DB31, ... DBX60.2 (encoder limit frequency 1 exceeded)
or
DB31, ... DBX60.3 (encoder limit frequency 2 exceeded)
are set.

If the spindle is in axis mode, the maximum encoder limit frequency must not be exceeded. The maximum velocity (MD32000 $MA_MAX_AX_VELO) must lie below the maximum encoder limit frequency; otherwise, alarm 21610 is output and the axis is brought to a standstill.

Maximum encoder limit frequency 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 override switch changed, etc.), the spindle is automatically synchronized with the next zero mark or the next proximity switch signal. The new synchronization will always be carried out for the active position measuring system that has lost its synchronization and whose maximum encoder limit frequency is currently undershot.

Special features

If the following functions are active, the maximum encoder frequency cannot be exceeded:

- Spindle positioning mode, axis mode
- Thread cutting (`G33`, `G34`, `G35`)
- Tapping without compensating chuck `G331`, `G332` (does not apply to `G63`)
- Revolutional feedrate (`G95`)
- Constant cutting rate (`G96`, `G961`, `G97`, `G971`)
- `SPCON`

17.8.8 End point monitoring

End point monitoring

During positioning (the spindle is in positioning mode), the system monitors the distance from the spindle (with reference to the actual position) to the programmed spindle position setpoint (end point).

For this to work, in machine data:
MD36000 $MA_STOP_LIMIT_COARSE (Exact stop limit coarse)
and
MD36010 $MA_STOP_LIMIT_FINE (Exact stop limit fine)
two limit values can be defined as an incremental path starting from the spindle position setpoint.

Regardless of the two limit values, the positioning of the spindle is always as accurate as defined by the connected spindle measurement encoder, the backlash, the transmission ratio, etc.

Exact stop window dependent on parameter set

Various parameter-set-dependent exact stop windows can be configured.
This makes it possible to work to different levels of accuracy in axis mode and spindle positioning. The exact stop window can be configured separately for each gear step for spindle positioning.

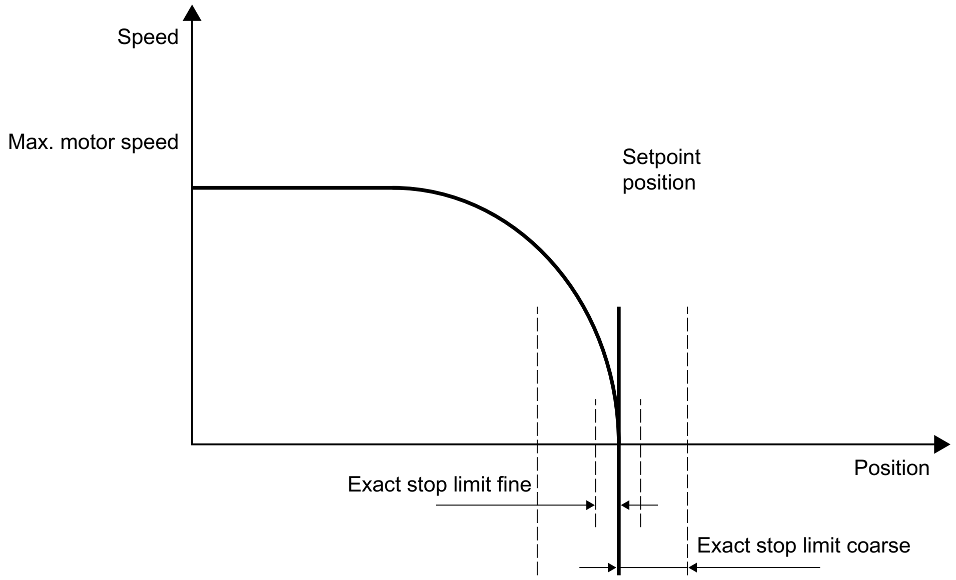


Figure 17-9 Exact stop zones of a spindle

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

The two limit values defined by machine data:
MD36000 $MA_STOP_LIMIT_COARSE (Exact stop limit coarse)
and
MD36010 $MA_STOP_LIMIT_FINE (Exact stop limit fine)
are output to the PLC using NC/PLC IS:
DB31, ... DBX60.7 (Position reached with exact stop coarse)
and
DB31, ... DBX60.6 (Position reached with exact stop fine).

Block change for SPOS and M19

When positioning the spindle with `SPOS` or `M19` the block is changed
dependent on end point monitoring with NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop fine).

All other functions programmed in the block must have achieved their end criterion (e.g., all auxiliary functions acknowledged by the PLC).

With `SPOSA`, the block change does not depend on the monitoring of the end point.

17.8.9 M40: Automatic gear stage selection for speeds outside the configured switching thresholds

Function

When M40 is active, an automatic gear stage selection is also made if the programmed spindle speed `S…` lies outside the configured switching thresholds.

In this case, a distinction is made between the following cases:

- **Programmed speed too high**
  The programmed speed is higher than the configured maximum speed of the numerically largest gear stage:
  S... > MD35110 $MA_GEAR_STEP_MAX_VELO[<n>]
  In this case, the **highest gear stage** is selected (according to MD35090 $MA_NUM_GEAR_STEPS).
- **Programmed speed too low**
  The programmed spindle speed is less than the configured minimum speed of the first gear stage:
  S... < MD35120 $MA_GEAR_STEP_MIN_VELO[1]
  In this case, the **first gear stage** is selected.
- **Programmed speed = 0**
  When programming speed 0 (`S0`) the behavior depends on the configuration of the minimum speed of the first gear stage MD35120 $MA_GEAR_STEP_MIN_VELO[1]:
  - If MD35120 $MA_GEAR_STEP_MIN_VELO[1] **= 0** is configured, then when programming `S0`, the **first gear stage** is selected.
  - If MD35120 $MA_GEAR_STEP_MIN_VELO[1] **> 0** is configured, when programming `S0` **no gear stage change** is performed and the last gear stage remains active. This means that it remains possible to stop the spindle with `S0` (instead of `M5`) without initiating a gear stage change.

Effectiveness

Selecting the highest gear stage or the first gear stage for automatic gear stage selection (M40) is active when programming spindle speeds `S...` using the part program, in synchronized actions or when entering via PLC FC18.

For speed programming from the part program for tapping with G331, the behavior is also supported for the second data set to select the gear stage (precondition:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, bit 5 = 1).

Supplementary conditions

Enabling the gear stage change

The precondition for the function is that the gear stage change is generally enabled via machine data:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear stage change).

MD35090 $MA_NUM_GEAR_STEPS (number of gear stages set up)

Example

Automatic gear stage selection M40 is the basic setting after NC reset.

Part program:

Program code	Comment
...	
N15 S3500 M3	; S3500 is greater than MD35110 of the 2nd gear stage. The 2nd gear stage is selected.
...	
N50 S0 M3	; Spindle is stopped, S0 does not request a gear stage change (special handling, S0).
...	
N95 S5 M3	; S5 is less than MD35120 of the 1st gear stage. The 1st gear stage is selected.
...	

Configuring data for spindle 1 (AX5):

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE[AX5] = 1	; Enable gear stage change
MD35090 $MA_NUM_GEAR_STEPS[AX5] = 2	; Number of existing gear stages
MD35110 $MA_GEAR_STEP_MAX_VELO[1,AX5] = 500	; Upper switching threshold for gear stage 1
MD35120 $MA_GEAR_STEP_MIN_VELO[1,AX5] = 10	; Lower switching threshold for gear stage 1
MD35110 $MA_GEAR_STEP_MAX_VELO[2,AX5] = 2000	; Upper switching threshold for gear stage 2
MD35120 $MA_GEAR_STEP_MIN_VELO[2,AX5] = 500	; Lower switching threshold for gear stage 2

17.9 Spindle with SMI 24 (Weiss spindle)

17.9.1 General Information

In order to be able to process the sensor data of the spindle in the control, the sensors must first be connected to I/O modules and transferred to the PLC via fieldbus (PROFIBUS DP or PROFINET I/O).

For a spindle with SMI 24 (Weiss spindle), the sensor data are transferred to the drive using DRIVE-CLiQ and are available there in drive parameters. When using cyclic drive telegram 139, sensor data from the drive are transferred to the control. They are then available there in the following system data:

- System variable
- OPI variables
- NC/PLC interface signals

Requirement

- The spindle is connected to the drive via Sensor Module SMI 24 using DRIVE-CLiQ.
- Drive telegram 139 is configured for the spindle.

Note

Drive telegram 139

In principle, a spindle with Sensor Module SMI 24 can also be operated with another drive telegram. However, sensor data is only transferred to the control using drive telegram 139.

17.9.2 Sensor data

Sensors in the spindle motor

The spindle sensors provide information about the clamping device and the angular position of the motor shaft:

- Analog sensor S1: Clamped state
  Voltage value in the range from 0 - 10 V depending on the position of the draw bar.
- Digital sensor S4: Piston end position
  - 0 = piston not in position
  - 1 = piston is in position, i.e. piston is free to move
- Digital sensor S5: Angular position of the motor shaft
  - 0 = motor shaft not aligned
  - 1 = motor shaft is in position (requirement: The spindle is stationary)

Note

Spindle with sensor module SMI 24 and axis container

A spindle with sensor module SMI 24 and drive telegram 139 for the transmission of sensor data to the control must not be part of an axis container whose axes are distributed over several NCUs via an NCU link.

Transmission of sensor data

Sensor data is transferred to the control from sensor module SMI 24 in cyclic drive telegram 139 as process data 11 - 14. Drive telegram 139 is based on drive telegram 136, where sensor data is transferred instead of the data of the 2nd encoder. A detailed description of drive telegram 139 can be found in:

References:
SINAMICS S120/S150 List Manual; Function Diagrams, Section: PROFIdrive

System data: Sensor data

Sensor data can be read into the control via the following system data:

Meaning	System variable $VA_	NC/PLC interface DB31, ...	OPI variables	Drive parameters
Sensor configuration	MOT_SENSOR_CONF[<axis>]	DBB132, DBB133	vaMotSensorConf	r5000
Clamped state 1)	MOT_CLAMPING_STATE[<axis>]	DBW134	vaMotClampingState	r5001
Measured value analog sensor S1 1) 2)	MOT_SENSOR_ANA[<axis>]	DBW136	vaMotSensorAna	r5002
Status digital sensors	MOT_SENSOR_DIGI[<axis>]	DBB138, DBB139	vaMotSensorDigi	r5003
<axis>: Machine axis name: AX1 ... AXn or spindle name: S1 ... Sm				

1) See Section "Clamped state (Page 1380)"

2) Sensor S1: 0 - 10 V

Analog actual value: 10000 increments, resolution 1 mV

Example:

SIMATIC S7 input module: 27648 increments, resolution 0.36 mV

Adaptation factor if you change to a spindle with SMI 24: (10000 incr. * 1 mV) / (27648 incr. * 0.36 mV) = 1.00469393

Detailed system data description

NC/PLC interface signals	NC Variables and Interface Signals List Manual
System variable:	List Manual, System Variables
OPI variables:	List Manual 2; Variables
Drive parameter:	SINAMICS S120/S150 List Manual

17.9.3 Clamped state

Sensor S1 supplies an analog voltage value 0 V - 10 V depending on the position of the clamping device. The voltage value is available in the system data for evaluation of the clamped state on the user side.

Note

The subsequently described evaluation of sensor S1 to generate the state value for the clamped state and limiting the spindle speed are only realized if the following state values are displayed in drive parameter r5000:

- r5000.0 == 1: Sensors available
- r5000.1 == 1: Sensor S1 (clamped state) available
- r5000.10 == 1: State values are generated, speed limits p5043 active

See also Section "Sensor data (Page 1378)", paragraph: "System data: sensor data"

State value

To simplify the evaluation, the clamped state in the system data is also available as state value 0 - 11.

A voltage range corresponds to a certain clamped state. The voltage ranges can be set using drive parameter p5041[0...5].

A voltage tolerance can also be set for the voltage ranges using drive parameter p5040.

Note

The voltage range ± voltage tolerance must not overlap.

Speed limits

For the clamped states with state values 3 - 10, speed limit values can be specified using drive parameter p5043[0...6]. In the other clamped states (state values 1, 2 and 11), a limit value of 0 [rpm] permanently applies.

In the various clamped states, the controller limits the spindle speed to the applicable limit.

Note

Changing the speed limits

A change of the speed limits in drive parameter p5043[0...6] is only effective in the controller (limitation of the spindle speed to the new speed limit) after:

- Power-on reset or when the controller is switched-off/switched-on
- Deselection of the "Parking" state for the spindle (see Section "Parking a machine axis (Page 125)")

Context: State value, voltage range and speed limit

State value [1]	Clamped state	Voltage range [2]		Speed limit
		Upper limit	Lower limit	
0	Sensor S1 not available or state values inactive	---	---	---
1	State initialization running	---	---	[3]
2	Released with signal (error state)	---	p5041[0] + p5040	[3]
3	Released	p5041[0]	p5041[1]	p5043[0]
4	Clamping with tool	---	---	p5043[1]
5	Releasing with tool	---	---	p5043[2]
6	Releasing without tool	---	---	p5043[3]
7	Clamped with tool AND S4 == 0	p5041[2]	p5041[3]	p5043[4]
8	Clamped with tool AND S4 == 1			p5043[4]
9	Clamping without tool	---	---	p5043[5]
10	Clamped without tool	p5041[4]	p5041[5]	p5043[6]
11	Clamped with signal (error state)	p5041[5] - p5040	---	[3]

[1] The state value can be read into the controller using the following system data:
- System variable: $VA_MOT_CLAMPING_STATE[<axis>]
- NC/PLC interface: DB31, ... DBW134
- OPI variables: vaMotClampingState
- Drive parameters: r5001

[2] p5041[0...5]: Voltage threshold values, p5040: Voltage threshold values tolerance

[3] Speed limit permanently set: 0 [rpm]

17.9.4 Additional drive parameters

P5042:Transition time

The following times can be set in drive parameter p5042 for the clamped state identification:

- p5042[0]: Stabilization time for "clamped with tool"
  The clamped state "clamped with tool" must be present in the spindle motor for at least the set stabilization time before the state is signaled to the controller.
- p5042[1]: Maximum time for clamping
  The transition from the "released" state to the "clamped with tool" or "clamped without tool" state may take – as a maximum – the set time.

r5044: Speed limitation from the clamping cycle

The speed limit from p5043[6] which is active in the clamped state "clamped without tool" is displayed in drive parameter r5044.

A value of 65535 means that the speed limit is not active.

17.10 Supplementary conditions

17.10.1 Changing control parameters

For spindles that are not in position-controlled mode, machine data changes also take effect when the spindle is not stationary with the NEWCONF command.

In the case of changes to control parameters, speed setpoint jumps may occur when the new values take effect. Control parameters are, for example:

- MD32200 $MA_POSCTRL_GAIN (servo gain factor)
- MD32210 $MA_POSCTRL_INTEGR_TIME (position controller integral time)
- MD32410 $MA_AX_JERK_TIME (time constant for the axial jerk filter)

17.11 Examples

17.11.1 Automatic gear step selection (M40)

Example

To illustrate the contents of the new block search variables:
Assumptions about automatic gear step selection (`M40`):

S0...500	1. Gear step
S501..1000	2. Gear step
S1001..2000	3. Gear step

Content of system variables:

$P_SEARCH_S	; Collected S value
$P_SEARCH_DIR	; Collected direction of rotation
$P_SEARCH_GEAR	; Collected gear step

Collected	S value:	Direction of rota-tion:	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)

17.12 Data lists

17.12.1 Machine data

17.12.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10192	GEAR_CHANGE_WAIT_TIME	Wait time for acknowledgment of a gear stage change during reorganization
10714	M_NO_FCT_EOP	M function for spindle active after `RESET`
12060	OVR_SPIND_IS_GRAY_CODE	Spindle override 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	Defines the reference of the path override
12090	OVR_FUNCTION_MASK	Selection of override specifications

17.12.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20090	SPIND_DEF_MASTER_SPIND	Initial setting for master spindle on channel
20092	SPIND_ASSIGN_TAB_ENABLE	Enabling/disabling of spindle converter
20850	SPOS_TO_VDI	Output of M19 to the PLC with SPOS/SPOSA
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET

17.12.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis
30310	ROT_IS_MODULO	Modulo conversion
31044	ENC_IS_DIRECT2	Encoder on intermediate gear
31050	DRIVE_AX_RATIO_DENOM	Load gear denominator
31060	DRIVE_AX_RATIO_NUMERA	Load gear numerator
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 of the current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant of the speed control loop for feedforward control
32910	DYN_MATCH_TIME	Time constant for dynamic response adaptation
34040	REFP_VELO_SEARCH_MARKER	Reference point creep velocity
34060	REFP_MAX_MARKER_DIST	Monitoring of zero mark distance
34080	REFP_MOVE_DIST	Reference point distance / destination point for distance-coded system
34090	REFP_MOVE_DIST_CORR	Reference point offset / absolute offset, distance-coded
34100	REFP_SET_POS	Reference point value
34200	ENC_REFP_MODE	Referencing mode
35000	SPIND_ASSIGN_TO_MACHAX	Assignment of spindle to machine axis
35010	GEAR_STEP_CHANGE_ENABLE	Type of gear stage change
35012	GEAR_STEP_CHANGE_POSITION	Gear stage change position
35014	GEAR_STEP_USED_IN_AXISMODE	Gear stage for axis mode with M70
35020	SPIND_DEFAULT_MODE	Basic spindle setting
35030	SPIND_DEFAULT_ACT_MASK	Activate basic spindle setting
35035	SPIND_FUNCTION_MASK	Setting of spindle-specific functions
35040	SPIND_ACTIVE_AFTER_RESET	Spindle active after reset
35090	NUM_GEAR_STEPS	Number of set gear stages
35092	NUM_GEAR_STEPS2	2nd gear stage data set: Number of set gear stages
35100	SPIND_VELO_LIMIT	Maximum spindle speed
35110	GEAR_STEP_MAX_VELO[n]	Maximum speed for automatic gear stage change
35112	GEAR_STEP_MAX_VELO2[n]	2nd gear stage data set: Maximum speed for automatic gear stage change
35120	GEAR_STEP_MIN_VELO[n]	Minimum speed for automatic gear stage change

Number	Identifier: $MA_	Description
35122	GEAR_STEP_MIN_VELO2[n]	2nd gear stage data set: Minimum speed for automatic gear stage change
35130	GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum speed of gear stage
35135	GEAR_STEP_PC_MAX_VELO_LIMIT[n]	Maximum speed of gear stage in position control
35140	GEAR_STEP_MIN_VELO_LIMIT[n]	Minimum speed of gear stage
35150	SPIND_DES_VELO_TOL	Spindle speed tolerance
35160	SPIND_EXTERN_VELO_LIMIT	Spindle speed limitation from PLC
35200	GEAR_STEP_SPEEDCTRL_ACCEL[n]	Acceleration in speed control mode
35210	GEAR_STEP_POSCTRL_ACCEL[n]	Acceleration in position control mode
35212	GEAR_STEP_POSCTRL_ACCEL2[n]	2nd gear stage data set: Acceleration in position control mode
35220	ACCEL_REDUCTION_SPEED_POINT	Speed limit for reduced acceleration
35230	ACCEL_REDUCTION_FACTOR	Reduced acceleration
35300	SPIND_POSCTRL_VELO	Position control activation speed
35350	SPIND_POSITIONING_DIR	Positioning direction of rotation for a 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
35550	DRILL_VELO_LIMIT[n]	Maximum speeds for tapping
35590	PARAMSET_CHANGE_ENABLE	Parameter set specification possible from PLC
36060	STANDSTILL_VELO_TOL	Threshold velocity "Axis/spindle stationary"
36200	AX_VELO_LIMIT	Threshold value for velocity monitoring.

17.12.2 Setting data

17.12.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42600	JOG_FEED_PER_REF_SOURCE	Revolutional feedrate control in JOG mode
42800	SPIND_ASSIGN_TAB	Spindle number converter
42900	MIRROR_TOOL_LENGTH	Mirror tool length offset
42910	MIRROR_TOOL_WEAR	Mirror wear values of tool length compensation
42920	WEAR_SIGN_CUTPOS	Mirror wear values of machining plane

Number	Identifier: $SC_	Description
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`)

17.12.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43200	SPIND_S	Speed for spindle start via PLC interface
43202	SPIND_CONSTCUT_S	Cutting rate for spindle start via PLC interface
43206	SPIND_SPEED_TYPE	For spindle start via PLC interface
43210	SPIND_MIN_VELO_G25	Progr. Spindle speed limiting `G25`
43220	SPIND_MAX_VELO_G26	Progr. Spindle speed limiting `G26`
43230	SPIND_MAX_VELO_LIMS	Progr. spindle speed limiting `G96/G961`
43235	SPIND_USER_VELO_LIMIT	Maximum spindle speed
43240	M19_SPOS	Spindle position for spindle positioning with M19
43250	M19_SPOSMODE	Spindle positioning approach mode for spindle positioning with M19
43300	ASSIGN_FEED_PER_REF_SOURCE	Rotational feedrate for positioning axes/spindles

17.12.3 signals

17.12.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate override A to H	DB31, ...DBX0.0-7	DB380x.DBX0.0-7
Axis/spindle disable	DB31, ...DBX1.3	DB380x.DBX1.3
Follow-up mode	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1	DB31, ...DBX1.5	DB380x.DBX1.5
Position measuring system 2	DB31, ...DBX1.6	DB380x.DBX1.6
Override active	DB31, ...DBX1.7	DB380x.DBX1.7
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Spindle reset/delete distancetogo	DB31, ...DBX2.2	DB380x.DBX2.2
Velocity/spindle speed limitation	DB31, ...DBX3.6	DB380x.DBX3.6
Program test Axis/Spindle Enable	DB31, ...DBX3.7	DB380x.DBX3.7
Actual gear stage A to C	DB31, ...DBX16.0-2	DB380x.DBX2000.0-2
Gear is changed over	DB31, ...DBX16.3	DB380x.DBX2000.3
Resynchronize spindle 1	DB31, ...DBX16.4	DB380x.DBX2000.4
Resynchronize spindle 2	DB31, ...DBX16.5	DB380x.DBX2000.5
no n-monitoring with gear change	DB31, ...DBX16.6	DB380x.DBX2000.6

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Delete S value	DB31, ...DBX16.7	DB380x.DBX2000.7
Feedrate override for spindle valid	DB31, ...DBX17.0	DB380x.DBX2001.0
Resynchronize spindle during positioning 1	DB31, ...DBX17.4	DB380x.DBX2001.4
Resynchronize spindle during positioning 2	DB31, ...DBX17.5	DB380x.DBX2001.5
Invert M3/M4	DB31, ...DBX17.6	DB380x.DBX2001.6
Oscillation controlled by the PLC	DB31, ...DBX18.4	DB380x.DBX2002.4
Oscillation enable (oscillation speed)	DB31, ...DBX18.5	DB380x.DBX2002.5
Oscillation rotation direction clockwise (Set rotation direction clockwise)	DB31, ...DBX18.6	DB380x.DBX2002.6
Oscillation rotation direction counterclockwise (Set rotation direction counterclockwise)	DB31, ...DBX18.7	DB380x.DBX2002.7
Spindle override A to H	DB31, ...DBX19.0-7	DB380x.DBX2003.0-7

17.12.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Spindle / rotary axis	DB31, ...DBX60.0	DB390x.DBX0.0
Encoder limit frequency exceeded 1	DB31, ...DBX60.2	DB390x.DBX0.2
Encoder limit frequency exceeded 2	DB31, ...DBX60.3	-
Referenced/synchronized 1	DB31, ...DBX60.4	DB390x.DBX0.4
Referenced/synchronized 2	DB31, ...DBX60.5	DB390x.DBX0.5
Position reached with exact stop coarse	DB31, ...DBX60.6	DB390x.DBX0.6
Position reached with exact stop fine	DB31, ...DBX60.7	DB390x.DBX0.7
Axis/spindle stationary ($n < n_{min}$)	DB31, ...DBX61.4	DB390x.DBX1.4
Position controller active	DB31, ...DBX61.5	DB390x.DBX1.5
Speed controller active	DB31, ...DBX61.6	DB390x.DBX1.6
Current controller active	DB31, ...DBX61.7	DB390x.DBX1.7
Restored 1	DB31, ...DBX71.4	DB390x.DBX11.4
Restored 2	DB31, ...DBX71.5	DB390x.DBX11.5
Setpoint gear stage A to C	DB31, ...DBX82.0-2	DB390x.DBX2000.0-2
Change gear	DB31, ...DBX82.3	DB390x.DBX2000.3
Speed limit exceeded	DB31, ...DBX83.0	DB390x.DBX2001.0
Setpoint speed limited	DB31, ...DBX83.1	DB390x.DBX2001.1
Setpoint speed increased	DB31, ...DBX83.2	DB390x.DBX2001.2
Spindle in setpoint range	DB31, ...DBX83.5	DB390x.DBX2001.5
Actual direction of rotation clockwise	DB31, ...DBX83.7	DB390x.DBX2001.7
Rigid tapping active	DB31, ...DBX84.3	DB390x.DBX2002.3
active spindle mode synchronous mode	DB31, ...DBX84.4	DB390x.DBX2002.4
Active spindle positioning mode	DB31, ...DBX84.5	DB390x.DBX2002.5
Active spindle mode oscillation mode	DB31, ...DBX84.6	DB390x.DBX2002.6
Active spindle control mode	DB31, ...DBX84.7	DB390x.DBX2002.7

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Spindle actually reached in position	DB31, ...DBX85.5	DB390x.DBX2003.5
M function for spindle	DB31, ...DBB86-87	DB370x.DBD0000
S function for spindle	DB31, ...DBB88-91	DB370x.DBD0004
Sensors available	DB31, ...DBX132.0	DB390x.DBX7000.0
Sensor S1 available (clamped state)	DB31, ...DBX132.1	DB390x.DBX7000.1
Sensor S4 available (piston end position)	DB31, ...DBX132.4	DB390x.DBX7000.4
Sensor S5 available (angular position of the motor shaft)	DB31, ...DBX132.5	DB390x.DBX7000.5
State value is generated, speed limitation p5043 is active	DB31, ...DBX133.2	DB390x.DBX7001.2
Clamped state	DB31, ...DBW134	DB390x.DBW7002
Analog value: Clamped state	DB31, ...DBW136	DB390x.DBW7004
Sensor S4 (piston end position)	DB31, ...DBX138.4	DB390x.DBX7006.4
Sensor S5 (angular position of the motor shaft)	DB31, ...DBX138.5	DB390x.DBX7006.5

V1: Feedrates

18

18.1 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 speed (G96, G961)
- Constant speed (G97, G971)
- Feedrate for thread cutting (G33, G34, G35)
- Feedrate for tapping without compensating chuck (G331, G332)
- Feedrate for tapping with compensating chuck (G63)
- Feedrate for chamfer/rounding FRC, FRCM
- Non-modal feedrate FB

Axis assignment of the feedrates

Feedrates can be assigned to the axes variably to adjust to the different technological requirements.

The following variants are possible:

- Separate feedrates for the working plane and the infeed axis
- Variable axis assignment for path feedrate
- Feedrate for positioning axes

Feedrate control

The programmed feedrate can be changed during the machining or for test purposes to enable adjustment to the changed technological conditions.

- Via the machine control panel
- Via the operator panel front

- Via the PLC
- Per program command

18.2 Path feedrate F

Path feedrate F

The path feedrate represents the geometrical total of the speed components in the participating axes. It is therefore generated from the individual motions of the interpolating axes.

The default uses the axial speeds of the geometry axes which have been programmed. The `FGROUP` command can be used to include other geometry and/or synchronized axes in the calculation of the path feedrate.

The path feedrate F determines the machining speed and is observed in every type of interpolation even where allowance is made for tool offsets. The value programmed under the address F remains in a program until a new F value or a new type of feedrate is programmed.

Range of values for path feedrate F

See Description of Functions G2: "Speeds, Setpoint / Actual Value Systems, Closed-Loop Control", Section: "Velocities (Page 343)".

F value at PLC interface

The F value of the current path feedrate is always entered in the channel-specific PLC interface for auxiliary functions (DB21, ... DBB158 to 193).

For explanations about the associated interface signals (change signal, F value), see Description of Functions "H2: Auxiliary function outputs to PLC (Page 401)".

Feedrate with transition circle

References:
Programming Manual, Fundamentals

Feedrate for internal radius and external radius path sections

For circular blocks or spline blocks with curvature in the same direction and tool radius offset activated (G41/G42), the programmed feedrate can act on the center point path or on the contour (depending on the internal radius or external radius path sections).

A group of G 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:
Programming Manual, Fundamentals

Maximum tool path velocity

The maximum path velocity results from the maximum velocities of the linear or rotary axes involved (MD32000 $MA_MAX_AX_VELO), i.e. the axis with the lowest maximum velocity determines the maximum path velocity. This cannot be exceeded.

If `G0` is programmed, traversing is at the path velocity resulting from the MD32000 $MA_MAX_AX_VELO limitation.

Limit velocity for path axes

In addition, the `FL[<axis>]` statement can be used to program a limit velocity for path axes (geometry and synchronized axes).

This enables separate feedrates to be programmed for the working plane and infeed axis. This means that a feedrate is specified for the path-related interpolation and for the infeed axis. The axis perpendicular to the selected machining plane is designated as the infeed axis. The infeed axis-specific feedrate can be programmed to limit the axis velocity and therefore the path velocity. No coordinate rotations through frames should be included, i.e. the infeed axis must be an axis of the standard coordinate system. This function can be used to compensate for the fact that a cutter has a lower cutting performance on the face side than across the cutter circumference.

Programming example:

Program code	**Comment**
`... G94 ...`	`; Selection of feedrate type (mm/min)`
`X30 Y20 F200`	`; Path feedrate = 200 mm/s`
`FL[Z]=50 Z-30`	`; Max. feedrate for Z axis: 50 mm/s`

Low-resolution encoders

When using low-resolution encoders, more continuous path or axis motions can be achieved with smoothed actual values. The larger the time constant, the better the smoothing of the actual values, and the longer the overtravel.

MD34990 $MA_ENC_ACTVAL_SMOOTH_TIME[<axis>] (smoothing time constant for actual values)

Smoothed actual values are used for:

- Thread cutting (G33, G34, G35)
- Feedrate per revolution ((G95, G96, G97, FPRAON)
- Display of speed, actual position and velocity

18.2.1 Feedrate type G93, G94, G95

Effectiveness

The feedrate types G93, G94, G95 are active for the G commands of group 1 (except G0) in the automatic modes.

G94 or G95 can be used for traversing in JOG mode.

References:
Function Manual, Extended Functions; Manual traversing and manual handwheel traversing (H1)

Inverse-time feedrate (G93)

The inverse-time feedrate is used when it is easier to program the duration, rather than the feedrate, for retraction of a block.

The inverse-time feedrate is calculated from the following formula:

`F = v / s`

with	`F`:	Inverse-time feedrate in rpm
	`v`:	Required path velocity in mm/min or inch/min
	`s`:	Path length in mm/inch

Programming example

Program code	Comment
`N10 G1 G93 X100 Y200 F2`	`; The programmed path is traversed in 0.5 min.`
`...`	

Note

`G93` may not be used when `G41` / `G42` is active. If the block length varies greatly from block to block, a new F value should be programmed in each block for `G93`.

Linear feedrate (G94)

The linear feedrate is programmed in the following units relative to a linear or rotary axis:

- [mm/min, degrees/min] on standard metric systems
- [inch/min, degrees/min] on standard imperial systems

Revolutional feedrate (G95)

The revolutional feedrate is programmed in the following units relative to a master spindle:

- [mm/rev] on standard metric systems
- [inch/rev] on standard imperial systems
- [degrees/rev] on a rotary axis

The path velocity is calculated from the actual speed of the spindle according to the following formula:

```
V = n * F
```

with	`V`:	Path velocity in mm/min or inch/min
	`n`:	Speed of the master spindle in rpm
	`F`:	Programmed revolutional feedrate in mm/rev or inch/rev

Note

The programmed F value is deleted when the system switches between the feedrate types G93, G94 and G95.

Tooth feedrate

Primarily for milling operations, the tooth feedrate `FZ...` (feed distance per tooth), which is more commonly used in practice, can be programmed instead of the revolutional feedrate `F...`:

The control system uses the $TC_DPNT (number of teeth per revolution) tool parameter associated with the active tool offset data record to calculate the effective revolutional feedrate for each traversing block from the programmed tooth feedrate.

```
F = FZ * $TC_DPNT
```

with	`F`:	Revolutional feedrate in mm/rev or inch/rev
	`FZ`:	Tooth feedrate in mm/tooth or inch/tooth
	`$TC_DPNT`:	Tool parameter: Number of teeth/rev

Example: Milling cutter with 5 teeth ($TC_DPNE = 5)

Program code	Comment
`N10 G0 X100 Y50`	
`N20 G1 G95 FZ=0.02`	`; Tooth feedrate 0.02 mm/tooth`
`N30 T3 D1`	`; Load tool and activate tool offset data block.`

```
Program code                Comment
M40 M3 S200                 ; Spindle speed 200 rpm
N50 X20                     ; Milling with FZ = 0.02 mm/tooth
                            ; effective revolutional feedrate:
                            ; F = 0.02 mm/tooth * 5 teeth/rev = 0.1 mm/rev
                            ; or:
                            ; F = 0.1 mm/rev * 200 rpm = 20 mm/min
...
```

Setting data

Revolutional feedrate in JOG mode

The behavior of an axis in terms of its revolutional feedrate relative to the master spindle of the channel to which the axis is currently assigned in JOG mode depends on the settings in the NC-specific setting data:

SD41100 $SN_JOG_REV_IS_ACTIVE, Bit <x>

Bit	Value	Meaning
0	0	The behavior depends on further setting data, depending on the function of the axis: ● **Axis / spindle** SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE ● **Geometry axis** on which a **frame with rotation** acts: SD42600 $SC_JOG_FEED_PER_REV_SOURCE ● **Orientation axis**: SD42600 $SC_JOG_FEED_PER_REV_SOURCE
	1	The axis is traversed with the revolutional feedrate. **Supplementary condition** Traversing is performed without the revolutional feedrate if the following applies: ● **Axis / spindle** The master spindle is stationary **AND** SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE == -3 ● **Geometry axis** on which a **frame with rotation** acts: The master spindle is stationary **AND** SD42600 $SC_JOG_FEED_PER_REV_SOURCE == -3 ● **Orientation axis**: The master spindle is stationary AND SD42600 $SC_JOG_FEED_PER_REV_SOURCE == -3
1	0	In **rapid traverse**, traversing is performed **with** the revolutional feedrate.
	1	In **rapid traverse**, traversing is performed **without** the revolutional feedrate.
2 1)	0	In **handwheel jogging**, traversing is performed **with** the revolutional feedrate.
	1	In **handwheel jogging**, traversing is performed **without** the revolutional feedrate.
3 1)	0	In **DRF handwheel traversing**, traversing is performed **with** the revolutional feedrate.
	1	In **DRF handwheel traversing**, traversing is performed **without** the revolutional feedrate.
1) The function is active in addition to the setting in bit 1.		

NC/PLC interface signals

- Revolutional feedrate active (**channel-specific**)
  The interface signal indicates that axes are traversed with revolutional feedrate in the channel:
  - AUTOMATIC mode: Path or synchronized axes
  - JOG mode: Geometry axes

  DB31, ... DBX62.2 == 1 (revolutional feedrate active)
- Revolutional feedrate active (**axis-specific**)
  The interface signal indicates that the axis is traversed with revolutional feedrate:
  DB31, ... DBX33.2 == 1 (revolutional feedrate active)

18.2.2 Type of feedrate G96, G961, G962, G97, G971

Constant cutting rate (G96, G961)

The constant cutting rate is used on turning machines to keep the cutting conditions constant, independently of the work diameter of the workpiece. This allows the tool to be operated in the optimum cutting performance range and therefore increases its service life.

Selection of G96, G961:

When programming G96, G961, the corresponding S value is interpreted as the cutting rate in m/min or ft/min along the transverse axis. If the workpiece diameter decreases during machining, the speed is increased until the constant cutting speed is reached.

When G96, G961 is first selected in the part program, a constant cutting rate must be entered in mm/min or ft/min.

With G96, the control system will automatically switch to revolutional feedrate (as with G95), i.e. the programmed feedrate F is interpreted in mm/rev or inch/rev.

When programming G961, linear feedrate is selected automatically (as with G94). A programmed feedrate F is interpreted in mm/min or inch/min.

Determining the spindle speed

Based on the programmed cutting rate (either S_{G96} or S_{G961}) and the actual cartesian position of the transverse axis (radius), the control system calculates the spindle speed at the TCP using the following formula:

$$n = \frac{S_{Speed}}{2 * \pi * r}$$

n:	Spindle speed
S_{Speed}:	Programmed cutting rate
π	Circle constant
r:	Radius (distance, center of rotation to TCP)

The following is assumed when determining the radius:

- The transverse axis position 0 in the WCS represents the center of rotation.
- Position offsets (such as online tool offset, external zero offset, $AA_OFF, DRF offset and compile cycles) and position components through couplings (e.g. following axis for TRAIL) are not taken into account when determining the radius.

Frames (e.g. programmable frames such as SCALE, TRANS or ROT) are taken into account in the calculation of the spindle speed and can bring about a change in speed, if the effective diameter at the TCP changes.

Diameter programming and reference axis for several transverse axes in one channel:

One or more transverse axes are permitted and can be activated simultaneously or separately:

- Programming and displaying in the user interface in the diameter
- Assignment of the specified reference axis with `SCC[<axis>]` for a constant cutting rate G96, G961, G962

For more information, see Description of Functions "P1: Transverse axes (Page 859)".

Example

S_{G96} = 230 m/min

- Where r = 0.2 m → n = 183.12 rpm
- Where r = 0.1 m → n = 366.24 rpm

⇒ The smaller the workpiece diameter, the higher the speed.

For G96, G961 or G962, a geometry axis must be defined as the transverse axis.

The transverse axis, whose position affects the speed of the mater spindle, is defined using channel-specific machine data:

MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function)

The function G96, G961 or G962 requires that the machine zero and the workpiece zero of the transverse axis are in the turning center of the spindle.

Constant speed (G97, G971)

G97, G971 deactivates the "Constant cutting rate function" (G96, G961) and saves the last calculated spindle speed. With G97, the feedrate is interpreted as a revolutional feedrate (as with G95). When programming G971, linear feedrate is selected (as with G94). The feedrate F is interpreted in mm/min or inch/min.

When G97, G971 is active, an S value can be reprogrammed to define a new spindle speed. This will not modify the cutting rate programmed in G96, G961.

G97, G971 can be used to avoid speed variations in motions along the transverse axis without machining (e.g. cutting tool).

Note

G96, G961 is only active during workpiece machining (G1, G2, G3, spline interpolation, etc., where feedrate F is active).

The response of the spindle speed for active G96, G961 and G0 blocks can be defined in the 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 override switch acts on the calculated spindle speed.

A DRF offset in the transverse axis does not affect the spindle speed setpoint calculation.

At the start of machining (after G0) and after NC stop, G60, G09, ... the path start waits for "nAct= nSet".

The interface signals "nAct = nSet" and "Set speed limited" are not modified by internal speed settings.

When the speed falls below the minimum speed or if the signal "Axis/spindle stationary" is recognized, "nAct =nSet" is reset.

A path operation which has started (G64, rounding), is not interrupted.

Spindle speed limitation with G96, G961

A maximum spindle speed can be specified for the "Constant cutting rate" function:

- In the setting data:
  SD43230 $SA_SPIND_MAX_VELO_LIMS (spindle speed limitation for G96/G961)
- In the part program (for the master spindle) with the programming command LIMS

The most recently changed value (LIMS or SD) is active.

LIMS is effective with G96, G961, G97 and can be specified on up to four speed limitations in the part program in one block. Spindle number <Sn> = 1, 2, 3, or 4 of the master spindle that is possible in the particular instance can be programmed in part program command LM[<Sn>].

When the block is loaded in the main run, all programmed values are transferred to the setting data SD43230 $SA_SPIND_MAX_VELO_LIMS.

Depending on the machine data:
MD10710 PROG_SD_RESET_SAVE_TAB[n] (setting data to be updated),
the speed limit set with LIMS remains stored after the control is switched off.
When G96, G961, G97 are reactivated, this speed limitation is also activated.

The maximum permissible spindle speed defined via G26 or via the setting data:
SD43220 $SA_SPIND_MAX_VELO_G26 (maximum spindle speed)
cannot be exceeded.

In the event of incorrect programming that would cause one of the speed limits (G26 or SD43220 $SA_SPIND_MAX_VELO_G26) to be exceeded, the following interface signal is set.

DB31, ... DBX83.1 (programmed speed too high)

In order to ensure smooth rotation with large part diameters, the spindle speed is not permitted to fall below a minimum level.
This speed can be set via the setting data:
SD43210 $SA_SPIND_MIN_VELO_G25 (minimum spindle speed)
and, depending on the gear stage, with the machine data:
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT (minimum speed of the gear stage)
.

The minimum spindle speed can be changed in the part program with G25. In the event of incorrect programming that would cause one of the speed limits (G25 or SD43210 $SA_SPIND_MIN_VELO_G25) to be undershot, the following interface signal is set.

DB31, ... DBX83.2 (speed setpoint too low)

For more information on the spindle-speed limitations, see function description S1: "Spindles", Section: "Spindle monitoring (Page 1366)".

Note

The speed limits changed with G25/G26/LIMS in the part program are taken into the setting data and therefore remain saved after the end of program.

However, if the speed limits changed with G25/G26/LIMS are no longer to apply after the end of program, the following definitions must be inserted in the GUD block of the machine manufacturer:

REDEF $SA_SPIND_MIN_VELO_G25 PRLOC

REDEF $SA_SPIND_MAX_VELO_G26 PRLOC

REDEF $SA_SPIND_MAX_VELO_LIMS PRLOC

Master spindle changeover with G96, G961

If the master spindle is switched over when G96, G961 are active, the speed of the former master spindle is retained. This corresponds to a transition from G96 to G97. The master spindle newly defined with SETMS executes the "Constant cutting rate" function generated in this way.

Alarms

Constant cutting rate G96, G961, G962

- If no F value is programmed, alarm 10860 "No feedrate programmed" is output. The alarm is not generated with G0 blocks.
- Alarm 14800 "Programmed path velocity smaller than or equal to zero" is output while programming a negative path velocity.
- If, with an active G96, G961 or G962, no transverse axis is defined in the machine data: MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function), alarm 10870 "No transverse axis defined" is output.

- If a negative maximum spindle speed is programmed with the `LIMS` program command when G96, G961 are active, alarm 14820 "Negative maximum spindle speed programmed for G96, G961" is output.
- If no constant cutting rate is programmed when G96, G961 is selected for the first time, alarm 10900 "No S value programmed for constant cutting rate" is output.

18.2.3 Feedrate for thread cutting (G33, G34, G35, G335, G336)

18.2.3.1 Feedrate with G33

G33

The function G33 can be used to machine threads with constant pitch of the following type:

Speed S, feedrate F, thread pitch

A revolutional feedrate [mm/revolution] is used for G33 threads. The revolutional feedrate is defined by programming the thread pitch [mm/revolution].

The speed of the axes for the thread length is calculated from the programmed spindle speed S and the thread pitch.

Feedrate F [mm/min] = speed S [rev/min] * pitch [mm/rev]

At the end of the acceleration ramp, the position coupling between the spindle actual value (spindle setpoint with SPCON on master spindle) and the axis setpoint is established. At this moment, the position of the axis in relation to the zero mark of the spindle (including zero mark offsets) is as if the axis had accelerated abruptly at the start of the block when the thread start position (zero mark plus SF) was crossed. Compensation is made for the following error of the axis.

Minimum spindle speed

In order to ensure smooth rotation at low speeds, the spindle speed is not permitted to fall below a minimum level.

This speed can be set:

- With the setting data:
  SD43210 $SA_SPIND_MIN_VELO_G25 (minimum spindle speed)
- For each gear stage with the machine data:
  MD35140 $MA_GEAR _STEP_MIN_VELO_LIMIT (minimum speed for gear stage change)

The minimum spindle speed can be changed in the part program with G25.

NC stop, single block

NC stop and single block (even at the block boundary) are only active after completion of thread chaining. All successive G33 blocks and the first following non-G33 block are traversed as a block.

Premature abortion without destruction

Thread cutting can be aborted without destruction before the end point is reached. This can be done by activating a retraction motion.

Thread cutting with ROT frame

With ROT frame and G33, G34, G35, alarm 10607 "Thread with frame not executable" is activated if the rotation causes a change in the thread length and thus the pitch. Rotation around the thread axis is permissible.

Alarm 10607 "Thread with frame not executable" can be suppressed by setting bit 12 in machine data MD11410 $MN_SUPPRESS_ALARM_MASK, if the ROT instruction is used intentionally in the application.

All other frames are accepted by the NC without alarm. Attention is drawn to the pitch-changing effect of SCALE.

18.2.3.2 Linear increasing/decreasing thread pitch change with G34 and G35

Function

The thread pitch increase (G34) defines the numerical increase in the pitch value. A larger pitch results in a larger distance between the threads on the workpieces. The velocity of the thread axis therefore increases with assumed constant spindle speed.

The opposite therefore applies for the decrease in thread pitch (G35).

The following definitions are made for the thread pitch change:

- G34: Increase in thread pitch corresponds to progressive change
- G35: Decrease in thread pitch corresponds to degressive change

Both G34 and G35 functions imply the functionality of G33 and also provide the option of programming an absolute pitch change value for the thread under F. If the start and end pitch of a thread is known, the thread pitch change can be determined using the following equation:

$$F = \frac{|k_e^2 - k_a^2|}{2 * l_G}$$

The meaning is as follows:

F:	The thread pitch change to be programmed [mm/rev^2]
k_e:	Thread pitch of axis target point coordinate, thread axis [mm/rev]
k_a:	Initial thread pitch (programmed under I, J or K) [mm/rev]
l_G:	Thread length [mm]

The absolute value of F must be applied to G34 or G35 depending on the required pitch increase of decrease.

When the thread length l_G, pitch change F and initial pitch k_a are known, the pitch increase at the end of block k_e can be determined as follows by modifying the formula:

- For G34 (increasing pitch):

$$k_e = \sqrt{k_a^2 + F * 2 * l_G}$$

- For G35 (decreasing pitch):

$$k_e = \sqrt{k_a^2 - F * 2 * l_G}$$

Note

If the formula results in a negative root expression, the thread cannot be machined!

In this case, the NC signals alarm 10605 or alarm 22275.

Application

The G34 and G35 functions can be used to produce self-shearing threads.

Example

Thread cutting G33 with decreasing thread pitch G35

Program code	Comment
`N1608 M3 S10`	`; Spindle speed`
`N1609 G0 G64 Z40 X216`	`; Approach starting point`
`N1610 G33 Z0 K100 SF=R14`	`; Thread with constant pitch 100 mm/rev`
`N1611 G35 Z-220 K100 F17.045455`	`; Thread pitch decrease 17.045455 mm/rev2`
	`; Thread pitch at end of block 50 mm/rev`
`N1612 G33 Z-240 K50`	`; Traverse thread block without jerk`
`N1613 G0 X218`	

Program code	Comment
N1614 G0 Z40	
N1616 M17	

Monitoring during the block preparation

Any pitch changes that would overload the thread axis when G34 is active or would result in an axis standstill when G35 is active, are detected in advance during block preparation. Alarm 10604 "Thread pitch increase too high" or 10605 "Thread pitch decrease too high" is signaled.

During thread cutting, certain practical applications require a correction of the spindle speed. In this case, the operator will base his correction on the permissible velocity of the thread axis.

To do this, it is possible to suppress the output of alarms 10604 and 10605 as follows:

MD11410 $MN_SUPPRESS_ALARM_MASK bit 10 = 1

Block preparation is then continued normally.

Monitoring during the execution

The following situations are monitored cyclically when the thread is machined (interpolation):

- Exceeding of maximum velocity of thread axis
- Reaching of axis standstill with G35

The following alarm is signaled when the monitoring function responds:

- Alarm 22269 "Maximum velocity of thread axis reached" or
- Alarm 22275 "Zero velocity of thread axis reached"

18.2.3.3 Acceleration behavior of the axis for G33, G34 and G35

The acceleration behavior of the feed axis when thread cutting/tapping with G33, G34 or G35 can be set via the channel-specific setting data:

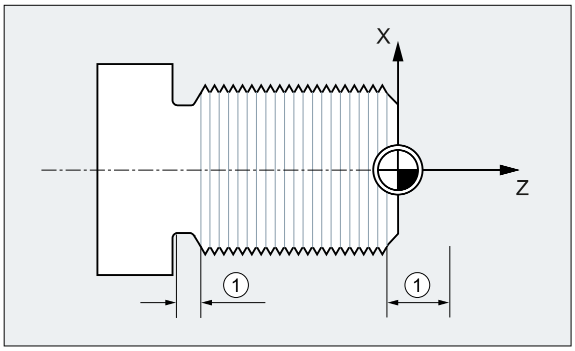
SD42010 $SC_THREAD_RAMP_DISP[<n>] = <Value>

<n>	Meaning	<Value>	Meaning
0	Acceleration behavior at the thread run-in:	< 0 (Default: -1)	The acceleration of the thread axis when running-in the thread is as programmed in `BRISK`/`SOFT`
		= 0	The acceleration of the thread axis when running-in the thread is sudden (step function) (≙ `BRISK`).
		> 0	The maximum thread run-in distance is specified. Specifying the maximum run-in thread path can also be specified via address `DITS` also in the part program (see "Programmed run-in and run-out path for G33, G34 and G35 (DITS, DITE) (Page 1404)"). When a block is inserted in the main run, the programmed run-in path is transferred into the setting data. **Note** Too short a distance can result in the axis being overloaded when accelerating.
1	Acceleration behavior at thread run-out:	< 0 (Default: -1)	The braking of the thread axis when running-ou the thread is as programmed in `BRISK`/ `SOFT`
		= 0	The braking of the thread axis when running-out the thread is sudden (step function) (≙ `BRISK`).
		> 0	The maximum thread run-out distance is specified. Specifying the maximum run-out thread path can also be specified via address `DITE` also in the part program (see "Programmed run-in and run-out path for G33, G34 and G35 (DITS, DITE) (Page 1404)"). When a block is inserted in the main run, the programmed run-out path is transferred into the setting data. **Note** Too short a distance can result in an acceleration overload of the axis.
2	Acceleration behavior for thread transitions within a thread chain	= -1 (default)	Permits smoothing corners between two thread blocks in order to maintain dynamic response limits. Rounding using smoothing is not carried out if the dynamic performance of the machine allows a hard transition, e.g. as a result of the effectiveness of the overload factor for axial velocity step/jumps (MD32310 $MA_MAX_ACCEL_OVL_FACTOR).
		= 0	Corners between two thread blocks are executed precisely without rounding - the axes follow the control loop commands.
		> 0	Reserved

18.2.3.4 Programmed run-in and run-out path for G33, G34 and G35 (DITS, DITE)

The run-in and run-out path of the thread can be specified in the part program with the `DITS` and `DITE` addresses.

The thread axis is accelerated or braked along the specified path.



① Run-in/run-out path, depending on the machining direction

Short run-in path

Due to the collar on the thread runin, little room is left for the tool start ramp.
This must therefore be specified shorter via `DITS`.

Short run-out path

Because of the shoulder at the thread run-out, there is not much room for the tool braking ramp, introducing a risk of collision between the workpiece and the tool cutting edge. The deceleration ramp can be specified shorter using `DITE`. Due to the inertia of the mechanical system, however, a collision can still occur.

Remedy: Program a shorter thread, reduce the spindle speed.

Note

`DITE` acts at the end of the thread as a rounding clearance. This achieves a smooth change in the axis motion.

Effects

The programmed run-in and run-out path only increases the rate of acceleration on the path. If one of the two paths is set larger than the thread axis needs with active acceleration, the thread axis is accelerated or decelerated with maximum acceleration.

Syntax

```
DITS=<Value> DITE=<Value>
```

Meaning

`DITS`:	Define thread run-in path
`DITE`:	Define thread run-out path
`<value>`:	Only paths, and not positions, are programmed with `DITS` and `DITE`. The programmed run-in/run-out path is handled according to the current dimension setting (inches, metric).

Example

Program code	Comment
`...`	
`N40 G90 G0 Z100 X10 SOFT M3 S500`	
`N50 G33 Z50 K5 SF=180 DITS=1 DITE=3`	`; Start of smoothing with Z=53.`
`N60 G0 X20`	

Further information

SD42010 $SC_THREAD_RAMP_DISP

When a block containing `DITS` and/or `DITE` is inserted in the main run, the programmed run-in/run-out path is transferred into the setting data SD42010 $SC_THREAD_RAMP_DISP:

- SD42010 $SC_THREAD_RAMP_DISP[**0**] = programmed value of `DITS`
- SD42010 $SC_THREAD_RAMP_DISP[**1**] = programmed value of `DITE`

If no run-in/run-out path is programmed before or in the first thread block, the current value of the setting data is used.

Behavior following channel / mode group / program end reset

SD 42010 values which have been overwritten by `DITS` and/or `DITE` remain active even following a channel / mode group / program end reset.

Behavior following warm start

In case of a warm start, the setting data is reset to the values which were active before overwriting by `DITS` and/or `DITE` (standard behavior).

If, however, the values programmed with `DITS` and `DITE` shall also be active following a warm restart, the setting data SD42010 $SC_THREAD_RAMP_DISP must be listed in the machine data MD10710 $MN_PROG_SD_RESET_SAVE_TAB:

MD10710 $MN_PROG_SD_RESET_SAVE_TAB[<n>] = 42010

Behavior if the run-in and/or run-out path is very short

If the run-in and/or run-out path is very short, the acceleration of the thread axis is higher than the configured value. This causes an acceleration overload on the axis.

Alarm 22280 "Programmed run-in path too short" is then issued for the thread run-in (with the appropriate configuration in MD11411 $MN_ENABLE_ALARM_MASK). The alarm is purely for information and has no effect on part program execution.

See also

Acceleration behavior of the axis for G33, G34 and G35 (Page 1402)

18.2.3.5 Fast retraction during thread cutting

Function

The "Rapid retraction during thread cutting" function can be used to interrupt thread cutting without causing irreparable damage in the following circumstances:

- NC stop (NC/PLC interface signal)
- Alarms that implicitly trigger NC stop
- Switching of a rapid input
  References
  Programming Manual, Job Planning; Section "Fast retraction from the contour"

The retraction motion can be programmed via:

- Retraction path and retraction direction (relative)
- Retraction position (absolute)

Note

Tapping

The "Fast retraction" function **cannot** be used with tapping (`G331`/`G332`).

Programming

Syntax

Enable fast retraction, retraction motion via retraction path and retraction direction:
```
G33 ... LFON DILF=<Value> LFTXT/LFWP ALF=<Value>
```

Enable fast retraction, retraction motion via retraction position:

```
POLF[<Axis name>]=<Value> LFPOS
POLFMASK/POLFMLIN(<Axis 1 name>,<Axis 2 name>, etc.)
G33 ... LFON
```

Disable fast retraction for thread cutting:
```
LFOF
```

Meaning

`LFON`:	Enable fast retraction for thread cutting (`G33`).
`LFOF`:	Disable fast retraction for thread cutting (`G33`).

`DILF=`: Define length of retraction path.

The value preset during MD configuration (MD21200 $MC_LIFTFAST_DIST) can be modified in the part program by programming `DILF`.

Note:
The configured MD value is always active following NC-RESET.

`LFTXT`
`LFWP`: The retraction direction is controlled in conjunction with `ALF` with G commands `LFTXT` and `LFWP`.

`LFTXT`:	The plane in which the retraction motion is executed is calculated from the path tangent and the tool direction (default setting).
`LFWP`:	The plane in which the retraction motion is executed is the active working plane.

`ALF=`: The direction is programmed in discrete degree increments with `ALF` in the plane of the retraction motion.

With `LFTXT`, retraction in the tool direction is defined for `ALF=1`.

With `LFWP`, the direction in the working plane is derived from the following assignment:

- `G17` (X/Y plane)
  `ALF=1` ; Retraction in the X direction
  `ALF=3` ; Retraction in the Y direction
- `G18` (Z/X plane)
  `ALF=1` ; Retraction in the Z direction
  `ALF=3` ; Retraction in the X direction
- `G19` (Y/Z plane)
  `ALF=1` ; Retraction in the Y direction

`ALF=3` ; Retraction in the Z direction

References:
Programming options with `ALF` are also described in "Traverse direction for fast retraction from the contour" in the Programming Manual, Job Planning.

`LFPOS`: Retraction of the axis declared with `POLFMASK` or `POLFMLIN` to the absolute axis position programmed with `POLF`.

`POLFMASK`: Release of axes `(<Axis 1 name>,<Axis 1 name>, etc.)` for independent retraction to absolute position.

`POLFMLIN`: Release of axes for retraction to absolute position in linear relation

Note:
Depending on the dynamic response of all the axes involved, the linear relation cannot always be established before the lift position is reached.

`POLF[]`: Define absolute retraction position for the geometry axis or machine axis in the index

Effectiveness: Modal

`=<Value>`: In the case of geometry axes, the assigned value is interpreted as a position in the workpiece coordinate system. In the case of machine axes, it is interpreted as a position in the machine coordinate system.

The values assigned can also be programmed as incremental dimensions:

`=IC<Value>`

`<Axis name>`: Name of a geometry axis or machine axis

Note

`LFON` or `LFOF` can always be programmed, but the evaluation is performed exclusively during thread cutting (`G33`).

Note

`POLF` with `POLFMASK`/`POLFMLIN` are not restricted to thread cutting applications.

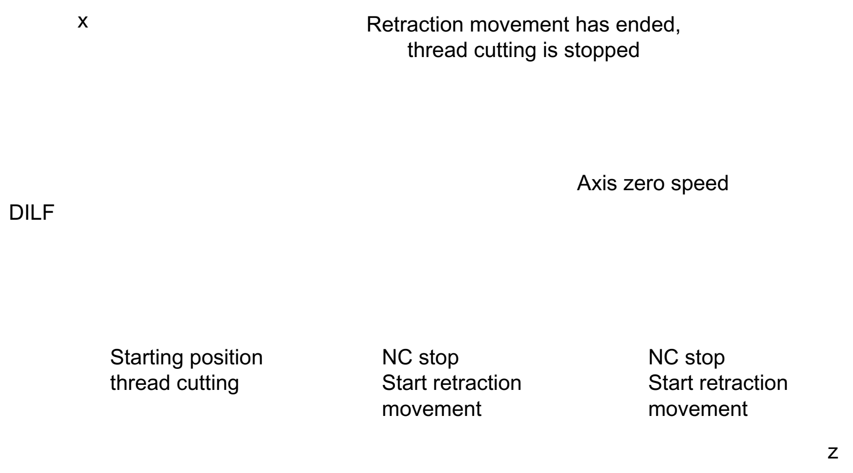

x
Retraction movement has ended, thread cutting is stopped
Axis zero speed
DILF
Starting position thread cutting
NC stop Start retraction movement
NC stop Start retraction movement
z

Figure 18-1 Interruption of G33 through retraction motion

Dynamic response of the retraction motion

The retraction motion is executed with maximum axis dynamic response:

- MD32000 $MA_MAX_AX_VELO[<Axis>] (velocity)
- MD32300 $MA_MAX_AX_ACCEL[<Axis>] (acceleration)
- MD32431 $MA_MAX_AX_JERK[<Axis>] (jerk)

Example

Program code	Comment
N55 M3 S500 G90 G18	; Set active machining plane.
...	
N65 MSG ("thread cutting")	
MM_THREAD:	
N67 $AC_LIFTFAST=0	; Reset before starting the thread.
N68 G0 Z5	
N69 X10	
N70 G33 Z30 K5 LFON DILF=10 LFWP ALF=7	; Enable fast retraction for thread cutting.
	; Retraction path = 10 mm
	; Retraction plane Z/X (because of G18).
	; Retraction direction -X (with ALF=3; retraction direction +X).
N71 G33 Z55 X15	
N72 G1	; Deselect thread cutting.
N69 IF $AC_LIFTFAST GOTOB MM_THREAD	; If thread cutting has been interrupted.
N90 MSG ("")	
...	
N70 M30	
N55 M3 S500 G90 G0 X0 Z0	
...	
N87 MSG ("tapping")	
N88 LFOF	; Deactivate fast retraction before tapping.
N89 CYCLE...	; Tapping cycle with G33.
N90 MSG ("")	
...	
N99 M30	

Behavior at power on and reset

After power on and reset, the following settings are activated:

- Initial settings for the retraction motion (`LFON` /`LFOF`) and retraction direction (`LFTXT`/ `LFWP`): MD20150 $MC_GCODE_RESET_VALUES
- Retraction path: MD21200 $MC_LIFTFAST_DIST

18.2.3.6 Convex thread (G335, G336)

Function

The G commands G335 and G336 can be used to turn convex threads (= differing to the cylindrical form). Application is the machining of extremely large components that sag in the machine because of their self-weight. Paraxial thread would result in the thread being too small in the middle of the component. This can be compensated with convex threads.

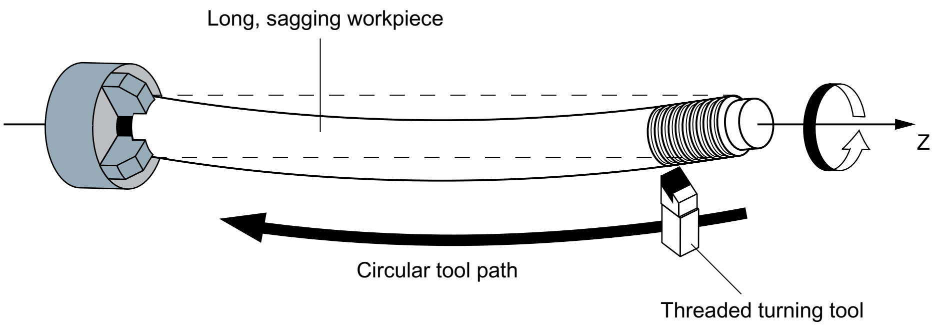


Figure 18-2 Turning a convex thread

Programming

The turning of a convex thread is programmed with `G335` or `G336`:

`G335`:	Turning of a convex thread on a circular tool path **in a clockwise direction**
`G336`:	Turning of a convex thread on a circular tool path **in a counter-clockwise direction**

The programming is performed first as for a linear thread by specifying the axial block end points and the pitch via parameters `I`, `J` and `K`.

An arc is also specified. As for `G2`/`G3`, this can be programmed via the center point, radius, opening angle or intermediate point specification. When programming the convex thread with center point programming, the following must be taken into account: Since `I`, `J` and `K` are used for the pitch in thread cutting, the circle parameters in the center point programming must be programmed with `IR=...`, `JR=...` and `KR=...`.

`IR=...`:	Cartesian coordinate for the circle center point in the X direction
`JR=...`:	Cartesian coordinate for the circle center point in the Y direction
`KR=...`:	Cartesian coordinate for the circle center point in the Z direction

Note

`IR`, `JR` and `KR` are the default values of the interpolation parameter names for a convex thread that can be set via machine data MD10651 $MN_IPO_PARAM_THREAD_NAME_TAB.

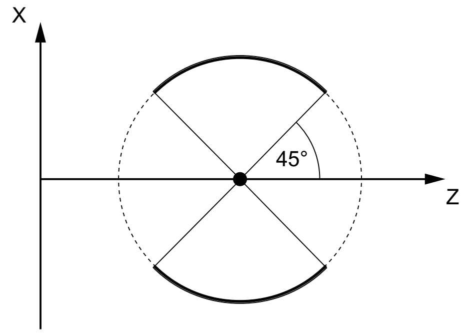
Optionally, a starting point offset `SF` can also be specified.

Syntax

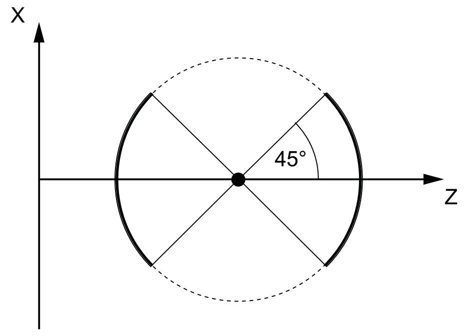
The syntax for the programming of a convex thread therefore has the following general form:
```
G335/G336 <axis target point coordinate(s)> <pitch> <arc> [<starting
point offset>]
```

Permissible arc areas

The arc programmed at `G335`/`G336` must be in an area in which the specified thread main axis (`I`, `J` or `K`) has the main axis share on the arc over the entire arc:



Permissible areas for the **Z** axis (pitch programmed with `K`)



Permissible areas for the **X** axis (pitch programmed with `I`)

A change of the thread main axis as shown in the following figure is **not** permitted:

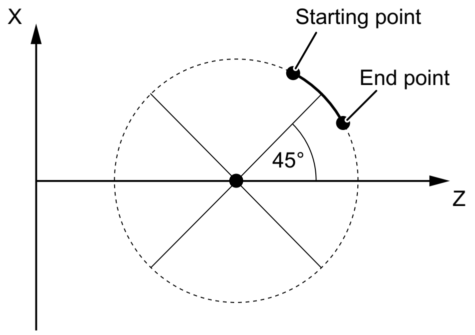


Figure 18-3 Convex thread: Area that is not permissible

Boundary conditions

Frames

G335 and G336 are also possible with active frames. However, you must ensure that the permissible arc areas are maintained in the basic coordinate system (BCS).

Response for ...

The G335/G336 response for:

- Power On / Power Off
- Mode change
- NCK / mode group / channel / part program end reset

- Block search / REPOS / ASUP
- Alarms / emergency stop / malfuncations

corresponds to the behavior for G33/G34/G35.
There are no specific restrictions.

Examples

Example 1: Convex thread in the clockwise direction with end and center point programming

Program code	**Comment**
`N5 G0 G18 X50 Z50`	`; Approach starting point.`
`N10` **`G335 Z100`** `K=3.5` **`KR=25 IR=-20`** `SF=90`	`; Turn convex thread in the clock-wise direction.`

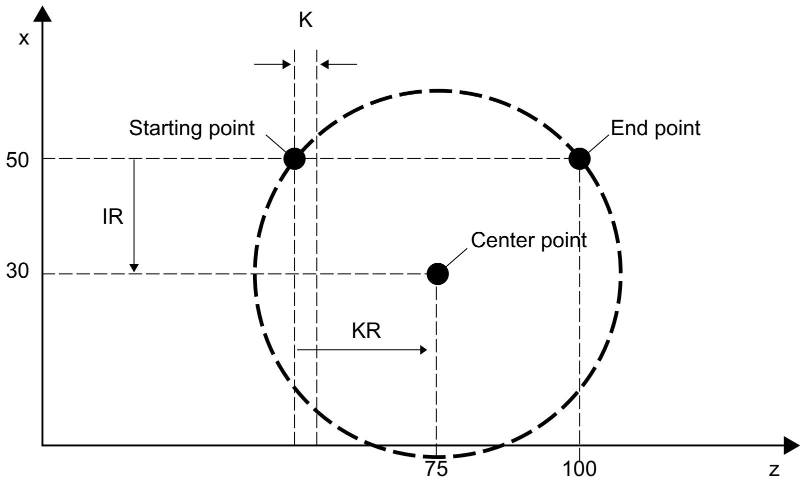


Figure 18-4 Convex thread in the clockwise direction with end and center point programming

Example 2: Convex thread in the counter-clockwise direction with end and center point programming

Program code	**Comment**
`N5 G0 G18 X50 Z50`	`; Approach starting point.`
`N10` **`G336 Z100`** `K=3.5` **`KR=25 IR=20`** `SF=90`	`; Turn convex thread in the coun-ter-clockwise direction.`

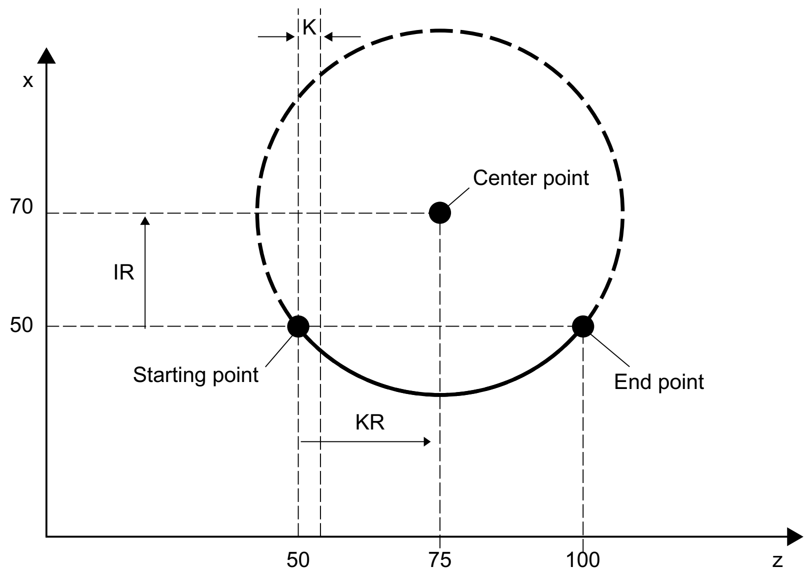


Figure 18-5 Convex thread in the counter-clockwise direction with end and center point programming

Example 3: Convex thread in the clockwise direction with end point and radius programming

Program code
N5 G0 G18 X50 Z50
N10 **G335 Z100** K=3.5 **CR=32** SF=90

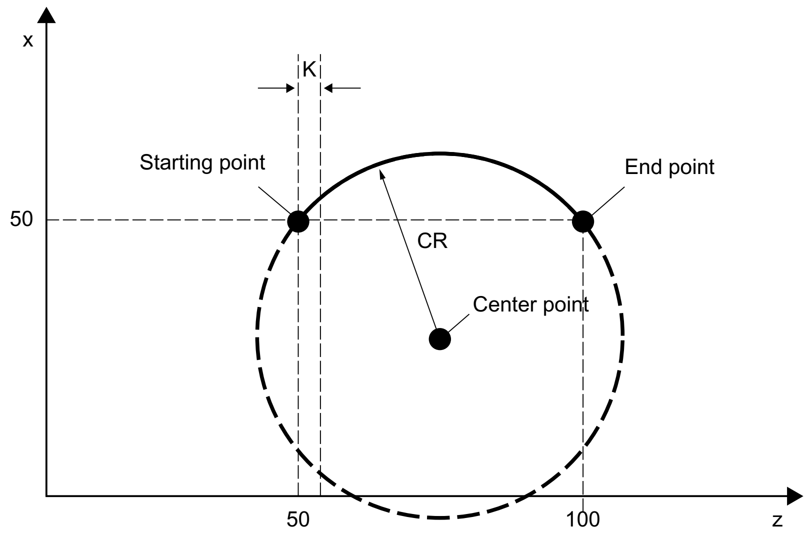


Figure 18-6 Convex thread in the clockwise direction with end point and radius programming

Example 4: Convex thread in the clockwise direction with end point and opening angle programming

Program code
N5 G0 G18 X50 Z50
N10 **G335 Z100** K=3.5 **AR=102.75** SF=90

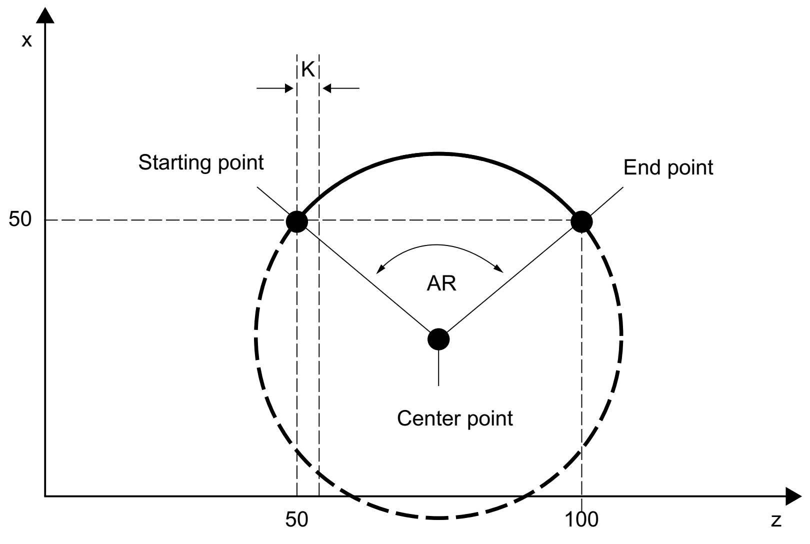


Figure 18-7 Convex thread in the clockwise direction with end point and opening angle programming

Example 5: Convex thread in the clockwise direction with center point and opening angle programming

```
Program code
N5 G0 G18 X50 Z50
N10 G335 K=3.5 KR=25 IR=-20 AR=102.75 SF=90
```

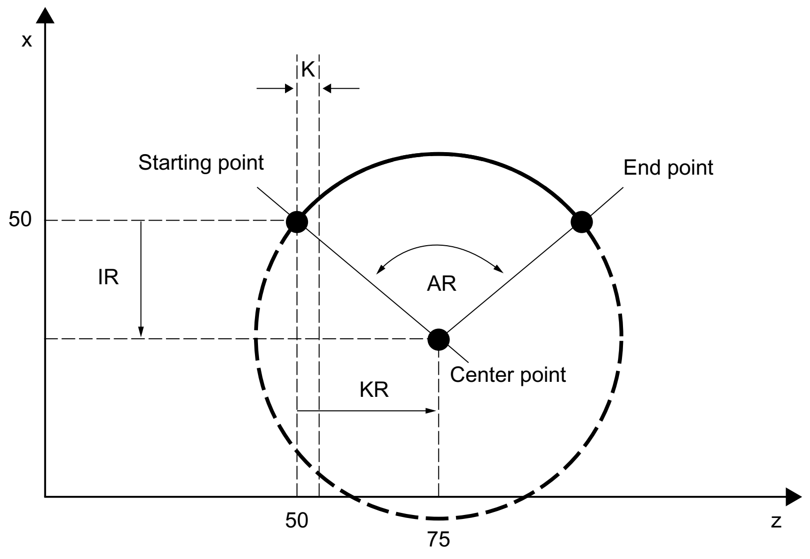


Figure 18-8 Convex thread in the clockwise direction with center point and opening angle programming

Example 6: Convex thread in the clockwise direction with end and intermediate point programming

```
Program code
N5 G0 G18 X50 Z50
N10 G335 Z100 K=3.5 I1=60 K1=64
```

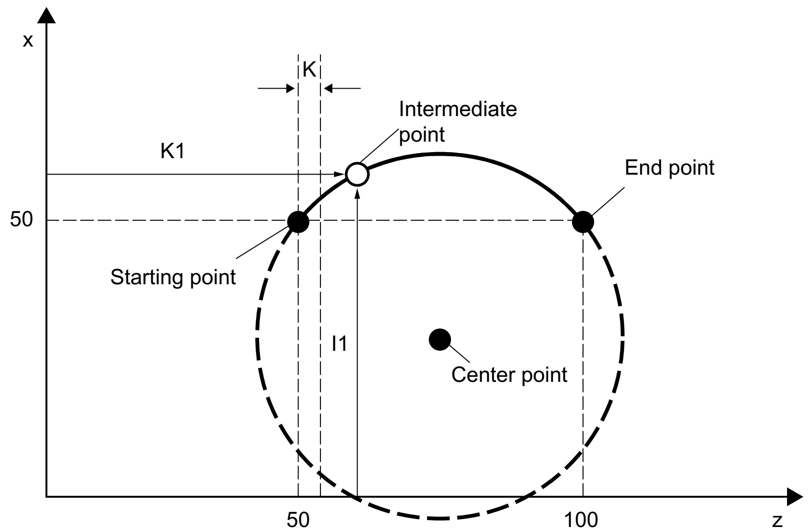


Figure 18-9 Convex thread in the clockwise direction with end and intermediate point programming

18.2.4 Feedrate for tapping without compensating chuck (G331, G332)

Function

A thread can be tapped by rigid tapping with the functions `G331` (tapping) and `G332` (tapping retraction).

