SIEMENS

SINUMERIK

SINUMERIK 828D NC programming

Programming Manual

Valid for

controller
SINUMERIK 828D

CNC software   version 5.22

07/2023
A5E48764001B AF

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
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E48764001B AF
Ⓟ 06/2023 Subject to change

Table of contents

1 Introduction

1.1 About SINUMERIK

From simple, standardized CNC machines to premium modular machine designs – the SINUMERIK CNCs offer the right solution for all machine concepts. Whether for individual parts or mass production, simple or complex workpieces – SINUMERIK is the highly dynamic automation solution, integrated for all areas of production. From prototype construction and tool design to mold making, all the way to large-scale series production.

Visit our website for more information SINUMERIK (https://www.siemens.com/sinumerik).

1.2 About this documentation

This documentation is part of the documentation category of SINUMERIK Programming Manuals.

"NC Programming" Programming Manual

The "NC Programming" Programming Manual contains all information relevant to the programming of NC functions of a SINUMERIK controller.

Programmers and configuration engineers are the target group.

Using the Programming Manual, the target group can develop, write, test, and debug programs and software user interfaces.

Overview of contents

The overview of contents on the title page shows that the "NC Programming" Programming Manual is divided into two main chapters:

1. **Fundamentals**
   The main chapter "Fundamentals" is intended for use by skilled machine operators with the appropriate expertise in drilling, milling and turning operations. Simple programming examples are used to explain the commands and statements, which are also defined according to DIN 66025.
2. **Work preparation**
   The main chapter "Work preparation" is intended for use by technicians with in-depth, comprehensive programming knowledge. By virtue of a special programming language, the SINUMERIK control enables the user to program complex workpiece programs (e.g. for free-form surfaces, channel coordination, ...), and makes the programming of complicated operations easier for technologists.

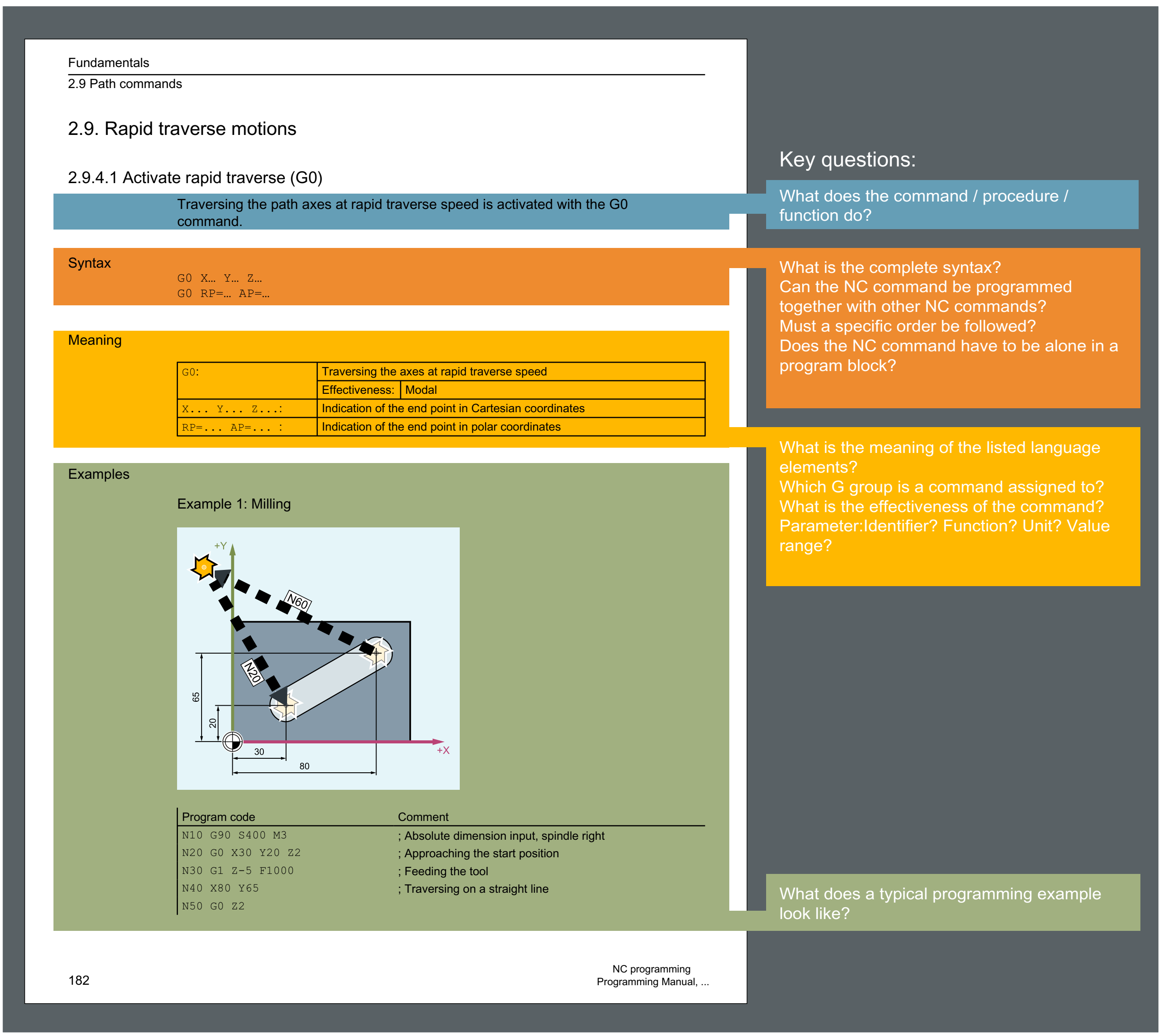
Validity

The title page also contains all information on the validity of a document, i.e. for which SINUMERIK control and for which software version this edition of the Programming Manual is valid.

Chapter structure

The descriptions of the NC language elements (G command, procedure, function, etc.) are structured in a uniform way; the contents are created under consideration of predefined key questions.

The following figure is to illustrate this with an example chapter:



In addition to the items Syntax, Meaning and Example(s) shown in the example, which are always present, there may be other items as well:

Additional items	Use
Requirements	If certain prerequisites must be met for the application of the programmable function (e.g. licensing of an option).
Effectiveness	If the programmable function is only effective under certain conditions (e.g. only in a specific operating mode).
Constraints	If interactions with other functions are to be considered during programming.
More information	If further explanation is needed to understand the programming of the function.

Notes on the "Syntax" subitem

Under the heading "Syntax" the user will find information on how to program the NC language elements and parameters described in the respective chapter in order to obtain a valid NC program.

The program lines given here follow a generally valid notation, which is only intended to help in understanding the rules to be considered in programming. Therefore, they may contain elements which are to be replaced by real values during programming or which serve only for identification purposes and must not be included in the program code:

<table>
<tr><th colspan="3">Generally applicable notation</th><th colspan="2">Program code</th></tr>
<tr><th>Element</th><th>Use</th><th>Example</th><th>Example</th><th>Notes on programming</th></tr>
<tr><td rowspan="2"><...></td><td rowspan="2">Angle brackets indicate variable elements.
The identifier of the variable element is given between the angle brackets.</td><td>SETAL(<No>)</td><td>SETAL(65679)</td><td rowspan="2">Note
A value to be specified in the NC program for a variable element must not be placed in angle brackets!</td></tr>
<tr><td>Explanation
<No> stands for the first and in this example only parameter in the SETAL call.
Value: Alarm number
A value must be specified for this parameter when programming SETAL.</td><td>Comment
Setting alarm No. 65679.</td></tr>
<tr><td rowspan="2">[...]</td><td rowspan="2">Square brackets indicate optional elements.
The identifier of the optional element is given between the square brackets.</td><td>SETAL(<No>[,<String>])</td><td>SETAL(65679, "My Text")</td><td rowspan="2">Note
A value to be specified in the NC program for an optional element must not be placed in square brackets!</td></tr>
<tr><td>Explanation
[,<String>] stands for the second parameter in the SETAL call.
Meaning: String of the data type STRING
For this parameter, the specification of a value is optional when programming SETAL.</td><td>Comment
Set alarm no. 65679 and adopt the string "My Text" as parameter value.
If the alarm parameter %4 is integrated in the configured alarm text, the string specified in the SETAL call is output at this position in the alarm text.</td></tr>
</table>

Note

The general notation is to facilitate understanding only. It is not suitable as a copy template for programming.

Standard scope

This documentation only describes the functionality of the standard version. This may differ from the scope of the functionality of the system that is actually supplied. Please refer to the ordering documentation only for the functionality of the supplied drive system.

It may be possible to execute other functions in the system which are not described in this documentation. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

For reasons of clarity, this documentation cannot include all of the detailed information on all product types. Further, this documentation cannot take into consideration every conceivable type of installation, operation and service/maintenance.

The machine manufacturer must document any additions or modifications they make to the product themselves.

Websites of third-party companies

This document may contain hyperlinks to third-party websites. Siemens is not responsible for and shall not be liable for these websites and their content. Siemens has no control over the information which appears on these websites and is not responsible for the content and information provided there. The user bears the risk for their use.

1.3 Documentation on the internet

1.3.1 Documentation overview SINUMERIK 828D

Comprehensive documentation about the functions provided in SINUMERIK 828D Version 4.8 SP4 and higher is provided in the 828D documentation overview (https://support.industry.siemens.com/cs/ww/en/view/109766724).

You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- User: Operating
- User: Programming
- Manufacturer/Service: Configuring
- Manufacturer/Service: Commissioning
- Manufacturer/Service: Functions
- Manufacturer/Service: Safety Integrated
- SINUMERIK Integrate/MindApp
- Info & Training

1.3.2 Documentation overview SINUMERIK operator components

Comprehensive documentation about the SINUMERIK operator components is provided in the Documentation overview SINUMERIK operator components (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW).

You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- Operator Panels
- Machine control panels

- Machine Pushbutton Panel
- Handheld Unit/Mini handheld devices
- Further operator components

An overview of the most important documents, entries and links to SINUMERIK is provided at SINUMERIK Overview - Topic Page (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW).

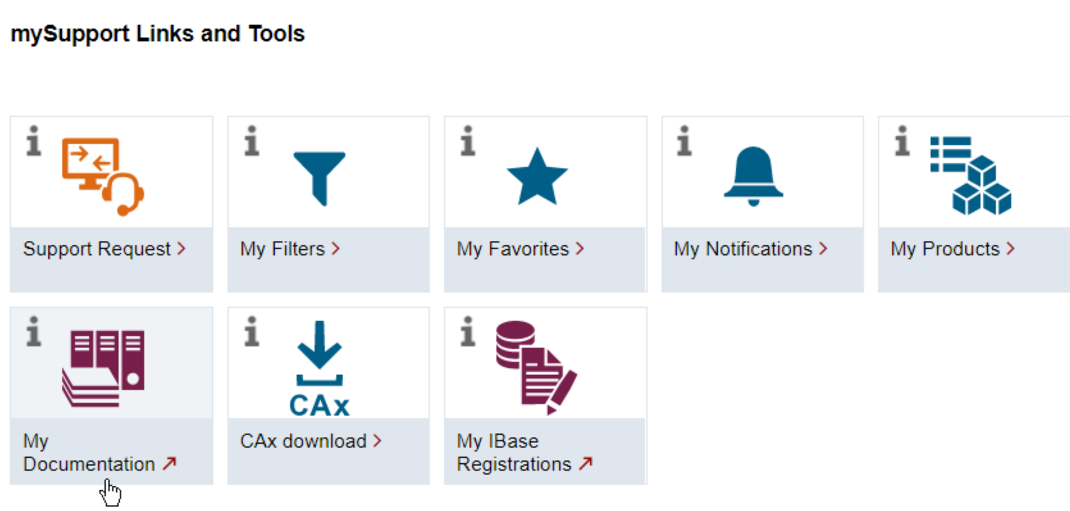
1.4 Feedback on the technical documentation

If you have any questions, suggestions or corrections regarding the technical documentation which is published in the Siemens Industry Online Support, use the link "Send feedback" link which appears at the end of the entry.

1.5 mySupport documentation

With the "mySupport documentation" web-based system you can compile your own individual documentation based on Siemens content, and adapt it for your own machine documentation.

To start the application, click on the "My Documentation" tile on the "mySupport links and tools" (https://support.industry.siemens.com/cs/ww/en/my) portal page:



The configured manual can be exported in RTF, PDF or XML format.

Note

Siemens content that supports the mySupport documentation application can be identified by the presence of the "Configure" link.

1.6 Service and Support

Product support

You can find more information about products on the internet:

Product support (https://support.industry.siemens.com/cs/ww/en/)

The following is provided at this address:

- Up-to-date product information (product announcements)
- FAQs (frequently asked questions)
- Manuals
- Downloads
- Newsletters with the latest information about your products
- Global forum for information and best practice sharing between users and specialists
- Local contact persons via our Contacts at Siemens database (→ "Contact")
- Information about field services, repairs, spare parts, and much more (→ "Field Service")

Technical support

Country-specific telephone numbers for technical support are provided on the internet at address (https://support.industry.siemens.com/cs/ww/en/sc/4868) in the "Contact" area.

If you have any technical questions, please use the online form in the "Support Request" area.

Training

You can find information on SITRAIN at the following address (https://www.siemens.com/sitrain).
SITRAIN offers training courses for automation and drives products, systems and solutions from Siemens.

Siemens support on the go

With the award-winning "Industry Online Support" app, you can access more than 300,000 documents for Siemens Industry products – any time and from anywhere. The app can support you in areas including:

- Resolving problems when implementing a project
- Troubleshooting when faults develop
- Expanding a system or planning a new system

Furthermore, you have access to the Technical Forum and other articles from our experts:

- FAQs
- Application examples
- Manuals
- Certificates
- Product announcements and much more

The "Industry Online Support" app is available for Apple iOS and Android.

1.7 Using OpenSSL

This product can contain the following software:

- Software developed by the OpenSSL project for use in the OpenSSL toolkit
- Cryptographic software created by Eric Young.
- Software developed by Eric Young

You can find more information on the internet:

- OpenSSL (https://www.openssl.org)
- Cryptsoft (https://www.cryptsoft.com)

1.8 Compliance with the General Data Protection Regulation

Siemens observes standard data protection principles, in particular the data minimization rules (privacy by design).

For this product, this means:

The product does not process or store any personal data, only technical function data (e.g. time stamps). If the user links this data with other data (e.g. shift plans) or if he/she stores person-related data on the same data medium (e.g. hard disk), thus personalizing this data, he/she must ensure compliance with the applicable data protection stipulations.

Fundamental safety instructions 2

2.1 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 against unauthorized access. • Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

2.2 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.

2.3 Security information

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
https://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 under
https://www.siemens.com/cert.

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, 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.
- On completion of commissioning, check all security-related settings.

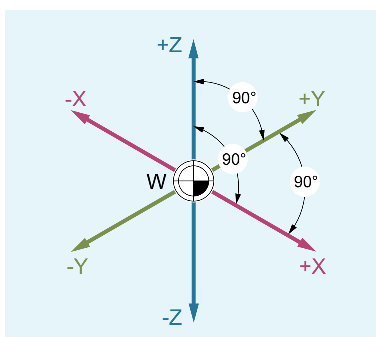
Fundamentals 3

3.1 Fundamental Geometrical Principles

3.1.1 Workpiece positions

3.1.1.1 Reference system of position specifications

In order that the machine or the control can work with the positions specified in the NC program, these position specifications have to be made in a reference system that can be transferred to the directions of motion of the machine axes. For this purpose, a right-handed Cartesian (rectangular) coordinate system with the three main axes X, Y and Z is used. The position of the axis directions in such a workpiece coordinate system is defined by DIN 66217. The workpiece zero (W) is the origin of a workpiece coordinate system.

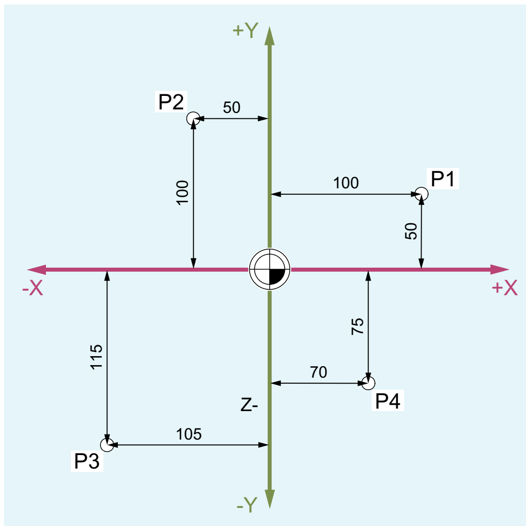


3.1.1.2 Cartesian coordinates

The axes in the coordinate system are assigned dimensions. In this way, it is possible to clearly describe every point in the coordinate system, and therefore every workpiece position based on the direction (X, Y and Z) and three numerical values. The workpiece zero always has the coordinates X0, Y0 and Z0.

Position specifications in the form of Cartesian coordinates

To simplify things, we will only consider one plane of the coordinate system in the following example, the X/Y plane:

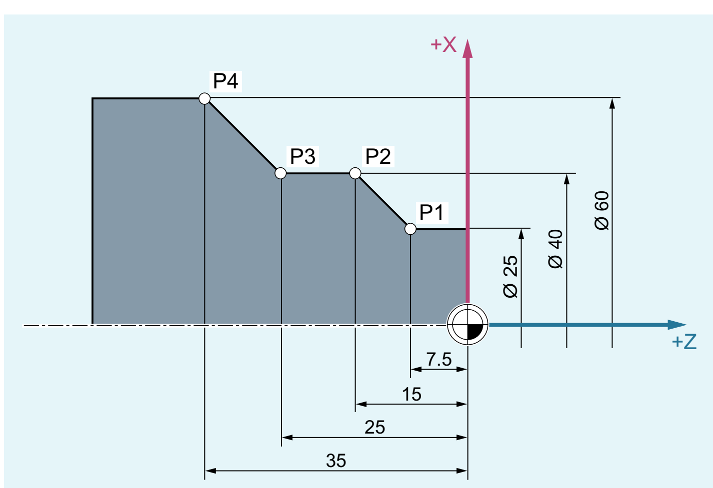


Points P1 to P4 have the following coordinates:

Position	Coordinates
P1	X100 Y50
P2	X-50 Y100
P3	X-105 Y-115
P4	X70 Y-75

Example: Workpiece positions for turning

With lathes, one plane is sufficient to describe the contour:

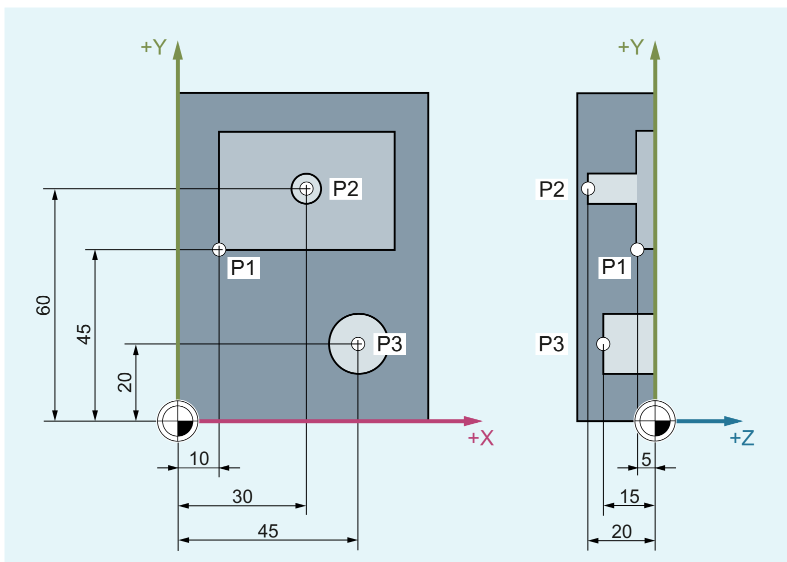


Points P1 to P4 have the following coordinates:

Position	Coordinates
P1	X25 Z-7.5
P2	X40 Z-15
P3	X40 Z-25
P4	X60 Z-35

Example: Workpiece positions for milling

For milling, the feed depth must also be described, i.e. the third coordinate (in this case Z) must also be assigned a numerical value.



Points P1 to P3 have the following coordinates:

Position	Coordinates
P1	X10 Y45 Z-5
P2	X30 Y60 Z-20
P3	X45 Y20 Z-15

3.1.1.3 Polar coordinates

Polar coordinates can be used instead of Cartesian coordinates to describe workpiece positions. This is useful when a workpiece or part of a workpiece has been dimensioned with radius and angle. The point from which the dimensioning starts is called the "pole".

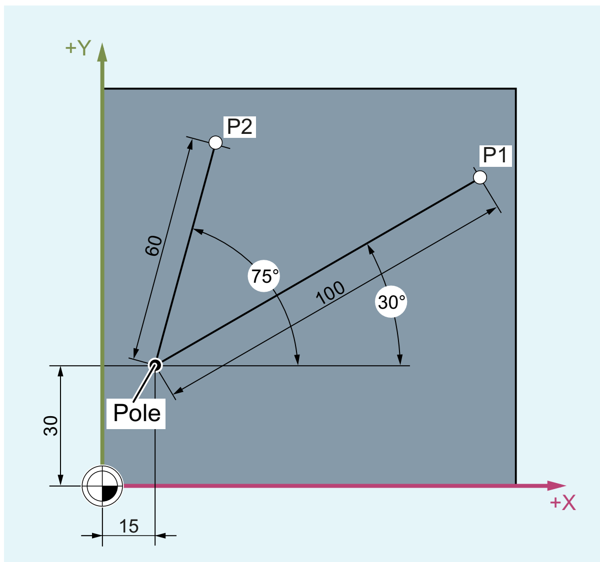
Position specifications in the form of polar coordinates

Polar coordinates are made up of the **polar radius** and the **polar angle**.

The polar radius is the distance between the pole and the position.

The polar angle is the angle between the polar radius and the horizontal axis of the working plane. Negative polar angles are in the clockwise direction, positive polar angles in the counterclockwise direction.

Example



Points P1 and P2 can then be described – with reference to the pole – as follows:

Position	Polar coordinates
P1	RP=100 AP=30
P2	RP=60 AP=75
RP: Polar radius AP: Polar angle	

3.1.1.4 Absolute dimensions

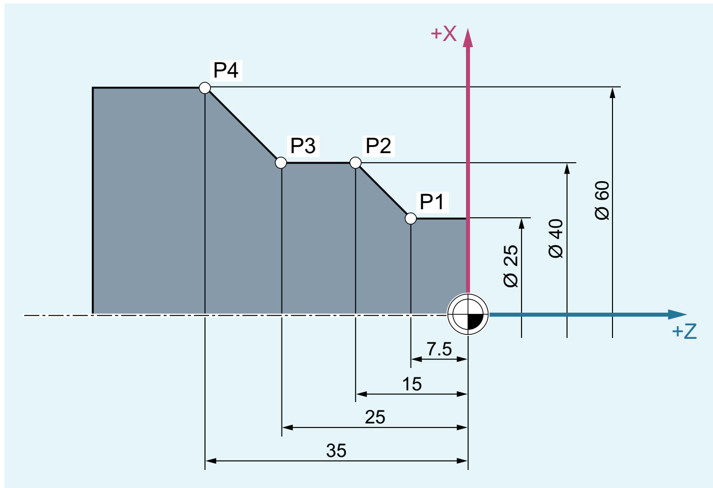
Position specifications in absolute dimensions

With absolute dimensions, all the position specifications refer to the currently valid zero point.

Applied to tool movement this means:

the position, to which the tool is to travel.

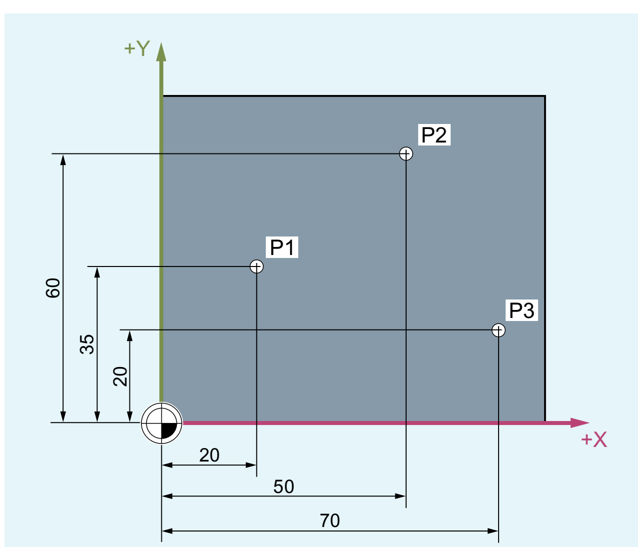
Example: Turning



In absolute dimensions, the following position specifications result for points P1 to P4:

Position	Position specification in absolute dimensions
P1	X25 Z-7.5
P2	X40 Z-15
P3	X40 Z-25
P4	X60 Z-35

Example: Milling



In absolute dimensions, the following position specifications result for points P1 to P3:

Position	Position specification in absolute dimensions
P1	X20 Y35
P2	X50 Y60
P3	X70 Y20

3.1.1.5 Incremental dimension

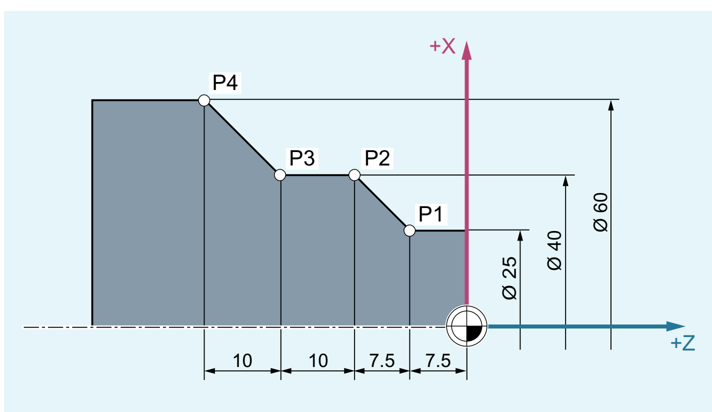
Position specifications in incremental dimensions

In production drawings, the dimensions often do not refer to a zero point, but rather to another workpiece point. So that these dimensions do not have to be converted, they can be specified in incremental dimensions. In this method of dimensional notation, a position specification refers to the previous point.

Applied to tool movement this means:

The incremental dimensions describe the distance the tool is to travel.

Example: Turning



In incremental dimensions, the following position specifications are obtained for points P2 to P4:

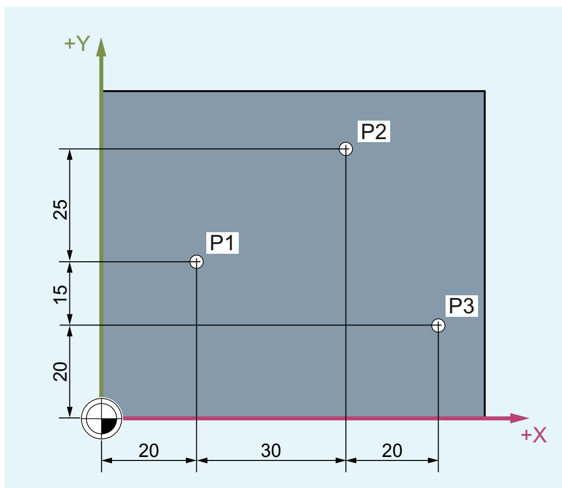
Position	Position specification in incremental dimensions	The specification refers to:
P2	X15 Z-7.5	P1
P3	Z-10	P2
P4	X20 Z-10	P3

Note

With DIAMOF or DIAM90 (Page 170) active, the set distance in incremental dimensions (G91) is programmed as a radius dimension.

Example: Milling

The position specifications for points P1 to P3 in incremental dimensions are:



In incremental dimensions, the following position specifications are obtained for points P1 to P3:

Position	Position specification in incremental dimensions	The specification refers to:
P1	X20 Y35	Zero point
P2	X30 Y20	P1
P3	X20 Y-35	P2

3.1.2 Working planes

An NC program requires information about the working plane. Because only then can the control correctly take into account the tool offset values, for example. The specification of the working plane is also required for certain types of circular-path programming and polar coordinates.

The working plane is specified in the Cartesian workpiece coordinate system used as basis using two coordinate axes. The third coordinate axis is perpendicular to this working plane and determines the infeed direction of the tool (e.g. for 2D machining).

The working planes are activated in the NC program using G commands G17, G18 and G19. The relationship is defined as follows:

G command	Working plane	Abscissa	Ordinate	Applicate ≙ infeed direction
G17	X/Y	X	Y	Z
G18	Z/X	Z	X	Y
G19	Y/Z	Y	Z	X

In the default setting, G18 (Z/X plane) is defined for turning and G17 (X/Y plane) is defined for milling:

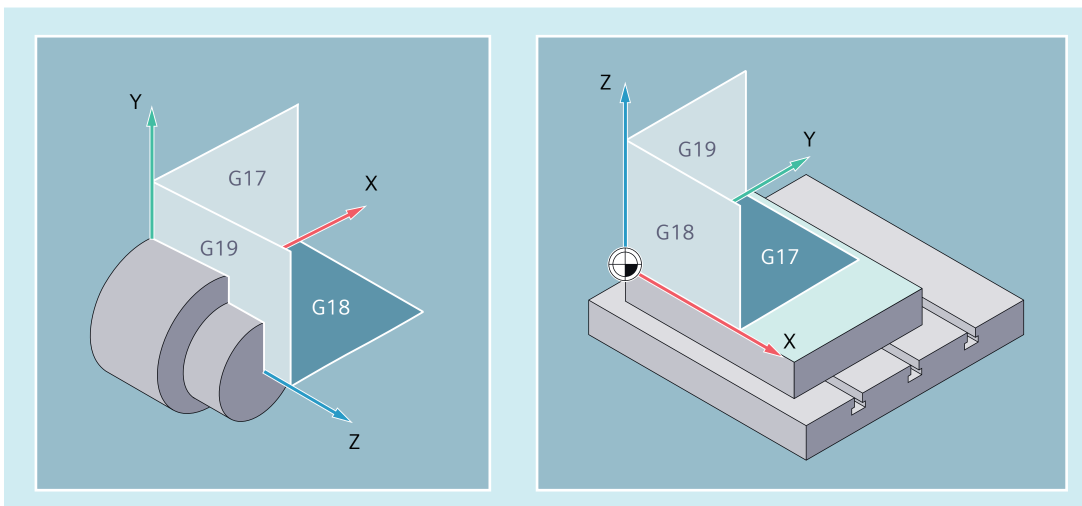


Figure 3-1 Working planes for turning and milling

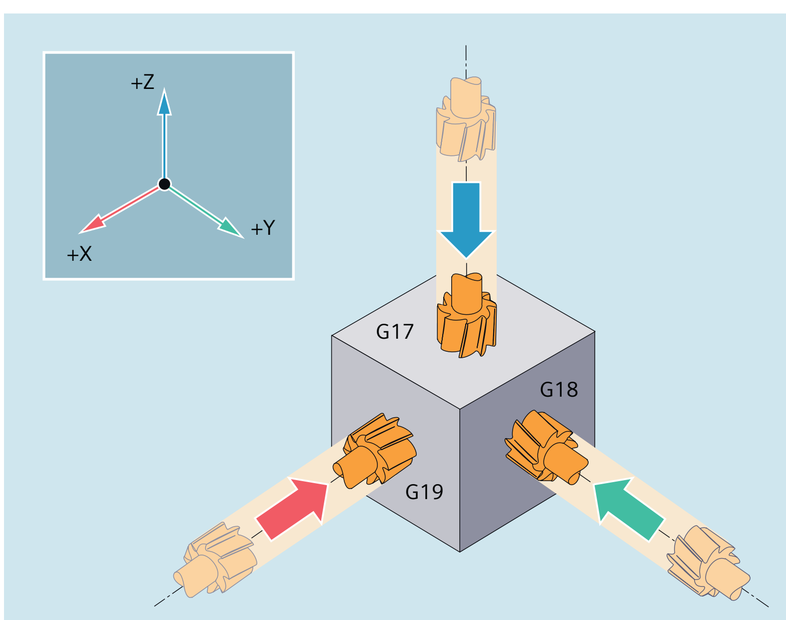
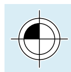
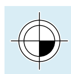
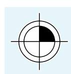
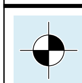
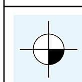
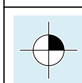
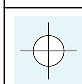


Figure 3-2 Feed directions for milling

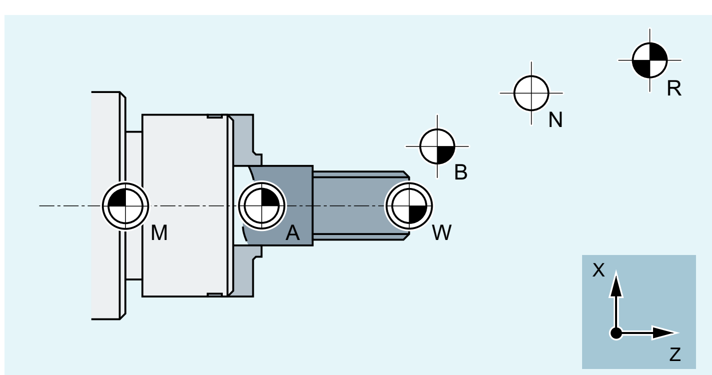
3.1.3 Zero points and reference points

Various zero points and reference points are defined on an NC machine:

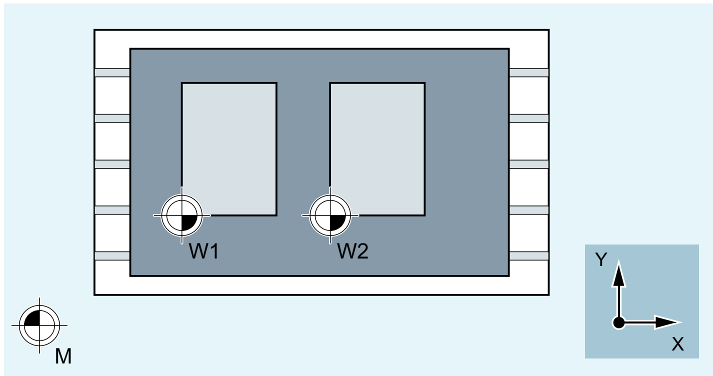
Zero points		
	**M**	Machine zero The machine zero defines the machine coordinate system (MCS). All other reference points refer to the machine zero.
	**W**	Workpiece zero = program zero The workpiece zero defines the workpiece coordinate system in relation to the machine zero.
	**A**	Blocking point Can be the same as the workpiece zero (only for lathes).

Reference points		
	**R**	Reference point Position defined by output cam and measuring system. The distance to the machine zero **M** must be known so that the axis position at this point can be set exactly to this value.
	**B**	Starting point Can be defined by the program. The 1st tool starts machining here.
	**T**	Toolholder reference point Is on the toolholder. By entering the tool lengths, the control calculates the distance between the tool tip and the toolholder reference point.
	**N**	Tool change point

Zero points and reference points for turning



Zero points for milling



3.1.4 Coordinate systems

A distinction is made between the following coordinate systems:

- Machine coordinate system (MCS) (Page 36) with the machine zero **M**
- Basic coordinate system (BCS) (Page 38)
- Basic zero system (BZS) (Page 39)
- Settable zero system (SZS) (Page 40)
- Workpiece coordinate system (WCS) (Page 41) with the workpiece zero **W**

3.1.4.1 Machine coordinate system (MCS)

The machine coordinate system comprises all the physically existing machine axes.

Reference points and tool and pallet changing points (fixed machine points) are defined in the machine coordinate system.

If programming is performed directly in the machine coordinate system (possible with some G commands), then the physical axes of the machine are directly addressed. Any workpiece clamping that is present is not taken into account.

Note

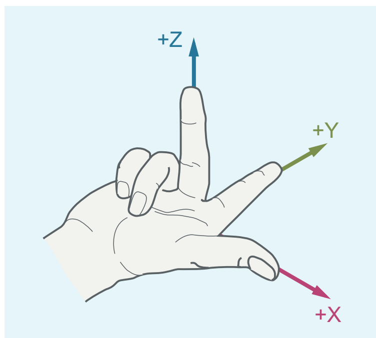
If there are various machine coordinate systems (e.g. 5-axis transformation), then an internal transformation is used to map the machine kinematics on the coordinate system in which the programming is performed.

Three-finger rule

The orientation of the coordinate system relative to the machine depends on the machine type. The axis directions follow the so-called "three-finger rule" of the **right** hand (according to DIN 66217).

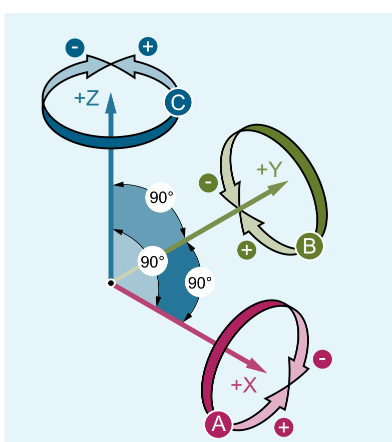
Seen from in front of the machine, the middle finger of the right hand points in the opposite direction to the infeed of the main spindle. Therefore:

- the thumb points in the +X direction
- the index finger points in the +Y direction
- the middle finger points in the +Z direction



Rotary motion around the coordinate axes X, Y and Z are designated A, B and C. The direction of rotation is obtained from the direction of the rotary motion when looking in the positive direction of the coordinate axis:

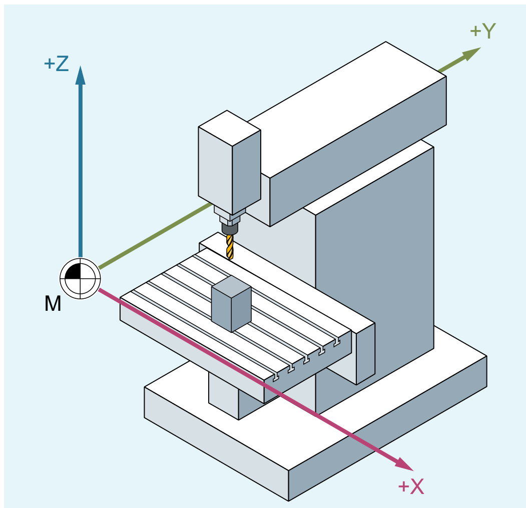
Direction of the rotary motion	Direction of rotation
clockwise	positive
counter-clockwise	negative



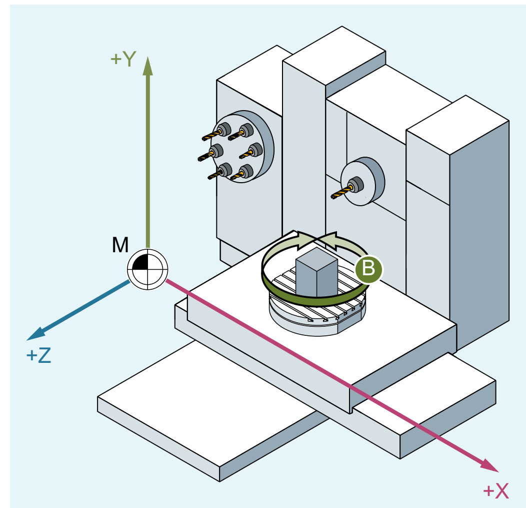
X, Y, Z Vertical coordinate axes arranged on top of one another
A, B, C Rotary axes, rotating around X, Y, Z

Position of the coordinate system in different machine types

The position of the coordinate system resulting from the "three-finger rule" can have a different orientation for different machine types, which are shown in the following two examples:



Vertical 3-axis milling machine



Horizontal 4-axis milling machine

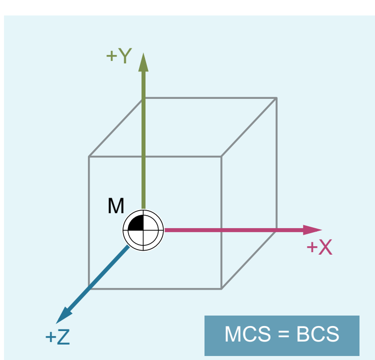
3.1.4.2 Basic coordinate system (BCS)

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 kinematics transformation

BCS and MCS always coincide when the BCS can be mapped onto the MCS without kinematics transformation (e.g. 5-axis transformation, TRANSMIT/TRACYL/TRAANG).

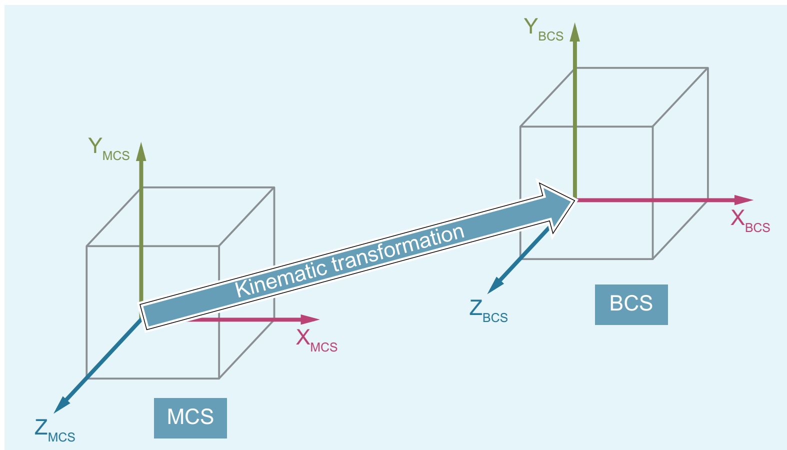
On such machines, machine axes and geometry axes can have the same names.



Machine tools with kinematic transformation

BCS and MCS do not coincide when the BCS is mapped onto the MCS with kinematics transformation (e.g. 5-axis transformation, TRANSMIT/TRACYL/TRAANG).

On such machines the machine axes and geometry axes must have different names.

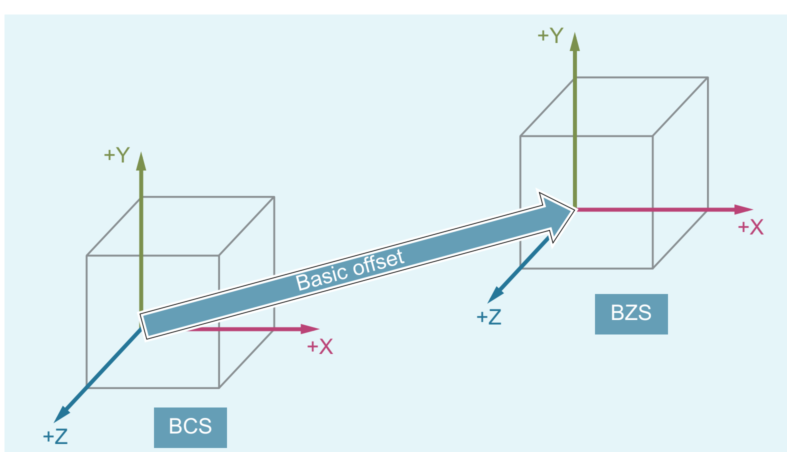


Machine kinematics

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 axis motion.

3.1.4.3 Basic zero system (BZS)

The basic zero system (BZS) is derived from the basic coordinate system through the basic offset.



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:

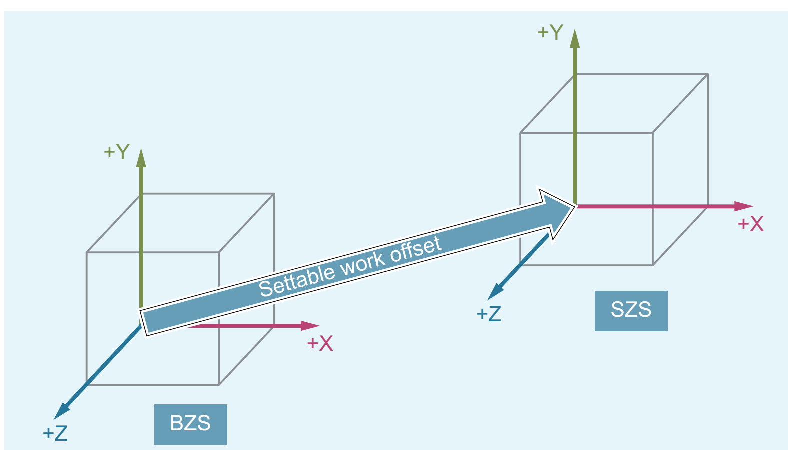
- External work offset
- DRF offset
- Overlaid movement
- Chained system frames
- Chained basic frames

3.1.4.4 Settable zero system (SZS)

Settable work offset

The "settable zero system" (SZS) is obtained from the basic zero system (BZS) as a result of the settable work offset.

Settable work offsets are activated in the NC program with the G commands G54 ... G57 and G505 ... G599.



If no programmable coordinate transformations (frames) are active, then the "settable zero system" is the workpiece coordinate system (WCS).

Programmable coordinate transformations (frames)

Sometimes it is useful or necessary within an NC program, to move the originally selected workpiece coordinate system (or the "settable zero system") to another position and, if required, to rotate it, mirror it and/or scale it. This is performed using programmable coordinate transformations (frames) (Page 303).

Note

Programmable coordinate transformations (frames) always refer to the "settable zero system".

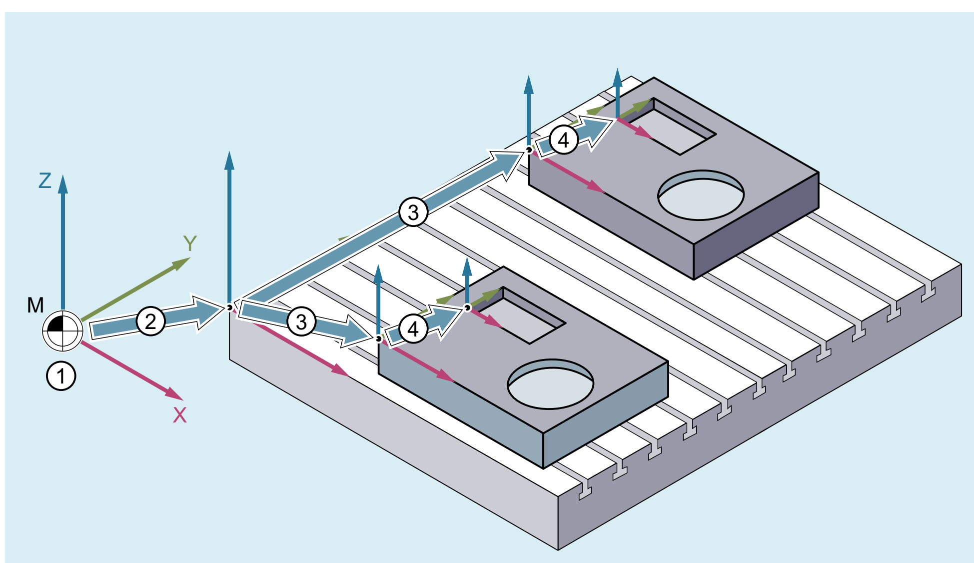
3.1.4.5 Workpiece coordinate system (WCS)

The geometry of a workpiece is described in the workpiece coordinate system (WCS). In other words, the data in the NC program refers to the workpiece coordinate system.

The workpiece coordinate system is always a Cartesian coordinate system and assigned to a specific workpiece.

3.1.4.6 What is the relationship between the various coordinate systems?

The following example should help clarify the relationships between the various coordinate systems:



①	A kinematic transformation is not active. This means that the machine coordinate system and the basic coordinate system coincide.
②	The basic zero system (BZS) with the pallet zero are obtained from the basic offset.
③	The settable work offset G54 or G55 specifies the "settable zero system" (SZS) for workpiece 1 or workpiece 2 respectively.
④	The workpiece coordinate system (WCS) results from the programmable coordinate transformation.

3.2 Fundamental Principles of NC Programming

Note

DIN 66025 is the guideline for NC programming.

3.2.1 Name of an NC program

Rules

Each NC program must be assigned a program name (identifier) when it is created. The program name can be chosen freely providing the following rules are observed:

- Permissible characters:
  - Letters: A ... Z, a ... z
  - Numbers: 0 ... 9
  - Underscore: _
- The first two characters should either be two letters or an underscore followed by a letter.