Path feedrate

With `G331` / `G332`, the path feedrate F for the axes involved in tapping is derived from the effective spindle speed S and the programmed pitch:

```
F [mm/min] = S [rpm] * thread pitch [mm/U]
```

Requirement

The requirement for rigid tapping is a **position-controlled spindle** with position measuring system.

Machine data

- Preventing stop events
  In the machine data, the stopping response when `G331` / `G332` is active is defined:
  MD11550 $MN_STOP_MODE_MASK, Bit <x> = <value>

Bit	Value	Meaning
0	0	When `G331` / `G332` is active, stopping is **not** performed during a path motion or dwell time (G4).
	1	When G331 / G332 is active, stopping is performed during interruption of path control mode, for example, by G60 or G4.

If you always want to prevent stopping the traverse movement when `G331` / `G332` is active, this area must be declared a stop delay area with the commands `DELAYFSTON` and `DELAYFSTOF`.
References:
Function Manual, Basic Functions, Section "Influencing the stop events by stop delay areas (Page 581)"

Note

Single block

If the single block is activated in the stop delay area, the NC stops at the end of the first block outside the stop delay area. If the single block is already selected before the stop delay area, the NC stops at each block limit, i.e. also in the stop delay area! This deselects the stop delay area.

Override changes

If the override is changed **before** a stop delay area, the override takes effect **in** the stop delay area.

If the override is changed **in** the stop delay area, the change takes effect **after** the stop delay area.

- Override
  Which override is active during rigid tapping is set in:
  MD12090 $MN_OVR_FUNCTION_MASK, Bit <x> = <value>

Bit	Value	Meaning
0	0	The **spindle override** is active for G331/G332 Depending on the setting in the following machine data, the override is related either to the programmed spindle speed or to the configured spindle speed limitation: MD12080 $MN_OVR_REFERENCE_IS_PROG_FEED
	1	The **path override** is active for G331/G332 Depending on the setting in the following machine data, the override is related either to the programmed path feedrate or to the configured path feedrate limitation. MD12082 $MN_OVR_REFERENCE_IS_MIN_FEED **Note** This only applies if the limitation value is lower than the programmed path feedrate.

Supplementary conditions

Further overrides

The following overrides are inactive during rigid tapping

- Programmable path feedrate override `OVR`
- Rapid traverse override

18.2.5 Feedrate for tapping with compensating chuck (G63)

Function

G63 is a subfunction for tapping threads using a tap with compensating chuck. An encoder (position encoder) is not required.

Speed S, feedrate F, thread pitch

With G63, a speed S must be programmed for the spindle and a feedrate F for the infeed axis (axis for thread length).

The feedrate F must be calculated by the programmer on the basis of the speed S and the thread pitch.

Feedrate F [mm/min] = speed S [rev/min] * pitch [mm/rev]

References

For more information on G63, see Programming Manual Fundamentals.

18.2.6 FGROUP and FGREF

Programming

It should be possible to program the effective machining feedrate in the usual way as a path feedrate via the F value in processing procedures where the tool, the workpiece or both are moved by a rotary axis (e.g. laser machining of rotating tubes).

In order to achieve this, it is necessary to specify an effective radius (reference radius) for each of the rotary axes involved. You can do this by programming the modal NC address:
`FGREF[<rotary axis>]=<reference radius>`

The unit of the reference radius depends on the `G70`/`G71`/`G700`/`G710` setting.

In order to include the axes in the calculation of the path feedrate, they must all be specified in the `FGROUP` command.

In order to ensure compatibility with the behavior with no `FGREF` programming, the evaluation 1 degree = 1 mm is activated on system powerup and RESET.

This corresponds to a reference radius of:

`FGREF` = 360 mm / (2π) = 57.296 mm

This default is independent of the active basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) and the currently active `G70`/`G71`/`G700`/`G710` setting.

Special features of the feedrate weighting for rotary axes in FGROUP:

Program code
N100 FGROUP(X,Y,Z,A)
N110 G1 G91 A10 F100
N120 G1 G91 A10 X0.0001 F100

The programmed F value in block `N110` is evaluated as a rotary axis feedrate in degrees/min, while the feedrate weighting in block `N120` is either 100 inch/min or 100 mm/min, depending on the current inch/metric setting.

NOTICE

Different systems of units

The `FGREF` factor also works if only rotary axes are programmed in the block. The normal `F` value interpretation as degree/min applies in this case only if the radius reference corresponds to the `FGREF` default:

- For `G71`/`G710`: `FGREF[A]=57.296`
- For `G70`/`G700`: `FGREF[A]=57.296/25.4`

Example

The following example is intended to demonstrate the effect of `FGROUP` on the path and path feedrate. The variable `$AC_TIME` contains the time of the block start in seconds. It can only be used in synchronized actions.

Program code	Comment
N100 G0 X0 A0	
N110 FGROUP(X,A)	
N120 G91 G1 G710 F100	; Feedrate = 100 mm/min or 100 degrees/min
N130 DO $R1=$AC_TIME	
N140 X10	; Feedrate = 100 mm/min, path = 10 mm, R1 = approx. 6 s
N150 DO $R2=$AC_TIME	
N160 X10 A10	; Feedrate = 100 mm/min, path = 14.14 mm, R2 = approx. 8 s
N170 DO $R3=$AC_TIME	
N180 A10	; Feedrate = 100 degrees/min, path = 10 degrees, R3 = approx. 6 s
N190 DO $R4=$AC_TIME	
N200 X0.001 A10	; Feedrate = 100 mm/min, path = 10 mm, R4 = approx. 6 s
N210 G700 F100	; Feedrate = 2540 mm/min or 100 degrees/min

Program code	Comment
`N220 DO $R5=$AC_TIME`	
`N230 X10`	`; Feedrate = 2540 mm/min, path = 254 mm, R5 = approx. 6 s`
`N240 DO $R6=$AC_TIME`	
`N250 X10 A10`	`; Feedrate = 2540 mm/min, path = 254.2 mm, R6 = approx. 6 s`
`N260 DO $R7=$AC_TIME`	
`N270 A10`	`; Feedrate = 100 degrees/min, path = 10 degrees, R7 = approx. 6 s`
`N280 DO $R8=$AC_TIME`	
`N290 X0.001 A10`	`; Feedrate = 2540 mm/min, path = 10 mm, R8 = approx. 0.288 s`
`N300 FGREF[A]=360/(2*$PI)`	`; Set 1 degree = 1 inch via the effective radius`
`N310 DO $R9=$AC_TIME`	
`N320 X0.001 A10`	`; Feedrate = 2540 mm/min, path = 254 mm, R9 = approx. 6 s`
`N330 M30`	

Diagnostics

Read reference radius

The value of the reference radius of a rotary axis can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variable:

$AA_FGREF[<axis>]	Current main run value

- Without preprocessing stop in the part program via the system variable:

$PA_FGREF[<axis>]	Programmed value

If no values are programmed, the default 360 mm / (2π) = 57.296 mm (corresponding to 1 mm per degree) will be read in both variables.

For linear axes, the value in both variables is always 1 mm.

Read path axes affecting velocity

The axes involved in path interpolation can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

$AA_FGROUP[<axis>]	Returns the value "1" if the specified axis affects the path velocity in the current main run record by means of the basic setting or through `FGROUP` programming. Otherwise, the variable returns the value "0".
$AC_FGROUP_MASK	Returns a bit key of the channel axes programmed with `FGROUP` which are to affect the path velocity.

- Without preprocessing stop in the part program via system variables:

$PA_FGROUP[<axis>]	Returns the value "1" if the specified axis affects the path velocity by means of the basic setting or through `FGROUP` programming. Otherwise, the variable returns the value "0".
$P_FGROUP_MASK	Returns a bit key of the channel axes programmed with `FGROUP` which are to affect the path velocity.

18.3 Feedrate for positioning axes (FA)

Function

The velocity of a positioning axis is programmed with axis-specific feedrate `FA`.

FA is modal.

The feedrate is always G94.

Note

The maximum axis velocity (MD32000 $MA_MAX_AX_VELO) is not exceeded.

Programming

No more than five axis-specific feedrates can be programmed in each part program block.

Syntax:

```
FA[<positioning axis>] = <feedrate value>
```

`<positioning axis>`:	Name of the channel axis (MD20080 $MC_AXCONF_CHANAX_NAME_TAB)	
`<feedrate value>`:	Feedrate	
	Value range:	0.001…999 999.999 mm/min, deg/min or 0.001…39 999.9999 inch/min

Default setting

If no axial feedrate `FA` is programmed, the axial default setting is applied:

MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)

Output to PLC

The feedrate value can be output to the the PLC:

- To the channel-specific NC/PLC interface via:
  DB21, ... DBB158 - DBB193
- To the axis-specific NC/PLC interface via:
  DB31, ... DBB78 - DBB81

The output time is specified with the machine data:

MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

The output is suppressed in the default setting (MD22240 = 3), because drops in velocity can occur through the output of F functions to the NC/PLC interface in continuous-path mode.

For more information, see Description of Functions "H2: Auxiliary function outputs to PLC (Page 401)".

Reset behavior

The behavior after the end of program or NC reset is specified by the machine data:

MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)

Value	Meaning
0	The default values are effective after NC reset.
1	The last programmed FA values are effective after NC reset.

18.4 Feedrate control

18.4.1 Feedrate disable and feedrate/spindle stop

Function

The "Feed disable" or "Feed/spindle stop" brings the axes to a standstill with adherence to the braking characteristics With the exception for thread cutting `G33`, the programmed contour is maintained.

Channel-specific feedrate disable

All axes (geometry and special axes) of a channel are stopped in all modes using the channel-specific NC/PLC interface signal:

DB21, ... DBX6.0 (feedrate disable)

Channel-specific "Feed stop" for geometry axes in the JOG mode

The traversing motion of a specific geometry axis of the channel is stopped using the channel-specific NC/PLC interface signals:

- DB21, ... DBX12.3 (feedrate stop, geometry axis 1)
- DB21, ... DBX16.3 (feedrate stop, geometry axis 2)
- DB21, ... DBX20.3 (feedrate stop, geometry axis 3)

The interface signals are only active in the **JOG** mode.

Axis-specific "Feedrate stop/spindle stop" for machine axes

Traversing motion of the machine axis is stopped using the axis-specific NC/PLC interface signal:

DB31, ... DBX4.3 (feedrate stop / spindle stop)

AUTOMATIC and MDI modes

The following applies in the AUTOMATIC and MDI modes:

- 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.

Thread cutting

Thread cutting	Effectiveness
`G33`, `G34`, `G35`	Effective **Notice** Contour deviation
`G331`, `G332`	Effective
`G63`	Axis: Effective Spindle: **Not** effective

Axis-specific axis/spindle disable for machine axes

Traversing motion of the machine axis is stopped using the axis-specific NC/PLC interface signal:

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

18.4.2 Feedrate override via machine control panel

Function

With the "Feedrate override via machine control panel", the operator can locally increase or decrease the path feedrate at the machine as a percentage with immediate effect. To achieve this, the programmed feedrates are multiplied with the override values available at the NC/PLC interface.

The feedrate can be changed axis-specifically for positioning axes.

The "Spindle override" can be used to change the spindle speed and the cutting rate (G96, G961).

With a feedrate change, the axial acceleration and velocity limits are maintained. There are no contour errors along the path.

The feedrate override can be changed separately for path and position axes.

The overrides influence the programmed values or the limits (e.g. G26, LIMS for spindle speed).

Channel-specific feedrate and rapid traverse override

For feedrate and rapid traverse override, dedicated enable signals and correction/offset factors are available in the NC/PLC interface:

DB21, ... DBX6.7 (feedrate override active)

DB21, ... DBB4 (feedrate override)

DB21, ... DBX6.6 (rapid traverse override active)

DB21, ... DBB5 (rapid traverse override)

The override factors can be specified from the PLC either in the binary or Gray-coded format. The format is communicated to the NC via the following machine data:

MD12020 $MN_OVR_FEED_IS_GRAY_CODE (path feedrate override switch Gray-coded)

MD12040 $MN_OVR_RAPID_IS_GRAY_CODE (rapid traverse override switch Gray-coded)

The following permanent assignment applies to binary code:

Binary code	Decimal	Override factor
00000000	0	0.00 ≙ 0%
00000001	1	0.01 ≙ 1%
00000010	2	0.02 ≙ 2%
00000011	3	0.03 ≙ 3%
00000100	4	0.04 ≙ 4%
...	...	...
01100100	100	1.00 ≙ 100%
...	...	...
11001000	200	2.00 ≙ 200%

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12030 $MN_OVR_FACTOR_FEEDRATE [<n>] (evaluation of the path feedrate override switch)

MD12050 $MN_OVR_FACTOR_RAPID_TRA [<n>] (evaluation of the rapid traverse override switch)

An active feedrate override acts on all path axes that are assigned to the current channel. An active rapid traverse override has an effect on all the axes that are traversed with rapid traverse and that are assigned to the current channel.

No rapid traverse override switch available

If there is no dedicated rapid traverse override switch, you can switch between rapid traverse override and feedrate override. The override to be active can be selected via the PLC or operator panel front. When rapid traverse override is active, the feedrate override values are limited to 100%.

- When the rapid traverse override is activated via the operator panel front, the basic PLC program:
  - Transfers the selection of the feedrate override for rapid traverse on the activation signal for the rapid traverse override:
    DB21, ... DBX6.6 = DB21, ... DBX25.3
  - Transfers the feedrate override value in the rapid traverse value
    DB21, ... DBB5 = DB21, ... DBB4
- When the rapid traverse override is selected via the PLC, the PLC user program:
  - Must set the activation signal for the rapid traverse override:
    DB21, ... DBX6.6 = 1
  - Must transfer the feedrate override value in the rapid traverse override value:
    DB21, ... DBB5 = DB21, ... DBB4

Effectiveness of the channel-specific feedrate and rapid traverse override:	
• With active G33, G34, G35:	**Not** effective
• With active G63:	**Not** effective
• With active G331, G332:	**Not** effective

Reference velocity for path feedrate override

The reference velocity for the "Path feedrate override via machine control panel" can be set differently to the standard feedrate (= programmed feedrate).

MD12082 $MN_OVR_REFERENCE_IS_MIN_FEED

Axis-specific feedrate override

An enable signal and a byte for the feedrate override factor are available in the NC/PLC interface for each positioning axis:

DB31, ... DBX1.7 (override effective)

DB31, ... DBB0 (feedrate override)

The override factor can be specified from the PLC either in the binary or Gray-coded format. The format is communicated to the NC via the following machine data:

MD12000 $MN_OVR_AX_IS_GRAY_CODE (axis feedrate override switch Gray-coded)

The following permanent assignment applies to binary code:

Binary code	Decimal	Override factor
00000000	0	0.00 ≙ 0%
00000001	1	0.01 ≙ 1%
00000010	2	0.02 ≙ 2%
00000011	3	0.03 ≙ 3%
00000100	4	0.04 ≙ 4%
...	...	...
01100100	100	1.00 ≙ 100%
...	...	...
11001000	200	2.00 ≙ 200%

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12010 $MN_OVR_ FACTOR_AX_ SPEED [<n>] (evaluation of the axis feedrate override switch)

Effectiveness of the axis-specific feedrate override:	
• With active G33, G34, G35:	**Not** effective
• With active G63:	**Not** effective (the override is set in the NC permanently to 100%)
• With active G331, G332:	**Not** effective (the override is set in the NC permanently to 100%)

Spindle override

An enable signal and a byte for the spindle override factor are available in the NC/PLC interface for each spindle:

DB31, ... DBX1.7 (override effective)

DB31, ... DBB19 (spindle override)

The override factor can be specified from the PLC either in the binary or Gray-coded format. The format is communicated to the NC via the following machine data:

MD12060 $MN_OVR_SPIND_IS_GRAY_CODE (spindle override switch Gray-coded)

The following permanent assignment applies to binary code:

Binary code	Decimal	Override factor
00000000	0	0.00 ≙ 0%
00000001	1	0.01 ≙ 1%
00000010	2	0.02 ≙ 2%
00000011	3	0.03 ≙ 3%

Binary code	Decimal	Override factor
00000100	4	0.04 ≙ 4%
...	...	...
01100100	100	1.00 ≙ 100%
...	...	...
11001000	200	2.00 ≙ 200%

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12070 $MN_OVR_FACTOR_SPIND_SPEED [<n>] (evaluation of the spindle override switch)

Effectiveness of the "spindle override":	
• With active G33, G34, G35:	Effective
• With active G63:	**Not** effective
• With active G331, G332:	Effective

Reference of the spindle override

The spindle override can refer to the speed or the programmed speed limited by the machine or setting data. The setting is realized via:

MD12080 $MN_OVR_REFERENCE_IS_PROG_FEED (override reference velocity)

Limiting the override factor

For binary-coded override factors, the maximum possible overrides for path feedrate, axis feedrate and spindle speed can be limited:

MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

Override active

For overrides that have been enabled, the specified override values entered via the machine control panel become immediately active in all operating modes and machine functions.

Override inactive

An override factor of 100% is internally effective if an override is not activated. The override factor at the NC/PLC interface is not evaluated.

An exception is the zero setting for a binary interface and the 1st switch setting for a Gray-coded interface. In these cases, the override factors entered at the NC/PLC interface are evaluated. For a binary interface, the override factor is always 0%. For a Gray-coded interface, the value entered in machine data for the 1st switch position value is output as override value. It should be assigned the value "0".

18.4.3 Programmable feedrate override

Function

The "Programmable feedrate override" function can be used to change the velocity level of path and positioning axes via the part program.

Programming

Syntax	Meaning
`OVR=<value>`	Feedrate change for path feedrate F
`OVRA[<axis>=<value>`	Feedrate change for positioning feedrate FA

The programmable range is between 0 and 200%.

Default setting: 100%

Effectiveness

The NC/PLC interface signals DB21, ... DBB6 (rapid traverse or feedrate override active) and DB31, ... DBX1.7 (axis-specific override active) do **not** refer to the programmable feedrate override. The programmable feedrate override remains active when these signals are deactivated.

The effective override is calculated from the product of the "Programmable feedrate override" and the "Feedrate override via machine control panel (Page 1423)".

The default setting for the "Programmable feedrate override" is 100%.

The default setting is effective:

- If no feedrate override is programmed or
- After reset if the machine data:
  MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset) is not set.

Note

OVR is not effective with G33, G34, G35.

18.4.4 Dry run feedrate

Function

The dry run feedrate is used when testing part programs without machining the workpiece in order to allow the program or program sections to execute with an increased path feedrate, for example.

Activation

The dry run feedrate can be selected in the automatic modes and activated from the PLC or the operator panel front.

When activated from the operator panel front, the interface signal:
DB21, ... DBX24.6 (dry run feedrate selected)
is set and transferred from the basic PLC program to the interface signal:
DB21, ... DBX0.6 (activate dry run feedrate).

When selected on the PLC, the interface signal DB21, ... DBX0.6 (activate dry run feedrate) must be set from the PLC user program.

Effectiveness

As long as the "Activate dry run feedrate" interface signal is set, instead of the programmed feedrate, the feedrate value set via SD42100 DRY_RUN_FEED is effective in the way specified via SD42101 $SC_DRY_RUN_FEED_MODE (see parameterization):

The dry run feedrate is always interpreted as linear feedrate (G94).

Parameterization

Activation of dry run feedrate

The time of activation depends on the setting in the machine data:

MD10704 $MN_DRYRUN_MASK (activation of dry run feedrate)

Value	Meaning
0	The dry run feedrate may only be switched on and off at the end of the block (default setting).
1	The dry run feedrate can also be activated during the program processing (in the part program block). **Notice:** Activation during processing triggers an internal reorganization operation on the controller which causes the axes to be stopped for a short time. This can affect the surface finish of the workpiece being machined.
2	The dry run feedrate can be activated/deactivated at any time without the axes being stopped. The function only takes effect with a block "later" in the program run.

Changing the dry run feedrate

The feedrate for the dry run is entered in the setting data:

SD42100 $SC_DRY_RUN_FEED (dry run feedrate)

The setting data can be changed via the operator panel front in the "Parameters" operating area.

If the selection has been accepted by the NC, the following NC/PLC interface signal is set:

DB21, ... DBX318.6 (dry run feedrate active)

"DRY" is displayed in the operator panel front status bar to indicate an active dry run feedrate if:

- Selection took place during the program stop at the end of a block or
- The machine data MD10704 $MN_DRYRUN_MASK was set to "1" during the program execution

Mode of operation of the dry run feedrate

The mode of operation of the dry run feedrate entered in SD42100 can be set via the setting data:

SD42101 $SC_DRY_RUN_FEED_MODE

Value	Meaning
0	The programmed feedrate is compared to the dry run feedrate in SD42100 and then traversing is performed with the higher of the two feedrates (default setting).
1	The programmed feedrate is compared to the dry run feedrate in SD42100 and then traversing is performed with the lower of the two feedrates.
2	The dry run feedrate entered in SD42100 takes effect directly, irrespective of the programmed velocity.
3 - 9	Reserved
10	As for configuration 0, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.
11	As for configuration 1, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.
12	As for configuration 2, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.

18.4.5 Multiple feedrate values in one block

Function

The function "Multiple feedrate values in one block" can be used to activate six different feedrate values of an NC block, a dwell time or a retraction motion-synchronously, depending on the external digital and/or analog inputs.

When the input for the sparking out time or retraction path is activated, the distance-to-go for the path axes or the particular single axis is deleted and the dwell time or retraction is started.

The retraction is started within an IPO cycle.

Signals

The input signals are combined in one input byte for the function. A fixed functional assignment applies within the byte.

Table 18-1 Input byte for the "Multiple feedrates in one block" function

	Bit							
	7	6	5	4	3	2	1	0
Input no.	I7	I6	I5	I4	I3	I2	I1	I0
Feedrate address	F7	F6	F5	F4	F3	F2	ST	SR

I7 to I2: Activation of feedrates F7 to F2

E1: Activation of the dwell time ST/STA (in seconds)

I0: Activation of the retraction motion SR/SRA

Priority of the signals

The signals are scanned in ascending order starting at I0. Therefore, the retraction motion (SR) has the highest priority and the feedrate F7 the lowest priority.

SR and ST end the feedrate motions that were activated with F2 to F7.
SR also ends ST, i.e. the complete function.

The signal with the highest priority determines the current feedrate.

The response to loss of the respective highest-priority input (F2 - F7) can be defined with the machine data:

MD21230 $MC_MULTFEED_STORE_MASK (storage behavior for the "Multiple feedrate values in one block" function)

Bit	Value	Meaning
2 ... 7	0	With the loss of the respective highest-priority input, the associated feedrate is not retained (default setting).
	1	Set bit 2 to 7 ensures that the associated feedrate (F2 to F7) that was selected by the respective highest-priority input signal is also retained when there is a loss of the input signal and a lower-priority input is active.

The end-of-block criterion is satisfied when:

- The programmed end position is reached
- The retraction motion ends (SR)
- The dwell time elapses (ST)

Hardware assignment

The input byte for the "Multiple feedrate values in one block" function can be assigned a maximum of two digital input bytes or comparator input bytes of the NC I/O:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN (assignment of the input bytes of the NC I/O for "Multiple feedrate values in one block"), bits 0 ... 15

The input bits can also be inverted:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN, bit 16 ... 31

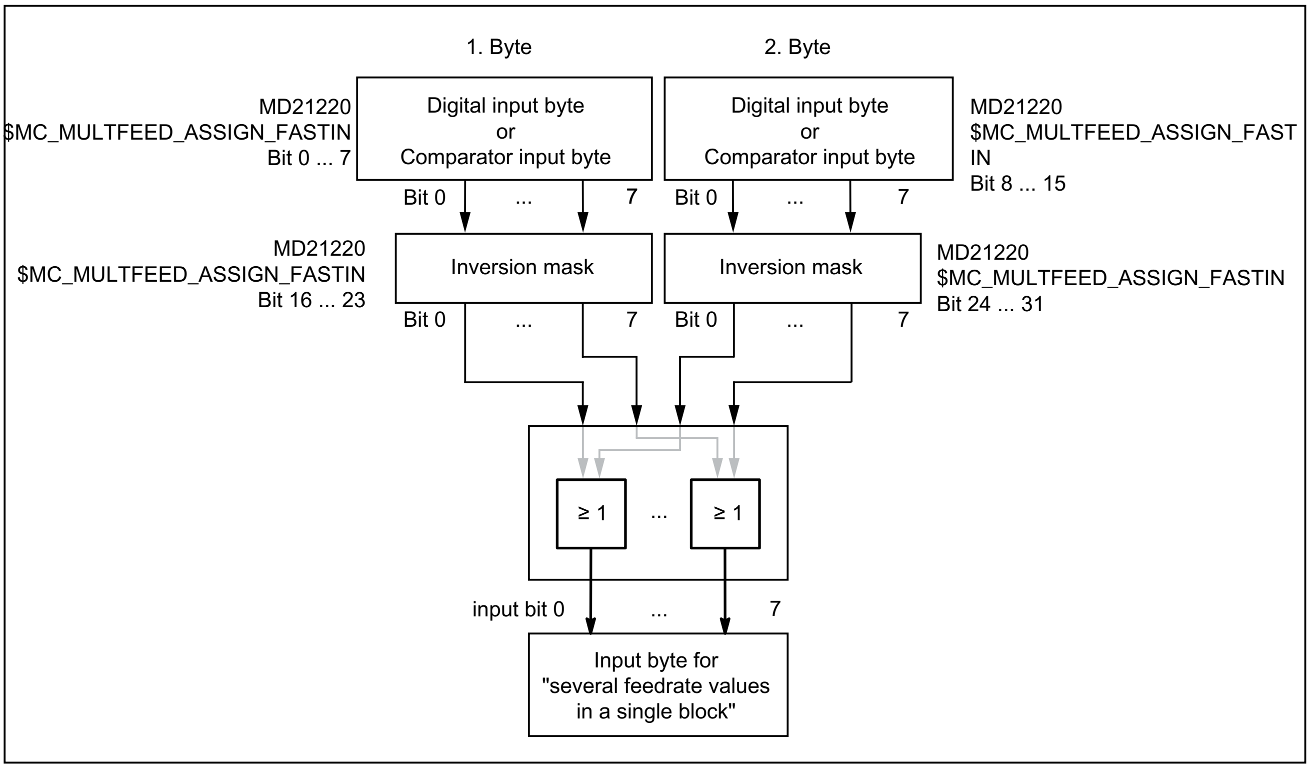

1. Byte
2. Byte
MD21220 $MC_MULTFEED_ASSIGN_FASTIN Bit 0 ... 7
Digital input byte or Comparator input byte
Digital input byte or Comparator input byte
MD21220 $MC_MULTFEED_ASSIGN_FASTIN Bit 8 ... 15
Bit 0 ... 7
Bit 0 ... 7
MD21220 $MC_MULTFEED_ASSIGN_FASTIN Bit 16 ... 23
Inversion mask
Inversion mask
MD21220 $MC_MULTFEED_ASSIGN_FASTIN Bit 24 ... 31
Bit 0 ... 7
Bit 0 ... 7
≥ 1 ... ≥ 1
input bit 0 ... 7
Input byte for "several feedrate values in a single block"

Figure 18-10 Signal assignment for the "Multiple feedrate values in one block" function

The assignment of the digital input bytes and parameterization of the comparators are described in:
References:
Function Manual, Extended Functions; Digital and analog NC I/O (A4)

Programming

Path motion

The path feedrate is programmed under the address `F` and remains valid until an input signal is present. This value acts modally.

`F2=...` to `F7=...` can be used in addition to the path feedrate to program up to six further feedrates in the block. The numerical expansion indicates the bit number of the input that activates the feedrate when changed:

Example:

```
F7=1000          ;7 corresponds to input bit 7
```

The programmed values act non-modally. The path feedrate programmed under `F` applies in the next block.

Dwell (sparking out time) and retraction path are programmed under separate addresses in the block:

`ST=...`	Dwell time (for grinding sparking out time)
`SR=...`	Retraction path

These addresses apply non-modally.

Axial motion

The axial feedrates are programmed under address `FA` and remain valid until an input signal is present. They act modally.

`FMA[2,<axis>]=...` to `FMA[7,<axis>]=...` can be used to program up to six further feedrates per axis in the block.

The first expression in square brackets indicates the bit number of the input that activates the feedrate when changed. The second expression indicates the axis to which the feedrate applies.

Example:

```
FMA[3,Y]=1000          ; Axial feedrate for Y axis, corresponds to input bit 3
```

The values programmed under `FMA` act non-modally. The feedrate programmed under `FA` applies to the next block.

Dwell (sparking out time) and retraction path can also be defined for a single axis:

`STA[<axis>]=...`	Axial dwell time (sparking out time)
`SRA[<axis>]=...`	Axial retraction path

The expression in square brackets indicates the axis for which the sparking out time and retraction path apply.

Examples:

```
STA[X]=2.5             ; The sparking out time for the X axis is 2.5 seconds.
SRA[X]=3.5             ; The retraction path for the X axis is 3.5 (unit e.g. mm).
```

These addresses apply non-modally.

Note

Retraction path

The unit for the retraction path refers to the current valid unit of measurement (mm or inch).

The reverse stroke is always made in the opposite direction to the current motion. `SR`/`SRA` always programs the value for the reverse stroke. No sign is programmed.

Note

POS instead of POSA

If feedrates, sparking out time (dwell time) or return path are programmed for an axis on account of an external input, this axis must not be programmed as POSA axis (positioning axis over multiple blocks) in this block.

Note

Status query

It is also possible to poll the status of an input for synchronous commands of various axes.

Note

LookAhead

Look Ahead is also active for multiple feedrates in one block. In this way, the current feedrate can be restricted by the Look Ahead value.

Application

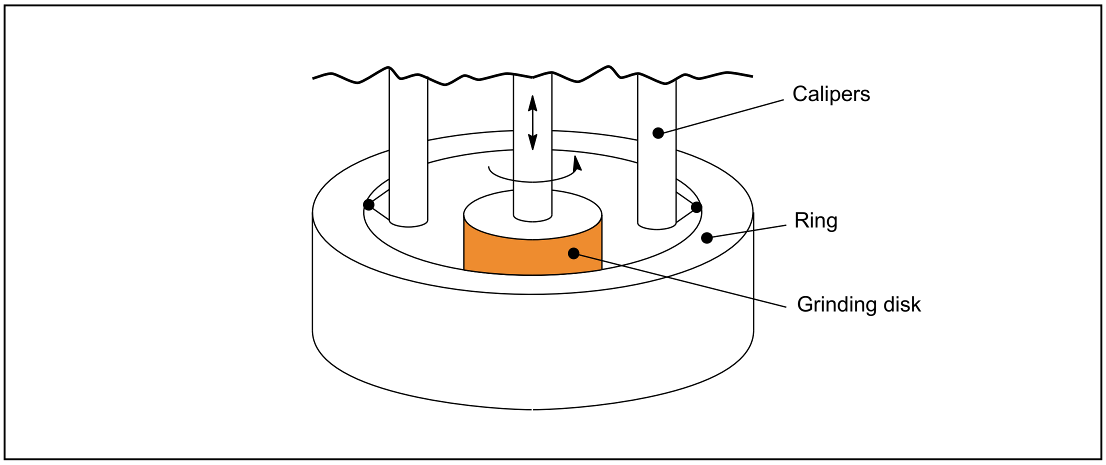
The "Multiple feedrate values in one block" function is used primarily for grinding, but is not restricted to it.

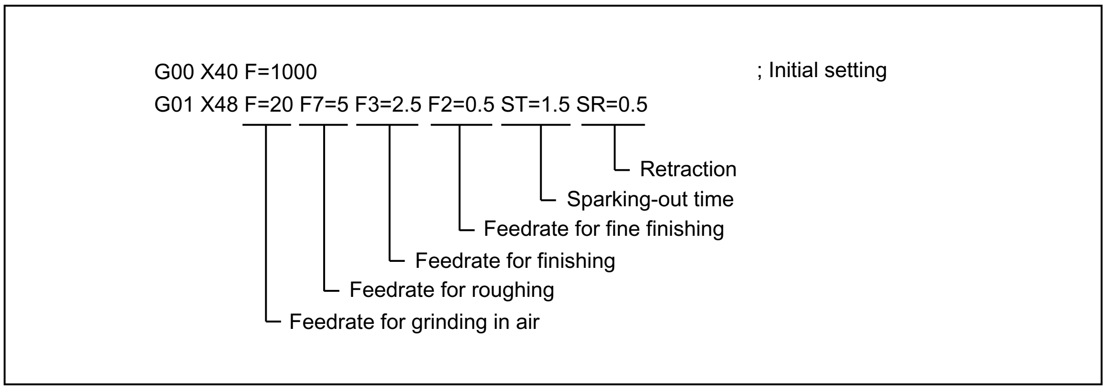
Typical applications are, for example:

- Analog or digital calipers
  Depending on whether the external inputs are analog or digital, various feedrate values, a dwell time and a retraction path can be activated. The limit values are defined via the setting data.
- Switching from infeed to working feedrate via proximity switch

Example

Internal grinding of a conical ring, where the actual diameter is determined using calipers and, depending on the limits, the feedrate value required for roughing, finishing or fine finishing is activated. The position of the calipers also provides the end position. Thus, the block end criterion is determined not only by the programmed axis position of the infeed axis but also by the calipers.





```
G00 X40 F=1000                                        ; Initial setting
G01 X48 F=20 F7=5 F3=2.5 F2=0.5 ST=1.5 SR=0.5
```

Restrictions for the function with SINUMERIK 828D

MD21220 =	1H	1st byte (4 bits on X242 and 4 bits via interface)
	2H	2nd byte (4 bits on X252 and 4 bits via interface)
	102H	8 bits via interface only

With the SINUMERIK 828D, a whole byte cannot be addressed because of the limited hardware input terminals. This therefore means the following:

- Only 4 different feedrates can be programmed on the hardware (4 bits on X242 or 4 bits on X252 on the PPU)
- The remaining 4 bits are implemented via the interface.
- "Only" the interface is used, not hardware inputs. Therefore, the feedrates no longer run in one IPO cycle.

Table for the assignment of the feedrate address to an input or interface:

Feedrate address	1st byte	$A_IN[x]	Interface	2nd byte	$A_IN[x]	Interface
SR	0	1	DB2800.DBX1.0	0	9	DB2800.DBX1001.0
ST	1	2	DB2800.DBX1.1	1	10	DB2800.DBX1001.1
F2	2	3	DB2800.DBX1.2	2	11	DB2800.DBX1001.2
F3	3	4	DB2800.DBX1.3	3	12	DB2800.DBX1001.3
F4	4	5	DB2800.DBX1.4	4	13	DB2800.DBX1001.4
F5	5	6	DB2800.DBX1.5	5	14	DB2800.DBX1001.5
F6	6	7	DB2800.DBX1.6	6	15	DB2800.DBX1001.6
F7	7	8	DB2800.DBX1.7	7	16	DB2800.DBX1001.7

Example:

G90 G0 X600

POS[X]=700 FA[X]=1000

POS[X]=700 FA[**3**,X]=2000 -> (if $A_IN[**4**]=**1** -> then the feedrate value is 2000, otherwise the value 1000 applies for the feedrate.)

Commissioning:

4 feedrate addresses through the hardware terminals and 4 through the interface

- -X242 = wire IN1 to IN4 (PIN 3 – 6) -> SR, ST, F2, F3
- Interface: DB2800.DBX1.4 – 1.7 = IN5 – IN8 -> F4, F5, F6, F7
- MD21220 =1H

18.4.6 Fixed feedrate values

Function

The "Fixed feedrate values" function can be used to activate fixed feedrates (max. four) defined via the machine data instead of the programmed feedrate or the configured JOG velocities.

The function is available in AUTOMATIC and JOG mode.

Behavior in AUTOMATIC mode

The contour travels at the activated fixed feedrate, instead of using the programmed feedrate.

Behavior in JOG mode

The axis is traversed with the activated fixed feedrate instead of the configured JOG velocity / JOG rapid traverse velocity. The travel direction is specified via the interface signal.

Parameterization

The setting of the fixed feedrates is performed:

- For linear axes with the machine data:
  MD12202 $MN_PERMANENT_FEED[<n>]
- For rotary axes with the machine data:
  MD12204 $MN_PERMANENT_ROT_AX_FEED[<n>]

where <n> = 0, 1, 2, 3 (for fixed feedrate 1, 2, 3, 4)

Note

The fixed feedrates are always linear feedrate values. Switchover to linear feedrate is conducted internally even in case of revolutional feedrate.

Activation

The fixed feedrates are activated via NC/PLC interface signals:

- In AUTOMATIC mode for path/geometry axes using the channel-specific interface signals:
  DB21, ... DBX29.0 (activate fixed feedrate 1)
  DB21, ... DBX29.1 (activate fixed feedrate 2)
  DB21, ... DBX29.2 (activate fixed feedrate 3)
  DB21, ... DBX29.3 (activate fixed feedrate 4)
- In JOG mode for machine axes using the axis-specific interface signals:
  DB31, ... DBX3.2 (activate fixed feedrate 1)
  DB31, ... DBX3.3 (activate fixed feedrate 2)
  DB31, ... DBX3.4 (activate fixed feedrate 3)
  DB31, ... DBX3.5 (activate fixed feedrate 4)

Supplementary conditions

Effectiveness

The function "Fixed feedrate values" is **not** active:

- For spindles
- For positioning axes
- When tapping

Override = 0

The traversing behavior for override = 0 depends on the setting in machine data:

MD12200 $MN_RUN_OVERRIDE_0

DRF offset

The DRF offset cannot be activated for a selected fixed feedrate.

18.4.7 Programmable feedrate characteristics

Function

To permit flexible definition of the feedrate characteristic, the feedrate programming according to DIN 66025 has been extended by linear and cubic characteristics.

The cubic profiles can be programmed directly or as an interpolating spline.

Programming

You can program the following feedrate profiles:

- FNORM
  Behavior in accordance with DIN 66025 (default setting).
  An F-value programmed in the block is applied over the entire path of the block, and is subsequently regarded as a fixed modal value.
- FLIN
  An F value programmed in the block is traversed linearly over the path from the current value at the beginning of the block to the end of the block, and is subsequently regarded as modal value.
- FCUB
  The non-modal programmed F values (relative to the end of the block) are connected by a spline. The spline starts and ends tangentially to the previous or following feedrate setting. If the F address is missing in one block, then the last programmed F value is used.
- FPO
  The F address [syntax: `F=FPO(...,...,...)`] designates the characteristic of the feedrate via a polynomial from the current value to the end of the block in which it was programmed. The end value is treated as modal from there onwards.

Parameterization

If FLIN and FCUB are used in connection with compression COMPON, a tolerance can be defined for the path feedrate:

MD20172 $MC_COMPRESS_VELO_TOL (max. permissible deviation of the path feedrate with compression)

Supplementary conditions

FLIN/FCUB

The path velocity profile programmed with FLIN or FCUB is not active together with revolutional feedrate for G95 as well as with constant cutting rate with G96/G961 and G97/G971.

References

For further information on the programmable feedrate characteristics, see Programming Manual, Job Planning.

18.4.8 Feedrate for chamfer/rounding FRC, FRCM

The machining conditions can change significantly during surface transitions to chamfer/rounding. Hence, the chamfer/rounding contour elements require dedicated, optimized feedrate values to achieve the desired surface quality.

Function

The feedrate for chamfer/rounding can be programmed via NC addresses.

Programming

Syntax:
```
... FRC/FRCM=<value>
```

Meaning:

`FRC`:	**Non-modal** feedrate for chamfer/rounding
`FRCM`:	**Modal** feedrate for chamfer/rounding
`<value>`:	The feedrate is interpreted according to the active feedrate type:

- G94, G961, G971: Feedrate in mm/min or inch/min or °/min
- G95, G96, G97: Revolutional feedrate in mm/rev or inch/rev

Note

FRC is only effective if a chamfer/rounding is programmed in the block or if RNDM has been activated.

FRC overwrites the F or FRCM value in the current block.

The feedrate programmed under FRC must be greater than zero.

FRCM=0 activates the feedrate programmed under F for chamfering/rounding.

Parameterization

Assignment of the chamfer/rounding to the previous or following block

The feedrate type (G94, G95, G96, G961 ...) and therefore the conversion to the internal format must be consistent within the block for F and FRC/FRCM. In this context, the following machine data must be taken into account:

MD20201 $MC_CHFRND_MODE_MASK (chamfer/rounding behavior)

Bit	Value	Meaning
0	0	The technology of the chamfer/rounding (feedrate, feedrate type, M commands, etc.) is determined by the following block (default setting).
	1	The technology of the chamfer/rounding is determined by the previous block (recommended setting).

Maximum number of empty blocks

The number of blocks without traversing information in the compensation plane (empty blocks) permitted between two blocks with traversing information during active chamfer/rounding, is limited. The maximum number is specified in the machine data:

MD20200 $MC_CHFRND_MAXNUM_DUMMY_BLOCKS (empty blocks for chamfer/radii)

Supplementary conditions

FLIN/FCUB

Feedrate interpolation FLIN and FCUB is **not** possible for chamfer/rounding.

G0

FRC/FRCM is not active when a chamfer is traversed with G0. The programming is possible in accordance with the F value without error message.

Change G94 ↔ G95

If FRCM is programmed, the FRCM value will need to be reprogrammed like F on change G94 ↔ G95, etc. If only F is reprogrammed and if the feedrate type FRCM > 0 before the change, an error message will be output.

Example

Example 1: MD20201 bit 0 = 0; take feedrate from following block (default setting!)

Program code	**Comment**
`N10 G0 X0 Y0 G17 F100 G94`	
`N20 G1 X10 CHF=2`	`; Chamfer N20-N30 with F=100 mm/min`
`N30 Y10 CHF=4`	`; Chamfer N30-N40 with FRC=200 mm/min`
`N40 X20 CHF=3 FRC=200`	`; Chamfer N40-N60 with FRCM=50 mm/min`
`N50 RNDM=2 FRCM=50`	
`N60 Y20`	`; Modal rounding N60-N70 with FRCM=50 mm/min`
`N70 X30`	`; Modal rounding N70-N80 with FRCM=50 mm/min`
`N80 Y30 CHF=3 FRC=100`	`; Chamfer N80-N90 with FRC=100 mm/min`
`N90 X40`	`; Modal rounding N90-N100 with F=100 mm/min (deselection of FRCM)`
`N100 Y40 FRCM=0`	`; Modal rounding N100-N120 with G95 FRC=1 mm/rev`
`N110 S1000 M3`	
`N120 X50 G95 F3 FRC=1`	
`...`	
`M02`	

Example 2: MD20201 bit 0 = 1; take feedrate from previous block (recommended setting!)

Program code	Comment
`N10 G0 X0 Y0 G17 F100 G94`	
`N20 G1 X10 CHF=2`	`; Chamfer N20-N30 with F=100 mm/min`
`N30 Y10 CHF=4 FRC=120`	`; Chamfer N30-N40 with FRC=120 mm/min`
`N40 X20 CHF=3 FRC=200`	`; Chamfer N40-N60 with FRC=200 mm/min`
`N50 RNDM=2 FRCM=50`	
`N60 Y20`	`; Modal rounding N60-N70 with FRCM=50 mm/min`
`N70 X30`	`; Modal rounding N70-N80 with FRCM=50 mm/min`
`N80 Y30 CHF=3 FRC=100`	`; Chamfer N80-N90 with FRC=100 mm/min`
`N90 X40`	`; Modal rounding N90-N100 with FRCM=50 mm/min`
`N100 Y40 FRCM=0`	`; Modal rounding N100-N120 with F=100 mm/min`
`N110 S1000 M3`	
`N120 X50 CHF=4 G95 F3 FRC=1`	`; Chamfer N120-N130 with G95 FRC=1 mm/rev`
`N130 Y50`	`; Modal rounding N130-N140 with F=3 mm/rev`
`N140 X60`	
`...`	
`M02`	

18.4.9 Non-modal feedrate FB

Function

The "Non-modal feedrate" function can be used to define a separate feedrate for a single part program block. After this block, the previous modal path feedrate is active again.

Programming

Syntax:
```
... FB=<value>
```

Meaning:

`FB`:	Separate feedrate for the current block
`<value>`:	The feedrate is interpreted according to the active feedrate type:

- G94, G961, G971: Feedrate in mm/min or inch/min or °/min
- G95, G96, G97: Revolutional feedrate in mm/rev or inch/rev

Note

The feedrate programmed under FB must be greater than zero.

If no traversing motion is programmed in the block (e.g. computation block), the FB has no effect.

If no explicit feed for chamfering/rounding is programmed, then the value of FB also applies for any contour element chamfering/rounding in this block.

Simultaneous programming of FB and FD (manual handwheel traversing with feedrate override) or F (modal path feedrate) is not possible.

18.4.10 Influencing the single axis dynamic response

Single axes

Single axes can be programmed in the part program, in synchronized actions and via the PLC:

• Part program:	`POS[<axis>]=...`
	`POSA[<axis>]=...`
	`SPOS[<axis>]=...`
	`SPOSA[<axis>]=...`
	`OS[<axis>]=...`
	`OSCILL[<axis>]=...`
• Synchronized actions:	`EVERY ... DO`
	`POS[<axis>]=...`
	`SPOS[<spindle>]=...`
	`MOV[<axis>]=...`
• PLC:	FC18

Dynamic response

The dynamic response of an axis is influenced by:

- MD32060 $MA_POS_AX_VELO (positioning axis velocity)
  The effective positioning axis velocity can be changed:
  - Part program / synchronized action: Axial feedrate FA or percentage feedrate override OVRA
  - PLC: Specification of FRate or overwriting the axial override
- MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)
  The effective maximum axis acceleration can be changed:
  - Part program indirectly: Writes the machine data with subsequent "Activate machine data"
  - Part program directly: Percentage acceleration override ACC
  - Synchronized actions indirectly: Writes the machine data and triggers an ASUP for the activation of "Activate machine data" function
  - Synchronized actions directly: Percentage acceleration override `ACC` (cannot be preset by the PLC).

  Via the PLC, the same options apply as in synchronized actions.
- Part program commands: BRISKA, SOFTA, DRIVEA, JERKA
  Cannot be programmed in synchronized actions (only indirectly via ASUP).
  Cannot be specified by the PLC (only indirectly via ASUP).
- Active servo parameter set
  The active parameter set can be changed:
  - Part program / synchronized action: SCPARA
  - PLC: DB31, … DBX9.0-2 (controller parameter set)

  For detailed information on the servo parameter sets, see "Parameter sets of the position controller (Page 385)".

Note

Dynamic response changes

Dynamic response changes made in the part program do not affect command or PLC axis motion. Dynamic response changes made in synchronized actions have no effect on traversing motion programmed in the part program.

Feedforward control

The type of feedforward control and the path axes that should be traversed with feedforward control can be directly programmed in the part program using FFWON/FFWOF. In synchronized actions and from the PLC, programming is only possible indirectly via an ASUP.

Percentage acceleration override (ACC)

In a part program or synchronized action, the acceleration specified in machine data:
MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)
can be changed in a range from 0% – 200% using the ACC command.

Syntax:
`ACC[<axis>]=<value>`

Meaning:

`ACC`:	Keyword for the programming of the percentage acceleration override
`<axis>`:	Name of the channel axis or spindle
`<value>`:	Acceleration change in percent relative to MD32300
	Value range: 0 ... 200

The actual axial acceleration value can be read via the system variable $AA_ACC. It is determined by:

$AA_ACC[<axis>] = (MD32300 $MA_MAX_AX_ACCEL[<axis>]) * ACC[<axis>] / 100

MD32320 $MA_DYN_LIMIT_RESET_MASK can be used to specify the basic setting of the value programmed with ACC for a channel reset or end of part program M30.

Note

The acceleration override programmed with ACC can be read using the system variable $AA_ACC. However, $AA_ACC is read in the part program at a different time than when reading in a synchronized action.

The system variables $AA_ACC only contain the value programmed in the part program with `ACC` if, in the meantime, the acceleration override was not changed by programming `ACC` in a synchronized action. The same applies for the reverse situation.

Percentage acceleration override and main run axes

Depending on whether the system variable $AA_ACC is read in the part program or synchronized action, the value for the acceleration override programmed with ACC is output for the NC axes or main run axes (command axes, PLC axes, asynchronous oscillating axes, etc.).

For correct results, system variable $AA_ACC must therefore always be read at the same location (part program or synchronized action) from where the acceleration override was programmed with `ACC`.

Examples:

Writing ACC in a part program:

```
N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]
```

Writing ACC in a synchronized action:

```
N100 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]
```

```
Writing ACC and reading $AA_ACC in a part program:
ACC[X]=50                                         ; writing
RO=$AA_ACC[X]                                     ; reading

IF (RO <> $MA_MAX_AX_ACCEL[X] * 0.5)              ; checking
   SETAL(61000)
ENDIF
```

```
Writing ACC and reading $AA_ACC in a synchronized action:
WHEN TRUE DO ACC[X]=25 R0=$AA_ACC[X]              ; writing and reading
G4 F1

IF (RO <> $MA_ MAX_AX_ACCEL[X] * 0.25)            ; checking
   SETAL(61001)
ENDIF
```

end-of-motion criterion for single axes

Similar to the block change criterion for path interpolation (G601, G602, G603) the end-of-motion criterion for traversing motion of individual axes can be programmed in part programs / synchronized actions:

Program command	End-of-motion criterion
`FINEA[<axis>]`	"Exact stop fine"
`COARSEA[<axis>]`	"Exact stop coarse"
`IPOENDA[<axis>]`	"Interpolator stop" (IPO stop)

The most recently programmed value is kept after the end of program or NC reset.

The effective end-of-motion criterion can be read using the axis-specific system variable $AA_MOTEND.

Note

Depending on whether the system variable $AA_MOTEND is read in the part program or synchronized action, it contains the value for the NC axes or the main run axes.

Example:

```
Part program:
N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 COARSEA[X]
```

```
Synchronized action:
N100 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOEN-
DA[X]
```

References:
For further information on block changes and end-of-motion criteria for FINEA, COARSEA and IPOENDA, see:
Function Manual, Extended Functions; Positioning Axes (P2), Section: Block change

Programmable servo parameter set (SCPARA)

In the part program / synchronized action, the servo parameter set can be specified using SCPARA.

Syntax
```
SCPARA[<axis>] = <parameter set number>
```

Meaning

`SCPARA`:	Keyword for the activation of the specified servo parameter set
`<axis>`:	Name of the channel axis
`<parameter set number>`:	Number of the servo parameter set to be activated

Note

The activation of the parameter set specified using SCPARA can be suppressed from the PLC user program:

DB31,… DBX9.3 = 1 (parameter set specification disabled by SCPARA)

In this case, **no** message is displayed.

The number of the active parameter set can be read using the system variable $AA_SCPAR.

Supplementary conditions

Different end-of-motion criteria

Different end-of-motion criteria will affect how quickly or slowly part program blocks are completed. This can have side effects for technology cycles and PLC user parts.

Parameter set change

The PLC user program must be expanded if the servo parameter set is to be changed both inside a part program or synchronized action and the PLC.

Power On

After POWER ON, the following basic settings are made:

- Percentage acceleration override for all single-axis interpolations: 100%
- End-of-motion criterion for all single-axis interpolations: FINEA
- Servo parameter set: 1

Mode change

When the operating mode is changed from AUTOMATIC to JOG, the programmed dynamic response changes remain valid.

Reset

With reset, the last programmed value remains for the part program specifications. The settings for main-run interpolations do not change.

Block search

The last end-of-motion criterion programmed for an axis is collected and output in an action block. The last block with a programmed end-of-motion criterion that was processed in the block search run serves as a container for all programmed end-of-motion criteria for all axes.

18.5 Supplementary conditions

Unit of measurement

The valid unit of measurement of the feedrates depends on the set measuring system and the entered axis type:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC (basic system of the control metric/inch)

MD30300 $MA_IS_ROT_AX (rotary or linear axis)

Initial setting for the feedrate type

The initial setting for the feedrate type is specified in the machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

The default setting is G94.

The initial setting of the feedrate type is only displayed when a part program is started.

Effectiveness after reset

Whether the last programmed F, FA, OVR, OVRA values are also active after reset depends on the setting in the machine data:

MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)

Spindle positioning

With active G95, G96, G961, G97, G971, G33, G34, G35 spindle positioning should not be performed, because the derived path feedrate following spindle positioning = 0. If the programmed axis position has not then been reached, the block cannot be completed.

18.6 Data lists

18.6.1 Machine data

18.6.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10651	IPO_PARAM_THREAD_NAME_TAB	Name of the interpolation parameters for convex threads
10704	DRYRUN_MASK	Activation of dry run feedrate
10710	PROG_SD_RESET_SAVE_TAB	Setting data to be updated
11410	SUPPRESS_ALARM_MASK	Mask for suppressing special alarms
11550	STOP_MODE_MASK	Defines the stop behavior
12000	OVR_AX_IS_GRAY_CODE	Axis feedrate override switch, Gray-coded
12010	OVR_FACTOR_AX_SPEED	Evaluation of the axis feedrate override switch
12020	OVR_FEED_IS_GRAY_CODE	Path feedrate override switch, Gray-coded
12030	OVR_FACTOR_FEEDRATE	Evaluation of the path feedrate 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	Defines the reference of the path override
12090	OVR_FUNCTION_MASK	Selection of override specifications
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

18.6.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20100	DIAMETER_AX_DEF	Geometry axes with transverse axis functions
20150	GCODE_RESET_VALUES	Basic setting of the G groups
20172	COMPRESS_VELO_TOL	Maximum permissible deviation from path feed for compression
20200	CHFRND_MAXNUM_DUMMY_BLOCKS	Empty blocks with phase/radii
20201	CHFRND_MODE_MASK	Behavior for chamfer/rounding
20750	ALLOW_GO_IN_G96	G0 logic for G96, G961
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour

Number	Identifier: $MC_	Description
21220	MULTFEED_ASSIGN_FASTIN	Assignment of input bytes of NC I/O for "Multiple feedrate values in one block"
21230	MULTFEED_STORE_MASK	Storage behavior for the "Multiple feedrate values in one block" function
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions
22410	F_VALUES_ACTIVE_AFTER_RESET	F function active after reset

18.6.1.3 Axis/Spindle-specific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis/spindle
32000	MAX_AX_VELO	Maximum axis velocity
32060	POS_AX_VELO	Initial setting for positioning axis velocity
32300	MAX_AX_ACCEL	Axis acceleration
32320	DYN_LIMIT_RESET_MASK	Reset behavior of dynamic limits
34990	ENC_ACTIVAL_SMOOTH_TIME	Smoothing time constant for actual values
35100	SPIND_VELO_LIMIT	Maximum spindle speed
35130	GEAR_STEP_MAX_VELO_LIMIT	Maximum speed of gear stage
35140	GEAR_STEP_MIN_VELO_LIMIT	Minimum speed of gear stage
35160	SPIND_EXTERN_VELO_LIMIT	Spindle-speed limitation via PLC

18.6.2 Setting data

18.6.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42010	THREAD_RAMP_DISP	Acceleration behavior of axis when thread cutting
42100	DRY_RUN_FEED	Dry run feedrate
42101	DRY_RUN_FEED_MODE	Dry run feed mode
42110	DEFAULT_FEED	Default value for path feed
42600	JOG_FEED_PER_REV_SOURCE	Revolutional feedrate control in the JOG mode
43300	ASSIGN_FEED_PER_RES_SOURCE	Revolutional feedrate for positioning axes/spindles

18.6.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43210	SPIND_MIN_VELO_G25	Programmed spindle speed limiting G25
43220	SPIND_MAX_VELO_G26	Programmed spindle speed limiting G26
43230	SPIND_MAX_VELO_LIMS	Spindle speed limiting with G96

18.6.3 Signals

18.6.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate dry run feedrate	DB21, ...DBX0.6	DB320x.DBX0.6
Feedrate override	DB21, ...DBB4	DB320x.DBB4
Rapid traverse override	DB21, ...DBB5	DB320x.DBB5
Feed disable	DB21, ...DBX6.0	DB320x.DBX6.0
Rapid traverse override active	DB21, ...DBX6.6	DB320x.DBX6.6
Feedrate override active	DB21, ...DBX6.7	DB320x.DBX6.7
Feed stop, geometry axis 1	DB21, ...DBX12.3	DB320x.DBX1000.3
Feed stop, geometry axis 2	DB21, ...DBX16.3	DB320x.DBX1004.3
Feed stop, geometry axis 3	DB21, ...DBX20.3	DB320x.DBX1008.3

18.6.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Dry run feedrate selected	DB21, ...DBX24.6	DB170x.DBX0.6
Feedrate override selected for rapid traverse	DB21, ...DBX25.3	DB170x.DBX1.3
Activate fixed feedrate 1 for path/geometry axes	DB21, ...DBX29.0	DB320x.DBX13.0
Activate fixed feedrate 2 for path/geometry axes	DB21, ...DBX29.1	DB320x.DBX13.1
Activate fixed feedrate 3 for path/geometry axes	DB21, ...DBX29.2	DB320x.DBX13.2
Activate fixed feedrate 4 for path/geometry axes	DB21, ...DBX29.3	DB320x.DBX13.3
Dry run feedrate active	DB21, ...DBX318.6	DB330x.DBX4002.6

18.6.3.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate override / spindle override	DB31, ...DBB0	DB380x.DBB0
Override active	DB31, ...DBX1.7	DB380x.DBX1.7

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate fixed feedrate 1 for machine axis	DB31, ...DBX3.2	DB380x.DBX3.2
Activate fixed feedrate 2 for machine axis	DB31, ...DBX3.3	DB380x.DBX3.3
Activate fixed feedrate 3 for machine axis	DB31, ...DBX3.4	DB380x.DBX3.4
Activate fixed feedrate 4 for machine axis	DB31, ...DBX3.5	DB380x.DBX3.5
Feed stop/spindle stop	DB31, ...DBX4.3	DB380x.DBX4.3

18.6.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Revolutional feedrate active	DB31, ...DBX62.2	DB390x.DBX2.2
F function for positioning axis	DB31, ...DBB81	-
Programmed speed too high	DB31, ...DBX83.1	DB390x.DBX2001.1

W1: Tool offset 19

19.1 Brief description

Calculating tool compensation data

The SINUMERIK 840D sl controller can be used to calculate the following tool compensation data:

- Length compensation
- Radius compensation
- Storage of tool data in a flexible tool offset memory:
  - Tool identification with T numbers from 0 to 32000
  - Definition of a tool with a maximum of 9 cutting edges
  - Cutting edge described by up to 25 tool parameters
- Tool selection selectable: Immediate or via selectable M function
- Tool radius compensation:
  - Selection and deselection strategy configurable: Normal or contour-related
  - Compensation active for all interpolation types:
    Linear
    Circle
    Helical
    Spline
    Polynomial
    Involute
  - Compensation at outer corners selectable:
    Transition circle/ellipse (`G450`) or equidistant intersection (`G451`)
  - Parameter-driven adaptation of `G450/G451` functions to the contour
  - Free traversing on outer corners with `G450` and `DISC` parameter
  - Number of dummy blocks without axis motion selectable in the compensation plane
  - Collision detection selectable:
    Possible contour violations are detected predictively, if:
    - Path is shorter than tool radius
    - Width of an inside corner is shorter than the tool diameter
  - Keep tool radius compensation constant
  - Intersection procedure for polynomials

Toolholder with orientation capability

This function permits the machining of inclined surfaces with allowance for tool length compensation, provided that the kinematics of the toolholder (without NC axes) permits a static orientation of the tool. The more complex 5-axis transformation is not required for this case.

Reference:
Function Manual, Special Functions; Multi-Axis Transformations (F2)

Appropriate selection of the tool data and toolholder data describes the kinematics for the controller such that it can make allowance for the tool length compensation. The controller can take some of the description data direct from the current frame.

Note

Please refer to the following documentation for further information on tools and tool compensations and a full technical description of the general and specific programming features for tool compensation (TLC and TRC):

References:
Programming Manual, Fundamentals

Flat/unique D number structure

Compensations can be selected via unique D numbers with management function.

Special handling of tool compensations

The evaluation of signs can be controlled for tool length and wear by the setting data:

SD42900 $SC_MIRROR_TOOL_LENGTH (sign change tool length when mirroring)

SD42960 $SC_TOOL_TEMP_COMP (temperature compens. regarding tool).

The same applies to the response of the wear components when mirroring geometry axes or changing the machining plane via setting data.

References:
Programming Manual, Fundamentals, Tool Offsets

G461/G462

In order to enable the solid machining of inside corners in certain situations with the activation and deactivation of tool radius compensation, commands `G461` und `G462` have been introduced and the approach/retraction strategy has thus been extended for tool radius compensation.

- `G461`
  If no intersection is possible between the last TRC block and a previous block, the controller calculates an intersection by extending the offset curve of this block with a circle whose center point coincides with the end point of the noncorrected block, and whose radius is equal to the tool radius.
- `G462`
  If no intersection is possible between the last TRC block and a previous block, the controller calculates an intersection by inserting a straight line at the end point of the last block with tool radius compensation (the block is extended by its end tangent).

Changing from G40 to G41/42

The change from `G40` to `G41/G42` and vice versa is no longer treated as a tool change for tools with relevant tool point direction (turning and grinding tools).

Tool compensation environments

Functions which enable the following actions in relation to the current states of tool data are available in SW 7.1:

- Save
- Delete
- Read
- Modify

Some of the functions were previously implemented in measuring cycles. They are now universally available.

A further function can be used to determine information about the assignment of the tool lengths of the active tool to the abscissa, ordinate and applicate.

19.2 Tool

19.2.1 General information

Select tool

A tool is selected in the program with the T function.

Whether the new tool will be loaded immediately by means of the T function depends on the setting in the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function)

Change tool immediately

MD22550 $MC_TOOL_CHANGE_MODE = 0 (new tool compensation with M function).

The new tool is changed immediately with the T function.

This setting is used mainly for turning machines with tool revolver.

Change tool with M06

MD22550 $MC_TOOL_CHANGE_MODE = 1 (new tool compensation with M function).

The new tool is prepared for changing with the T function.

This setting is used mainly on milling machines with a tool magazine, in order to bring the new tool into the tool change position without interrupting the machining process.

The old tool is removed from the spindle and the new tool is loaded into the spindle with the entered M function in the machine data:

- MD22560 $MC_TOOL_CHANGE_M_CODE (M function for tool change)

This tool change must be programmed with the M function `M06`, in accordance with DIN 66025.

The next tool is preselected with the machine data:

- MD20121 $MC_TOOL_PRESEL_RESET_VALUE (Preselected tool at RESET)

Its tool length compensation values must be considered at `RESET` and powerup according to machine data:

- MD20110 $MC_RESET_MODE_MASK (Determination of control default settings after RESET/TP end).

Value range of the T function or the tool number

The T function or the tool number can assume the following integer numbers:

- Minimum value: `T0` (no tool)
- Maximum value: `T32000` (tool with number 32000)

Tool cutting edge D

A tool Tx can be assigned various tool cutting edges `D1` ... `Dn`.

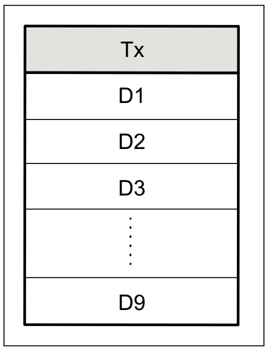


Figure 19-1 Example of a tool Tx with 9 cutting edges (D1 to D9)

Maximum number of tool cutting edges per tool

The maximum number of tool cutting edges (D numbers) that can be defined for each tool, is specified using the following machine data:

- MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL

D function

The selection of the tool cutting edge 1 ... n required for machining is realized by programming the appropriate D function, e.g. in the part program with `D1` ... `Dn`. The selected tool cutting edge always refers to the currently active tool. A tool cutting edge without active tool (actual T number is `T0` ⇒ no tool selected), is not effective.

All tool offsets of the active tool Tx are deselected using `D0`.

Selecting the cutting edge when changing tool

When selecting a new tool using a new T number,and loading this tool, the following options are available to select the cutting edge:

- The cutting edge number Dx is programmed.
- The cutting edge number is defined by the machine data:
  MD20270 $MC_CUTTING_EDGE_DEFAULT =<value> (basic position of the tool cutting edge without programming)

Value	Meaning
= 0	No automatic cutting edge selection in accordance with `M06`
≠ 0	Number of the cutting edge, which is selected in accordance with `M06`
= -1	The cutting edge number of the old tool is retained and is also selected for the new tool, in accordance with `M06`.

Activating the tool offset

`D1` to `Dn` activate the tool compensation for a cutting edge on the active tool. Tool length compensation and tool radius compensation can be activated at different times:

- **Tool length compensation** (TLC) is performed on the first traversing motion of the axis, on which the TLC is to act.
  This traversing motion must be a linear interpolation (`G0,G1`, `POS,POSA`) or polynomial interpolation (`POLY`). If the `POS/POSA` axis is one of the active geometry axes, the tool length compensation is applied with the first axis motion in which the WLK is supposed to act.
- **Tool radius compensation** (TRC) becomes active when `G41/G42` is programmed in the active plane (`G17`, `G18` or `G19`).
  In the NC block in which the tool radius correction is activated with `G41`/`G42`, `G0` (rapid traverse) or `G1` (linear interpolation) must be active and at least one geometry axis of the active working plane must be programmed.
  If only one geometry axis is specified when switching on, the last position of the second geometry axis is automatically supplemented and then traversed in both axes.
  If the axes are not geometry axes, then they must be transformed into geometry axes using `GEOAX` programming.
  The selection of tool radius compensation with `G41/G42` is only permitted in a program block with `G0` (rapid traverse) or `G1` (linear interpolation).

19.2.2 Compensation memory structure

Tool offset memory size

Each channel can have a dedicated tool offset memory (TO unit).

Which tool offset memory exists for the relevant channel is set with the machine data:

MD28085 $MC_MM_LINK_TOA_UNIT (assignment of TO unit to a channel)

The maximum number of tool cutting edges for all tools managed by the NC is set with the machine data:

MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA (number of tool cutting edges in NC)

Tools

The tool offset memory consists of tools numbered `T1` to `T32000`.

Each tool can be set up via TOA files or individually, using the "New tool" softkey. Offset values not required must be assigned the value zero. This is the default setting when the offset memory is created. The individual values in the offset memory (tool parameters) can be read and written from the program using system variables.

Note

The tools (`T1` to `T32000`) do not have to be stored in ascending order or contiguously in the tool offset memory, and the first tool does not have to be assigned number `T1`.

Tool cutting edges

Each tool can have up to nine cutting edges (`D1` to `D9`). The first cutting edge (`D1`) is set up automatically when a new tool is loaded to the tool offset memory. Other cutting edges (max. eight) are set up consecutively and contiguously using the "New cutting edge" softkey. A different number of tool cutting edges can assigned to each tool in this way.