  Note

  If this condition is satisfied, then an NC program can be called as subprogram from another program just by specifying the program name. However, if the program name starts with digits, the subprogram call is then only possible via the CALL statement.

- Maximum length: 24 characters

Note

Uppercase/lowercase letters

The SINUMERIK NC language does **not** distinguish between uppercase and lowercase letters.

Note

Impermissible program names

To avoid problems with Microsoft Windows applications, the following program names may **not** be used:

- CON, PRN, AUX, NUL
- COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9
- LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9

Further restrictions, see "Names (Page 370)".

Control-internal extensions

The program name assigned when the program is created is extended within the control with the addition of a prefix and a suffix:

- Prefix: _N_
- Suffix:
  - Main programs: _MPF
  - Subprograms: _SPF

Files in punch tape format

Externally created program files that are read via the V.24 interface must be present in punch tape format.

The following additional rules apply for the program name of a file in punch tape format:

- First character: %
- Then a four-character file extension: _xxx

Examples:

- %_N_SHAFT123_MPF
- %Flange3_MPF

3.2.2 Structure and contents of an NC program

3.2.2.1 Blocks and block components

Blocks

An NC program consists of a sequence of NC blocks. Each block contains the data for executing a step in the workpiece machining.

Block components

NC blocks consist of the following components:

- Commands (statements) according to DIN 66025
- Elements of the NC high-level language

Commands according to DIN 66025

The commands according to DIN 66025 consist of an address character and a digit or sequence of digits representing an arithmetic value.

Address character (address)

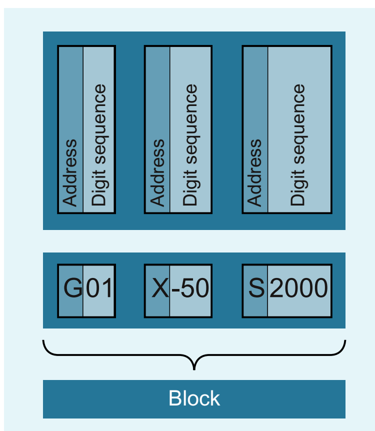
The address character (generally a letter) defines the meaning of the command.

Examples:

Address character	Meaning
G	G command (preparatory function)
X	Position data for the X axis
S	Spindle speed

Digit sequence

The digit sequence is the value assigned to the address character. The sequence of digits can contain a sign and decimal point. The sign always appears between the address letter and the sequence of digits. Positive signs (+) and leading zeros (0) do not have to be specified.



Elements of the NC high-level language

As the command set according to DIN 66025 is no longer adequate for programming complex machining sequences in modern machine tools, it has been extended by the elements of the NC high-level language.

These include, for example:

- Commands of the NC high-level language
  In contrast to the commands according to DIN 66025, the commands of the NC high-level language consist of several address letters, e.g.
  - `OVR` for speed override
  - `SPOS` for spindle positioning
- Identifiers (defined names) for:
  - System variables
  - User-defined variables
  - Subprograms
  - Keywords
  - Jump markers
  - Macros

  Note

  An identifier must be unique and cannot be used for different objects.

- Comparison operators
- Logic operators
- Arithmetic functions
- Control structures

Effectiveness of commands

Commands are either modal or non-modal:

- Modal
  Modal commands retain their validity with the programmed value (in all following blocks) until:
  - A new value is programmed under the same command.
  - A command is programmed that revokes the effect of the previously valid command
- Non-modal
  Non-modal commands only apply to the block in which they were programmed.

End of program

The last block in the execution sequence contains a special word for the end of program: `M2`, `M17` or `M30`.

3.2.2.2 Block rules

Start of block

NC blocks can be identified at the start of the block by block numbers. These consist of the character "N" and a positive integer, e.g.
`N40 ...`

The order of the block numbers is arbitrary, however, block numbers in rising order are recommended.

Note

Block numbers must be unique within a program in order to achieve an unambiguous result when searching.

End of block

A block ends with the character LF (LINE FEED = new line).

Note

The LF character does not have to be written. It is generated automatically by the line change.

Block length

A block can contain a maximum of **512 characters** (including the comment and end-of-block character LF).

Note

Three blocks of up to 66 characters each are normally displayed in the current block display on the screen. Comments are also displayed. Messages are displayed in a separate message window.

Order of the statements

In order to keep the block structure as clear as possible, the statements in a block should be arranged in the following order:
`N... G... X... Y... Z... F... S... T... D... M... H...`

Address	Meaning
N	Address of block number
G	Preparatory function
X,Y,Z	Positional data
F	Feedrate
S	Speed

T	Tool
D	Tool offset number
M	Additional function
H	Auxiliary function

Note

Certain addresses can be used repeatedly within a block, e.g.

G..., M..., H...

3.2.2.3 Value assignments

Values can be assigned to the addresses. The following rules apply:

- An "=" sign must be inserted between the address and the value if:
  - The address comprises more than one letter.
  - The value includes more than one constant.

  The "=" sign can be omitted if the address is a single letter and the value consists of only one constant.
- Signs are permitted.
- Separators are permitted after the address letter.

Examples:

X10	Value assignment (10) to address X, "=" not required
X1=10	Value assignment (10) to address (X) with numeric extension (1), "=" required
X=10*(5+SIN(37.5))	Value assignment by means of a numeric expression, "=" required

Note

A numeric extension must always be followed by one of the special characters "=", "(", "[", ")", "]", ",", or an operator, in order to distinguish an address with numeric extension from an address letter with a value.

3.2.2.4 Comments

To make an NC program easier to understand, comments can be added to the NC blocks.

A comment is at the end of a block and is separated from the program section of the NC block by a semicolon (";").

Example 1:

Program code	Comment
N10 G1 F100 X10 Y20	; Comment to explain the NC block

Example 2:

Program code	Comment
N10	; Company G&S, order no. 12A71
N20	; Program written by H. Smith, Dept. TV 4 on November 21, 1994
N50	; Section no. 12, housing for submersible pump type TP23A

Note

Comments are stored and appear in the current block display when the program is running.

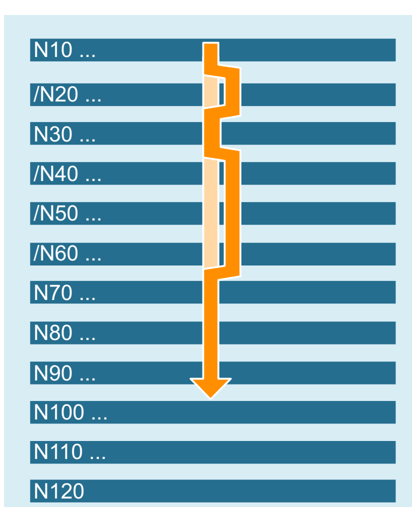
3.2.2.5 Skipping blocks

NC blocks, which are not to be executed every time the program runs, can be skipped and not processed. This function is used when testing and/or running-in new programs, for example.

Skip blocks

Blocks to be skipped are indicated in the part program by the character "/" before the block number. Several consecutive blocks can also be skipped. The instructions in the skipped blocks are not executed and the program resumes with the next block that is not skipped.

Example:



Program code	Comment
N10 …	; Is executed
/N20 …	; Skipped
N30 …	; Is executed
/N40 …	; Skipped
/N50 …	; Skipped
/N60 …	; Skipped
N70 …	; Is executed
...	

Skip levels

Blocks can be assigned to skip levels (max. 10), which can be activated via the user interface or the PLC user program.

The assignment is made in the NC program using a forward slash, followed by the number of the skip level. Only one skip level can be specified for each block.

Example:

Program code	Comment
/ ...	; Block is skipped (1st skip level)
/0 ...	; Block is skipped (1st skip level)
/1 N010...	; Block is skipped (2nd skip level)
/2 N020...	; Block is skipped (3rd skip level)
...	
/7 N100...	; Block is skipped (8th skip level)
/8 N080...	; Block is skipped (9th skip level)
/9 N090...	; Block is skipped (10th skip level)

Note

The levels to be skipped can only be changed when the control system is in the STOP/reset state.

Note

The number of skip levels that can be used depends on a display machine data.

Note

Skipping blocks also remains active during block searches.

Note

System and user variables can also be used in conditional jumps in order to control program execution.

3.3 Creating an NC program

3.3.1 Basic procedure

The programming of the individual operation steps in the NC language generally represents only a small proportion of the work in the development of an NC program.

Programming of the actual instructions should be preceded by the planning and preparation of the operation steps. The more accurately you plan in advance how the NC program is to be structured and organized, the faster and easier it will be to produce a complete program, which is clear and free of errors. Clearly structured programs are especially advantageous when changes have to be made later.

As every part is not identical, it does not make sense to create every program in the same way. However, the following procedure has shown itself to be suitable in the most cases.

Procedure

1. **Prepare the workpiece drawing**
   - Define the workpiece zero
   - Draw the coordinate system
   - Calculate any missing coordinates
2. **Define the machining sequence**
   - Which tools are used when and for the machining of which contours?
   - In which order will the individual elements of the workpiece be machined?
   - Which individual elements are repeated (possibly also rotated) and should be stored in a subroutine?
   - Are there contour sections in other part programs or subroutines that could be used for the current workpiece?
   - Where are zero offsets, rotating, mirroring and scaling useful or necessary (frame concept)?

3. **Create a machining plan**
   Define all machining operations step-by-step, e.g.
   – Rapid traverse movements for positioning
   – Tool change
   – Define the machining plane
   – Retraction for checking
   – Switch spindle, coolant on/off
   – Call up tool data
   – Feed
   – Path correction
   – Approaching the contour
   – Retraction from the contour
   – etc.
4. **Compile machining steps in the programming language**
   – Write each individual step as an NC block (or NC blocks).
5. **Combine the individual steps into a program**

3.3.2 Available characters

The following characters are available for writing NC programs:

- Uppercase characters:
  A, B, C, D, E, F, G, H, I, J, K, L, M, N,(O),P, Q, R, S, T, U, V, W, X, Y, Z
- Lowercase characters:
  a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
- Numbers:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Special characters:
  See the table below.

Special characters	Meaning
%	Program start character (used only for writing programs on an external PC)
(	For bracketing parameters or expressions
)	For bracketing parameters or expressions
[	For bracketing addresses or indexes
]	For bracketing addresses or indexes
<	Less than
>	Greater than
:	Main block, end of label, chain operator
=	Assignment, part of equation

Special characters	Meaning
/	Division, block suppression
*	Multiplication
+	Addition
-	Subtraction, minus sign
"	Double quotation marks, identifier for character string
'	Single quotation marks, identifier for special numerical values: hexadecimal, binary
$	System variable identifiers
s_	Underscore, belonging to letters
?	Reserved
!	Reserved
.	Decimal point
,	Comma, parameter separator
;	Comment start
&	Format character, same effect as space character
LF	End of block
Tab character	Separator
Blank	Separator (blank)

Note

Take care to differentiate between the letter "O" and the digit "0".

Note

No distinction is made between uppercase and lowercase characters (exception: tool call).

Note

Non-printable special characters are treated like blanks.

3.3.3 Program header

The NC blocks that are placed in front of the actual motion blocks for the machining of the workpiece contour are called the program header.

The program header contains information/statements regarding:

- Tool change
- Tool offsets
- Spindle motion
- Feed control
- Geometry settings (zero offset, selection of the working plane)

Program header for turning

The following example shows the typical structure of an NC program header for turning:

Program code	Comment
N10 G0 G153 X200 Z500 T0 D0	; Retract toolholder before tool turret is rotated.
N20 T5	; Swing in tool 5.
N30 D1	; Activate cutting edge data set of the tool.
N40 G96 S300 LIMS=3000 M4 M8	; Constant cutting rate (Vc) = 300 m/min, speed limitation = 3000 rpm, direction of rotation counterclockwise, cooling on.
N50 DIAMON	; X axis will be programmed in the diameter.
N60 G54 G18 G0 X82 Z0.2	; Call zero offset and working plane, approach starting position.
...	

Program header for milling

The following example shows the typical structure of an NC program header for milling:

Program code	Comment
N10 T="SF12"	; Alternative: T123
N20 M6	; Trigger tool change.
N30 D1	; Activate cutting edge data set of the tool.
N40 G54 G17	; Zero offset and working plane.
N50 G0 X0 Y0 Z2 S2000 M3 M8	; Approach to the workpiece, spindle and coolant on.
...	

If tool orientation / coordinate transformation is being used, any transformations still active should be deleted at the start of the program:

Program code	Comment
N10 CYCLE800()	; Resetting of the swiveled plane
N20 TRAFOOF	; Resetting of TRAORI, TRANSMIT, TRACYL, ...
...	

3.3.4 Program examples

3.3.4.1 Example 1: First programming steps

Program example 1 is to be used to perform and test the first programming steps.

Procedure

1. Create a new part program (name)
2. Edit the part program
3. Select the part program
4. Activate single block
5. Start the part program.

Note

In order that the program can run on the machine, the machine data must have been appropriately set (→ see machine manufacturer's data!).

Note

When testing a program, alarms can occur, which must first be reset.

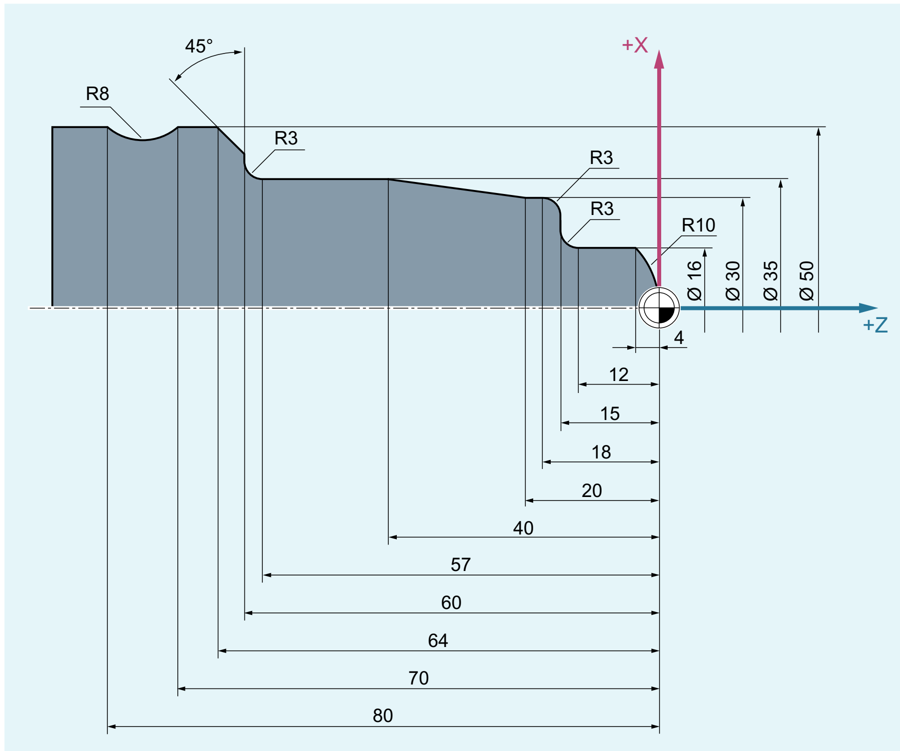
NC program

Program code	Comment
N10 MSG ("THIS IS MY NC PROGRAM")	; Message "THIS IS MY NC PROGRAM" displayed in the alarm line.
N20 F200 S900 T1 D2 M3	; Feedrate, spindle, tool, tool offset, spindle clockwise.
N30 G0 X100 Y100	; Approach position in rapid traverse.
N40 G1 X150	; Rectangle with feedrate, straight line in X.
N50 Y120	; Straight line in Y.
N60 X100	; Straight line in X.
N70 Y100	; Straight line in Y.
N80 G0 X0 Y0	; Retraction in rapid traverse.
N100 M30	; End of block.

3.3.4.2 Example 2: NC program for turning

Program example 2 is intended for machining a workpiece on a lathe. It contains radius programming and tool radius compensation.

Dimension drawing of the workpiece



NC program

Program code	Comment
N5 G0 G53 X280 Z380 D0	; Starting point.
N10 TRANS X0 Z250	; Work offset.
N15 LIMS=4000	; Speed limitation (G96).
N20 G96 S250 M3	; Select constant cutting rate.
N25 G90 T1 D1 M8	; Select tool selection and offset.
N30 G0 G42 X-1.5 Z1	; Set tool with tool radius compensation.
N35 G1 X0 Z0 F0.25	
N40 G3 X16 Z-4 I0 K-10	; Turn radius 10.
N45 G1 Z-12	
N50 G2 X22 Z-15 CR=3	; Turn radius 3.
N55 G1 X24	
N60 G3 X30 Z-18 I0 K-3	; Turn radius 3.
N65 G1 Z-20	
N70 X35 Z-40	
N75 Z-57	
N80 G2 X41 Z-60 CR=3	; Turn radius 3.
N85 G1 X46	
N90 X52 Z-63	

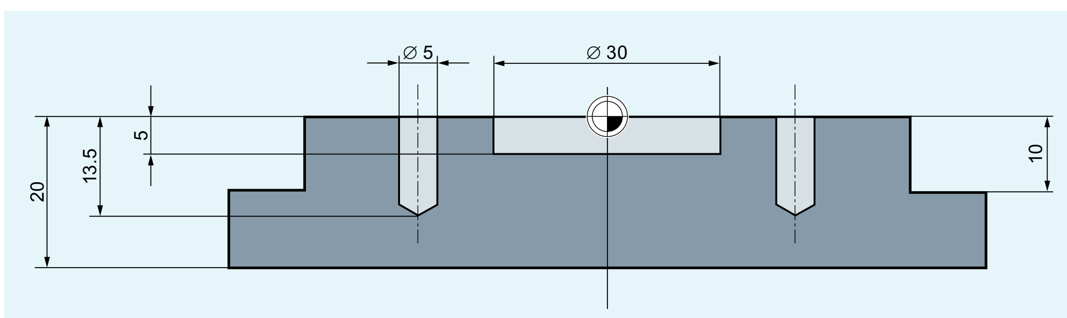
Program code	Comment
N95 G0 G40 G97 X100 Z50 M9	; Deselect tool radius compensation and approach tool change location.
N100 T2 D2	; Call tool and select offset.
N105 G96 S210 M3	; Select constant cutting rate.
N110 G0 G42 X50 Z-60 M8	; Set tool with tool radius compensation.
N115 G1 Z-70 F0.12	; Turn diameter 50.
N120 G2 X50 Z-80 I6.245 K-5	; Turn radius 8.
N125 G0 G40 X100 Z50 M9	; Retract tool and deselect tool radius compensation.
N130 G0 G53 X280 Z380 D0 M5	; Approach tool change location.
N135 M30	; End of program

3.3.4.3 Example 3: NC program for milling

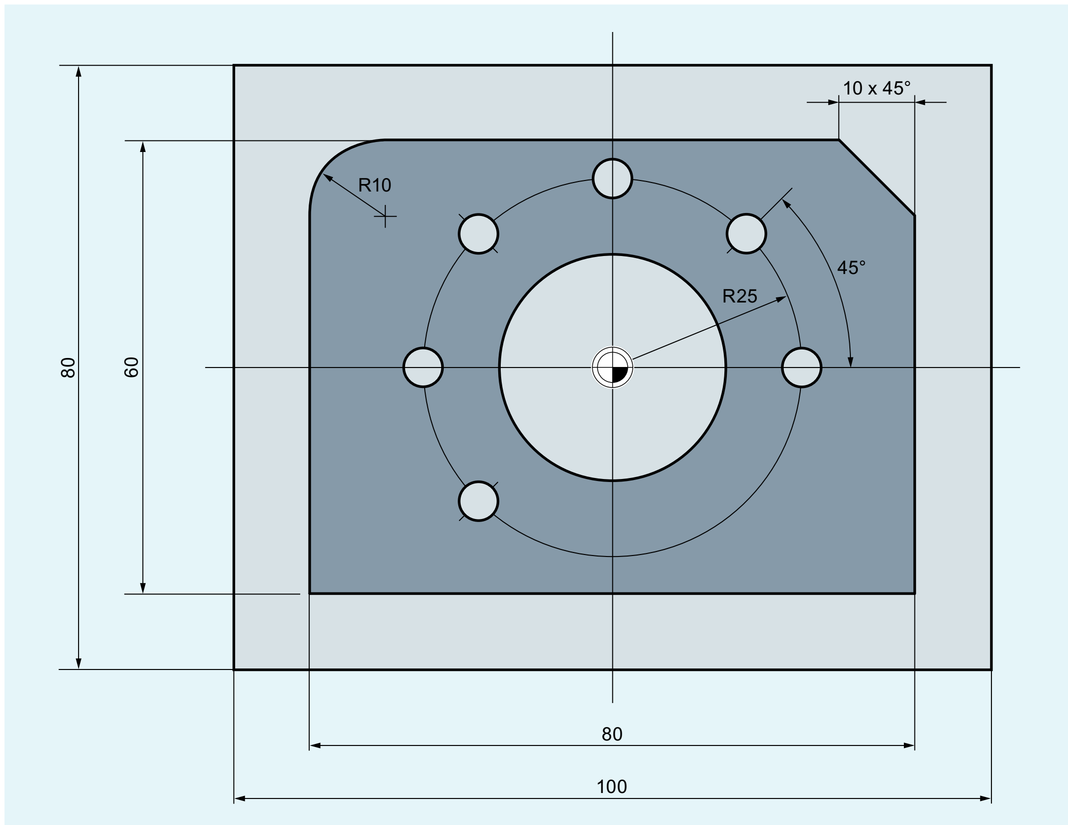
Program example 3 is intended for machining a workpiece on a vertical milling machine. It contains surface and side milling as well as drilling.

Dimension drawing of the workpiece

Side view



Top view



NC program

Program code	Comment
N10 T="PF60"	; Preselection of the tool with name PF60.
N20 M6	; Load the tool into the spindle.
N30 S2000 M3 M8	; Speed, direction of rotation, cooling on.
N40 G90 G64 G54 G17 G0 X-72 Y-72	; Basic settings of the geometry and approach starting point.
N50 G0 Z2	; Z axis to safety clearance.
N60 G450 CFTCP	; Behavior with active G41/G42.
N70 G1 Z-10 F3000	; Milling tool to machining depth with feedrate = 3000 mm/min.
N80 G1 G41 X-40	; Activation of the milling tool radius compensation.
N90 G1 X-40 Y30 RND=10 F1200	; Travel to the contour with feedrate = 1200 mm/min.
N100 G1 X40 Y30 CHR=10	
N110 G1 X40 Y-30	
N120 G1 X-41 Y-30	

Program code	Comment
N130 G1 G40 Y-72 F3000	; Deselection of the milling tool radius compensation.
N140 G0 Z200 M5 M9	; Retraction of the milling tool, spindle + cooling off.
N150 T="SF10"	; Preselection of the tool with name SF10.
N160 M6	; Load the tool into the spindle.
N170 S2800 M3 M8	; Speed, direction of rotation, cooling on.
N180 G90 G64 G54 G17 G0 X0 Y0	; Basic settings for the geometry and approach starting point.
N190 G0 Z2	
N200 POCKET4(2,0,1,-5,15,0,0,0,0,0,800,1300,0,21,5,,,2,0.5)	; Call pocket milling cycle.
N210 G0 Z200 M5 M9	; Retraction of the milling tool, spindle + cooling off.
N220 T="ZB6"	; Call 6 mm centering drill.
N230 M6	
N240 S5000 M3 M8	
N250 G90 G60 G54 G17 X25 Y0	; Exact stop G60 for exact positioning.
N260 G0 Z2	
N270 MCALL CYCLE82(2,0,1,-2.6,,0)	; Modal call of the drilling cycle.
N280 POSITION:	; Jump mark for repetition.
N290 HOLES2(0,0,25,0,45,6)	; Position pattern for drilling.
N300 ENDLABEL:	; End delimiter for repetition.
N310 MCALL	; Reset modal call.
N320 G0 Z200 M5 M9	
N330 T="SPB5"	; Call D 5 mm drill.
N340 M6	
N350 S2600 M3 M8	
N360 G90 G60 G54 G17 X25 Y0	
N370 MCALL CYCLE82(2,0,1,-13.5,,0)	; Modal call of the drilling cycle.
N380 REPEAT POSITION	; Repetition of the position description from centering.
N390 MCALL	; Resetting of the drilling cycle.
N400 G0 Z200 M5 M9	
N410 M30	; End of program.

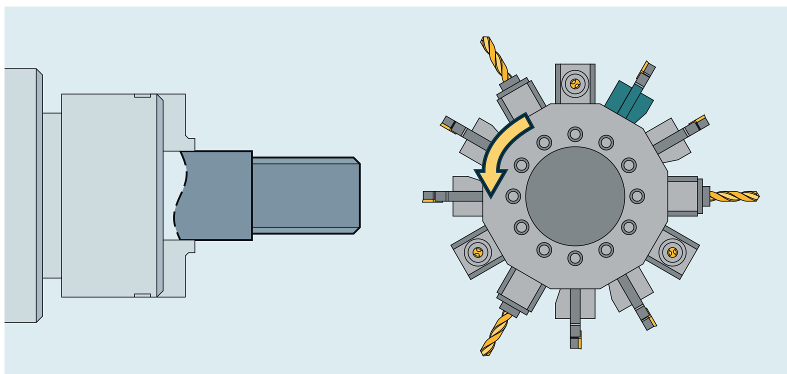
3.4 Tool change

Tool change method

Note

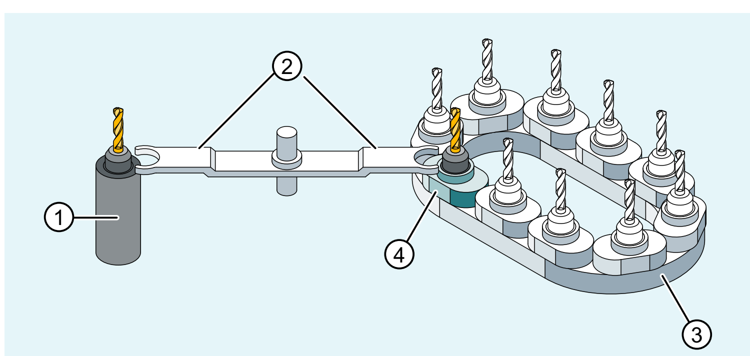
The type of tool change mechanism is specified by the machine OEM during the commissioning.

Tool changing on turning machines with tool turrets



In turret magazines on turning machines, the tool change, that is the search for and change of the tool, is called with the T command only.

Tool changing for machine tools with chain, rotary-plate or plane magazines



① Spindle
② Gripper
③ Magazine (here: chain magazine)
④ Change position for spindle

In chain, rotary-plate and plane magazines, a tool change normally takes place in two stages:

1. The tool is sought in the magazine with the T command.
2. The tool is then loaded into the spindle with the M command.

Programming of the tool change when tool management is active / not active

Programming of the tool change when tool management is active / not active differs from programming of the tool change with deactivated tool management. Both variants are therefore described:

→ Tool change with active tool management (Page 60)

→ Tool change with deactivated tool management (Page 64)

Programming of the working plane

The appropriate machining plane (Page 33) has to be programmed for the tool change (initial state: G18). This ensures that the tool length compensation is assigned to the correct axis.

Activation of the tool offset

The tool change activates the tool offset values stored under a D number (Page 90).

3.4.1 Tool change with active tool management

Tool management

The "Tool management" function ensures that at any given time the correct tool is in the correct location and that the data assigned to the tool are up to date. It also allows fast tool changes and avoids both scrap by monitoring the tool service life and machine downtimes by considering replacement tools.

Tool name

On a machine tool with active tool management, the tools must be assigned a name and number for clear identification (e.g. "Drill", "3").

The tool can then be called with the tool name, e.g.
`T="Drill"`

Note

The tool name may not contain any special characters.

3.4.1.1 Tool change with active tool management with the function T=location

A direct tool change takes place if the function T=location is programmed.

Application

For turning machines with circular magazine.

Syntax

Selecting a tool

```
T=<No>
T=<Name>
T<n>=<No>
T<n>=<Name>
```

Deselecting a tool

```
T0
```

Meaning

<table>
<tr><td rowspan="3">T=:</td><td colspan="2">Address for tool selection including tool change and activation of the tool offset
The following values can be assigned:</td></tr>
<tr><td><No>:</td><td>Number of the magazine location</td></tr>
<tr><td><name>:</td><td>Name of tool
Note:
The correct notation (uppercase/lowercase) must be used when programming a tool name.</td></tr>
<tr><td><n>:</td><td colspan="2">Spindle number as address extension
Note:
The possibility of programming a spindle number as an address extension depends on the configuration of the machine (→ see machine manufacturer's specifications).</td></tr>
<tr><td>T0:</td><td colspan="2">Tool deselection (magazine location unoccupied)</td></tr>
</table>

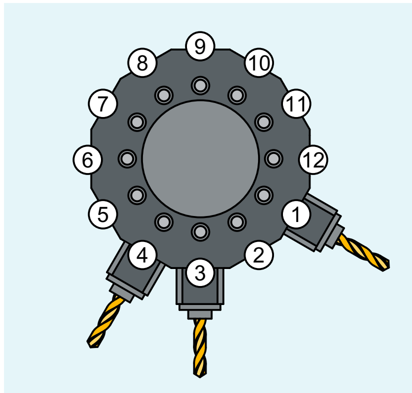
Note

If the selected magazine location is not occupied in a tool magazine, the command acts as for T0. The selection of the unoccupied magazine location can be used to position the empty location.

Example

A circular magazine has locations 1 to 12 with the following tool assignment:

Location	Tool	Internal T number	Status
1	Drill, duplo no. = 1	T15	Disabled
2	Not occupied		
3	Drill, duplo no. = 2	T10	Enabled
4	Drill, duplo no. = 3	T1	Active
5 ... 12	Not occupied		



① ... ⑫ Magazine/location number

The following tool call is programmed in the NC program:

```
N10 T=1
```

The call is processed as follows:

1. Magazine location 1 is considered and the tool identifier determined.
2. The tool management recognizes that this tool is blocked and therefore cannot be used.
3. A tool search for T="drill" is initiated in accordance with the set search strategy:
"Find the active tool; or select the one with the next highest duplo number."
4. The following usable tool is then found:
"Drill", duplo no. 3 (at magazine location 4)
This completes the tool selection process and the tool change is initiated.

Note

If the "Select the first available tool from the group" search strategy is employed, the order of the groups is not defined. In this case, T10 **or** T1 is selected.

When the search strategy "Take the first tool with "active" status from the group" is applied, T1 is loaded.

3.4.1.2 Tool change with active tool management with M6

The tool is selected when the T command is programmed. The tool only becomes active with M6 (including tool offset).

Application

For machine tools with chain, rotary-plate or plane magazines.

Syntax

Selecting a tool

```
T=<No>
T=<Name>
T<n>=<No>
T<n>=<Name>
```

Tool change

```
M6
```

Deselecting a tool

```
T0
```

Meaning

T=:	Address for the tool selection The following values can be assigned:	
	<No>:	Number of the magazine location
	<name>:	Name of tool **Note:** The correct notation (uppercase/lowercase) must be used when programming a tool name.
<n>:	Spindle number as address extension **Note:** The possibility of programming a spindle number as an address extension depends on the configuration of the machine (→ see machine manufacturer's specifications).	
M6:	M function for the tool change (according to DIN 66025) M6 activates the selected tool (T...) and the tool offset (D...).	
T0:	Tool deselection (magazine location unoccupied)	

Example

Program code	Comment
N10 T=1 M6	; Loading of the tool from magazine location 1.
N20 D1	; Selection of tool length compensation.
N30 G1 X10 ...	; Machining with tool T=1.
...	
N70 T="Drill"	; Preselection of the tool with name "Drill".
N80 ...	; Machining with tool T=1.
...	
N100 M6	; Loading of tool T="Drill".
N140 D1 G1 X10 ...	; Machining with tool T="Drill".
...	

3.4.2 Tool change with deactivated tool management

3.4.2.1 Tool change with deactivated tool management with T number

A direct tool change takes place if the T number is programmed.

Application

For turning machines with circular magazine.

Syntax

Selecting a tool

```
T<No>
T=<No>
T<n>=<No>
```

Deselecting a tool

```
T0
T0=<No>
```

Meaning

`T`:	Address for tool selection including tool change and activation of the tool offset	
`<No>`:	Number of the tool	
	Range of values:	0 ... 32000
`<n>`:	Spindle number as address extension **Note:** The possibility of programming a spindle number as an address extension depends on the configuration of the machine (→ see machine manufacturer's specifications).	
`T0`:	Deselecting the active tool	

Example

Program code	Comment
`N10, T1, D1`	`; Loading of tool T1 and activation of the tool offset D1.`
`...`	
`N70 T0`	`; Deselect tool T1.`
`...`	

3.4.2.2 Tool change with deactivated tool management with M6

The tool is selected when the T command is programmed. The tool only becomes active with M6 (including tool offset).

Application

For machine tools with chain, rotary-plate or plane magazines.

Syntax

Selecting a tool
```
T<No>
T=<No>
T<n>=<No>
```

Tool change
```
M6
```

Deselecting a tool
```
T0
T0=<No>
```

Meaning

`T:`	Address for the tool selection	
`<No>:`	Number of the tool	
	Range of values:	0 ... 32000
`<n>:`	Spindle number as address extension **Note:** The possibility of programming a spindle number as an address extension depends on the configuration of the machine (→ see machine manufacturer's specifications).	
`M6:`	M function for the tool change (according to DIN 66025) M6 activates the selected tool (T...) and the tool offset (D...).	
`T0:`	Deselecting the active tool	

Example

Program code	Comment
`N10 T1 M6`	`; Loading of tool T1.`
`N20 D1`	`; Selection of tool length compensation.`
`N30 G1 X10 ...`	`; Machining with T1.`
`...`	
`N70 T5`	`; Preselection of tool T5.`
`N80 ...`	`; Machining with T1.`
`...`	
`N100 M6`	`; Loading of tool T5.`
`N110 D1 G1 X10 ...`	`; Machining with tool T5`
`...`	

3.4.3 Behavior with faulty T programming

The behavior with faulty T programming depends on the configuration of the machine:

MD22562 TOOL_CHANGE_ERROR_MODE		
Bit	**Value**	**Meaning**
7	0	Basic setting! With the T programming, a check is made immediately as to whether the NC recognizes the T number. If not, an alarm is triggered.
	1	The programmed T number will only be checked following D selection. If the NC does not recognize the tool number, an alarm is issued during D selection. This response is desirable if, for example, tool programming is also intended to achieve positioning and the tool data is not necessarily available (circular magazine).

3.5 Tool offsets

3.5.1 Programmed contour and tool path

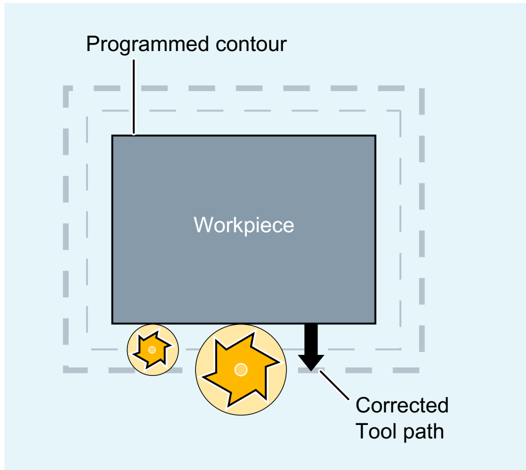
Workpiece dimensions are programmed directly (e.g. according to the production drawing). Therefore, tool data, such as milling tool diameter, cutting edge position of the turning tool (counterclockwise/clockwise turning tool) and tool length, does not have to be taken into consideration when creating the program.

The control corrects the travel path

When machining a workpiece, the tool paths are controlled according to the tool geometry so that the programmed contour can be created with any tool.

So that the control can calculate the tool paths, the tool data must be entered in the tool compensation memory of the control. Only the required tool (T...) and the required offset data record (D...) are called via the NC program.

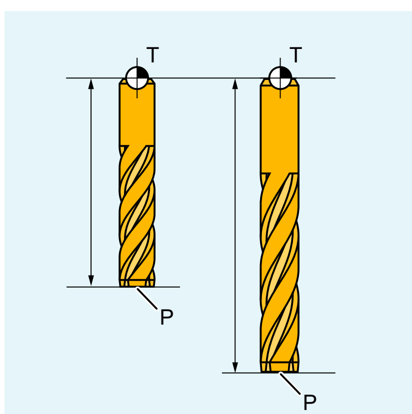
While the program is being processed, the control fetches the offset data it requires from the tool offset memory, and corrects the tool path individually for different tools:

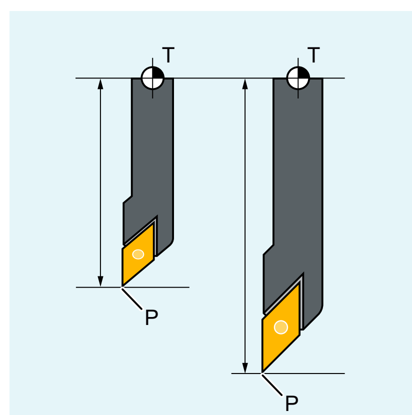


3.5.2 Tool length compensation

The tool length compensation compensates for the differences in length between the tools used.

The tool length is the distance between the tool carrier reference point and the tool tip:





T	Tool carrier reference point
P	Tool tip

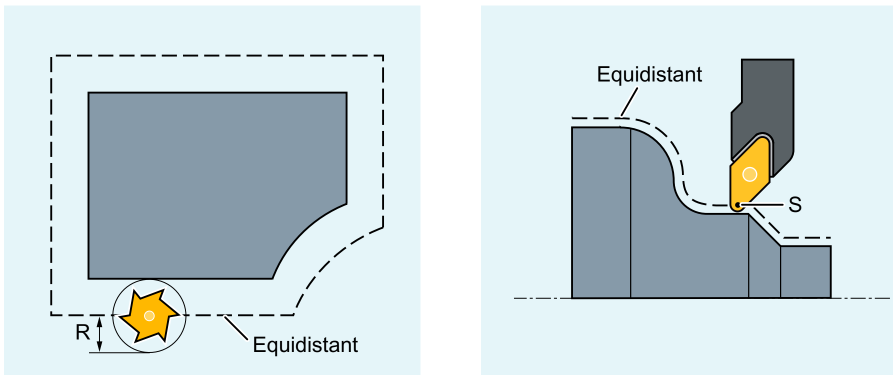
This length is measured and entered in the tool compensation memory of the control together with definable wear values. From this data, the control calculates the traversing movements in the infeed direction.

Note

The correction value of the tool length depends on the spatial orientation of the tool.

3.5.3 Tool radius compensation

The contour and tool path are not identical. The milling tool or cutting edge center point must travel along a path corresponding to the tool radius that is equidistant from the contour (tool center point path). To do this, while executing the program, the control shifts the programmed tool center point path – based on the radius of the active tool (tool offset memory) – so that the tool cutting edge traverses precisely along the programmed contour.



R Tool radius

S Cutting edge center point

The tool radius compensation is described in detail in the "Tool radius compensation (Page 254)" Chapter.

See also

2 1/2 D tool offset (CUT2D, CUT2DD, CUT2DF, CUT2DFD) (Page 283)

3.5.4 Tool compensation memory

The following data must be available in the tool offset memory of the control system for each tool edge:

- Tool type
- Cutting edge position
- Tool geometry variables (length, radius)

These data are entered as tool parameters (max. 40). Which parameters are required for a tool depends on the tool type. Any tool parameters that are not required must be set to "zero" (corresponds to the default setting of the system).

Note

Values that have been entered once in the offset memory are included in the processing at each tool call.

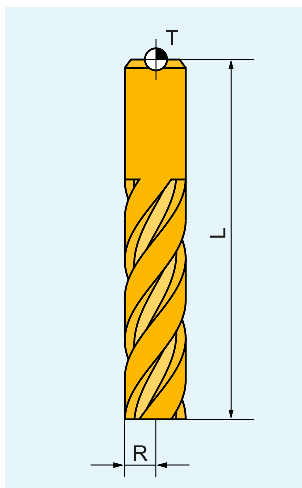
Tool type

The tool type (drill, milling or turning tool) determines which geometry data are necessary and how they are calculated.

Cutting edge position

The cutting edge position describes the position of the tool tip in relation to the cutting edge center point. The cutting edge position together with the cutting edge radius is required for the calculation of the tool radius compensation for turning tools (tool type 5xx) (Page 79).

Tool geometry variables (length, radius)



T	Tool carrier reference point
R	Tool radius
L	Tool length

The tool geometry variables consist of several components (geometry, wear). The control computes the components to a resulting size (e.g. total length 1, total radius). The relevant overall dimension becomes operative when the offset memory is activated.

How these values are calculated in the axes is determined by the tool type and the current plane (G17/G18/G19).

3.5.5 Tool types

3.5.5.1 Tool types and tool parameters (overview)

Tool type number and tool groups

Each tool type is assigned a unique 3-digit number. The assignment of the tool to one of the following technologies or tool groups is realized using the first digit (the hundreds position):

Tool type	Tool group
1xy	Milling tools (Page 72)
2xy	Drills (Page 76)
3xy	Reserved
4xy	Grinding tools (Page 77)
5xy	Turning tools (Page 79)
6xy	Reserved
7xy	Special tools (Page 86)

Relevant tool parameters

The parameter values to be entered in accordance with the tool type (tool offset data; TOA data) are stored in system variables ($TC_DPx and $TC_TPGx).

The following tables show which parameter values are required for which tool type:

Cutting edge parameter $TC_DPx, x =	1 Tool type	2 Cutting edge position	3 Length 1 (Geometry - tool length)	4 Length 2 (Geometry - tool length)	5 Length 3 (Geometry - tool length)	6 Radius 1 (Geometry - tool shape)	7 Radius 2 (Geometry - tool shape)	8 Length 4 (Geometry - tool shape)	9 Length 5 (Geometry - tool shape)	10 Angle 1 (Geometry - tool shape)	11 Angle 2 (Geometry - tool shape)	12 Length 1 (Wear - tool length)	13 Length 2 (Wear - tool length)	14 Length 3 (Wear - tool length)	15 Radius 1 (Wear - tool shape)	16 Radius 2 (Wear - tool shape)	17 Length 4 (Wear - tool shape)	18 Length 5 (Wear - tool shape)	19 Angle 1 (Wear - tool shape)	20 Angle 2 (Wear - tool shape)	21 Length 1 (Base dimension / adapter dimension)	22 Length 2 (Base dimension / adapter dimension)	23 Length 3 (Base dimension / adapter dimension)	24 Clearance angle	$TC_DPV Predefined ORI vector	$TC_DPVx ORI vector components	$TC_DPROT Clamping angle	36 Outside radius 2	37 Wear outside radius 2
Milling tools	100 Milling tool acc. to CLDATA [1]		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
	110 Ball end mill		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
	111 Cylindrical die-sinking milling cutter		✓	✓	✓	✓	✓					✓	✓	✓	✓	✓					✓	✓	✓		(✓)	(✓)		✓	
	114 Barrel cutter (barrel shape)		✓	✓	✓	✓	✓		✓			✓	✓	✓	✓	✓					✓	✓	✓		(✓)	(✓)		✓	✓
	115 Drop cutter (drop shape)		✓	✓	✓	✓	✓					✓	✓	✓	✓	✓					✓	✓	✓		(✓)	(✓)		✓	✓
	116 Conical barrel cutter (cone shape)		✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓					✓	✓	✓		(✓)	(✓)		✓	✓
	120 End mill		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
	121 End mill with corner rounding		✓	✓	✓	✓	✓					✓	✓	✓	✓	✓					✓	✓	✓		(✓)	(✓)			
	130 Angle head cutter		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
	131 Angle head mill with corner rounding		✓	✓	✓	✓	✓					✓	✓	✓	✓	✓					✓	✓	✓		(✓)	(✓)			
	140 Facing tool		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
	145 Thread cutter		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
	150 Side mill		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
	155 Bevel cutter		✓	✓	✓	✓			✓		✓	✓	✓	✓	✓			✓		✓	✓	✓	✓		(✓)	(✓)			
	156 Bevel cutter with corner rounding		✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓		(✓)	(✓)			
	157 Tapered die milling tool		✓	✓	✓	✓					✓	✓	✓	✓	✓					✓	✓	✓	✓		(✓)	(✓)			
	160 Drill and thread milling cutter		✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		(✓)	(✓)			
Drilling tools	200 Twist drill		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	205 Drill		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	210 Boring bar		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	220 Center drill		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	230 Countersink		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	231 Counterbore		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	240 Tap regular thread		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	241 Tap fine thread		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	242 Tap Whitworth thread		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
	250 Reamer		✓	✓	✓	✓						✓	✓	✓							✓	✓	✓		(✓)	(✓)			
Turning tools	500 Roughing tool	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				(✓)		
	505 Y axis roughing tool	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓		
	510 Finishing tool	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				(✓)		
	515 Y axis finishing tool	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓		
	520 Plunge cutter	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				(✓)		
	525 Y axis plunge cutter	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓		
	530 Cutting tool	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				(✓)		
	535 Y axis cutting tool	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓		
	540 Threading tool	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				(✓)		
	550 Button tool / Sectional steel (TOOLMAN)	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				(✓)		
	560 Rotary drill (ECOCUT)	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				(✓)		
	580 Probe with cutting edge position parameters	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓			(✓)		

[1] CLDATA = "cutter location data" (tool position data according to DIN66215)

Figure 3-3 Tool parameters for milling, drilling and turning tools

Group	x = 1: Tool type		$TC_DP 2: Cutting edge position	$TC_DP 3: Length 1 (Geometry - tool length)	$TC_DP 4: Length 2 (Geometry - tool length)	$TC_DP 5: Length 3 (Geometry - tool length)	$TC_DP 6: Radius 1 (Geometry - tool shape)	$TC_DP 7: Radius 2 (Geometry - tool shape)	$TC_DP 8: Length 4 (Geometry - tool shape)	$TC_DP 9: Length 5 (Geometry - tool shape)	$TC_DP 10: Angle 1 (Geometry - tool shape)	$TC_DP 11: Angle 2 (Geometry - tool shape)	$TC_DP 12: Length 1 (Wear - tool length)	$TC_DP 13: Length 2 (Wear - tool length)	$TC_DP 14: Length 3 (Wear - tool length)	$TC_DP 15: Radius 1 (Wear - tool shape)	$TC_DP 16: Radius 2 (Wear - tool shape)	$TC_DP 17: Length 4 (Wear - tool shape)	$TC_DP 18: Length 5 (Wear - tool shape)	$TC_DP 19: Angle 1 (Wear - tool shape)	$TC_DP 20: Angle 2 (Wear - tool shape)	$TC_DP 21: Length 1 (Base dimension / adapter dimension)	$TC_DP 22: Length 2 (Base dimension / adapter dimension)	$TC_DP 23: Length 3 (Base dimension / adapter dimension)	$TC_DP 24: Clearance angle	Tool parameter $TC_TPG 3: Min. wheel radius	$TC_TPG 4: Min. wheel width	$TC_TPG 5: Act. wheel width	$TC_TPG 6: Max. speed	$TC_TPG 7: Max. peripheral speed	$TC_TPG 8: Angle of inclined wheel
Grinding tools	400	Surface grinding wheel	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓						✓	✓
Grinding tools	401	Surface grinding wheel with monitoring	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		✓	✓	✓	✓	✓	✓
Grinding tools	403	As with 401, but without base dimension for GWPS [1]	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		✓	✓	✓	✓		✓
Grinding tools	410	Faceplate	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓						✓	
Grinding tools	411	Faceplate with monitoring	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		✓	✓	✓	✓	✓	
Grinding tools	412	Faceplate without monitoring	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		✓	✓	✓	✓		
Grinding tools	413	As with 411, but without base dimension for GWPS [1]	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓		✓	✓	✓	✓		
Grinding tools	490	Dresser	✓	✓	✓	✓	✓						✓	✓	✓	✓						✓	✓	✓							
SW [2]	700	Grooving saw		✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓				✓	✓	✓							

[1] GWPS = Grinding wheel peripheral speed

[2] ST = Special tools

Figure 3-4 Tool parameters for grinding and special tools

Note

Unlisted numbers are also permitted, in particular with grinding tools (400-499).