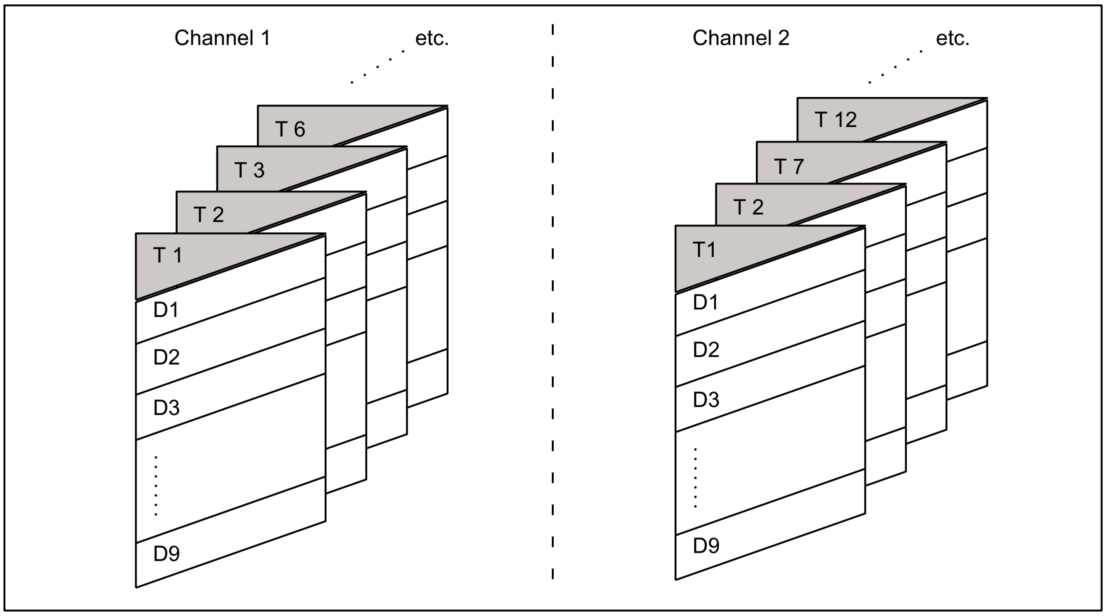
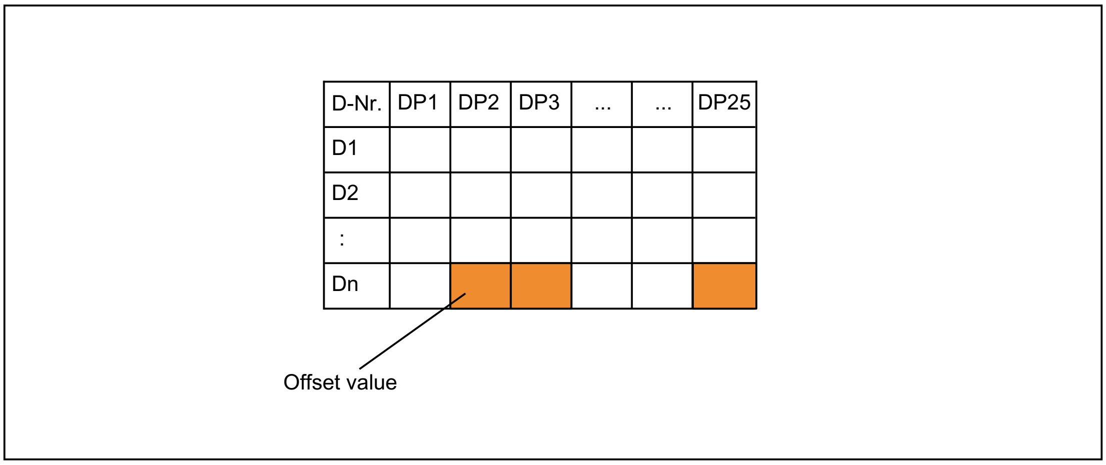


Figure 19-2 Example of a tool offset memory structure for two channels

19.2.3 Calculating the tool compensation

D No.

The **D no.** is sufficient for calculating the tool compensations (can be set via MD).



Programming

The above compensation block is to be calculated in the NC.

Part program call:

```
...
Dn
```

19.2.4 Address extension for NC addresses T and M

MD20096

Whether the address extension of T and M is also to be interpreted as spindle number when the tool management is **not** activated, can be set via the machine data:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO (spindle number as address extension)

The same rules then apply to the relationship between the D number and T number as when the "Tool management" function is active.

Effect on the D number

An offset data set is determined by the D number.

The D address cannot be programmed with an address extension.

The evaluation of the D address always refers to the currently active tool.

The programmed D address refers to the active tool in relation to the master spindle (same as for tool management function), when machine data is set:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE (spindle number as address extension).

Effect on the T number

If the "Tool management" function is active, the values programmed with reference to the master spindle (or master toolholder) are displayed as programmed/active T numbers.

If tool management is not active, **all** programmed T values are displayed as programmed/ active, regardless of the programmed address extension.

Only the T value programmed in relation to the master spindle is shown as programmed/active, when:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE (spindle number as address extension).

Example

The following example below shows the effect of MD20096.

Two spindles are considered. Spindle 1 is the master spindle. `M6` was defined as the tool change signal.

T1 = 5	
M1 = 6	
T2 = 50	
M2 = 6	
D4	

- If tool management is active, `D4` refers to tool "5".
  T2=50 defines the tool for the secondary spindle, whose tool does not influence the path compensation. The path is determined exclusively by the tool programmed for the master spindle.
- `D4` refers to tool "50" without active tool management and with the machine data:
  MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = **FALSE** (significance of address extension for T, M tool change).
  The address extensions of neither T nor M are evaluated in the NC.
  Each tool change command defines a new path compensation.
- `D4` refers to tool "5" (as when tool management is active) without active tool management and with the machine data:
  MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = **TRUE**.
  Address extension 1 (`T1=` ..., `M1=` ...) addresses the master spindle.

Note

Previously, when tool management was not activated, each tool change command (programmed with T or M) caused the tool offset to be recalculated in the path. The address extension is not defined further by this operation. The significance of the extension is defined by the user (in the PLC user program).

19.2.5 Free assignment of D numbers

"Relative" D numbers

In the NC 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.

Functions

Expansions to functions when assigning D numbers:

- The maximum permitted D numbers are defined via the machine data:
  MD18105 $MN_MM_MAX_CUTTING_EDGE_NO (max. value of the D numbers (DRAM))
  The default value is 9, in order to maintain compatibility with existing applications.
- The number of cuts (or the offset data sets) **for each tool** can be defined via the machine data:
  MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL (max. number of the D numbers per tool (DRAM))
  This allows you to customize the number of cutting edges to be configured for each tool to the actual number of real cutting edges for monitoring purposes.
- It is also possible to rename D numbers in the NC and thus to allocate arbitrary 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 NC on compensation selection during a tool change (only available via the OPI).

The cutting edge number CE is defined with system variable **$TC_DPCE[t,d]**:

- **t** stands for the internal T number.
- **d** stands for the D number.

Write accesses are monitored for collisions, i.e. all cutting edge numbers of a tool must be different. The variable $TC_DPCE is a component of the cutting edge parameter data set $TC_DP1 to $TC_DP25.

It is only practical to parameterize $TC_DPCE if the maximum cutting edge number (MD18105) is greater than the maximum number of cutting edges per tool (MD18106).

In this case, the default cutting edge number is the same as the classification number of the cutting edge. Compensations of a tool are created starting at number 1 and are incremented up to the maximum number of cutting edges per tool (MD18106).

The cutting edge number CE is the same as the D number (in compatibility with the behavior till now) if:

MD18105 ≤ MD18106.

A read operation returns CE=D. A write operation is ignored without an alarm message.

Note

The compensation values $TC_DP1 to $TC_DP25 of the active tool compensation can be read with system variable $P_AD[n], where n=1 to 25. The CE cutting edge number of the active compensation is returned with n=26.

Commands

If the maximum cutting edge number is **higher** than the maximum number of cutting edges for each tool, then the commands listed in the following table, are available

- (MD18105 $MN_MM_MAX_CUTTING_EDGE_NO) > (MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL)

Command	Meaning
`CHKDNO`	Checks the uniqueness of the available D numbers. The D numbers of all tools defined within a TO unit may not occur more than once. No allowance is made for replacement tools.
`GETDNO`	Determines the D number for the cutting edge of a tool. If no D number matching the input parameters exists, d=0. If the D number is invalid, a value greater than 32000 is returned.
`SETDNO`	Sets or changes the D number of the CE cutting edge of tool T. If there is no data block for the specified parameter, the value `FALSE` is returned. Syntax errors generate an alarm. The D number cannot be set explicitly to 0.
`GETACTTD`	Determines the associated T number for an absolute D number. There is not check for uniqueness. If several D numbers within a TO unit are the same, the T number of the first tool found in the search is returned. This command is not suitable for use with "flat" D numbers, because the value 1 is always returned in this case (no T numbers in database).
`DZERO`	Marks all D numbers of the TO unit as invalid. This command is used for support during retooling. Compensation data sets tagged with this command are no longer verified by the `CHKDNO` language command. These data sets can be accessed again by setting the D number once more with `SETDNO`.

Note

If the maximum cutting edge number is **less** than the maximum number of cutting edges for each tool, then the commands listed in the table are ineffective. This default setting is in the system for reasons of compatibility.

The individual commands are described in detail in:

References:
Programming Manual Fundamentals

Activation

In order to work with unique D numbers and, therefore, with the defined language commands, it must be possible to name D numbers freely for the tools.

The following conditions must be fulfilled for this purpose:

- MD18105 **>** MD18106
- The 'flat D number' function is not activated.
  MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE (type of D number programming (SRAM)).

Examples

MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 1 (max. value of the D numbers)

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.

Example: Assigning D (cutting edges) to T numbers (tools) and listing options for checking uniqueness:

- Assignment:
  - T No. 1 ⇒ assignment of D numbers 1, 2, 3
  - T No. 2 ⇒ assignment of D numbers 10, 20, 30, 40, 50
  - T No. 3 ⇒ assignment of D numbers 100, 200, 30 (**error**: 30 was entered instead of 300)
- Check for uniqueness:
  The D numbers of all tools defined within a TO unit must be unique.
  The check is made using the `CHKDNO` command. When no parameters are specified, all D numbers of all tools are checked with respect to one another. In this particular case, the return value == `FALSE`, as D30 is available in tool T2 and T3.

Command	Test	Test result
`CHKDNO`	All D numbers of all tools are checked against all other tools	Return value == `FALSE`, as D=30 is duplicated.
`CHKDNO (2, 3, 30)`	D number 30 in tools 2 and 3	Return value == `FALSE`, as D=30 is duplicated.
`CHKDNO (1, 3)`	All D numbers of tools 1 and 3	Return value == `TRUE` although there is a collision between the D=30 of the third tool and D=30 of the second tool.

Note

Cutting edges per tool

If tools with **n** cutting edges are used, then the value of the machine data should be set to **n**. This prevents inadmissible cutting edges to be defined for a tool.

MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL = **n**

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:

- `$TC_DPx[ <tool Tn>, <cutting edge Dm> ]`
- Internal T number Tn = 1
- D number Dm = 2
- Tool with one cutting edge with:

Program code	**Comment**
`$TC_DP2[ 1, 2 ] = 120`	`Tool length = 120`
`$TC_DP3[ 1, 2 ] = 5.5`	`Tool radius = 5.5`
`$TC_DPCE[ 1, 2 ] = 3`	`Cutting edge number CE = 3`

- MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 20 (max. value of the D numbers)

Within the part program, this compensation is programmed as standard with `T1, ....D2`.

You assign the current D number of cutting edge 3 to a variable (DNoOld) and define the variable DNoNew for the new D number:

Program code
`def int DNoOld, DNoNew = 17`
`DNoOld = GETDNO( 1, 3 )`
`SETDNO( 1, 3, DNoNew )`

The new D value 17 is then assigned to cutting edge CE=3.

Now the data for the cutting edge are addressed via D number 17, both via the system variable and in programming with the NC address D.

This compensation is now programmed in the part program with `T1, ....D17` and the data is addressed as follows:

Program code	Comment
`$TC_DP2[ 1, 17 ] = 120`	
`$TC_DP3[ 1, 17 ] = 5.5`	
`$TC_DPCE[ 1, 17 ] = 3`	`; Cutting edge number CE`

Note

If the tool has defined a further cutting edge, 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 the cutting edge 3, i.e. `SETDNO( 1, 1, 17)` returns the status = `FALSE` as return value.

DZERO - Invalidate D numbers

The activation of this command invalidates all D numbers of the tools in the TO unit. It is no longer possible to activate a compensation until valid D numbers are again available in the NC. The D numbers must be reassigned using the `SETDNO` command.

The following tools must be defined (all with cutting edge number 1):

`T1, D1`	D no. of cutting edge CE=1
`T2, D10`	D no. of cutting edge CE=1
`T3, D100`	D no. of cutting edge CE=1

The following command is then programmed:

Program code
`DZERO`

If one of the compensations is now activated (e.g. with `T3 D100`), an alarm is generated, because `D100` is not currently defined.

The D numbers are redefined with:

Program code	Comment
`SETDNO( 1, 1, 100 )`	`; T=1, cutting edge 1 is assigned the (new) D number 100`
`SETDNO( 2, 1, 10 )`	`; T=2, cutting edge 1 is assigned the (old) D number 10`
`SETDNO( 3, 1, 1 )`	`; T=3, cutting edge 1 is assigned the (new) D number 1`

Note

In the event of a power failure, the `DZERO` command can leave the NC 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 NC is arbitrary. The D numbers in the part programs for the new machining operation generally do not match the D numbers of the actual cutting edges. The retooling program can have the following appearance:

Program code	Comment
`DZERO`	`; All D numbers of the TO unit are tagged as invalid.`
`....`	`; One or more loops over the locations of the magazine 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.`

19.2.6 Compensation block in case of error during tool change

MD22550

If a tool preparation has been programmed in the part program and the NC detects an error (e.g. the data set for the programmed T number does not exist in the NC), the user can assess the error situation and perform appropriate tasks, in order to subsequently resume machining.

The tool change may be programmed independently, depending on the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool offset for M function)

MD22550 $MC_TOOL_CHANGE_MODE = 0

```
T= 'T no.'        ; Tool preparation + tool change in one NC block,
                  ; i.e. when T is programmed a new D compensation becomes
                  ; active in the NC. See machine data:
                  ; MD20270 $MC_CUTTING_EDGE_DEFAULT (basic position of the tool cut-
                  ting edge without programming)
```

MD22550 $MC_TOOL_CHANGE_MODE = 1

```
T= 'T no.'          ; Tool preparation
M06                 ; Tool change
                    ; (the number of the tool-change M code can be set),
                    ; i.e. when M06 is programmed, a new D compensation becomes
                    ; active in the NC (see
                    ; machine data MD20270 $MC_CUTTING_EDGE_DEFAULT)
```

If the tool management is not active, the following problems can be detected:

- D compensation data set missing
- Error in the part program

Note

The "tool is not in the magazine" problem cannot be detected since the NC did not have access to any magazine information for the tool offset.

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.

The D number and otherwise also the T number are required for the flat D number function. These parameters are transferred when the alarm is triggered.

Error in the part program

The options for intervention in the event of an error depend on how the tool change was programmed, defined via the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool offset for M function)

Tool change with T programming (MD22550 = 0)

In this case, the "Compensation block" function available in the NC is used. The NC program stops at the NC block in which an error was detected for the programmed T value. The "Compensation block" is executed again when the program is resumed.

The operator can perform the following:

- Correct the part program.
- Reload the missing cutting edge compensation data from the HMI.
- Include the missing cutting edge compensation data in the NC 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 resumed. 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 of the tool preparation (programming of T), which however is to be ignored at first. Execution is continued until the tool change request (usually `M06`) is executed in the NC program. The program is to stop at this point.

The programmed T address can be any number of program lines before the M06 command, or the two instructions can be in different (sub)programs. For this reason, it is not usually possible to modify a block which has already been executed via the compensation block.

The operator has the same options for intervention as with = 0.

Reloading of missing data is possible. In this case, however, T must be programmed with "Overstore".

If a program error has occurred, the line with the error cannot be corrected (Txx); only the line at which the program stopped and which generated the alarm can be edited. Only when machine data:

MD22562 $MC_TOOL_CHANGE_ERROR_MODE bit 0 = 1 (response to errors during tool change)

This results in the following sequence:

```
Txx              ; Error! Data set with xx does not exist
                 ; Note state; note 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 is interpreted and is OK.
                 ; Execution continues.
```

Renewed execution of the program points results in the following:

```
Txx              ; Error! Data set with xx does not exist,
                 ; Note state; note xx;
                 ; continue in program
....
Tyy M06          ; Detect bit memory "xx missing" → reject without further response,
                 ; as Tyy M06 is correct → program does not stop (correct).
```

If required, the original point of the T call can be corrected at 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 NC, Txx is programmed in "Overstore" and the program is subsequently resumed.

As in the case of "missing D number", the required parameter (T number) can be accessed by the user for "missing T number" via the appropriate alarm (17191).

Note

In order to enable program correction, it stops immediately at the faulty Txx block.

The program test mode is also stopped when machine data:

MD22562 $MC_TOOL_CHANGE_ERROR_MODE bit 0=1 (response to errors during tool change).

19.2.7 Definition of the effect of the tool parameters

MD20360

The effect of the tool parameters on the transverse axis in connection with diameter programming can be controlled selectively with the machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).

Details are described with the mentioned MD

DRF manual handwheel traversing with half distance

During DRF handwheel traversal, it is possible to move a transverse axis through only half the distance of the specified increment as follows:

Specify the distance with handwheel via the machine data:

MD11346 $MN_HANDWHEEL_TRUE_DISTANCE = 1 (handwheel path or speed specification)

Define the DRF offset in the transverse axis as a diameter offset with the machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK bit 9 = 1 (definition of tool parameters)

Deselecting an axial DRF compensation (`DRFOF`) also deletes an existing tool compensation (handwheel override in tool direction).

Note

For further information about superimposed movements with the handwheel, please refer to:

References:
Function Manual, Extended Functions; Manual traversing and manual handwheel traversing (H1)

Programming Manual, Fundamentals

(The Programming Manual describes the complete technical program options in order to deselect the DRF offset axis-specifically.)

19.3 Flat D number structure

19.3.1 General information

For **lathes**, simple tool management is possible using flat D numbers structure () exclusively via D numbers.

As a result, the following boundary conditions apply:

- Only available for **not** activated general tool management
- No replacement tools can be defined
- No magazine can be defined
- No grinding tools can be defined

Activation

The type of D number management that is effective is parameterized using:

MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE = <value>

Value	Meaning
0	No flat D number management active
1	Flat D number structure with absolute **direct D programming** active

19.3.2 Creating a new D number (compensation block)

Programming

Tool offsets can be programmed with system variables $TC_DP1 to $TC_DP25. The content has the same meaning as previously.

The notation changes: A T number is no longer specified.

- "Flat D number" function **active**:
  $TC_DPx[d] = value; where x = parameter no., d = D number
  This means that data with this syntax can only be loaded to the NC when 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 NC memory when a D number that does not exist is created for the first time.

The maximum number of D or offset data blocks (max. 600) is set via the machine data:

MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA (tool offsets in TO area)

Data backup

The data backup is in the same format; i.e. a backup file that is created with the "Flat D number" function cannot be loaded to the NC of a control that has not activated the function.

This also applies in reverse for a transfer.

D number range

1 - 99 999 999

19.3.3 D number programming

MD18102 = 1

If MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE = 1, then D compensation is activated without reference to a certain tool.

`D0` still contains the previous significance, "Deselection of active compensation in NC".

Address extension of D

It is not possible to extend the address of D. Only one active compensation data block is possible for the tool path at a given time.

Programming

Programming in the part program is carried out as before. Only the value range of the programmed D number is increased.

Example 1:

MD parameterization		Meaning
MD22550 $MC_TOOL_CHANGE_MODE	= 0	Tool change with programming of T.
MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE	= 1	D number programming without reference to a certain tool.
MD20270 $MC_CUTTING_EDGE_DEFAULT	= -1	D compensation remains unchanged if tool is changed.

```
Program code             Comment
...
D92
X0                       ; Traverse with compensations from D92.
T17                      ; Outputs T=17 to the PLC
X1                       ; Traverse with compensations from D92.
D16
X2                       ; Traverse with compensations from D16.
```

Program code	Comment
D32000	
X3	; Traverse with compensations from D32000.
T29000	; Outputs T=29000 to the PLC.
X4	; Traverse with compensations from D32000.
D1	
X5	; Traverse with compensations from D1.
...	

Example 2:

MD22550 = 0

Program code	Comment
T1	
T2	
T3	
D777	; No waiting, D777 is activated, T3= programmed and active tool in the display, D777= programmed and active compensation.

Note

The tool change and the assignment of a D compensation to an actual tool are the responsibility of the NC program and of the PLC program, if applicable.

Delete D no. via part program

- **With** flat D number:
  $TC_DP1[d] = 0
  Compensation data block with the number D in the TO unit is deleted.
  The memory is then free for the definition of another D number.
- **Without** flat D number:
  $TC_DP1[t][d] = 0
  Cutting edge d of tool t is deleted.
- $TC_DP1[0] = 0
  All D compensations of the TO unit are deleted.

Active compensation data blocks (D numbers) cannot be deleted. This means, that it may be necessary to program `D0` before deleting.

Tool MDs

The following machine data affect the way tools and cutting edges (D numbers) work in the NC:

Machine data	Meaning
MD20270 $MC_CUTTING_EDGE_DEFAULT	Standard tool cutting edge after tool change
MD20130 $MC_CUTTING_EDGE_RESET_VALUE	Tool cutting edge - length compensation during power-up (RESET / TP end)
MD20120 $MC_TOOL_RESET_VALUE	Tool - length compensation during power-up (RESET / TP end)
MD20121 $MC_TOOL_PRESEL_RESET_VALUE	Preselected tool at RESET
MD22550 $MC_TOOL_CHANGE_MODE	Tool change with M function instead of T function
MD22560 $MC_TOOL_CHANGE_M_CODE	M function for tool change
MD20110 $MC_RESET_MODE_MASK	Definition of basic control setting after RESET / TP end
MD20112 $MC_START_MODE_MASK	Definition of basic control setting at NC start

19.3.4 Programming the T number

When the "Flat D number structure" function is active, NC address T continues to be evaluated, i.e., the programmed T number and the active T number are displayed. However, the NC determines the D number without reference to the programmed T value.

The NC detects 1 master spindle per channel (via the spindle number, which can be set using MD). Compensations and the `M6` command (tool change) are only calculated in reference to the master spindle.

An address extension T is interpreted as a spindle number (e.g., T2 = 1; tool 1 to be selected on spindle 2); a tool change is only detected if spindle 2 is the master spindle.

19.3.5 Programming M6

MD22550 and MD22560

The NC detects 1 master spindle per channel (via the spindle number, which can be set using MD). Compensations and the `M6` command (tool change) are only calculated in reference to the master spindle.

Whether the tool change command is performed with an M function is defined via the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool offset for M function)

T is used as the tool preparation command.

The name of the M function for the tool change is defined via the machine data:

MD22560 $MC_TOOL_CHANGE_M_CODE (M function for tool change)

The default is `M6`. An address extension of `M6` is identified 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 required, 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
```

19.3.6 Program test

MD20110

To have the active tool and the tool compensation included as follows, can be defined via the machine data:

MD20110 $MC_RESET_MODE_MASK, Bit 3 (Definition of control default settings after RESET/TP end).

Value		Significance
Bit 3	= 1	From the last test program to finish in test mode
	= 0	From the last program to finish before activation of the program test

Prerequisite

The bits 0 and 6 must be set by the machine data:

MD20110 $MC_RESET_MODE_MASK, Bit 3 (Definition of control default settings after RESET/TP end).

19.3.7 Tool management or "Flat D number structure"

Features

Tool management

The active NC tool management works on the basis of the following assumptions:

1. Tools are managed in magazines.
2. Cutting edges are monitored, that means limits reached cause the tool to be disabled.
3. Replacement tools: Tools are programmed for selection only on the basis of their name. The NC selects the specific tool.

Note

For SINUMERIK 828D, this function is only available as an option!

This means that it only makes sense to use the tool management when specific tools have been defined which are managed by the NC.

Flat D number structure

The use of a flat D number structure means that the tool management is performed outside the NC (PLC) and no reference is made to the T numbers.

Tool management OR flat D number structure

The simultaneous use of tool management (NC) and flat D number structure (PLC) makes no sense because the main argument for the use of the tool management is the **time** factor. This is only ensured if the management tasks are performed by the NC. This is not the case for the flat D number structure, however.

Note

If tool management AND the flat D number structure are activated at the same time, the tool management takes priority.

19.4 Tool cutting edge

19.4.1 General information

Tool cutting edge

The following data are used to describe a tool cutting edge uniquely:

- Tool type (end mill, drill, etc.)
- Geometrical description
- Technological description

Tool parameter

The geometrical description, the technological description and the tool type are mapped to tool parameters for each tool cutting edge.

The following tool parameters are available for the relevant tool types:

Tool parameter	Meaning	Remark
1	Tool type	
2	Cutting edge position	For turning tools or for milling/grinding tools with 2D TRC with contour tools.
Geometry - tool lengths		
3	Length 1	
4	Length 2	
5	Length 3	
Geometry - tool shape		
6	Radius 1	With 2D TRC. Turning: Rounding radius of cutting edge 3D face milling: Shank radius Slotting saw: Diameter
7	Radius 2	3D face milling: Corner radius Slotting saw: Slot width
8	Length 4	Slotting saw: Projection
9	Length 5	3D face milling: Upper bevel cutter diameter (of spherical tools)
10	Angle 1	Turning: Holder angle 2D TRC with contour tools: Minimum limit angle

Tool parameter	Meaning	Remark
11	Angle 2	Turning: Cutting direction 2D TRC with contour tools: Maximum limit angle 3D face milling: Angle between side line and tool longitudinal axis (of conical tools)
Wear - tool lengths		
12	Wear length 1	
13	Wear length 2	
14	Wear length 3	
Wear - tool shape		
15	Wear radius 1	
16	Wear radius 2	
17	Wear length 4	
18	Wear length 5	
19	Wear angle 1	
20	Wear angle 2	
Tool base dimension / adapter dimension		
21	Basic length 1	
22	Basic length 2	
23	Basic length 3	
Technology		
24	Undercut angle	Only for turning tools.
25	Reserved*	

* "Reserved" means that this tool parameter is not used and is reserved for expansions.

3D face milling

Milling cutter types 111, 120, 121, 130, 155, 156 and 157 are given special treatment for 3D-face milling by evaluating tool parameters (1-23).

References

For more information about various tool types, see:

- Function Manual, Basic Functions; Tool Offset (W1), Chapter: "Tool type (tool parameters)"
- Programming Manual, Fundamentals; Chapter: "Tool compensations" > "List of tool types"
- Functions Manual - Special Functions; 3D tool radius compensation (W5)

19.4.2 Tool parameter 1: Tool type

Description

The tool type (3digit number) defines the tool in question. The selection of this tool type determines further components such as geometry, wear and tool base dimensions in advance.

Conditions

The following is applicable to the "Tool type" parameter:

- The tool type must be specified for each tool cutting edge.
- Only the values specified can be used for the tool type.
- Tool type "0" (zero) means that no valid tool has been defined.

Tool types and tool parameters

Different tool types and the most important tool parameters are listed in the following table. The tool parameters available for a certain tool type are designated with "x".

Table of most important tool parameters (only tool type is necessary)																	
1	Tool parameter $TC_DP	2	3	4	5	6	7	8/9	12/13/14	15/16	21/22/23	24	TPG 3	TPG 4/5	TPG 6	TPG 7	TPG 8
Tool type	Billing by NCK	Cutting edge position	Geometry – length 1	– Length 2	– Length 3	Geometry – Radius 1	Slot width / – Radius 2	Projection / – Length 4/5	Wear – Length 1/2/3	Wear – Radius 1/2	Adap/Tool base dimension – Length 1/2/3	Undercut angle	– Min. wheel radius	– Min./curr. wheel width	– Max speed	– Max. circumferential velocity	– Angle of inclined wheel
Milling tool and tapping tools (all other)																	
100	Mill tool acc. to CLDATA*1		x	x	x	x			x	x	x						
110	Ball end mill		x	x	x	x			x	x	x						
120	End mill		x	x	x	x			x	x	x						
121	End mill with corner rounding		x	x	x	x	x		x	x	x						
130	Angle head cutter		x	x	x	x			x	x	x						
131	Angle head mill with corner rounding		x	x	x	x	x		x	x	x						
140	Facing tool		x	x	x	x			x	x	x						
145	Thread cutter		x	x	x	x			x	x	x						
150	Side mill		x	x	x	x			x	x	x						
155	Bevel cutter		x	x	x	x			x	x	x						
200	Twist drill		x	x	x	x			x		x						
205	Drill		x	x	x	x			x		x						
210	Boring bar		x	x	x	x			x		x						
220	Center drill		x	x	x	x			x		x						
230	Countersink		x	x	x	x			x		x						
231	Counterbore		x	x	x	x			x		x						
240	Tap regular thread		x	x	x	x			x		x						
241	Tap fine thread		x	x	x	x			x		x						
242	Tap Whitworth thread		x	x	x	x			x		x						
250	Reamer		x	x	x	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	Facing wheel	x	x	x	x	x			x	x	x					x	
411	Facing wheel with monitoring	x	x	x	x	x			x	x	x		x	x	x	x	
413	As for 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					
550	Button tool / Sectional steel (TOOLMAN)	x	x	x	x	x			x	x	x	x					
Special tools (700)																	
700	Slotting saw		x	x	x	x	x	x	x	x	x						
CLDATA*1 "cutter Location Data" — Bl 1SUG*2 —	Cutter location data in accordance with DIN 66215, Grinding wheel peripheral speed																

Note

The tool type has no significance in the turning tool groups.

Nonlisted numbers are also permitted, in particular with grinding tools (400-499).

Tool offset data

Tool offset data (TOA data) is stored in the system variables.

Example Slotting saw tool type (Type 700)

	Geometry	Wear	Base	Einheit
Length compensation				
Length 1	$TC_DP3	$TC_DP12	$TC_DP21	mm
Length 2	$TC_DP4	$TC_DP13	$TC_DP22	mm
Length 3	$TC_DP5	$TC_DP14	$TC_DP23	mm
Saw blade compensation				
Diameter d	$TC_DP6	$TC_DP15		mm
Slot width b	$TC_DP7	$TC_DP16		mm
Projection k	$TC_DP8	$TC_DP17		mm

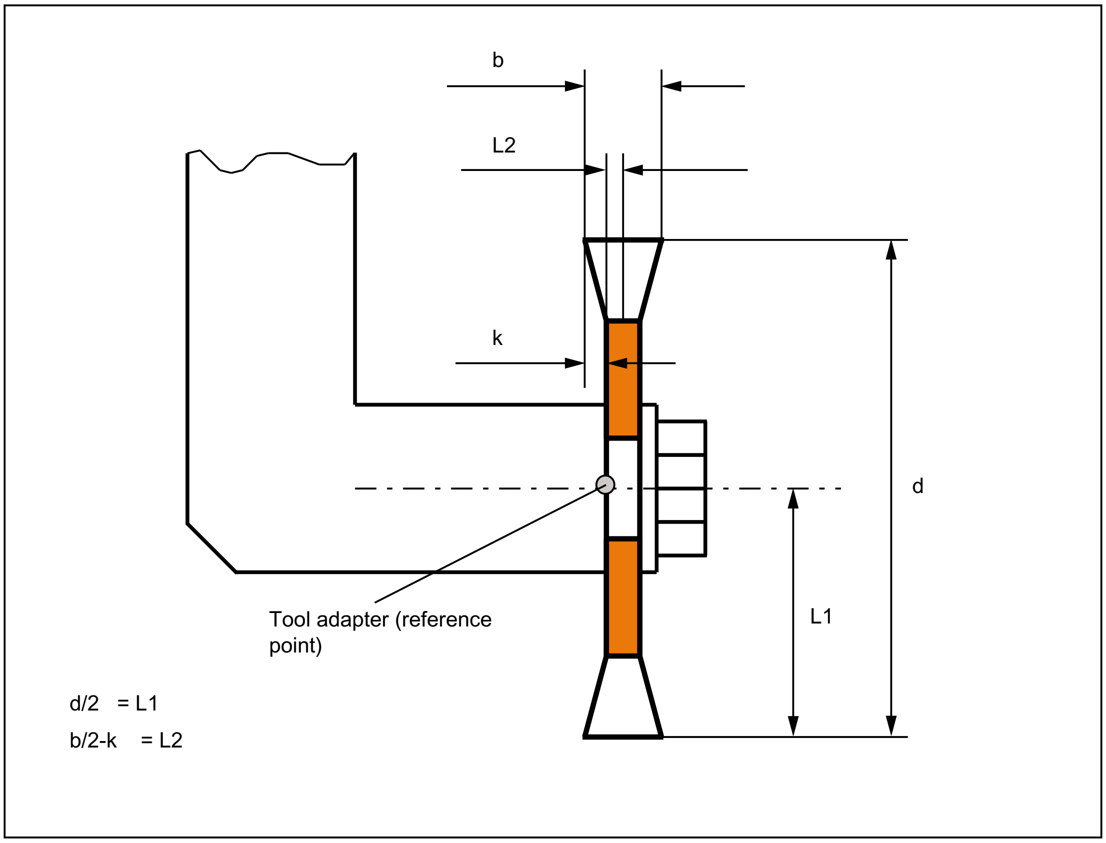


Figure 19-3 Geometry of slotting saw (analogous to angle head cutter)

The width of the saw blade is accounted for with tool radius compensation (`G40` to `G42` as follows:

Command	Significance
`G40`	No saw blade compensation
`G41`	Saw blade compensation left
`G42`	Saw blade compensation right

19.4.3 Tool parameter 2: Cutting edge position

Description

The cutting edge position describes the position of the tool tip P in relation to the cutting edge center point S. It is entered in tool parameter 2.

The cutting edge position is required together with the cutting edge radius (tool parameter 8) for the calculation of the tool radius compensation for turning tools (tool type 5xx).

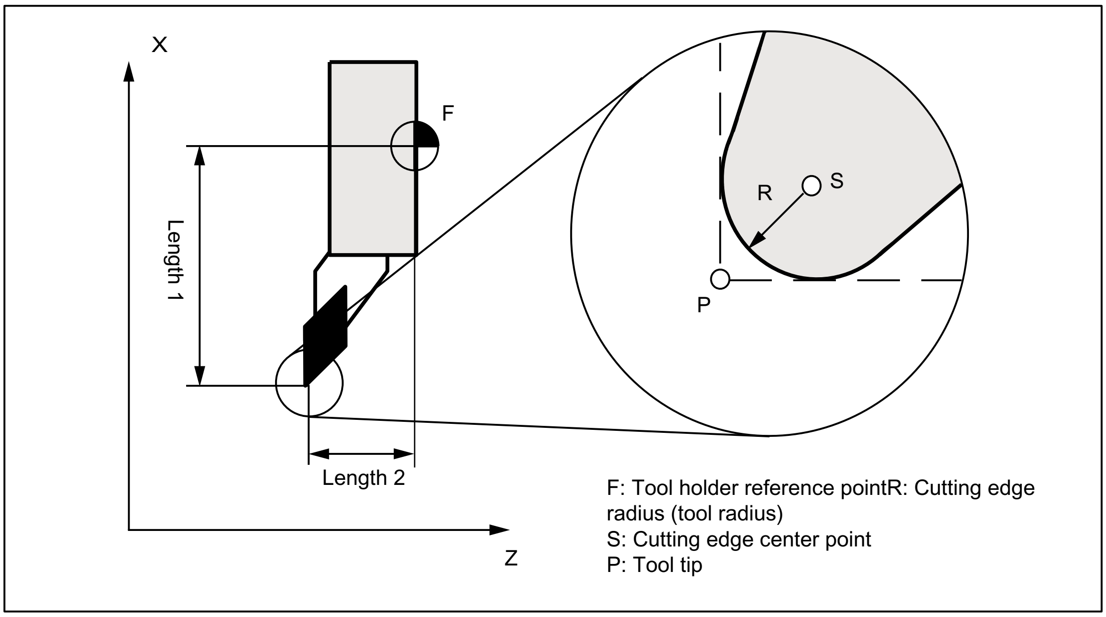

X
F
Length 1
R
S
P
Length 2
Z
F: Tool holder reference pointR: Cutting edge radius (tool radius)
S: Cutting edge center point
P: Tool tip

Figure 19-4 Dimensions for turning tools: Turning tool

Cutting edge position parameter values

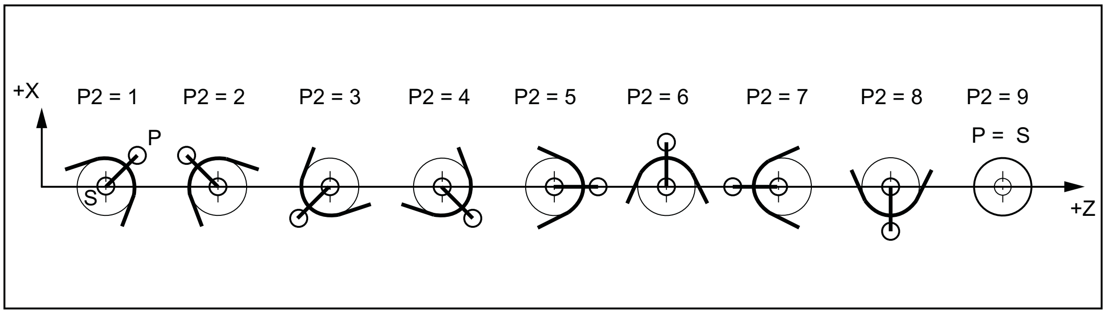


Figure 19-5 Tool parameter 2 (P2): Machining behind the turning center

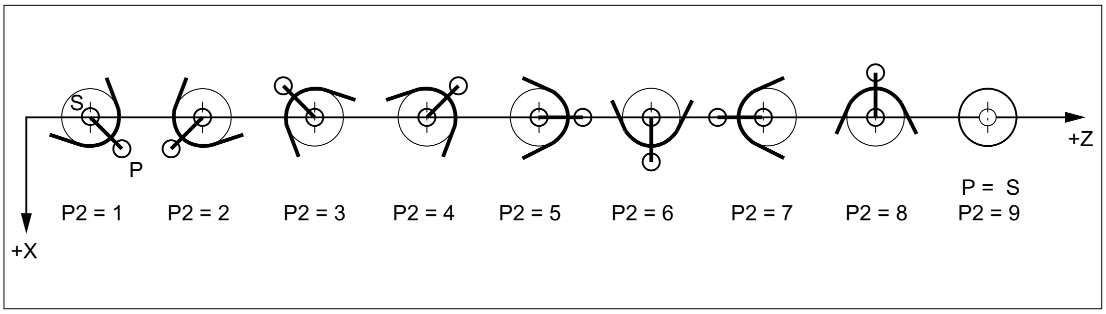


Figure 19-6 Tool parameter 2 (P2): Machining in front of the turning center

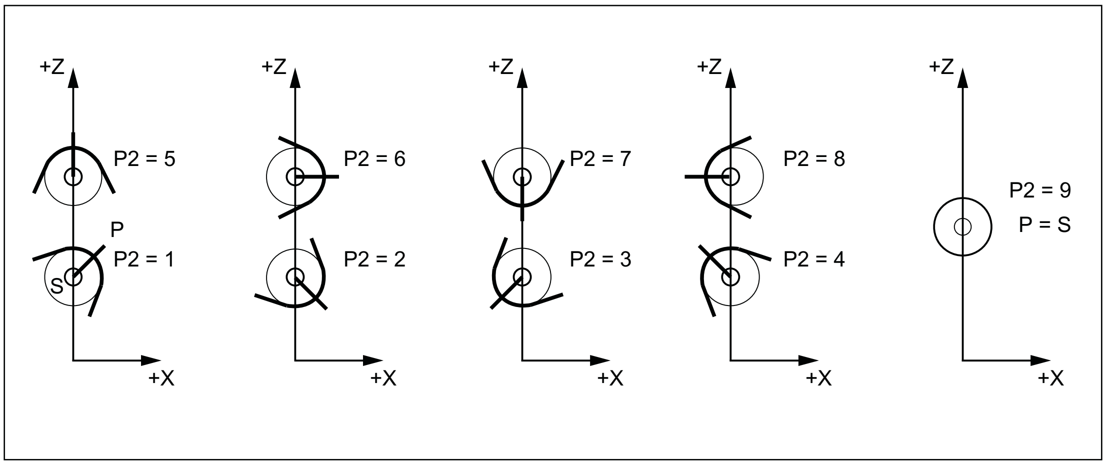


Figure 19-7 Tool parameter 2 (P2): Cutting edge position for vertical boring and turning mills

Special points to be noted

- If the cutting edge center point S is used instead of point P as a reference point to calculate the tool length compensation, the identifier 9 must be entered for the cutting edge position.
- The identifier 0 (zero) is not permitted as a cutting edge position.

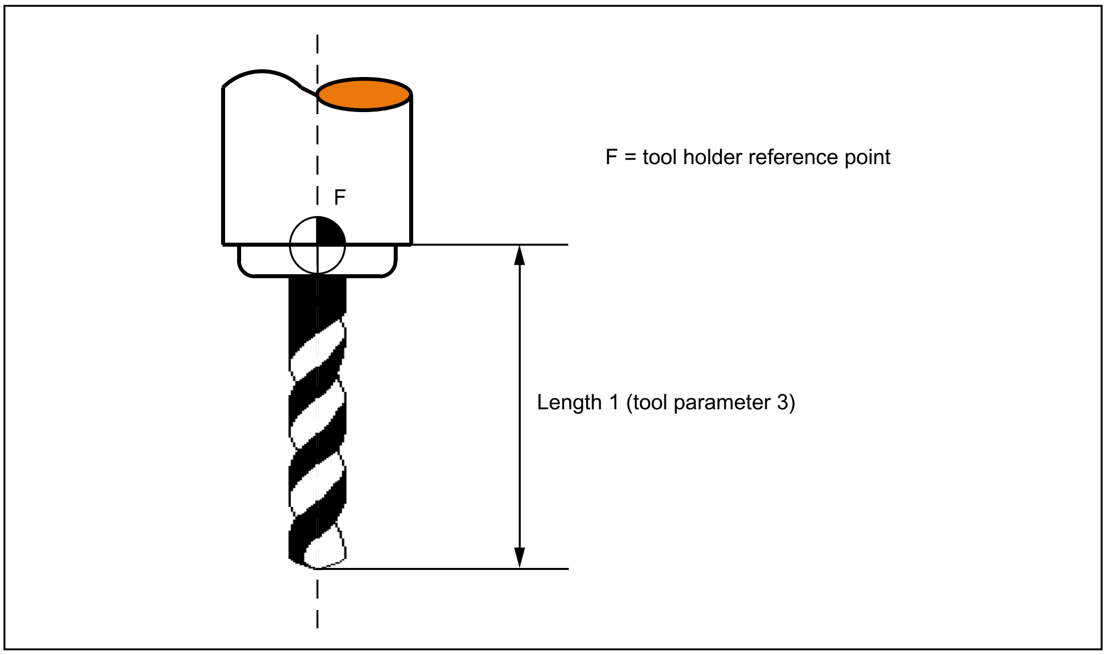
19.4.4 Tool parameters 3 - 5: Geometry - tool lengths

Description

The lengths of the tools are required for the geometry tool length compensation. They are input as tool lengths 1 to 3 in the tool parameters 3 to 5. The following length specifications must be entered as a minimum for each tool type:

Tool type	Required tool lengths
Tool type 12x, 140, 145, 150:	Tool length 1
Tool type 13x:	Tool length 1 to 3 (depending on plane `G17`-`G19`)
Tool type 2xx:	Tool length 1
Tool type 5xx:	Tool length 1 to 3

Example Twist drill (tool type 200) with tool length (tool parameter 3)



Note

All three tool parameters 3 to 5 (tool length 1 to 3) are always calculated in the three geometry axes, irrespective of the tool type.

If more tool lengths are input in the tool parameters 3 to 5 for a tool type than is required as the minimum, then these extra tool lengths are settled in the geometry axes without any alarm.

Special points to be noted

The active size of the tool is only defined when the geometry tool length compensation (tool parameters 3 to 5) and the wear tool length compensation (tool parameters 12 to 14) are added together. The base-dimension/adapter-dimension tool length compensation is also added in order to calculate the total tool length compensation in the geometry axes.

References

For information about entering tool dimensions (lengths) in tool parameters 3 to 5 (tool lengths 1 to 3) and how these are calculated in the three geometry axes, please refer to → Operating Manual.

19.4.5 Tool parameters 6 - 11: Geometry - tool shape

Meaning

The shape of the tool is defined using the tool parameters 6 to 11. The data is required for the geometry tool radius compensation.

In most cases, only tool parameter 6 (tool radius 1) is used.

Tool parameter	Meaning	Use	
6	Tool radius 1	The tool radius must be specified for the following tool types:	
		1xx	Milling tools
		5xx	Turning tools Tool parameter 2: Cutting edge position (Page 1480) must also be specified for turning tools.
		Note Tool type 2xx: A tool radius does not have to be specified for drilling tools.	
7	Tool radius 2	Not used	
8	Tool length 4	Not used	
9	Tool length 5	Not used	
10	Tool angle 1	Not used	
11	Tool angle 2	Not used	

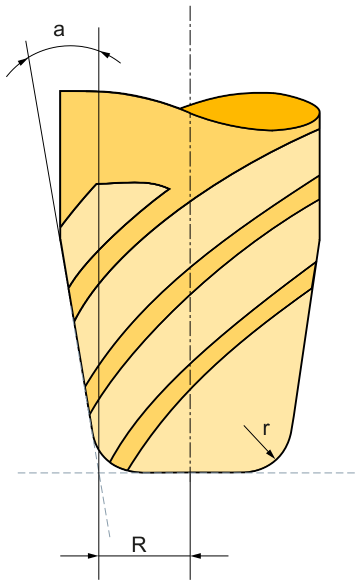
2D TRC with contour tools

For the definition of contour tools with multiple tool cutting edges, the minimum and maximum limit angle can be entered. Both limit angles each relate to the vector of the cutting edge center point to the cutting edge reference point and are counted clockwise.

Tool angle 1	Minimum limit angle per tool cutting edge
Tool angle 2	Maximum limit angle per tool cutting edge

3D face milling

The tool parameters relevant to the tool description in 3D face milling are dependent on the tool type used. So, for example, for a ball end mill, only tool parameter 6, or for a bevel cutter with corner radius, additionally tool parameters 7 and 11 are relevant.



R	Tool parameter 6: Shank radius
r	Tool parameter 7: Corner radius
a	Tool parameter 11: Angle between side line and tool longitudinal axis (of conical tools)

Figure 19-8 Tool description for 3D face milling using the example of a bevel cutter with corner rounding

References

Please refer to the following documentation for information about entering tool shapes (radius for tool radius compensation) in tool parameters 6 to 11 and how these are calculated by geometry tool radius compensation in the three geometry axes:

- Programming Manual, Fundamentals; Chapter: "Tool compensations" > "2½ D tool compensation"
- Function Manual, Special Functions; Chapter "W5: 3D tool radius compensation"

For 3D face milling, please refer to:

- Function Manual, Special Functions; Chapter "W5: 3D tool radius compensation"

19.4.6 Tool parameters 12 - 14: Wear - tool lengths

Description

While geometry tool length compensation (tool parameters 3 to 5) is used to define the size of the tool, wear tool length compensation can be used to correct the change in the active tool size.

The active tool dimensions can change due to:

- Differences between the tool fixture on the tool measurement machine and the tool fixture on the machine tool
- Tool wear caused during service life by machining
- Definition of the finishing allowances

Active tool size

The geometry tool compensation (tool parameters 3 to 5) and the wear tool length compensation (tool parameters 12 to 14) are added together (geometry tool length 1 is added to wear tool length 1, etc.) to arrive at the size of the active tool.

19.4.7 Tool parameters 15 - 20: Wear - tool shape

Description

While geometry tool radius compensation (tool parameters 6 to 11) is used to define the shape of the tool, wear tool radius compensation can be used to correct the change in the active tool shape.

The active tool dimensions can change due to:

- Tool wear caused during service life by machining
- Definition of the finishing allowances

Active tool shape

The geometry tool radius compensation (tool parameters 6 to 11) and the wear tool radius compensation (tool parameters 15 to 20) are added together (geometry tool radius 1 is added to wear tool radius 1, etc.) to arrive at the shape of the active tool.

19.4.8 Tool parameters 21 - 23: Tool base dimension/adapter dimension

Description

Tool base dimension/adapter dimension can be used when the reference point of the toolholder (tool size) differs from the reference point of the toolholder.

This is the case when:

- The tool and the tool adapter are measured separately but are installed on the machine in one unit (the tool size and adapter size are entered separately in a cutting edge).
- The tool is used in a second tool fixture located in another position (e.g. vertical and horizontal spindle).
- The tool fixtures of a tool turret are located at different positions.

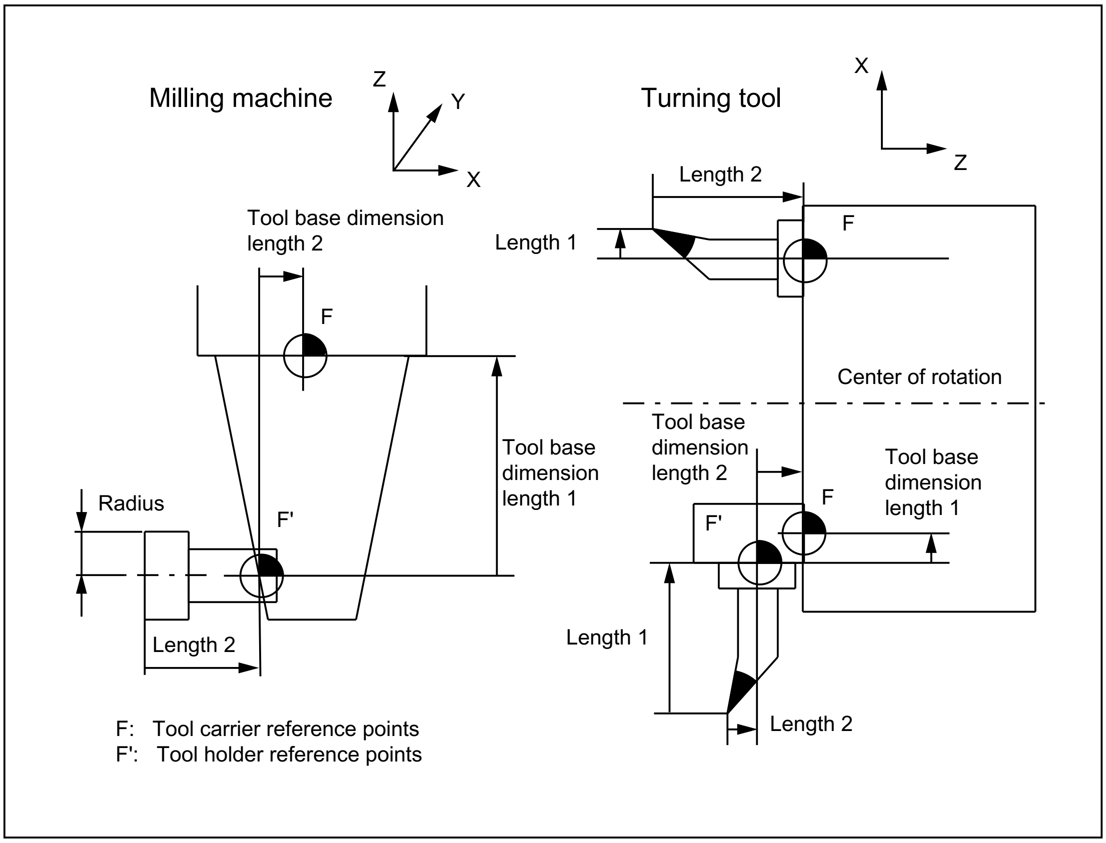

Milling machine
Z
Y
X
Turning tool
X
Z
Length 2
F
Length 1
Tool base dimension length 2
F
Tool base dimension length 1
Center of rotation
Tool base dimension length 2
Tool base dimension length 1
F
Radius
F'
F'
Length 2
Length 1
Length 2
F: Tool carrier reference points
F': Tool holder reference points

Figure 19-9 Application examples for base-dimension/adapter-dimension TLC

Tool basic length 1 to 3 (tool parameters 21 to 23)

In order that the discrepancy between the toolholder reference point F and the toolholder reference point F' can be corrected on the three geometry axes (three dimensional), all 3 basic lengths are active irrespective of the tool type. In other words, a twist drill (tool type 200) with a tool length compensation (length 1) can also have a tool base dimension/adapter dimension in 3 axes.

References

Please refer to the following documentation for more information about base-dimension/ adapter-dimension tool length compensation:

- Programming Manual, Fundamentals

19.4.9 Tool parameter 24: Undercut angle

Meaning

Certain turning cycles, in which traversing motions with tool clearance are generated, monitor the tool clearance angle of the active tool for possible contour violations.

Value range

The angle (0 to 90° with no leading sign) is entered in tool parameter 24 as the tool clearance angle.

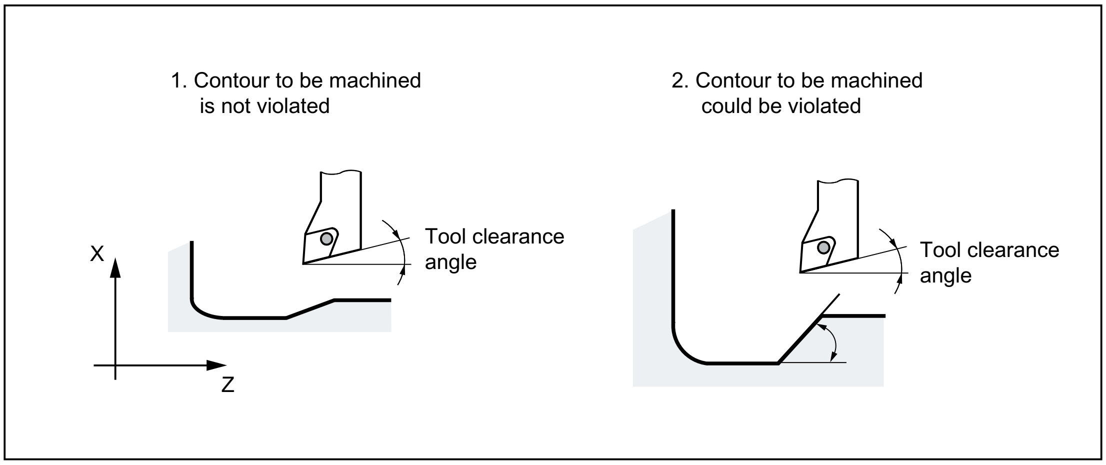

1. Contour to be machined is not violated
2. Contour to be machined could be violated
Tool clearance angle
Tool clearance angle
X
Z

Figure 19-10 Tool clearance angle of the turning tool during relief cutting

Machining type, longitudinal or transverse

The tool clearance angle is entered in different ways according to the type of machining (longitudinal or face). If a tool is to be used for both longitudinal and face machining, two cutting edges must be entered for different tool clearance angles.

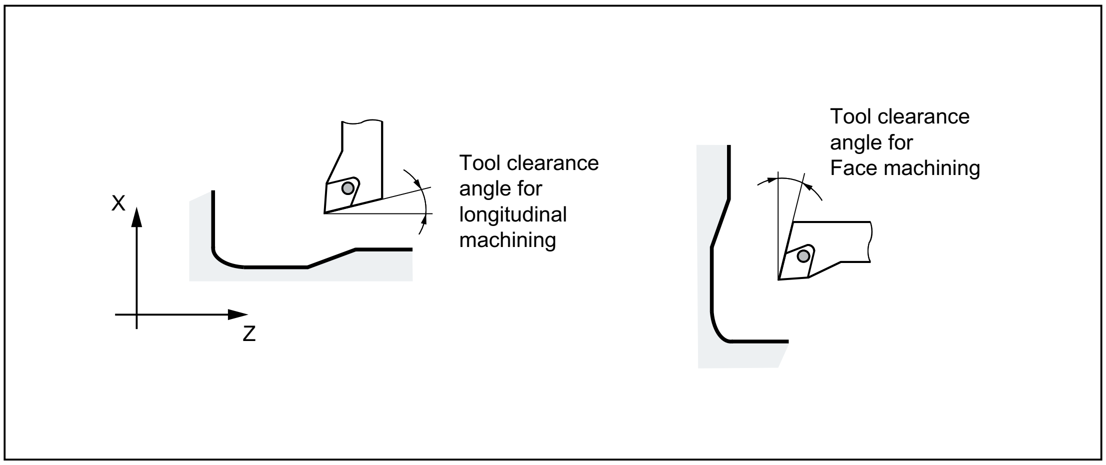

Tool clearance angle for longitudinal machining
Tool clearance angle for Face machining
X
Z

Figure 19-11 Tool clearance angle for longitudinal and face machining

Note

If a tool clearance angle (tool parameter 24) of zero is entered, relief cutting is not monitored in the turning cycles.

References

Please refer to the following documentation for a detailed description of the tool clearance angle:

- Programming Manual Cycles

19.4.10 Tools with a relevant tool point direction

The following must be observed for tools with relevant cutting edge position:

- The straight line between the tool edge center points at the block start and block end is used to calculate intersection points with the approach and retraction block. The difference between the tool edge reference point and the tool edge center point is superimposed on this movement.
  For approach and/or retraction with `KONT`, the movement is superimposed in the linear subblock of the approach or retraction movement. Therefore, the geometric conditions for tools with or without relevant cutting edge position are identical.
- In circle blocks and in motion blocks containing rational polynomials with a denominator degree > 4, it is not permitted to change a tool with active tool radius compensation in cases where the distance between the tool edge center point and the tool edge reference point changes. With other types of interpolation, it is now possible to change when a transformation is active (e.g. `TRANSMIT`).
- For tool radius compensation with variable tool orientation, the transformation from the tool edge reference point to the tool edge center point can no longer be performed by means of a simple zero offset. Tools with a relevant cutting edge position are therefore not permitted for 3D peripheral milling (an alarm is output).

  Note

  The subject is irrelevant with respect to face milling as only defined tool types without relevant cutting edge position are permitted for this operation anyway. (A tool with a type, which has not been explicitly approved, is treated as a ball end mill with the specified radius. A cutting edge position parameter is ignored).

19.5 2D tool radius compensation (2D-WRK)

19.5.1 General information

Note

For tool radius compensation (TRC) see:
References:
Programming Manual Fundamentals

Only the Programming Guide contains a complete technical description of the tool radius compensation (TRC) and its special aspects.

Why TRC?

The contour (geometry) of the workpiece programmed in the part program should be independent of the tools used in production. This makes it necessary to draw the values for the tool length and tool radius from a current offset memory. Tool radius compensation can be used to calculate the equidistant path to the programmed contour from the current tool radius.

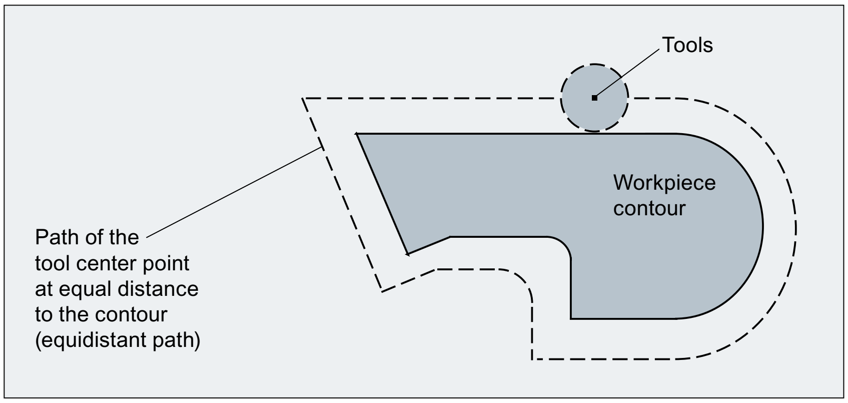

Tools
Workpiece contour
Path of the tool center point at equal distance to the contour (equidistant path)

Figure 19-12 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`

19.5.2 Selecting the TRC (G41/G42)

Direction of compensation

TRC calculates a path, which is equidistant to the programmed contour. Compensation can be performed on the left- or righthand side of the programmed contour in the direction of motion.

Command	Significance
`G41`	TRC on the lefthand side of the contour in the direction of motion
`G42`	TRC on the righthand side of the contour in the direction of motion
`G40`	Deselection of TRC

Intermediate blocks

In general, only program blocks with positions on geometry axes in the current plane are programmed when TRC is active. However, dummy blocks can still also be programmed with active TRC. Dummy blocks are program blocks, which do not contain any positions on a geometry axis in the current plane:

- Positions on the infeed axis
- Auxiliary functions,
- etc.

The maximum number of dummy blocks can be defined in the machine data:

MD20250 $MC_CUTCOM_MAXNUM_DUMMY_BLOCKS (Max no. of dummy blocks with no traversing movements for TRC).

Special points to be noted

- TRC can only be selected in a program block with `G0` (rapid traverse) or `G1` (linear interpolation).
- A tool must be loaded (T function) and the tool cutting edge (tool compensation) (`D1` to `D9`) activated no later than in the program block with the tool radius compensation selection.
- Tool radius compensation is not selected with a tool cutting edge/tool compensation of `D0`.
- If only one geometry axis is programmed on the plane when tool radius compensation is selected, the second axis is automatically added on the plane (last programmed position).
- If no geometry axis is programmed for the current plane in the block with the tool radius compensation selection, no selection takes place.
- If tool radius compensation is deselected (`G40`) in the block following tool radius compensation selection, no selection takes place.
- If tool radius compensation is selected, the approach behavior is determined by the `NORM`/`KONT` instructions.

19.5.3 Approach and retraction behavior (NORM/KONT/KONTC/KONTT)

NORM and KONT

The `NORM` and `KONT` instructions can be used to control approach behavior (selection of tool radius compensation with `G41`/`42`) and retraction behavior (deselection of tool radius compensation with `G40`):

Command	Meaning
`NORM`	Normal setting at start point / end point (basic setting)
`KONT`	Travel around contour at start point / end point
`KONTC`	Approach/retraction with constant curvature
`KONTT`	Approach/retraction with constant tangent

Special features

- `KONT` only differs from `NORM` when the tool start position is behind the contour.

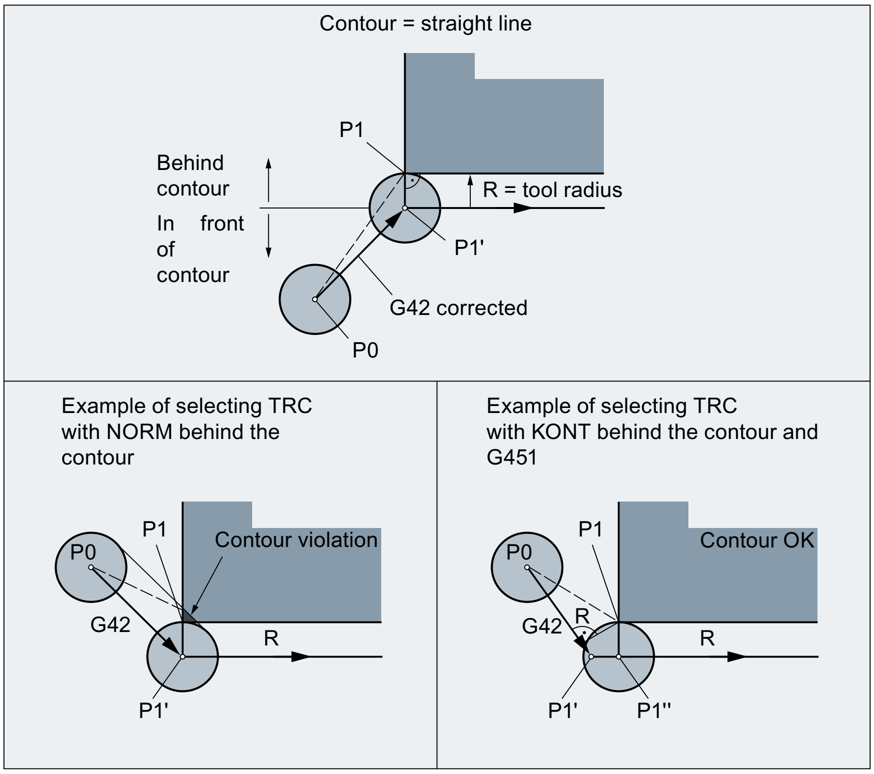


Figure 19-13 Example for selecting TRC with `KONT` or `NORM` in front of and behind the contour

- `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 of KONTC

The two figures below show a typical application for approach and retraction with constant curvature:

The full circle is approached beginning at the circle center point. The direction and curvature radius of the approach circle at the block end point are identical to the values of the next circle. Infeed takes place in the Z direction in both approach/retraction blocks simultaneously.

The associated NC program segment is as follows:

```
$TC_DP1[1,1]=121                                  Milling tool
$TC_DP6[1,1]=10                                   Radius 10 mm
N10 G1 X0 Y0 Z60 G64 T1 D1 F10000
N20 G41 KONTC X70 Y0 Z0
N30 G2 I-70                                       Full circle
N40 G40 G1 X0 Y0 Z60
N50 M30
```

Explanation:

In this example, a full circle with a radius of 70 mm is machined in the X/Y plane. Since the tool has a radius of 10 mm, the resulting tool center point path describes a circle with a radius of 60 mm. The start/end points are at X0 Y0 Z60, with the result that a movement takes place in the Z direction at the same time as the approach/retraction movement in the compensation plane.

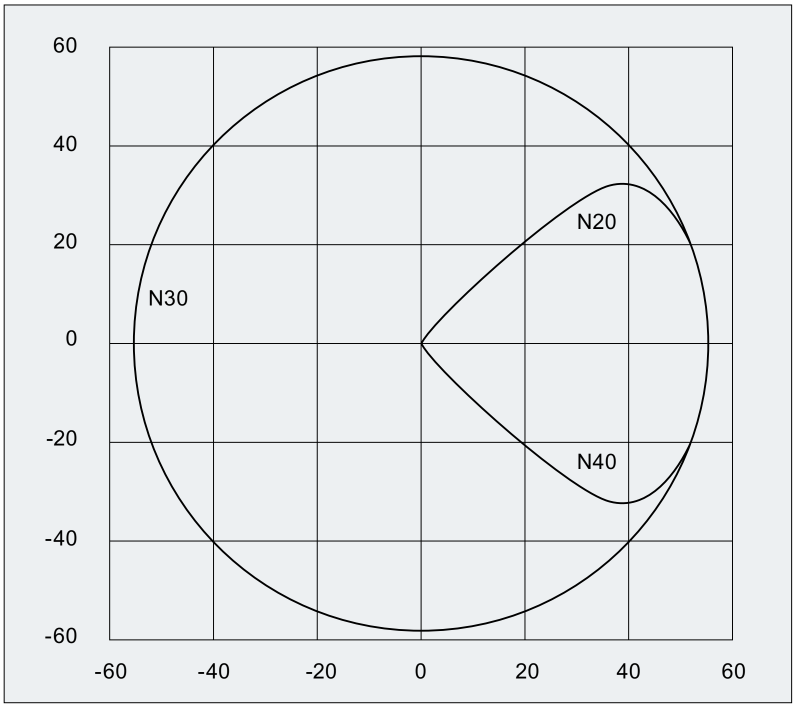


Figure 19-14 Approach and retraction with constant curvature during inside machining of a full circle: Projection in the X-Y plane.

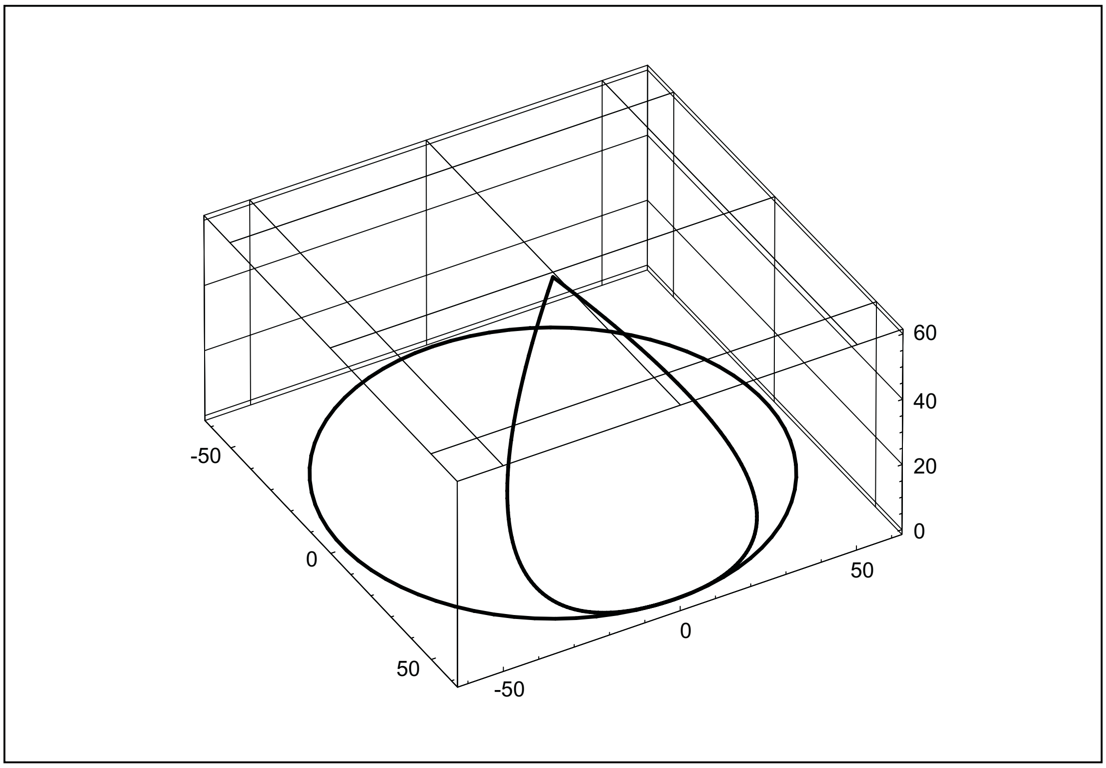


Figure 19-15 Approach and retraction with constant curvature during inside machining of a full circle: 3D representation.

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 block is at X40 Y30. The transition between the circular block and the retraction block is at the zero point. Due to the extended continuity of curvature associated with `KONTC`, the retraction block first executes a movement with a negative Y component. This will often be undesired. This response does not occur with the `KONTT` retraction block. However, with this block, an acceleration step change occurs at the block transition.

If the `KONTT` or `KONTC` block is the approach block rather than the retraction block, the contour is exactly the same, but is simply machined in the opposite direction, i.e. the **approach and retraction behavior are symmetrical.**

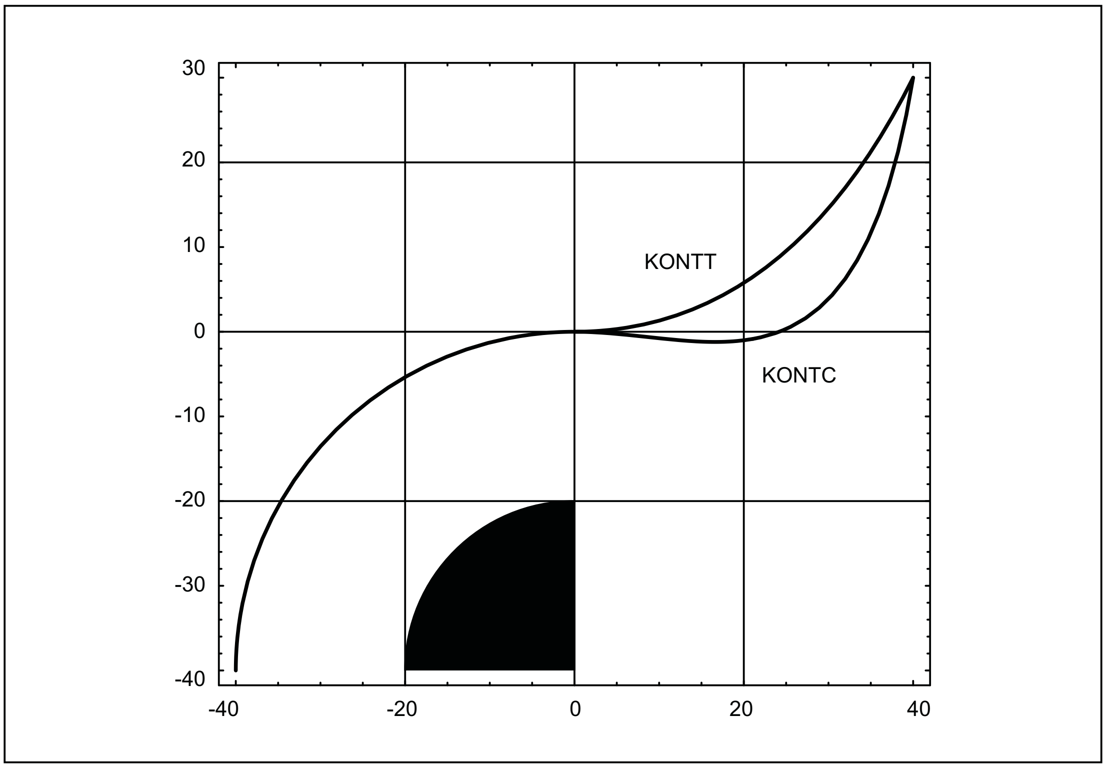


Figure 19-16 Differences between `KONTT` and `KONTC`

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.

19.5.4 Smooth approach and retraction

19.5.4.1 Function

Description

The SAR (Smooth Approach and Retraction) function is used to achieve a tangential approach to the start point of a contour, regardless of the position of the start point.

The approach behavior can be varied and adapted to special needs using a range of additional parameters.

The two functions, smooth approach and smooth retraction, are largely symmetrical. The following section is, therefore, restricted to a detailed description of approach; special reference is made to differences affecting retraction.

Sub-movements

There are maximum 4 sub-movements in case of soft retraction and approach with the following positions:

- Start point of the movement P_0
- Intermediate points P_1, P_2 and P_3
- End point P_4

Points P_0, P_3 and P_4 are always defined. Intermediate points P_1 and P_2 can be omitted, according to the parameters defined and the geometrical conditions.

On retraction, the points are traversed in the reverse direction, i.e. starting at P_4 and ending at P_0.

19.5.4.2 Parameters

The response of the smooth approach and retraction function is determined by up to 9 parameters:

Non-modal G command for defining the approach and retraction contour

This G command 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

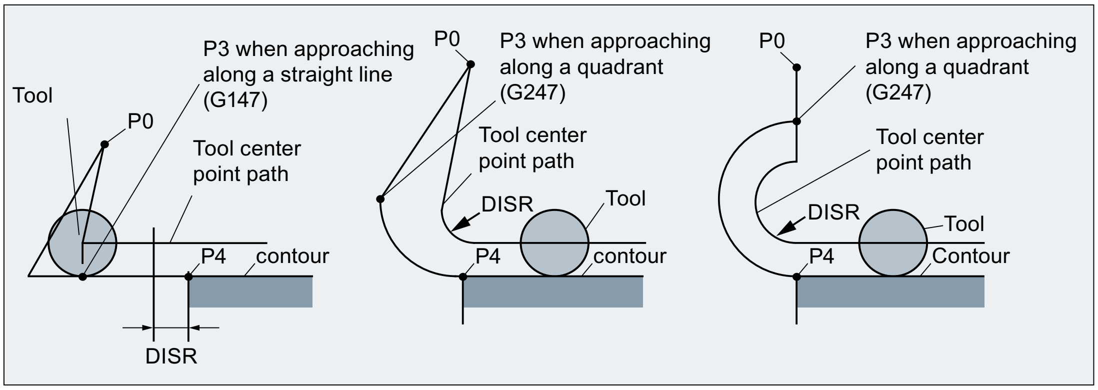

Tool
P0
P3 when approaching along a straight line (G147)
Tool center point path
P4
contour
DISR
P0
P3 when approaching along a quadrant (G247)
Tool center point path
DISR
Tool
P4
contour
P0
P3 when approaching along a quadrant (G247)
Tool center point path
DISR
Tool
P4
Contour

Figure 19-17 Approach behavior depending on `G147` to `G347` and `DISR` (with simultaneous activation of tool radius compensation)

Modal G command for defining the approach and retraction contour

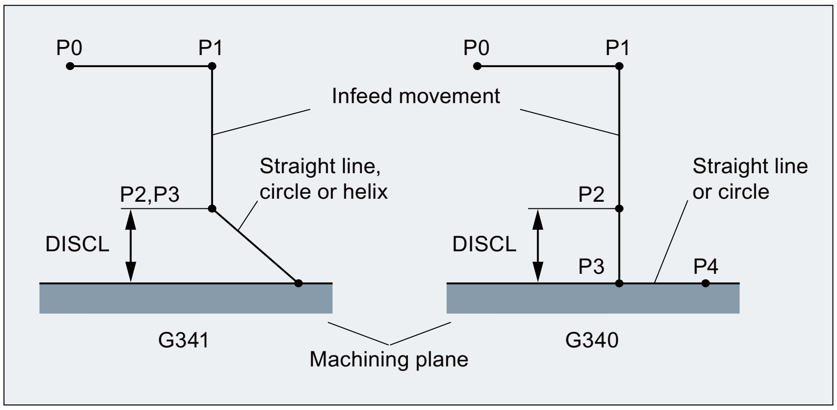
This G command is only relevant if the approach contour is a quadrant or semicircle. The approach and retraction direction can be determined as follows:

- `G140`:
  Defining the approach and retraction direction using active tool radius compensation. (`G140` is the basic setting value.)
  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 command (G340, G341), which defines the subdivision of the movement into individual blocks from the start point to the end point



`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`.

Figure 19-18 Sequence of the approach movement depending on `G340/G341`

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

`DISR`Specifies the length of a straight approach line or the radius of an approach arc.

Retraction/approach with straight lines

On approach/retraction along a straight line, `DISR` specifies the distance from the cutter edge to the start point of the contour, i.e. the length of the straight line with active TRC is calculated as the total of the tool radius and the programmed value of `DISR`.

An alarm is displayed:

- If `DISR` is negative and the amount is greater than the tool radius (the length of the resulting approach line is less than or equal to zero).

Retraction/approach with circles

Approach/retraction with circles
`DISR` indicates always the radius of the tool center point path. If tool radius compensation is activated, a circle is generated internally, the radius of which is dimensioned such that the tool center path is derived, in this case also, from the programmed radius.

An alarm is output on approach and retraction with circles:

- If the radius of the circle generated internally is zero or negative
- If `DISR` is not programmed
- If the radius value ≤ 0.

DISCL

`DISCL`specifies the distance from point P2 from the machining plane.

If the position of point P_2 is to be specified by an absolute reference on the axis perpendicular to the circle plane, the value must be programmed in the form `DISCL = AC( ....)`.

If `DISCL` is not programmed, points P_1, P_2 and P_3 are identical with `G340` and the approach contour is mapped from P_1 to P_4.

The system checks that the point defined by `DISCL` lies between P_1 and P_3, i.e. in all movements, which have a component perpendicular to the machining plane (e.g. infeed movements, approach movements from P_3 to P_4), this component must have the same leading sign. It is not permitted to change direction. An alarm is output if this condition is violated.

On detection of a direction reversal, a tolerance is permitted that is defined by the machine data:

MD20204 $MC_WAB_CLEARANCE_TOLERANCE (direction reversal on SAR).

However, if P_2 is outside the range defined by P_1 and P_3 and the deviation is less than or equal to this tolerance, it is assumed that P_2 is in the plane defined by P_1 and/or P_3.

Example:

An approach is made with `G17` starting at position Z=20 of point P_1. The SAR plane defined by P_3 is at Z=0. The point defined by `DISCL` must, therefore, lie between these two points. MD20204=0.010. If P_2 is between 20.000 and 20.010 or between 0 and -0.010, it is assumed that the value 20.0 or 0.0 is programmed. The alarm is output if the Z position of P_2 is greater than 20.010 or less than -0.010.

Depending on the relative position of start point P_0 and end point P_4 with reference to the machining plane, the infeed movements are performed in the negative (normal for approach) or positive (normal for retraction) direction, i.e. with `G17` it is possible for the Z component of end point P_4 to be greater than that of start point P_0.

Programming the end point P_4 (or P_0 for retraction) generally with X... Y... Z...

Possible ways of programming the end point $P_4$for approach

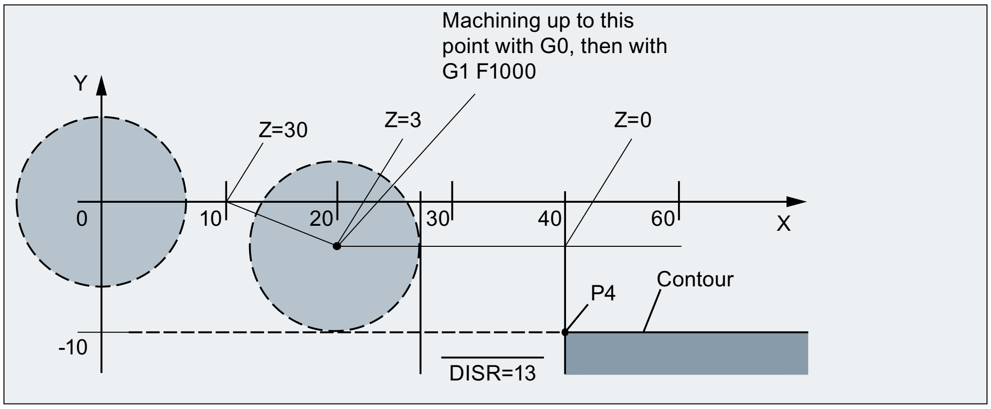
End point P_4 can be programmed in the actual SAR block.

P_4 can be determined by the end point of the next traversing block.

Further blocks (dummy blocks) can be inserted between the SAR block and the next traversing block without moving the geometry axes.

The end point is deemed to have been programmed in the actual SAR block for approach if at least one geometry axis is programmed on the machining plane (X or Y with `G17`). If only the position of the axis perpendicular to the machining plane (Z with `G17`) is programmed in the SAR block, this component is taken from the SAR block, but the position in the plane is taken from the following block. In this case, an alarm is output if the axis perpendicular to the machining plane is also programmed in the following block.

Example:



Program code	Comment
`$TC_DP1[1,1]=120`	`; Milling tool T1/D1`
`$TC_DP6[1,1]=7`	`; Tool with 7mm radius`
`N10 G90 G0 X0 Y0 Z30 D1 T1`	
`N20 X10`	
`N30 G41 G147 DISCL=3 DISR=13 Z=0 F1000`	
`N40 G1 X40 Y-10`	
`N50 G1 X50`	
`...`	
`...`	

`N30`/`N40` can be replaced by:

```
N30 G41 G147 DISCL=3 DISR=13 X40 Y-10 Z0 F1000
```

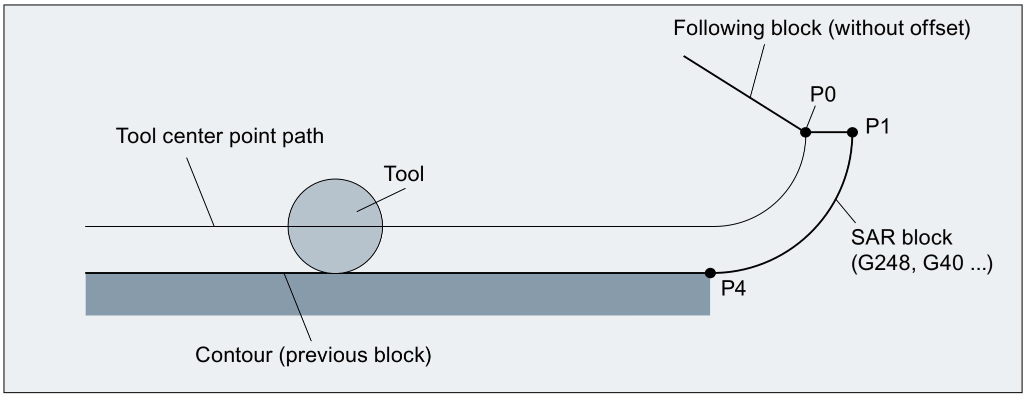
or:

```
N30 G41 G147 DISCL=3 DISR=13 F1000
N40 G1 X40 Y-10 Z0
```

Possible ways of programming the end point $P_0$for retraction

The end position is always taken from the SAR block, no matter how many axes have been programmed. We distinguish between the following situations:

1. No geometry axis is programmed in the SAR block.
   In this case, the contour ends at point P_2 (or at point P_1, if P_1 and P_2 coincide). The position in the axes, which describe the machining plane, is determined by the retraction contour (end point of the straight line or arc). The axis component perpendicular to this is defined by `DISCL`. If, in this case, `DISCL = 0`, the movement takes place completely within the plane.
2. Only the axis perpendicular to the machining plane is programmed in the SAR block.
   In this case, the contour ends at point P_1. The position of the two other axes is determined in the same way as in 1.

   Retraction with SAR with simultaneous deactivation of TRC
   If the SAR retraction block is also used to deactivate tool radius compensation, in the case of 1. and 2., an additional path from P_1 to P_0 is inserted such that no movement is produced when tool radius compensation is deactivated at the end of the retraction contour, i.e. this point defines the tool center point and not a position on a contour to be corrected.
3. At least one axis of the machining plane is programmed.
   The second axis of the machining plane can be determined modally from its last position in the preceding block. The position of the axis perpendicular to the machining plane is generated as described in 1. or 2., depending on whether this axis is programmed or not. The position generated in this way defines the end point P_0.
   No special measures are required for deselection of tool radius compensation, because the programmed point P_0 already directly defines the position of the tool center point at the end of the complete contour.
   The start and end points of the SAR contour (P_0 and P_4) can coincide on approach and retraction.

Velocity of the preceding block (typically G0).

All movements from point P_0 to point P_2 are performed at this velocity, i.e. the movement parallel to the machining plane and the part of the infeed movement up to the safety clearance.

Programming the feedrate with FAD

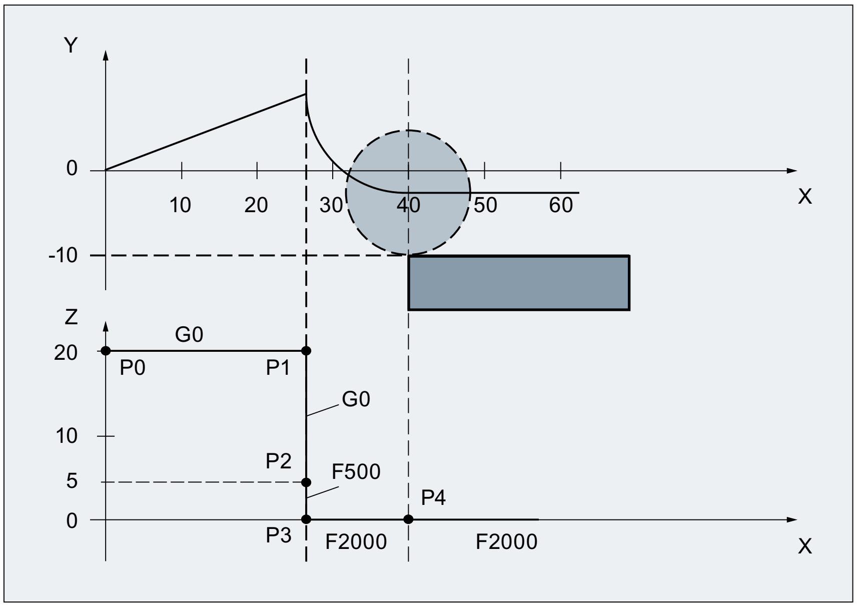
FAD programmed with ...	
G340	Feedrate from P_2 or P_3 to P_4.
G341	Feedrate of the infeed movement perpendicular to the machining plane from P_2 to P_3.

If FAD is not programmed, this part of the contour is traversed at the velocity, which is active modally from the preceding block, in the event that no F command defining the velocity is programmed in the SAR block.

Programmed response:

FAD=0 or negative	→	Alarm Output
FAD=...	→	Programmed value acts in accordance with the active G command of group 15 (feed type; G93, G94, etc.)
FAD=PM(...)	→	Programmed value is interpreted as linear feed (like G94), irrespective of the active G command of group 15
FAD=PR(...)	→	Programmed value is interpreted as revolutional feed (like G95), irrespective of the active G command of group 15

Example:



Program code	Comment
$TC_DP1[1,1]=120	; Milling tool T1/D1
$TC_DP6[1,1]=7	; Tool with 7mm radius
N10 G90 G0 X0 Y0 Z20 D1 T1	
N20 G41 G341 G247 DISCL=AC(5) DISR=13FAD 500 X40 Y-10 Z=0 F2000	
N30 X50	
N40 X60	

Program code	Comment
...	

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.

19.5.4.3 Velocities

Velocities at approach

In both approach diagrams below, it is assumed that no new velocity is programmed in the block following the SAR block. If this is not the case, the new velocity comes into effect after point P_4.

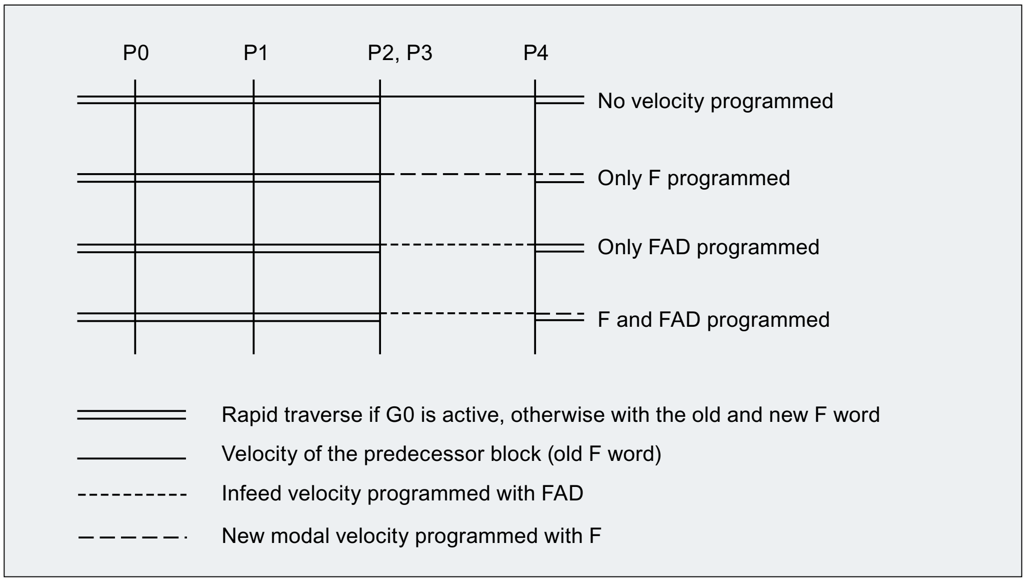

P0
P1
P2, P3
P4
No velocity programmed
Only F programmed
Only FAD programmed
F and FAD programmed
Rapid traverse if G0 is active, otherwise with the old and new F word
Velocity of the predecessor block (old F word)
Infeed velocity programmed with FAD
New modal velocity programmed with F

Figure 19-19 Velocities in the SAR subblocks on approach with `G340`

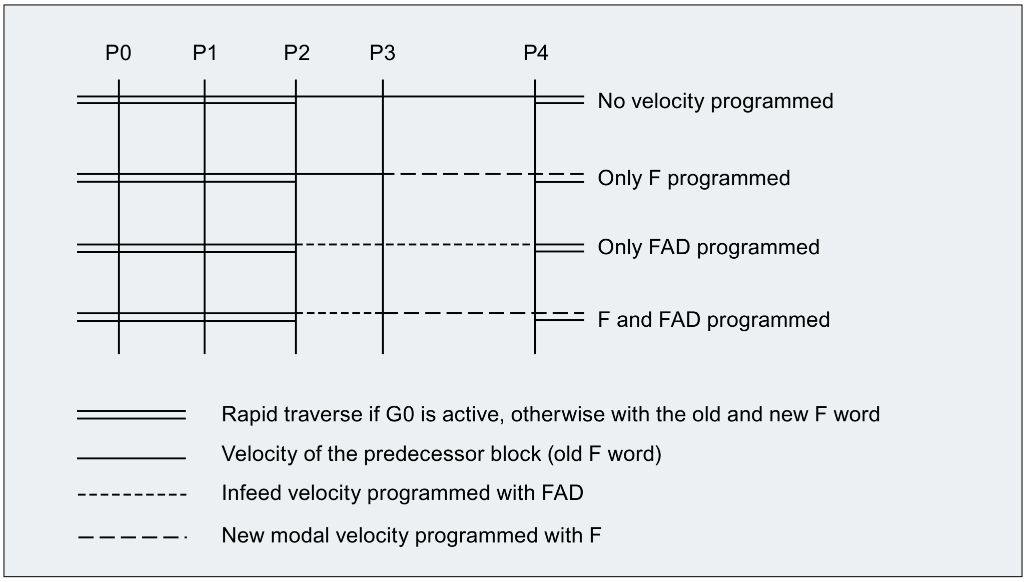

P0
P1
P2
P3
P4
No velocity programmed
Only F programmed
Only FAD programmed
F and FAD programmed
Rapid traverse if G0 is active, otherwise with the old and new F word
Velocity of the predecessor block (old F word)
Infeed velocity programmed with FAD
New modal velocity programmed with F

Figure 19-20 Velocities in the SAR subblocks on approach with `G341`

Velocities at retraction

During retraction, the rolls of the modally active feedrate from the previous block and the programmed feedrate value in the SAR block are interchanged, i.e., the actual retraction contour (straight line, circle, helix) is traversed with the old feedrate value and a new velocity programmed with the F word applies from point P_2 up to P_0.

If even retraction is active and `FAD` is programmed, the path from P_3 to P_2 is traversed with `FAD`, otherwise it is traversed with the old velocity. The last F command programmed in a preceding block always applies for the path from P_4 to P_2. `G0` has no effect in these blocks.

Traversing from P_2 to P_0 takes place with the F command programmed in the SAR block or, if no F command is programmed, with the modal F command from a preceding block. This applies on the condition that `G0` is not active.

If rapid traverse is to be used on retraction in the blocks from P_2 to P_0, `G0` must be activated before the SAR block or in the SAR block itself. If an additional F command is programmed in the actual SAR blocks, it is then ineffective. However, it remains modally active for following blocks.

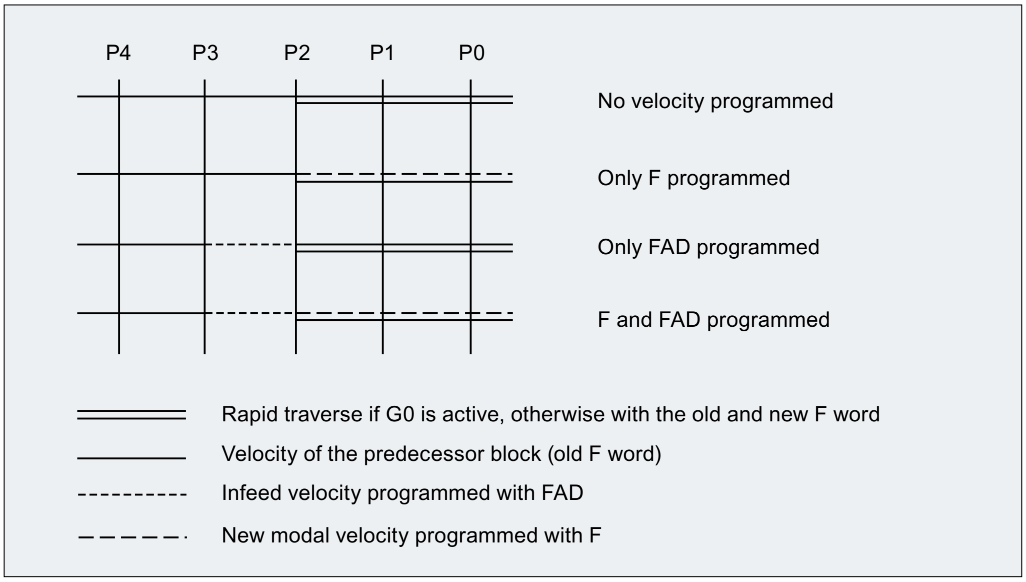

P4
P3
P2
P1
P0
No velocity programmed
Only F programmed
Only FAD programmed
F and FAD programmed
Rapid traverse if G0 is active, otherwise with the old and new F word
Velocity of the predecessor block (old F word)
Infeed velocity programmed with FAD
New modal velocity programmed with F

Figure 19-21 Velocities in the SAR subblocks on retraction

19.5.4.4 System variables

Points P_3 and P_4 can be read in the WCS as system variables during approach.

$P_APR:	Read P_3 (start point) in WCS	
$P_AEP:	Read P_4 (contour start point) in WCS	
$P_APDV	=1	If the content of $P_APR and $P_AEP is valid, i.e., if these contain the position values belonging to the last SAR approach block programmed.
	=0	The positions of an older SAR approach block are read.

Changing the WCS between the SAR block and the read operation has no effect on the position values.

19.5.4.5 Supplementary conditions

Supplementary conditions

- Any further NC commands (e.g. auxiliary function outputs, synchronous axis movements, positioning axis movements, etc.) can be programmed in an SAR block.
  These are executed in the first subblock on approach and in the last subblock on retraction.
- If the end point P_4 is not taken from the SAR block but from a subsequent traversing block, the actual SAR contour (straight line, quadrant or semicircle) is traversed in this block. The last subblock of the original SAR block does not then contain traversing information for geometry axes. It is always output, however, because further actions (e.g. single axes) may have to be executed in this block.

- At least two blocks must always be taken into consideration:
  - The SAR block itself
  - The block, which defines the approach or retraction direction

  Further blocks can be programmed between these two blocks.
  The number of possible dummy blocks is limited with the machine data:
  MD20202 $MC_WAB_MAXNUM_DUMMY_BLOCKS (maximum number of blocks with no traversing motions with SAR).
- If tool radius compensation is activated simultaneously in an approach block the first linear block of the SAR contour is the block in which activation takes place.
  The complete contour generated by the SAR function is treated by tool radius compensation as if it has been programmed explicitly (collision detection, calculation of intersection, approach behavior `NORM`/`KONT`).
- The direction of the infeed motion and the position of the circle plane or the helix axis are defined exclusively by the active plane (`G17` - `G19`) - rotated with an active frame where appropriate.
- On approach, a preprocessor stop must not be inserted between the SAR block and the following block which defines the direction of the tangent.
  Whether programmed explicitly or inserted automatically by the control, a preprocessor stop results, in this case, in an alarm.

Behavior with REPOS

If an SAR cycle is interrupted and repositioned, it resumes at the point of interruption on `RMIBL`. With `RMEBL`, the contact point is the end point of the last SAR block; with `RMBBL`, it is the start point of the first SAR block.

If `RMIBL` is programmed together with `DISPR` (reapproach at distance `DISPR` in front of interruption point), the reapproach point can appear in a subblock of the SAR cycle before the interruption subblock.

19.5.4.6 Examples

Example 1

The following conditions must be true:

- Smooth approach is activated in block `N20`
- X=40 (end point); Y=0; Z=0
- Approach movement performed with quadrant (`G247`)
- Approach direction not programmed, `G140` is valid, i.e. because TRC is active (`G42`) and compensation value is positive (10), the contour is approached from the right
- Approach circle generated internally (SAR contour) has radius 20, so that the radius of the tool center path is equal to the programmed value `DISR=10`

- Because of `G341`, the approach movement takes place with a circle in the plane, resulting in a start point at (20, -20, 0)
- Because `DISCL=5`, point P2 is at position (20, -20, 5) and, because of Z30, point P1 is in `N10` at (20, -20, 30)

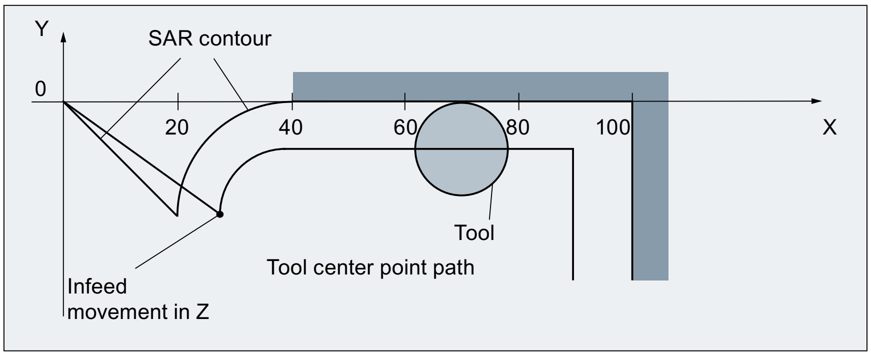


Figure 19-22 Contour example 1

Part program:

Program code	Comment
`$TC_DP1[1,1]=120`	`; Tool definition T1/D1`
`$TC_DP6[1,1]=10`	`; Radius`
`N10 G0 X0 Y0 Z30`	
`N20 G247 G341 G42 NORM D1 T1 Z0 FAD=1000 F=2000 DISCL=5 DISR=10`	
`N30 X40`	
`N40 X100`	
`N50 Y-30`	
`...`	

Example 2

The following conditions must be true for approach:

- Smooth approach is activated in block `N20`
- Approach movement performed with quadrant (`G247`)
- Approach direction not programmed, `G140` is valid, i.e. because TRC is active (`G41`), the contour is approached from the left
- Contour offset `OFFN=5` (`N10`)
- Current tool radius=10, and so the effective compensation radius for TRC=15; the radius of the SAR contour is thus equal to 25, with the result that the radius of the tool center path is equal to `DISR=10`

- The end point of the circle is obtained from `N30`, since only the Z position is programmed in `N20`
- Infeed 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`
    (In order that this velocity becomes effective in the following blocks, the active G-code `G0` in `N30` must be overwritten with `G1`. Otherwise, the contour would continue to be machined with `G0`.)

The following conditions must be true for retraction:

- Smooth retraction is activated in block `N60`
- Retraction movement performed with quadrant (`G248`) and helix (`G340`)
- `FAD` not programmed, since irrelevant for `G340`
- Z=2 in the start point; Z=8 in the end point, since `DISCL=6`
- When `DISR=5`, the radius of SAR contour=20; that of the tool center point path=5
- After the circle block, the retraction movement leads from Z8 to Z20 and the movement is parallel to the XY plane up to the end point at X70 Y0

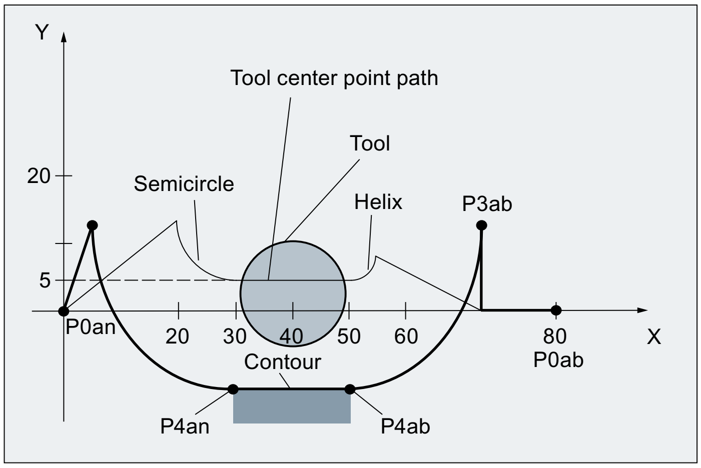


Figure 19-23 Contour example 2

Part program:

Program code	Comment
`$TC_DP1[1,1]=120`	`; Tool definition T1/D1`
`$TC_DP6[1,1]=10`	`; Radius`
`N10 G0 X0 Y0 Z20 G64 D1 T1 OFFN = 5`	`; (P0app)`
`N20 G41 G247 G341 Z0 DISCL = AC(7) DISR = 10 F1500 FAD=200`	`; (P3app)`
`N30 G1 X30 Y-10`	`; (P4app)`
`N40 X40 Z2`	
`N50 X50`	`; (P4ret)`
`N60 G248 G340 X70 Y0 Z20 DISCL = 6 DISR = 5 G40 F10000`	`; (P3ret)`
`N70 X80 Y0`	`; (P0ret)`

Program code	Comment
`N80 M 30`	

Note

The contour generated in this way is modified by tool radius compensation, which is activated in the SAR approach block and deactivated in the SAR retraction block.

The tool radius compensation allows for an effective radius of 15, which is the sum of the tool radius (10) and the contour offset (5). The resulting radius of the tool center path in the approach block is therefore 10, and 5 in the retraction block.

19.5.5 Deselecting the TRC (G40)

G40 instruction

TRC is deselected with the `G40` instruction.

Special points to be noted

- TRC can only be deselected in a program block with `G0` (rapid traverse) or `G1` (linear interpolation).
- If `D0` is programmed when tool radius compensation is active, compensation is not deselected and error message 10750 is output.
- If a geometry axis is programmed in the block with the tool radius compensation deselection, then the compensation is deselected even if it is not on the current plane.

19.5.6 Compensation at outside corners

G450/G451

The G commands `G450`/`G451` can be used to control the response for discontinuous block transitions at outside corners:

Command	Meaning
`G450`	Discontinuous block transitions with transition circle
`G451`	Discontinuous block transitions with intersection of equidistant paths

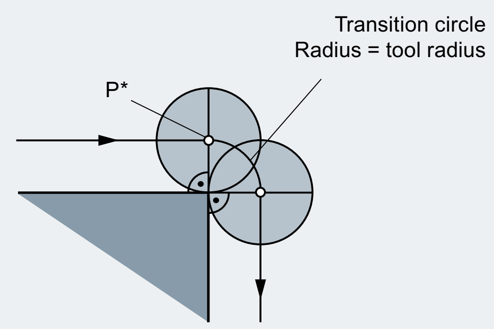
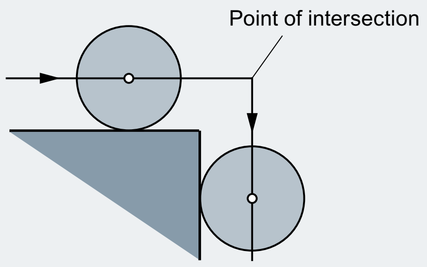
Programming: G450 transition circle	Programming: G451 point of intersection
Outside corner 	Outside corner 

Figure 19-24 Example of a 90 degree outside corner with `G450` and `G451`

G450 (transition circle)

With the active G command `G450`, on outside corners, the center point of the tool travels a circular path along the tool radius. The circular path starts with the normal position (perpendicular to the path tangent) at the end point of the previous path section (program block) and ends in normal position at the start point of the new path section (program block).

Where outside corners are very flat, the response with `G450` (transition circle) and `G451` (intersection) becomes increasingly similar (see "Very flat outside corners").

If pointed outside corners are desired, the tool must be retracted from the contour (see Section "DISC").

DISC

The `G450` transition circle does not produce sharp outside contour corners because the path of the tool center point through the transition circle is controlled so that the cutting edge stops at the outside corner (programmed position). When sharp outside corners are to be machined with `G450`, the `DISC` statement 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` statement is 0 to 100, in increments of 1.

Value	Meaning
`DISC = 0`	Overshoot disabled, transition circle active
`DISC = 100`	Overshoot large enough to theoretically produce a response similar to intersection (`G451`).

The programmable maximum value for `DISC` can be set via the machine data:

MD20220 $MC_CUTCOM_MAX_DISC (max. value for DISC).

Values greater than 50 are generally not advisable with `DISC`.

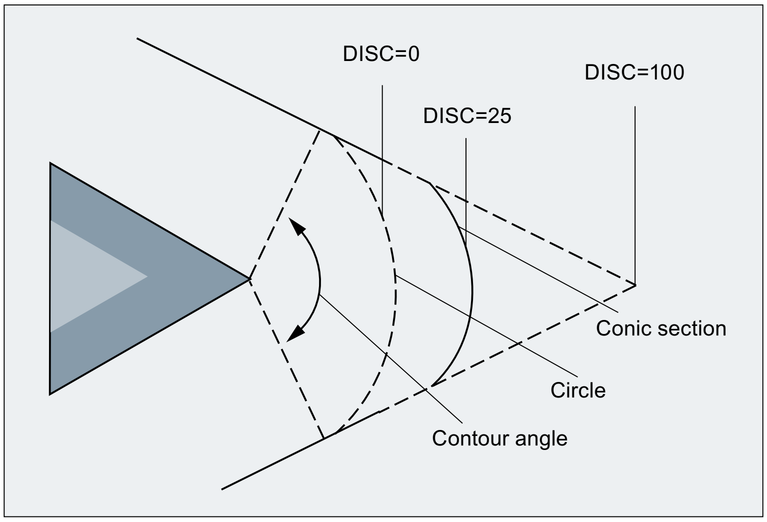


Figure 19-25 Example: Overshoot with `DISC`= 25

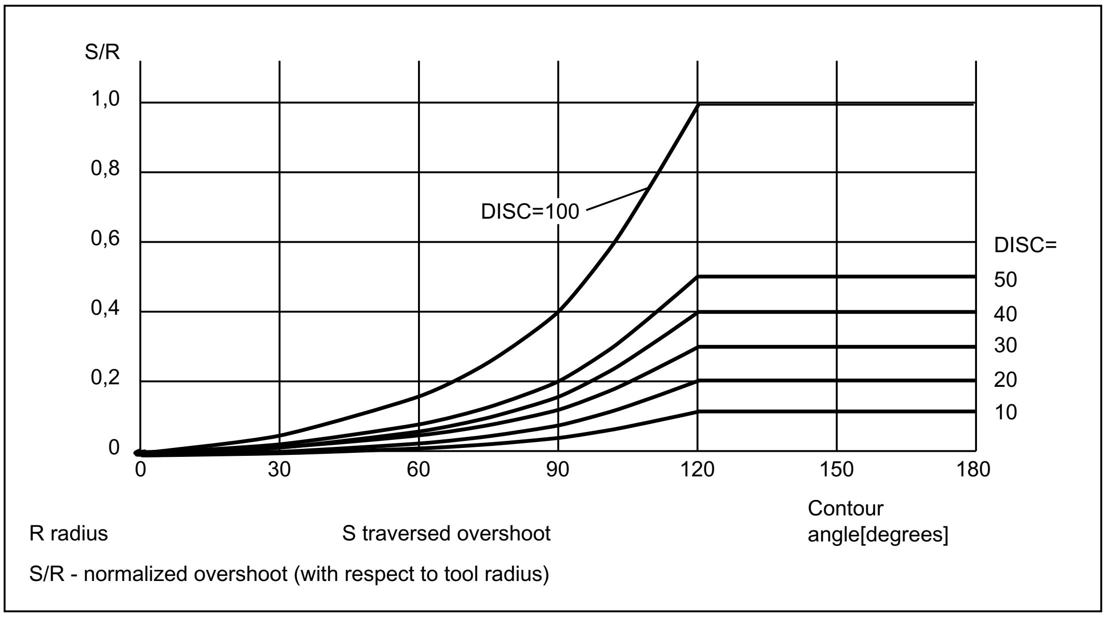


Figure 19-26 Overshoot with `DISC` depending on contour angle

G451 (intersection)

If the `G451` G command 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. Splines and polynomials are never extended.

Very pointed outside corners

Where outside corners are very pointed, `G451` can result in excessive idle paths. Therefore, the system switches automatically from `G451` (intersection) to `G450` (transition circle, with `DISC` where appropriate) when outside corners are very pointed.

The threshold angle (contour angle) for this automatic switchover (intersection point → transition circle) can be specified in the machine data:

MD20210 $MC_CUTCOM_CORNER_LIMIT (Max. angle for compensation blocks with tool radius compensation).

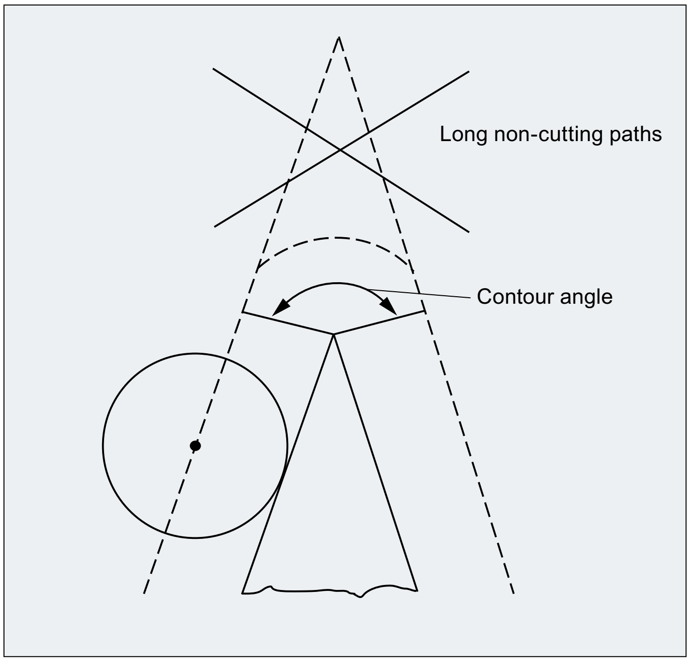


Figure 19-27 Example of automatic switchover to transition circle

Very flat outside corners

Where outside corners are very flat, the response with `G450` (transition circle) and `G451` (intersection) becomes increasingly similar. In this case, it is no longer advisable to insert a transition circle. One reason why it is not permitted to insert a transition circle at these outside corners with 5-axis machining is that this would impose restrictions on speed in contouring mode (`G64`). Therefore, the system switches automatically from `G450` (transition circle, with `DISC` where appropriate) to `G451` (intersection) when outside corners are very flat.

The threshold angle (contour angle) for this automatic switchover (transition circle → intersection point) can be specified in the machine data:

MD20230 $MC_CUTCOM_CURVE_INSERT_LIMIT (Max. angle for intersection calculation with tool radius compensation).

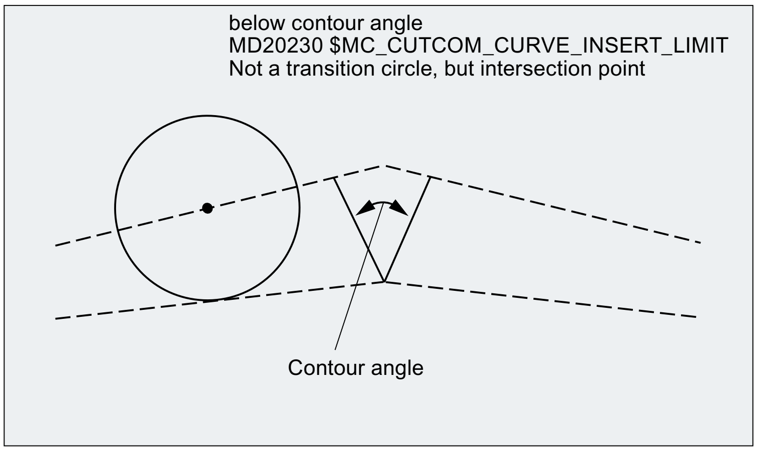
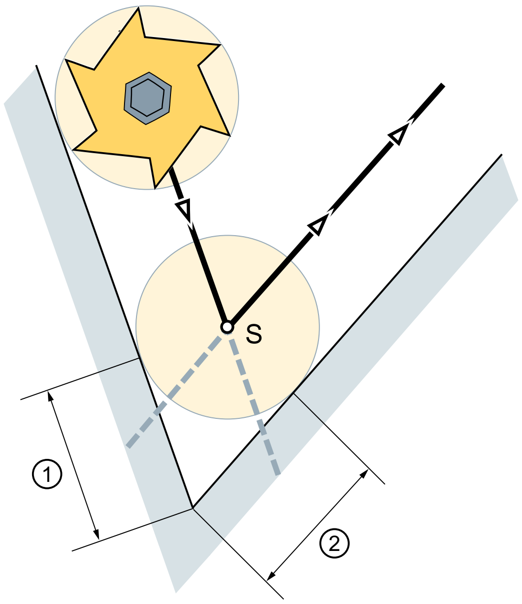

below contour angle
MD20230 $MC_CUTCOM_CURVE_INSERT_LIMIT
Not a transition circle, but intersection point
Contour angle

Figure 19-28 Example of automatic switchover to intersection

19.5.7 Compensation and inner corners

Point of 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:

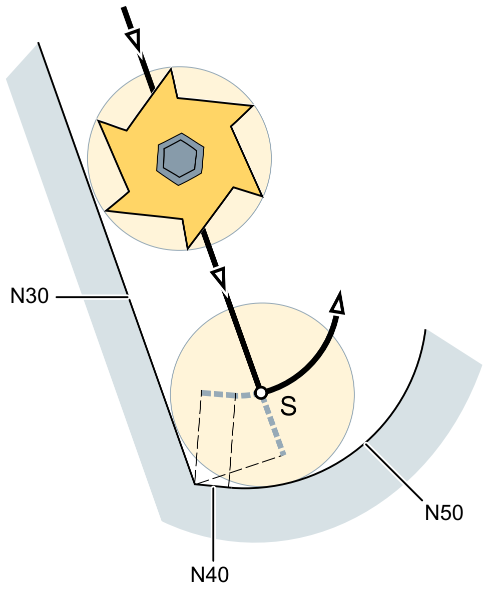


S: Point of intersection
① First block shortened at end.
② Second block shortened at beginning

Figure 19-29 Programmed contour is shortened

No intersection

For inside corners, it is possible that no intersection is found between two consecutive blocks. In this case, the control automatically checks the next block and attempts to find an intersection with the equidistant paths of this block:



S: Point of intersection

Figure 19-30 Predictive contour calculation

This automatic check of the next blocks, predictive contour calculation, is always performed until a number of blocks defined via machine data has been reached.

MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS (blocks for predictive contour calculation for TRC)

If no intersection is found within the number of blocks defined for the looking ahead, program execution is interrupted and an alarm is output.

Multiple intersections

With inside corners it is also possible that predictive contour calculation finds multiple intersections of the equidistant paths in several consecutive blocks. In this case, the last intersection is always used as the valid intersection. The previous intersection points are not approached:

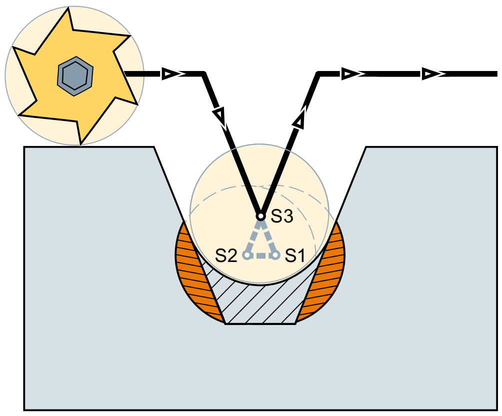


Figure 19-31 In this example, the pocket is machined only as much as is possible without causing a contour violation.

For further information, see also Chapter "Collision monitoring ("bottleneck detection") (Page 1516)".

Special features

Where multiple intersections with the next block are found, the intersection nearest the start of the next block applies.

19.5.8 Collision monitoring ("bottleneck detection")

19.5.8.1 Function

If tool radius compensation is active, collision monitoring ("bottleneck detection") checks by predictive contour calculation whether the equidistant paths of **non-consecutive** blocks intersect. This look-ahead function allows possible collisions to be detected in advance and permits the control to actively avoid them.

The maximum number of blocks that are looked at in advance can be set by channel-specific machine data (see "Parameterization (Page 1517)").

If an intersection is detected, the motions programmed between these blocks on the compensation plane are not executed. Alarm 10763 is displayed:

"The path component of the block in the compensation plane will become zero."

The NC program is not interrupted. The bottleneck is bypassed. All other motions and executable instructions (M commands, traversal of positioning axes, etc.) contained in the omitted blocks are executed at the position of the last intersection found in the sequence in which they are programmed in the NC program.

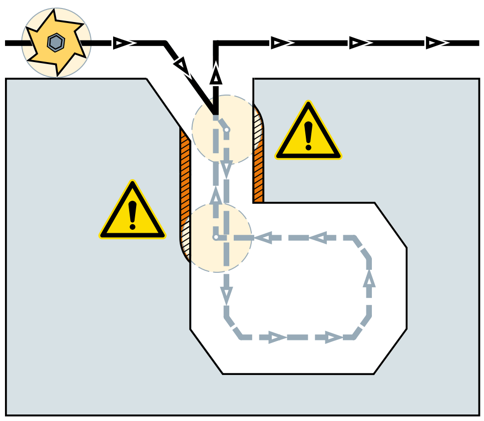


Figure 19-32 Bypassing a bottleneck during active collision monitoring (looking 8 blocks ahead)

Activation / deactivation

The function is activated / deactivated in the NC program with commands of G Group 23.

See "Programming (Page 1517)."

19.5.8.2 Parameterization

Maximum number of blocks for predictive contour calculation

The maximum number of blocks to be predictively checked is set in:

MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS (blocks for predictive contour calculation for TRC)

Suppress alarm 10763

Alarm 10763 (Page 1516) can be suppressed by the following setting:

MD11410 $MN_SUPPRES_ALARM_MASK (mask supporting special alarm outputs), bit 1 = 1

19.5.8.3 Programming

The collision detection ("bottleneck detection") with active TRC is activated or deactivated in the NC program with the commands of G group 23.

Syntax

```
G41/G42 CDON
...
CDOF/CDOF2
```

Meaning

`CDON`:	Activating collision detection ("bottleneck detection") CDON performs a check over an adjustable (MD20240 (Page 1517)) number of blocks as to whether the tool paths of **non-adjacent** blocks intersect. This look-ahead function allows possible collisions to be detected in advance and permits the control to actively avoid them.
`CDOF`:	Deactivating collision detection ("bottleneck detection") With CDOF, a search is made in the **previous** traversing block (at inside corners) for a common intersection for the current block; if necessary the search is extended to even earlier blocks. If an intersection is found, no further blocks are examined. With outside corners, an intersection is always found between two consecutive blocks. **Note:** CDOF can be used to avoid the faulty detection of bottlenecks which may occur due to a lack of information in the NC program, for example.
`CDOF2`:	Deactivating collision detection for 3D circumferential milling The tool offset direction is determined from adjacent block parts with CDOF2. CDOF2 is only effective for 3D circumferential milling and has the same meaning as CDOF for all other types of machining (e.g. 3D face milling).

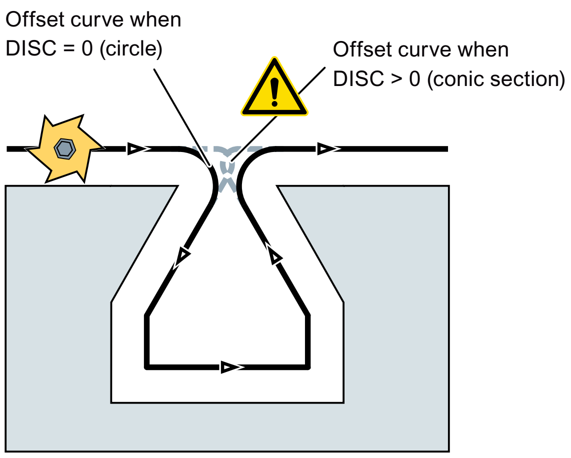
19.5.8.4 Supplementary conditions

Program test

To avoid program stops, the tool with the largest radius from the range of used tools should always be used during the program test.

"Bottleneck detection" due to overhangs at outside corners

When the intersections of non-consecutive blocks are checked, not the programmed original contours 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.



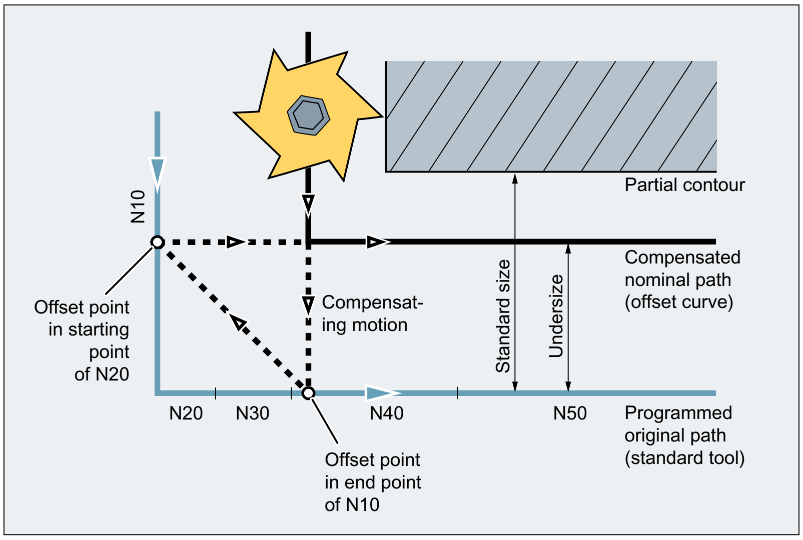
19.5.8.5 Example

Effect of collision detection using an example

The NC program describes the center point path of a standard tool. The contour for a tool that is actually used results in undersize, which is shown unrealistically large to demonstrate the geometric relationships in the following figure.

The control also only has an overview of three blocks in the example:

MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS = 3


Partial contour
N10
Compensated nominal path (offset curve)
Standard size
Undersize
Offset point in starting point of N20
Compensat-ing motion
N20
N30
N40
N50
Programmed original path (standard tool)
Offset point in end point of N10

Since an intersection exists only between the offset curves of the two blocks N10 and N40, the two blocks N20 and N30 would have to be omitted. In the example, the control does not know in block N40 if N10 has to be completely processed. Only a single block can therefore be omitted.

With active CDOF2, the compensation motion shown in the figure is executed and not stopped. In this situation, an active CDOF or CDON would result in an alarm.

19.5.9 Slot shape recognition (option) - 840D sl only

If tool radius correction is active and collision monitoring (Page 1516) is switched on, a programmed slot with a width smaller than the tool diameter is detected as a possible contour violation and it is omitted from machining. In certain technology-dependent application cases (e.g. manufacturing deflection lines with a laser cutting machine), however, it is necessary to machine the slots when the radius of the tool (laser beam) is somewhat larger than the programmed width of the slot would allow. To give the user this option, the tool radius correction has been expanded to include the function "slot shape recognition".

Note

The "slot shape recognition" function is an option for SINUMERIK 840D sl that requires a license.

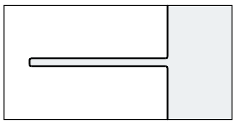
Article number: 6FC5800-0AS18-0YB0

Function

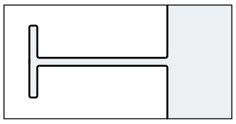
If the "slot shape recognition" function is active, slots that have a predefined shape are automatically identified and their machining is ensured.

The following are detected:

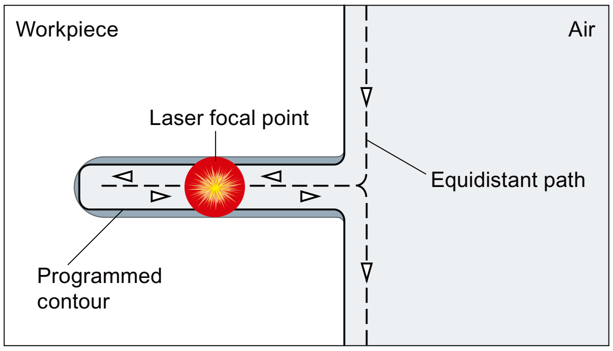
- I-slot



- T-slot



If the slot width is greater than or equal to the tool diameter, the slot will be traversed as programmed. If the slot width is smaller than the tool diameter, the equidistant path is calculated in such a way that the tool traverses along the center of the programmed slot:



The following conditions must be met:

- The radius of a new tool (R2) must not exceed the radius of the original tool (R1) by the factor 2:
  R2 < 2 * R1
- The slot must have an odd number of blocks.
- The central block must comprise a straight line.
- The radius of the rounding at the start of the slot or the length of the block preceding the slot must not be less than the difference R2 - R1.

Activation

To activate the function "slot shape recognition", the following channel-specific setting data must be set to "1":

SD42977 $SC_SLOT_FORM_RECOGN = 1

Parameterization

Number of blocks for predictive feature recognition

The slot shape recognition is only active if the following channel-specific machine data is set to a value greater than zero:

MD28620 $MC_MM_NUM_FEATURE_BLOCKS > 0

With MD28620, the minimum number of blocks is defined that are considered at any one time in order to detect the slot shape.

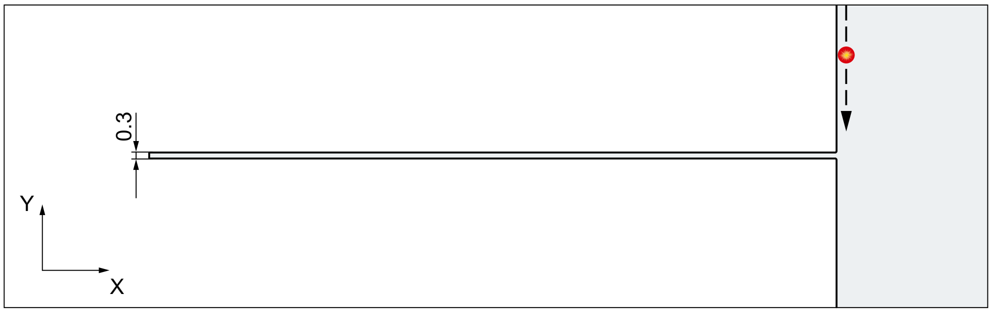
Recommended value: 15 (This corresponds to the number of contour elements for a T-slot with an arc between the straight lines.)

Example

In the following part program section, a deflection line is programmed that will be cut with a laser beam.

Program code	**Comment**
`N1 G01 G90 G71 F10000`	
`N2 $TC_DP1[1,1] = 120`	
`N3 $TC_DP3[1,1] = 0.0`	
`N4 $TC_DP6[1,1] = `**`0.2`**	`; Tool radius = 0.2 mm`
`N6 G91 `**`G41`**` G01 T=1 D=1`	`; Activate TRC.`
`N7 X100.00`	
`N8 Y-24.500`	
`N9 G02 X-0.200 Y-0.200 I-0.200`	`; Rounding at the start of the deflection line contour.`
`N10 G01 X-49.700`	
`N11 G03 X-0.100 Y-0.100 J-0.100`	
`N12 G01 Y-0.100`	

Program code	Comment
N13 G03 X0.100 Y-0.100 I0.100	
N14 G01 X49.700	
N15 G02 X0.200 Y-0.200 J-0.200	; Rounding at the end of the deflection line contour
N16 G01 Y-24.500	



If slot shape recognition is inactive (SD42977 $SC_SLOT_FORM_RECOGN = 0), the tool radius correction will detect and omit the programmed deflection line as an impermissible contour for the tool used (laser beam) because the slot width is 0.3 mm but the tool has radius 0.2 mm. On the other hand, if the slot shape recognition is active (SD42977 $SC_SLOT_FORM_RECOGN = 1), the tool radius correction will detect the programmed deflection line as an I-slot and the laser beam will cut the deflection line along the center of the programmed slot.

Because, in this example, the slot comprises seven contour elements (4 arcs, 3 straight lines), the following minimum value must be set in MD28620 $MC_MM_NUM_FEATURE_BLOCK:

MD28620 $MC_MM_NUM_FEATURE_BLOCK ≥ 7

19.5.10 Blocks with variable compensation value

Supplementary conditions

A variable compensation value is permissible for all types of interpolation (including circular and spine interpolation).

It is also permitted to change the sign (and, therefore, the compensation side).

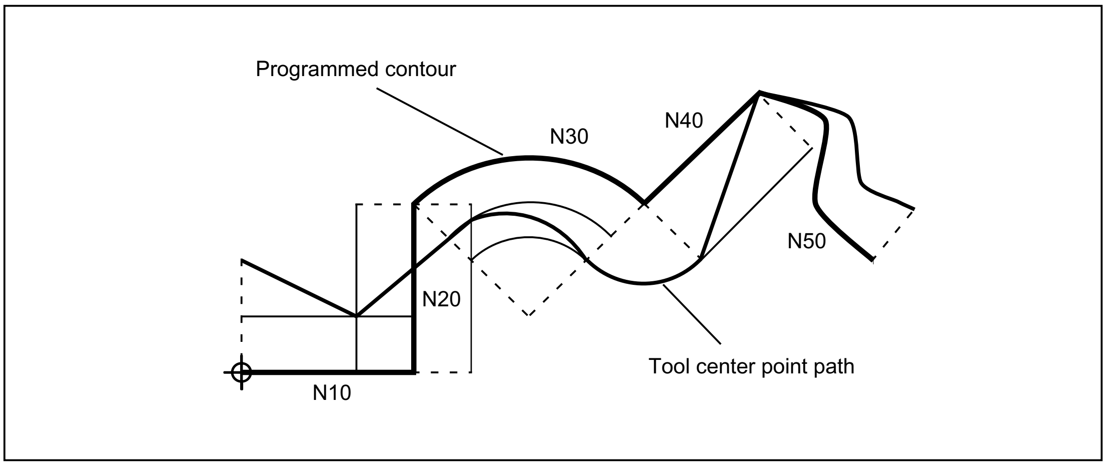


Figure 19-33 Tool radius compensation with variable compensation value

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.

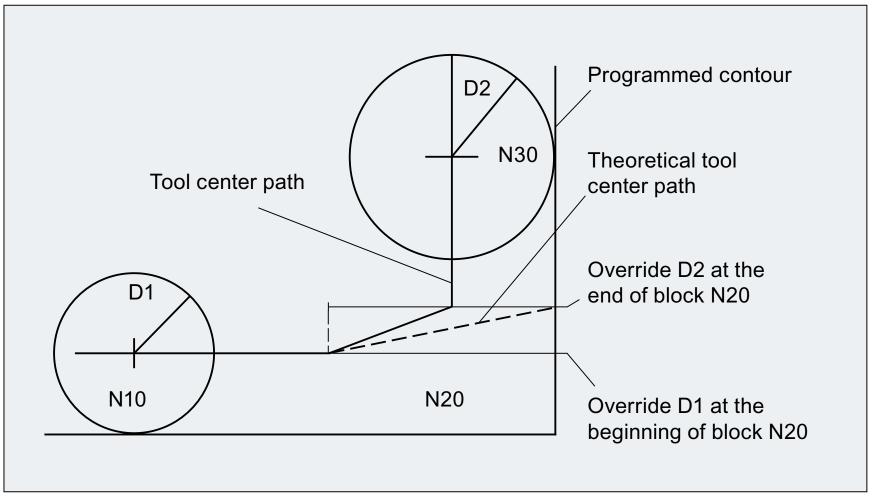

D2
N30
Programmed contour
Theoretical tool center path
Tool center path
Override D2 at the end of block N20
D1
N10
N20
Override D1 at the beginning of block N20

Figure 19-34 Intersection calculation with variable compensation value

Restrictions

If during machining on the inside of the circle the compensation radius becomes geater than the programmed circle radius, then the machining is rejected with the following alarm:

Alarm 10758 "Curvature radius with variable compensation value too small"

Maintain stability of closed contour

If a radius of two circles is increased slightly, a third block may be necessary in order to maintain the stability of the closed contour. This is the case if two adjacent blocks, which represent two possible intersection points for a closed contour, are skipped due to the compensation.

A stable closed contour can be achieved by choosing the first intersection point instead of the second.

SD42496 $SC_CUTCOM_CLSD_CONT ≠ 0 (response of TRC for closed contour).

In that case, the second intersection point is always reached, even if the block is skipped. A third block is then not required.

19.5.11 Alarm behavior

Alarm during preprocessing

If a tool radius compensation alarm is output during preprocessing, main-run machining stops at the next block end reached, i.e. usually at the end of the block currently being interpolated (if Look Ahead is active, once the axes have come to a stop).

Alarms for preprocessing stop and active tool radius compensation

Tool radius compensation generally requires at least one of the following traversing blocks (even more for bottlenecks) to determine the end point of a block. Since the preprocessing stop of such a block is not available, traversing continues to the offset point in the last block. Correspondingly, the offset point in the start point is approached in the first block after a preprocessing stop.

The contour obtained may deviate considerably from the one that would result without preprocessing stop. Contour violations in particular are possible. Therefore the following setting data was introduced:

SD42480 $MC_STOP_CUTCOM_STOPRE (alarm response for TRC and preprocessing stop).

The response of the tool radius compensation remains unchanged compared to the previous status, and/or an alarm is output for preprocessing stop during active tool radius compensation and the program is halted, depending on the value.

The user can acknowledge this alarm and continue the NC program with NC start or abort it with `RESET`.

19.5.12 Intersection procedure for polynomials

Function

If two curves with active tool radius compensation form an outside corner, depending on the G command of the 18th group (corner behavior with tool compensation; `G450`/`G451`) and regardless of the type of curves involved (straight lines, circles, polynomials):

- Either a conic is inserted to bypass the corner
  or
- The curves involved are extrapolated to form an intersection.

If no intersection is found with `G451` activated, or if the angle formed by the two curves is too steep, switchover to insert mode is automatic.

The intersection procedure for polynomials is released with the machine data:

MD20256 $MC_CUTCOM_INTERS_POLY_ENABLE (intersection process possible for polynomials)

Note

If this machine data is set to inactive, a block (can be very short) is always inserted (even if transitions are almost tangential). These short blocks always produce unwanted drops in speed during `G64` operation.

19.5.13 G461/G462 Approach/retract strategy expansion

Function

In certain special geometrical situations, extended approach and retraction strategies, compared with the previous implementation, are required in order to activate or deactivate tool radius compensation (see figure below).

Note

The following example describes only the situation for deactivation of tool radius compensation: The response for approach is virtually identical.

Example

```
G42 D1 T1                          ; Tool radius 20 mm
...
G1 X110 Y0
N10 X0
N20 Y10
N30 G40 X50 Y50
```

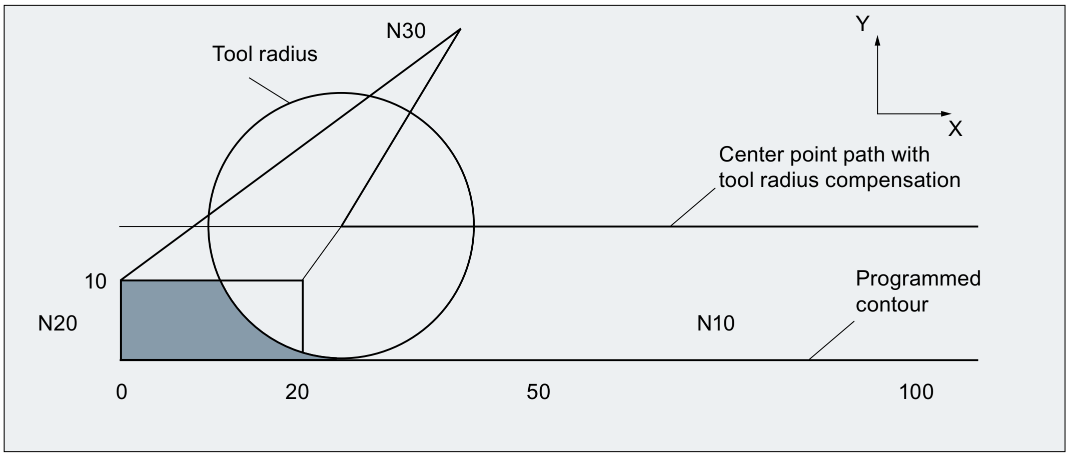


Figure 19-35 Retraction behavior with `G460`

The last block with active tool radius compensation (`N20`) is so short that an intersection no longer exists between the offset curve and the preceding block (or a previous block) for the current tool radius. An intersection between the offset curves of the following and preceding blocks is therefore sought, i.e. between `N10` and `N30` in this example. The curve used for the retraction block is not a real offset curve, but a straight line from the offset point at the end of block `N20` to the programmed end point of `N30`. The intersection is approached if one is found. The colored area in the figure is not machined, although the tool used would be capable of this.

G460

With `G460`, the approach/retraction strategy is the same as before.

G461

If no intersection is possible between the last TRC block and a preceding block, the offset curve of this block is extended with a circle whose center point lies at the end point of the uncorrected block and whose radius is equal to the tool radius.

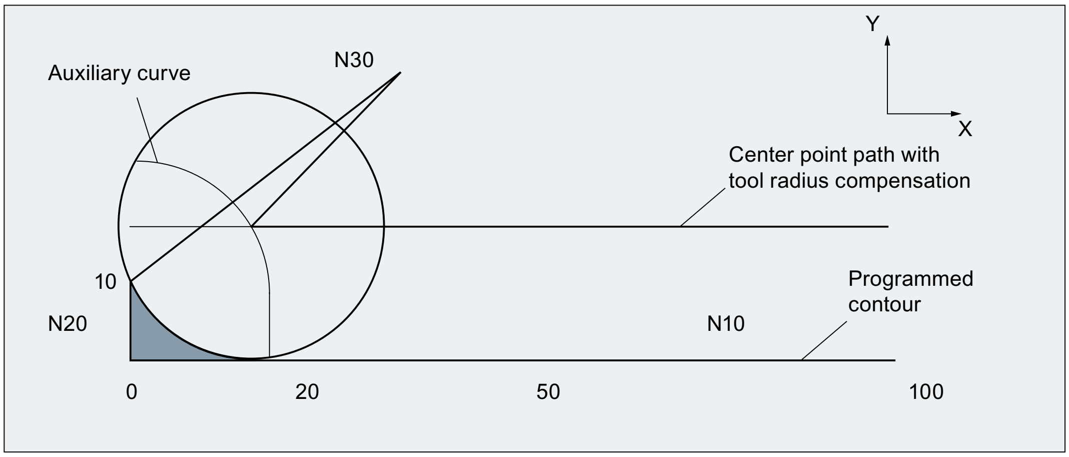


Figure 19-36 Retraction behavior with G461

The control attempts to cut this circle with one of the preceding blocks. If `CDOF` is active, the search is terminated when an intersection is found, i.e. the system does not check for more intersections with even earlier blocks.

If `CDON` is active, the search for more intersections continues after the first intersection is found.

An intersection point, which is found in this way, is the new end point of a preceding block and the start point of the deactivation block. The inserted circle is used exclusively to calculate the intersection and does not produce a traversing movement.

Note

If no intersection is found, the following alarm is output:

Alarm "10751 Collision danger"

G462

If no intersection is possible between the last TRC block and a preceding block, a straight line is inserted, on retraction with `G462` (basic setting), at the end point of the last block with tool radius compensation (the block is extended by its end tangent).

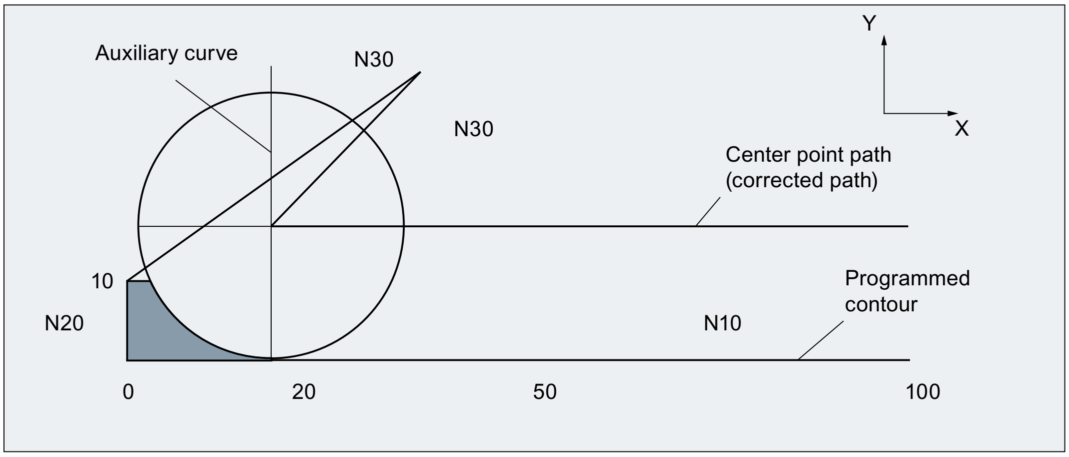


Figure 19-37 Retraction behavior with `G462`

The search for the intersection is then identical to the procedure for `G461`.

With `G462`, the corner generated by `N10` and `N20` in the sample program is not machined to the full extent actually possible with the tool used. However, this behavior may be necessary if the part contour (as distinct from the programmed contour), to the left of `N20` in the example, is not permitted to be violated even with y values greater than 10 mm.

If `KONT` is active (travel round contour at start or end point), behavior will differ according to whether the end point is in front of or behind the contour.

End point in front of contour

If the end point is located in front of the contour, the retraction behavior is the same as for `NORM`. This feature does not change, even if the last contour block with `G451` is extended with a straight line or a circle. Additional circumnavigation strategies to avoid a contour violation in the vicinity of the contour end point are therefore not required.

End point behind contour

If the end point is behind the contour, a circle or straight line is always inserted depending on `G450`/`G451`. In this case, `G460`-`G462` has no effect.

If, in this situation, the last traversing block has no intersection with a preceding block, an intersection with the inserted contour element or with the linear section from the end point of the circumnavigation circle to the programmed end point can result.

If the inserted contour element is a circle (`G450`), and it intersects with the preceding block, this is the same as the intersection, which would be produced with `NORM` and `G461`. In general, however, a remaining section of the circle still has to be traversed. An intersection calculation is no longer required for the linear section of the retraction block.

In the second case (if no intersection is found between the inserted contour element and the preceding blocks), the intersection between the retraction straight line and a preceding block is approached.

Therefore, when `G461` or `G462` is active, behavior deviating from `G460` can only arise if `NORM` is active or if behavior with `KONT` is identical to `NORM` due to the geometrical conditions.

Note

The approach behavior is symmetrical to the retraction behavior.

The approach/retraction behavior is determined by the state of the G command in the approach/ retraction block. The approach behavior can therefore be set independently of the retraction behavior.

Example:

Program for using `G461` during approach:

```
N10 $TC_DP1[1,1]=120                    ; Milling tool type
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
```

19.6 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 traveled tool center path established by tool radius compensation in the previous blocks is retained as the offset.

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 (bypass strategies) are not wanted.

Activation

The "Keep tool radius compensation constant" function is activated with the G command `CUTCONON` (CUTter compensation CONstant ON) and deactivated with the G command `CUTCONOF` (CUTter compensation CONstant OFF).

`CUTCONON` and `CUTCONOF` form a modal G group.

The basic 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	; Activation of the 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 when deactivating the TRC
N140 X0 Y0 G40	
N150 M30	

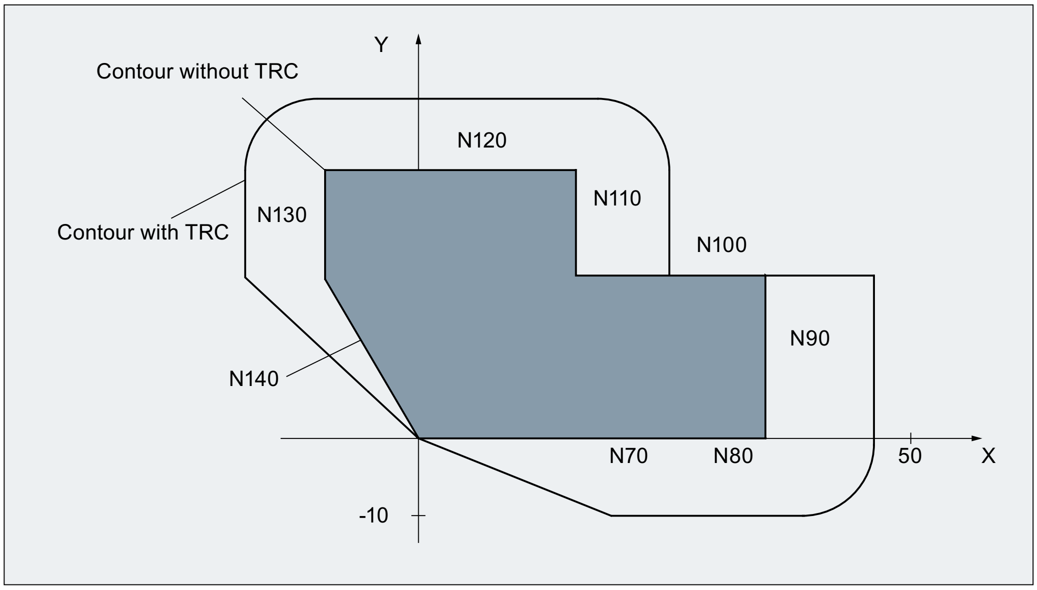


Figure 19-38 Sample program for contour suppression

Special cases

- If tool radius compensation is not active (G40), CUTCONON has no effect. No alarm is produced. The G command 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 command in the 7th G 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 offset.
  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 repositioning the contour with CUTCONOF, the 17th G group (approach and retraction behavior with tool compensation; NORM / KONT) is evaluated, i.e. a bypass circle is inserted if necessary for KONT. A bypass circle is inserted under the same conditions as for activation of tool radius compensation with G41 or G42.
- The number of blocks with suppressed tool radius compensation is restricted:
  MD20252 $MC_CUTCOM_MAXNUM_SUPPR_BLOCKS (Maximum number of blocks with compensation suppression).
  If it is exceeded, machining is aborted and an error message issued.
  The restriction is necessary because the internal block processing in the last block before CUTCONON must be resumed when repositioning.
- The response after reprogramming G41 or G42 when tool radius compensation is already active is similar to compensation suppression.
  The following deviations apply:
  - Only linear blocks are permissible
  - A single traversing block that contains G41 or G42 is modified so that it ends at the offset point of the start point in the following block. Thus it is not necessary to insert an intermediate 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 command 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 monitoring and bottleneck detection is deactivated for all blocks with active contour suppression.

19.7 Toolholder with orientation capability

19.7.1 General information

Introduction

The orientation of the tool can vary (e.g. **due to retooling**) for one single class of machine tools. When the machine is operating, the orientation that has been set is **permanent**, however, and cannot be changed during traversing. For this reason, kinematic orientation transformation (3-, 4- or 5axis transformations, `TRAORI`) is neither necessary nor does it make sense for such machines. However, it is necessary to take account of the changes in the tool length components caused by changing the orientation, without having to trouble the user with mathematics involved. The control performs these calculations.

Availability

For SINUMERIK 828D, the "toolholder with orientation capability" function is only available for the milling versions.

Required data

The following requirements must be met if the control is to take tool compensations into account for toolholders with orientation capability:

- Tool data (geometry, wear, etc.)
- Toolholder data (data for the geometry of the toolholder with orientation capability)

Toolholder selection

A toolholder defined in the control must be specified for the "Toolholder with orientation capability" function. The NC program command below is used for this purpose:

```
TCARR = m
```

m: Number of the toolholder

The toolholder has an associated toolholder data block that 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 number T can be combined freely. In the real application, however, certain combinations can be ruled out for machining or mechanical reasons. The control does not check whether the combinations make sense.

Description of the kinematics of the toolholder

The kinematics of the toolholder with orientation capability is described with a total of 33 parameter sets.

The data of the data block can be edited by the user.

Toolholder with orientation capability

Example: Cardan toolholder with two axes for the tool orientation

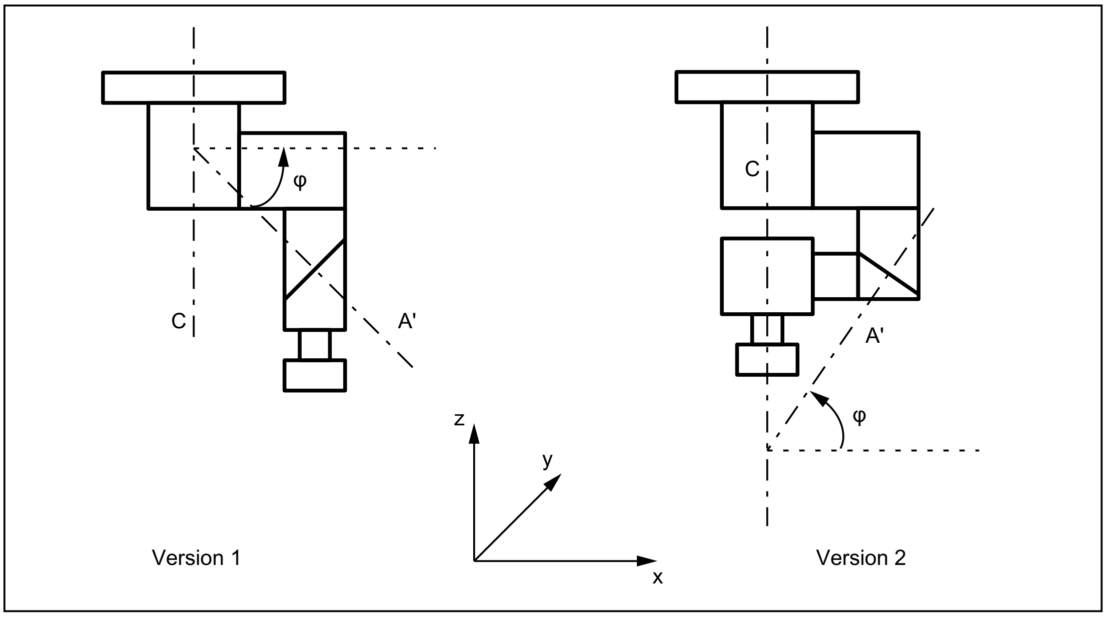


Figure 19-39 Cardan toolholder with two axes

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 command `TOFRAME` defines a frame such that the Z direction in this frame is the same as the current tool orientation.

If no toolholder or a toolholder without change in orientation is active, then the Z direction is in the new frame:

- The same as the old Z direction with `G17`.
- The same as the old Y direction with `G18`.
- The same as the old X direction with `G19`.

TCOABS for active frame

The absolute toolholder orientation is set using:

`TCOABS` (Tool Carrier Orientation ABSolute)

The orientation taken into account for the tool length compensation is **independent** of the orientation of the active frame.

Only one of the commands `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 is 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 is reactivated.

Tool compensation types

TRC (tool radius compensation) 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 group 22, with the exception of `CUT3DC` and `CUT3DF`. The response remains the same with respect to the plane used for compensation. This is determined independent of the tool orientation from the active frame.

For `CUT2DF` and `CUT3DFF`, the compensation plane used for TRC is determined from the frame **independently** of the current tool orientation. The active plane (`G17`/`G18`/`G19`) is considered.

CUT3DC and CUT3DF

3D tool compensation for circumferential milling

3D tool compensations for face milling with active 5-axis transformation are not affected by the "Toolholder with orientation capability" function.

The orientation information is determined by the active kinematic 5-axis transformation.

Limited toolholder orientation

An alarm is output if an orientation that cannot be reached with the defined toolholder kinematic is specified by the frame.

The following kinematics cannot achieve any orientation:

- If the two rotary axes which are necessary to define the kinematics are not perpendicular to each other and if the tool axis which defines the tool direction is not perpendicular to the second rotary axis
  or
- Less 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 parameter set for a toolholder with orientation capability consists of 33 values.

The following system variables are available:

- $TC_CARR1 to $TC_CARR33
- In addition, $TC_CARR34 to $TC_CARR65 are freely available for the user and for fine offsets.

The significance of the individual parameters is distinguished as follows:

Machine kinematics:

$TC_CARR1 to $TC_CARR20 and $TC_CARR23

$TC_CARR18 to $TC_CARR20 define a further vector $l_{4,}$ which is needed to describe the machine with extended kinematics (both tool and workpiece can be rotated).

$TC_CARR21 and $TC_CARR22 contain the channel-axis names 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 an alarm if it is tried to activate the toolholder with orientation capability:

Alarm "14153 Channel %1 block %2 unknown toolholder type: %3"

Rotary axis parameters:

$TC_CARR24 to $TC_CARR33

The system variables in $TC_CARR24 to $TC_CARR33 can be used to define offsets, angle compensations, Hirth tooth system and axis limits.

Note

The system variables are available with and without active tool management.

Components and presetting of the chain/data block

The values $TC_CARR1 to $TC_CARR20 and $TC_CARR24 to $TC_CARR33 in the toolholder data block are of NC language format type `REAL`..

The values $TC_CARR21 and $TC_CARR22 for the axis name of the first rotary axis (v_1) and the second rotary axis (v_2) are of NC language format type AXIS. They are all preset to zero.

The value $TC_CARR23 is initialized with the uppercase letter "T" (only tool can be rotated).

$TC_CARRn[m]

$TC_CARR[0]= 0 has a special significance

System variables for toolholders with orientation capability

$TC_CARRn[m]

n: Parameters 1...33

m: Number of the toolholder 1 that can be oriented...Value of the machine data:
MD18088 $MN_MM_NUM_TOOL_CARRIER (maximum number of definable toolholders)

Description	NC variable	Language format	Default setting
x component of offset vector l_1	$TC_CARR1	REAL	0
y component of offset vector l_1	$TC_CARR2	REAL	0
z component of offset vector l_1	$TC_CARR3	REAL	0
x component of offset vector l_2	$TC_CARR4	REAL	0
y component of offset vector l_2	$TC_CARR5	REAL	0
z component of offset vector l_2	$TC_CARR6	REAL	0
x component of rotary axis v_1	$TC_CARR7	REAL	0
y component of rotary axis v_1	$TC_CARR8	REAL	0
z component of rotary axis v_1	$TC_CARR9	REAL	0
x component of rotary axis v_2	$TC_CARR10	REAL	0
y component of rotary axis v_2	$TC_CARR11	REAL	0
z component of rotary axis v_2	$TC_CARR12	REAL	0
Angle of rotation α1 (in degrees)	$TC_CARR13	REAL	0
Angle of rotation α_2 (in degrees)	$TC_CARR14	REAL	0
x component of offset vector l_3	$TC_CARR15	REAL	0
y component of offset vector l_3	$TC_CARR16	REAL	0
z component of offset vector l_3	$TC_CARR17	REAL	0
x component of offset vector l_4	$TC_CARR18	REAL	0
y component of offset vector l_4	$TC_CARR19	REAL	0
z component of offset vector l_4	$TC_CARR20	REAL	0
Axis name of the rotary axis v_1	$TC_CARR21	AXIS	0
Axis name 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	NC variable	Language format	Default setting
Toolholder name	$TC_CARR34	String[32]	""
Axis name 1	$TC_CARR35 *)	String[32]	""
Axis name 2	$TC_CARR36 *)	String[32]	""
Identifier	$TC_CARR37 *)	INT	0
Position component X	$TC_CARR38 *)	REAL	0
Position component Y	$TC_CARR39 *)	REAL	0
Position component Z	$TC_CARR40 *)	REAL	0
x comp. fine offset of offset vector l_1	$TC_CARR41	REAL	0
y comp. fine offset of offset vector l_1	$TC_CARR42	REAL	0
z comp. fine offset of offset vector l_1	$TC_CARR43	REAL	0
x comp. fine offset of offset vector l_2	$TC_CARR44	REAL	0
y comp. fine offset of offset vector l_2	$TC_CARR45	REAL	0
z comp. fine offset of offset vector l_2	$TC_CARR46	REAL	0
x comp. fine offset of offset vector l_3	$TC_CARR55	REAL	0
y comp. fine offset of offset vector l_3	$TC_CARR56	REAL	0
z comp. fine offset of offset vector l_3	$TC_CARR57	REAL	0
x comp. fine offset of offset vector l_4	$TC_CARR58	REAL	0
y comp. fine offset of offset vector l_4	$TC_CARR59	REAL	0
z comp. fine offset of offset vector l_4	$TC_CARR60	REAL	0
Offset of fine offset of rotary axis v_1	$TC_CARR64	REAL	0
Offset of fine offset of rotary axis v_2	$TC_CARR65	REAL	0
Remarks:			
*)	The system variables $TC_CARR35 to $TC_CARR40 are used in the measuring cycles as well as ShopMill and ShopTurn.		

19.7.2 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:

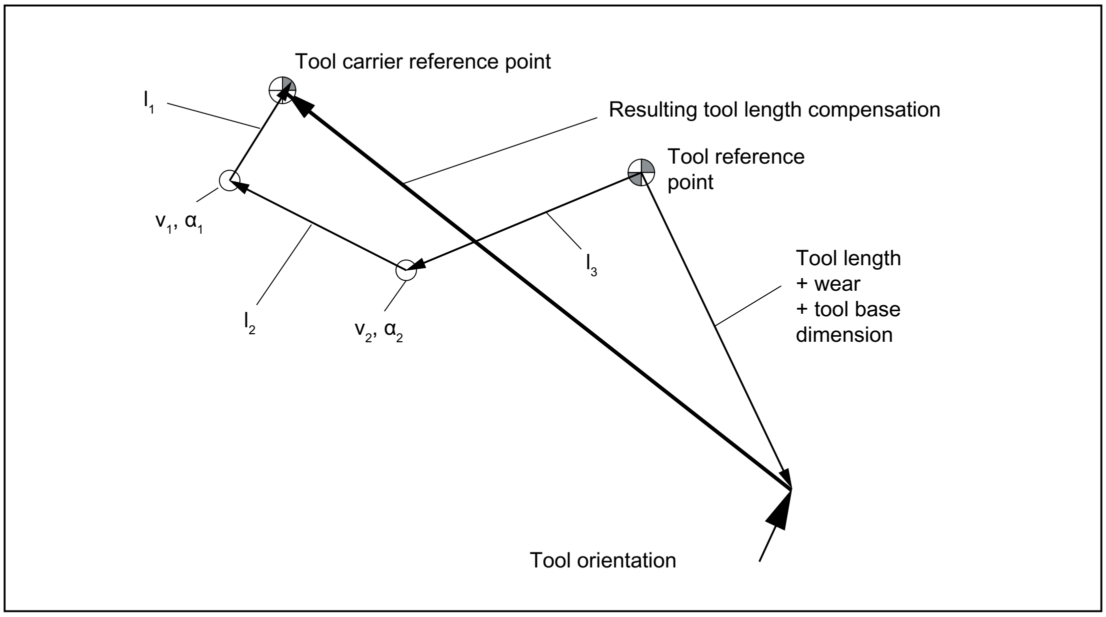

Tool carrier reference point
l_1
Resulting tool length compensation
Tool reference point
v_1, α_1
l_3
Tool length + wear + tool base dimension
l_2
v_2, α_2
Tool orientation

Figure 19-40 Kinematic chain to describe a tool with orientation

Vectors, which describe offsets in the rotary head, are positive in the direction from the tool tip to the reference point of the toolholder.

The following kinematic type is defined for machines with a rotary tool:

$TC_CARR23 using letter T

Machine with rotary workpiece

On machines with rotary workpiece, the vector l_1 has no significance. If it contains a value other than zero, this is ignored.

The kinematic chain for machines with rotary workpiece is shown in the figure below.

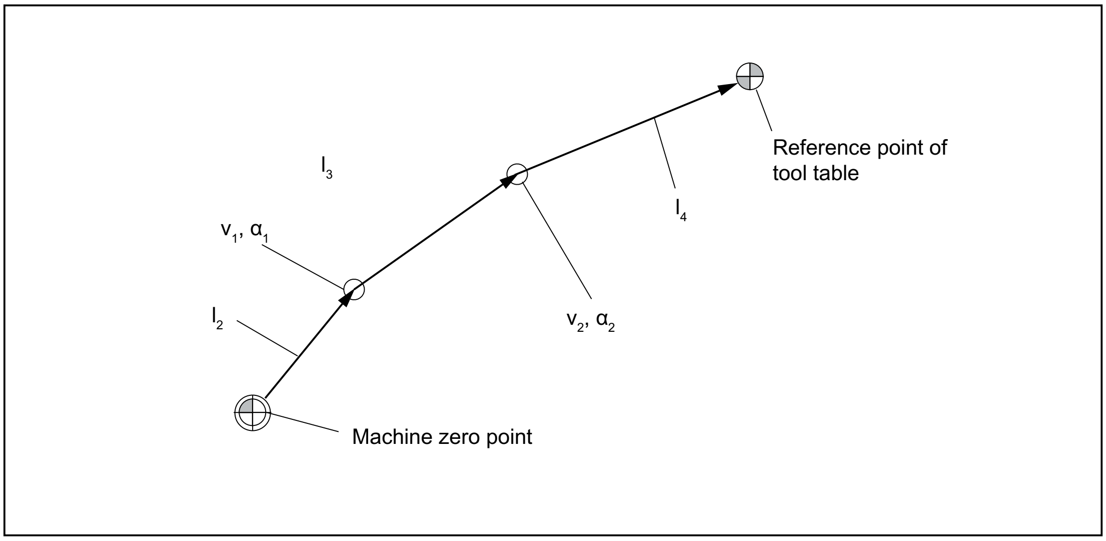


Figure 19-41 Kinematic chain to describe a rotary table

Vectors, which describe offsets in the rotary table, are positive in the direction from the machine reference point to the table.

The following kinematic type is defined for machines with a rotary workpiece:

$TC_CARR23 using letter P

Note

On machines with rotary workpiece it is generally useful if the selected machine reference point and the reference point of the table are identical. Selecting the reference points in this way has the advantage that the position of the workpiece zero in the initial state (i.e. with rotary axes not turned) does not change when the rotary table is activated. The (open) kinematic chain (see figure) is then closed.

In this special case, therefore, the following formula applies: $l_2 = -(l_3 + l_4)$

Machines with extended kinematics

On machines with extended kinematics (both tool and workpiece are rotary), it is only possible to turn each of the components with one axis.

The kinematic of the rotary tool is described with the first rotary axis (v_1) and the two vectors l_1 and l_2, that of the rotary table with the second rotary axis (v_2) and the two vectors l_3 and l_4. The two kinematic chain components for machines with rotary tool and rotary workpiece are shown in the figure below.

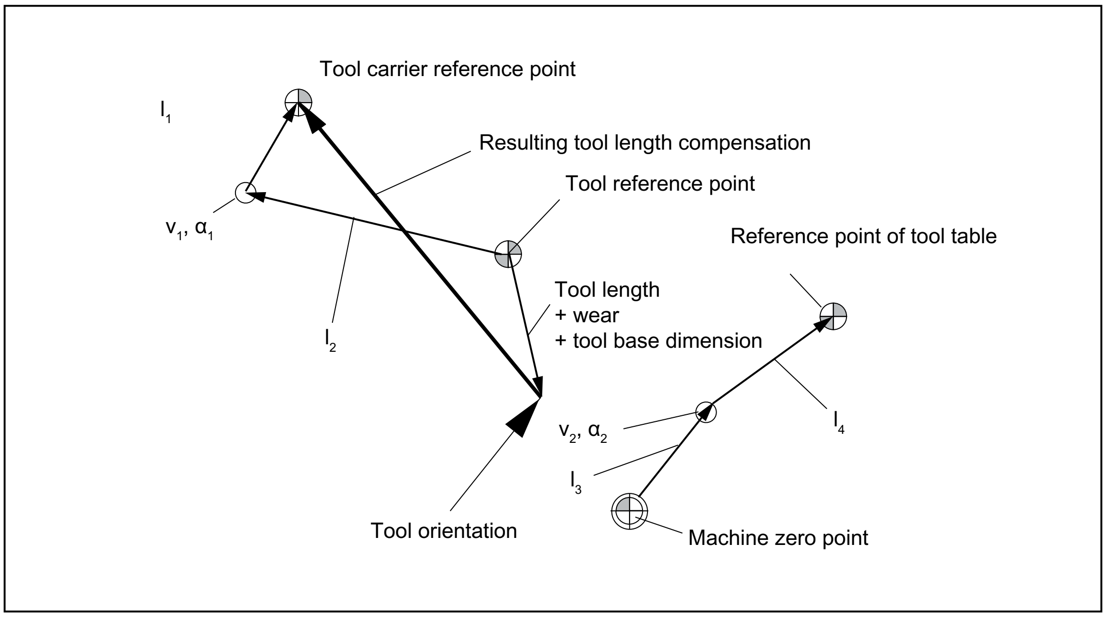


Figure 19-42 Kinematic sequence with extended kinematics

The following kinematic type is defined for machines with a rotary tool and rotary workpiece:

$TC_CARR23 using letter M (extended kinematics)

Note

On machines with extended kinematics it is generally useful, as with machines where only the table can be rotated, for the machine reference point and the reference point of the table to be identical. The (open) chain component to describe the table (see figure) is then closed.

In this special case, the following formula applies: $l_3 = - l_4$

Rotary tool types T and M

For machine kinematics with a rotary tool (types T and M), the toolholder component with orientation capability, which describes the tool or head component (as opposed to the table component), acts, in conjunction with the active tool, as a new overall tool.

Fine offset

The offset vectors l_1 to l_4 and the offsets of the rotary axes v_1 and v_2 can be represented as the sum of a basic value and a fine offset. The fine offset parameters assigned to the basic values are achieved by **adding a value of 40 to the index of the basic value.**

Example:

The parameter $TC_CARR5 is assigned to the fine offset $TC_CARR45.

Note

For the significance of the system variables $TC_CARR41 to $TC_CARR65 available for the fine offset see:

References:
Programming Manual, Job Planning; Tool Offsets:

Activation

The following setting adds the fine offset values to the basic values:

SD42974 $SC_TOCARR_FINE_CORRECTION = 1 (fine offset TCARR on/off)

Supplementary conditions

The amount is limited to the permissible fine offset.

The maximum permissible value is defined:

For:	With machine data:
• The components of vectors l_1 to l_4:	MD20188 $MC_TOCARR_FINE_LIM_LIN
• The offsets of the two rotary axes v_1 and v_2:	MD20190 $MC_TOCARR_FINE_LIM_ROT

An illegal fine offset value is only detected when:

- A toolholder with orientation capability, which contains such a value, is activated
  and
- at the same time the following setting data is set:
  SD42974 $SC_TOCARR_FINE_CORRECTION

Description of a rotation

A data block for describing a rotation comprises one vector v_1 /v_2 to describe the direction of rotation of the rotary axis in its initial state and an angle α1/α2. The angle of rotation is counted positively for clockwise rotation facing the direction of the rotation vector.

The two toolholder angles α1 and α2 are determined using a frame, independent of the active plane currently selected (`G17` - `G19`).

The tool orientation in the initial state (both angles α1 and α2 are zero) is (as in the default case):

- `G17`: Parallel to Z.
- `G18`: Parallel to Y.
- `G19`: Parallel to Z

Assigning data to the toolholder

Example of a machine with rotary toolholder

The following settings are obtained at the mill head shown for a machine with toolholder with orientation capability of kinematic type T:

Component of the offset vector l_1 =	(-200, 0, 0)
Component of the offset vector l_2 =	(0, 0, 0)
Component of the offset vector l_3 =	(-100, 0, 0)
Component of rotary axis v_1 =	(1, 0, 0)
Component of rotary axis v_2 =	(-1, 0, 1)
Tool base dimension of tool reference point	(0, 0, 250)

Note

The tool reference point for the tool base dimension is defined by the reference point at the machine.

For further information about the reference points in the working area, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 705)".

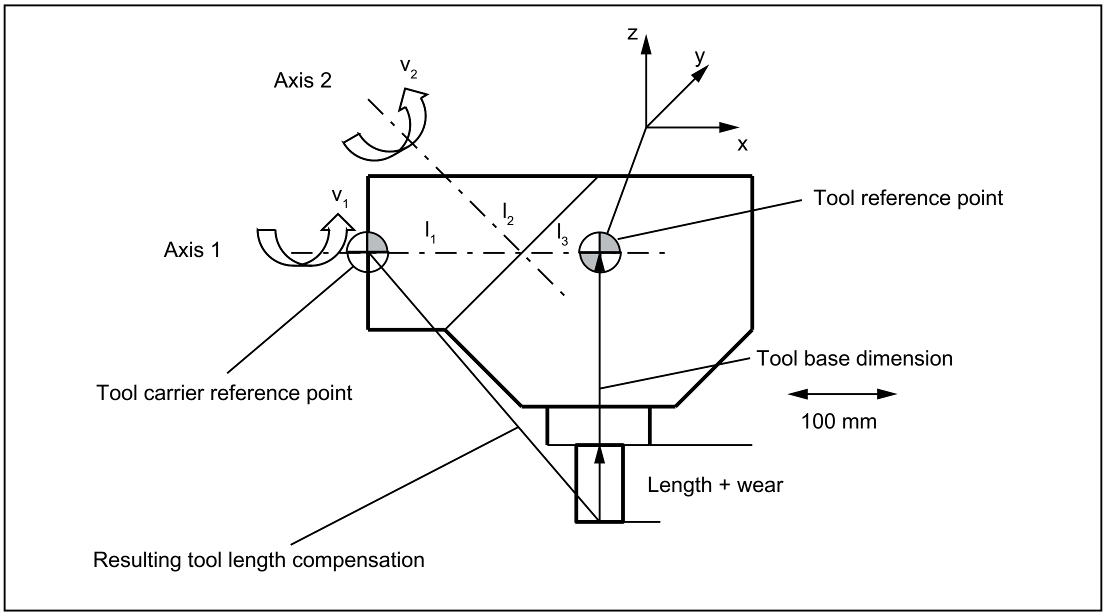


Figure 19-43 Assignment of the toolholder data

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	NC variable	Value
x component of offset vector l_1	$TC_CARR1	-200
y component of offset vector l_1	$TC_CARR2	0
z component of offset vector l_1	$TC_CARR3	0
x component of offset vector l_2	$TC_CARR4	0
y component of offset vector l_2	$TC_CARR5	0
z component of offset vector l_2	$TC_CARR6	0
x component of rotary axis v_1	$TC_CARR7	1
y component of rotary axis v_1	$TC_CARR8	0
z component of rotary axis v_1	$TC_CARR9	0
x component of rotary axis v_2	$TC_CARR10	-1
y component of rotary axis v_2	$TC_CARR11	0
z component of rotary axis v_2	$TC_CARR12	1
Angle of rotation α1 (in degrees)	$TC_CARR13	0
Angle of rotation α2 (in degrees)	$TC_CARR14	0
x component of offset vector l_3	$TC_CARR15	-100
y component of offset vector l_3	$TC_CARR16	0
z component of offset vector l_3	$TC_CARR17	0

Explanations

The toolholder kinematic chosen in the example is such that the two rotary axes form an angle of 45 degrees, which means that the orientation cannot take just any value. In concrete terms, this example does not permit the display of orientations with negative X components.

x component of the tool base dimension:	0
y component of the tool base dimension:	0
z component of the tool base dimension:	250

Note

The required data cannot be determined unequivocally from the geometry of the toolholder, i.e. the user is free to a certain extent to decide the data to be stored. Thus, for the example, it is possible to specify only one z component for the tool base dimension up to the second axis. In this case, l_2 would no longer be zero, but would contain the components of the distance between this point on the second axis and a further point on the first axis. The point on the first axis can also be selected freely. Depending on the point selected, l_1 must be selected such that the reference point (which can also be selected freely) is reached.

In general: Vector components that are not changed by rotation of an axis can be distributed over any vectors "before" and "after" rotation.

19.7.3 Tool carrier with kinematic chains

Modeling tool carriers with kinematic chains

On machines whose structures are defined with kinematic chains, machine parts that act as tool carriers can also be described using kinematic chains. Identical geometry data for kinematic chains and tool carriers can be managed jointly.

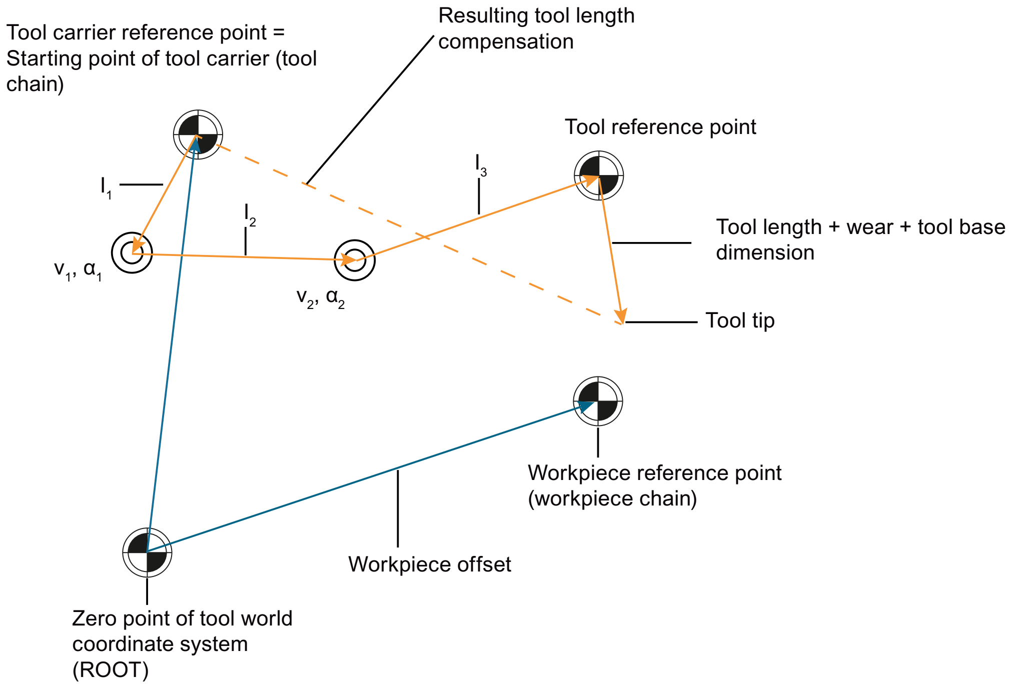
Two kinematic chains are required to define the machine kinematics.

- One chain points from machine zero (zero point of world coordination system) to the workpiece reference point (workpiece part).
- The other chain points from machine zero to the tool carrier reference point (tool part).
- You will find information about kinematic chains in the Function Manual "Special functions" under "K7: Kinematic chain".

Three different kinematic types are available for tool carriers, which are parametrized via $TC_CARR23.

The following graphics show an example of the kinematics of a machine with rotatable tool (type T) and rotatable workpiece (type P) and the resolved kinematics (type M). The resolved kinematics (type M) is a combination of both types.

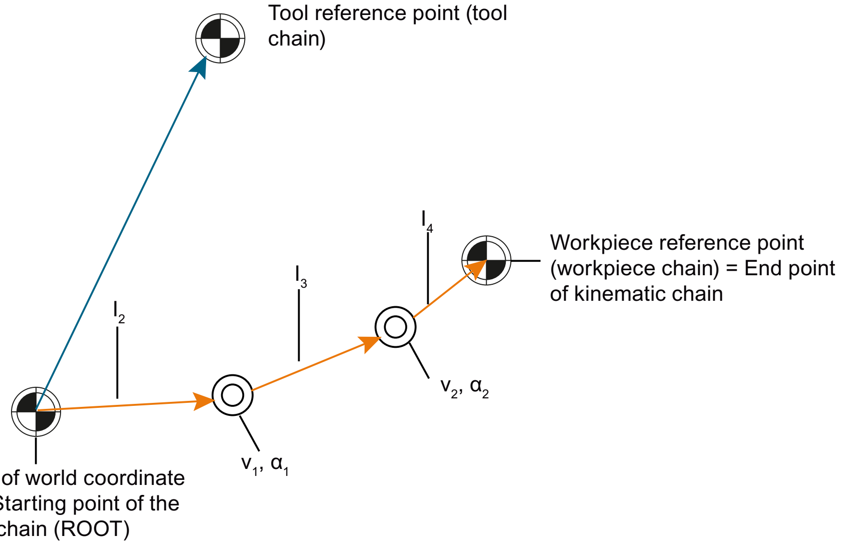
Example for a rotatable tool


Tool carrier reference point = Starting point of tool carrier (tool chain)
Resulting tool length compensation
Tool reference point
l_1
l_2
l_3
Tool length + wear + tool base dimension
v_1, α_1
v_2, α_2
Tool tip
Workpiece reference point (workpiece chain)
Workpiece offset
Zero point of tool world coordinate system (ROOT)

Orange:	Kinematic chain of the tool carrier
Blue:	Kinematic chain of the machine

Figure 19-44 Tool carrier with rotatable tool

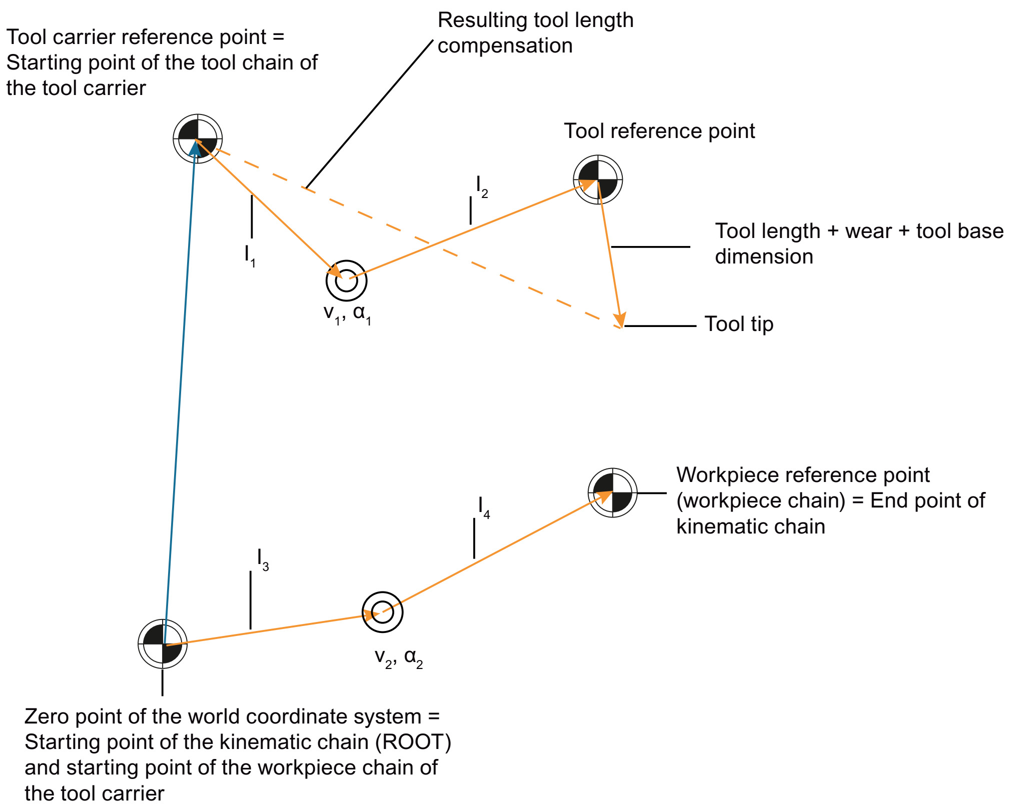
Example of a rotatable workpiece


Tool reference point (tool chain)
l_4
Workpiece reference point (workpiece chain) = End point of kinematic chain
l_3
l_2
v_2, α_2
v_1, α_1
Zero point of world coordinate system = Starting point of the kinematic chain (ROOT)

Orange:	Kinematic chain of the tool carrier
Blue:	Kinematic chain of the machine

Figure 19-45 Tool carrier with rotatable workpiece

Example of mixed kinematics


Tool carrier reference point = Starting point of the tool chain of the tool carrier
Resulting tool length compensation
Tool reference point
l_2
l_1
Tool length + wear + tool base dimension
v_1, α_1
Tool tip
Workpiece reference point (workpiece chain) = End point of kinematic chain
l_4
l_3
v_2, α_2
Zero point of the world coordinate system = Starting point of the kinematic chain (ROOT) and starting point of the workpiece chain of the tool carrier

l_1 and l_2	Offset of the tool chain of the tool holder
v_1	Rotary axis of the tool chain
l_3 and l_4	Offset of the workpiece chain of the tool holder
v_2	Rotary axis of the tool chain

Figure 19-46 Mixed kinematics

The kinematics of the tool carrier is defined together with the machine kinematics.

Definition of the tool carrier:

- Definition of the characteristics of tool carrier with $TC_CARR_KIN_CNTRL:
  - Close tool carrier via kinematic chain, tool chain, and table chain of the tool carrier.
- The starting point of the kinematic chain of the tool carrier is set via system variables $TC_CARR_KIN_TOOL_START and $TC_CARR_KIN_PART_START (chain element name).
- Definition of the rotary axes of the tool carrier with $TC_CARR21 (Page 1532) and $TC_CARR22
- Definition of the tool carrier type with $TC_CARR23 (T, P, or M)

Definition of the kinematics of the tool carrier:

The workpiece part of a kinematic chain for a tool carrier can be structured as follows (example):

Define the linear axes of the kinematic chain (geometric axes X, Y, and Z).

Define a kinematic element of type "OFFSET" (=l_2) in the X, Y, and Z direction. Corresponds to $TC_CARR4 - $TC_CARR6.

Define a kinematic element of type "AXIS_ROT" (=v_1) in the X, Y, and Z direction. Corresponds to $TC_CARR7 - $TC_CARR9.

Define an element of type "OFFSET" (=l_3) in the X, Y, and Z direction. Corresponds to $TC_CARR15 - $TC_CARR17.

Define an element of type "AXIS_ROT" (=v_2) in the X, Y, and Z direction. Corresponds to $TC_CARR10 - $TC_CARR12.

Define an element of type "OFFSET" (=l_4) in the X, Y, and Z direction. Corresponds to $TC_CARR18 - $TC_CARR20.

If required, definition of a rotary axis offset for axes v1 and v2. The offset is entered in the system variable $NK_A_OFF, which is assigned to the "AXIS_ROT"-type chain element. Corresponds to $TC_CARR24 – TC_CARR25.

At least one rotary axis must be defined.

Note

Sign for rotary axis offset

A positive value for the rotary axis offset for the rotary axes in $TC_CARR24 or $TC_CARR25 is entered in the kinematic chain as a negative value.

Selecting the tool carrier:

The tool carrier is selected with the NC program command in the part program:

```
TCARR = <name>
```

Note

Starting point of the kinematic chain

If a starting point for the kinematic chain of the tool carrier (workpiece or tool) has not been defined via TC_CARR_KIN_PART_START or TC_CARR_KIN_TOOL_START, the ROOT element (world coordinate system) is automatically defined as the starting point of the chains.

$TC_CARR_KIN_CNTRL

$TC_CARR_KIN_CNTRL specifies whether the geometry data of a tool carrier are read from kinematic chain elements or from conventional tool carrier data ($TC_CARRxx). The system variable is bit-coded.

Syntax	Description
$TC_CARR_KIN_CNTRL [n]	Controls transfer of the geometry data of a machine model defined with kinematic chains to parameterize a tool carrier. If Bit0 = 1, the data from $TC_CARRxx are ignored but data are taken from the kinematic chains. It may furthermore be set how the kinematic chain is closed.
	Data type: INT
n	Index of the tool carrier; the maximum number of tool carriers can be defined in machine data MM_NUM_TOOL_CARRIER.

Syntax	Description	
Bit0=1	The tool carrier is parameterized from the kinematic chain elements. The following data of the tool carrier, including the fine offset, are replaced with geometry data from the kinematic chain elements:	
	Conventional tool carrier	Kinematic chain
	$TC_CARR1 – TC_CARR3 (offset vector l_1)	X, Y, and Z component of the offset vector l_1
	$TC_CARR4 – TC_CARR6 (offset vector l_2)	X, Y, and Z component of the offset vector l_2
	$TC_CARR7 – TC_CARR9 (rotary axis direction v_1)	X, Y, and Z component of the rotary axis v_1
	$TC_CARR10 – TC_CARR12 (rotary axis direction v_2)	X, Y, and Z component of the rotary axis v_2
	$TC_CARR15 – TC_CARR17 (offset vector l_3);	X, Y, and Z component of the offset vector l_3
	$TC_CARR18 – TC_CARR20 (offset vector l_4)	X, Y, and Z component of the offset vector l_4
	$TC_CARR24 – TC_CARR25 (rotary axis offsets)	
Bit0=0	The tool carrier parameters are assigned with the data from $TC_CARRxx.	
Bit1=1	The content of offset vector l_1 is changed in such a way that the end point of the **tool** chain coincides with the zero point of the world coordinate system (chain is closed).	
Bit2=1	The content of offset vector l_4 is changed in such a way that the end point of the **workpiece** chain coincides with the zero point of the world coordinate system (chain is closed).	

$TC_CARR_KIN_TOOL_START and $TC_CARR_KIN_PART_START

System variables $TC_CARR_KIN_TOOL_START and $TC_CARR_KIN_PART_START define the starting points of the kinematic chains that are used to parameterize the tool or workpiece part of a tool carrier. If the system variables are empty, the starting point of each kinematic chain is the root element of the kinematic description.

The option of parameterizing only part of the kinematic chain that does not start with the root element makes particular sense and indeed is necessary if the tool carrier is to be deployed in its original function as a "tool extension" and not for the static modeling of the complete transformation kinematics of a machine tool.

Syntax	Description
$TC_CARR_KIN_TOOL_START[n] = <name> $TC_CARR_KIN_PART_START[n] = <name>	Defines the name of the starting elements of the tool or workpiece chain. Data type: STRING
n	Index of the tool carrier; the maximum number of tool carriers can be defined in machine data MM_NUM_TOOL_CARRIER.

$TC_CARR_KIN_TOOL_END and $TC_CARR_KIN_PART_END

System variables $TC_CARR_KIN_TOOL_END and $TC_CARR_KIN_PART_END define the end points of the kinematic chains that are used to parameterize the tool or workpiece part of a tool carrier. At least one of these two names must be not equal to the null string.

Syntax	Description
$TC_CARR_KIN_TOOL_END[n] = <name> $TC_CARR_KIN_PART_END[n] = <name>	Defines the name of the end elements of the tool or workpiece chain.
	Data type: STRING
n	Index of the tool carrier; the maximum number of tool carriers can be defined in machine data MM_NUM_TOOL_CARRIER.

Supplementary conditions and notes

- If the system data of the tool carrier (TC_CARRxx) are replaced with data from kinematic chain elements, the content of system variable $TC_CARRxx is not changed.
- Tool carriers and transformations that were defined with kinematic chains can be active at the same time. It does not matter how the tool carrier is paramerized (directly from the system data TC_CARR… or from kinematic chains).
- It must be possible to map the kinematic chains that will be used to parameterize a tool carrier onto a tool carrier configuration. If this condition is not met, alarm 14149 is triggered.

Measuring tool carriers from kinematic chains.

When measuring the kinematics of a tool carrier and the subsequent offset, the effective offset vectors have to be modified. Because such an offset vector can in principle be formed from any number of partial vectors of the kinematic chain supplying the parameter settings, the chain element to be corrected is not automatically identifiable.

Syntax	Description
$TC_CARR_CORR_ELEM[n, m] = <string>	Name of the next element in the kinematic chain from which the tool carrier data are to be read. The element is used as the correction element for the offset vector with index m.
	Data type: STRING
n	Index of the tool carrier; the maximum number of tool carriers can be defined in machine data MM_NUM_TOOL_CARRIER.
m	Index of the correction vector (value range 0...3)

These chain elements can be uniquely identified with system variable $TC_CARR_CORR_ELEM[m, n]. where m is the index of the tool carrier data set and n is the index of the correction vector (n = 0...3]).

Read out length and direction of the vectors of TCARR.

The currently active offset vectors, rotary axis vectors, and rotary axis offsets can be read out via system variables $PC_TCARR_OFFSET, $PC_TCARR_AX_VECT, and $PC_TCARR_AX_OFFSET. For more detailed information about offset vectors, see General information (Page 1532).

Syntax	Description
$PC_TCARR_OFFSET[m,n] = <value>	Current value of the n-th vector component of offset l_m of an active tool carrier. Maximum 4 offset vectors (l1 to l4) are defined for an active tool carrier. Vector components can be read out with this system variable.
	Data type: DOUBLE
m	Offset vector[m] (where m = 1...4) of the active tool carrier.
n	Vector component n (where 0 ≤ n ≤ 2) of the vector that was selected with field index m.

Syntax	Description
$PC_TCARR_AX_VECT[m,n] = <value>	Current value of the n-th vector component of rotary axis v_m of an active tool carrier. A maximum of 2 rotary axes (v1 to v2) are defined for an active tool carrier. Vector components can be read out with this system variable.
	Data type: DOUBLE
m	Rotary axis direction v1, v2, where m = 1...2 of the active tool carrier.
n	Vector component n (where 0 ≤ n ≤ 3) of the vector that was selected with m.

Syntax	Description
$PC_TCARR_AX_OFFSET[m] = <value>	Current value of the rotational offset of rotary axes 1 and 2. Maximum 2 rotary axes are defined for an active tool carrier. The positions of the rotary axes in their initial state can be read out with this system variable.
	Data type: DOUBLE
m	Rotary offset where m = 1...2 of the active tool carrier.

Tool carrier with more than two rotary axes

For tool carriers that are defined with more than two rotary axes, the third axis is ignored. It describes a constant angle (knee) between the two rotary axes. Axis v_2 in the example is not taken into account. Instead, offset l_2 is formed.

The relevant rotary axes are defined with the system variable $TC_CARR_KIN_ROTAX_NAME in the sequence in which they occur in the chain. Up to two rotary axes may be defined.

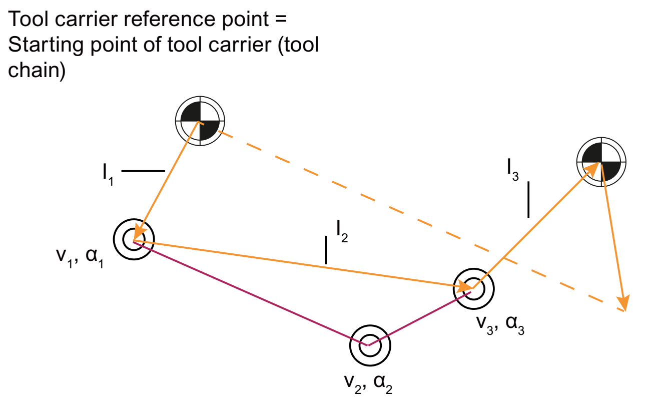

Tool carrier reference point = Starting point of tool carrier (tool chain)
l1
l2
l3
v1, α1
v2, α2
v3, α3

Figure 19-47 Tool carriers with orientation capability with 3 rotary axes.

The rotary axes are not orientation axes, as used for the 5-axis-transformation.

Defining rotation axes (more than two rotary axes)

If a tool carrier comprises more than two rotation axes, this is defined via the relevant axes using the system variable $TC_CARR_KIN_ROTAX_NAME. The name of the chain element is entered in the system variable in this regard. The maximum two rotary axes are defined in the sequence in which they are defined in the kinematic chain. The entries must begin with the index "0".

Syntax	Description
$TC_CARR_KIN_ROTAX_NAME[n, m] = <String>	Name of the element in the kinematic chain which is to be defined as the rotary axis.
	Data type: STRING
n	Index of the tool carrier; the maximum number of tool carriers can be defined in machine data MM_NUM_TOOL_CARRIER.
m	Index of rotary axes (value range 0...1)

Example

You will find a part program for a tool carrier via a kinematic chain under: Example: Tool carrier with orientation capability via kinematic chain (Page 1571).

19.7.4 Inclined surface machining with 3 + 2 axes

Description of function

Inclined machining with 3 + 2 axes describes an extension of the concept of toolholders with orientation capability and applies this concept to machines with a rotary table, on which the orientation of tool and table can be changed simultaneously.

The "Inclined machining with 3 + 2 axes" function is used to machine surfaces with any rotation with reference to the main planes X/Y (`G17`), Z/X (`G18`) and Y/Z (`G19`).

It is possible to produce any orientation of the tool relative to the workpiece by rotating either the tool, the workpiece or both the tool and the workpiece.

The software automatically calculates the necessary compensating movements resulting from the tool lengths, lever arms and the angle of the rotary axis. It is always assumed that the required orientation is set first and not modified during a machining process such as pocket milling on an inclined plane.

Furthermore, the following 3 functions are described, which are required for oblique machining:

- **Position programming** in the direction of the tool orientation independent of an active frame
- Definition of a **frame rotation** by specifying the solid angle
- Definition of the **component of rotation in tool direction** in the programmed frame while maintaining the remaining frame components

Demarcation to 5-axis transformation

If the required functionality specifies that the TCP (Tool Center Point) does not vary in the event of reorientation with reference to the workpiece, even during interpolation, the 5axis software is required.

For more explanations on 5-axis transformations, see:

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

Specification of the toolholder with orientation capability

The toolholder with orientation capability is represented by a general 5axis kinematic sequence described by a data block in the tool compensation memory with a total of 33 REAL values. For toolholders that have two rotary axes for setting the orientation (e.g. a millhead), 31 of these values are constant.

In the current SW version, a data block in the tool compensation memory is described with a total of 47 REAL values. For toolholders that have two rotary axes for setting the orientation, 45 of these values are constant.

The remaining two values are variable and are used to specify the orientation. The constant values describe offsets and directions and setting options for the rotary axes; the variable values describe the angles of the rotary axes.

19.7.5 Machine with rotary work table

System variables

To date, the angles stored in $TC_CARR13 and $TC_CARR14 were used for the calculation of the active tool length with `TCOABS`. This still applies if $TC_CARR21 and $TC_CARR22 do not refer to rotary axes. If $TC_CARR21 or $TC_CARR22 contains a reference to a rotary axis in the channel, the axis position of the relevant axis at the start of the current block is used as the angle, rather than the entry in $TC_CARR13 or $TC_CARR14.

A mixed operating mode is permissible, i.e. the angles can be determined from the entry in the system variables $TC_CARR13 or $TC_CARR14 for one axis, and from the position of a channel axis for the other.

This makes it possible for machines, on which the axes used to set the toolholder with orientation capability are known within the NC, to access their position directly, whereas it was previously necessary, for example, to read system variable $AA_IM[axis] and write the result of the read operation to $TC_CARR13/14. In particular, this removes the implicit preprocessing stop when reading the axis positions.

MD20180

The rotary axis position is used with its programmed or calculated value, when the machine data:

MD20180 $MC_TOCARR_ROT_ANGLE_INCR[i] = 0 (rotary axis increment of the tool carrier that can be oriented)

If the machine data is not zero however, the position used is the nearest grid point obtained for a suitable integer value n from the equation:

φ = $MC_TOCARR_ROT_ANGLE_OFFSET[i] + n * $MC_TOCARR_ROT_ANGLE_INCR[i]

This functionality is required if the rotary axes need to be indexed and cannot, therefore, assume freely-defined positions (e.g. with Hirth tooth systems). System variable $P_TCANG[i] delivers the approximated valued and system variable $P_TCDIFF[i] the difference between the exact and the approximated value.

Frame orientation TCOFR

With `TCOFR` (determination of the angle from the orientation defined by an active frame), the increments are scaled after determination of the angle from the active frame rotation. If the requested orientation is not possible due to the machine kinematic, the machining is aborted with an alarm. This also applies if the target orientation is very close to an achievable orientation. In particular the alarm in such situations cannot be prevented through the angle approximation.

TCARR frame offset

A frame offset as a result of a toolholder change becomes effective immediately on selection of `TCARR=...`. A change in the tool length, on the other hand, only becomes effective immediately if a tool is active.

TCOFR/TCOABS frame rotation

A frame rotation does not take place on activation and a rotation which is already active is not changed. As in case T (only the tool can be rotated), the position of the rotary axes used for the calculation is dependent on the G command `TCOFR`/`TCOABS` and determined from the rotation component of an active frame or from the entries $TC_CARRn.

Activation of a frame changes the position in the workpiece coordinate system accordingly, without compensating motion by the machine itself. The ratios are shown in the figure below:

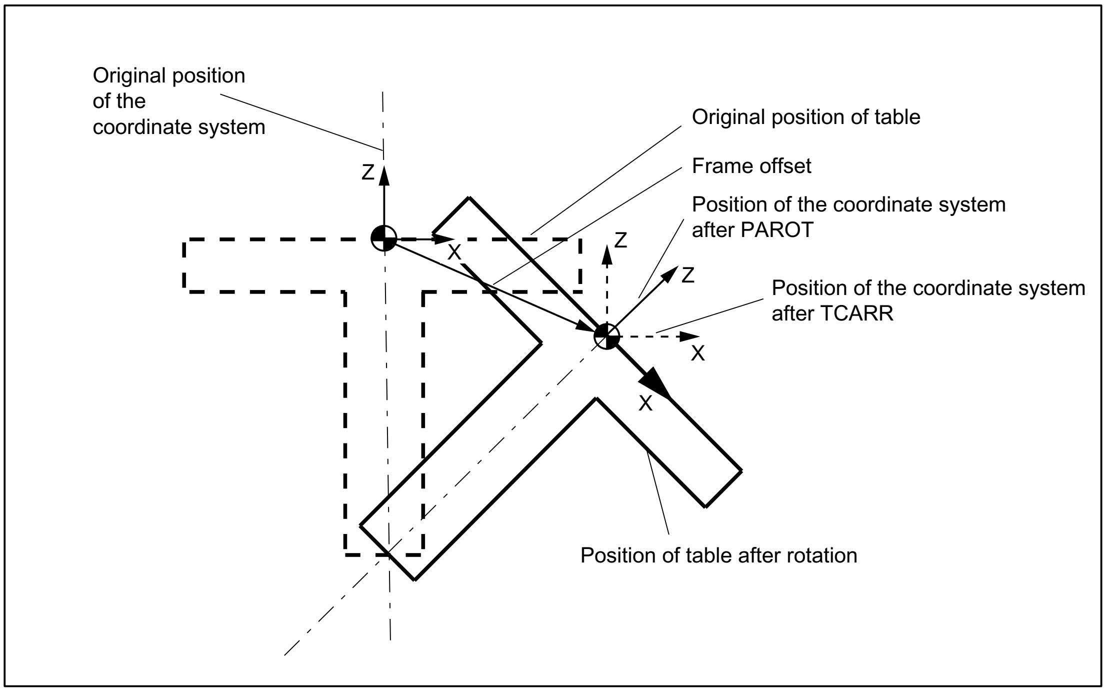

Original position
of the
coordinate system
Original position of table
Frame offset
Position of the coordinate system
after PAROT
Position of the coordinate system
after TCARR
Z
X
Z
Z
X
X
Position of table after rotation

Figure 19-48 Zero offset on activation of a rotary table with `TCARR`

Example

On the machine in the figure, the rotary axis of the table is pointing in the positive Y direction. The table is rotated by +45 degrees. `PAROT` defines a frame, which similarly describes a rotation of 45 degrees about the Y axis. The coordinate system is not rotated relative to the actual environment (marked in the figure with "Position of the coordinate system after `TCARR`"), but is rotated by -45 degrees relative to the defined coordinate system (position after `PAROT`). If this coordinate system is defined with `ROT` Y-45, for example, and if the toolholder is then selected with active `TCOFR`, an angle of +45 degrees will be determined for the rotary axis of the toolholder.

Rotary table

With rotary tables (kinematic types P and M), activation with `TCARR` similarly does not lead to an immediate rotation of the coordinate system (see figure), i.e. even though the zero point of the coordinate system is offset relative to the machine, while remaining fixed relative to the zero point of the workpiece, the orientation remains unchanged in space.

Activation of kinematic types P and M

With kinematics of type P and M the selection of a toolholder activates an additive frame (table offset of the toolholder with orientation capability), which takes into account the zero point offset as a result of the rotation of the table.

The zero offset can be written to a dedicated system frame $P_PARTFR. For this, the bit 2 must be set in the machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK (system frames (SRAM))

The basic frame identified by following machine data is then no longer required for the zero offset:

MD20184 $MC_TOCARR_BASE_FRAME_NUMBER (number of the basic frames for taking the table offset)

Activation of kinematic type M

With kinematics of type M (tool and table are each rotary around one axis), the activation of a toolholder with `TCARR` simultaneously produces a corresponding change in the effective tool length (if a tool is active) and the work offset.

Rotations

Depending on the machining task, it is necessary to take into account not only a work offset (whether as frame or as tool length) when using a rotary toolholder or table, but also a rotation. However, the activation of an orientable toolholder never leads directly to a rotation of the coordinate system.

TOROT

If only the tool can be rotated, a frame whose Z axis points in the direction of the tool can be defined with `TOFRAME` or `TOROT`.

PAROT

If the coordinate system needs to be fixed relative to the workpiece, i.e. not only offset relative to the original position but also rotated according to the rotation of the table, then `PAROT` can be used to activate such a rotation in a similar manner to the situation with a rotary tool.

With `PAROT`, the translations, scalings and mirrorings in the active frame are retained, but the rotation component is rotated by the rotation component of a toolholder with orientation capability corresponding to the table.

`PAROT` and `TOROT` take into account the overall change in orientation in cases where the table or the tool are oriented with two rotary axes. With mixed kinematics, only the corresponding component caused by a rotary axis is considered. It is thus possible, for example, when using `TOROT`, to rotate a workpiece such that an oblique plane lies parallel to the XY plane fixed in space, whereby rotation of the tool must be taken into account in machining where any holes to be drilled, for example, are not perpendicular to this plane.

Language command `PAROT` is not rejected if no orientable toolholder is active. This causes no changes in the programmed frame.

Note

For further information about the `TCARR` and `TOROT` as well as `PAROT` functions with regard to channel-specific system frames, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 705)".

19.7.6 Procedure when using toolholders with orientation capability

Creating a tool carrier

Setting the number of available tool carrier data sets

The number of available tool carrier data sets in the control system is defined with machine data:

MD18088 $MN_MM_NUM_TOOL_CARRIER (maximum number of definable tool carriers)

The number must be set according to the following rule:

MD18088 $MN_NUM_TOOL_CARRIER =	
	<highest number of tool carriers required in **one** channel> *
	<number of parameterized channels>

The number of tool carriers parameterized in the machine data is then distributed equally across the parameterized channels and TO units.

<available number of tool carriers per channel or TO unit> =	
	MD18088 $MN_NUM_TOOL_CARRIER /
	<number of parameterized channels>

Note

For further explanations on definition and assignment of a TO unit to a channel with machine data MD28085 by machine data $MC_MM_LINK_TOA_UNIT, see:

References:
Function Manual, Extended Functions; Memory Configurations (S7)

Zero setting of tool carrier data:

You can use the command $TC_CARR1[0] = 0 to zero all values of all data sets.

Individual tool carrier data sets can be deleted selectively with the NC command `DELTC` or the PI service `_N_DELTCAR`.

Accessing the data of a tool carrier:

- Part program
  - Write: `$TC_CARR<n>[<m>] = <value>`
    Value <value> is written to parameter <n> of tool carrier <m> .
  - Read: `<value> = $TC_CARR<n>[<m>]`
    Parameter <n> of tool carrier <m> is read. If the referenced tool carrier is not defined, an alarm is displayed.
- OPI interface
  The parameters of a tool carrier with orientation capability can be read and written with the system variables $P_TCANG[<n>].

Data backup

The system variables specified above are saved as part of the general NC data backup.

Selecting the tool carrier

A tool carrier with number <m> is selected with the `TCARR = <m>` NC program command (`TCARR`**T**ool **Carr**ier).

`TCARR = 0` deselects an active tool carrier.

New tool or new tool carrier

When a new tool is activated, it is always treated as if it was mounted on the active tool carrier.

A new tool carrier is activated immediately when it is programmed. It is not necessary to change tools or reprogram the active tool. The tool carrier (number) and tool (number) are independent and can be used in any combination.

Tool carrier from G command 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 command `TCOABS` is activated in G 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 commands is active in G group 42 when the tool carrier is selected:

- `TCOFR` or `TCOFRZ`
  The tool carrier with orientation capability is set so that the tool points in the Z direction.
- `TCOFRX`
  The tool carrier with orientation capability is set so that the tool points in the X direction.
- `TCOFRY`
  The tool carrier 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 command `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 tool carrier is changed.

Tool carrier from G command of group 53

The G commands 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 command of group 6 (`G17` - `G19`), which is active at the time `TOFRAME` is called, determines the tool orientation.

If no tool carrier is active, or if a tool carrier 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 tool carriers. The same applies to the new X and Y directions.

Instead of `TOFRAME` or `TOROT`, one of the G commands `TOFRAMEX`, `TOFRAMEY`, `TOROTX`, or `TOROTY` can be used. The meanings of the axes are interchanged accordingly.

Group change

Changing the G command from group 42 (`TCOABS`, `TCOFR`, etc.) causes recalculation of the tool length components.

The (programmed) angles of rotation stored in the tool carrier data is not affected, with the result that the angles originally stored in the tool carrier data is 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 group 22 (tool compensation type).

For nonrotating tool carriers, the behavior depends solely on the active plane of the G command of 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 commands of group 22, `CUT3DC` and `CUT3DF`, are not affected by the tool carrier 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 tool carrier is activated is used to specify the angle.
2. If $TC_CARR21 or $TC_CARR22 does not contain a reference to a rotary axis, the values contained in $TC_CARR13 or $TC_CARR14 are used.

Example

The control first calculates an angle of 10 degrees for one axis. The specified angle is 750 degrees. 720 degrees (= 2 * 360 degrees) are then added to the initial angle, resulting in a final angle of 730 degrees.

Rotary axis offset

Rotary axis offsets can be specified with system variables $TC_CARR24 and $TC_CARR25. A value not equal to zero in one of these parameters means that the initial state of the associated rotary axis is the position specified by the parameter (and not position zero). All angle specifications then refer to the coordinate system displaced by this value.

When the machining plane is changed (`G17` - `G19`), only the tool length components of the active tool are interchanged. The components of the tool carrier are not interchanged. The resulting tool length vector is then rotated in accordance with the current tool carrier and, if necessary, modified by the offsets belonging to the tool carrier.

The two tool carrier angles α_1 and α_2 are determined using a frame, independent of the active plane currently selected (`G17` - `G19`).

Limit values

Limit angles (software limits) can be specified for each rotary axis in the system variable set ($TC_CARR30 to $TC_CARR33) used to describe the tool carrier 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 tool carrier with orientation capability is activated.

19.7.7 Programming

Selecting the toolholder

A toolholder is selected with the number **m** of the toolholder with:

```
TCARR = m
```

Access to toolholder data blocks

The following access is possible from the part program:

The current value of the parameter **n** for the tool holder **m** is **written** with the new "value" with::

```
$TC_CARRn[m] = value
```

The parameters of a tool holder m can, as far as the toolholder data set is already defined, **read** with:

`value = $TC_CARRn[m]` (Value must be a `REAL` variable)

The toolholder data set number must lie in the range, which is defined by the machine data:

MD18088 $MN_MM_NUM_TOOL_CARRIER (Total number of toolholder data sets that can be defined)

This number of toolholder data sets, divided by the number of active channels, can be defined for a channel.

Exception:

If settings, which deviate from the standard, are selected via the machine data:

MD28085 $MC_MM_LINK_TOA_UNIT (Assignment of TO unit to a channel).

Canceling all toolholder data blocks

All values of all toolholder data sets can be deleted from within the part program using one command.

$TC_CARR1[0] = 0

Values not set by the user are preset to 0.

Activation

A toolholder becomes active when both a **toolholder** and a **tool** have been activated. The selection of the toolholder alone has no effect. The effect of selecting a toolholder depends on the G command `TCOABS` / `TCOFR` (modal G group for toolholders).

Changing the G command in the `TCOABS` / `TCOFR` group causes recalculation of the tool length components when the toolholder is active. With `TCOABS`, the values stored in the toolholder data for both angles of rotation α1 and α2 are used to determine the tool orientation.

With `TCOFR`, the two angles are determined from the current frame. The values stored in the toolholder data are not changed, however. These are also used to resolve the ambiguity that can result when the angle of rotation is calculated from one frame. Here, the angle that deviates least from the programmed angle is selected from the various possible angles.

Note

For more explanations on the programming of tool compensations with toolholder kinematic and for the system variables see:

References:
Programming Manual, Job Planning

19.7.8 Supplementary conditions and control system response for orientation

Full orientation

For a given data set that describes a certain kinematic, all the conceivable special orientations can only be displayed when the following conditions are satisfied:

- The two vectors v_1 and v_2 that describe the rotary axes must also be defined (i.e. both vectors must not be equal to zero).
- The two vectors v_1 and v_2 must be perpendicular to each other.
- The tool orientation must be perpendicular to the second rotary axis.

Non-defined orientation

If these conditions are not satisfied and an orientation that cannot be achieved by an active frame is requested with `TCOFR`, an alarm is output.

Vector/angle of rotation dependencies

If vector v_1 or v_2, which describes the direction of a rotary axis, is set to zero, the associated angle of rotation α1 or α2 must also be set to zero. Otherwise, an alarm is produced. The alarm is not output until the toolholder is activated, i.e. when the toolholder is changed.

Tool fine compensation combined with orientation

Tool fine compensations and toolholders **cannot** be combined. The activation of tool fine compensation when a toolholder is active, and vice versa the activation of the toolholder when tool fine compensation is active, produces an alarm.

Automatic toolholder selection, RESET

For `RESET` or at program start, a toolholder can be selected automatically via the machine data:

MD20126 $MC_TOOL_CARRIER_RESET_VALUE (Active toolholder at RESET)

It is handled similar to the controlled selection of a tool via the machine data:

MD20120 $MC_TOOL_RESET_VALUE (Tool length compensation Power up (RESET/TP-End))

The behavior at `RESET` or at program start is controlled as in the case of tool selection via the same bit 6 in the machine data:

MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET/TP-End)

Or:

MD20112 $MC_START_MODE_MASK (definition of initial control system settings at NC-START)

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 479)".

SW 6.3 and higher

If `TCOABS` was active for the last selection before reset, the behavior is unchanged compared to previous versions. A different active G code causes the toolholder with orientation capability to be activated with the frame that was active before the last reset. Modified toolholder data ($TC_CARR...) are also considered. If these data are unchanged, the toolholder is activated in exactly the same state as before reset. If the toolholder data were changed after the toolholder selection before reset, selection corresponding to the last frame is not always possible. In this case, the toolholder with orientation capability is selected according to the G-Code (group 42) values valid at this time and the active frame.

MD22530 output of auxiliary functions to PLC

That, optionally, a constant or an M code is output when the toolholder is selected, whose number of the code is derived from the toolholder number. Can be set with the machine data:

MD22530 $MC_TOCARR_CHANGE_M_CODE (M code at toolholder change)

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 479)".

Toolholder kinematics

The following supplementary conditions must be met for toolholder kinematics:

- Tool orientation in initial state, both angles $α_1$ and $α_2$ zero, as per default setting, even if:
  - `G17` parallel to Z
  - `G18` Parallel to Y
  - `G19` parallel to Z
- A permissible position in terms of the axis limits must be achievable.
- For any possible orientation to be set, the two rotary axes must be perpendicular to each other.
  For machines, on which the table is rotated by both axes, the tool orientation must also be perpendicular to the first rotary axis.
  For machines with mixed kinematics, the tool orientation must be perpendicular to the axis which rotates the tool, i.e. also the first rotary axis.

The following applies to orientations specified **in a frame**:

- The orientation specified in a frame must be achievable with the defined toolholder kinematics, otherwise an alarm is output.
  This situation can occur if the two rotary axes required to define the kinematics are not perpendicular to each other.
  This applies if fewer than two rotary axes are defined and is the case:
  - With **kinematic type T with rotary tool**, if the tool axis, which defines the tool direction, is not perpendicular to the **second** axis.
  - With **kinematic types M and P with rotary workpiece**, if the tool axis, which defines the tool direction, is not perpendicular to the **first** axis.
- Rotary axes, which require a frame with a defined tool orientation in order to reach a specific position, are only determined unambiguously in the case of one rotary axis. Two general solutions exist for two rotary axes.
- In all cases where ambiguities may arise, it is particularly important that the approximate angles expected from the frame are stored in the tool data, and that the rotary axes are in the vicinity of the expected positions.

Response with ASUP, REPOS

The toolholder can be changed in an asynchronous subprogram (ASUB). When the interrupted program is resumed with `REPOS`, the approach motion of the new toolholder is taken into account and the program continues with this motion. The treatment here is analogous to tool change in an ASUB.

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 479)".

19.7.9 Examples

19.7.9.1 Example: Toolholder with orientation capability

Requirement

The following example uses a toolholder, which is described fully by a rotation about the Y axis. It is therefore sufficient to enter only one value to define the rotary axis (block `N20`).

Blocks `N50` to `N70` describe an end mill with radius 5 mm and length 20 mm.

Block `N90` defines a rotation of 37 degrees about the Y axis.

Block `N120` activates the tool radius compensation and all settings are made to describe the compensation in the following blocks with a rotation of 37 degrees about the Y axis.