3.5.5.2 Milling tools

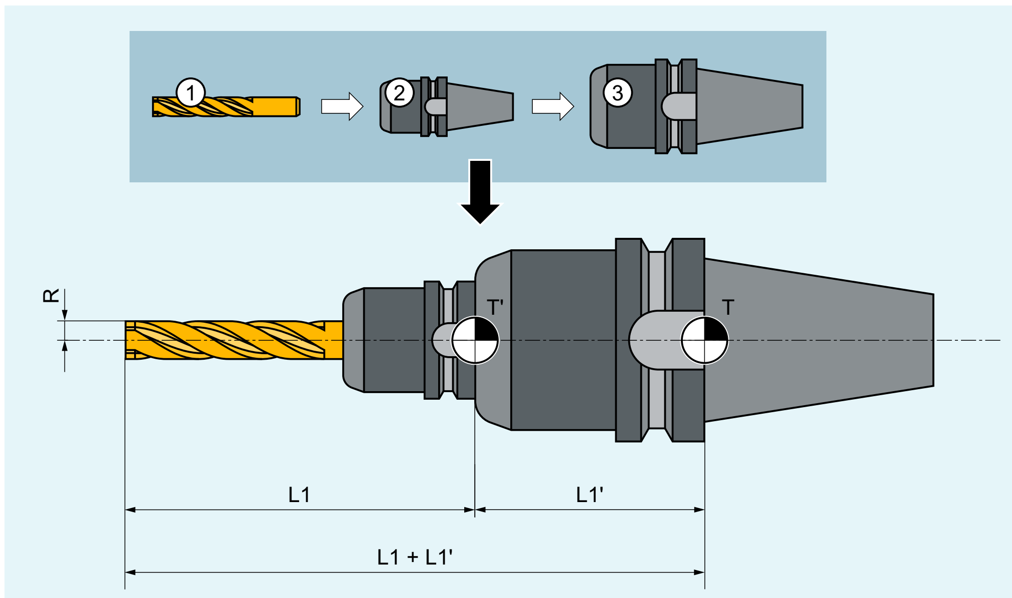
The following tool types are available in the "Milling tools" group:

100	Milling tool according to CLDATA (Cutter Location Data)
110	Ball end mill
111	Cylindrical die-sinking milling cutter
114	Barrel cutter (barrel shape)
115	Drop cutter (drop shape)
116	Conical barrel cutter (cone shape)
120	End milling cutter without corner rounding
121	End mill with corner rounding
130	Angle head cutter without corner rounding
131	Angle head mill with corner rounding
140	Facing tool
145	Thread cutter
150	Side mill
151	Saw
155	Bevel cutter without corner rounding
156	Bevel cutter with corner rounding

157	Tapered die milling tool
160	Drill and thread milling cutter

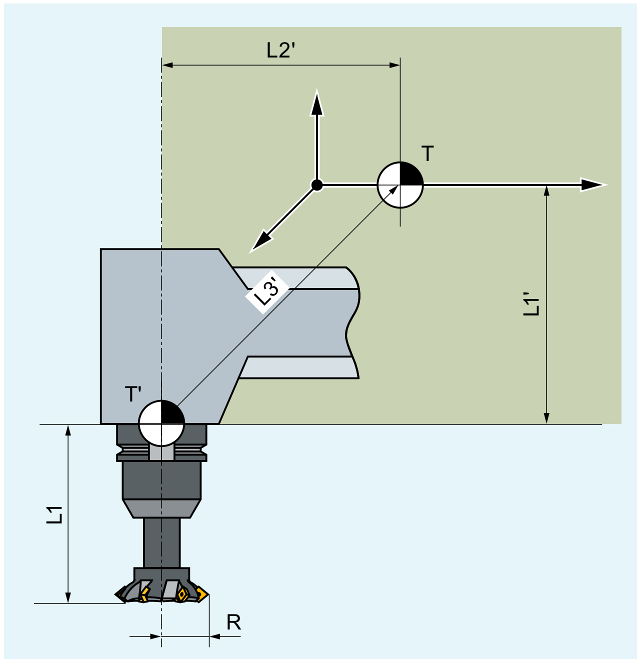
Tool parameters

The following diagrams provide an overview of which milling tool parameters are entered in the compensation memory:



①	Tool
②	Tool holder
③	Tool adapter
T	Adapter reference point (for inserted tool = tool carrier reference point)
T'	Tool carrier reference point
L1	Geometry - length 1
L1'	Adapter dimension - length 1
L1 + L1'	Total length L1
R	Radius

Tool parameters	Meaning
$TC_DP1	Tool type 1xy
$TC_DP3	Geometry - length 1
$TC_DP6	Geometry - radius
$TC_DP21	Adapter dimension - length 1
• Wear values corresponding to the requirements. • Other values should be set to 0.	



T	Tool carrier reference point
T'	Tool carrier reference point
L1	Geometry - length 1
R	Tool radius
L1'	Base dimension - length 1
L2'	Base dimension - length 2
L3'	Base dimension - length 3

Tool parameters	Meaning
$TC_DP1	Tool type
$TC_DP3	Geometry - length 1
$TC_DP6	Geometry - radius
$TC_DP21	Base dimension - length 1
$TC_DP22	Base dimension - length 2
$TC_DP23	Base dimension - length 3
• Wear values corresponding to the requirements. • Other values should be set to 0.	

Tool parameters $TC_DP6 ... $TC_DP11: Tool shape

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 the tool parameter $TC_DP6 (tool radius 1) is used.

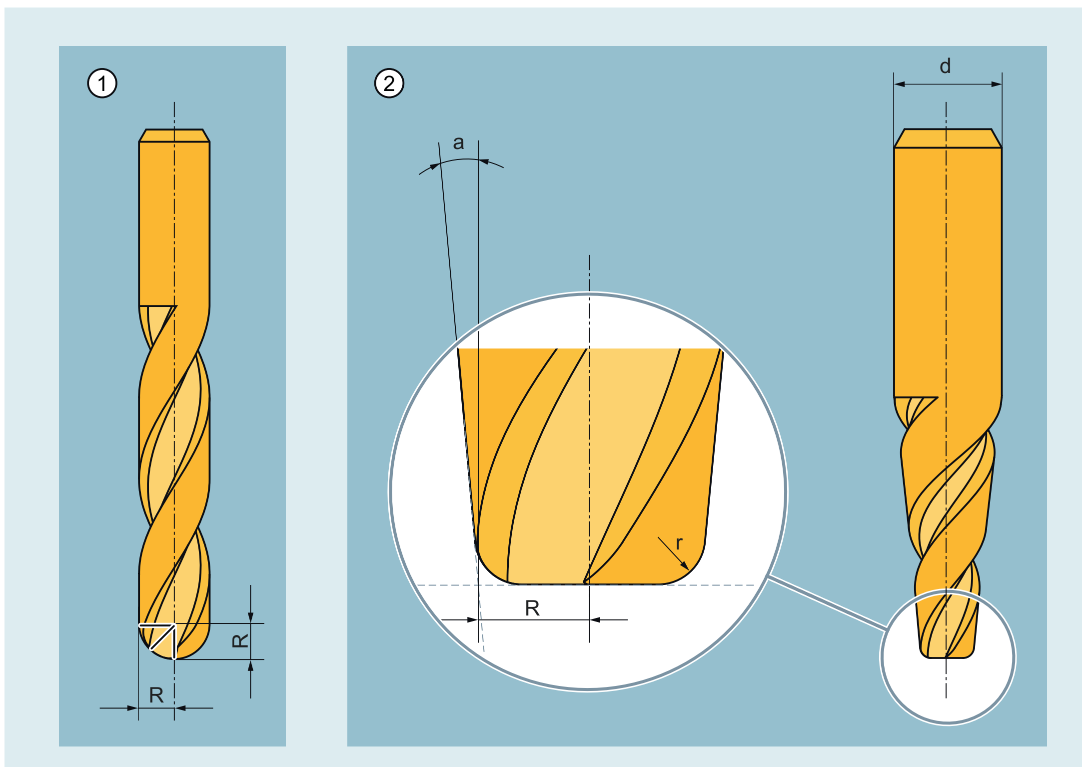
2D TRC with contour tools

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

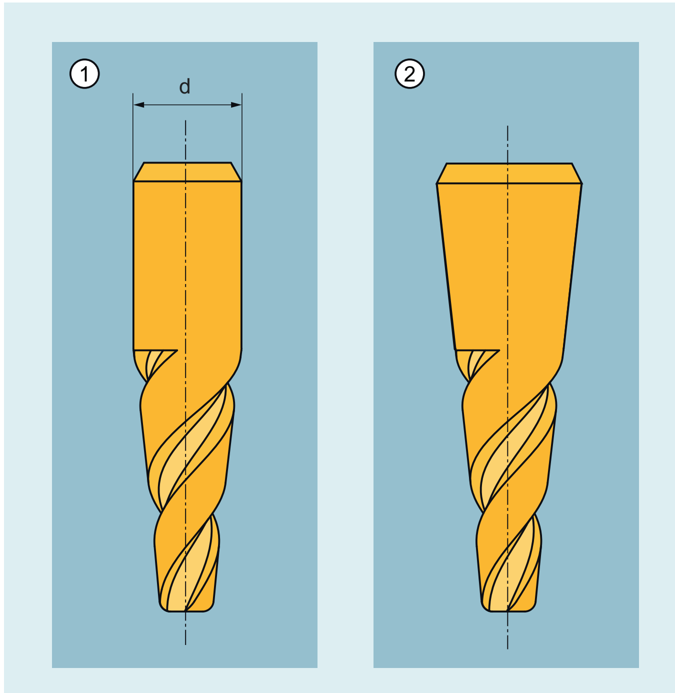
3D face milling

The tool parameters relevant to the tool description in 3D face milling are dependent on the tool type used. Thus, for example, for a ball end mill, only tool parameter 6, for a bevel cutter with corner rounding additionally tool parameters 7, 9 and 11, are relevant.



① Ball end mill
② Bevel cutter with corner rounding
R Tool parameter 6: Tool radius
r Tool parameter 7: Corner radius
d Tool parameter 9: Upper bevel cutter diameter
a Tool parameter 11: Angle between envelope line and tool longitudinal axis

Specification of tool parameter 9 (upper bevel diameter) is optional. If this tool parameter is not specified, a conical shape stretching along the entire tool length is assumed:



① With specification of tool parameter 9
② Without specification of tool parameter 9

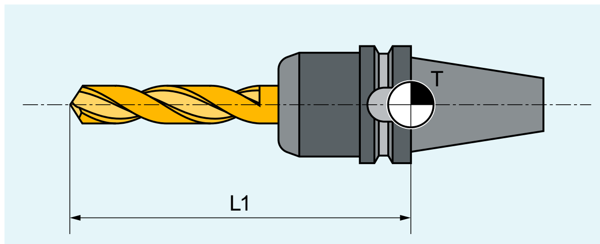
3.5.5.3 Drills

The following tool types are available in the "Drills" group:

No.	Tool type
200	Twist drill
205	Drill
210	Boring bar
220	Center drill
230	Countersink
231	Counterbore
240	Tap regular thread
241	Tap fine thread
242	Tap Whitworth thread
250	Reamer

Tool parameters

The following diagram provides an overview of which drill tool parameters are entered in the compensation memory:



T Tool carrier reference point

L1 Length 1

Tool parameters	Meaning
$TC_DP1	Tool type
$TC_DP3	Geometry - length 1
• Wear values corresponding to the requirements. • Other values should be set to 0.	

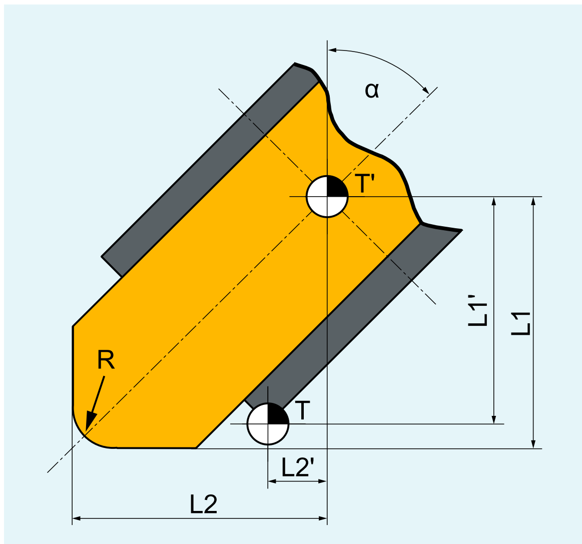
3.5.5.4 Grinding tools

The following tool types are available in the "Grinding tools" group:

400	Surface grinding wheel
401	Surface grinding wheel with monitoring
402	Surface grinding wheel without monitoring without base dimension (TOOLMAN)
403	Surface grinding wheel with monitoring without base dimension for grinding wheel peripheral speed GWPS
410	Facing wheel
411	Facing wheel (TOOLMAN) with monitoring
412	Facing wheel (TOOLMAN) without monitoring
413	Facing wheel with monitoring without base dimension for grinding wheel peripheral speed GWPS
490	Dresser

Tool parameters

The following diagram provides an overview of which grinding tool parameters are entered in the compensation memory:



T	Tool carrier reference point
T'	Tool holder reference point
L1	Geometry - length 1
L1'	Base dimension - length 1
L2	Geometry - length 2
L2'	Base dimension - length 2
R	Radius
α	Angle of inclined wheel

Cutting edge-specific parameters	Meaning
$TC_DP1	Tool type 4xy
$TC_DP2	Cutting edge position
$TC_DP3	Geometry length 1
$TC_DP4	Geometry length 2
$TC_DP6	Radius
$TC_DP21	Base dimension length 1
$TC_DP22	Base dimension length 2
• Wear values corresponding to the requirements. • Set other values to 0.	

Tool-specific parameters	Meaning
$TC_TPG1	Spindle number
$TC_TPG2	Chaining rule [1)]
$TC_TPG3	Minimum wheel radius

Tool-specific parameters	Meaning
$TC_TPG4	Minimum wheel width
$TC_TPG5	Actual wheel width
$TC_TPG6	Maximum speed
$TC_TPG7	Maximum circumferential velocity
$TC_TPG8	Angle of inclined wheel
$TC_TPG9	Parameter number for radius calculation
$TC_TPG_DRSPATH	Directory path to the dressing program
$TC_TPG_DRSPROG	Dressing program name

[1] The geometry length compensations, wear and base dimension can be chained for the left and right tool nose radius compensation. This means that the length compensations for the left cutting edge are changed so that the values are also automatically entered for the right cutting edge, and vice versa.

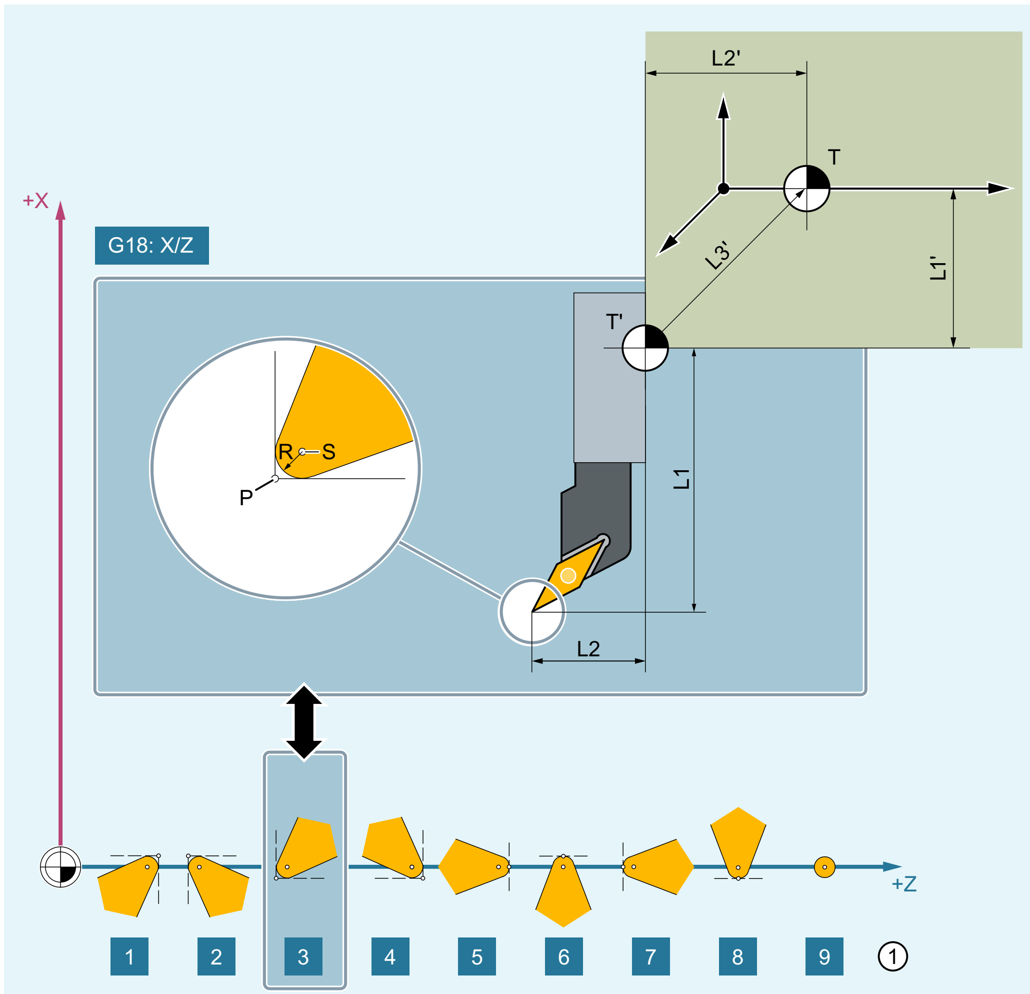
3.5.5.5 Turning tools

The following tool types are available in the "Turning tools" group:

500	Roughing tool
505	Y axis roughing tool
510	Finishing tool
515	Y axis finishing tool
520	Plunge cutter
525	Y axis plunge cutter
530	Cutting tool
535	Y axis cutting tool
540	Threading tool
550	Button tool / forming tool (TOOLMAN)
560	Rotary drill (ECOCUT)
580	Probe with cutting edge position parameters

Tool parameters

The following diagram provides an overview of which turning tool parameters are entered in the compensation memory:

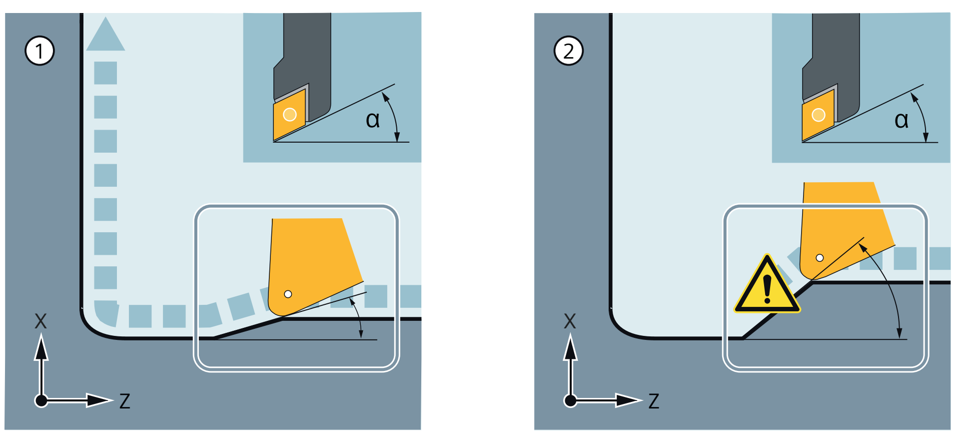


①	Cutting edge position (1 ... 9) for machining behind the turning center
P	Tool tip
S	Cutting edge center point
R	Cutting edge radius
T	Tool carrier reference point
T'	Tool carrier reference point
L1	Geometry - length 1
L2	Geometry - length 2
L1'	Base dimension - length 1
L2'	Base dimension - length 2
L3'	Base dimension - length 3

Tool parameters	Meaning
$TC_DP1	Tool type
$TC_DP2	Cutting edge position The cutting edge position describes the position of the tool tip P in relation to the cutting edge center point S. The cutting edge position together with the cutting edge radius ($TC_DP6) is required for the calculation of the tool radius compensation for turning tools.
$TC_DP3	Geometry - length 1
$TC_DP4	Geometry - length 2
$TC_DP6	Geometry - radius (cutting edge radius)
$TC_DP21	Base dimension - length 1
$TC_DP22	Base dimension - length 2
$TC_DP23	Base dimension - length 3
• Wear values corresponding to the requirements. • Other values should be set to 0.	

Tool parameter $TC_DP24: Clearance angle

Certain turning cycles, in which traversing motions with relief cuts are generated, monitor the clearance angle of the active tool for possible contour violation.



α Clearance angle

① The contour to be machined is not violated.

② The contour to be machined would be violated.

The clearance angle is entered in tool parameter 24.

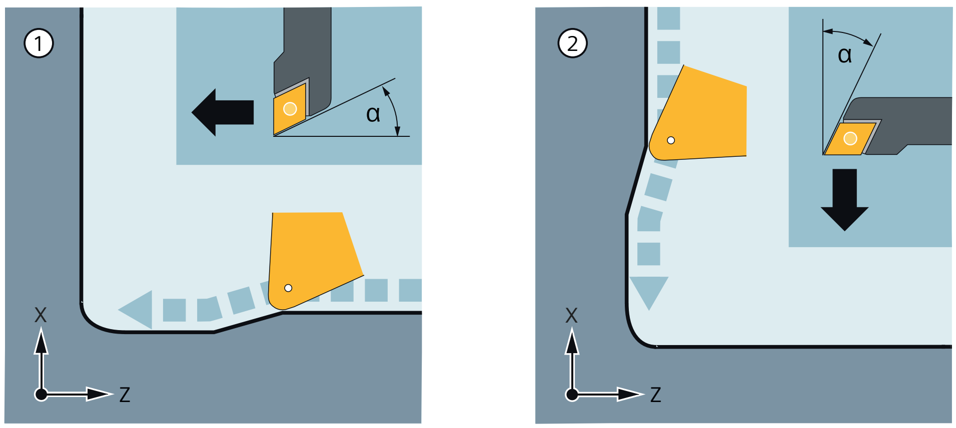
Value range: 0° ... 90° (without sign)

Note

If a clearance angle of zero is entered in tool parameter 24, then relief cuts are not monitored in the turning cycles.

Longitudinal or face machining

The clearance angle is entered differently according to the type of processing. If a tool is to be used for both longitudinal **and** face machining, two tool cutting edges must be entered for different clearance angles.



α	Clearance angle
①	Longitudinal machining
②	Face machining

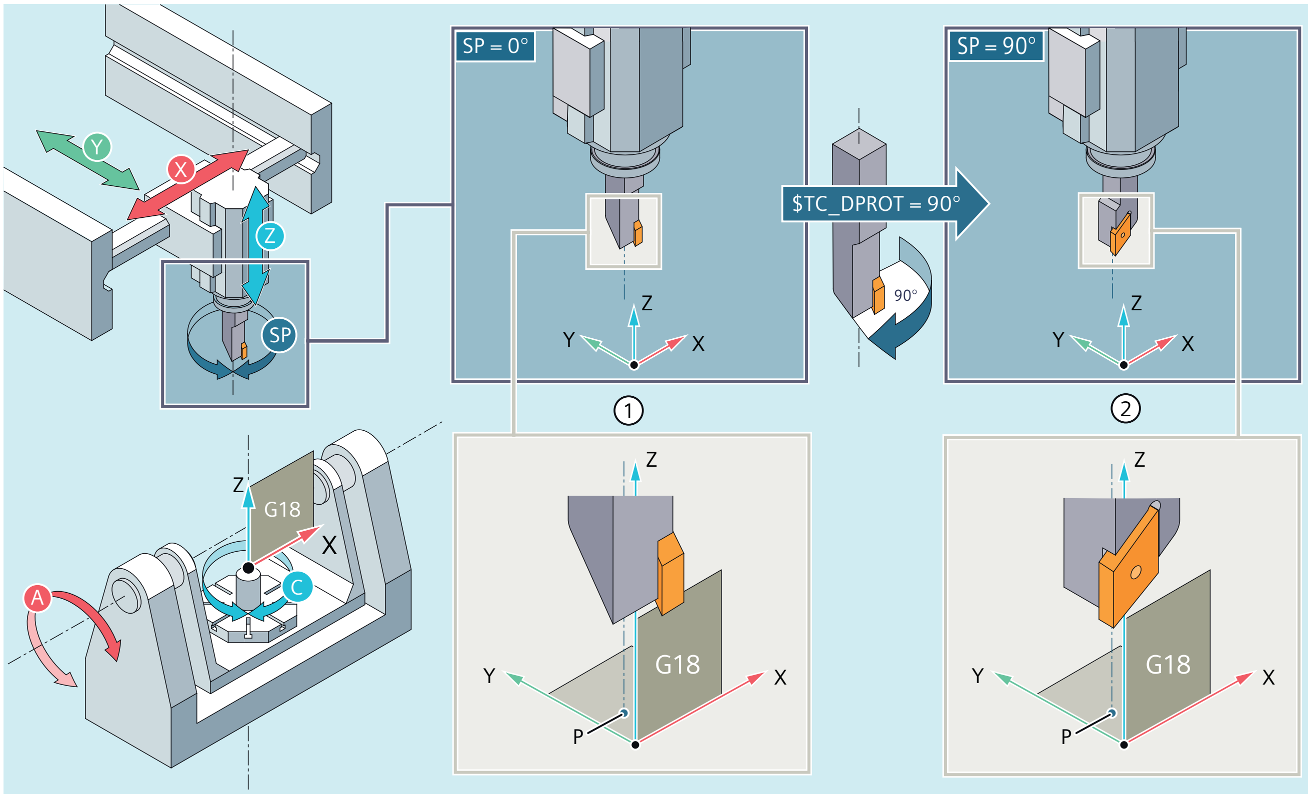
Tool parameter $TC_DPROT: Clamping angle

In certain applications, it is necessary to specify the clamping angle via the tool parameter $TC_DPROT. This is the angle by which the tool spindle must rotate out of the spindle zero position to bring the clamped turning tool into the position in which the tool is to be dimensioned.

Application: Conventional turning tools in combination with the function "Turning on milling machine (TRAORI_STAT)" or "Interpolation turning (TRAINT)"

$TC_DPROT must be set when the clamped turning tool in the spindle zero position is not in the G18 plane. The angle must be specified through which the tool spindle has to be rotated out of the spindle zero position to move the tool into the correct position for machining in the G18 plane.

The following figure illustrates this using the example "Interpolation turning (TRAINT)":



① In the spindle zero position (SP = 0°), the cutting edge of the clamped turning tool is not in the G18 plane.

② Only after rotation of the tool spindle through the clamping angle specified in tool parameter $TC_DPROT (in this case: +90°) is the tool cutting edge in the correct position for machining in the G18 plane.

P Imaginary point of intersection of the tool spindle axis with the XY plane

Application: Y axis turning tools on the turning machine in conjunction with CYCLE805

$TC_DPROT must be set when the clamped Y turning tool in the basic position is not in the position in which the tool is to be dimensioned. Specify the angle by which the tool must be rotated from the basic position in order to bring it into the correct position for dimensioning.

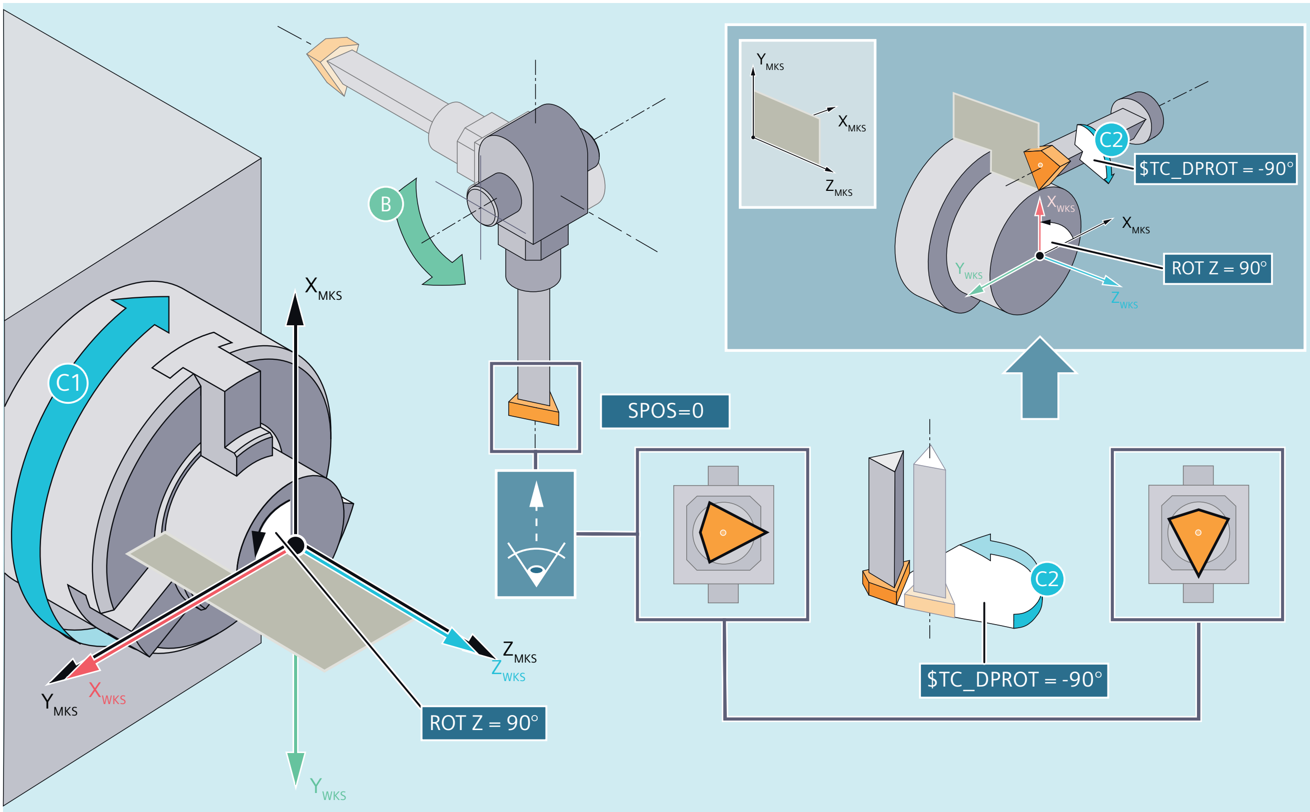
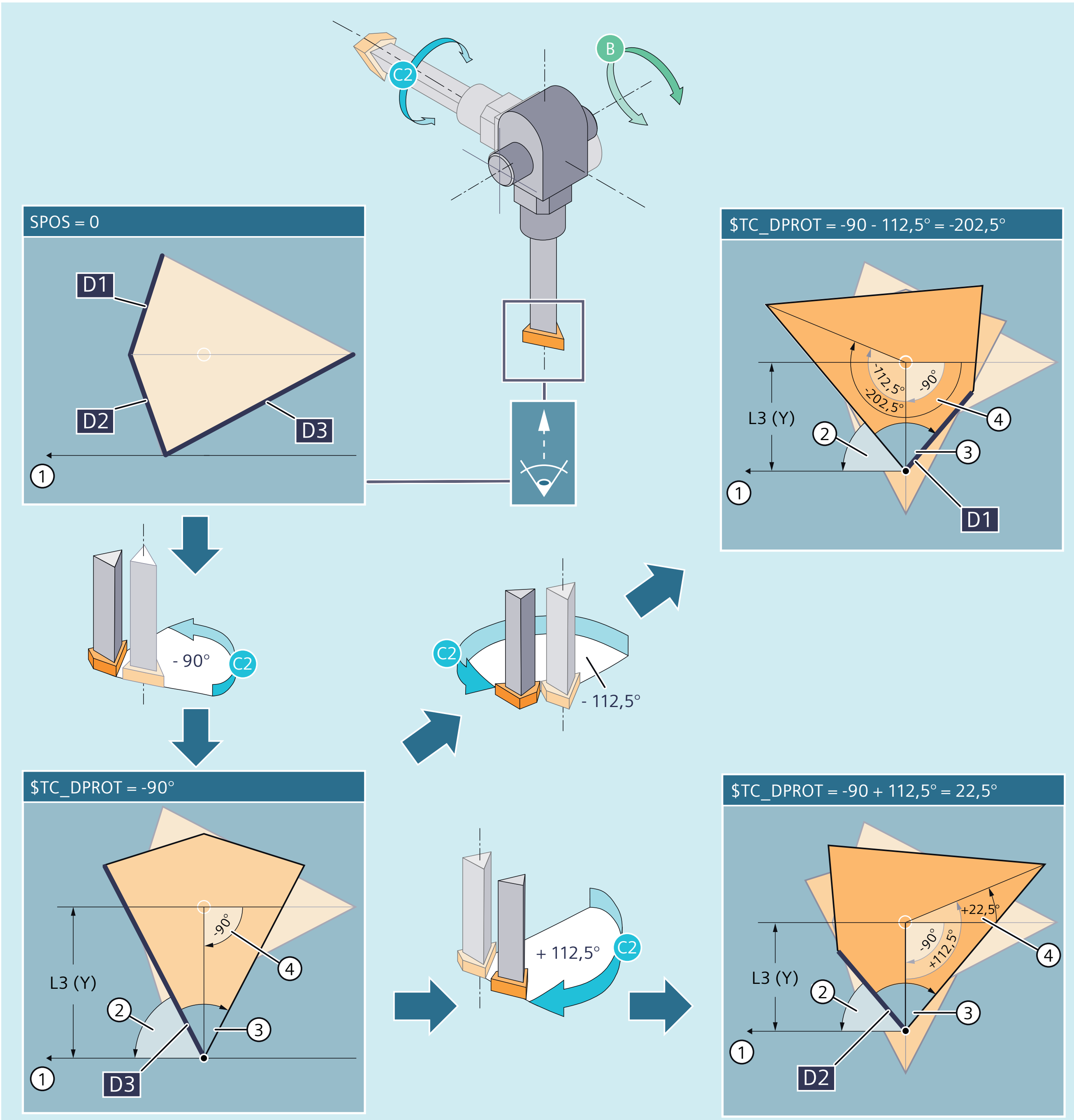


Figure 3-5 Y turning in working range 2 (WCS rotation around Z: +90°): Tool positioning via the setting angle $TC_DPROT

Example:

On a Y axis turning tool with three cutting edges (D1 ... D3), all three cutting edges are to be used. For each cutting edge, the clamping angle $TC_DPROT required for dimensioning must be entered:

Cutting edge	Clamping angle $TC_DPROT	
D1	-90° + (-112.5°)	= -202.5°
D2	-90° + 112.5°	= 22.5°
D3	- 90° + 0°	= -90°



① Reference direction
② Holder angle
③ Cutting tip angle
④ Clamping angle

Figure 3-6 Clamping angle $TC_DPROT of a Y turning tool with three cutting edges

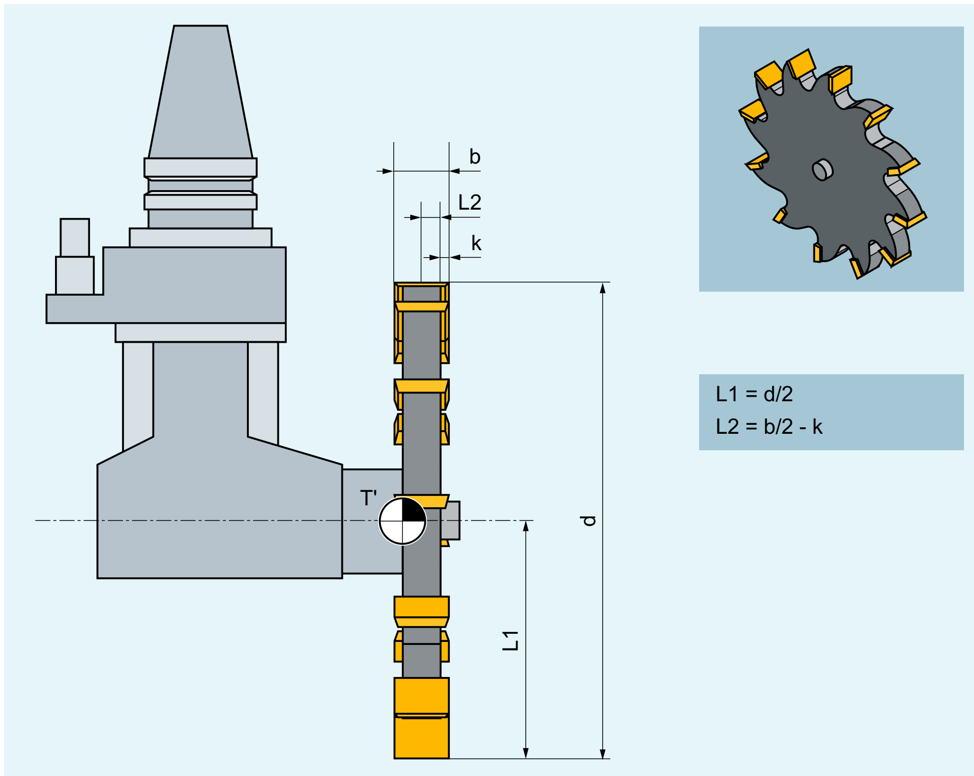
3.5.5.6 Special tools

The following tool types are available in the "Special tools" group:

700	Slotting saw
710	3D probe
711	Edge probe
712	Mono probe
713	L probe
714	Star probe
725	Calibration tool
730	Stop
731	Spindle sleeves
732	End support

Tool parameters

The following diagram provides an overview of which tool parameters for "Slotting saw" tool type are entered in the compensation memory:



T' Tool carrier reference point
L1 Geometry - length 1
L2 Geometry - length 2
d Diameter
b Slot width
k Projection

Tool parameters	Meaning
$TC_DP1	Tool type
$TC_DP3	Geometry - length 1
$TC_DP4	Geometry - length 2
$TC_DP6	Diameter
$TC_DP7	Slot width
$TC_DP8	Projection
$TC_DP21	Base dimension length 1
$TC_DP22	Base dimension length 2
$TC_DP23	Base dimension length 3
• Wear values corresponding to the requirements. • Other values should be set to 0.	

3.5.6 Basic tool orientation ($TC_DPV[...], $TC_DPV3 - 5[...] and $TC_DPVN3 - 5[...])

If the function "Parameterizable basic tool orientation" is active (→ MD18114 $MN_MM_ENABLE_TOOL_ORIENTATION), a separate basic orientation can be assigned with the system variable $TC_DPV[...] or the system variables $TC_DPV3 - 5[...] and $TC_DPVN3 - 5[...] to each tool cutting edge.

Setting options

Basically, the following setting options are available:

- **$TC_DPV[...] == 0 AND $TC_DPV3 - 5[...] == 0**
  The vector for the basic tool orientation results from the active working plane:
  - G17: Z coordinate
  - G18: Y coordinate
  - G19: X coordinate
- **$TC_DPV[...] == 0 AND $TC_DPV3 - 5[...] <> 0**
  The vector for the basic tool orientation is prescribed by $TC_DPV3 - 5[...] and $TC_DPVN3 - 5[...]:
  - `$TC_DPV3[...] = <L1 component of the orientation vector>`
  - `$TC_DPV4[...] = <L2 component of the orientation vector>`
  - `$TC_DPV5[...] = <L3 component of 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>`

  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
```

  Note

  The system variables $TC_DPVN3-5[...] are not relevant for rotationally symmetrical tools such as milling tools.

- **$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 3-1 Turning / grinding tools ($TC_DP1 = 400 ... 599)

<value>	$TC_DPV3[...] / $TC_DPVN3[...]	$TC_DPV5[...] / $TC_DPVN5[...]	$TC_DPV4[...] / $TC_DPVN4[...]
1	0 / 0	1 / 0	0 / 1
2	1 / 0	0 / 1	0 / 0
3	0 / 1	0 / 0	1 / 0
4	0 / 0	-1 / 0	0 / -1
5	-1 / 0	0 / -1	0 / 0
6	0 / -1	0 / 0	-1 / 0

Table 3-2 Milling / special tools ($TC_DP1 <> 400 ... 599)

<value>	$TC_DPV5[...] / $TC_DPVN5[...]	$TC_DPV4[...] / $TC_DPVN4[...]	$TC_DPV3[...] / $TC_DPVN3[...]
1	0 / 1	0 / 0	1 / 0
2	0 / 0	1 / 0	0 / 1
3	1 / 0	0 / 1	0 / 0
4	0 / -1	0 / 0	-1 / 0
5	0 / 0	-1 / 0	0 / -1
6	-1 / 0	0 / -1	0 / 0

Examples

Turning/grinding tools:

`$TC_DPV[...] = 2`	Corresponds to	`$TC_DPV3[...] = 1`
		`$TC_DPV4[...] = 0`
		`$TC_DPV5[...] = 0`
		`$TC_DPVN3[...] = 0`
		`$TC_DPVN4[...] = 0`
		`$TC_DPVN5[...] = 1`

Milling/special tools:

`$TC_DPV[...] = 3`	Corresponds to	`$TC_DPV3[...] = 0`
		`$TC_DPV4[...] = 0`
		`$TC_DPV5[...] = 1`
		`$TC_DPVN3[...] = 0`
		`$TC_DPVN4[...] = 1`
		`$TC_DPVN5[...] = 0`

3.5.7 Activating / deactivating tool offsets (D, D0):

Cutting edges 1 to 8 of a tool (with active tool management 12) can be assigned different tool offset data blocks (e.g. different correction values for the left and right cutting edges of a grooving tool).

The offset data (including the data for the tool length compensation) of a special cutting edge is activated by calling the D number. When D0 is programmed, offsets for the tool have no effect.

A tool radius compensation must also be activated by G41/G42.

Note

Tool length compensations take immediate effect when the D number is programmed. If no D number is programmed, the default setting defined by the machine data is active for a tool change (→ see machine manufacturer's specifications).

Syntax

```
D<No>
X... Y... Z...
G41/G42 X... Y... Z...
G40
D0
```

Meaning

<table>
<tr><td>D:</td><td colspan="2">Address for activating an offset data block for the active tool
The tool length compensation is applied with the first programmed traverse of the associated length compensation axis.
Notice:
A tool length compensation can also take effect without D programming, if the automatic activation of a tool edge has been configured for the tool change (→ see machine manufacturer's specifications).</td></tr>
<tr><td rowspan="2"><No>:</td><td colspan="2">The tool offset data block to be activated is specified via the D number.
The type of D programming depends on the configuration of the machine (→ see information provided by the machine manufacturer).
There are the following options:
• D number = cutting edge number
D numbers ranging from 1 to max. 12 are available for every tool T<No> (without TOOLMAN) or T="Name" (with TOOLMAN). These D numbers are assigned directly to the tool cutting edges. An offset data set ($TC_DPx[<t>,<d>]) is assigned to each D number (= cutting edge number).
• Free selection of D numbers
The D numbers can be freely assigned to the cutting edge numbers of a tool. A machine data specifies the upper limit for the D numbers that may be used.</td></tr>
<tr><td>Range of values:</td><td>0 ... 32000</td></tr>
<tr><td>D0:</td><td colspan="2">Deactivating the offset data block for the active tool</td></tr>
</table>

`G41`:	Command for activating the tool radius compensation with machining direction **left** of the contour
`G42`:	Command for activating the tool radius compensation with machining direction **right** of the contour
`G40`:	Command for deactivating the tool radius compensation

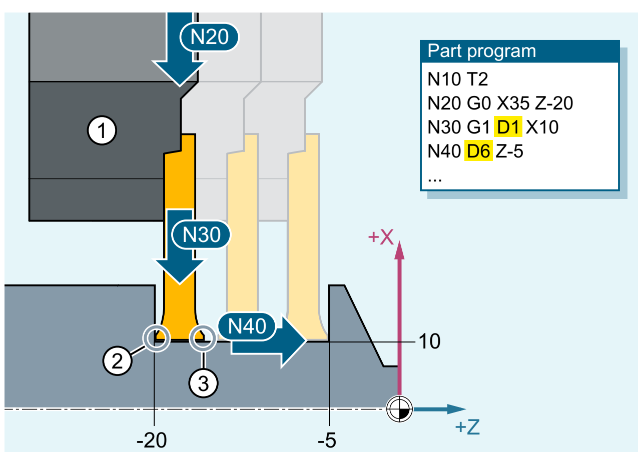
Additional information on G40/G41/G42 is provided in the section "Tool radius compensation (Page 254)".

Examples

Example 1: Tool change with T command (turning)

Program code	Comment
`N10, T1, D1`	`; Load tool T1 and activate tool offset data block D1 of T1.`
`N11 G0 X... Z...`	`; The tool length compensations are applied.`
`N50, T4, D2`	`; Load tool T4 and activate tool offset data block D2 of T4.`
`...`	
`N70 G0 Z... D1`	`; Activate other cutting edge D1 for tool T4.`

Example 2: Different correction values for the left and right cutting edges of a grooving tool



① Plunge turning tool (T2)
② Cutting edge D1
③ Cutting edge D6

See also

Tool radius compensation (Page 68)

Further information

Change in the tool offset data

In the standard setting, a change in the tool offset data takes only effect the next time the T or D number is programmed.

The following machine data can be used to specify that entered tool offset data are activated immediately:

MD9440 $MM_ACTIVATE_SEL_USER

> ⚠ **WARNING**
>
> **Risk of collision**
>
> If MD9440 is set, tool offsets resulting from changes in tool offset data during the part program stop, are applied when the part program is continued.

3.5.8 Suppressing tool offsets (SUPD)

The SUPD instruction allows you to suppress the tool offsets block by block.

Tool offset deselection with D0 provides the benefit that the tool offset data are retained with SUPD. As a result, the tool offset data need not be reactivated by programming the D number.

SUPD is only used to suppress the tool length compensation, e.g. in part program blocks in which the zero point shall be included in calculation instead of the tool length. It is not recommended to use it with active G41/G42 for additional suppression of the tool radius compensations.

Syntax

```
D<No.>
X... Y... Z...
X... Y... Z... SUPD
...
D0
```

Meaning

`SUPD`:	G command for deactivating the offset data block for the active tool in the active block	
	Effectiveness:	Non-modal

Constraints

- SUPD shall only be used in linear blocks.
- SUPD cannot be used in synchronized actions.

- SUPD may not be used in conjunction with the following functions:
  - 3D tool radius compensation for 3D face milling (CUT3DFxx)
  - Curve tables (CTAB)
- The use of SUPD with active tool radius compensation (G41/G42) is possible, but not recommended.
  To activate the function nevertheless, the following setting data must be set to "0":
  SD42480 $SC_STOP_CUTCOM_STOPRE = 0
  This prevents that the program is interrupted with active G41/G42.

Example

When traversing, the tool length shall be suppressed in the subprogram SUB_SUP.

Part program

Program code	Comment
...	
N300 $P_UIFR[1]=CTRANS(X,1000,Y,400,Z,-120)	
N310 T="BALL_D3"	
N320 M6	
N330 TRAFOOF	
N340 G54 G0 Z49 D1	
N350 G0 X1100 Y500 C0 A0	
N360 **SUB_SUP**	; Calling the subroutine.
N370 G0 X1200	
N380 M30	

Subprogram with D0

Program code	Comment
N10 PROC **SUB_SUP**	
N20 DEF INT NUMBER	
N30 NUMBER=$P_TOOL	
N40 G0 Z49 **D0**	; Deselection of the tool offsets
N50 D=NUMBER	; Reselection of tool offsets.
N60 RET	

Subprogram with SUPD

Program code	Comment
N10 PROC **SUB_SUP**	
N40 G0 Z49 **SUPD**	; Suppression of tool offsets in the active block.
N60 RET	

3.5.9 Programmable tool offset (TOFFL, TOFF, TOFFR, TOFFLR):

Based on the TOFFx addresses, the user can modify the effective tool length and the effective tool radius in the NC program, without changing the tool offset data stored in the compensation memory.

These programmed offsets are deleted again at the end of the program.

Syntax

Tool length offset