```
N10                                       ; Definition of toolholder 1
N20 $TC_CARR8[1] = 1                      ; Component of first rotary axis in Y di-
                                          rection
N30
N40                                       ; Definition of tool offset 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
```

19.7.9.2 Example of toolholder with orientation capability with rotary table

Use of the MOVT command

For use of the `MOVT` command it is assumed that the program is running on a 5axis machine, on which the tool rotates about the Y axis in case of a rotation of the B axis:

```
N10 TRAORI()
N20 X0 X0 Z0 B45 F2000                      ; Setting the tool orientation
N30 MOVT=-10                                ; Infeed movement 10 mm in tool direc-
                                            tion
                                            ; (under 45 degrees in the Y-Z plane)
```

```
N40 MOVT=AC(20)                             ; Retraction in tool direction at dis-
                                            tance 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 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 component of 3rd offset vector
N190 $TC_CARR17[1] = 0                   ; Z component of 3rd offset vector

N200 $TC_CARR18[1] = 0                   ; X component of 4th offset vector
N210 $TC_CARR19[1] = 0                   ; Y component of 4th offset vector
N220 $TC_CARR20[1] = 15                  ; Z component of 4th offset vector

N230 $TC_CARR21[1] = A                   ; Reference for 1st axis
N240 $TC_CARR22[1] = B                   ; Reference for 2nd axis
N250 $TC_CARR23[1] = "P"                 ; Toolholder type

N260 X0 Y0 Z0 A0 B45 F2000
N270 TCARR=1 X0 Y10 Z0 T1 TCOABS
```

```
N280 PAROT
N290 X0 Y0 Z0
N300 G18 MOVT=AC(20)
N310 G17 X10 Y0 Z0
N320 MOVT=-10
N330 PAROTOF
N340 TCOFR
N350 X10 Y10 Z-13 A0 B0
N360 ROTS X-45 Y45
N370 X20 Y0 Z0 D0
N380 Y20
N390 X0 Y0 Z20
N400 M30
```

The definition of the toolholder with orientation capability is given in full. The components which contain the value 0 need not actually be given, as they are preset to zero in any case.

The toolholder is activated in `N270`.

As *$TC_CARR21* and *$TC_CARR22* refer to the machine axes `A` and `B` and `TCOABS` is active, the values in *$TC_CARR13* and *$TC_CARR14* are ignored, i.e. the axis position `A0 B45` is used for the rotation.

The rotation of the 4th offset vector (length 15 mm in Z direction) around the B axis causes an offsetting of the zero point by X10.607 [= 15 * sin(45)] and Z-4.393 [= -15 * (1. - cos(45))]. This zero offset is taken into account by an automatically written basic or system frame so that the position X10.607 Y10.000 Z8.607 is approached. In the Z direction the tool selection leads to an additional offset of 13 mm; the Y component is not affected by the table rotation.

`N280` defines a rotation in accordance with the rotation of the table of the toolholder with orientation capability. The new X direction thus points in the direction of the bisecting line in the 4th quadrant, the new Z axis in the direction of the bisecting line in the 1st quadrant.

The zero point is approached in `N290`, i.e. the machine position X10.607 Y0 Z-4.393, since the position of the zero point is not changed by the rotation.

`N300` traverses in Y to the position Y33.000, since `G18` is active and the Y component is not affected by the active frame. The X and Z positions remain unchanged.

The position X17.678 Y0 Z1.536 is approached in `N310`.

`N320` changes only the Z position to the value -8.464 as a result of the `MOVT` command. As only the table can be rotated, the tool orientation remains unchanged parallel to the machine Z direction, even if the Z direction of the active frame is rotated by 45 degrees.

`N330` deletes the basic or system frame; thus the frame definition from `N280` is undone.

In `N340`, `TCOFR` specifies that the toolholder with orientation capability is to be aligned according to the active frame. Since a rotation is no longer active in `N330` due to the `PAROTOF` command, the default setting is applied. The frame offset becomes 0.

`N350` thus approaches the position X10 X10 Z0 (= Z-13 + tool length). Notice: Through the simultaneous programming of both rotary axes A and B the actual position of the toolholder with orientation capability is made to match that used in `N340`. The position approached by the three linear axes is dependent on this position, however.

In N360, solid angles are used to define a plane whose intersecting lines in the XZ and in the YZ plane each form an angle of +45 degrees or -45 degrees with the X or Y axis. The plane defined in such a way therefore has the following position: The surface normal points towards the solid diagonals.

N370 traverses to the position X20 Y0 Z0 in the new coordinate system. Since the tool is deselected with D0 at the same time, there is no longer an additional offset in Z. Since the new X axis lies in the old X-Z 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.

19.7.9.3 Calculation of compensation values on a location-specific and workpiece-specific basis

Tool with adapter

A tool with adapter and tool carrier with orientation capability is defined in the following program example. To simplify the overview, only length L1 is different from zero for the additive and insert offsets and for the adapter in case of the tool itself. The offset vectors of the tool carrier with orientation capability are all zero.

Program code	Comment
N10 $TC_TP2[1]="MillingTool"	; Name of identifier
N20 $TC_TP7[1]=9	; Location type
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]=	; Length offset 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: Turret
N120 $TC_MAP2[1]="Turret"	; Name of a magazine
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

Program code	Comment
N170 $TC_MPP2[1,1]=9	; Location type
N180 $TC_MPP4[1,1]=2	; Location status
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"	; Name of a magazine
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 type
N280 $TC_MPP4[9999,1]=2	; Location status
N290 $TC_MPP5[9999,1]=1	; Spindle no. 1
N300 $TC_MDP2[1,1]=0	; Distance from spindle to mag. 1
; Definition of tool carrier 1	
N310 $TC_CARR10[1]=1	; Component 2. Axis of rotation in the X direction
N320 $TC_CARR14[1]=45	; Angle of rotation of 2nd axis
N330 $TC_CARR23[1]="T"	; Tool mode
N340 Stopre	
N350 $SC_WEAR_TRANSFORM='B101'	
N360 T0 D0 DL=0	
N370 ROT X30	
N380 G90 G1 G17 F10000 X0 Y0 Z0	
N390 T="MillingTool" X0 Y0 Z0 TOWSTD	; X 0.000 Y11.110 Z 0.001
N400 T="MillingTool" X0 Y0 Z0 TOWMCS	; X 0.000 Y10.100 Z 1.011
N410 T="MillingTool" X0 Y0 Z0 TOWWCS	; X 0.000 Y 9.595 Z 0.876
N420 TCARR=1 X0 Y0 Z0	; X 0.000 Y 6.636 Z 8.017
N430 G18 X0 Y0 Z0	; X10.100 Y-0.504 Z 0.876
N440 M30	

Explanations regarding the example above

Starting at block `N390`, various methods are used to approach position `X0 Y0 Z0`. The machine positions reached are specified in the blocks in comments. After the program a description is given of how the positions were reached.

`N390`: The adapter transformation 5 (block `N90`) transforms length `L1` into length `L2`. Only the actual adapter dimension is not subject to this transformation. The Y value (`L2` with `G17`) results from the sum of the tool length (10), tool wear (1), sum offset (0.1), and insert offset (0.01). The adapter dimension (0.001) is in Z (`L1`).

`N400`: In block `N350`, bits 0 and 2 are enabled in setting data:

SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)

This means that the tool wear and the insert offset are not subject to the adapter transformation because of `TOWMCS` in block `N400`. The sum of these two compensations is 1.01. The Z position is, therefore, increased by this amount and the Y position is reduced by this amount compared with block `N390`.

`TOWWCS` is active in `N410`. The sum of the tool wear and the insert offset is thus effective in the active workpiece coordinate system. In block `N370`, a rotation through 30 degrees is activated around 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 tool carrier 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 tool carrier are zero, there is no additional work offset. The tool carrier with orientation capability acts on the sum of the tool length, sum offset and adapter dimension. The resulting vector component is `X0 Y7.141 Z7.142`. To this, as in block `N410`, the sum of tool wear and insert offset evaluated in WCS is added.

`G18` is activated in `N430`. The components of the tool length sum, sum offset and adapter dimension are interchanged accordingly. The tool carrier with orientation capability continues to act on this new vector (rotation through 45 degrees about X axis). The resulting vector component thereby is `X10.100 Y0.0071 Z0.0071`. The vector formed from tool wear and insert offset (`X0 Y-0.505 Z0.875`) is not affected by the change of plane. The sum of the two vectors yields the dimension specified in the comment in `N430`.

19.7.9.4 Example: Tool carrier with orientation capability via kinematic chain

Example of a tool carrier with mixed kinematics (rotatable tool and table)

The example shows a tool carrier that is defined via a kinematic chain. The chain is automatically closed at the endpoints of the two separate chains (tool and table).

Program code	Comment
	; Definition tool carrier 1
N1070 _MIXED_1_S:	
N1080 IF (DELOBJ("KIN_CHAIN_ELEM") <> 0)	
N1080 IF (DELOBJ("KIN_CHAIN_ELEM") <> 0)	
N1090 SETAL(62000)	
N1100 ENDIF	
N1110 _KIE_CNT = 0	
N1120 _CARR_CNT = 1	
N1130 TCARR=0	; Deactivate current tool carrier.
N1140 $TC_CARR1[0] = 0	; Delete all tool carrier data.
N1150 $TC_CARR_KIN_CNTRL[_CARR_CNT]=7	; Tool carrier via kinematic chain
	; + Close tool and part chain.
N1160 $TC_CARR_KIN_TOOL_START[_CARR_CNT]="Y_AXIS"	; Start element of the tool chain.
N1170 $TC_CARR_KIN_TOOL_END[_CARR_CNT]="B_OFFSET_CORR"	; End element of the tool chain.

Program code	**Comment**
N1180 $TC_CARR_KIN_PART_START[_CARR_CNT]="X_AXIS"	; Start element of the part chain.
N1190 $TC_CARR_KIN_PART_END[_CARR_CNT]="END_PART_CHAIN"	; End element of the part chain.
N1200 $TC_CARR21[_CARR_CNT]=B	
N1210 $TC_CARR22[_CARR_CNT]=C	
N1220 $TC_CARR23[_CARR_CNT]="M"	
N1230 $NK_NAME[_KIE_CNT] = "ROOT"	
N1240 $NK_PARALLEL[_KIE_CNT]="X_AXIS"	
N1250 $NK_TYPE[_KIE_CNT] = "OFFSET"	
N1260 $NK_NEXT[_KIE_CNT] = "Y_AXIS"	
N1270 _KIE_CNT=_KIE_CNT+1	
	;***** Start tool chain *****
N1280 $NK_NAME[_KIE_CNT] = "Y_AXIS"	
N1290 $NK_TYPE[_KIE_CNT] = "AXIS_LIN"	
N1300 $NK_OFF_DIR[_KIE_CNT,1] = 1	
N1310 $NK_AXIS[_KIE_CNT] = "Y1"	
N1320 $NK_NEXT[_KIE_CNT] = "Z_AXIS"	
N1330 _KIE_CNT=_KIE_CNT+1	
N1340 $NK_NAME[_KIE_CNT] = "Z_AXIS"	
N1350 $NK_TYPE[_KIE_CNT] = "AXIS_LIN"	
N1360 $NK_OFF_DIR[_KIE_CNT,2] = 1	
N1370 $NK_AXIS[_KIE_CNT] = "Z1"	
N1380 $NK_NEXT[_KIE_CNT] = "CLOSE_HEAD"	
N1390 _KIE_CNT=_KIE_CNT+1	
N1400 $NK_NAME[_KIE_CNT] = "CLOSE_HEAD"	
N1410 $NK_TYPE[_KIE_CNT] = "OFFSET"	
N1420 $NK_OFF_DIR[_KIE_CNT,0] = 0	
N1430 $NK_OFF_DIR[_KIE_CNT,1] = 0	
N1440 $NK_OFF_DIR[_KIE_CNT,2] = 0	
N1450 $NK_NEXT[_KIE_CNT] = "B_AXIS"	
N1460 _KIE_CNT=_KIE_CNT+1	
N1470 $NK_NAME[_KIE_CNT] = "B_AXIS"	
N1480 $NK_TYPE[_KIE_CNT] = "AXIS_ROT"	
N1490 $NK_OFF_DIR[_KIE_CNT,1] = 1	
N1500 $NK_OFF_DIR[_KIE_CNT,2] = 1	
N1520 $NK_A_OFF[_KIE_CNT] = -27	
N1540 $NK_AXIS[_KIE_CNT] = "B1"	
N1550 $NK_NEXT[_KIE_CNT] = "B_OFFSET"	
N1560 _KIE_CNT=_KIE_CNT+1	
N1570 $NK_NAME[_KIE_CNT] = "B_OFFSET"	
N1580 $NK_TYPE[_KIE_CNT] = "OFFSET"	
N1590 $NK_OFF_DIR[_KIE_CNT,2] = -30.6	
N1600 $NK_NEXT[_KIE_CNT] = "B_OFFSET_CORR"	
N1610 _KIE_CNT=_KIE_CNT+1	
N1620 $NK_NAME[_KIE_CNT] = "B_OFFSET_CORR"	

Program code	Comment
N1630 $NK_TYPE[_KIE_CNT] = "OFFSET"	
N1640 $NK_NEXT[_KIE_CNT] = ""	
N1650 _KIE_CNT=_KIE_CNT+1	
	;***** End tool chain *****
	;***** Start part chain *****
N1660 $NK_NAME[_KIE_CNT] = "X_AXIS"	
N1670 $NK_TYPE[_KIE_CNT] = "AXIS_LIN"	
N1680 $NK_OFF_DIR[_KIE_CNT,0] = 1	
N1690 $NK_AXIS[_KIE_CNT] = "X1"	
N1700 $NK_NEXT[_KIE_CNT] = "C_OFFSET"	
N1710 _KIE_CNT=_KIE_CNT+1	
N1720 $NK_NAME[_KIE_CNT] = "C_OFFSET"	
N1730 $NK_TYPE[_KIE_CNT] = "OFFSET"	
N1740 $NK_OFF_DIR[_KIE_CNT,0] = 300	
N1750 $NK_OFF_DIR[_KIE_CNT,1] = 150	
N1760 $NK_NEXT[_KIE_CNT] = "C_OFFSET_CORR"	
N1770 _KIE_CNT=_KIE_CNT+1N1780 $NK_NAME[_KIE_CNT] = "C_OFF-SET_CORR"	
N1790 $NK_TYPE[_KIE_CNT] = "OFFSET"	
N1800 $NK_OFF_DIR[_KIE_CNT,0] = 0	
N1810 $NK_OFF_DIR[_KIE_CNT,1] = 0	
N1820 $NK_OFF_DIR[_KIE_CNT,2] = 0	
N1830 $NK_NEXT[_KIE_CNT] = "C_AXIS"	
N1840 _KIE_CNT=_KIE_CNT+1	
N1850 $NK_NAME[_KIE_CNT] = "C_AXIS"	
N1860 $NK_TYPE[_KIE_CNT] = "AXIS_ROT"	
N1870 $NK_OFF_DIR[_KIE_CNT,2] = -1	
N1890 $NK_A_OFF[_KIE_CNT] = -58	
N1910 $NK_AXIS[_KIE_CNT] = "C1"	
N1920 $NK_NEXT[_KIE_CNT] = "CLOSE_PART"	
N1930 _KIE_CNT=_KIE_CNT+1	
N1940 $NK_NAME[_KIE_CNT] = "CLOSE_PART"	
N1950 $NK_TYPE[_KIE_CNT] = "OFFSET"	
N1960 $NK_OFF_DIR[_KIE_CNT,0] = 0	
N1970 $NK_OFF_DIR[_KIE_CNT,1] = 0	
N1980 $NK_OFF_DIR[_KIE_CNT,2] = 0	
N1990 $NK_NEXT[_KIE_CNT] = "END_PART_CHAIN"	
N2000 _KIE_CNT=_KIE_CNT+1	
N2010 $NK_NAME[_KIE_CNT] = "END_PART_CHAIN"	
N2020 $NK_TYPE[_KIE_CNT] = "OFFSET"	
	;***** End part chain *****

19.8 Modification of the offset data for rotatable tools

19.8.1 Introduction

Function

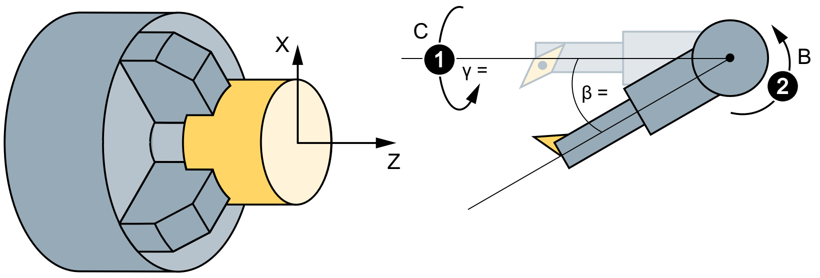
With function "Modification of the offset data for rotatable tools", the modified geometrical relationships that result when a tool is rotated (predominantly turning tools, but also drilling and milling tools) relative to the workpiece being machined can be taken into account.

The actual tool rotation is always determined from a currently active tool carrier with orientation capability (see Chapter "Toolholder with orientation capability (Page 1532)") or from an orientation transformation defined with kinematic chains (see Function Manual Special Functions; Chapter "Transformation definition with kinematic chains").

The angle of rotation of the tool carrier with orientation capability is normally (but not necessarily) defined with the TCOFR command from an active frame. This method can be used to define the tool orientation independently of the actual kinematics with which the tool is rotated, identically in each case with the help of two angles.

In an active kinematic transformation, the required angles can be determined with the function ORISOLH (see "Calculating orientations (ORISOLH) (Page 1586)").

The two machine-independent orientation angles β (Beta) and γ (Gamma) are used to define the tool rotation. β is the angle of rotation and the applicate (typically a B axis in `G18`) and γ a rotation around the abscissa (Typically a C axis in G18). The rotation is first executed around Y, finally around β. I.e., the γ axis is rotated by the β axis:



① Rotation around the abscissa

② Rotation around the applicate

Figure 19-49 Tool rotation

Commissioning

The function is commissioned using the machine and setting data.

See Section "Commissioning (Page 1583)".

Activation

Modification of the offset data for rotatable tools is activated in the NC program with the language commands CUTMOD (in combination with tool carriers with orientation capability) or CUTMODK (for orientation transformations that were defined with kinematic chains).

See Section "Activating the modification of the offset data for rotatable tools (CUTMOD, CUTMODK) (Page 1594)".

Results

When the function and tool rotation are active, the modified data are made available in system variables and OPI variables.

See Section "Activating the modification of the offset data for rotatable tools (CUTMOD, CUTMODK) (Page 1594)".

19.8.2 Rotating turning tools

19.8.2.1 Cutting edge position, cut direction, and angle for turning tools

Turning tools

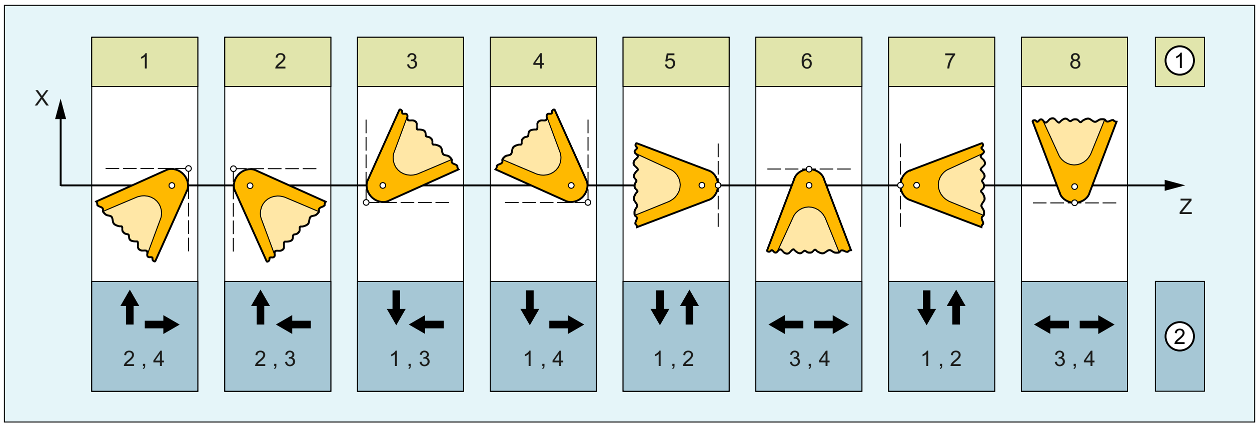
Turning tools means the following tools whose tool type ($TC_DP1) has values in the range of 500 to 599. Grinding tools (tool types 400 to 499) are equivalent to turning tools.

Tools are treated independently of tool type such as turning tools if:

SD42950 $SC_TOOL_LENGTH_TYPE = 2

Cutting edge position and cut direction

Turning tools are limited by their main and secondary cutting edges. The cutting edge position is defined via the position of the primary and secondary cutting edges relative to the coordinate axes. Two different cutting directions can be assigned to each cutting edge position.



① Cutting edge position (1 - 8)

② Cutting directions (1 - 4) assigned to the cutting edge position

Figure 19-50 Cutting edge position and cut direction

Cutting edge position

For cutting edge position 1 - 4, the primary and secondary cutting edge are in the same quadrant.

For a cutting edge position of 5 - 8, the primary and secondary cutting edge are in adjacent quadrants or there is a coordinate axis between the two cutting edges.

The cutting edge position is stored in the tool parameter $TC_DP2.

Cut direction

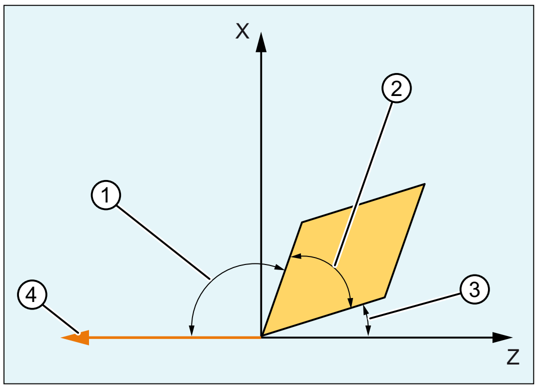
The cut directions 1 - 4 denote a positive or negative direction of the coordinate axes:

- 1: Ordinate **-**
- 2: Ordinate **+**
- 3: Abscissa **-**
- 4: Abscissa **+**

The cut direction is stored in the tool parameter $TC_DP11.

Holder angle and clearance angle

The following figure shows the holder angle and clearance angle for a rotary tool with cutting edge position 3. The machining plane is G18 (Z/X). The cut direction is 3 (negative Z or abscissa direction).



①	Holder angle
②	Wedge or plate angle = 180° - holder angle - clearance angle
③	Clearance angle
④	Cut direction

Figure 19-51 Angle and cut direction for a turning tool

Cut direction

The cut direction specifies the reference direction of the holder angle. The clearance angle is the angle measured between the inverse cut direction and the adjacent cutting edge (positive).

Holder angle and clearance angle are stored in the tool parameters $TC_DP10 or $TC_DP24.

Note

Cut direction and tool angle are relevant only in the cutting edge positions 1 - 8.

19.8.2.2 Modifications during the rotation of turning tools

Description of tool orientation

Unlike milling tools, turning tools are not rotation-symmetric. This means that normally 3 degrees of freedom or three rotary axes are required to describe the tool orientation. The concrete kinematics therefore, is independent of the machine only to the extent the desired orientation can be set. If necessary, the third degree of freedom can be substituted by a rotation of the tool coordinate system.

Note

The division of the orientation into one component created by the tool carrier with orientation capability and a second component achieved via a rotation of the coordinate system is the responsibility of the application. The control does not provide any further functionality in this regard.

If the tool is oriented using a kinematic transformation, the user can use function ORISOLH to calculate the orientation. (see Chapter "Calculating orientations (ORISOLH) (Page 1586)").

Angle between tool insert and machining plane

If a turning tool turns by an angle against the machining plane (i.e. around an axis in the machining plane, typically a C axis) that is not a multiple of 180°, then the configuration of the (circular) tool cutting edge in the machining plane becomes an ellipse. It is assumed that the deviations from the circular form arising on account of such rotations is so insignificant that they can be ignored (tilt angle < 5°). i.e. the control always ignores the tool orientation and assumes a circular cutting edge.

This also means that with reference to the active plane, the control accepts only a rotation by 180° as a setting deviating from the initial position. This limitation is valid for the shape of cutting edge only. The tool lengths are always considered correctly in random spatial rotations.

A rotation by 180° around an axis in the machining plane means that while using the tool at the same position, the spindle rotation direction with reference to the use of the unturned tool must be inverted.

A maximum value allowed for the angle between the tool insert and the machining plane can be specified in setting data SD42998 $SC_CUTMOD_PLANE_TOL (see also Chapter "Parameter assignment (Page 1583)").

Cutting edge position, cutting direction, and tool angle

Cutting edge position and cut direction are also not modified like the cutting edge reference point (see below) if the tool is rotated from the plane by ± 90° (with a tolerance of app. 1°) because then the configuration of the cutting edge is not defined in the current plane.

If a transformation is active, the active complete frame is always included in the calculation of the actual tool rotation relative to the workpiece. If a tool carrier with orientation capability is active, the frame is only included in the calculation if bit 21 is set in machine data MD20360 $MC_TOOL_PARAMETER_DEF_MASK (Page 1583). In this case, any table component of the tool carrier is no longer treated separately because it must be assumed that this frame component is already included in the complete frame (system frame PAROT).

When orientation transformation is active, bit 17 of machine data MD20360 $MC_TOOL_PARAMETER_DEF_MASK can be used, should the cutting edge no longer be in the active machining plane, either to project the cutting edge into the machine plane (bit 17 = 0) or alternatively to rotate the cutting edge into the machining plane (bit 17 = 1) in order to calculate the modified cutting data.

Note

MD20360, bit 17

If this bit is set, it is the responsibility of the user to ensure, using an additional frame or by changing the orientation, to ensure that the geometric conditions during machining match those that exist during calculation.

If bit 20 of machine data MD20360 $MC_TOOL_PARAMETER_DEF_MASK is set to value "0", standard values that deviate from zero can be used to calculate the modified cutting edge position and the modified cutting direction for when the cutting edge parameters for the holder angle and cutting direction ($TC_DP10 or $TC_DP24) have value 0. The following rules apply:

- A holder angle of 0° has the same effect as an angle of 112.5° on cutting edge positions 1 to 4.
- A holder angle of 0° has the same effect as an angle of 67.5° on cutting edge positions 5 to 8.
- A clearance angle of 0° has the same effect as an angle of 22.5° on cutting edge positions 1 to 4.
- A clearance angle of 0° has the same effect as an angle of 67.5° on cutting edge positions 5 to 8.

The holder angle and clearance angle might only be temporarily mapped onto the standard values specified above to be able to calculate the modified cutting edge position and modified cutting direction. I.e., reading system variables $TC_DP10 or $TC_DP24 still returns value "0".

If the tool rotates in the plane (rotation around an axis vertical to the machining place or around the Y axis for G18), the cutting edge position is determined from the resulting angle for the clearance and holder angles. If these two angles are not specified for the tool (i.e. $TC_DP10 and $TC_DP24 are both zero), the new cutting edge position is determined just from the turning angle. A special feature here is that the cutting edge position only changes in steps of 90°. I.e. the cutting edge position remains independent of the initial state either in the value range 1 to 4 or 5 to 8. The new cutting edge position is then determined exclusively from the angle of rotation if the specified values for holder angle and clearance angle are impermissible (negative values, resulting tool insert angle negative or greater than 90°). Impermissible angles can be set to output an alarm (see MD20125 $MC_CUTMOD_ERR (Page 1583)). Clearance angle and holder angle are not modified in all these cases.

Depending on the rotation, the cut direction is modified in such a way that the resulting clearance angle remains less than 90°. If the original cut direction and the original cutting edge position do not match, then the cut direction is not modified during rotation of the tool. This situation can be set to output an alarm (see MD20125 $MC_CUTMOD_ERR).

The angle of rotation in the plane, as it was determined from the tool carrier with orientation capability or from the active kinematic transformation, is provided in the OPI variables pTCutMod or ptCut- ModS and in system variables $P_CUTMOD_ANG or $AC_CUTMOD_ANG. This angle is the original angle without any final rounding to multiples of 45° or 90°.

Limit cases

If, for a turning tool, the cutting edge position, cut direction, clearance and holder angles have valid values so that all cutting edge positions (1 to 8) are possible through suitable rotations in the plane, then the cutting edge positions 1 to 4 are preferred to cutting edge positions 5 to 8 in the cases in which one of the cutting edges (main or secondary cutting edge) is away from the coordinate axis by less than half the input increment (0.0005° for an input specification of 3 decimal digits).

The following is applicable in all other cases (milling tools or turning tools without valid cutting edge parameters) in which rotation is possible only in 90° steps: If the amount of the rotation angle is smaller than 45° + 0.5 input increments (corresponds to 45.0005° for an input specification of 3 decimal places), the cutting edge position and cut direction are not changed. I.e. these cases are treated as rotations that are smaller than 45°. Rotations, the amount of which deviates from 180° by less than 45° + 0.5 input increments are treated identically as rotations in the range of 135° to 225°.

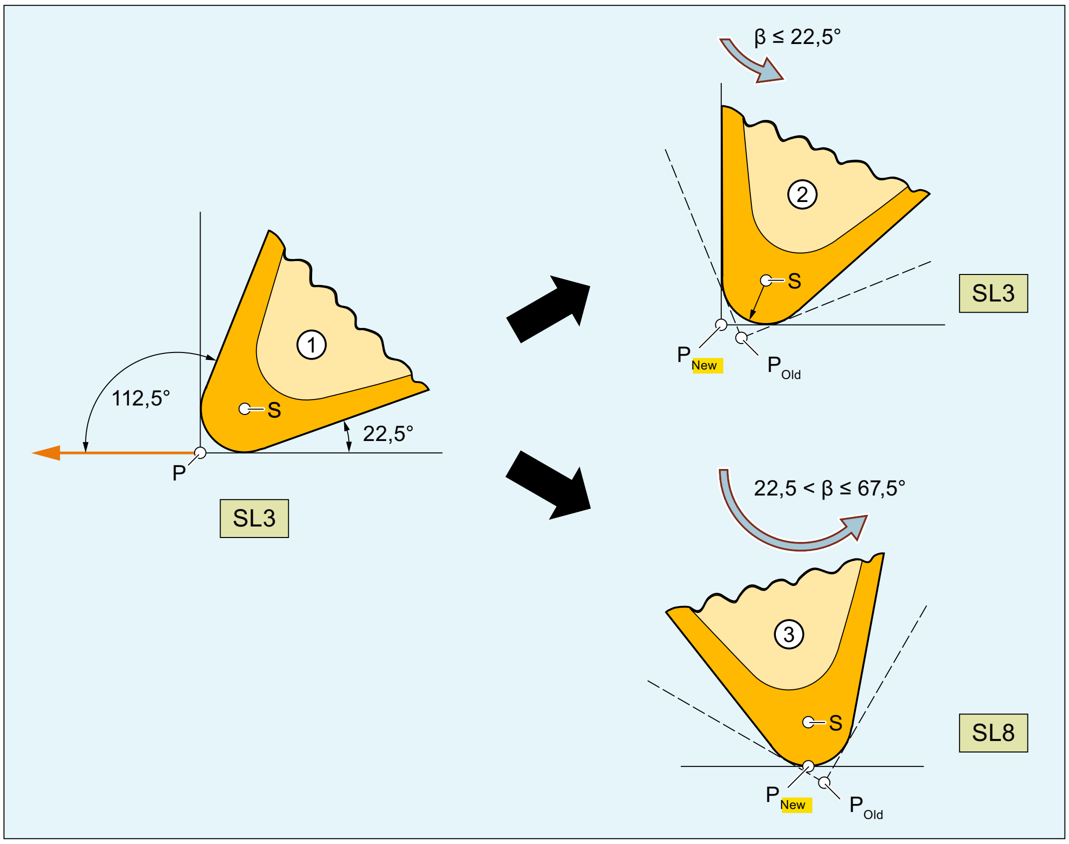
Cutting edge reference point

The cutting edge center point and the cutting edge reference point are defined for turning tools. The position of these two points relative to each other is defined by the cutting edge position.

The distance of the two points for cutting edge positions 1 to 4 is equal to √2 times the cutting edge radius; for cutting edge positions 5 to 8 it is equal to 1 times the cutting edge radius. In the first case, the cutting edge reference point relative to the cutting edge center point lies in the machining plane on a bisecting line, while in the second case it lies on a coordinate axis.

If your rotate the tool by a random angle around an axis vertical to the machining plane, the the cutting edge reference point would also rotate if it had a fixed position relative to the tool. The above-mentioned condition (position on an axis or a bisecting axis) is not fulfilled in most cases. This is not desirable. Instead, the cutting edge reference point should always be modified in such a way that the distance vector between cutting edge reference point and cutting edge center point has one of the mentioned 8 directions. The cutting edge position must be modified for this if necessary.

The ratios are shown with examples in the figure below:



S:	Cutting edge center point
P:	Cutting edge reference point
SL:	Cutting edge position
①	Tool with cutting edge position 3, clearance angle 22.5°, and holder angle 112.5°
②	For rotations of the tool up to 22.5°, the cutting edge position is maintained, the position of the cutting edge reference point relative to the tool however, is compensated in such a way that the relative position of both points is maintained in the machining plane.
③	For bigger tool rotations (up to 67.5°), the cutting edge position changes to value 8.

Figure 19-52 Cutting edge reference point and cutting edge position for tool rotation

Note

As the cutting edge reference point is defined by the tool length vector, modifying the cutting edge reference point changes the effective tool length.

Tool rotation

The rotation of the tool tip and the tool adapter is described by tool parameter $TC_DPROT. This parameter (angle) is only practical for tools that do not rotate symmetrically (e.g. turning tools, boring bars). Rotation is performed around the direction that is defined by the vector product of tool normal vector and tool orientation vector. For an active tool, this vector can be read with system variable $P_TOOLBIN. For standard turning tools, this vector points toward the Z axis for G18. I.e. it turns around the axis around which angle γ also turns.

19.8.3 Rotation of milling and drilling tools

19.8.3.1 Cutting edge position for milling and tapping tools

Milling and drilling tools

Milling and drilling tools means the following tools whose tool type ($TC_DP1) has values in the range of 100 to 299.

Tools are treated independently of tool type such as milling and tapping tools if:

SD42950 $SC_TOOL_LENGTH_TYPE = 1

Cutting edge position

A cutting edge position which is modified accordingly has also been introduced for the milling and tapping tools defined as specified above (see Chapter "Modifications during rotation of milling and tapping tools (Page 1583)").

Note

A cutting edge position that is declared for tools that are not milling or tapping tools, or turning tools according to the definitions stated above, is not evaluated.

The cutting edge position of the tapping and milling tools is stored in tool parameter $TC_DP2 as in the case of turning tools. Based on the definition of the cutting edge position for turning tools, this parameter can assume the values 5 to 8. Here, the cutting edge position specifies the orientation (the direction of the rotation axis) of the tool:

Cutting edge position	Direction of rotation axis of tool
5	Abscissa **+**
6	Ordinate **+**
7	Abscissa **-**
8	Ordinate **-**

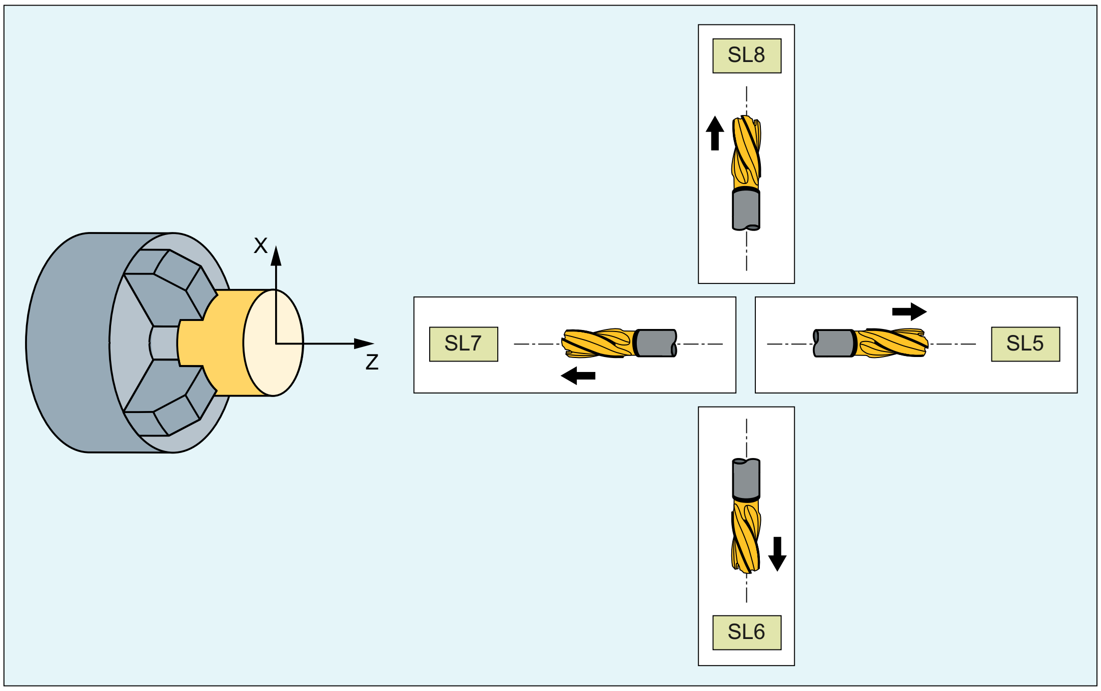


Figure 19-53 Cutting edge position 5 - 8 of a milling tool

19.8.3.2 Modifications during rotation of milling and tapping tools

The cutting edge position is recalculated appropriately during a rotation of a milling or tapping tool. Cut direction and tool angle (clearance angle or holder angle) are not defined for milling and tapping tools so that the change in cutting edge position is derived exclusively from the rotation. Thus, for milling and tapping tools, the cutting edge position always changes when the amount of rotation with reference to the zero setting is more than 45°.

19.8.4 Commissioning

19.8.4.1 Parameter assignment

Response in the event of an error

Different fault conditions can occur during the activation of the function "Modification of the offset data for rotatable tools" (by an explicit call in the part program or by tool selection). For errors with error numbers < 100 (see Diagnostics Manual), it is possible to define whether the error will trigger output of an alarm and, if so, whether such an alarm will only be displayed or will also trigger program stop. The setting is made in the following machine data:

MD20125 $MC_CUTMOD_ERR

Two bits of the machine data are assigned to each fault condition:

Fault status	Bit	Meaning
No valid cutting direction is defined for the active tool.	0	Alarm output
	1	Program stop
The cutting edge angle (clearance angle and holder angle) of the active tool are both zero.	2	Alarm output
	3	Program stop
The clearance angle of the active tool has an impermissible value (< 0° or > 180°).	4	Alarm output
	5	Program stop
The holder angle of the active tool has an impermissible value (< 0° or > 90°).	6	Alarm output
	7	Program stop
The tool insert angle of the active tool has an impermissible value (< 0° or > 90°).	8	Alarm output
	9	Program stop
The cutting edge position - holder angle combination of the active tool is not permitted. (The holder angle must be ≤ 90° for cutting edge position 1 to 4; for cutting edge positions 5 to 8 it must be ≥ 90°.)	10	Alarm output
	11	Program stop
The tool insert is not in the machining plane and angle γ between the tool insert and the machining plane exceeds the upper limit specified with setting data SD42998 $SC_CUTMOD_PLANE_TOL.	12	Alarm output
	13	Program stop
The tool insert is not in the machining plane. The magnitude of angle α is greater than 1°. Angle α is the angle of rotation around the coordinate axis, which is perpendicular both to the axis of rotation of angle β and to the axis of rotation of angle γ (in the case of G18, the X axis).	14	Alarm output
	15	Program stop

Note

The 2nd bit (program stop) is only effective if the corresponding 1st bit (alarm output) is set.

Note

Errors with error numbers ≥ 100 always result in an alarm being output and program stop.

Initialization value for CUTMOD

The function that can be programmed with NC command CUTMOD (Page 1586) is automatically initialized with the following value after a warm restart:

MD20127 $MC_CUTMOD_INIT = <initialization value for CUTMOD>

Default value: 0

If MD20127 = -2, the value from the following machine data becomes the initialization value:

MD20126 $MC_TOOL_CARRIER_RESET_VALUE (active tool carrier on reset)

Function-specific tool settings

The following machine data is a bit string with which various functions relating to tools can be controlled:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK

The following bits are relevant for function "Modification of the offset data for rotatable tools".

Bit	Meaning	
17	Bit 17 is used to set whether, on modification of the offset data for turning and grinding tools, the cutting edge plane for calculating the offset values will be projected into the machining plane or rotated.	
	= 0	The cutting edge plane is projected into the machining plane.
	= 1	The cutting edge plane is rotated into the machining plane.
	Note: Bit 17 is only evaluated if the function is active in combination with a kinematic orientation transformation.	
18	Bit 18 is used to determine how the plane is defined for modification of offset data for turning and grinding tools.	
	= 0	Priority is given to defining the plane with setting data SD42940 $SC_TOOL_LENGTH_CONST over defining the plane with the G commands of G Group 6 (plane selection G17 - G19).
	= 1	The active plane (G17 - G19) is always used.
20	If tool parameters $TC_DP10 (holder angle) and/or $TC_DP24 (clearance angle) have the value "0", default values deviating from "0" can be used to calculate the modified cutting edge position and the modified cutting direction.	
	= 0	The following default values are used: • Holder angle 112.5° for cutting edge positions 1 - 4 • Holder angle 67.5° for cutting edge positions 5 - 8 • Clearance angle 22.5° for cutting edge positions 1 - 4 • Clearance angle 67.5° for cutting edge positions 5 - 8
	= 1	An alarm is output.
	Note: Bit 20 is used to establish compatibility with older software versions.	
21	Bit 21 is used to set whether the active total frame will be taken into account in the modification of the offset data when tool carrier with orientation capability is active.	
	= 0	Rotation of the table component of the tool carrier is taken into account. Frames are ignored.
	= 1	Instead of the table component of the tool carrier, the active total frame is used. The total frame can also contain a table component of the tool carrier.

Difference between tool tip plane and machining plane for CUTMOD or CUTMODK

The maximum permissible angle through which the tool insert can be turned away from the machining plane (standard G18) when function CUTMOD or CUTMODK (Page 1594) is called (≙ maximum permissible deviation of γ from one of the two standard positions 0° or 180°) is set in the setting data:

SD42998 $SC_CUTMOD_PLANE_TOL

Example:

SD42998 = 5.0 ⇒ The tool insert must not be turned by more than 5° away from the machining plane.

Note

SD42998 = 0

If SD42998 $SC_CUTMOD_PLANE_TOL is set to "0", variations of up to 89° are permissible!

Difference between tool tip plane and machining plane for ORISOLH

The maximum permissible angle through which the tool insert can be turned away from the machining plane when function ORISOLH (Page 1594) is called is set in the setting data:

SD42999 $SC_ORISOLH_INCLINE_TOL

19.8.5 Programming

19.8.5.1 Calculating orientations (ORISOLH)

The predefined ORISOLH function helps the user to set the rotary axis positions of a machine so that a turning tool can be brought into a defined, kinematic-independent position relative to the workpiece. Prerequisite is that a 6-axis transformation is active that has been parameterized with kinematic chains.

Two basic functions are available:

- Tool alignment
  The β and γ angles are specified. The function calculates the angles of the three orientation axes required for this.
- Direct tool alignment
  The angles of the second and third orientation axes are specified. The function calculates the associated β and γ angles as well as that of the missing first orientation axis.

Note

Order of the orientation axes

If you run through the kinematic chain that describes the structure of the machine, from the workpiece to the tool, then the following specifications apply for the order of the three orientation axes of a 6-axis transformation:

- The orientation axis that is closest to the **workpiece** is the **first** orientation axis.
- The orientation axis that is closest to the **tool** is the **third** orientation axis.

Generally, the first orientation axis is a spindle and the corresponding rotation is therefore implemented in these cases through a rotating frame.

Syntax

```
<RetVal> = ORISOLH(<Cntrl>,<W1>,<W2>)
```

Meaning

`ORISOLH`:	Function call		
`<RetVal>`:	Function return value		
	Data type:	INT	
	Range of values:	0, -2, -3, ..., -17	
	Values:	0	Function has ended without an error.
		-2	No valid transformation (6-axis orientation transformation) is active.
		-3	The first parameter (<Cntrl>) is negative.
		-4	The unit position of the first parameter (<Cntrl>) is invalid. Only the values 0 and 1 are permissible.
		-5	The tens position of the first parameter (<Cntrl>) is invalid. Only the values 0 to 3 are permissible.
		-6	The hundreds position of the first parameter (<Cntrl>) is invalid. Only the values 0 and 1 are permissible.
		-7	The thousands position of the first parameter (<Cntrl>) is invalid. Only the values 0 to 3 are permissible.
		-8	Angle γ is too large for the "Direct tool alignment" function.
		-9	At least one of the specified axis positions violates an axis limit for the "Direct tool alignment" function.
		-10	No tool is active.
		-11	The requested orientation cannot be set.
		-12	The adaptation of the free axis angle for the Hirth joint is not possible for the first or only solution.
		-13	The adaptation of the free axis angle for the Hirth joint is not possible for the second solution.
		-14	The adaptation of the free axis angle for the Hirth joint is not possible for either of the two solutions.
		-15	The first orientation axis is parameterized as Hirth axis.
		-16	The second as well as the third rotary axis has been parameterized as Hirth axis. Only one of the two axes can be the Hirth axis.
		-17	At least one of the specified axis positions is not compatible with the associated Hirth joint for the "Swivel directly" function.

`<Cntrl>`:	Controls the behavior of the function		
	Data type:	INT	
	The <Cntrl> parameter is decimal coded (unit to thousands position):		
	Unit position:	The unit position controls the response to errors.	
		xxx**0**	In the event of an error (return value < 0), alarm 14106 is output and program processing is aborted. **Note:** The alarm is also output irrespective of the value of the unit position when the <Cntrl> parameter is negative.
		xxx**1**	In the event of an error (return value < 0) **no** alarm is output. The user can react suitably in the program.
	Tens position:	Controls the behavior when an orientation axis with Hirth joint is present. **Note:** This parameter is only evaluated for the "Tool alignment" function (i.e. when the hundreds position has the value "0").	
		xx**0**x	The axis position is rounded off to the nearest position.
		xx**1**x	The axis positions are rounded off so that the difference of the β angle to its programmed value is minimal.
		xx**2**x	The axis positions are rounded off so that the β angle is equal to the highest possible value which is less than the programmed value (β is rounded down).
		xx**3**x	The axis positions are rounded off so that the β angle is equal to the lowest possible value which is greater than the programmed value (β is rounded up).
	Hundreds position:	Specifies which function is to be executed or the significance of the two following parameters <W1> and <W2>.	
		x**0**xx	"Tool alignment" function Parameters <W1> and <W2> have the following meaning: • <W1> = β • <W2> = γ The associated angles of the orientation axes are calculated.
		x**1**xx	"Direct tool alignment" function <W1> is the position specification for the second orientation axis, <W2> is the position specification for the third orientation axis of a 6-axis transformation. The position of the first orientation axis and the β and γ angles are defined which are compatible with the two position specifications. If no error occurs, two solutions are always output in the $P_ORI_POS[<n>, <m>] system variables. The first index <n> (0 or 1) refers to the solution and the second index <m> (0 ... 2) to the orientation axis: • $P_ORI_POS[0/1, **0**]: Position of the first orientation axis • $P_ORI_POS[0/1, **1**]: Angle β • $P_ORI_POS[0/1, **2**]: Angle γ A check is made as to whether the position specifications <W1> and <W2> are compatible with any Hirth joints or active software limits. If this is not the case, a corresponding error number is returned (see <RetVal> parameter).

			If the angles <W1> and <W2> are selected arbitrarily, the cutting edge of the tool is generally not in the machining plane. The angle γ through which the cutting edge is rotated out of the machining plane, must not be greater than the limit value which is defined by the setting data SD42999 $SC_OR-ISOLH_INCLINE_TOL.
	Thousands position:	Specifies which positions of the solutions may be modified when the hundreds position has the value "0", i.e. for the "Tool alignment" function.	
		0xxx	The calculated axis positions should be as close as possible to the current machine axis positions.
		1xxx	The calculated axis positions for modulo axes should be as close as possible to the middle of the modulo range, for other axes as close as possible to 0. For non-modulo axes, this means that the axis positions are reduced to the range -180° … +180°.
		2xxx	The calculated axis positions should be reduced to the range -180° … +180° irrespective of the axis type.
`<W1>`:	First angle The meaning results from the hundreds position of the <Cntrl> paramter.		
	Data type:	REAL	
`<W1>`:	Second angle The meaning results from the hundreds position of the <Cntrl> paramter.		
	Data type:	REAL	

Note

Parameters that have not been programmed have the default value "0".

19.8.5.2 Calculating orientations (ORISOLH): Further information

Further information

The number of solutions found together with further status information when executing the ORISOLH function, can be read via the following system variables:

System variable	Meaning		
$P_ORI_POS [<n>, <m>]	Returns the angles of the orientation axes that result from the orientation programming.		
	<n>:	Index of the solution	
		Range of values:	0, 1
	<m>:	Index of the orientation axis	
		Range of values:	0 ... 2
			The order of the orientation axes (1 ... 3) refers to the definition of the axes in $NT_ROT_AX_NAME.
	When the ORISOLH function is called in the "Direct tool alignment" mode, the $P_ORI_POS[0/1, 1] and P_ORI_POS[0/1, 2] variables contain the values of the two angles β and γ belonging to the two solutions. The first solution entered in $P_ORI_POS[<n>, <m>], i.e. with the index <n> = 0, is always the solution that is selected by the control when the requested orientation is approached directly. The second index <m> refers to the orientation axis, i.e. on $NT_ROT_AX_NAME. The axis positions entered in $P_ORI_POS[<n>, <m>] take into account the offsets entered in $NK_OFF and $NK_OFF_FINE, i.e. these axis angles can be used in the following blocks to set the required orientation without any further modification. If a rotary axis is a Hirth axis, the solution positions are rounded off to the nearest position of rest of the Hirth joint. For Hirth jointed rotary axes, you can read the differences between the axis positions for the exact solutions and those of the solutions adapted to the Hirth incrementing in the $P_ORI_DIFF system variable.		
$P_ORI_DIFF [<n>, <m>]	Returns the difference between the exact positions of the orientation axes and those provided in $P_ORI_POS that result from the orientation programming.		
	<n>:	Index of the solution	
		Range of values:	0, 1
	<m>:	Index of the orientation axis	
		Range of values:	0 ... 2
			The order of the orientation axes (1 ... 3) refers to the definition of the axes in $NT_ROT_AX_NAME.
	The content can only be not equal to zero when the positions are incremented (Hirth joint), i.e. when the system data $NT_HIRTH_INCR of the relevant axis is not equal to zero and when this axis is a manual rotary axis.		

System variable	Meaning		
$P_ORI_SOL	If for an orientation transformation with more than one orientation axis, the axis angles are calculated that should result in a specified orientation, there is generally more than one solution. The $P_ORI_SOL system variables contain the number of valid solutions together with additional status information. The content of $P_ORI_SOL is coded as follows:		
	Values < 0	General error states	
		-1	No solutions have been calculated yet for the active transformation (missing call of ORISOLH).
		-2	A transformation is not active, or the active transformation is not an orientation transformation (6-axis transformation) that can provide positions for a specified orientation programming.
		-4	The desired orientation cannot be set with the present kinematics.
		-5	No solution was found when the ORISOLH function was called in the "Direct tool alignment" mode.
		-6	Angle γ is too large when the ORISOLH function was called in the "Direct tool alignment" mode.
		-7	An angle was specified when the ORISOLH function was called in the "Direct tool alignment" mode that cannot be set because of the Hirth joint.
		-8	The first orientation axis (frame axis) must not be parameterized as Hirth axis.
		-9	The second as well as the third rotary axis has been parameterized as Hirth axis. Only one of the two axes can be the Hirth axis.
		-10	No adaptation of the solution(s) to the Hirth joint has been found.
	Values > 0 Unit position	Number of mathematically possible solutions without consideration of axis limits and any error conditions.	
		0	There is no solution, i.e. the requested orientation cannot be set. There can be three different causes for this case: ● In principle, the requested orientation cannot be achieved because of the machine kinematics (orientation axes not arranged at right angles) even with an arbitrary traversing range of the orientation axes. In this case, the tens and hundreds positions of $P_ORI_SOL are both zero, the $P_ORI_STAT status variables assigned to the orientation axis have the value "-4". ● The calculated solutions cannot be achieved because they would violate the axis limits. The positions of the orientation axes that would result without the axis limits, can be read in $P_ORI_POS. ● Axis positions were specified when the ORISOLH function was called in the "Direct tool alignment" mode which would result in either the orientation vector or the orientation normal vector of the tool being aligned parallel to the first orientation axis, whose position is to be calculated. The position of this axis is not defined in these cases.

System variable	Meaning			
		1	There is a solution. There can be three different causes for this case: • Based on the specified orientation and the machine kinematics, there is only one solution (from the mathematical point of view, two coinciding solutions) even without consideration of the axis limits. This case occurs at the edge of the orientation range for kinematics that are not at right angles. $P_ORI_POS contains both (identical) solutions. • There is only one solution because a second solution is invalid due to the violated axis limits. The valid solution is always the first solution in $P_ORI_POS. The second solution which would result when the axis limits are not taken into account, can also be read in $P_ORI_POS. • This is the normal case when the ORISOLH function is called in the "Direct tool alignment" mode. For the specified axis positions of two orientation axes, there is generally only one valid position for the missing orientation axis to be calculated.	
		2	There are two solutions.	
		8	There are an infinite number of solutions, i.e. the position of an orientation axis (the polar axis) is arbitrary. However, from the two possible positions of the other axes, one is excludes because of the violated axis limits.	
		9	There are an infinite number of solutions, i.e. the position of an orientation axis (the polar axis) is indefinite. The indefinite axis can be determined from the hundreds position or from the $P_ORI_STAT system variable.	
	Values > 0 Tens position	Bit-coded display for violated axis limits. The precise cause of the error can be determined from the $P_ORI_STAT system variable.		
		Bit 0 (value 10):	For at least one solution, at least one axis limit of the first orientation axis is violated.	
		Bit 1 (value 20):	For at least one solution, at least one axis limit of the second orientation axis is violated.	
		Bit 2 (value 40):	For at least one solution, at least one axis limit of the third orientation axis is violated.	
	Values > 0 Hundreds position	Bit-coded display for non-defined axis positions (can only occur when there is an infinite number of solutions, i.e. when the unit position is equal to "9").		
		Bit 0 (value 100):	The position of the first orientation axis is not defined.	
		Bit 1 (value 200):	The position of the second orientation axis is not defined.	
		Bit 2 (value 400):	The position of the third orientation axis is not defined.	
	The designations first, second and third orientation axis refer to the definition of the axes in $NT_ROT_AX_NAME.			

System variable	Meaning		
$P_ORI_STAT [<n>]	Returns the status for each of the maximum three orientation axes after ORISOLH has been called.		
	<n>:	Index of the orientation axis (correspnds to the index of the relevant orientation axis in $NT_ROT_AX_NAME)	
		Range of values:	0 ... 2
			The order of the orientation axes (1 ... 3) refers to the definition of the axes in $NT_ROT_AX_NAME.
	The content of $P_ORI_STAT is coded as follows:		
	Values < 0	General error states	
		-1	The status is not defined (missing call of ORISOLH).
		-2	A transformation is not active, or the active transformation is not an orientation transformation (6-axis transformation) that can provide positions for a specified orientation programming.
		-3	The axis is not included in the active transformation.
		-4	The position of the axis cannot be calculated because the requested orientation cannot be achieved with the present kinematics even with an arbitrary assumed traversing range of the axis.
		-5	Axis positions were specified when the ORISOLH function was called in the "Direct tool alignment" mode which would result in either the orientation vector or the orientation normal vector of the tool being aligned parallel to the first orientation axis, whose position is to be calculated. The position of this axis is not defined in these cases.
		-6	Angle γ is too large when the ORISOLH function was called in the "Direct tool alignment" mode.
		-7	An angle was specified when the ORISOLH function was called in the "Direct tool alignment" mode that cannot be set because of the Hirth joint.
		-8	The first orientation axis (frame axis) must not be parameterized as Hirth axis.
		-9	The second as well as the third rotary axis has been parameterized as Hirth axis. Only one of the two axes can be the Hirth axis.
		-10	No adaptation of the solution(s) to the Hirth joint has been found.
	Values > 0 Unit position	Bit-coded display for violated axis limits of the first solution.	
		Bit 0 (value 1):	The first solution violates the lower axis limit.
		Bit 1 (value 2):	The first solution violates the upper axis limit.
	Values > 0 Tens position	Bit-coded display for violated axis limits of the second solution.	
		Bit 0 (value 10):	The second solution violates the lower axis limit.
		Bit 1 (value 20):	The second solution violates the upper axis limit.
	Values > 0 Hundreds position	Display of a non-defined axis position.	
		Bit 0 (value 100):	The position of the orientation axis is not defined, i.e. the requested orientation is achieved with each arbitrary setting of the rotary axis (polar po-

System variable	Meaning		
			sition). This information is also contained in the $P_ORI_SOL system variable.
	Of the error numbers that indicate a violation of the axis limits, several can occur simultaneously. When an axis limit is violated, an attempt is made to reach a position within the permissible axis limits by adding or subtracting multiples of 360°. If this is not possible, it is not clearly defined whether the lower or the upper axis limit has been violated. If there is no solution for the requested orientation ($P_ORI_SOL = 0), the status of the orientation axes in the transformation is "0".		

Note

$NT_ROT_AX_NAME

This system variable refers to a maximum of three axes used for setting the orientation. It contains the names of the chain elements ($NK_NAME) that define the machine axes (rotary axes) that must perform the orientation movements resulting from a kinematic transformation. The order in which the maximum three rotary axes are contained in this system variable is irrelevant for the machine kinematics because this is derived from the structure of the kinematic chains. However, as it defines the order in which other variables access the rotary axes, the order of the orientation axes in $NT_ROT_AX_NAME must match the kinematic description.

Note

Status information

The status information that shows, for example, that an orientation cannot be achieved or can only be achieved when relevant axis limits are violated, does not trigger an NC alarm. It is the responsibility of the user to react suitably to the specified conditions.

19.8.5.3 Activating the modification of the offset data for rotatable tools (CUTMOD, CUTMODK)

The modification of the offset data for rotatable tools is activated in the NC program via the CUTMOD (in combination with orientable tool carriers) or CUTMODK language command (for orientation transformations that were defined by means of kinematic chains).

Note

As the orientable tool carriers and orientation transformations that were defined by means of kinematic chains cannot be active at the same time, there are no conflicts between the two variants.

Syntax

```
CUTMOD = <Value>
```

or

```
CUTMODK = <Command>
```

Meaning

`CUTMOD`:	Function call in combination with orientable tool carriers		
`<Value>`:	Assigned value		
	Data type:	INT	
	Value:	0	The function is deactivated. The values supplied from system variables $P_AD... are the same as the corresponding tool parameters.
		> 0	The function is activated if an orientable tool carrier with the specified number is active, i.e. the activation is linked to a specific orientable tool carrier. The values supplied from system variables $P_AD... may be modified with respect to the corresponding tool parameters depending on the active rotation. The deactivation of the designated orientable tool carrier temporarily deactivates the function; the activation of another orientable tool carrier permanently deactivates it. This is the reason why in the first case, the function is re-activated when again selecting the same orientable tool carrier; in the second case, a new selection is required - even if at a subsequent time, the orientable tool carrier is re-activated with the specified number. The function is not influenced by a reset.
		-1	The function is always activated if an orientable tool carrier is active. When changing the tool carrier or when de-selecting it and a subsequent new selection, CUTMOD does not have to be set again.
		-2	The function is always activated if an orientable tool carrier is active whose number is the same as the currently active orientable tool carrier. If an orientable tool carrier is not active, then this has the same significance as `CUTMOD=0`. If an orientable tool carrier is active, then this has the same significance as when directly specifying the actual tool carrier number.
		< -2	Values less than 2 are ignored, i.e. this case is treated as if CUTMOD was not programmed. **Note:** This value range should not be used as it is reserved for possible subsequent expansions.
`CUTMODK`:	Function call in combination with orientation transformations that have been defined by means of kinematic chains		

`<Command>`:	Assigned Command		
	Data type:	STRING	
	Value:	`"NEW"`	The states of an active transformation defined with kinematic chains relevant for the "Modification of the offset data", the name of the transformation and the current contour frame are saved. **Note:** This command is only permissible when a suitable transformation (TRAORI_DYN, TRAORI_STAT or TRAANG_K) is active.
		`"OFF"`	Switches the active "Modification of the offset data" off. The data previously stored with "NEW" is retained. **Note:** This command is also permissible when CUTMODK is not active. It then remains without effect. Any data set present for the "Modification of the offset data" is retained.
		`"ON"`	With this command, the "Modification of the offset data" is re-activated with a data set previously stored with the "NEW" command. If a transformation with the name of the stored data set is active when this command is executed, the "Modification of the offset data" takes effect immediately. Otherwise, the activation is delayed until an active transformation is activated.
		`"CLEAR"`	As with the "OFF" command, switches the "Modification of the offset data" off and also deletes the stored data set. **Note:** This command is also permissible when CUTMODK is not active.

Note

SD42984 $SC_CUTDIRMOD

The CUTMOD or CUTMODK command replaces the function that can be activated using the setting data SD42984 $SC_CUTDIRMOD. However, this function remains available unchanged. However, as it doesn't make sense to use both functions in parallel, it can only be activated if CUTMOD is equal to zero and CUTMODK is the zero string.

19.8.5.4 Activating the modification of the offset data for rotatable tools (CUTMOD, CUTMODK): Further information

Further information

Reading modified offset data

The modified offset data is provided in the following system variables and OPI variables:

Meaning	System variable	OPI variable
Cutting edge position	$P_AD[2]	cuttEdgeParam2
Holder angle	$P_AD[10]	cuttEdgeParam10

Meaning	System variable	OPI variable
Cut direction	$P_AD[11]	cuttEdgeParam11
Clearance angle	$P_AD[24]	cuttEdgeParam24

The data is always modified with respect to the corresponding tool parameters ($TC_DP2[..., ...] etc.) when the "Modification of the offset data for rotatable tools" function was activated with the `CUTMOD` or `CUTMODK` command and the tool was rotated by an orientable tool carrier or a suitable orientation transformation.

Further function-relevant system variables

System variable	Meaning
$P_CUTMOD_ANG / $AC_CUTMOD_ANG	Returns the angle through which a tool was rotated in the active machining plane and the modified cutting edge data available for the CUTMOD and CUTMODK functions.
$P_CUTMOD / $AC_CUTMOD	Reads the currently valid value that was last programmed with the CUTMOD command (number of the tool carrier for which the modification of the offset data should be activated). If the last programmed value was `CUTMOD = -2` (activation with the currently active orientable tool carrier), then the value "-2" is not returned in the system variable, but rather the number of the orientable tool carrier active at the time of programming.
$P_CUTMODK / $AC_CUTMODK	Reads the name of the transformation under which the currently valid data set for the "Modification of the offset data" was created.
$P_CUT_INV / $AC_CUT_INV	Supplies the value TRUE if the tool is rotated so that the spindle direction of rotation must be inverted. To do this, the following four conditions must be fulfilled in the block to which the read operations refer: 1. If a turning or grinding tool is active (tool types 400 to 599 and / or SD42950 $SC_TOOL_LENGTH_TYPE = 2). 2. The modification of the offset data was activated with the `CUTMOD` or `CUTMODK` command. 3. An orientable tool carrier or an orientation transformation defined with kinematic chains is active, which was selected with the CUTMOD or CUTMODK command. 4. The tool is rotated by the orientable tool carrier or the kinematic orientation transformation so that the resulting normal of the tool cutting edge is rotated with respect to the initial position by more than 90° (typically 180°). If at least one of the specified four conditions is not fulfilled, the variable returns the value FALSE. For tools whose cutting edge position is not defined, the value of the variable is always FALSE.

System variable	Meaning	
$P_CUTMOD_ERR	Error state after the last call of the CUTMOD function The CUTMOD function can also be called implicitly for a tool change. At a reset, the variable is reset to zero. It is reset at every tool change and, if required, rewritten. The variable is bit-coded. The bits have the following meanings:	
	Bit 0:	No valid cut direction is defined for the active tool.
	Bit 1:	The cutting edge angle (clearance angle and holder angle) of the active tool are both zero.
	Bit 2:	The clearance angle of the active tool has an impermissible value (< 0° or > 180°).
	Bit 3:	The holder angle of the active tool has an impermissible value (< 0° or > 90°).
	Bit 4:	The plate angle of the active tool has an impermissible value (< 0° or > 90°).
	Bit 5:	The cutting edge position - holder angle combination of the active tool is not permitted (the holder angle must be ≤ 90° for cutting edge position 1 to 4; for cutting edge positions 5 to 8 it must be ≥ 90°).
	Bit 6:	Illegal rotation of the active tool. The tool was rotated out of the active machining plane by ± 90° (with a tolerance of about 1°). The cutting edge position is therefore no longer defined in the machining plane.
	Bit 7:	The cutting plate is not in the machining plane and the angle between the cutting plate and the machining plane exceeds the upper limit specified with the setting data SD42998 $SC_CUTMOD_PLANE_TOL.
	Bit 8:	The cutting plate is not in the machining plane. Angle α is greater than 1°. Angle α is the angle of rotation around the coordinate axis which is perpendicular to the axis of rotation of angle β as well as to the axis of rotation of angle γ (the X axis for G18).

$P_...: Preprocessing variables

$AC_...: Main run variables

All main run variables can be read in synchronized actions. A read access operation from the preprocessing generates a preprocessing stop.

Plane change

To determine the modified cutting edge position, cutting direction and holder or clearance angle, the evaluation of the cutting edge in the active plane (G17 - G19) is decisive.

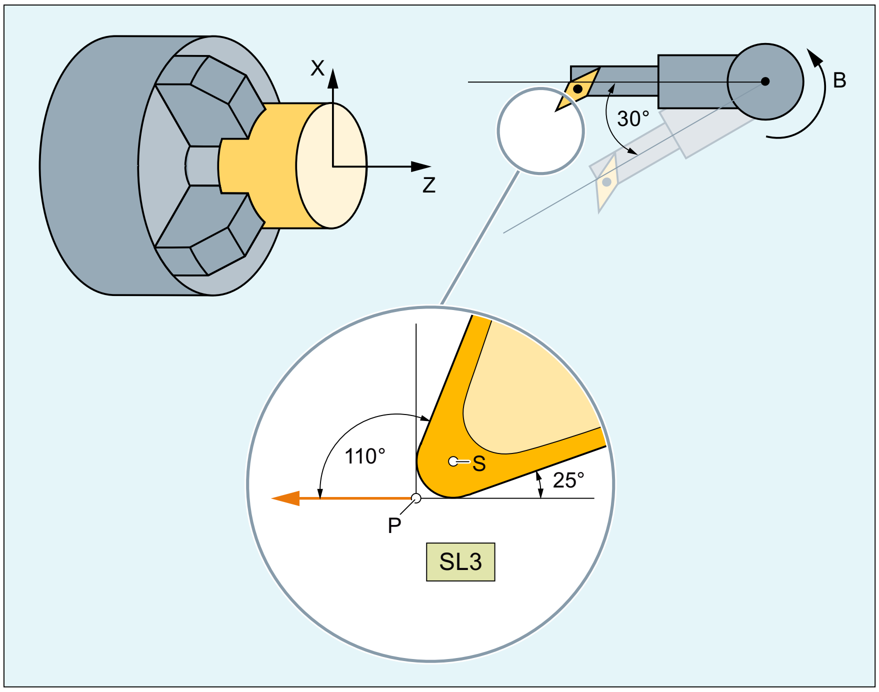
However, if setting data SD42940 $SC_TOOL_LENGTH_CONST (change of the tool length component when selecting the plane) has a valid non-zero value (plus or minus 17, 18 or 19), its contents define the plane in which the relevant quantities are evaluated.

This priority rule of the setting data over the G code can be deactivated by setting bit 18 of the machine data $MC_TOOL_PARAMETER_DEF_MASK. This means that when this bit is set, the plane defined with the G command of group 6 is still valid.

Effectiveness of the modified cutting data

The modified cutting edge position and the modified cutting edge reference point are immediately effective when programming, even for a tool that is already active. A tool does not have to be re-selected for this purpose.

19.8.6 Example



S:	Cutting edge center point
P:	Cutting edge reference point
SL:	Cutting edge position

Figure 19-54 Tool with cutting edge position 3 and an orientable tool carrier that can rotate the tool around the B axis.

Program code	Comment
N10 $TC_DP1[1,1]=500	
N20 $TC_DP2[1,1]=3	;Cutting edge position
N30 $TC_DP3[1,1]=12	
N40 $TC_DP4[1,1]=1	
N50 $TC_DP6[1,1]=6	
N60 $TC_DP10[1,1]=110	; Holder angle
N70 $TC_DP11[1,1]=3	; Cut direction
N80 $TC_DP24[1,1]=25	; Clearance angle
N90 $TC_CARR7[2]=0 $TC_CARR8[2]=1 $TC_CARR9[2]=0	; B axis
N100 $TC_CARR10[2]=0 $TC_CARR11[2]=0 $TC_CARR12[2]=1	; C axis

```
Program code                                          Comment
N110 $TC_CARR13[2]=0
N120 $TC_CARR14[2]=0
N130 $TC_CARR21[2]=X
N140 $TC_CARR22[2]=X
N150 $TC_CARR23[2]="M"

N160 TCOABS CUTMOD=0
N170 G18 T1 D1 TCARR=2                                ; X         Y          Z
N180 X0 Y0 Z0 F10000                                  ; 12.000   0.000     1.000

N190 $TC_CARR13[2]=30
N200 TCARR=2
N210 X0 Y0 Z0                                         ; 10.892   0.000    -5.134
N220 G42 Z-10                                         ;  8.696   0.000   -17.330
N230 Z-20                                             ;  8.696   0.000   -21.330
N240 X10                                              ; 12.696   0.000   -21.330
N250 G40 X20 Z0                                       ; 30.892   0.000    -5.134

N260 CUTMOD=2 X0 Y0 Z0                                ;  8.696   0.000    -7.330
N270 G42 Z-10                                         ;  8.696   0.000   -17.330
N280 Z-20                                             ;  8.696   0.000   -21.330
N290 X10                                              ; 12.696   0.000   -21.330
N300 G40 X20 Z0                                       ; 28.696   0.000    -7.330


N310 M30
```

The numerical values in the comments specify the end of block positions in the machine coordinates (MCS) in the sequence X → Y → Z.

Explanations

In block `N180`, initially the tool is selected for `CUTMOD=0` and non-rotated tool holders that can be orientated. As all offset vectors of the tool holder that can be orientated are 0, the position that corresponds to the tool lengths specified in `$TC_DP3[1,1]` and `$TC_DP4[1,1]` is approached.

The tool holder that can be orientated with a rotation of 30° around the B axis is activated in block `N200`. As the cutting edge position is not modified due to `CUTMOD=0`, the old cutting edge reference point is decisive just as before. This is the reason why in block `N210` the position is approached, which keeps the old tool nose reference point at the zero (i.e. the vector (1, 12) is rotated through 30° in the Z/X plane).

In block `N260`, contrary to block `N200, CUTMOD=2` is effective. As a result of the tool holder rotation that can be orientated, the modified cutting edge position becomes 8. Deviating axis positions also result from this.

The tool radius compensation (TRC) is activated in blocks `N220` and/or `N270`. The different cutting edge position in both program sections has no effect on the end positions of the blocks in which the TRC is active; the corresponding positions are therefore identical. The different cutting edge positions only become effective again in the deselect blocks `N260` and/or `N300`.

19.9 Incrementally programmed compensation values

19.9.1 G91 extension

Requirements

Incremental programming with `G91` is defined such that the compensation value is traversed additively to the incrementally programmed value when a tool compensation is selected.

Applications

For applications such as scratching, it is necessary only to traverse the path programmed in the incremental coordinates. The activated tool compensation is not traversed.

Sequence

Selection of a tool compensation with incremental programming

- Scratch workpiece with tool tip.
- Save the actual position in the basic frame (set actual value) after reducing it by the tool compensation.
- Traverse incrementally from the zero position.

Activation

It is possible to set whether a changed tool length is traversed with `FRAME` and incremental programming of an axis, or whether only the programmed path is traversed with the setting data:

SD42442 $SC_TOOL_OFFSET_INCR_PROG (tool length compensations)

Zero offset / frames G91

It is possible to set whether a zero offset is traversed as standard with value = 1 with `FRAME` and incremental programming of an axis, or whether only the programmed path is traversed with value = 0 with the setting data:

SD42440 $SC_FRAME_OFFSET_INCR_PROG (zero offset in frames)

For further information, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 705)".

Supplementary condition

If the behavior is set such that the offset remains active even after the end of the program and `RESET`
MD20110 $MC_RESET_MODE_MASK, bit6=1 (specification of the controller initial setting after reset / TP end)
and if an incremental path is programmed in the first part program block, the compensation is always traversed additively to the programmed path.

Note

With this configuration, part programs must always begin with absolute programming.

19.9.2 Traversing in the direction of tool orientation (MOVT)

Typical application

On machines with toolholders with orientation capability, traversing should take place in the tool direction (typically, when drilling) without activating a frame (e.g., using `TOFRAME` or `TOROT`), on which one of the axes points in the direction of the tool.

This is also true of machines on which a frame defining the oblique plane is active during oblique machining operations, but the tool cannot be set exactly perpendicular because an indexed toolholder (Hirth tooth system) is restricting the setting of the tool orientation.

In these cases it is then necessary - contrary to the motion actually requested perpendicular to the plane - to drill in the tool direction, as the drill would otherwise not be guided in the direction of its longitudinal axis, which, among other things, would lead to breaking of the drill.

MOVT

The end point of such a motion is programmed with `MOVT= ...`. The programmed value is effective incrementally in the tool direction as standard. The positive direction is defined from the tool tip to the toolholder. The content of `MOVT` is thus generally negative for the infeed motion (when drilling), and positive for the retraction motion. This corresponds to the situation with normal paraxial machining, e.g., with `G91Z ...`.

If the motion is programmed in the form `MOVT=AC( ...)`, `MOVT` functions absolutely. In this case a plane is defined, which runs through the current zero point, and whose surface normal vector is parallel to the tool orientation. `MOVT` then gives the position relative to this plane:

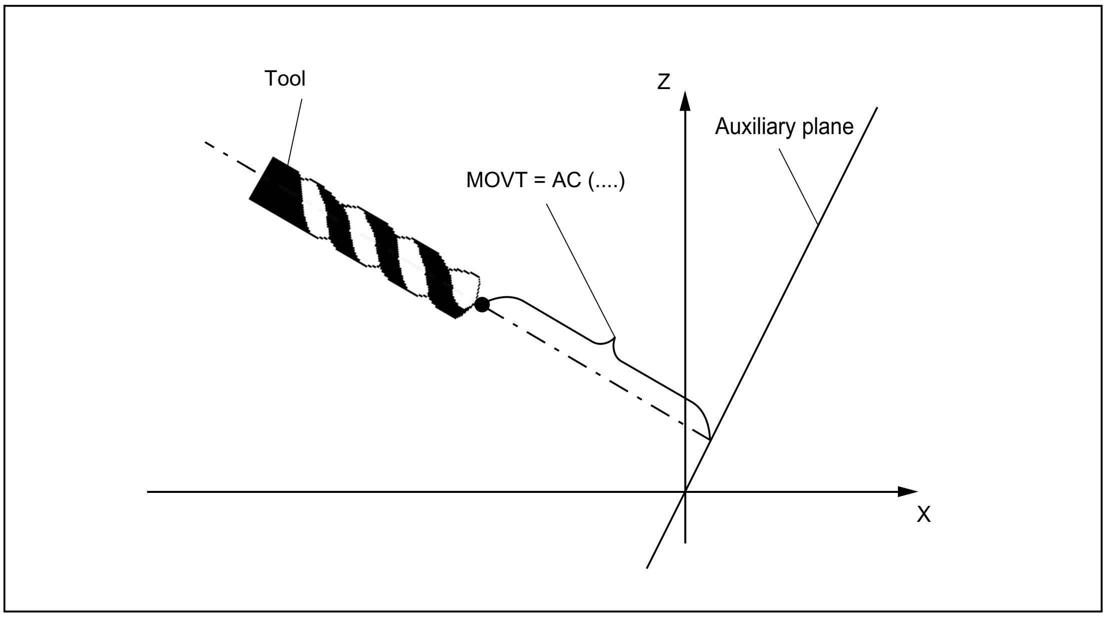


Figure 19-55 Definition of the position for absolute programming of a motion in tool direction

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).

19.10 Assignment of tool length components to geometry axes

19.10.1 Assignment according to tool type and working plane.

The values of the tool parameters length 1 ... 3 are stored in the system variables $TC_DP3 ... $TC_DP5 (see Chapter "Tool cutting edge (Page 1475)"). The assignment to the geometry axes and therefore the resultant cutting direction of the tool length components depends on the tool type ($TC_DP1) and the active machining plane (G17/G18/G19).

Table 19-1 Turning/grinding tools ($TC_DP1 = 400 … 599)

Working plane	Length 1	Length 2	Length 3
G17 (X/Y)	Y	X	Z
G18 (Z/X)	X	Z	Y
G19 (Y/Z)	Z	Y	X

Table 19-2 Milling tools / special tools ($TC_DP1 <> 400 … 599)

Working plane	Length 1	Length 2	Length 3
G17 (X/Y)	Z	Y	X
G18 (Z/X)	Y	X	Z
G19 (Y/Z)	X	Z	Y

19.10.2 Assignment when changing plane

The assignment of tool length components (length, wear, and tool base dimension) to the geometry axes does **not** change when the machining plane is changed if the following setting data is set to not equal to zero:

SD42940 $SC_TOOL_LENGTH_CONST <> 0

The assignment of the tool length components to the geometry axes is then derived from the ones and tens position of the setting data as shown in the following tables

Table 19-3 Turning / grinding tools ($TC_DP1 = 400 … 599)

SD42940	Assignment of tool length components to geometry axes		
	Length L1	Length L2	Length L3
= x17	Y	X	Z
= x18	X	Z	Y
= x19	Z	Y	X
= -x17	X	Y	Z
= -x18	Z	X	Y
= -x19	Y	Z	X

Table 19-4 Milling / special tools ($TC_DP1 <> 400 … 599)

SD42940	Assignment of tool length components to geometry axes		
	Length L1	Length L2	Length L3
= x17	Z	Y	X
= x18	Y	X	Z
= x19	X	Z	Y
= -x17	Z	X	Y
= -x18	Y	Z	X
= -x19	X	Y	Z

Each value not equal to 0 and not equal to one of the six listed values is evaluated as value "17" (for milling/special tools) or "18" (for turning/grinding tools)..

Note

Assignment of tool orientation components

The assignment of the tool orientation components is not influenced by SD42940 $SC_TOOL_LENGTH_CONST.

If necessary, the following setting data must be adapted:

- SD42954 $SC_TOOL_ORI_CONST_M
- SD42956 $SC_TOOL_ORI_CONST_T

See Chapter "Tool orientation for plane change (Page 1606)".

19.10.3 Assignment independent of tool type

With the following setting data, the assignment of the tool length components (length, wear and tool base dimension) to the geometry axes that is independent of the actual tool type ($TC_DP1) can be defined:

SD42950 $SC_TOOL_LENGTH_TYPE = <value>

<value>	Assignment of the tool length components
0 (default value)	In accordance with the specification in the system variable $TC_DP1[...]. A distinction is made between the following tool types: • $TC_DP1[...] == 400 ... 599 ⇒ turning/grinding tools • $TC_DP1[...] <> 400 ... 599 ⇒ milling/special tools
1	Always as for milling/special tools.
2	Always as for turning/grinding tools.

<value>	Assignment of the tool length components	
3	Activates setting data SD42942 $SC_TOOL_LENGTH_CONST_T. With this setting, it is possible to define the assignment of the tool length components that is effective on a machining plane change (Page 1604) separately for milling/special tools and turning/grinding tools:	
	SD42940 $SC_TOOL_LENGTH_CONST	Assignment for milling/special tools
	SD42942 $SC_TOOL_LENGTH_CONST_T	Assignment for turning/grinding tools
	Example: SD42940 $SC_TOOL_LENGTH_CONST = 17 SD42942 $SC_TOOL_LENGTH_CONST_T = 18 Tool length compensations for milling/special tools take effect in plane G17; length compensations for turning/grinding tools take effect in plane G18.	
> 3	As for SD42950 = 0.	

19.11 Paraxial tool orientation

19.11.1 Basic tool orientation

The basic tool orientation results from the active machining plane:

Working plane	Tool orientation (basic setting)
G17 (X/Y)	Parallel to Z
G18 (Z/X)	Parallel to Y
G19 (Y/Z)	Parallel to X

19.11.2 Tool orientation for plane change

A setting in the setting data determines how the tool orientation changes when the machining plane is changed.

- SD42954 $SC_TOOL_ORI_CONST_M (for milling/special tools)
- SD42956 $SC_TOOL_ORI_CONST_T (for turning/grinding tools)

Standard behavior

With the default setting (SD42954 and SD42956 = 0), the tool orientation changes with a plane change as follows:

Plane change	Change in tool orientation
G17 → G18	1. Rotation by -90° about the Z coordinate 2. Rotation by -90° about the X coordinate
G18 → G19	
G19 → G17	
G18 → G17	1. Rotation by 90° about the X coordinate 2. Rotation by 90° about the Z coordinate
G19 → G18	
G17 → G19	

Constant tool orientation for a plane change

If SD42954 or SD42956 are not equal to zero, a right-handed orthogonal tool coordinate system is defined, which remains unchanged if the machining plane is changed (G17-G19). The orientation coordinate system is determined by the orientation vector and an orientation normal vector perpendicular to it. The basic orientation is defined by the ones and tens position of the setting data:

Table 19-5 Milling / special tools ($TC_DP1 <> 400 … 599)

SD42954	Coordinate system of the tool orientation	
	Orientation vector	Orientation normal vector
= xx17	(0, 0, 1)	(0, 1, 0)
= xx18	(0, 1, 0)	(1, 0, 0)
= xx19	(1, 0, 0)	(0, 0, 1)

Table 19-6 Turning / grinding tools ($TC_DP1 = 400 … 599)

SD42956	Coordinate system of the tool orientation	
	Orientation vector	Orientation normal vector
= xx17	(0, 0, 1)	(0, 1, 0)
= xx18	(0, 1, 0)	(1, 0, 0)
= xx19	(1, 0, 0)	(0, 0, 1)

Each value not equal to 0 and not equal to one of the three listed values is evaluated as value "17" (for milling/special tools) or "18" (for turning/grinding tools).

References:
A detailed description of SD42954 and SD42956 is given in the System variables List Manual.

Note

In the case of tools whose orientation is defined by cutting edge data ($TC_DPV…), SD42954 or SD42956 is usually ignored (see Chapter "Parameterizable basic tool orientation (Page 1608)").

19.12 Parameterizable basic tool orientation

19.12.1 Function

Note

Tool T and cutting edge D

In the following, the syntax [...] represents [<t>, <d>] in relation to the system variables of the basic tool orientation. In this regard, <t> designates the number of the tool T=<t> and <d> the number of the tool cutting edge D=<d>.

The "Parameterizable basic tool orientation" function allows each tool cutting edge to be assigned an individual initial orientation with the following system variables:

- Selection of a predefined orientation vector
  - $TC_DPV[...]: Orientation vector (values 1 ... 6)
- Definition of an orientation vector.
  - $TC_DPV3[...]: L1 component of the orientation vector
  - $TC_DPV4[...]: L2 component of the orientation vector
  - $TC_DPV5[...]: L3 component of the orientation vector
- Definition of a normal vector for the orientation vector
  - $TC_DPVN3[...]: L1 component of the normal vector
  - $TC_DPVN4[...]: L2 component of the normal vector
  - $TC_DPVN5[...]: L3 component of the normal vector

If the system variable $TC_DPV[...] is equal to zero, the three system variables $TC_DPV3 - 5[...] define the direction vector of the basic tool orientation. Additionally, the three system variables $TC_DPVN3 - 5[...] can define the orientation of the tool in the plane perpendicular to the orientation vector. The magnitudes of the orientation vectors are immaterial.

19.12.2 Commissioning

19.12.2.1 Activation

The "Parameterizable basic tool orientation" is activated via the following machine data:

MD18114 $MN_MM_ENABLE_TOOL_ORIENTATION = <value>

<value>	Meaning
0	The function "Parameterizable basic tool orientation" is not active.
1	Activation of system variables: • Function selection: $TC_DPV[...] With the system variable $TC_DPV[...] = 1, 2, ... 6, one of six predefined basic tool orientations can be assigned for each tool cutting edge D=<d> of a tool T=<t>.
2	Activation of system variables: • Function selection: $TC_DPV[...] • Orientation vector: $TC_DPV3[...], $TC_DPV4[...] and $TC_DPV5[...] With the system variables $TC_DPV[...] = 0 and $TC_DPV3[...], $TC_DPV4[...] and $TC_DPV5[...], any orientation vector can be assigned for each tool cutting edge D=<d> of a tool T=<t>.
3	Activation of system variables: • Function selection: $TC_DPV[...] • Orientation vector: $TC_DPV3[...], $TC_DPV4[...] and $TC_DPV5[...] • Normal vector for the orientation vector: $TC_DPVN3[...], $TC_DPVN4[...] and $TC_DPVN5[...] With the setting $TC_DPV[...] = 0, any orientation vector can be assigned for each tool cutting edge D=<d> of a tool T=<t> with the system variables $TC_DPV3[...], $TC_DPV4[...] and $TC_DPV5[...]. Additionally, a normal vector can be defined for the orientation vector with the system variables $TC_DPVN3[...], $TC_DPVN4[...] and $TC_DPVN5[...].

19.12.2.2 Parameterization

Assignment of the system variables $TC_DPVx[...]

A differentiation is made between the following tool types in accordance with the specification in system variable $TC_DP1[...]:

- $TC_DP1[...] == 400 ... 599 ⇒ turning/grinding tools
- $TC_DP1[...] <> 400 ... 599 ⇒ milling/special tools

The basic tool orientation programmed in $TC_DPVx[...] is assigned to the coordinate axes depending on the tool type.

The tool type may be changed over by the setting in setting data SD42950 $SC_TOOL_LENGTH_TYPE (as described in Chapter "Assignment independent of tool type (Page 1605)" for tool length components).

If no other orientation is defined in setting data SD42954 $SC_TOOL_ORI_CONST_M and SD42956 $SC_TOOL_ORI_CONST_T (Page 1606), the standard case (G17 for milling/special tools, G18 for turning/grinding tools) applies for assignment of the system variables $TC_DPVx[...]:

Tool type	TC_DPV3[…]	TC_DPV4[…]	TC_DPV5[…]
Turning/grinding tools	X	Z	Y
Milling/special tools	Z	Y	X

Note

SD42954 / SD42956

The assignment of the system variables $TC_DPVx[...] can only be changed with SD42954 $SC_TOOL_ORI_CONST_M and SD42956 $SC_TOOL_ORI_CONST_T if the 1000s position is equal to "1"

Irrespective of the value of setting data SD42950 $SC_TOOL_LENGTH_TYPE, SD42954 is only effective for a tool for which parameter $TC_DP1[...] defines a milling tool. Analogously, SD42956 is only effective for a tool for which parameter $TC_DP1[...] defines a turning/grinding tool.

19.12.3 Programming

Using system variables $TC_DPV3 - 5[...] or $TC_DPV[...], a separate basic orientation can be assigned to every tool cutting edge.

Setting options

Basically, the following setting options are available

- **$TC_DPV[...] == 0 AND $TC_DPV3 - 5[...] == 0**
  The vector for the basic tool orientation results from the active machining plane:
  - G17: Z coordinate
  - G18: Y coordinate
  - G19: X coordinate

  See also "Paraxial tool orientation (Page 1606)".
- **$TC_DPV[...] == 0 AND $TC_DPV3 - 5[...] <> 0**
  The vector for the basic tool orientation is prescribed by $TC_DPV3 - 5[...]:
  - `$TC_DPV3[...] = <value in L1 direction>`
  - `$TC_DPV4[...] = <value in L2 direction>`
  - `$TC_DPV5[...] = <value in L3 direction>`

  Example:
  Basic tool orientation points in the direction of the bisectors in the L1-L3 plane, i.e. for a milling tool and active plane G17, in the direction of the bisectors in the ZX plane.
  ```
  $TC_DPV[1,1]  = 0
  $TC_DPV3[1,1] = 1.0
  $TC_DPV4[1,1] = 0.0
  $TC_DPV5[1,1] = 1.0
  ```
- **$TC_DPV[...] == 1, 2, ... 6**
  The vector for the basic tool orientation is prescribed by $TC_DPV[...].
  The following tables show which basic tool orientations are predefined and can be selected via $TC_DPV[...].

Selection of a predefined orientation vector

```
$TC_DPV[...] = <value>
```

Table 19-7 Turning / grinding tools ($TC_DP1 = 400 … 599)

<value>	Meaning		
0	$TC_DPV3[...]	$TC_DPV5[...]	$TC_DPV4[...]
1	0	1	0
2	1	0	0
3	0	0	1
4	0	-1	0
5	-1	0	0
6	0	0	-1

Table 19-8 Milling / special tools ($TC_DP1 <> 400 … 599)

<value>	Meaning		
0	$TC_DPV5[...]	$TC_DPV4[...]	$TC_DPV3[...]
1	0	0	1
2	0	1	0
3	1	0	0
4	0	0	-1
5	0	-1	0
6	-1	0	0

Examples

Turning/grinding tools:

`$TC_DPV[...] = 2`	Corresponds to	`$TC_DPV3[...] = 1`
		`$TC_DPV4[...] = 0`
		`$TC_DPV5[...] = 0`

Milling/special tools:

`$TC_DPV[...] = 3`	Corresponds to	`$TC_DPV3[...] = 0`
		`$TC_DPV4[...] = 0`
		`$TC_DPV5[...] = 1`

19.12.4 Examples

A milling tool is defined with length L1=10 whose basic tool orientation is in the bisector of the X-Z plane.

Example 1

Program code	Comment
`$SC_TOOL_LENGTH_TYPE=2`	`; assignment to the coordinate axes as for turning/ grinding tools.`
`$SC_TOOL_ORI_CONST_M=1019`	`; Working plane G19`
`N10 $TC_DP1[1,1]=120`	`; Tool type: Milling tool`
`N20 $TC_DP3[1,1]=10`	`; Length compensation vector: L1=10`
`N30 $TC_DPV[1,1]=0`	`; Basic tool orientation via $TC_DPV3-5`
`N40 $TC_DPV3[1,1]=1`	`; Z coordinate`
`N50 $TC_DPV4[1,1]=0`	`; Y coordinate`
`N60 $TC_DPV5[1,1]=1`	`; X coordinate`
`N70 TRAFOON(...)`	`; Activating a transformation.`
`N80 G17 F1000 X0 Y0 Z0 T1 D1`	`; Activate tool and approach zero position.`
	`; Plane selection G18 in block N80 is only relevant for length compensations.`
`...`	

Example 2

Program code	Comment
`N10 $TC_DP1[1,1]=120`	`; Tool type: Milling tool`
`N20 $TC_DP3[1,1]=10`	`; Length compensation vector: L1=10`
`N30 $TC_DPV[1,1]=0`	`; Basic tool orientation via $TC_DPV3-5`
`N40 $TC_DPV3[1,1]=1`	`; Z coordinate`
`N50 $TC_DPV4[1,1]=0`	`; Y coordinate`
`N60 $TC_DPV5[1,1]=1`	`; X coordinate`
`N70 TRAFOON(...)`	`; Activating a transformation.`
`N80 G17 F1000 X0 Y0 Z0 T1 D1`	`; Activate tool and approach zero position.`
	`; Due to tool length L1=10 =>`
	`; Machine positions: X=0, Y=0, Z=10`
`N90 MOVT=10`	`; Incr. Traversing movement in the tool direction.`
	`; Resulting axis positions: X=7.071 Y=0 Z=17.071`
`N100 M30`	

19.13 Special handling of tool compensations

19.13.1 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 bis $TC_DP20
- Sum offset, consisting of:
  - Wear values: $SCPX3 to $SCPX11
  - Setup values: $ECPX3 to $ECPX11

You will find detailed information about sum and tool offsets in:

References:
Function Manual Tool Management

Programming Manual. Fundamentals; Tool Offsets

Required setting data

- SD42900 $SC_MIRROR_TOOL_LENGTH (mirroring of tool length components and components of the tool base dimension) (Page 1614)
- SD42910 $SC_MIRROR_TOOL_WEAR (mirroring of wear values of tool length components) (Page 1614)
- SD42920 $SC_WEAR_SIGN_CUTPOS (sign evaluation of the wear components) (Page 1615)
- SD42930 $SC_WEAR_SIGN (inverts the sign of the wear dimensions) (Page 1615)
- SD42935 $SC_WEAR_TRANSFORM (transformation of wear values) (Page 1632)
- SD42940 $SC_TOOL_LENGTH_CONST (allocation of the tool length components to the geometry axes) (Page 1604)

- SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type) (Page 1605)
- SD42960 $SC_TOOL_TEMP_COMP (tool length offsets) (Page 1617)

19.13.2 Mirroring tool lengths

Activation

Tool length mirroring is activated via the setting data:

SD42900 $SC_MIRROR_TOOL_LENGTH <> 0 (TRUE) (Sign change tool length when mirroring)

Function

The following components are mirrored by inverting the sign:

- Tool lengths: $TC_DP3, $TC_DP4, $TC_DP5
- Tool base dimensions: $TC_DP21, $TC_DP22, $TC_DP23

Mirroring is performed for all tool base dimensions whose associated axes are mirrored. Wear values are **not** mirrored.

Mirror wear values

The following setting data should be set in order to mirror the wear values:

SD42910 $SC_MIRROR_TOOL_WEAR <> 0 (Sign change tool wear when mirroring)

Inverting the sign mirrors the wear values of the tool length components whose associated axes are mirrored.

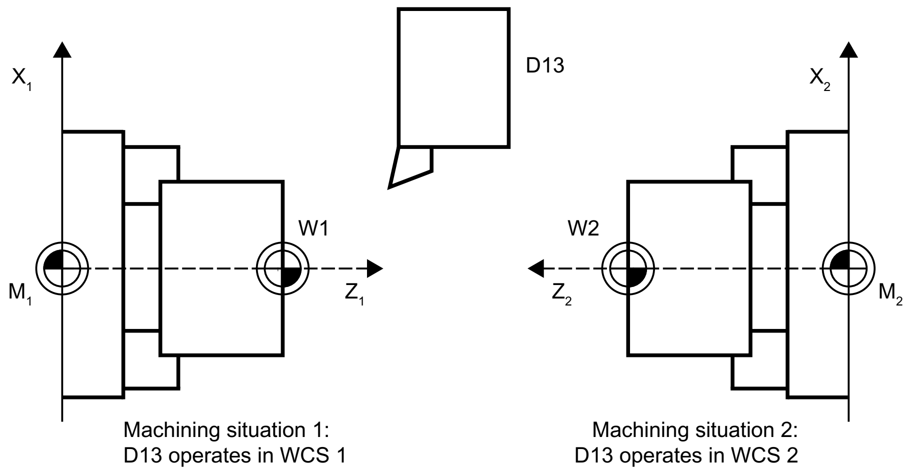


Figure 19-56 Application example: Double-spindle turning machine

19.13.3 Mirroring wear lengths

Activation

Wear length mirroring is activated by:

SD42920 $SC_WEAR_SIGN_CUTPOS <> 0 (TRUE) (Sign of wear for tools with cutting edge systems)

Function

Length of cutting edge	Length 1	Length 2
1	---	---
2	---	Inverted
3	Inverted	Inverted
4	Inverted	---
5	---	---
6	---	---
7	---	Inverted
8	Inverted	---
9	---	---

In the case of tool types without a relevant cutting edge position, the wear length is not mirrored.

Note

The mirroring (sign inversion) in one or more components can cancel itself through a simultaneous activation of the functions:

Tool length-mirroring (SD42900 <> 0)

And:

Tool length-mirroring (SD42920 <> 0)

SD42930 $SC_WEAR_SIGN

Setting data not equal to zero:

Inverts the sign of all wear dimensions. This affects both the tool length and other variables such as tool radius, rounding radius, etc.

Entering a positive wear dimension makes the tool "shorter" and "thinner".

Activation of modified setting data

When the setting data described above are modified, the tool components are not reevaluated until the next time a tool edge is selected. If a tool is already active and the data of this tool are to be reevaluated, the tool must be selected again.

Example:

```
N10 $SC_WEAR_SIGN = 0               ; No sign inversion of the wear values
N20 $TC_DP1[1,1] = 120              ; End mill
N30 $TC_DP6[1,1] = 100              ; Tool radius 100 mm
N40 $TC_DP15[1,1] = 1               ; Wear dimension of tool radius 1 mm, resulting
                                      tool radius 101 mm
N100 T1 D1 G41 X150 Y20
....
N150 G40 X300N10
....
N200 $SC_WEAR_SIGN = 1              ; Sign inversion for all wear values; the new ra-
                                      dius 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.

19.13.4 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.

19.13.5 Tool length offsets in tool direction

Temperature compensation in real time

On 5-axis machines with a moving tool, temperature fluctuations can occur in the machining heads. These can result directly in expansion fluctuations which are transmitted to the tool spindle in the form of linear expansion. A typical case on 5-axis heads, for example, is thermal expansion in the direction of the longitudinal spindle axis.

It is possible to compensate this thermal expansion even when the tool is orientated by assigning the temperature compensation values to the tool rather than to the machine axes. In this way, linear expansion fluctuations can be compensated even when the tool orientation changes.

Using the orientation transformation whose direction is determined by the current tool orientation, it is possible to overlay motions in real time and rotate them simultaneously. At the same time, the compensation values are adjusted continuously in the tool coordinate system.

The temperature compensation only becomes effective if the axis to be compensated is really referenced.

Activation

The temperature compensation in the tool direction is activated by setting the following machine data to a value **not equal to zero**.

MD20390 $MC_TOOL_TEMP_COMP_ON (activation of temperature compensation for tool length)

In addition, bit 2 must be set for each affected channel axis in the machine data:

MD32750 $MA_TEMP_COMP_TYPE [<axis index>] (temperature compensation type)

This can be more than three axes in cases where more than three channel axes in succession can be temporarily assigned to geometry axes as a result of geometry axis replacement of transformation switchover. If this bit is not set for a particular channel axis, the compensation value cannot be applied in the axis. This does not have any effect on other axes. In this case, an alarm is not output.

Applicability

The temperature compensation in the tool direction is only effective with generic 5-axis transformations with:

- Transformation type 24
  Two axes rotate the tool
- Transformation type 56
  One axis rotates the tool, the other axis rotates the workpiece without temperature compensation

In generic 5-axis transformation with:

- Transformation type 40
  The tool orientation is constant with a rotary workpiece, which means that the motion of the rotary axes on the machine does not affect the temperature compensation direction.

Temperature compensation in the tool direction also works in conjunction with orientation transformations (not generic 5-axis transformations) with:

- Transformation type 64 to 69
  Rotating linear axis

Note

Temperature compensation can be activated with all other types of transformation. It is not affected by a change in tool orientation. The axes move as if no orientation transformation with temperature compensation were active.

Limit values

The compensation values are restricted to the maximum values by the machine data:

MD20392 $MC_TOOL_TEMP_COMP_LIMIT[0...2] (maximum temperature compensation for tool length)

The limit value default setting is 1 mm. If a temperature compensation value higher than this limit is specified, it will be limited without an alarm.

SD42960

The three temperature compensation values together form a compensation vector and are contained in setting data:

SD42960 $SC_TOOL_TEMP_COMP[0...2] (temperature compensation with reference to tools)

The setting data is user-defined, e.g. using synchronized actions or from the PLC. The compensation values can, therefore, also be used for other compensation purposes.

In the initial state or when orientation transformation is deactivated, all three compensation values apply in the direction of the three geometry axes (in the typical order X, Y, Z). The assignment of components to geometry axes is independent of the tool type (turning, milling or grinding tools) and the selected machining plane `G17` to `G19`. Changes to the setting data values take effect immediately.

Toolholder with orientation capability

If a toolholder with orientation capability is active, the temperature compensation vector is rotated simultaneously to any change in orientation. This applies independently of any active orientation transformation.

If a toolholder with orientation capability is active in conjunction with a generic 5-axis transformation or a transformation with rotating linear axis, the temperature compensation vector is subjected to both rotations.

Note

While transformations with rotating linear axes take changes in the tool vector (length) into account, they **ignore** its change in orientation, which can be effected by a toolholder with orientation capability.

Temperature compensation values immediately follow any applied change in orientation. This applies in particular when an orientation transformation is activated or deactivated.

The same is true when the assignment between geometry axes and channel axes is changed. The temperature compensation value for an axis is reduced to zero (interpolatively), for example, when it ceases to be a geometry axis after a transformation change. Conversely, any temperature compensation value for an axis which changes over to geometry axis status is applied immediately.

Examples

Temperature compensation in tool direction

Example of a 5-axis machine with rotating tool on which the tool can be rotated around the C and B axes.

In its initial state, the tool is parallel to the Z axis. If the B axis is rotated through 90 degrees, the tool points in the X direction.

Therefore, a temperature compensation value in the following setting data is also effective in the direction of the machine X axis if transformation is active:

SD42960 $SC_TOOL_TEMP_COMP[2] (temperature compensation with reference to tools)

If the transformation is deactivated with the tool in this direction, the tool orientation is, by definition, parallel again to the Z axis and thus different to its actual orientation. The temperature offset in the X axis direction is therefore reduced to zero and reapplied simultaneously in the Z direction.

Example of a 5-axis machine with rotating tool (transformation type 24). The relevant machine data is listed below:

- The first rotary axis rotates around Z, C-axis
- The second rotary axis rotates around Y, B-axis

The essential machine data is shown in the table below:

Machine data	Value	Remark
MD20390 $MC_TOOL_TEMP_COMP_ON	= TRUE	Temperature compensation active
MD32750 $MA_TEMP_COMP_TYPE[AX1]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX2]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX3]	= 4	Compensation in tool direction
		Assignment of transformation type 24
MD24100 $MC_TRAFO_TYPE_1	= 24	Transformer type 24 in 1st channel
MD24110 $MC_TRAFO_AXES_IN_1[0]	= 1	First axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[1]	= 2	Second axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[2]	= 3	Third axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[3]	= 5	Fifth axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[4]	= 4	Fourth axis of the transformation

Machine data	Value	Remark
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]	= 1	Geometry axis for channel axis 1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]	= 2	Geometry axis for channel axis 2
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]	= 3	Geometry axis for channel axis 3
MD24570 $MC_TRAFO5_AXIS1_1[0]	= 0.0	
MD24570 $MC_TRAFO5_AXIS1_1[1]	= 0.0	Direction
MD24570 $MC_TRAFO5_AXIS1_1[2]	= 1.0	First rotary axis is parallel to Z
MD24572 $MC_TRAFO5_AXIS1_2[0]	= 0.0	Direction
MD24572 $MC_TRAFO5_AXIS1_2[1]	= 1.0	Second rotary axis is parallel to Y
MD24572 $MC_TRAFO5_AXIS1_2[2]	= 0.0	
MD25574 $MC_TRAFO5_BASE_ORIENT_1[0]	= 0.0	
MD25574 $MC_TRAFO5_BASE_ORIENT_1[1]	= 0.0	Basic tool orientation
MD25574 $MC_TRAFO5_BASE_ORIENT_1[2]	= 1.0	In Z direction

Temperature compensation values in the NC program

The compensation values assigned to axes X and Z are not zero and are applied for temperature compensation with respect to tool length. The machine axis positions reached in each case are specified as comments in the program lines.

```
Program code                          Comment
$SC_TOOL_TEMP_COMP[0] = -0.3          ; Compensation value in X
$SC_TOOL_TEMP_COMP[1] = 0.0
$SC_TOOL_TEMP_COMP[2] = -1.0          ; Compensation value in Z
                                      ; Position setpoints of the machine
                                      axes
N10 G74 X0 Y0 Z0 A0 B0                ; X Y Z
N20 X20 Y20 Z20 F10000                ; 20.30 20.00 21.00
N30 TRAORI()                          ; 20.30 20.00 21.00
N40 X10 Y10 Z10 B90                   ; 11.00 10.00 9.70
N50 TRAFOOF                           ; 10.30 10.00 11.00
N60 X0 Y0 Z0 B0 C0                    ; 0.30 0.00 1.00
N70 M30
```

With the exception of block `N40`, temperature compensation always acts in the original directions, as the tool is pointing in the basic orientation direction. This applies particularly in block `N50`. The tool is actually still pointing in the direction of the X axis because the B axis is still at 90 degrees. However, because the transformation is already deactivated, the applied orientation is parallel to the Z axis again.

Machine data	Value	Remark
MD20390 $MC_TOOL_TEMP_COMP_ON	= TRUE	Temperature compensation active
MD32750 $MA_TEMP_COMP_TYPE[AX1]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX2]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX3]	= 4	Compensation in tool direction

Additional references

For more details on "Temperature compensation" see:

References:
Function Manual Extended Functions; Compensations (K3)

For information on "Generic 5-axis transformations" see:

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

19.13.6 Special characteristics of orientable tool carriers

Setting data SD42900 - SD42950

Setting data SD42900 - SD42950 have no effect on the components of an active tool carrier with orientation capability. The calculation with a tool carrier 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 tool carriers 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 tool carrier 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 tool carriers 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 tool carrier.

19.14 Sum offsets and setup offsets

19.14.1 General information

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 is addressed by a **DL number** (**DL:** Location-dependent; offsets regarding 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 is 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 NC. The operator subsequently only has access to the "sum offset fine" via HMI.

The "sum offset fine" and "sum offset coarse" are added internally in the NC. This value is referred to below as the sum offset.

Note

The function is enabled via the machine data setting:

MD18080 $MN_MM_TOOL_MANAGEMENT_MASK, Bit 8=1 (step-by-step memory reservation for tool management).

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 command of group 56.

19.14.2 Description of function

Sum offsets

Several sum offsets (DL numbers) can be defined per D number. In this way, for example, **workpiece location-dependent** offset values can be determined and assigned to a cutting edge. Sum offsets have the same effect as wear, i.e. they are added to the offset values of the D number. The data is permanently assigned to a D number.

Settings

You can define the following settings via machine data:

- Activate sum offset
- Define maximum number of DL data sets to be created in NC
- Define maximum number 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
  - 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 item 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	Cutting edge position

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	Compensations
Length compensations		
$TC_DP3	$TC_DP12	Length 1
$TC_DP4	$TC_DP13	Length 2
$TC_DP5	$TC_DP14	Length 3
Radius compensation		
$TC_DP6	$TC_DP15	Radius 1
$TC_DP7	$TC_DP16	Radius 2
Other compensations		
$TC_DP8	$TC_DP17	Length 4
$TC_DP9	$TC_DP18	Length 5
$TC_DP10	$TC_DP19	Angle 1
$TC_DP11	$TC_DP20	Angle 2

Tool base dimension or adapter dimension

$TC_DP21	Adapter length 1
$TC_DP22	Adapter length 2
$TC_DP23	Adapter length 3

Technology

System variable	Clearance angle
$TC_DP24	The clearance angle is stored here for ManualTurn; tool type 5xx. Same significance as in standard cycles for turning tools. The tip angle of the drill is stored here for ShopMill; tool type 2xx. Used in standard cycles for turning tools; tool type 5xx. This is the angle at the secondary cutting edge for these tools.
$TC_DP25	The value for the cutting rate is stored here for ManualTurn. A 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 parameters (to which the compensation is added)	**Sum / setup parameters**	**Tool wear parameter**
Length compensations		
$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
Other 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 NC are defined via 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 management data.

19.14.3 Activation

Function

The function must be activated via the machine data:

MD18108 $MN_MM_NUM_SUMCORR (sum offsets in TO area).

System variables $TC_ECPx and $TC_SCPx and setup and sum offsets ("fine") defined via the OPI interface can be activated in the part program.

This is done by programming the language command `DL=`"number".

When a new D number is activated, either a new DL number is programmed, or the DL number defined via the following machine data becomes active:

MD20272 $MC_SUMCORR_DEFAULT (basic setting of the sum offset without a program)

DL programming

The sum offset is always programmed relative to the active D number with the command:

`DL =` "n"

The sum offset "n" is added to the wear of the active D number.

Note

If you use "setup offset" **and** "sum offset fine", both compensations are combined and added to the tool wear.

The sum offset is deselected with the command:

```
DL = 0
```

Note

DL0 is not permitted. If compensation is deselected (`D0` and `T0`), the sum offset also becomes ineffective.

Programming a sum offset that does not exist triggers an alarm, similar to programming a D compensation that does not exist.

Thus, only the defined wear remains part of the compensation (defined in system variables $TC_DP12 to $TC_DP20).

Programming a sum offset when a D compensation is active (also applies to deselection) has the same effect on the path as programming a D command. An active radius compensation will, therefore, lose its reference to adjacent blocks.

Configuration

MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4=0: (Properties of sum offset in the TO area) default setting:

Only **one** set of sum offsets exists per DL number.

We refer in general to the sum offset.

This describes the data represented by $TC_SCPx.

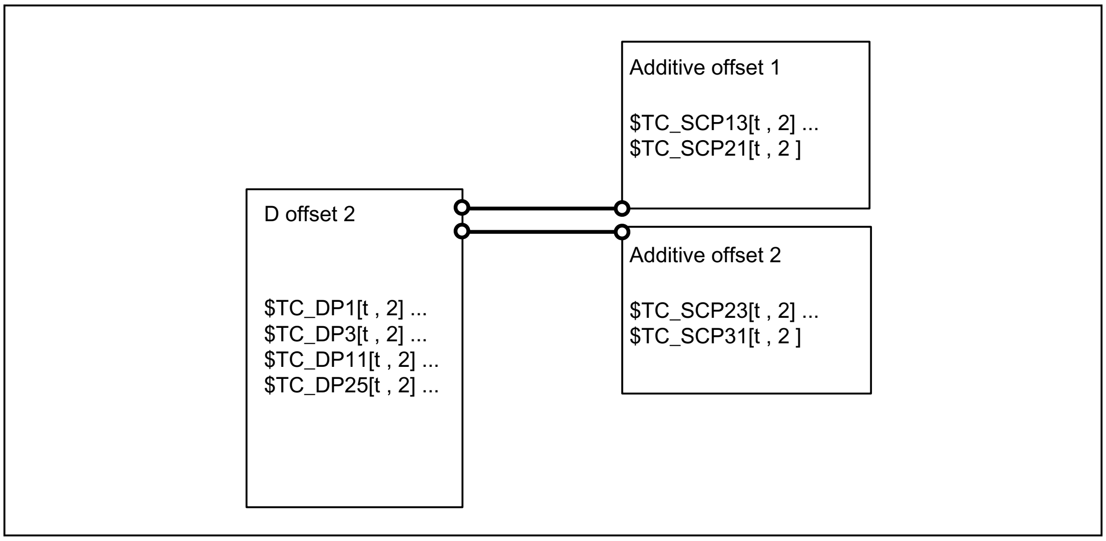


Figure 19-57 MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4 = 0

The tool with 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_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 remains active.

MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4=1: Setup offsets are available

The sum offset is now composed of the "sum offset fine" (represented by $TC_SCPx) and the setup offset (represented by $TC_ECPx). Two data sets therefore exist for one DL number. The sum offset is calculated by adding the corresponding components ($TC_ECPx + $TC_SCPx).

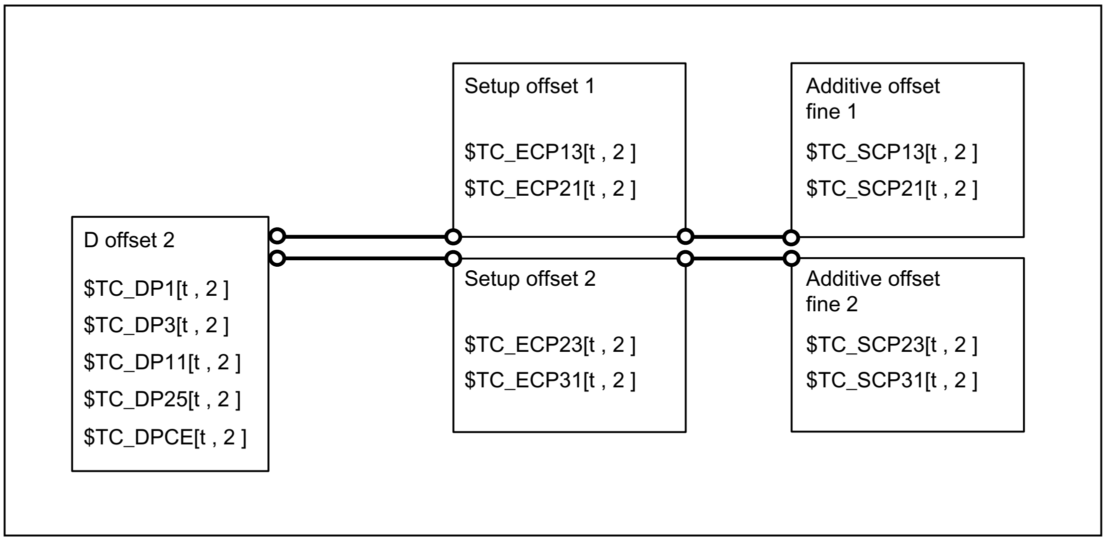


Figure 19-58 MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4 = 1 "setup offsets" + "sum offsets fine"

The tool with 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 remains active.

Reading/writing in the part program

The individual sets of sum offset parameters are differentiated according to the number ranges of system variable $TC_SCP.

The significance of the individual variables is similar to geometry variables $TC_DP3 to $TC_DP11. Only length 1, length 2 and length 3 are enabled for the basic functionality (variables $TC_SCP13 to $TC_SCP15 for the first sum offset of the cutting edge).

R5 = $TC_SCP13[t, d]	;	Sets the value of the R parameter to the value of the first component of sum offset 1 for cutting edge (d) on tool (t).
R6 = $TC_SCP21[t, d]	;	Sets the value of the R parameter to the value of the last component of sum offset 1 for cutting edge (d) on tool (t).
R50 = $TC_SCP23[t, d]	;	Sets the value of the R parameter to the value of the first component of sum offset 2 for cutting edge (d) on tool (t).
$TC_SCP43[t, d] = 1.234	;	Sets the value of the first component of sum offset 4 for cutting edge (d) on tool (t) to the value 1.234.

The above statements also apply to the setup offsets (if the NCK is configured with this option), i.e.

R5 = $TC_ECP13[t, d]	;	Sets the value of the R parameter to the value of the first component of setup offset 1 for cutting edge (d) on tool (t).
R6 = $TC_ECP21[t, d]	;	Sets the value of the R parameter to the value of the last component of setup offset 1 for cutting edge (d) on tool (t).
etc.		

When working with setup offsets, "sum offsets fine" are written with the $TC_SCPx system variables.

Creating a new sum offset

If the compensation data set (x) does not yet exist, it is created on the first write operation to one of its parameters (y).

$TC_SCPxy[t, d] = r.r	;	The value "r.r" is assigned to parameter y of sum offset x. The other parameters of x have the value zero.

When working with setup offsets, "sum offsets fine" are written with the $TC_SCPx system variables.

Note

When working with setup offsets, the data set for the setup offset is created when a data set is created for "sum offset fine", if a data set did not already exist for [t, d].

Creating a new setup offset

If the compensation data set (x) does not yet exist, it is created on the first write operation to one of its parameters (y).

$TC_ECPxy[t, d] = r.r	;	The value "r.r" is assigned to parameter y of setup offset x. The other parameters of x have the value zero.

Note

When working with setup offsets, the data set for the "sum offset fine" is created when a data set is created for setup offsets, if a data set did not already exist for [t, d].

DELDL - Delete sum offset

Sum offsets are generally only relevant when machining with a cutting edge at a certain time at a certain location of the workpiece. You can use the NC language command `DELDL` to delete sum offsets from cutting edges (in order to release memory).

status = DELDL(t, d)	;	Deletes all sum offsets of cutting edge d on tool t.
	;	t, d are optional parameters.

If d is not specified, all sum offsets of all cutting edges on 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 is 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 Power On can be defined by machine data.

If the setting of the following machine data indicates that the last active tool compensation number (D) is to be activated after Power On, the last active DL number is then no longer active:

MD20110 $MC_RESET_MODE_MASK (definition of control default settings after reset / part program end)

19.14.4 Examples

Example 1

That no compensation and no sum offset will come into effect must be defined during tool change via the machine data:

- MD20270 $MC_CUTTING_EDGE_DEFAULT=0 (Basic setting of tool cutting edge without programming)
- MD20272 $MC_SUMCORR_DEFAULT=0 (default setting sum offset without program).

```
T5 M06              ; Tool number 5 is loaded - no compensation active.
D1 DL=3             ; Compensation D1 + sum offset 3 of D1 are activated.
X10
DL=2                ; Compensation D1 + sum offset 2 are activated.
X20
DL=0                ; Sum offset deselection, only compensation D1 is now active.
D2                  ; Compensation D2 is activated - the sum offset is not included
                      in the compensation.
X1
DL=1                ; Compensation D2 + sum offset 1 are activated.
X2
D0                  ; Compensation deselection
X3
DL=2                ; No effect - DL2 of D0 is zero (same as programming T0 D2).
```

Example 2

During tool change it has to be defined that offset `D2` and sum offset `DL=1` are activated via the machine data:

MD20270 $MC_CUTTING_EDGE_DEFAULT=2 (Basic setting of tool cutting edge without programming)

MD20272 $MC_SUMCORR_DEFAULT=1 (default setting sum offset without program)

```
T5 M06          ; Tool number 5 is loaded - D2 + DL=1 are active (= values of
                  machine data)
D1 DL=3         ; Compensation D1 + sum offset 3 of D1 are activated.
X10
DL=2            ; Compensation D1 + sum offset 2 are activated.
X20
DL=0            ; Sum offset deselection, only compensation D1 is now active.
D2              ; Compensation D2 is activated - sum offset DL=1 is activated.
X1
DL=2            ; Compensation D2 + sum offset 2 are activated.
D1              ; Compensation D1 + sum offset 1 are activated.
```

19.14.5 Upgrades for Tool Length Determination

19.14.5.1 Calculation of compensation values on a location-specific and workpiece-specific basis

Composition of the effective tool length

For a tool compensation without active kinematic transformation, the effective tool length consists of up to 8 vectors:

• Tool length (geometry)	($TC_DP3 - $TC_DP5)
• Wear	($TC_DP12 - $TC_DP14)
• Tool base dimension (see note)	($TC_DP21 - $TC_DP23)
• Adapter dimension (see note)	($TC_ADPT1 - $TC_ADPT3)
• Total offsets fine	($TC_SCPx3 - $TC_SCPx5)
• Sum offsets coarse or setup offsets	($TC_ECPx3 - $TC_ECPx5)
• Offset vector l_1 of toolholder with orientation capability	($TC_CARR1 - $TC_CARR3)
• Offset vector l_2 of toolholder with orientation capability	($TC_CARR4 - $TC_CARR6)
• Offset vector l_3 of toolholder with orientation capability	($TC_CARR15 - $TC_CARR17)

Note

The tool base dimension and adapter dimension can only be applied as alternatives.

Type of action of the individual vectors

The type of action of the individual vectors or groups of vectors depends on the following further quantities:

Influencing quantity	Operating principle
G commands	Active machining plane
Tool type	Milling tool or turning/grinding tools
Machine data	Tool management active/not active, toolholder with orientation capability available/not available
Setting data	Behavior of tool length components when mirroring or when changing the plane
Toolholder with orientation capability	Set values of toolholder with orientation capability
Adapter transformations	Transformed tool compensation values

Distribution over the geometry-axis components

How the three vector components of partial totals of the vectors involved are distributed over the three geometry-axis components is determined by the following quantities:

Influencing quantity	Dependencies
Active processing level: `G17` X/Y direction `G18` Z/X direction `G19` Y/Z direction	Infeed plane: Z Y X
Tool type: Milling tools, drilling tools, grinding tools, turning tools	See Section "Tool parameter 1: Tool type (Page 1477)", Table "Minimum number of required tool parameters"
SD42900 $SC_MIRROR_TOOL_LENGTH SD42910 $SC_MIRROR_TOOL_WEAR SD42920 $SC_WEAR_SIGN_CUTPOS SD42930 $SC_WEAR_SIGN SD42940 $SC_TOOL_LENGTH_CONST SD42950 $SC_TOOL_LENGTH_TYPE	See Section "Special handling of tool compensations (Page 1613)" and Section "Setting data (Page 1668)".
Adapter transformations	**References:** Function Manual Tool Management

The resulting tool orientation always remains parallel to one of the three axis directions X, Y or Z and exclusively depends on the active machining plane `G17-G19`, since it has not yet been possible to assign the tool an orientation.

Stepless variation of the tool orientation

The toolholder with orientation capability also enables the tool orientation to be varied steplessly, in addition to providing further offsets or linear expansion fluctuations with the aid of offset vectors l_1 - l_3.

For further explanations, see Section "Toolholder with orientation capability (Page 1532)".

Minor operator compensations

Minor compensations, however, must also be modified during the normal production mode.

The reasons for this are, for example:

- Tool wear
- Clamping errors
- Temperature sensitivity of the machine:

These compensations are defined as follows:

Definition	Wear components
Wear	$TC_DP12 - $TC_DP14,
Total offsets fine	$TC_SCPx3 - $TC_SCPx5,
Sum offsets coarse or setup offsets	$TC_ECPx3 - $TC_ECPx5

In particular, compensations, which affect the tool length calculation, should be entered in the coordinates used for measurement.

These workpiece-specific compensations can be achieved more simply using the G group 56 with the three values `TOWSTD`, `TOWMCS` and `TOWWCS` and the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformation of tool components)

SD42935

Which of the wear components:

- Wear ($TC_DP12 - $TC_DP14)
- Setup offsets or sum offsets coarse ($TC_ECPx3 - $TC_ECPx5)
- Sum offsets fine ($TC_SCPx3 - $TC_SCPx5)

are to be transformed in the transformations:

- Adapter transformation
- Toolholder with orientation capability

are to be or not to be transformed, can be defined via the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformation of wear values)

With the setting data in its initial state, all wear values are transformed.

The setting data is considered in the following functions:

- Wear values in the machine coordinate system
  Part program instruction: `TOWMCS`
- Wear values in the workpiece coordinate system
  Part program instruction: `TOWWCS`

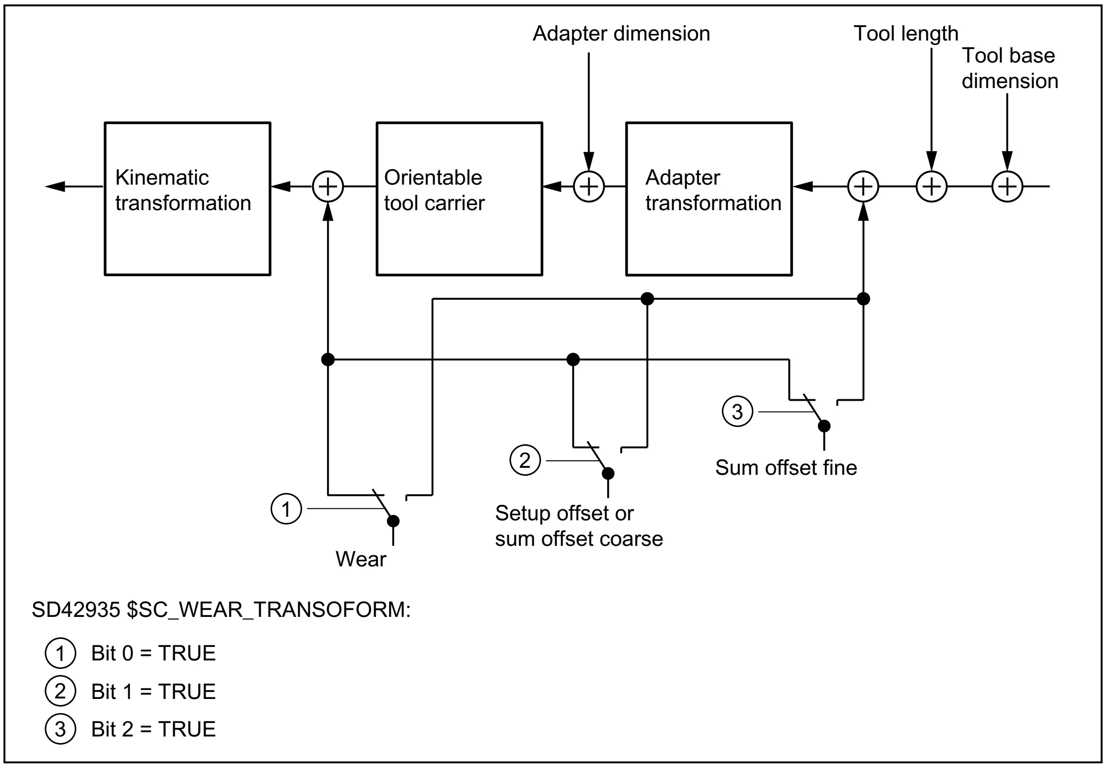


Figure 19-59 Transformation of wear data dependent on SD42935

Programming

G-code group 56 can be used to define the following values:

Syntax	Corrections
`TOWSTD`	Basic setting value for offsets in tool length
`TOWMCS`	Wear values in the machine coordinate system (MCS)
`TOWWCS`	Wear values in the workpiece coordinate system (WCS)
`TOWBCS`	Wear values in the basic coordinate system (BCS)
`TOWTCS`	Wear values in the TCS (Tool Coordinate System) at the toolholder (tool carrier reference point T)
`TOWKCS`	Wear values in tool coordinate system for kinematic transformation (KCS) of tool head

Coordinate systems for offsets in tool length

G commands `TOWMCS`, `TOWWCS`, `TOWBCS`, `TOWTCS` and `TOWKCS` can be used, e.g. to measure the wear tool length component in five different coordinate systems.

1. Machine coordinate system	MCS
1. Basic coordinate system	BCS
1. Workpiece coordinate system	WCS
1. Tool coordinate system of kinematic transformation	KCS
1. Tool coordinate system	TCS

The calculated tool length or a tool length component can be represented and read out using the predefined GETTCOR (Page 1644) function in one of these coordinate systems.

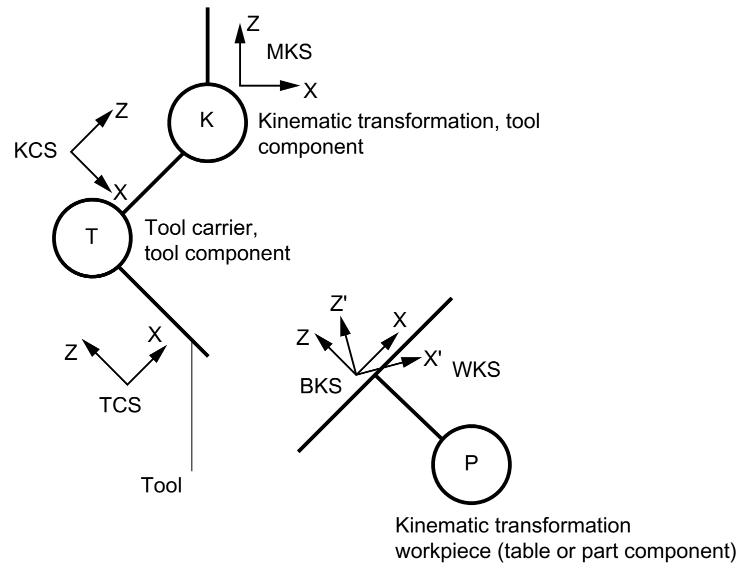

Z
MKS
X
Z
KCS
X
K
Kinematic transformation, tool component
T
Tool carrier, tool component
Z'
Z
X
X'
WKS
BKS
X
Z
TCS
Tool
P
Kinematic transformation workpiece (table or part component)

Figure 19-60 Coordinate system for the evaluation of tool lengths

19.14.5.2 Functionality of the individual wear values

TOWSTD

Basic setting (default behavior):

- The wear values are added to the other tool length components.
  The resulting total tool length is then used in further calculations.

In the case of an active tool carrier with orientation capability:

- The wear values are subjected to the appropriate rotation.

TOWMCS

Wear data in the MCS (machine coordinate system):

In the case of an active rotation by means of a tool carrier with orientation capability:

- The tool carrier only rotates the vector of the resultant tool length. Wear is ignored. Then the tool length vector rotated in this way and the wear are added. The wear is not subjected to the rotation.

If **no** tool carrier with orientation capability is active or this does not result in a rotation, TOWMCS and TOWSTD are identical.

Linear transformation

The tool length can be uniquely defined in the MCS only if the MCS is generated by linear transformation from the BCS.

This would also be the case under one of the following conditions:

- No kinematic transformation active.
- Orientation transformations (3-axis, 4-axis, and 5-axis transformations) are active.

TOWWCS

Wear values in WCS (workpiece coordinate system):

- If a tool carrier with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data is interpreted in the workpiece coordinate system.

The wear vector in the workpiece coordinate system is converted to the machine coordinate system and added to the tool vector.

TOWBCS

Wear values in BCS (basic coordinate system):

- If a tool carrier with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data is interpreted in the workpiece coordinate system.

The wear vector in the basic coordinate system is converted to the workpiece coordinate system and added to the tool vector.

Non-linear transformation

If a non-linear transformation is active, e.g. with TRANSMIT, and the MCS is specified as the desired coordinate system, the BCS is automatically used instead of the MCS.

Tool carrier with orientation capability

A table component of the tool carrier with orientation capability, if available, is not applied directly to the coordinate systems, unlike a table (or part) component of the kinematic transformation. A rotation described by such a component is represented in a basic frame or system frame and is thus included in the transition from WCS to BCS.

Kinematic transformation

The table (or part) component of the kinematic transformation is described by the transition from BCS to MCS.

TOWTCS

Wear values in TCS (tool coordinate system):

- If a tool carrier with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data is interpreted in the tool coordinate system.

The wear vector in the TCS (Tool Coordinate System) is converted to the machine coordinate system by way of the tool coordinate system of the kinematic transformation (KCS) and added to the tool vector.

TOWKCS

The wear value specifications for the kinematic transformation are interpreted in the associated TCS (Tool Coordinate System).

The wear vector is converted to the machine coordinate system by way of the tool coordinate system of the kinematic transformation and added to the tool vector.

G command change when a tool is active

Changing the G command in the group TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, and TOWKCS does not affect an already active tool, and does not become effective until the next tool is selected.

A new G command of this group will also come into effect if it is programmed in the same block, in which a tool is selected.

Evaluation of individual wear components

Evaluation of individual wear components (assignment to geometry axes, sign evaluation) is influenced by:

- The active plane
- The adapter transformation
- The five setting data shown in the table below

Setting data	Wear components		
SD42910 $SC_MIRROR_TOOL_WEAR	TOWSTD	TOWMCS	TOWWCS
SD42920 $SC_WEAR_SIGN_CUTPOS	X	X	—
SD42930 $SC_WEAR_SIGN	X	—	—
SD42940 $SC_TOOL_LENGTH_CONST	X	X	X
SD42950 $SC_TOOL_LENGTH_TYPE	X	X	X

Note

Wear components which are subjected to an active rotation by an adapter transformation or a tool carrier with orientation capability are referred to as non-transformed wear components.

Special features

If TOWMCS or TOWWCS is active, the following setting data does not affect the non-transformed wear components:

SD42920 $SC_WEAR_SIGN_CUTPOS (Sign of wear for tools with cutting edge systems)

The following setting data also does not affect the non-transformed wear components in case of TOWWCS:

SD42910 $SC_MIRROR_TOOL_WEAR (Sign change tool wear when mirroring)

In this case, a possibly active mirroring is already contained in the frame, which is referred to for evaluating the wear components.

On a plane change, the assignment between the non-transformed wear components and the geometry axes is retained, i.e. these are not interchanged as with other length components. The assignment of components depends on the active plane for tool selection.

Example

Let's assume a milling tool is used where only the wear value $TC_DP12 assigned to length L1 is not equal to zero.

If G17 is active, this length is effective in the direction of the z axis.

This measure always acts in the Z-direction also upon a plane change after the tool selection, when `TOWMCS` or `TOWWCS` are active and the bit 1 is set in the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)

If, for example, `G18` is active on tool selection, the component is always effective in the Y direction instead.

19.15 Working with tool environments

Overview of functions

- Save tool environment (TOOLENV) (Page 1640)
- Delete tool environment (DELTOOLENV) (Page 1642)
- Read T, D and DL number (GETTENV) (Page 1643)
- Read tool lengths and/or tool length components (GETTCOR) (Page 1644)
- Change tool components (SETTCOR) (Page 1650)

System variables overview

- Read information about the saved tool environments ($P_TOOLENVN, ($P_TOOLENV) (Page 1644)

19.15.1 Save tool environment (TOOLENV)

The TOOLENV 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 command of group:
  - 6 (G17, G18, G19)
  - 56 (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS)
- The active transverse axis
- Machine data:
  - MD18112 $MN_MM_KIND_OF_SUMCORR (properties of the summed offsets in the TO area)
  - MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).
- Setting data:
  - SD42900 $SC_MIRROR_TOOL_LENGTH (sign change tool length when mirroring)
  - SD42910 $SC_MIRROR_TOOL_WEAR (sign change tool wear when mirroring)
  - SD42920 $SC_WEAR_SIGN_CUTPOS (sign of the tool wear with cutting edge systems)
  - SD42930 $SC_WEAR_SIGN (sign of wear)
  - SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)
  - SD42940 $SC_LENGTH_CONST (change of the tool length components for a plane change)
  - SD42942 $SC_TOOL_LENGTH_CONST_T (change of tool length components for turning tools at change of plane)
  - SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)
  - SD42954 $SC_TOOL_ORI_CONST_M (change of tool orientation components for milling tools at change of plane)
  - SD42956 $SC_TOOL_ORI_CONST_T (change of tool orientation components for turning tools at change of plane)
- The orientation component of the current complete frame (rotation and mirroring, no work offsets or scaling)
- 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.

Syntax

```
<Status> = TOOLENV(<name>)
```

Meaning

`TOOLENV(...)`:		Predefined function to save a tool environment		
		Alone in the block:		Yes
`<Status>`:		Function return value. Negative values indicate error states.		
		Data type:		INT
		Value:	0	Function OK
			-1	No memory reserved for tool environments: MD18116 $MN_MM_NUM_TOOL_ENV = 0 This means that the "tool environments" functionality is not available.
			-2	No more free memory locations for tool environments available.
			-3	Null string illegal as name of a tool environment.
			-4	No parameter (<name>) specified.
Parameters				
1	`<name>`:	Name, under which the current data set should be saved. If a data set of the same name already exists, then it is overwritten. In this case, the status is "0".		
		Data type:		STRING

Additional information

Base dimension/adapter dimension – tool length compensation

When the tool magazine management is active (only available with the "Tool management" option!), the value of the following machine data defines whether the adapter length or the tool base dimension (cutting edge-specific parameters $TC_DP21, $TC_DP22 and $TC_DP23) is incorporated in the calculation of the tool length:

MD18104 $MN_MM_NUM_TOOL_ADAPTER (tool adapter in TO area).

Since a change to this machine data only takes effect after the control system has powered up, it is not saved in the tool environment.

Resulting length of toolholders with orientation capability and transformations:

Note

Both toolholders with orientation capability and transformations can use system variables or machine data, which act as additional tool length components, and which can be subjected partially or completely to the rotations performed. The resulting additional tool length components must also be saved 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 a different tool can be calculated assuming that it would be used under the same conditions as the tool, for which the status was saved.

Maximum number of data sets for tool environments

Machine data MD18116 $MN_MM_NUM_TOOL_ENV is used to define the maximum number of data sets that can be saved to describe the tool environments. The data are in the TOA area. They are kept even when the control system is switched off.

Data cannot be backed up. This means that this data cannot be transferred between the different control systems.

19.15.2 Delete tool environment (DELTOOLENV)

The DELTOOLENV function is used to delete the data sets that are 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).

Note

Data sets can only be deleted using the DELTOOLENV function, by an INITIAL.INI download or by a cold start (NC power up with default machine data). There are no additional automatic deletion operations.

Syntax

```
<Status> = DELTOOLENV(<name>)
<Status> = DELTOOLENV()
```

Meaning

`DELTOOLENV(...)`:		Predefined function to delete a tool environment		
		Alone in the block:		Yes
`<Status>`:		Function return value. Negative values indicate error states.		
		Data type:		INT
		Value:	0	Function OK
			-1	No memory reserved for tool environments: MD18116 $MN_MM_NUM_TOOL_ENV = 0 This means that the "tool environments" functionality is not available.
			-2	A tool environment with the specified name does not exist.
Parameters				
1	`<name>`:	Name of data set to be deleted		
		Data type:		STRING

`DELTOOLENV():`	`DELTOOLENV()` deletes data sets describing tool environments without specifying a name

19.15.3 Read T, D and DL number (GETTENV)

The GETTENV function is used to read the T, D and DL numbers stored in a tool environment.

Syntax

```
<Status> = GETTENV(<name>, <TDDL>)
```

Meaning

`GETTENV(...):`		Predefined function to read T, D and DL numbers in a data set to describe a tool environment		
		Alone in the block:		Yes
`<Status>:`		Function return value. Negative values indicate error states.		
		Data type:		INT
		Value:	0	Function OK
			-1	No memory reserved for tool environments: MD18116 $MN_MM_NUM_TOOL_ENV = 0 This means that the "tool environments" functionality is not available.
			-2	A tool environment with the specified name does not exist.
Parameters				
1	`<name>:`	Name of the data set from which the T, D and DL numbers are to be read		
		Data type:		STRING
2	`<TDDL>:`	The field of this result parameter contains the T, D and DL numbers of the tool, whose tool environment is saved in the specified data set: • <TDDL> [0]: T number • <TDDL> [1]: D number • <TDDL> [2]: DL number		
		Data type:		INT[3]

`GETTENV(,<TDDL>),`
`GETTENV("",<TDDL>):`
	When calling function GETTENV, it is permissible to omit the first parameter – or to transfer the null string as first parameter. In these two special cases, in <TDDL>, the T, D and DL numbers of the**active** tool are returned.

19.15.4 Read information about the saved tool environments ($P_TOOLENVN, ($P_TOOLENV)

Information regarding the saved tool environments can be read using the following system variables:

`$P_TOOLENVN`:	Supplies the number of data sets (which have still not been deleted) – defined using TOOLENV – to describe tool environments			
	Syntax:	`<n> = $P_TOOLENVN`		
	Meaning:	`<n>`:	Number of defined data sets	
			Data type:	INT
			Value range:	0 ... MD18116 $MN_MM_NUM_TOOL_ENV
	This system variable can be accessed even if no tool environments are possible (MD18116 = 0). In this case, the return value is "0".			
`$P_TOOLENV`:	Supplies the name of the <i>th data set to describe a tool environment			
	Syntax:	`<Name> = $P_TOOLENV[<i>]`		
	Meaning:	`<name>`:	Name of the data set with number <i>	
			Data type:	STRING
		`<i>`:	Number of the data set	
			Data type:	INT
			Value range:	1 ... $P_TOOLENVN
	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. If <i> refers to a data set that has not been defined, then the null string is returned. If index <i> is not valid, i.e. <i> is less than 1 or higher than that the maximum number of data sets for tool environments (MD18116 $MN_MM_NUM_TOO-LENV), then the following alarm is output: Alarm 17020 "inadmissible array index 1"			

19.15.5 Read tool lengths and/or tool length components (GETTCOR)

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.

Syntax

```
<Status> = GETTCOR(<Len>[, <Comp>, <Stat>, <T>, <D>, <DL>])
```

Meaning

`GETTCOR(...)`:	Predefined function to read tool lengths or to read tool length components	
	Alone in the block:	Yes

`<Status>`:		Function return value. Negative values indicate error states.		
		Data type:		INT
		Value:	0	Function OK
			-1	No memory reserved for tool environments: MD18116 $MN_MM_NUM_TOOL_ENV = 0 This means that the "tool environments" functionality is not available.
			-2	A tool environment with the name specified under <Stat> does not exist.
			-3	Invalid string in parameter <Comp>. Causes of this error can be invalid characters or characters programmed twice.
			-4	Invalid T number
			-5	Invalid D number
			-6	Invalid DL number
			-7	Attempt to access a non-existent memory module.
			-8	Attempt to access a non-existent option (programmable tool orientation, tool management).
			-9	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.
Parameters				
1	`<Len>`:	Result vector		
		Data type:		REAL[11]
		The vector components are arranged in the following order: • <Len> [0]: Tool type • <Len> [1]: Cutting edge position • <Len> [2]: Abscissa • <Len> [3]: Ordinate • <Len> [4]: Applicate • <Len> [5]: Tool radius The coordinate system defined in <Comp> and <Stat> is used as the reference coordinate system for the length components. If a coordinate system is not defined in <Comp>, then tool lengths are displayed in the machine coordinate system. The assignment of the abscissa, ordinate and applicate to the geometry axes depends on the active plane used in the tool environment. This means, for 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>[6] 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).		

2	`<Comp>`:	Tool length components (optional)			
		Data type:	STRING		
		The character string consists of two substrings, which are separated from one another by a colon. General form: `"<SubStr_1> [: <SubStr_2]"`			
		`<SubStr_1>`:	The **first substring** designates the tool length components to be taken into account when calculating the tool length. The order of the characters in the substrings, and their notation (upper or lower case), is arbitrary. Any number of blanks or white spaces can be inserted between the characters. **Note:** It is **not permissible** that the characters in the substring are programmed twice.		
			Charac-ters:	`-`	Minus symbol (only allowed as first character) The complete tool length is calculated, minus the components specified in the next string.
				`C`	Adapter or tool base dimension (whichever of the two alternative components is active for the tool in use)
				`E`	Setup offsets
				`G`	Geometry
				`K`	Kinematic transformation (is only evaluated for generic 3, 4 and 5-axis transformation)
				`S`	Summed 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.		
		`<Substr_2>`:	The optional **second substring** identifies the coordinate system, in which the tool length is to be output. The second substring only comprises one single relevant character.		
			Charac-ters:	`A`	Adjustable coordinate system (ACS)
				`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)
			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>.		
3	`<_Stat>`:	Name of the data set for describing a tool environment (optional)			
		Data type:	STRING		
		If the value of this parameter is the null string (""), or is not specified, then the current status is used. The current tool is used if a tool is not specified.			

4	`<T>`:	Internal T number of the tool (optional).	
		Data type:	INT
		If this parameter is not specified or if its value is "0", then the tool stored in <Stat> is used. If the value of this parameter is "-1", then the T number of the active tool is used. It is also possible to explicitly specify the number of the active tool. **Note:** If <Stat> is not specified, the actual status is used as the tool environment. Since <T> = 0 refers to the T number saved in the tool environment, the active tool is used in this environment, i.e. parameters <T> = 0 and <T> = -1 have the same meaning in this special case.	
5	`<D>`:	Cutting edge of the tool (optional).	
		Data type:	INT
		If this parameter is not specified, or if its value is "0", then the D number used is based on the source of the T number. If the T number from the tool environment is used, then the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.	
6	`<DL>`:	Number of the offset dependent on the location (optional).	
		Data type:	INT
		If this parameter is not specified, then the DL number used is based on the source of the T number. If the T number from the tool environment is used, then the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.	

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. Output in the workpiece coordinate system.
`GETTCOR (_LEN,"-K:B")`	Calculates the complete tool length of the active tool without allowing for the length components of a possibly 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". However, the calculation is performed for cutting edge number D3, regardless of the cutting edge number stored.

Additional information

Adapter transformation/toolholder with orientation capability/kinematic transformation

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.

Turning and grinding tools: Calculating the tool length depending on MD20360 $MC_TOOL_PARAMETER_DEF_MASK

The following machine data defines how the wear and tool length are to be evaluated if a diameter axis is used for turning and grinding tools.

MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).

Bit	Value	
0	For turning and grinding tools, the **wear parameter** of the transverse axis is taken into account as the diameter value:	
	= 0 (default)	No
	= 1	Yes
1	For turning and grinding tools, the **tool length component** of the transverse axis is taken into account as the diameter value:	
	= 0 (default)	No
	= 1	Yes

If the bits involved are set, the associated entry is weighted with a factor of 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 (geometry axis with transverse axis function) = "X"

X is diameter axis (standard turning machine configuration)

```
Program code                         Comment
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)
```

Program code	Comment
`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`	

Length components of the kinematic transformation and 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	l1 and l2
T	l1, l2 and l3
P	The 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:

Transforma-tion 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 (moving tool and moving workpiece) corresponds to type M for toolholders with orientation capability.

For this 5-axis transformation, in the previous software releases, vector MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2 (vector of kinematic offset of the 1st/2nd 5-axis transformation in the channel) corresponds to the sum of the two vectors l_1 and l_3 for a type M tool carrier with orientation capability.

Only the sum is relevant for the transformation in both cases. The way, in which the two individual components are composed, is insignificant. However, when calculating the tool length, it is relevant which component is assigned to the tool and which is assigned to the tool table. This is the reason that machine data MD24558/24658 $MC_TRAFO5_JOINT_OFFSET_PART_1/2 (vector kinematic offset in table) was introduced. It corresponds to vector l3. Machine data:MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2 no longer corresponds to the sum of l1 and l3, but only to vector l1. If machine data MD24558/24658 $MC_TRAFO5_JOINT_OFFSET_PART_1/2 is equal to zero, the behavior is the same as before.

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 functions mentioned above can be configured to reproduce the behavior of the measuring cycles in relation to the tool length calculation.

19.15.6 Change tool components (SETTCOR)

The SETTCOR function is used to change tool components taking into account all general conditions that can be involved when evaluating the individual components.

Note

Regarding the terminology: If in the following, in conjunction with the tool length, tool components are involved, then the components considered from a vectorial perspective are meant, which make up the complete tool length (e.g. geometry or wear). Such a component comprises three individual values (L1, L2, L3), which are called coordinate values in the following.

The tool component "geometry" therefore comprises three coordinate values $TC_DP3 to $TC_DP5.

Syntax

```
<Status> = SETTCOR(<CorVal>, <Comp>, [<CorComp>, <CorMode>, <GeoAx>,
<Stat>, <T>, <D>, <DL>])
```

Meaning

`SETTCOR(...)`:	Predefined function to change tool components	
	Alone in the block:	Yes

`<Status>`:		Function return value. Negative values indicate error states.		
		Data type:		INT
		Value:	0	Function OK
			-1	No memory reserved for tool environments: MD18116 $MN_MM_NUM_TOOL_ENV = 0 This means that the "tool environments" functionality is not available.
			-2	A tool environment with the name specified under <Stat> does not exist.
			-3	Invalid string in parameter <Comp>. Causes of this error can be invalid characters or characters programmed twice.
			-4	Invalid T number.
			-5	Invalid D number.
			-6	Invalid DL number.
			-7	Attempt to access a non-existent memory module.
			-8	Attempt to access a non-existent option (programmable tool orientation, tool management).
			-9	Illegal numerical value for parameter <CorComp>.
			-10	Illegal numerical value for parameter <CorMode>.
			-11	The contents of parameters <Comp> and <CorComp> are contradictory.
			-12	The contents of parameters <Comp> and <CorMode> are contradictory.
			-13	The content of the <GeoAx parameter does not designate a geometry axis.
			-14	Write attempt to a non-existent setup offset.
Parameters				
1	`<CorVal>`:	Correction vector In the workpiece coordinate system (WCS) defined by <Stat>, the following assignment applies: ● <CorVal> [0]: Abscissa ● <CorVal> [1]: Ordinate ● <CorVal> [2]: Applicate If only one tool component is to be corrected (i.e. no vectorial correction, see parameter <CorMode>), the correction value is always in <CorVal>[0], independent of the axis on which it acts. The contents of the other two components are then not evaluated. If <CorVal> or a component of <CorVal> refers to the transverse axis, then the data is evaluated as **radius dimension**. This means that a tool is, for example, "longer" by the specified dimension; this correspondingly results in a change to the workpiece diameter that is twice as large. The dimensions refer to the **basic system** (inch or metric) of the control system.		
		Data type:		REAL[3]

2	`<Comp>`:	Tool component(s)			
		Data type:	STRING		
		The character string consists of two substrings, which are separated from one another by a colon. General form: `"<SubStr_1> [: <SubStr_2]"`			
		`<SubStr_1>`:	The **first substring** must always be available, and can either comprise one or two characters. The first or only character for the 1st component (Val_1) and the second character for the 2nd component (Val_2), which are processed according to the subsequent parameters <CorComp> and <CorMode>.		
			Charac-ters:	`C`	Adapter or tool base dimension (whichever of the two alternative components is active for the tool in use)
				`E`	Setup offsets
				`G`	Geometry
				`S`	Sum offsets
				`W`	Wear
		`<Substr_2>`:	The **second substring** is optional. Alternatively, it can comprise (individual) letters "W" or "T".		
			Charac-ters:	`W`	If the second substring is empty or contains the letter "W", then the offset values are taken into account as if they had been measured in the **workpiece**coordinate system (WCS).
				`T`	If the second substring contains the letter "T", then the offset values are taken into account as if they had been measured in the **tool**coordinate system (**T**ool Coordinate System, TCS).
		The notation of the characters in the string (upper or lower case) is arbitrary. Any number of spaces or tabs (white spaces) can be inserted.			

3	`<CorComp>`:	Specifies the component(s) of the tool data sets that are to be described (optional).		
		Data type:		INT
		Value:	`0`	Offset value <CorVal>[0] refers to the geometry axis transferred in parameter <GeoAx> in the workpiece coordinate system – or in the tool coordinate system (also see a description of parameter <Comp>). This means that the offset value must be calculated in the designated tool components so that, taking account all the parameters that can influence the tool length calculation, a change of the total tool length by the specified value in the specified axis direction is obtained. This change should be achieved by correcting the component specified in <Comp> and the symbolic algorithm specified in <CorMode> (see the following parameters). The resulting correction can therefore have an effect on all three axis components.
			`1`	Like "0", however, vectorial. The content of vector <CorVal> refers to abscissa, ordinate and applicate in the workpiece coordinate system or tool coordinate system (see the description of parameter <Comp>). Subsequent parameter <GeoAx> is not evaluated.
			`2`	Vectorial offset, i.e. L1, L2 and L3 can change simultaneously. In contrast to the versions from "0 and "1", the offset values contained in <CorVal> refer to the coordinates of Val_1 components (see following parameter <CorMode>) of the tool. Any possible inclination of an existing tool compared with the workpiece coordinate system has no influence on the offset.
			`3 - 5`	Correction of tool lengths L1 to L3 ($TC_DP3 to $TC_DP5) or the corresponding values for wear, setting up or additive offsets. The offset value is contained in <CorVal>[0]. It is measured in the coordinates of the Val_1 component (see following parameter <CorMode>) of the tool. Any possible inclination of an existing tool compared with the workpiece coordinate system has no influence on the offset.
			`6`	Correction of the tool radius ($TC_DP6) or the corresponding values for wear, setting up or additive offsets. Bits 10 and 11 (evaluation of the diameter and/or diameter wear data, either specified as a radius or diameter) in machine data MD20360 $MC_TOOL_PARAMETER_DEF_MASK are taken into account.
			`7 – 11`	Correction of $TC_DP7 to $TC_DP11 or the corresponding values for wear, setting up or additive offsets. These parameters are treated just like the tool radius.
		If this parameter is not specified then its value is "0".		

4	`<CorMode>`:	Specifies the type of write operation to be executed (optional).		
		Data type:		INT
		Value:	0	Val_{1new} = <CorVal>
			1	Val_{1new} = Val_{1old} + <CorVal>
			2	Val_{1new} = <CorVal> Val_{2new} = 0
			3	Val_{1new} = Val_{1old} + Val_{2old} + <CorVal> Val_{2new} = 0
		The notation Val_{1old} + Val_{2old} is symbolic. If the two components (due to the status of <_Stat>) are evaluated in different ways, i.e. if a rotation is effective between the two components, then Val_{2old} is transformed prior to addition so that the resulting tool length after deleting Val_{2new} and prior to the addition of <CorVal> remains unchanged. <CorVal> always refers to Val_1. <CorVal> is a value, which dependent on the second part of parameter <Comp>, is measured in the workpiece coordinate system (WCS) or in the tool coordinate system (TCS). It is therefore already transformed with respect to the tool components, in which it should be calculated. Therefore, it cannot be directly calculated together with the saved value, but must be transformed back prior to adding to Val_1 or Val_2. This can mean that the offset acts on an axis different than the one defined by <CorComp> – or that it acts on several axes. For the case <CorComp> = 0, i.e. when <CorVal> does not contain a vector, but only an individual value, then the described operations are executed in the coordinates in which <CorVal> was measured (WCS/TCS). In particular, this also applies to setting Val_{2new} to zero in variants 2 and 3. This result is then transformed back into the coordinates of the tool. This can mean that none of the coordinate values to be set to zero (L1, L2, L3) become zero, or coordinate values, that were previously zero, are now not equal to zero. However, if the corresponding operations are successively executed for all three geometry axes, then it is guaranteed that all three coordinate values of the components to be deleted are zero. If the tool is not rotated with respect to the workpiece coordinate system or is rotated so that all tool components remain parallel to the coordinate axes (axis exchange operations), then this also ensures that only one tool coordinate changes. The successive execution of the same operation (<CorMode>) with <CorComp> = 0 for all three coordinate axes in any sequence is identical with the single execution of the same operation with <CorComp>=1. For parameter values "0" and "1", parameter <Comp> must contain one character, and for parameter values "2" and "3", two characters. Example: <Comp> contains string "ES", <CorMode> the value "2" ⇒ Setup $offset_{new}$ = <CorVal>, summed $offset_{new}$ = 0 If parameter <CorMode> is not specified, then its value is "0".		
5	`<GeoAx>`:	Specifies the index of the geometry axis in which the offset value <CorVal>[0] was read (optional)		
		Data type:		INT
		Value range:		0 ... 2
		Indices 0 to 2 refer to abscissa, ordinate and applicate in the active plane (G17/G18/G19) of the current tool environment. The content of this parameter is only evaluated if parameter <CorComp> has a value of "0".		

6	`<Stat>`:	Name of the data set for describing a tool environment (optional)	
		Data type:	STRING
		If the value of this parameter is the null string (""), or is not specified, then the current status is used. The current tool is used if a tool is not specified.	
7	`<T>`:	Internal T number of the tool (optional).	
		Data type:	INT
		If this parameter is not specified or if its value is "0", then the tool stored in <Stat> is used. If the value of this parameter is "-1", then the T number of the active tool is used. It is also possible to explicitly specify the number of the active tool. **Note:** If <Stat> is not specified, the actual status is used as the tool environment. Since <T> = 0 refers to the T number saved in the tool environment, the active tool is used in this environment, i.e. parameters <T> = 0 and <T> = -1 have the same meaning in this special case.	
8	`<D>`:	Cutting edge of the tool (optional).	
		Data type:	INT
		If this parameter is not specified, or if its value is "0", then 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.	
9	`<TL>`:	Number of the offset dependent on the location (optional).	
		Data type:	INT
		If this parameter is not specified, then the DL number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read. If T, D and DL specify a tool without location-dependent offsets, no summed or setup offsets may be specified in parameter <Comp> (error code in <Status>).	

Note

Not all possible combinations of the three parameters <Comp>, <CorComp> and <CorMode> make sense. For example, algorithm 3 in <CorComp> requires that two characters are specified in <Comp>. If an invalid parameter combination is specified, then a corresponding error code is returned in the <Status>.

Examples

Example 1

Program code	**Comment**
`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`	

Program code	Comment
N70 R1=**SETTCOR(_CORVAL,"G",0,0,2)**	
N80 T1 D1 X0 Y0 Z0	; ==> MCS position X0.000 Y0.000 **Z1.333**
N90 M30	

<CorComp> is "0", therefore, the coordinate value of the geometry component acting in the Z direction must be replaced by the offset value 0.333.

The resulting total tool length is thus: L1 = 0.333 + 1.000 = 1.333

Example 2

Program code	Comment
N10 DEF REAL _CORVAL[3]	
N20 $TC_DP1[1,1]=120	; milling tool
N30 $TC_DP3[1,1]=10.0	; geometry L1
N40 $TC_DP12[1,1]=1.0	; wear L1
N50 _CORVAL[0]=0.333	
N60 T1 D1 G17 G0	
N70 R1=**SETTCOR(_CORVAL,"W",0,1,2)**	
N80 T1 D1 X0 Y0 Z0	; ==> MCS position X0.000 Y0.000 Z11.333
N90 M30	

<CorComp> is "1", this means that the offset value of 0.333 – acting in the Z axis – is added to the wear value of 1.0.

The resulting total tool length is thus: L1 = 10.0 + 1.333 = 11.333

Example 3

Program code	Comment
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 offset effective in the Z axis 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

Program code	Comment
N10 DEF REAL _CORVAL[3]	
N20 $TC_DP1[1,1]=120	; Milling tool
N30 $TC_DP3[1,1]=10.0	; Geometry L1

Program code	Comment
`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

Program code	Comment
`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), which for a milling tool, is assigned to tool component L3 due to G17. As a consequence, when calling SETTCOR, tool parameters $TC_DP3 and $TC_DP12 are not influenced. Instead, the compensation value is entered in $TC_DP5.

Example 6

Program code	Comment
`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 <CorMode> = 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:

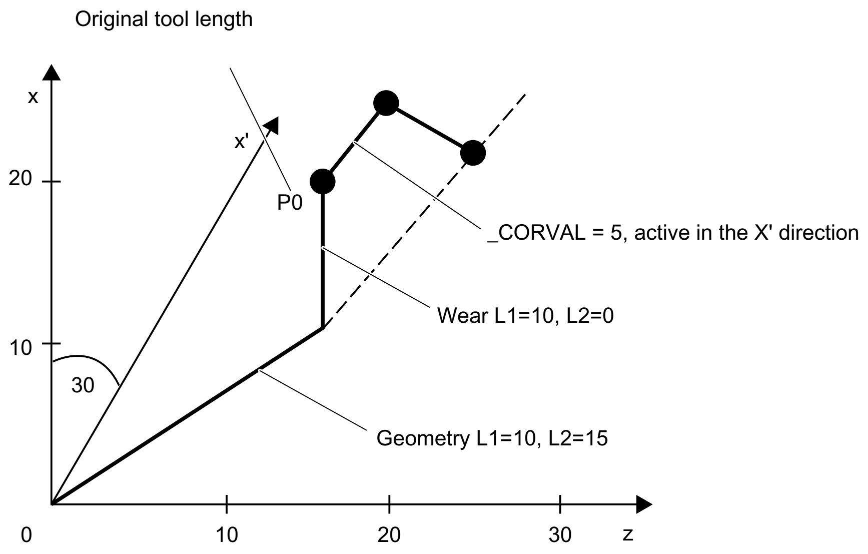
$TC_DP3[1,1]: 21.830 ; geometry L1

$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 geometrical relationships are shown in the figure below: 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.



If the program example is continued after N110 with the following instructions, then the remaining wear is included completely in the geometry because the compensation is now effective in the Z' axis (parameter <GeoAx> = 0):

```
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
```

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 <CorComp> = 1 (vectorial compensation) just once:

Program code	Comment
`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

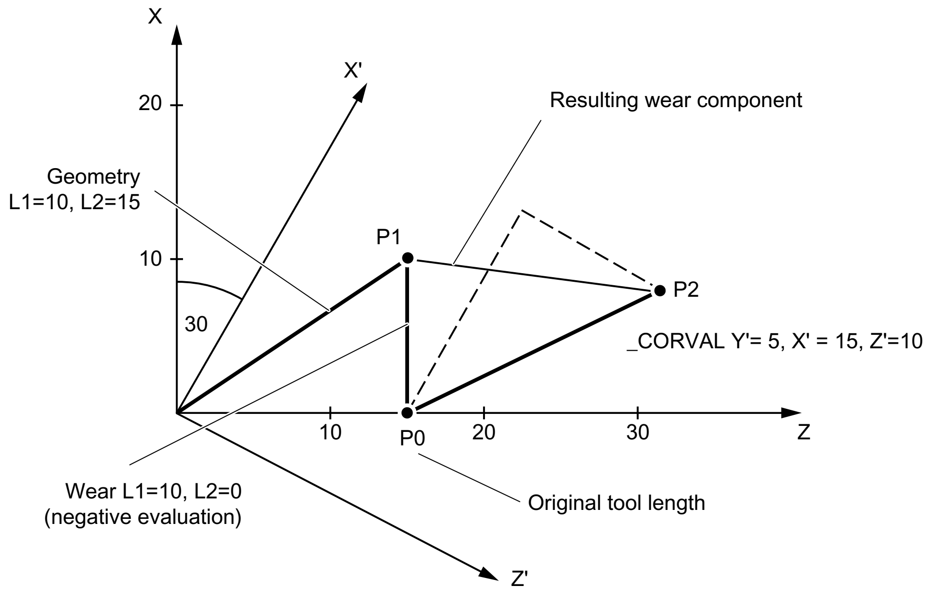
Program code	Comment
`N10 DEF REAL _CORVAL[3]`	
`N20 $TC_DP1[1,1]=500`	`; turning tool`
`N30 $TC_DP3[1,1]=10.0`	`; geometry L1`
`N40 $TC_DP4[1,1]=15.0`	`; geometry L2`
`N50 $TC_DP12[1,1]=10.0`	`; wear L1`
`N60 $TC_DP13[1,1]=0.0`	`; wear L2`
`N70 _CORVAL[0]=5.0`	
`N80 ROT Y-30`	
`N90 T1 D1 G18 G0`	
`N100 R1=`**`SETTCOR(_CORVAL,"GW",3,3)`**	
`N110 T1 D1 X0 Y0 Z0`	`; ==> MCS position X25.000 Y0.000 Z15.000`
`N120 M30`	

When compared 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

Program code	Comment
`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`
`N170 M30`	

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 geometrical relationships in the Z/X plane are shown in the diagram below:



The geometry component of the tool remains unchanged due to <CorMode> = 1. The compensation vector defined in the WCS (rotation around the y axis) must be included in the wear component such that the total tool length in Fig. 3 refers to point P_2. Therefore, the resulting wear component of the tool is given by the distance of the two points P_1 and P_2.

However, since the wear is evaluated negatively, due to setting data SD42930 $SC_WEAR_SIGN, 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 $SC_WEAR_SIGN on the L3 component in the Y direction can be recognized without the additional complication caused by the frame rotation.

Additional information

Turning/grinding tools: Calculating the tool length depending on MD20360 $MC_TOOL_PARAMETER_DEF_MASK

The following machine data defines how the wear and tool length are to be evaluated if a diameter axis is used for turning/grinding tools:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK.<Bit> = <Value>

<Bit>	<Value>	Meaning
0	0	For turning/grinding tools, the **wear parameter** of the transverse axis is taken into account in the **radius value**:
	1	For turning/grinding tools, the **wear parameter** of the transverse axis is taken into account as the **diameter value**:
1	0	For turning/grinding tools, the **tool length component** of the transverse axis is taken into account as the **radius value**:
	1	For turning/grinding tools, the **tool length component** of the transverse axis is taken into account as the **diameter value**:

If the bits involved are set, the associated entry is weighted with a factor of 0.5. The correction using SETTCOR is executed so that the total effective tool length change is equal to the value transferred in <CorVal>. If, when calculating the length, a length is evaluated with a factor of 0.5 as a result of machine data MD20360 $MC_TOOL_PARAMETER_DEF_MASK, then the compensation of this component must be realized with twice the value transferred.

Example

MD20360 $MC_TOOL_PARAMETER_DEF_MASK = 2 (tool length must be evaluated in the diameter axis using a factor of 0.5)

Axis X is the diameter axis.

```
Program code                                Comment
N10 DEF REAL _LEN[11]
N20 DEF REAL _CORVAL[3]
N30 $TC_DP1[1,1]=500                        ; Tool type
```

Program code	Comment
N40 $TC_DP2[1,1]=2	; Cutting edge position
N50 $TC_DP3[1,1]=3.	; Geometry - length 1
N60 $TC_DP4[1,1]=4.	; Geometry - length 2
N70 $TC_DP5[1,1]=5.	; Geometry - length 3
N80 _CORVAL[0]=1.	
N90 _CORVAL[1]=1.	
N100 _CORVAL[2]=1.	
N110 T1 D1 G18 G0 X0 Y0 Z0	; ==> MCS position X1.5 Y5 Z4
N120 R1=SETTCOR(_CORVAL,"G",1,1)	
N130 T1 D1 X0 Y0 Z0	; ==> MCS position X2.5 Y6 Z5
N140 R3=$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	

In each axis, the tool length compensation should be 1 mm (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.

19.16 Read the assignment of the tool lengths L1, L2, L3 to the coordinate axes (LENTOAX)

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[<t>,<d>] to $TC_DP5[<t>,<d>]) is considered, i.e. a different axis assignment for other components (e.g. wear) has no effect on the result.

Syntax

```
<Status> = LENTOAX(<AxInd>, <Matrix>[, <Coord>])
```

Principle

LENTOAX(...):	Predefined function to read the assignment of tool lengths L1, L2 and L3 of the active tool to the coordinate axes	
	Alone in the block:	Yes

`<Status>`:		Function return value. Negative values indicate error states.		
		Data type:		INT
		Value:	0	Function OK Information provided in <AxInd> is sufficient for the description (all tool length components are in 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.
Parameters				
1	`<AxInd>`:	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. ● <AxInd> [0]: Abscissa ● <AxInd> [1]: Ordinate ● <AxInd> [2]: Applicate		
		Data type:		INT[3]
		Value:	0	No assignment exists (axis does not exist)
			1 ... 3 or -1 ... -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.
		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), then the return value is <Status> = 1. The mapping of tool length components L1 to L3 to geometry axes 1 to 3 is then described completely by the content of the 2nd parameter <Matrix>.		
2	`<Matrix>`:	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.		
		Data type:		REAL
		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.		

3	`<Coord>`:	coordinate system applicable for the assignment (optional)		
		Data type:		STRING
		Charac-ters:	MCS M	The tool length is represented in the machine coordinate system.
			BCS B	The tool length is represented in the basic coordinate system.
			WCS W	The tool length is represented in the workpiece coordinate system (default setting).
			KCS K	The tool length is represented in the tool coordinate system of the kinematic transformation.
			TCS 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. If the parameter <Coord> is not specified, then WCS is used (default setting).		

Note

In the TCS, all tool length components are always parallel or antiparallel to the axes.

The components can only be antiparallel when mirroring is active and the following setting data is activated:

SD42900 $SC_MIRROR_TOOL_LENGTH (sign change tool length when mirroring)

Example

Standard application, milling tool for G17.

L1 applies in Z (applicate), L2 applies in Y (ordinate), L3 applies in X (abscissa).

Function call in the form:
`<Status>=LENTOAX(<AxInd>,<Matrix>,"WCS")`

The result parameter <AxInd> then 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:

$$\text{<Matrix>} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

A change from G17 to G18 or G19 does not alter the result, because the assignment of the length components to the geometry axes changes in the same way as the assignment of the abscissa, ordinate and applicate.

A frame rotation of Z through 60 degrees is now programmed with G17 active, e.g.

```
ROT Z60
```

The direction of the applicate (Z direction) remains unchanged; the main component of L2 now lies in the direction of the new X axis; the main component of L1 now lies in the direction of the negative Y axis. As a consequence, the return value (<Status>) is "1", <AxInd> contains the values (2, -3, 1).

In this case, the associated matrix (<Matrix>) is:

$$\text{<Matrix>} = \begin{pmatrix} 0 & \sin 60° & \cos 60° \\ 0 & \cos 60° & -\sin 60° \\ 1 & 0 & 0 \end{pmatrix}$$

19.17 Supplementary conditions

19.17.1 Flat D number structure

Block search

For a block search, you can parameterize when the help functions should be output to the PLC using:

- MD22080 $MC_AUXFU_PREDEF_SPEC
- MD22035 $MC_AUXFU_ASSIGN_SPEC
- MD11110 $MN_AUXFU_GROUP_SPEC

19.17.2 SD42935 expansions

Transformation of wear values

Which wear components in conjunction with the instructions `TOWMCS` (wear values in the machine coordinate system) and `TOWWCS` (wear values in the workpiece coordinate system) are transformed or not transformed can be defined using:

SD42935 $SC_WEAR_TRANSFORM (transformation of wear values)

19.17.3 Scratching

Scratching

If the "Scratch" function is used in the reset state, then regarding the initial setting value, the following must be taken into consideration:

- The wear components are evaluated depending on the initial settings of the G 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.

19.18 Data lists

19.18.1 Machine data

19.18.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
18082	MM_NUM_TOOL	Number of tools that the NC can manage (SRAM)
18088	MM_NUM_TOOL_CARRIER	Maximum number of the defined toolholders
18094	MM_NUM_CC_TDA_PARAM	Number of tool data (SRAM)
18096	MM_NUM_CC_TOA_PARAM	Number of data, per tool cutting edge for compile cycles (SRAM)
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Tool offsets in the TOA area (SRAM)
18102	MM_TYPE_OF_CUTTING_EDGE	Type of D number programming (SRAM)
18105	MM_MAX_CUTTING_EDGE_NO	Maximum value of D number
18106	MM_MAX_CUTTING_EDGE_PERTOOL	Maximum number of D numbers per tool
18108	MM_NUM_SUMCORR	Number of all sum offsets in the NC
18110	MM_MAX_SUMCORR_PER_CUTTEDGE	Number of sum offsets per cutting edge
18112	MM_KIND_OF_SUMCORR	Properties of sum offsets in the TO area (SRAM)
18114	MM_ENABLE_TOOL_ORIENT	Assign orientation to cutting edges
18116	MM_NUM_TOOL_ENV	Tool environments in the TOA area (SRAM)

19.18.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20096	T_M_ADDRESS_EXT_IS_SPINO	Meaning of the address extension with T, M tool change
20110	RESET_MODE_MASK	Definition of initial control setting after RESET/part program end

Number	Identifier: $MC_	Description
20120	TOOL_RESET_VALUE	Tool length compensation at power-up (Reset/TP end)
20121	TOOL_PRESEL_RESET_VALUE	Preselect tool on Reset
20125	CUTMOD_ERR	Fault handling for the function "Modification of the offset data for rotatable tools"
20126	TOOL_CARRIER_RESET_VALUE	Operative tool holder on Reset
20127	CUTMOD_INIT	Initialization value for CUTMOD
20130	CUTTING_EDGE_RESET_VALUE	Tool cutting edge length compensation at power-up (Reset/TP end)
20132	SUMCORR_RESET_VALUE	Additive offset effective on reset
20140	TRAFO_RESET_VALUE	Transformation data record at power up (Reset/TP end)
20180	TOCARR_ROT_ANGLE_INCR[i]	Rotary axis increment of the tool carrier with orientation capability
20182	TOCARR_ROT_ANGLE_OFFSET[i]	Rotary axis offset of tool carrier with orientation capa-bility
20184	TOCARR_BASE_FRAME_NUMBER	Number of the basic frames to accept the table offset
20188	TOCARR_FINE_LIM_LIN	Limit linear fine offset TCARR
20190	TOCARR_FINE_LIM_ROT	Limit of the rotary fine offset TCARR
20202	WAB_MAXNUM_DUMMY_BLOCKS	Maximum number of blocks with no traversing motions with SAR
20204	WAB_CLEARANCE_TOLERANCE	Direction reversal for WAB
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value for DISC
20230	CUTCOM_CURVE_INSERT_LIMIT	Maximum value for intersection calculation with TRC
20240	CUTCOM_MAXNUM_CHECK_BLOCKS	Blocks for predictive contour calculation with tool radius compensation
20250	CUTCOM_MAXNUM_DUMMY_BLOCKS	Maximum number of blocks without traversing motion for TRC
20252	CUTCOM_MAXNUM_SUPPR_BLOCKS	Maximum number of blocks with compensation sup-pression
20256	CUTCOM_INTERS_POLY_ENABLE	Intersection process possible for polynomials
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20272	SUMCORR_DEFAULT	Initial setting of additive offset without program
20360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
20390	TOOL_TEMP_COMP_ON	Activation of temperature compensation for tool length
20392	TOOL_TEMP_COMP_LIMIT	Maximum temperature compensation for tool length
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements
21080	CUTCOM_PARALLEL_ORI_LIMIT	Minimum angle (path tangent and tool orientation) for 3D tool radius compensation
22530	TOCARR_CHANGE_M_CODE	M code for change of tool carrier
22550	TOOL_CHANGE_MODE	New tool compensations with M function
22560	TOOL_CHANGE_M_CODE	M function for tool change
22562	TOOL_CHANGE_ERROR_MODE	Response when errors occur at tool change
24558	TRAFO5_JOINT_OFFSET_PART_1	Vector of kinematic offset in table, transformation 1

Number	Identifier: $MC_	Description
24658	TRAFO5_JOINT_OFFSET_PART_2	Vector of kinematic offset in table, transformation 2
28085	MM_LINK_TOA_UNIT	Assigning ´the TO unit to a channel (SRAM)

19.18.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32750	TEMP_COMP_TYPE	Temperature compensation type

19.18.2 Setting data

19.18.2.1 Channelspecific setting data

Number	Identifier $SC_	Description
42442	TOOL_OFFSET_INCR_PROG	Tool length compensation
42470	CRIT_SPLINE_ANGLE	Core limit angle, for compressor
42480	STOP_CUTCOM_STOPRE	Alarm response for tool radius compensation and pre-processing stop
42494	CUTCOM_ACT_DEACT_CTRL	Approach and retraction behavior for tool radius compensation
42496	CUTCOM_CLSDT_CONT	Behavior of the tool radius compensation for closed contour
42900	MIRROR_TOOL_LENGTH	Sign change, tool lengths when mirroring
42910	MIRROR_TOOL_WEAR	Sign change, tool wear when mirroring
42920	WEAR_SIGN_CUTPOS	Sign of wear for tools with cutting edge position
42930	WEAR_SIGN	Sign of the wear
42935	WEAR_TRANSFORM	Transformations for tool components
42940	TOOL_LENGTH_CONST	Change of tool length components for change of plane
42950	TOOL_LENGTH_TYPE	Assignment of the tool length offset independent of tool type
42960	TOOL_TEMP_COMP	Temperature compensation value in relation to tool
42974	TCARR_FINE_CORRECTION	Fine offset `TCARR` on/off
42984	CUTDIRMOD	Modification of $P_AD[2] or $P_AD[11]
42998	CUTMOD_PLANE_TOL	Difference between tool tip plane and machining plane for CUTMOD or CUTMODK
42999	ORISOLH_INCLINE_TOL	Difference between tool tip plane and machining plane for ORISOLH

19.18.3 Signals

19.18.3.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
T function 1 change	DB21, ...DBX61.0	-
D function 1 change	DB21, ...DBX62.0	-
T function 1	DB21, ...DBB116-119	DB250x.DBD2000
D function 1	DB21, ...DBB128-129	DB250x.DBD5000
Active G command of group 7	DB21, ...DBB214	DB350x.DBB6
Active G command of group 16	DB21, ...DBB223	DB350x.DBB15
Active G command of group 17	DB21, ...DBB224	DB350x.DBB16
Active G command of group 18	DB21, ...DBB225	DB350x.DBB17
Active G command of group 23	DB21, ...DBB230	DB350x.DBB22

Z1: NC/PLC interface signals 20

From Edition 05/2017, a detailed description of the NC/PLC interface signals is provided in the NC Variables and Interface Signals List Manual.

Appendix A

A.1 List of abbreviations

A	
O	Output
ADI4	(Analog drive interface for 4 axes)
AC	Adaptive Control
ALM	Active Line Module
ARM	Rotating induction motor
AS	Automation system
ASCII	American Standard Code for Information Interchange: American coding standard for the exchange of information
ASIC	Application-Specific Integrated Circuit: User switching circuit
ASUB	Asynchronous subprogram
AUXFU	Auxiliary function: Auxiliary function
STL	Statement List
UP	User Program

B	
OP	Operating Mode
BAG	Mode group
BCD	Binary Coded Decimals: Decimal numbers encoded in binary code
BERO	Contact-less proximity switch
BI	Binector Input
BICO	Binector Connector
BIN	BINary files: Binary files
BIOS	Basic Input Output System
BCS	Basic Coordinate System
BO	Binector Output
OPI	Operator Panel Interface

C	
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CC	Compile Cycle: Compile cycles
CEC	Cross Error Compensation
CI	Connector Input
CF Card	Compact Flash Card

C	
CNC	Computerized Numerical Control: Computer-Supported Numerical Control
CO	Connector Output
CoL	Certificate of License
COM	Communication
CPA	Compiler Projecting Data: Configuring data of the compiler
CRT	Cathode Ray Tube: picture tube
CSB	Central Service Board: PLC module
CU	Control Unit
CP	Communication Processor
CPU	Central Processing Unit: Central processing unit
CR	Carriage Return
CTS	Clear To Send: Ready to send signal for serial data interfaces
CUTCOM	Cutter radius Compensation: Tool radius compensation

D	
DAC	Digital-to-Analog Converter
DB	Data Block (PLC)
DBB	Data Block Byte (PLC)
DBD	Data Block Double word (PLC)
DBW	Data Block Word (PLC)
DBX	Data block bit (PLC)
DDE	Dynamic Data Exchange
DDS	Drive Data Set: Drive data set
DIN	Deutsche Industrie Norm
DIO	Data Input/Output: Data transfer display
DIR	Directory: Directory
DLL	Dynamic Link Library
DO	Drive Object
DPM	Dual Port Memory
DPR	Dual Port RAM
DRAM	Dynamic memory (non-buffered)
DRF	Differential Resolver Function: Differential revolver function (handwheel)
DRIVE-CLiQ	Drive Component Link with IQ
DRY	Dry Run: Dry run feedrate
DSB	Decoding Single Block: Decoding single block
DSC	Dynamic Servo Control / Dynamic Stiffness Control
DW	Data Word
DWORD	Double Word (currently 32 bits)

E	
I	Input
EES	Execution from External Storage
I/O	Input/Output
ENC	Encoder: Actual value encoder
EFP	Compact I/O module (PLC I/O module)
ESD	Electrostatic Sensitive Devices
EMC	ElectroMagnetic Compatibility
EN	European standard
ENC	Encoder: Actual value encoder
EnDat	Encoder interface
EPROM	Erasable Programmable Read Only Memory: Erasable, electrically programmable read-only memory
ePS Network Services	Services for Internet-based remote machine maintenance
EQN	Designation for an absolute encoder with 2048 sine signals per revolution
ES	Engineering System
ESR	Extended Stop and Retract
ETC	ETC key ">"; softkey bar extension in the same menu

F	
FB	Function Block (PLC)
FC	Function Call: Function Block (PLC)
FEPROM	Flash EPROM: Read and write memory
FIFO	First In First Out: Memory that works without address specification and whose data is read in the same order in which they was stored
FIPO	Fine interpolator
FPU	Floating Point Unit: Floating Point Unit
CRC	Cutter Radius Compensation
FST	Feed Stop: Feedrate stop
FBD	Function Block Diagram (PLC programming method)
FW	Firmware

G	
GC	Global Control (PROFIBUS: Broadcast telegram)
GDIR	Global part program memory
GEO	Geometry, e.g. geometry axis
GIA	Gear Interpolation dAta: Gear interpolation data
GND	Signal Ground
GP	Basic program (PLC)
GS	Gear Stage
GSD	Device master file for describing a PROFIBUS slave

G	
GSDML	Generic Station Description Markup Language: XML-based description language for creating a GSD file
GUD	Global User Data: Global user data

H	
HEX	Abbreviation for hexadecimal number
AuxF	Auxiliary function
HLA	Hydraulic linear drive
HMI	Human Machine Interface: SINUMERIK user interface
MSD	Main Spindle Drive
HW	Hardware

I	
IBN	Commissioning
ICA	Interpolatory compensation
IM	Interface Module: Interconnection module
IMR	Interface Module Receive: Interface module for receiving data
IMS	Interface Module Send: Interface module for sending data
INC	Increment: Increment
INI	Initializing Data: Initializing data
IPO	Interpolator
ISA	Industry Standard Architecture
ISO	International Standardization Organization

J	
JOG	Jogging: Setup mode

K	
K_V	Gain factor of control loop
K_p	Proportional gain
$K_Ü$	Transformation ratio
LAD	Ladder Diagram (PLC programming method)

L	
LAI	Logic Machine Axis Image: Logical machine axes image
LAN	Local Area Network
LCD	Liquid Crystal Display: Liquid crystal display
LED	Light Emitting Diode: Light-emitting diode
LF	Line Feed

L	
PMS	Position Measuring System
LR	Position controller
LSB	Least Significant Bit: Least significant bit
LUD	Local User Data: User data (local)

M	
MAC	Media Access Control
MAIN	Main program: Main program (OB1, PLC)
MB	Megabyte
MCI	Motion Control Interface
MCIS	Motion Control Information System
MCP	Machine Control Panel: Machine control panel
MD	Machine Data
MDA	Manual Data Automatic: Manual input
MDS	Motor Data Set: Motor data set
MSGW	Message Word
MCS	Machine Coordinate System
MM	Motor Module
MPF	Main Program File: Main program (NC)
MCP	Machine control panel

N	
NC	Numerical Control: Numerical control with block preparation, traversing range, etc.
NCU	Numerical Control Unit: NC hardware unit
NRK	Name for the operating system of the NC
IS	Interface Signal
NURBS	Non-Uniform Rational B-Spline
WO	Work Offset
NX	Numerical Extension: Axis expansion board

O	
OB	Organization block in the PLC
OEM	Original Equipment Manufacturer
OP	Operator Panel: Operating equipment
OPI	Operator Panel Interface: Interface for connection to the operator panel
OPT	Options: Options
OLP	Optical Link Plug: Fiber optic bus connector
OSI	Open Systems Interconnection: Standard for computer communications

P	
PIQ	Process Image Output
PII	Process Image Input
PC	Personal Computer
PCIN	Name of the SW for data exchange with the control
PCMCIA	Personal Computer Memory Card International Association: Plug-in memory card standardization
PCU	PC Unit: PC box (computer unit)
PG	Programming device
PKE	Parameter identification: Part of a PIV
PIV	Parameter identification: Value (parameterizing part of a PPO)
PLC	Programmable Logic Control: Adaptation control
PN	PROFINET
PNO	PROFIBUS user organization
PO	POWER ON
POU	Program Organization Unit
POS	Position/positioning
POSMO A	Positioning Motor Actuator: Positioning motor
POSMO CA	Positioning Motor Compact AC: Complete drive unit with integrated power and control module as well as positioning unit and program memory; AC infeed
POSMO CD	Positioning Motor Compact DC: Like CA but with DC infeed
POSMO SI	Positioning Motor Servo Integrated: Positioning motor, DC infeed
PPO	Parameter Process data Object: Cyclic data telegram for PROFIBUS DP transmission and "Variable speed drives" profile
PPU	Panel Processing Unit (central hardware for a panel-based CNC, e.g SINUMERIK 828D)
PROFIBUS	Process Field Bus: Serial data bus
PRT	Program Test
PSW	Program control word
PTP	Point-To-Point: Point-To-Point
PUD	Program global User Data: Program-global user variables
PZD	Process data: Process data part of a PPO

Q	
QEC	Quadrant Error Compensation

R	
RAM	Random Access Memory: Read/write memory
REF	REFerence point approach function
REPOS	REPOSition function
RISC	Reduced Instruction Set Computer: Type of processor with small instruction set and ability to process instructions at high speed
ROV	Rapid Override: Input correction

R	
RP	R Parameter, arithmetic parameter, predefined user variable
RPA	R Parameter Active: Memory area in the NC for R parameter numbers
RPY	Roll Pitch Yaw: Rotation type of a coordinate system
RTLI	Rapid Traverse Linear Interpolation: Linear interpolation during rapid traverse motion
RTS	Request To Send: Control signal of serial data interfaces
RTCP	Real Time Control Protocol

S	
SA	Synchronized Action
SBC	Safe Brake Control: Safe Brake Control
SBL	Single Block: Single block
SBR	Subroutine: Subprogram (PLC)
SD	Setting Data
SDB	System Data Block
SEA	Setting Data Active: Identifier (file type) for setting data
SERUPRO	SEarch RUn by PROgram test: Block search, program test
SFB	System Function Block
SFC	System Function Call
SGE	Safety-related input
SGA	Safety-related output
SH	Safe standstill
SIM	Single in Line Module
SK	Softkey
SKP	Skip: Function for skipping a part program block
SLM	Synchronous Linear Motor
SM	Stepper Motor
SMC	Sensor Module Cabinet Mounted
SME	Sensor Module Externally Mounted
SMI	Sensor Module Integrated
SPF	Sub Routine File: Subprogram (NC)
PLC	Programmable Logic Controller
SRAM	Static RAM (non-volatile)
TNRC	Tool Nose Radius Compensation
SRM	Synchronous Rotary Motor
LEC	Leadscrew Error Compensation
SSI	Serial Synchronous Interface: Synchronous serial interface
SSL	Block search
STW	Control word
GWPS	Grinding Wheel Peripheral Speed
SW	Software
SYF	System Files: System files
SYNACT	SYNchronized ACTion: Synchronized Action

T	
TB	Terminal Board (SINAMICS)
TCP	Tool Center Point: Tool tip
TCP/IP	Transport Control Protocol / Internet Protocol
TCU	Thin Client Unit
TEA	Testing Data Active: Identifier for machine data
TIA	Totally Integrated Automation
TM	Terminal Module (SINAMICS)
TO	Tool Offset: Tool offset
TOA	Tool Offset Active: Identifier (file type) for tool offsets
TRANSMIT	Transform Milling Into Turning: Coordination transformation for milling operations on a lathe
TTL	Transistor-Transistor Logic (interface type)
TZ	Technology cycle

U	
UFR	User Frame: Work offset
SR	Subprogram
USB	Universal Serial Bus
UPS	Uninterruptible Power Supply

V	
VDI	Internal communication interface between NC and PLC
VDI	Verein Deutscher Ingenieure [Association of German Engineers]
VDE	Verband Deutscher Elektrotechniker [Association of German Electrical Engineers]
VI	Voltage Input
VO	Voltage Output
FDD	Feed Drive

W	
SAR	Smooth Approach and Retraction
WCS	Workpiece Coordinate System
T	Tool
TLC	Tool Length Compensation
WOP	Workshop-Oriented Programming
WPD	Workpiece Directory: Workpiece directory
TRC	Tool Radius Compensation
T	Tool
TO	Tool Offset
TM	Tool Management
TC	Tool change

X	
XML	Extensible Markup Language

Z	
WOA	Work Offset Active: Identifier for work offsets
ZSW	Status word (of drive)

A.2 Documentation overview

General documentation

SINUMERIK

Advertising brochure
- SINUMERIK 840D sl
– SINUMERIK 828D
– SINUMERIK 828D BASIC

Catalog NC 62
SINUMERIK 840D sl

Catalog NC 82
SINUMERIK 828D

Catalog PM 21
SIMOTION,
SINAMICS S120

SINUMERIK
840D sl

System Manual
Intelligent machine
tool operation

SINUMERIK
840D sl
828D

System Manual
Ctrl-Energy

User documentation

SINUMERIK
840D sl
828D

Operating Manual
– Universal
– Turning
– Milling
– Grinding

SINUMERIK
840D sl
828D

Programming Manual
– Fundamentals
– Job Planning
– Measuring Cycles

SINUMERIK
840D sl
828D

Programming Manual
– ISO Turning
– ISO Milling

SINUMERIK
840D sl
SINAMICS
S120

Diagnostics Manual
Alarms

SINUMERIK
828D
SINAMICS
S120

Diagnostics Manual
Alarms

Manufacturer/service documentation

SINUMERIK
840D sl

Equipment Manual
– NCU
– Operator components and networking
– ADI4

SINUMERIK
828D

Equipment Manual
Commissioning Manual
Service Manual

SINUMERIK
840D sl

Commissioning Manual
– CNC: NCK, PLC, Drive
– Base software and operating software

SINUMERIK
840D sl
SINAMICS
S120

Lists Manual
– Machine data
– Interface signals
– Variables

SINUMERIK
828D
SINAMICS
S120

Lists Manual
– Machine data
– Interface signals
– Parameters
– Variables

SINUMERIK
840D sl
SINAMICS
S120

System Manual
Guidelines for configuring machines

Manufacturer/service documentation

SINUMERIK
840D sl
828D

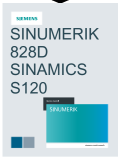
Function Manual
– Basic Functions
– Extended Functions
– Special Functions
– Synchronized Actions
– ISO Dialects

SINUMERIK
840D sl

Function Manual
Tool management

SINUMERIK
840D sl
SINAMICS
S120

Function Manual
Safety Integrated


SINUMERIK
828D
SINAMICS
S120

Function Manual
Safety Integrated

SINUMERIK

Configuration Manual
– EMC installation guideline
– Industrial security

Info/training

SINUMERIK

Training documentation
– Simple milling with ShopMill
– Simple turning with ShopTurn

SINUMERIK

Manual
Tool and mold making

Electronic documentation

DOConCD

Industry Online Support (SIOS)

Industry Mall

Glossary

Absolute dimensions

A destination for an axis motion is defined by a dimension that refers to the origin of the currently valid coordinate system. See → Incremental dimension

Acceleration with jerk limitation

In order to optimize the acceleration response of the machine whilst simultaneously protecting the mechanical components, it is possible to switch over in the machining program between abrupt acceleration and continuous (jerk-free) acceleration.

Address

An address is the identifier for a certain operand or operand range, e.g. input, output, etc.

Alarms

All → messages and alarms are displayed on the operator panel in plain text with date and time and the corresponding symbol for the deletion criterion. Alarms and messages are displayed separately.

1. Alarms and messages in the part program:
   Alarms and messages can be displayed in plain text directly from the part program.
2. Alarms and messages from the PLC:
   Alarms and messages for the machine can be displayed in plain text from the PLC program. No additional function block packages are required for this purpose.

Archiving

Reading out of files and/or directories on an **external** memory device.

Asynchronous subprogram

Part program that can be started asynchronously to (independently of) the current program status using an interrupt signal (e.g. "Rapid NC input" signal).

Automatic

Operating mode of the controller (block sequence operation according to DIN): Operating mode for NC systems in which a → subprogram is selected and executed continuously.

Auxiliary functions

Auxiliary functions enable → part programs to transfer → parameters to the → PLC, which then trigger reactions defined by the machine manufacturer.

Axes

In accordance with their functional scope, the CNC axes are subdivided into:

- Axes: Interpolating path axes
- Auxiliary axes: Non-interpolating feed and positioning axes with an axis-specific feedrate. Auxiliary axes are not involved in actual machining, e.g. tool feeder, tool magazine.

Axis address

See → Axis name

Axis name

To ensure clear identification, all channel and → machine axes of the control system must be designated with unique names in the channel and control system. The → geometry axes are called X, Y, Z. The rotary axes rotating around the geometry axes → are called A, B, C.

Backlash compensation

Compensation for a mechanical machine backlash, e.g. backlash on reversal for ball screws. Backlash compensation can be entered separately for each axis.

Backup battery

The backup battery ensures that the → user program in the → CPU is stored so that it is safe from power failure and so that specified data areas and bit memory, timers and counters are stored retentively.

Basic axis

Axis whose setpoint or actual value position forms the basis of the calculation of a compensation value.

Basic Coordinate System

Cartesian coordinate system which is mapped by transformation onto the machine coordinate system.

The programmer uses axis names of the basic coordinate system in the → part program. The basic coordinate system exists parallel to the → machine coordinate system if no → transformation is active. The difference lies in the → axis names.

Baud rate

Rate of data transfer (bits/s).

Blank

Workpiece as it is before it is machined.

Block

"Block" is the term given to any files required for creating and processing programs.

Block search

For debugging purposes or following a program abort, the "Block search" function can be used to select any location in the part program at which the program is to be started or resumed.

Booting

Loading the system program after power ON.

C axis

Axis around which the tool spindle describes a controlled rotational and positioning motion.

C spline

The C spline is the most well-known and widely used spline. The transitions at the interpolation points are continuous, both tangentially and in terms of curvature. 3rd order polynomials are used.

Channel

A channel is characterized by the fact that it can process a → part program independently of other channels. A channel exclusively controls the axes and spindles assigned to it. Part program runs of different channels can be coordinated through → synchronization.

Circular interpolation

The → tool moves on a circle between specified points on the contour at a given feedrate, and the workpiece is thereby machined.

CNC

See → NC

Computerized Numerical Control: includes the components → NC, → PLC, HMI, → COM.

CNC

See → NC

Computerized Numerical Control: includes the components → NC, → PLC, HMI, → COM.

COM

Component of the NC for the implementation and coordination of communication.

Compensation axis

Axis with a setpoint or actual value modified by the compensation value

Compensation table

Table containing interpolation points. It provides the compensation values of the compensation axis for selected positions on the basic axis.

Compensation value

Difference between the axis position measured by the encoder and the desired, programmed axis position.

Continuous-path mode

The objective of continuous-path mode is to avoid substantial deceleration of the → path axes at the part program block boundaries and to change to the next block at as close to the same path velocity as possible.

Contour

Contour of the → workpiece

Contour monitoring

The following error is monitored within a definable tolerance band as a measure of contour accuracy. An unacceptably high following error can cause the drive to become overloaded, for example. In such cases, an alarm is output and the axes are stopped.

Coordinate system

See → Machine coordinate system, → Workpiece coordinate system

CPU

Central processing unit, see → PLC

CU

Transformation ratio

Curvature

The curvature k of a contour is the inverse of radius r of the nestling circle in a contour point (k = 1/r).

Cycles

Protected subprograms for execution of repetitive machining operations on the → workpiece.

Data block

1. Data unit of the → PLC that → HIGHSTEP programs can access.
2. Data unit of the → NC: Data blocks contain data definitions for global user data. This data can be initialized directly when it is defined.

Data word

Two-byte data unit within a → data block.

Diagnostics

1. Operating area of the control.
2. The control has a self-diagnostics program as well as test functions for servicing purposes: status, alarm, and service displays

Dimensions specification, metric and inches

Position and pitch values can be programmed in inches in the machining program. Irrespective of the programmable dimensions (`G70`/`G71`), the control is set to a basic system.

DRF

Differential Resolver Function: NC function which generates an incremental work offset in Automatic mode in conjunction with an electronic handwheel.

Drive

The drive is the unit of the CNC that performs the speed and torque control based on the settings of the NC.

Dynamic feedforward control

Inaccuracies in the → contour due to following errors can be practically eliminated using dynamic, acceleration-dependent feedforward control. This results in excellent machining accuracy even at high → path velocities. Feedforward control can be selected and deselected on an axis-specific basis via the → part program.

Editor

The editor makes it possible to create, edit, extend, join, and import programs / texts / program blocks.

Exact stop

When an exact stop statement is programmed, the position specified in a block is approached exactly and, if necessary, very slowly. To reduce the approach time, → exact stop limits are defined for rapid traverse and feed.

Exact stop limit

When all path axes reach their exact stop limits, the control responds as if it had reached its precise destination point. A block advance of the → part program occurs.

External work offset

Work offset specified by the → PLC.

Fast retraction from the contour

When an interrupt occurs, a motion can be initiated via the CNC machining program, enabling the tool to be quickly retracted from the workpiece contour that is currently being machined. The retraction angle and the distance retracted can also be parameterized. An interrupt routine can also be executed following the fast retraction.

Feed override

The programmed velocity is overriden by the current velocity setting made via the → machine control panel or from the → PLC (0 to 200%). The feedrate can also be corrected by a programmable percentage factor (1 to 200%) in the machining program.

Finished-part contour

Contour of the finished workpiece. See → Raw part.

Fixed machine point

Point that is uniquely defined by the machine tool, e.g. machine reference point.

Fixed-point approach

Machine tools can approach fixed points such as a tool change point, loading point, pallet change point, etc. in a defined way. The coordinates of these points are stored in the control. The control moves the relevant axes in → rapid traverse, whenever possible.

Frame

A frame is an arithmetic rule that transforms one Cartesian coordinate system into another Cartesian coordinate system. A frame contains the following components: → work offset, → rotation, → scaling, → mirroring.

Functionality

The path-jerk limitation can be activated/deactivated by programming the setting data.

Parameter: *Value*

- Value range: TRUE, FALSE

Application:

- Part program
- Static synchronized action

Geometry

Description of a → workpiece in the → workpiece coordinate system.

Geometry axis

The geometry axes form the 2 or 3-dimensional → workpiece coordinate system in which, in → part programs, the geometry of the workpiece is programmed.

Ground

Ground is taken as the total of all linked inactive parts of a device which will not become live with a dangerous contact voltage even in the event of a malfunction.

Helical interpolation

The helical interpolation function is ideal for machining internal and external threads using form milling cutters and for milling lubrication grooves.

The helix comprises two motions:

- Circular motion in one plane
- A linear motion perpendicular to this plane

High-level CNC language

The high-level language is used to write NC programs, → synchronized actions, and → cycles. It provides: control structures → user-defined variables, → system variables, → macro programming.

High-speed digital inputs/outputs

The digital inputs can be used for example to start fast CNC program routines (interrupt routines). High-speed, program-driven switching functions can be initiated via the digital CNC outputs

HIGHSTEP

Summary of programming options for → PLCs of the AS300/AS400 system.

HW Config

SIMATIC S7 tool for the configuration and parameterization of hardware components within an S7 project

Identifier

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subprograms, key words, and words with multiple address letters. These supplements have the same meaning as the words with respect to block format. Identifiers must be unique. It is not permissible to use the same identifier for different objects.

Inch measuring system

Measuring system which defines distances in inches and fractions of inches.

Inclined surface machining

Drilling and milling operations on workpiece surfaces that do not lie in the coordinate planes of the machine can be performed easily using the function "inclined-surface machining".

Increment

Travel path length specification based on number of increments. The number of increments can be stored as → setting data or be selected by means of a suitably labeled key (i.e. 10, 100, 1000, 10000).

Incremental dimension

Incremental dimension: A destination for axis traversal is defined by a distance to be covered and a direction referenced to a point already reached. See → Absolute dimension.

Intermediate blocks

Motions with selected → tool offset (`G41`/`G42`) may be interrupted by a limited number of intermediate blocks (blocks without axis motions in the offset plane), whereby the tool offset can still be correctly compensated for. The permissible number of intermediate blocks which the controller reads ahead can be set in system parameters.

Interpolator

Logic unit of the → NC that defines intermediate values for the motion to be carried out in individual axes based on information on the end positions specified in the part program.

Interpolatory compensation

Mechanical deviations of the machine are compensated for by means of interpolatory compensation functions, such as → leadscrew error, sag, angularity, and temperature compensation.

Interrupt routine

Interrupt routines are special → subprograms that can be started by events (external signals) in the machining process. A part program block which is currently being worked through is interrupted and the position of the axes at the point of interruption is automatically saved.

Inverse-time feedrate

The time required for the path of a block to be traversed can also be programmed for the axis motion instead of the feed velocity (G93).

JOG

Operating mode of the control (setup mode): The machine can be set up in JOG mode. Individual axes and spindles can be traversed in JOG mode by means of the direction keys. Additional functions in JOG mode include: → Reference point approach, → Repos, and → Preset (set actual value).

Key switch

The key switch on the → machine control panel has four positions that are assigned functions by the operating system of the controller. The key switch has three different colored keys that can be removed in the specified positions.

Keywords

Words with specified notation that have a defined meaning in the programming language for → part programs.

KV

Servo gain factor, a control variable in a control loop.

Leading axis

The leading axis is the → gantry axis that exists from the point of view of the operator and programmer and, thus, can be influenced like a standard NC axis.

Leadscrew error compensation

Compensation for the mechanical inaccuracies of a leadscrew participating in the feed. The controller uses stored deviation values for the compensation.

Limit speed

Maximum/minimum (spindle) speed: The maximum speed of a spindle can be limited by specifying machine data, the → PLC or → setting data.

Linear axis

In contrast to a rotary axis, a linear axis describes a straight line.

Linear interpolation

The tool travels along a straight line to the destination point while machining the workpiece.

Load memory

The load memory is the same as the → working memory for the CPU 314 of the → PLC.

Look Ahead

The **Look Ahead** function is used to achieve an optimal machining speed by looking ahead over an assignable number of traversing blocks.

Machine axes

Physically existent axes on the machine tool.

Machine control panel

An operator panel on a machine tool with operating elements such as keys, rotary switches, etc., and simple indicators such as LEDs. It is used to directly influence the machine tool via the PLC.

Machine coordinate system

A coordinate system, which is related to the axes of the machine tool.

Machine zero

Fixed point of the machine tool to which all (derived) measuring systems can be traced back.

Machining channel

A channel structure can be used to shorten idle times by means of parallel motion sequences, e.g. moving a loading gantry simultaneously with machining. Here, a CNC channel must be regarded as a separate CNC control system with decoding, block preparation and interpolation.

Macro techniques

Grouping of a set of statements under a single identifier. The identifier represents the set of consolidated statements in the program.

Main block

A block preceded with ":" that contains all information to start the operating sequence in a → part program.

Main program

The term "main program" has its origins during the time when part programs were split strictly into main and → subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program in the channel can be selected and started. It then runs through in → program level 0 (main program level). Further part programs or → cycles as subprograms can be called up in the main program.

MDI

Operating mode of the control: Manual Data Input. In the MDI mode, individual program blocks or block sequences with no reference to a main program or subprogram can be input and executed immediately afterwards through actuation of the NC start key.

Messages

All messages programmed in the part program and → alarms detected by the system are displayed on the operator panel in plain text with date and time and the corresponding symbol for the deletion criterion. Alarms and messages are displayed separately.

Metric measuring system

Standardized system of units: For length, e.g. mm (millimeters), m (meters).

Mirroring

Mirroring reverses the signs of the coordinate values of a contour, with respect to an axis. It is possible to mirror with respect to more than one axis at a time.

Mode

An operating concept on a SINUMERIK control The following modes are defined: → Jog, → MDI, → Automatic.

Mode group

Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a mode group can be controlled by one or more → channels. The same → mode type is always assigned to the channels of the mode group.

NC

Numerical Control component of the → CNC that executes the → part programs and coordinates the movements of the machine tool.

Network

A network is the connection of multiple S7-300 and other end devices, e.g. a programming device via a → connecting cable. A data exchange takes place over the network between the connected devices.

NRK

Numeric robotic kernel (operating system of → NC)

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). This provides a uniform procedure for all internal interpolations.

OEM

The scope for implementing individual solutions (OEM applications) has been provided for machine manufacturers, who wish to create their own user interface or integrate technology-specific functions in the control.

Offset memory

Data range in the control, in which the tool offset data is stored.

Oriented spindle stop

Stops the workpiece spindle in a specified angular position, e.g. in order to perform additional machining at a particular location.

Overall reset

In the event of an overall reset, the following memories of the → CPU are deleted:

- → Working memory
- Read/write area of → load memory
- → System memory
- → Backup memory

Override

Manual or programmable possibility of intervention that enables the user to override programmed feedrates or speeds in order to adapt them to a specific workpiece or material.

Part program

Series of statements to the NC that act in concert to produce a particular → workpiece. Likewise, this term applies to execution of a particular machining operation on a given → raw part.

Part program block

Part of a → part program that is demarcated by a line feed. There are two types: → main blocks and → subblocks.

Part program management

Part program management can be organized by → workpieces. The size of the user memory determines the number of programs and the amount of data that can be managed. Each file (programs and data) can be given a name consisting of a maximum of 24 alphanumeric characters.

Path axis

Path axes include all machining axes of the → channel that are controlled by the → interpolator in such a way that they start, accelerate, stop, and reach their end point simultaneously.

Path feedrate

Path feedrate affects → path axes. It represents the geometric sum of the feedrates of the → geometry axes involved.

Path velocity

The maximum programmable path velocity depends on the input resolution. For example, with a resolution of 0.1 mm the maximum programmable path velocity is 1000 m/min.

PCIN data transfer program

PCIN is a utility program for sending and receiving CNC user data (e.g. part programs, tool offsets) via the serial interface. The PCIN program can run under MS-DOS on standard industrial PCs.

Peripheral module

I/O modules represent the link between the CPU and the process.

I/O modules are:

- → Digital input/output modules
- → Analog input/output modules
- → Simulator modules

PLC

Programmable **L**ogic **C**ontroller: → Programmable logic controller. Component of → NC: Programmable control for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D sl: The PLC user program, the user data and the basic PLC program are stored together in the PLC user memory.

PLC programming

The PLC is programmed using the **STEP 7** software. The STEP 7 programming software is based on the **WINDOWS** standard operating system and contains the STEP 5 programming functions with innovative enhancements.

Polar coordinates

A coordinate system which defines the position of a point on a plane in terms of its distance from the origin and the angle formed by the radius vector with a defined axis.

Polynomial interpolation

Polynomial interpolation enables a wide variety of curve characteristics to be generated, such as **straight line, parabolic, exponential functions** (SINUMERIK 840D sl).

Positioning axis

Axis that performs an auxiliary motion on a machine tool (e.g. tool magazine, pallet transport). Positioning axes are axes that do not interpolate with → path axes.

Pre-coincidence

Block change occurs already when the path distance approaches an amount equal to a specifiable delta of the end position.

Program block

Program blocks contain the main program and subprograms of → part programs.

Program level

A part program started in the channel runs as a → main program on program level 0 (main program level). Any part program called up in the main program runs as a → subprogram on a program level 1 ... n of its own.

Programmable frames

Programmable → frames enable dynamic definition of new coordinate system output points while the part program is being executed. A distinction is made between absolute definition using a new frame and additive definition with reference to an existing starting point.

Programmable logic controller

Programmable logic controllers (PLCs) are electronic controllers, the function of which is stored as a program in the control unit. This means that the layout and wiring of the device do not depend on the function of the controller. The programmable logic control has the same structure as a computer; it consists of a CPU (central module) with memory, input/output modules and an internal bus system. The peripherals and the programming language are matched to the requirements of the control technology.

Programmable working area limitation

Limitation of the motion space of the tool to a space defined by programmed limitations.

Programming key

Characters and character strings that have a defined meaning in the programming language for → part programs.

Protection zone

Three-dimensional zone within the → working area into which the tool tip must not pass.

Quadrant error compensation

Contour errors at quadrant transitions, which arise as a result of changing friction conditions on the guideways, can be virtually entirely eliminated with the quadrant error compensation. Parameterization of the quadrant error compensation is performed by means of a circuit test.

R parameters

Arithmetic parameter that can be set or queried by the programmer of the → part program for any purpose in the program.

Rapid traverse

The highest traverse velocity of an axis. It is used, for example, 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 item.

Reference point

Machine tool position that the measuring system of the → machine axes references.

Rotary axis

Rotary axes apply a workpiece or tool rotation to a defined angular position.

Rotation

Component of a → frame that defines a rotation of the coordinate system around a particular angle.

Rounding axis

Rounding axes rotate a workpiece or tool to an angular position corresponding to an indexing grid. When a grid index is reached, the rounding axis is "in position".

RS-232-C

Serial interface for data input/output. Machining programs as well as manufacturer and user data can be loaded and saved via this interface.

Safety functions

The controller is equipped with permanently active monitoring functions that detect faults in the → CNC, the → PLC, and the machine in a timely manner so that damage to the workpiece, tool, or machine is largely prevented. In the event of a fault, the machining operation is interrupted and the drives stopped. The cause of the malfunction is logged and output as an alarm. At the same time, the PLC is notified that a CNC alarm has been triggered.

Scaling

Component of a → frame that implements axis-specific scale modifications.

Setting data

Data which communicates the properties of the machine tool to the NC as defined by the system software.

Softkey

A key, whose name appears on an area of the screen. The choice of softkeys displayed is dynamically adapted to the operating situation. The freely assignable function keys (softkeys) are assigned defined functions in the software.

Software limit switch

Software limit switches limit the traversing range of an axis and prevent an abrupt stop of the slide at the hardware limit switch. Two value pairs can be specified for each axis and activated separately by means of the → PLC.

Spline interpolation

With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

Standard cycles

Standard cycles are provided for machining operations which are frequently repeated:

- For the drilling/milling technology
- For turning technology

The available cycles are listed in the "Cycle support" menu in the "Program" operating area. Once the desired machining cycle has been selected, the parameters required for assigning values are displayed in plain text.

Subblock

Block preceded by "N" containing information for a sequence, e.g. positional data.

Subprogram

The term "subprogram" has its origins during the time when part programs were split strictly into →main and subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program or any → cycle can be called up as a subprogram within another part program. It then runs through in the next → program level (x+1) (subprogram level (x+1)).

Synchronization

Statements in → part programs for coordination of sequences in different → channels at certain machining points.

Synchronized actions

1. Auxiliary function output
   During workpiece machining, technological functions (→ auxiliary functions) can be output from the CNC program to the PLC. For example, these auxiliary functions are used to control additional equipment for the machine tool, such as quills, grabbers, clamping chucks, etc.
2. Fast auxiliary function output
   For time-critical switching functions, the acknowledgement times for the → auxiliary functions can be minimized and unnecessary hold points in the machining process can be avoided.

Synchronized axes

Synchronized axes take the same time to traverse their path as the geometry axes take for their path.

Synchronized axis

A synchronized axis is the → gantry axis whose set position is continuously derived from the motion of the → leading axis and is, thus, moved synchronously with the leading axis. From the point of view of the programmer and operator, the synchronized axis "does not exist".

Syntax

$SC_IS_SD_MAX_PATH_JERK = *value*

System memory

The system memory is a memory in the CPU in which the following data is stored:

- Data required by the operating system
- The operands timers, counters, markers

System variable

A variable that exists without any input from the programmer of a → part program. It is defined by a data type and the variable name preceded by the character **$**. See → User-defined variable.

Tapping without compensating chuck

This function allows threads to be tapped without a compensating chuck. By using the interpolating method of the spindle as a rotary axis and the drilling axis, threads can be cut to a precise final drilling depth, e.g. for blind hole threads (requirement: spindles in axis operation).

Text editor

See → Editor

TOA area

The TOA area includes all tool and magazine data. By default, this area coincides with the → channel area with regard to the access of the data. However, machine data can be used to specify that multiple channels share one → TOA unit so that common tool management data is then available to these channels.

TOA unit

Each → TOA area can have more than one TOA unit. The number of possible TOA units is limited by the maximum number of active → channels. A TOA unit includes exactly one tool data block and one magazine data block. In addition, a TOA unit can also contain a toolholder data block (optional).

Tool

Active part on the machine tool that implements machining (e.g. turning tool, milling tool, drill, LASER beam, etc.).

Tool nose radius compensation

Contour programming assumes that the tool is pointed. Because this is not actually the case in practice, the curvature radius of the tool used must be communicated to the controller which then takes it into account. The curvature center is maintained equidistantly around the contour, offset by the curvature radius.

Tool offset

Consideration of the tool dimensions in calculating the path.

Tool radius compensation

To directly program a desired → workpiece contour, the control must traverse an equistant path to the programmed contour taking into account the radius of the tool that is being used (`G41`/`G42`).

Transformation

Additive or absolute zero offset of an axis.

Travel range

The maximum permissible travel range for linear axes is ± 9 decades. The absolute value depends on the selected input and position control resolution and the unit of measurement (inch or metric).

User interface

The user interface (UI) is the display medium for a CNC in the form of a screen. It features horizontal and vertical softkeys.

User memory

All programs and data, such as part programs, subprograms, comments, tool offsets, and work offsets / frames, as well as channel and program user data, can be stored in the shared CNC user memory.

User program

User programs for the S7-300 automation systems are created using the programming language STEP 7. The user program has a modular layout and consists of individual blocks.

The basic block types are:

- Code blocks
  These blocks contain the STEP 7 commands.
- Data blocks
  These blocks contain constants and variables for the STEP 7 program.

User-defined variable

Users can declare their own variables for any purpose in the → part program or data block (global user data). A definition contains a data type specification and the variable name. See → System variable.

Variable definition

A variable definition includes the specification of a data type and a variable name. The variable names can be used to access the value of the variables.

Velocity control

In order to achieve an acceptable traverse rate in the case of very slight motions per block, an anticipatory evaluation over several blocks (→ Look Ahead) can be specified.

WinSCP

WinSCP is a freely available open source program for Windows for the transfer of files.

Work offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a → frame.

1. Settable
   A configurable number of settable work offsets are available for each CNC axis. The offsets - which are selected by means of G commands - take effect alternatively.
2. External
   In addition to all the offsets which define the position of the workpiece zero, an external work offset can be overridden by means of the handwheel (DRF offset) or from the PLC.
3. Programmable
   Work offsets can be programmed for all path and positioning axes using the `TRANS` statement.

Working area

Three-dimensional zone into which the tool tip can be moved on account of the physical design of the machine tool. See → Protection zone.

Working area limitation

With the aid of the working area limitation, the traversing range of the axes can be further restricted in addition to the limit switches. One value pair per axis may be used to describe the protected working area.

Working memory

The working memory is a RAM in the → CPU that the processor accesses when processing the application program.

Workpiece

Part to be made/machined by the machine tool.

Workpiece contour

Set contour of the → workpiece to be created or machined.

Workpiece coordinate system

The workpiece coordinate system has its starting point in the → workpiece zero-point. In machining operations programmed in the workpiece coordinate system, the dimensions and directions refer to this system.

Workpiece zero

The workpiece zero is the starting point for the → workpiece coordinate system. It is defined in terms of distances to the → machine zero.

Index