```
TOFFL=<Value>
TOFFL[1]=<Value> TOFFL[2]=<Value> TOFFL[3]=<Value>
TOFF[<GeoAx>]=<Value>
```

The tool length can be changed simultaneously in all three components. However, commands of the TOFFL/TOFFL[1..3] group and commands of the TOFF[<geometry axis>] group may not be used simultaneously in one block. Similarly TOFFL and TOFFL[1] may not be written simultaneously in one block.

If all three tool length components are not programmed in one block, the components not programmed remain unchanged. In this way, it is possible to build up offsets for several components block-by-block. However, this only applies as long as the tool components have been modified only with either TOFFL or TOFF. Changing the programming version from TOFFL to TOFF or vice versa deletes any previously programmed tool length offsets (see example 3).

Tool radius offset

```
TOFFR=<Value>
```

Simultaneous tool length offset and tool radius offset

```
TOFFLR=<Value>
```

Meaning

`TOFFL`:	Correction of the effective tool length TOFFL can be programmed with or without index:	
	`TOFFL=...`	The programmed offset value is applied in the same direction as the tool length component **L1** stored in the compensation memory. The TOFFL and TOFFL[1] instructions have an identical effect.

	`TOFFL[1]=...`
	`TOFFL[2]=...`
	`TOFFL[3]=...`
		The programmed offset value is effective in the same direction as the tool length components **L1**, **L2** and **L3** stored in the offset memory.

	Note: How these tool length compensation values are calculated in the axes is determined by the tool type and the current working plane (G17/G18/G19).	

`TOFF`:	Correction of the tool length in the component parallel to the specified geometry axis TOFF is applied in the direction of the tool length component which is effective with non-rotated tool (orientable tool carrier or orientation transformation) parallel to the geometry axis specified in the index. **Note:** A frame does not influence the assignment of the programmed values to the tool length components. This means that the workpiece coordinate system (WCS) is not used to assign the tool length components to the geometry axes, but rather the tool coordinate system in the basic tool position.	
`<GeoAx>`:	Identifier of the geometry axis	
`TOFFR`:	Correction of the effective tool radius TOFFR changes the effective tool radius **with active tool radius compensation** by the programmed offset value.	
`TOFFLR`:	Correction of the effective tool length in the component L1 **and** the effective tool radius **Note:** For tools with corner rounding (types 111, 121, 131 and 156), TOFFLR also corrects the corner radius.	
`<Value>`:	Offset value	
	Type:	REAL

Examples

Example 1: Positive tool length offset

The active tool is a drill with length L1 = 100 mm.

The active plane is G17. This means that the drill points in the Z direction.

The effective drill length is to be increased by 1 mm. The following variants are available for programming this tool length offset:

- `TOFFL=1`
- `TOFFL[1]=1`
- `TOFF[Z]=1`

Example 2: Negative tool length offset

The active tool is a drill with length L1 = 100 mm.

The active plane is G18. This means that the drill points in the Y direction.

The effective drill length is to be decreased by 1 mm. The following variants are available for programming this tool length offset:

- `TOFFL=-1`
- `TOFFL[1]=-1`
- `TOFF[Y]=1`

Example 3: Change of programming version from TOFFL to TOFF

The active tool is a milling tool. The active plane is G17.

Program code	Comment
N10 TOFFL[1]=3 TOFFL[3]=5	; Effective offsets: L1=3, L2=0, L3=5
N20 TOFFL[2]=4	; Effective offsets: L1=3, L2=4, L3=5
N30 TOFF[Z]=1.3	; Effective offsets: L1=0, L2=0, L3=1.3

Example 4: Assignment of the offset values after a plane change

Program code	Comment
N10 $TC_DP1[1,1]=120	
N20 $TC_DP3[1,1]= 100	; Tool length L1=100 mm.
N30 T1 D1 G17	
N40 **TOFF[Z]=1.0**	; Offset in Z direction (corresponds to L1 for G17).
N50 G0 X0 Y0 Z0	; Machine axis position X0 Y0 Z101
N60 G18 G0 X0 Y0 Z0	; Machine axis position X0 Y100 Z1.
N70 G17	
N80 **TOFFL=1.0**	; Offset in L1 direction (corresponds to Z for G17).
N90 G0 X0 Y0 Z0	; Machine axis position X0 Y0 Z101.
N100 G18 G0 X0 Y0 Z0	; Machine axis position X0 Y101 Z0.

In this example, the offset of 1 mm in the Z axis is retained when changing to G18 in block N60; the effective tool length in the Y axis is the unchanged tool length of 100 mm.

However, in block N100, the offset is effective in the Y axis when changing to G18 as it was assigned to tool length L1 in the programming, and this length component is effective in the Y axis with G18.

Example 5: Simultaneous tool length offset and tool radius offset

a, end milling cutter **without** corner rounding (tool type 120):

Program code	Comment
...	
TOFFLR=0.1	; Effective offsets:
	; Tool length offset (L1) = 5
	; Tool radius offset = 5
...	

b, end mill **with** corner rounding (tool type 121):

Program code	Comment
...	
TOFFLR=0.1	; Effective offsets:
	; Tool length offset (L1) = 0.1
	; Tool radius offset = 0.1
	; Offset corner radius = 0.1
...	

Further information

Tool length offsets

Depending on the type of programming, programmed tool length offsets are assigned either to the tool length components L1, L2 and L3 (TOFFL) stored in the compensation memory or to the geometry axes (TOFF). The programmed offsets are treated accordingly for a plane change (G17/G18/G19 ↔ G17/G18/G19):

- If the offset values are assigned to the tool length components, the directions in which the programmed offsets apply are replaced accordingly.
- If the offset values are assigned to the geometry axes, a plane change does not affect the assignment in relation to the coordinate axes.

The following setting data is evaluated when assigning the programmed offset values to the tool length components:

SD42940 $SC_TOOL_LENGTH_CONST (change of tool length components on change of planes).

SD42950 $SC_TOOL_LENGTH_TYPE (assignment of the tool length offset independent of tool type).

If this setting data has valid values not equal to 0, then these have priority over the contents of G group 6 (plane selection G17/G18/G19) or the tool type ($TC_DP1[<T no.>, <D no.>]) contained in the tool data. This means that this setting data affects the evaluation of the offsets in the same way as the tool length components L1 to L3.

Tool radius offset

The TOFFR address has almost the same effect as the OFFN (Page 254) address. There is only a difference with active peripheral curve transformation (TRACYL) and active slot side compensation. In this case, the tool radius is affected by OFFN with a negative sign, but by TOFFR with a positive sign.

OFFN and TOFFR can be effective simultaneously. They then generally have an additive effect (except for slot side compensation).

Tool change

All offset values are retained during a tool change (cutting edge change). This means that they are also effective for the new tool (new cutting edge).

System variables for reading the current offset values

The currently effective offsets can be read with the following system variables:

System variable	Meaning
$P_TOFFL [<n>] with 0 ≤ n ≤ 3	Reads the current offset value of TOFFL (for n = 0) or TOFFL[1...3] (for n = 1, 2, 3) in the preprocessing context.
$P_TOFF [<GeoAx>]	Reads the current offset value of TOFF[<GeoAx>] in the preprocessing context.
$P_TOFFR	Reads the current offset value of TOFFR in the preprocessing context.
$P_TOFFCR	Reads the current offset value of the corner radius in the preprocessing context.

System variable	Meaning
$AC_TOFFL[<n>] with 0 ≤ n ≤ 3	Reads the current offset value of TOFFL (for n = 0) or TOFFL[1...3] (for n = 1, 2, 3) in the main run context (synchronized actions).
$AC_TOFF[<GeoAx>]	Reads the current offset value of TOFF[<GeoAx>] in the main run context (synchronized actions).
$AC_TOFFR	Reads the current offset value of TOFFR in the main run context (synchronized actions).
$AC_TOFFCR	Reads the current offset value of the corner radius in the main run context (synchronized actions).

Note

The system variables $AC_TOFFL, $AC_TOFF, AC_TOFFR and AAC_TOFFCR trigger an automatic preprocessing stop when reading from the preprocessing context (NC program).

Applications

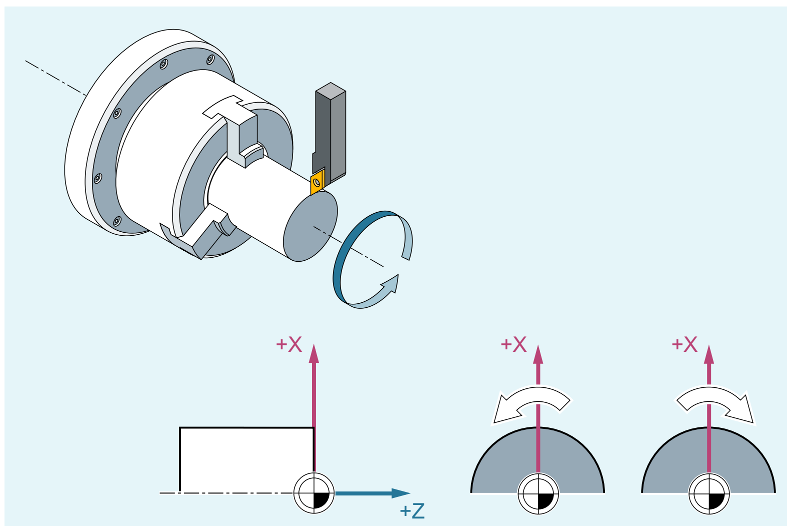
The "Programmable tool offset" function is especially interesting for ball mills and milling tools with corner radii as these are often calculated in the CAM system to the ball center instead of the ball tip. However, the tool tip is generally measured when the tool is measured, and stored as tool length in the compensation memory.

For the 3D tool radius compensation with a ball mill it is advantageous to correct the tool length and radius by the same value simultaneously. The TOFFLR address is available to the user for this purpose.

3.6 Spindle motion

3.6.1 Spindle speed (S), spindle direction of rotation (M3, M4, M5)

The spindle speed and direction of rotation values set the spindle in rotary motion and provide the conditions for chip removal.



Other spindles may be available in addition to the main spindle (e.g. the counterspindle or an actuated tool on turning machines). As a rule, the main spindle is declared the master spindle in the machine data. This assignment can be changed using an NC command.

Syntax

`S...` / `S<n>=...`

`M3` / `M<n>=3`

`M4` / `M<n>=4`

`M5` / `M<n>=5`

`SETMS(<n>)`	
`...`	
`SETMS`	

Meaning

`S...:`	Spindle speed in rpm for the master spindle
`S<n>=...:`	Spindle speed in rpm for spindle <n>
	Note: The speed specified with S0=... applies to the master spindle.

M3:	Direction of spindle rotation clockwise for master spindle
M<n>=3:	Spindle direction of rotation clockwise for spindle <n>
M4:	Direction of spindle rotation counter-clockwise for master spindle
M<n>=4:	Spindle direction of rotation counter-clockwise for spindle <n>
M5:	Spindle stop for master spindle
M<n>=5:	Spindle stop for spindle <n>
SETMS(<n>):	Set spindle <n> as master spindle
SETMS:	If SETMS is programmed without a spindle name, the configured master spindle is used instead.

Note

Up to three S-values can be programmed per NC block, e.g.:

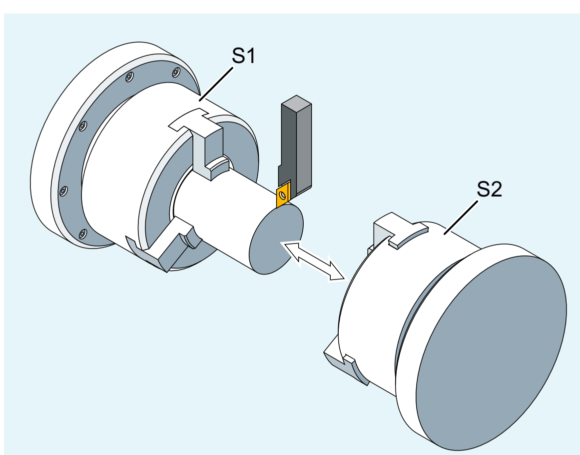
```
S... S2=... S3=...
```

Note

SETMS must be in a separate block.

Example

S1 is the master spindle, S2 is the second spindle. The part is to be machined from two sides. To do this, it is necessary to divide the operations into steps. After the cut-off point, the synchronizing device (S2) takes over machining of the workpiece after the cut off. To do this, this spindle S2 is defined as the master spindle to which G95 then applies.



Program code	Comment
N10 S300 M3	; Speed and direction of rotation for drive spindle = preset master spindle.
...	; Machining of the right-hand workpiece side.
N100 SETMS(2)	; S2 is now the master spindle.
N110 S400 G95 F…	; Speed for new master spindle.
...	; Machining of the left-hand workpiece side.

Program code	Comment
N160 SETMS	; Switching back to master spindle S1.

Further information

Interpretation of the S value for the master spindle

If function G331 or G332 is active in G group 1 (modally valid motion commands), the programmed S-value will always be interpreted as the speed in rpm. Otherwise, the interpretation of the S-value will depend upon G group 15 (feedrate type): If G96, G961 or G962 is active, the S-value is interpreted as a constant cutting rate in m/min; otherwise, it is interpreted as a speed in rpm.

Changing from G96/G961/G962 to G331/G332 sets the value of the constant cutting rate to zero; changing from G331/G332 to a function within the G group 1 other than G331/G332 sets the speed value to zero. The corresponding S-values have to be reprogrammed if required.

Preset M commands M3, M4, M5

In a block with axis commands, functions M3, M4, M5 are activated **before** the axis movements commence (basic setting on the control).

Example:

Program code	Comment
N10 G1 F500 X70 Y20 S270 M3	; The spindle ramps up to 270 rpm and the movements then executed in X and Y.
N100 G0 Z150 M5	; Spindle stop before the retraction movement in Z.

Note

Machine data can be used to set when axis movements should be executed; either once the spindle has powered up to the setpoint speed, or immediately after the programmed switching operations have been traversed.

Working with multiple spindles

Five spindles (master spindle plus four additional spindles) can be available in one channel at the same time.

One of the spindles is defined in machine data as the **master spindle**. Special functions such as thread cutting, tapping, revolutional feedrate, and dwell time apply to this spindle. For the remaining spindles (e.g. a second spindle and an actuated tool) the numbers corresponding to the speed and the direction of rotation / spindle stop must be specified.

Example:

Program code	Comment
N10 S300 M3 S2=780 M2=4	; Master spindle: 300 rpm, CW rotation 2nd spindle: 780 rpm, CCW rotation

Programmable switchover of master spindle

The SETMS(<n>) command can be used in the NC program to define any spindle as the master spindle. SETMS must be in a separate block.

Example:

Program code	Comment
N10 SETMS (2)	; Spindle 2 is now the master spindle.

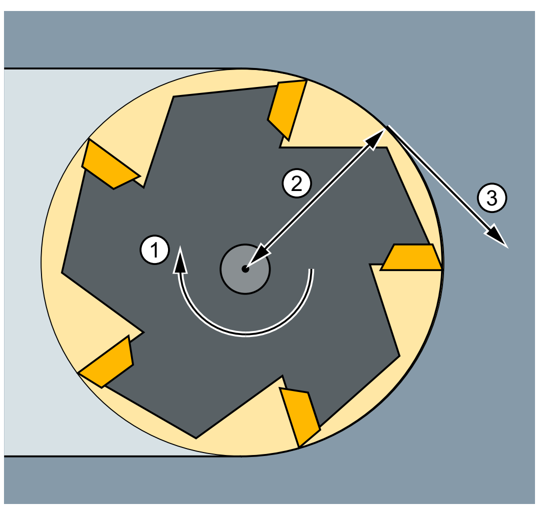
Note

The speed specified with S..., along with the functions programmed with M3, M4, M5, now apply to the newly declared master spindle.

If SETMS is programmed without a spindle name, the master spindle programmed in the machine data is used instead.

3.6.2 Tool cutting speed (SVC)

As an alternative to the spindle speed, the tool cutting speed, which is more commonly used in practice, can be programmed for milling operations.



① Spindle speed
② Tool radius
③ Tool cutting speed

The control uses the radius of the active tool to calculate the effective spindle speed from the programmed tool cutting speed:

$S = (SVC * 1000) / (R_T * 2\pi)$

with: S: Spindle speed in rpm
SVC: Tool cutting speed in m/min or ft/min
R_T: Radius of the active tool in mm

The tool type ($TC_DP1) of the active tool is not taken into account.

The programmed tool cutting speed is independent of path feedrate F and G function group 15 (feedrate type). The direction of rotation and the spindle start are implemented via M3 and M4, and the spindle stop via M5.

A change to the tool radius data in the offset memory will be applied the next time a tool offset is selected or the next time the active offset data is updated.

Changing the tool or selecting/deselecting a tool offset data set generates a recalculation of the effective spindle speed.

Requirements

Programming of the tool cutting speed requires:

- The geometric ratios of a rotating tool (milling cutter or drilling tool)
- An active tool offset data set

Syntax

```
T... D... SVC[<n>]=<Value>
...
S... M3/M4
```

Meaning

SVC:	Keyword for programming of the tool cutting speed	
[<n>]:	Number of spindle This address extension specifies which spindle the programmed cutting speed is to be applied for. In the absence of an address extension, the rate is always applied to the master spindle. **Note:** A separate cutting speed can be preset for each spindle. **Note:** Programming SVC without an address extension requires that the master spindle has the active tool. If the master spindle changes, the user will need to select a tool accordingly.	
<Value>:	Value of tool cutting speed	
	Unit:	m/min (for G71/G710) or ft/min (for G70/G700)
T... D...:	The tool radius must be established in the block with SVC. Thus, a corresponding tool including tool offset data block must either be active or selected in the block. There is no fixed sequence for SVC and T/D selection during programming in the same block.	
S... M3/M4:	Programming of the spindle speed will effect deselection of the tool cutting speed. **Note:** Switching between SVC programming and S programming is possible at any time, even while the spindle is rotating. In each case, the value that is not active is deleted.	

Note

SVC programming is not possible if the following spindle feedrate movements are active:

- Constant cutting speed: G96/G961/G962 S... (Page 108)
- Constant grinding wheel peripheral speed: SUG (Page 113)
- Position spindle: SPOS/SPOSA/M19 (Page 127)
- ; Switch master spindle over to axis mode: M70 (Page 127)

Conversely, programming one of these functions will effect a deselection of SVC (tool cutting speed).

Note

Maximum tool speed

System variable $TC_TP_MAX_VELO[<tool number>] can be used to preset a maximum tool speed (spindle speed).
If no speed limit has been defined, there will be no monitoring.

Note

The tool paths of "standard tools" generated, e.g. using CAD systems which already take the tool radius into account and only contain the deviation from the standard tool in the tool nose radius, are not supported in conjunction with SVC programming.

Examples

The following shall apply to all examples: Tool carrier = spindle (for standard milling)

Example 1: Milling cutter 6 mm radius

Program code	Comment
N10 G0 X10 T1 D1	; Selection of milling tool with, e.g. $TC_DP6[1,1] = 6 (tool radius = 6 mm)
N20 SVC=100 M3	; Cutting speed = 100 m/min
	⇒ Resulting spindle speed:
	S = (100 m/min * 1000) / (6.0 mm * 2 * 3.14) = 2653.93 rpm
N30 G1 X50 G95 FZ=0.03	; SVC and tooth feedrate
...	

Example 2: Tool selection and SVC in the same block

Program code	Comment
N10 G0 X20	
N20 T1 D1 SVC=100	; Tool and offset data set selection together with SVC in block (no specific sequence).
N30 X30 M3	; Spindle start with CW direction of rotation, cutting speed 100 m/min

Program code	Comment
N40 G1 X20 F0.3 G95	; SVC and revolutional feedrate

Example 3: Defining cutting speeds for two spindles

Program code	Comment
N10 SVC[3]=100 M6 T1 D1	
N20 SVC[5]=200	; The tool radius of the active tool offset is the same for both spindles. The effective speed is different for spindle 3 and spindle 5.

Example 4:

Assumptions:

Master or tool change is determined by the tool carrier:

MD20124 $MC_TOOL_MANAGEMENT_TOOL CARRIER > 1

In the event of a tool change the old tool offset is retained. A tool offset for the new tool is only activated when `D` is programmed:

MD20270 $MC_CUTTING_EDGE_DEFAULT = - 2

Program code	Comment
N10 $TC_MPP1[9998,1]=2	; Magazine location is tool carrier
N11 $TC_MPP5[9998,1]=1	; Magazine location is tool carrier 1
N12 $TC_MPP_SP[9998,1]=3	; Tool carrier 1 is assigned to spindle 3
N20 $TC_MPP1[9998,2]=2	; Magazine location is tool carrier
N21 $TC_MPP5[9998,2]=4	; Magazine location is tool carrier 4
N22 $TC_MPP_SP[9998,2]=6	; Tool carrier 4 is assigned to spindle 6
N30 $TC_TP2[2]="WZ2"	
N31 $TC_DP6[2,1]=5.0	; Radius = 5.0 mm of T2, offset D1
N40 $TC_TP2[8]="WZ8"	
N41 $TC_DP6[8,1]=9.0	; Radius = 9.0 mm of T8, offset D1
N42 $TC_DP6[8,4]=7.0	; Radius = 7.0 mm of T8, offset D4
...	
N100 SETMTH(1)	; Set master tool carrier number
N110 T="WZ2" M6 D1	; Tool T2 is loaded and offset D1 is activated.
N120 G1 G94 F1000 M3=3 SVC=100	; S3 = (100 m/min * 1000) / (5.0 mm * 2 * 3.14) = 3184.71 rpm
N130 SETMTH(4)	; Set master tool carrier number
N140 T="WZ8"	; Corresponds to T8="WZ8"
N150 M6	; Corresponds to M4=6
	Tool "WZ8" is in the master tool carrier, but because MD20270=-2, the old tool offset remains active.
N160 SVC=50	; S3 = (50 m/min * 1000) / (5.0 mm * 2 * 3.14) = 1592.36 rpm
	The offset applied to tool carrier 1 is still active and assigned to spindle 3.
N170 D4	; Offset D4 of the new tool "WZ8" becomes active (in tool carrier 4).

Program code	Comment
N180 SVC=300	; S6 = (300 m/min * 1000) / (7.0 mm * 2 * 3.14) = 6824.39 rpm Spindle 6 is assigned to tool carrier 4.

Example 5:

Assumptions:

Spindles are tool carriers at the same time:

MD20124 $MC_TOOL_MANAGEMENT_TOOL CARRIER = 0

In the event of a tool change, tool offset data set D4 is selected automatically:

MD20270 $MC_CUTTING_EDGE_DEFAULT = 4

Program code	Comment
N10 $TC_MPP1[9998,1]=2	; Magazine location is tool carrier
N11 $TC_MPP5[9998,1]=1	; Magazine location is tool carrier 1 = spindle 1
N20 $TC_MPP1[9998,2]=2	; Magazine location is tool carrier
N21 $TC_MPP5[9998,2]=3	; Magazine location is tool carrier 3 = spindle 3
N30 $TC_TP2[2]="WZ2"	
N31 $TC_DP6[2,1]=5.0	; Radius = 5.0 mm of T2, offset D1
N40 $TC_TP2[8]="WZ8"	
N41 $TC_DP6[8,1]=9.0	; Radius = 9.0 mm of T8, offset D1
N42 $TC_DP6[8,4]=7.0	; Radius = 7.0 mm of T8, offset D4
...	
N100 SETMS(1)	; Spindle 1 = master spindle
N110 T="WZ2" M6 D1	; Tool T2 is loaded and offset D1 is activated.
N120 G1 G94 F1000 M3 SVC=100	; S1 = (100 m/min * 1000) / (5.0 mm * 2 * 3.14) = 3184.71 rpm
N200 SETMS(3)	; Spindle 3 = master spindle
N210 M4 SVC=150	; S3 = (150 m/min * 1000) / (5.0 mm * 2 * 3.14) = 4777.07 rpm Refers to tool offset D1 of T="WZ2", S1 continues to turn at previous speed.
N220 T="WZ8"	; Corresponds to T8="WZ8"
N230 M4 SVC=200	; S3 = (200 m/min * 1000) / (5.0 mm * 2 * 3.14) = 6369.43 rpm Refers to tool offset D1 of T="WZ2".
N240 M6	; Corresponds to M3=6 Tool "WZ8" is in the master spindle, tool offset D4 of the new tool becomes active.
N250 SVC=50	; S3 = (50 m/min * 1000) / (7.0 mm * 2 * 3.14) = 1137.40 rpm Offset D4 on master spindle is active.
N260 D1	; Offset D1 of new tool "WZ8" active.
N270 SVC[1]=300	; S1 = (300 m/min * 1000) / (9.0 mm * 2 * 3.14) = 5307.86 rpm S3 = (50 m/min * 1000) / (9.0 mm * 2 * 3.14) = 884.64 rpm
...	

Further information

Tool radius

The following tool offset data (associated with the active tool) affect the tool radius when:

- $TC_DP6 (radius - geometry)
- $TC_DP15 (radius - wear)
- $TC_SCPx6 (offset for $TC_DP6)
- $TC_ECPx6 (offset for $TC_DP6)

The following are not taken into account:

- Online radius compensation
- Allowance on the programmed contour (OFFN)

Tool radius compensation (G41/G42)

Although tool radius compensation (G41/G42) and SVC both relate to the tool radius, they are not functionally linked and are independent of one another.

Tapping without compensating chuck (G331, G332)

SVC programming is also possible in conjunction with G331 or G332.

Synchronized actions

SVC cannot be programmed from synchronized actions.

Reading the cutting speed and the spindle speed programming variant

The cutting speed of a spindle and the speed programming variant (spindle speed S or tool cutting speed SVC) can be read using system variables:

- With preprocessing stop in the part program via system variables:

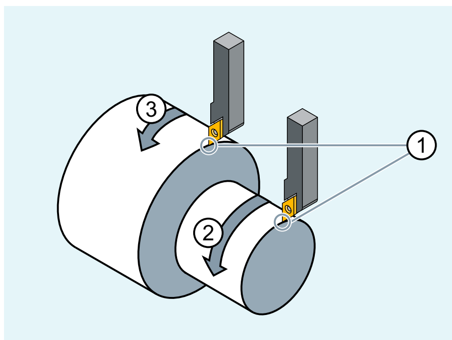
$AC_SVC[<n>]	Effective cutting speed when the current main run block for spindle with number <n> was preprocessed.
$AC_S_TYPE[<n>]	Effective spindle speed programming variant when the current main run block for spindle with number <n> was preprocessed. **Value:** **Meaning:** 1 Spindle speed S in rpm 2 Tool cutting speed SVC in m/min or ft/min

- Without preprocessing stop in the part program via system variables:

$P_SVC[<n>]	Programmed cutting speed for spindle <n>
$P_S_TYPE[<n>]	Programmed spindle speed programming variant for spindle <n> **Value:** **Meaning:** 1 Spindle speed S in rpm 2 Tool cutting speed SVC in m/min or ft/min

3.6.3 Constant cutting rate (G96/G961/G962, G97/G971/G972, G973, LIMS, SCC)

If the "Constant cutting speed" function is active, the spindle speed is modified as a function of the respective workpiece diameter so that the cutting speed S in m/min or ft/min remains constant at the tool edge.



①	Constant cutting rate
②	Increased spindle speed
③	Reduced spindle speed

This results in the following advantages:

- Uniformity and consequently improved surface quality of turned parts
- Machining with less wear on tools

Syntax

Activating/deactivating constant cutting speed for the master spindle:

```
G96/G961/G962 S...
...
G97/G971/G972/G973
```

Speed limitation for the master spindle:

```
LIMS=<value>
LIMS[<spindle>]=<value>
```

Other reference axis for G96/G961/G962:

```
SCC[<axis>]
```

Note

SCC[<Axis>] can be programmed together with G96/G961/G962 or separately.

Meaning

`G96`:	Revolutional feedrate (as for G95 (Page 115)) and constant cutting speed G95 is activated automatically with G96. If G95 has not been activated previously, a new feedrate value F... has to be specified when G96 is called.	
`G961`:	Linear feedrate (as for G94 (Page 115)) and constant cutting speed	
`G962`:	Linear feedrate or revolutional feedrate and constant cutting speed	
`S...`:	In conjunction with G96, G961 or G962, S... is not interpreted as a spindle speed but rather as a cutting speed. The cutting speed is always applied to the master spindle.	
	Unit:	m/min (for G71/G710) or ft/min (for G70/G700)
	Range of values:	0.1 m/min to 9999 9999.9 m/min
`G97`:	Revolutional feedrate and constant spindle speed (constant cutting speed OFF)	
`G971`:	Linear feedrate and constant spindle speed (constant cutting speed OFF)	
`G972`:	Linear feedrate or revolutional feedrate and constant spindle speed (constant cutting speed OFF)	
`G973`:	Revolutional feedrate without spindle speed limitation and constant spindle speed (G97 without LIMS for ISO mode)	
	Note: After G97 (or G971 ... G973), S... is again interpreted as a spindle speed in rpm. In the absence of a new spindle speed being specified, the last speed set with G96 (respectively G961 or G962) is retained.	
`LIMS`:	Speed limitation for the master spindle (only applied if G96/G961/G97 active) On machines with selectable master spindles, limitations of differing values can be programmed for up to four spindles within one block.	
	`<spindle>`:	Number of spindle
	`<value>`:	Spindle speed upper limit in rpm
`SCC`:	If any of the G96/G961/G962 functions are active, SCC[<axis>] can be used to assign any geometry axis as a reference axis.	

Note

If G96/G961/G962 is selected for the first time, a constant cutting speed S... must be entered; if G96/G961/G962 is selected again, the entry is optional.

Note

The speed limitation programmed with LIMS must not exceed the speed limit programmed with G26 or defined in the setting data.

Note

The reference axis for G96/G961/G962 must be a geometry axis assigned to the channel at the time when SCC[<axis>] is programmed. SCC[<axis>] can also be programmed when any of the G96/G961/G962 functions are active.

Examples

Example 1: Activating the constant cutting speed with speed limitation

Program code	Comment
N10 SETMS (3)	
N20 G96 S100 LIMS=2500	; Constant cutting speed = 100 m/min, max. speed = 2500 rpm
...	
N60 G96 G90 X0 Z10 F8 S100 LIMS=444	; Max. speed = 444 rpm

Example 2: Defining speed limitation for four spindles

Speed limitations are defined for spindle 1 (master spindle) and spindles 2, 3, and 4:

Program code
N10 LIMS=300 LIMS[2]=450 LIMS[3]=800 LIMS[4]=1500
...

Example 3: Y-axis assignment for face cutting with X axis

Program code	Comment
N10 G18 LIMS=3000 T1 D1	; Speed limitation at 3000 rpm
N20 G0 X100 Z200	
N30 Z100	
N40 G96 S20 M3	; Constant cutting speed = 20 m/min, is dependent upon X axis.
N50 G0 X80	
N60 G1 F1.2 X34	; Face cutting in X at 1.2 mm/revolution.
N70 G0 G94 X100	
N80 Z80	
N100 T2 D1	
N110 G96 S40 SCC[Y]	; Y axis is assigned to G96 and G96 is activated (can be achieved in a single block). Constant cutting speed = 40 m/min, is dependent upon X axis.
...	
N140 Y30	
N150 G01 F1.2 Y=27	; Plunge-cutting in Y, feedrate F = 1.2 mm/revolution.
N160 G97	; Constant cutting speed off.
N170 G0 Y100	

Further information

Calculation of the spindle speed

The SZS position of the face axis (radius) is the basis for calculating the spindle speed from the programmed cutting rate.

Note

Frames between WCS and SZS (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 (for example, if there is a change in the effective diameter in the case of SCALE).

Speed limitation LIMS

If a workpiece that varies greatly in diameter needs to be machined, it is advisable to specify a speed limit for the spindle with LIMS (maximum spindle speed). This prevents excessively high speeds with small diameters. LIMS is only applied if G96, G961 and G97 are active. LIMS is not applied if G971 is selected. On loading the block into the main run, all programmed values are transferred into the setting data.

Note

The speed limits changed with 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 LIMS are no longer to apply after the end of program, the following definition must be inserted in the GUD block of the machine manufacturer:

REDEF $SA_SPIND_MAX_VELO_LIMS PRLOC

Deactivating the constant cutting rate (G97/G971/G972/G973)

After G97 (or G971 ... G973), S... is again interpreted as a spindle speed in rpm. In the absence of a new spindle speed being specified, the last speed set with G96 (respectively G961 or G962) is retained.

The G96/G961 function can also be deactivated with G94 or G95. In this case, the last speed programmed S... is used for subsequent machining operations.

G97 can be programmed without G96 beforehand. The function then has the same effect as G95; LIMS can also be programmed.

Using G973, the constant cutting rate can be deactivated without activating a spindle speed limitation.

Note

The transverse axis must be defined in machine data.

Rapid traverse G0

With rapid traverse G0, there is no change in speed.

Exception:

If the contour is approached in rapid traverse and the next NC block contains a G1/G2/G3/... path command, the speed is adjusted in the G0 approach block for the next path command.

Other reference axis for G96/G961/G962

If any of the G96/G961/G962 functions are active, SCC[<axis>] can be used to assign any geometry axis as a reference axis. If the reference axis changes, which will in turn affect the TCP (tool center point) reference position for the constant cutting rate, the resulting spindle speed will be reached via the set braking or acceleration ramp.

Axis exchange of the assigned channel axis

The reference axis property for G96/G961/G962 is always assigned to a geometry axis. In the event of an axis exchange involving the assigned channel axis, the reference axis property for G96/G961/G962 is retained in the old channel.

A geometry axis exchange will not affect how the geometry axis is assigned to the constant cutting rate. If the TCP reference position for G96/G961/G962 is affected by a geometry axis exchange, the spindle will reach the new speed via a ramp.

If no new channel axis is assigned as a result of a geometry axis exchange (e.g. GEOAX(0,X)), the spindle speed will be frozen in accordance with G97.

Examples for geometry axis exchange with assignments of the reference axis:

Program code	Comment
N05 G95 F0.1	
N10 GEOAX(1, X1)	; Channel axis X1 becomes the first geometry axis.
N20 SCC[X]	; First geometry axis (X) becomes the reference axis
	; for G96/G961/G962.
N30 GEOAX(1, X2)	; Channel axis X2 becomes the first geometry axis.
N40 G96 M3 S20	; Reference axis for G96 is channel axis X2.

Program code	Comment
N05 G95 F0.1	
N10 GEOAX(1, X1)	; Channel axis X1 becomes the first geometry axis.
N20 SCC[X1]	; X1 and implicitly the first geometry axis (X) becomes
	the reference axis for G96/G961/G962.
N30 GEOAX(1, X2)	; Channel axis X2 becomes the first geometry axis.
N40 G96 M3 S20	; Reference axis for G96 is X2 or X, no alarm.

Program code	Comment
N05 G95 F0.1	
N10 GEOAX(1, X2)	; Channel axis X2 becomes the first geometry axis.
N20 SCC[X1]	; X1 is not a geometry axis, alarm.

Program code	Comment
N05 G0 Z50	
N10 X35 Y30	
N15 SCC[X]	; Reference axis for G96/G961/G962 is X.
N20 G96 M3 S20	; Constant cutting rate ON at 10 mm/min.
N25 G1 F1.5 X20	; Face cutting in X at 1.5 mm/revolution.

Program code	Comment
N30 G0 Z51	
N35 SCC[Y]	; Reference axis for G96 is Y, reduction of spindle speed (Y30).
N40 G1 F1.2 Y25	; Face cutting in Y at 1.2 mm/revolution.

3.6.4 Switching constant grinding wheel peripheral speed (GWPSON, GWPSOF) on/off:

With the predefined procedures GWPSON(...) and GWPSOF(...), the constant grinding wheel peripheral speed (GWPS) for grinding tools (tool type: 400 to 499) is switched on and off.

Syntax

```
GWPSON(<TNo>)
S<n>=... :
...
GWPSOF(<TNo>)
```

Meaning

GWPSON(...):	Switch on the constant grinding wheel peripheral speed
GWPSOF(...):	Switch off the constant grinding wheel peripheral speed
<TNo>:	T number **Note:** Only required if the constant grinding wheel peripheral speed is to be switched on or off for an inactive grinding wheel rather than the active grinding wheel that is currently in use.
S<n>=...:	Grinding wheel peripheral speed in m/s or ft/s for spindle <n>
S0=... or S...:	Grinding wheel peripheral speed for the master spindle

Query status

The following system variable can be used to query from the part program whether the constant grinding wheel peripheral speed is active for a specific spindle.

$P_GWPS[<n>] ; where <n> = spindle number

Value	Meaning
0 (= FALSE)	GWPS is **inactive**.
1 (= TRUE)	GWPS is **active**.

3.6.5 Programmable spindle speed limitation (G25, G26)

The minimum and maximum spindle speeds defined in the machine and setting data can be modified by means of a part program command.

Programmed spindle speed limitations are possible for all spindles of the channel.

Syntax

```
G25 S… S1=… S2=…
G26 S… S1=… S2=…
```

Meaning

`G25`:	**Lower** spindle speed limit
`G26`:	**Upper** spindle speed limit
`S... S1=… S2=…` :	Minimum or maximum spindle speed(s) **Note:** A maximum of three spindle speed limits can be programmed for each block. Range of values: 0.1 to 9999 9999.9 rpm

---

Note

A spindle speed limitation programmed with `G25` or `G26` overwrites the speed limits in the setting data and, therefore, remains stored even after the end of the program.

However, if the speed limits changed with `G25`/`G26` 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

Example

Program code	Comment
`N10 G26 S1400 S2=350 S3=600`	`; Upper speed limit for master spindle, spindle 2 and spindle 3.`

3.7 Feed control

3.7.1 Feedrate (G93, G94, G95, F, FGROUP, FL, FGREF)

These commands are used in the NC program to set the feedrates for all axes involved in the machining sequence.

Syntax

```
G93
G94
G95
F<value>
FGROUP(<axis_1>,<axis_2>,...)
FGREF[<rotary axis>]=<reference radius>
FL[<axis>]=<value>
```

Meaning

G93:	Path feed type: Inverse-time feedrate [rpm]
G94:	Path feed type: Linear feedrate [mm/min], [inch/min] or [degrees/min]
G95:	Path feed type: Revolutional feedrate [mm/revolution] or [inch/revolution] The revolutional feedrate can be derived from a master spindle, any other spindle or a rotary axis.
F<value>	Path feedrate for all or the path axes selected with FGROUP.
FGROUP:	Definition of the path axes to which the F-programmed path feed refers
FGREF:	FGREF is used to program the effective radius (<reference radius>) for each of the rotary axes specified under FGROUP
FL:	Limit velocity for synchronized/path axes The unit set with G94 applies One FL value can be programmed per axis (channel axes, geometry axis or orientation axis)
<axis>:	Name of a channel axis, type: AXIS

Examples

Example 1: Mode of operation of FGROUP

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 = 100mm/min or 100 degrees/min
N130 DO $R1=$AC_TIME	

Program code	Comment
N140 X10	; Feedrate = 100 mm/min, path = 10 mm, R1 = approx. 6 s
N150 DO $R2=$AC_TIME	
N160 X10 A10	; Feedrate = 100 mm/min, path = 14.14 mm, R2 = approx. 8 s
N170 DO $R3=$AC_TIME	
N180 A10	; Feedrate = 100 degrees/min, path = 10 degrees, R3 = approx. 6 s
N190 DO $R4=$AC_TIME	
N200 X0.001 A10	; Feedrate = 100 mm/min, path = 10 mm, R4 = approx. 6 s
N210 G700 F100	; Feedrate = 2540 mm/min or 100 degrees/min
N220 DO $R5=$AC_TIME	
N230 X10	; Feedrate = 2540 mm/min, path = 254 mm, R5 = approx. 6 s
N240 DO $R6=$AC_TIME	
N250 X10 A10	; Feedrate = 2540 mm/min, path = 254.2 mm, R6 = approx. 6 s
N260 DO $R7=$AC_TIME	
N270 A10	; Feedrate = 100 degrees/min, path = 10 degrees, R7 = approx. 6 s
N280 DO $R8=$AC_TIME	
N290 X0.001 A10	; Feedrate = 2540 mm/min, path = 10 mm, R8 = approx. 0.288 s
N300 FGREF[A]=360/(2*$PI)	; Set 1 degree = 1 inch via the effective radius
N310 DO $R9=$AC_TIME	
N320 X0.001 A10	; Feedrate = 2540 mm/min, path = 254 mm, R9 = approx. 6 s
N330 M30	

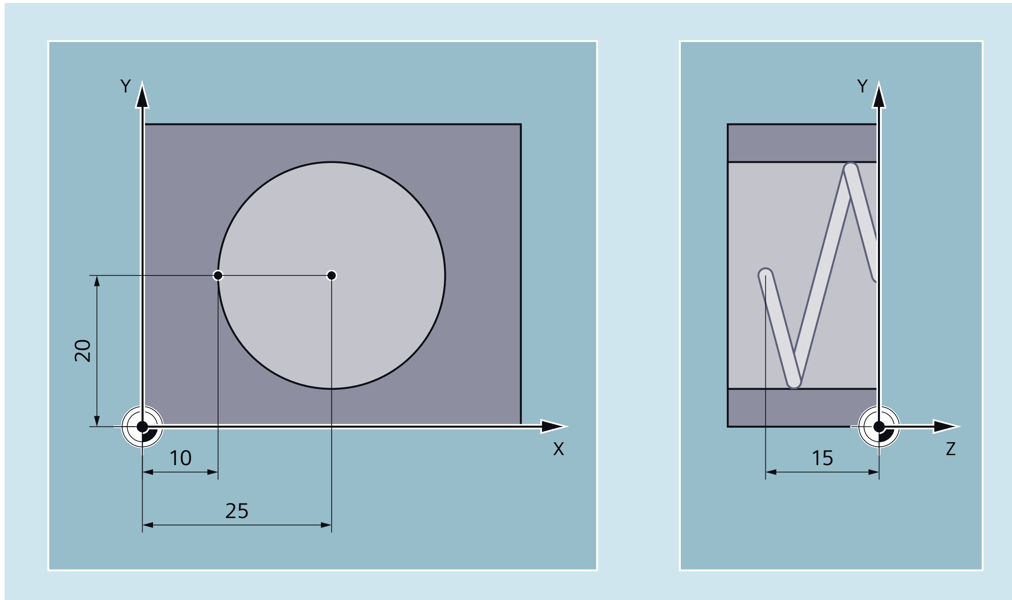
Example 2: Traverse synchronized axes with limit speed FL

The path velocity of the path axes is reduced if the synchronized axis Z reaches the limit velocity.

Program code
N10 G0 X0 Y0
N20 FGROUP(X)
N30 G1 X1000 Y1000 G94 F1000 FL[Y]=500
N40 Z-50

Example 3: Helical interpolation

Path axes X and Y traverse with the programmed feedrate, the infeed axis Z is a synchronized axis.

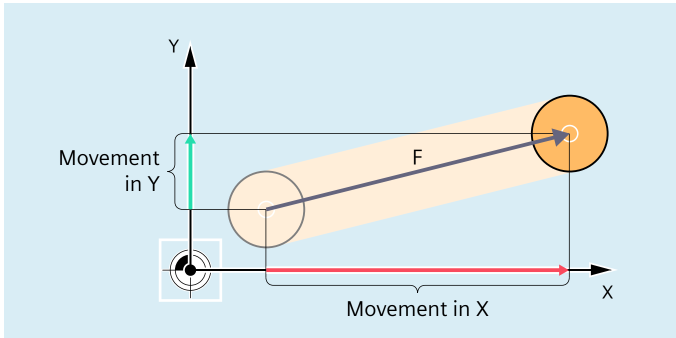


Program code	Comment
N10 G17 G94 G1 Z0 F500	; Feed of the tool.
N20 X10 Y20	; Approach the starting position.
N25 FGROUP(X,Y)	; Axes X/Y are path axes, Z is a synchronized axis.
N30 G2 X10 Y20 Z-15 I15 J0 F1000 FL[Z]=200	; On the circular path, the feedrate is 1,000 mm/min, traversing in the Z direction is synchronized.
...	
N100 FL[Z]=$MA_AX_VELO_LIMIT[0,Z]	; The limit speed is deselected by reading the speed from the MD. Read the value from the MD.
N110 M30	; End of program

Further information

Feedrate for path axes (F)

The path feedrate is generally composed of the individual speed components of all geometry axes participating in the movement and refers to the center point of the cutter or the tip of the turning tool.



The feedrate is specified under address `F`. Depending on the default setting in the machine data, the units of measurement specified with the G commands are either in mm or inch.

One `F` value can be programmed per NC block. The feedrate unit is defined using one of the G commands `G93`/`G94`/`G95`. The feedrate `F` acts only on path axes and remains active until a new feedrate is programmed. Separators are permitted after the address `F`.

Examples:

`F100` or `F 100`

`F.5`

`F=2*FEED`

Feedrate type (G93/G94/G95)

The G commands `G93`, `G94` and `G95` are modal. In the event of switching between `G93`, `G94` and `G95`, the path feedrate value has to be reprogrammed. When machining with rotary axes, the feedrate can also be specified in degrees/min.

Inverse-time feedrate (G93)

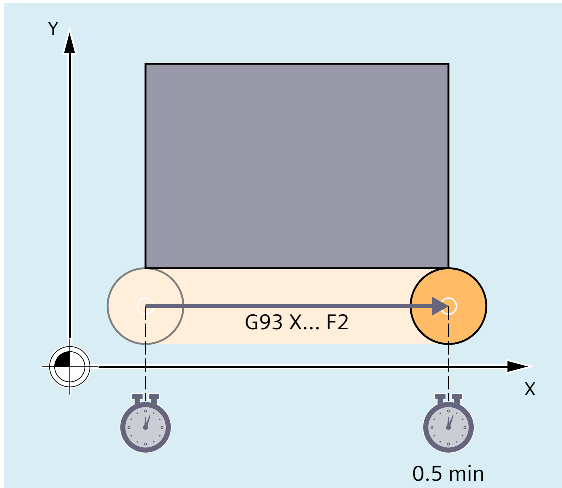
The inverse-time feedrate specifies the time required to execute the motion commands in a block.

Unit: rpm

Example:

`N10 G93 G01 X100 F2`

Means: The programmed path is traversed in 0.5 min.



Note

If the path lengths vary greatly from block to block, a new `F` value should be specified in each block with `G93`. When machining with rotary axes, the feedrate can also be specified in degrees/min.

Feedrate for synchronized axes

The feedrate programmed under address `F` applies to all the path axes programmed in a block but not to the synchronized axes. The synchronized axes are controlled such that they require the same time for their path as the path axes, and all axes reach their end point at the same time.

Limit velocity for synchronized axes (FL)

The `FL` command can be used to program a limit velocity for synchronized axes. In the absence of a programmed `FL`, the rapid traverse velocity applies. `FL` is deselected by assignment to MD (MD36200 $MA_AX_VELO_LIMIT).

Traverse path axis as synchronized axis (FGROUP)

`FGROUP` is used to define whether a path axis should be traversed with path feedrate or as a synchronized axis. In helical interpolation, for example, it is possible to define that only two geometry axes, X and Y, are to be traversed at the programmed feedrate. The infeed axis Z is the synchronized axis in this case.

Example: `FGROUP(X,Y)`

Change FGROUP

The setting made with FGROUP can be changed:

1. By reprogramming FGROUP: e.g. FGROUP(X,Y,Z)
2. By programming FGROUP without a specific axis: FGROUP()
   In accordance with FGROUP(), the initial setting in the machine data applies: Geometry axes are now once again traversed in the path axis grouping.

Note

With FGROUP, axis identifiers must be the names of channel axes.

Units of measurement for feedrate F

In addition to the geometrical settings G700 and G710, the G commands are also used to define the measuring system for the feedrates F. In other words:

- For G700: [inch/min]
- For G710: [mm/min]

Note

G70/G71 have **no** effect on feedrate settings.

Unit of measurement for synchronized axes with limit speed FL

The unit set for F using G command G700/G710 is also valid for FL.

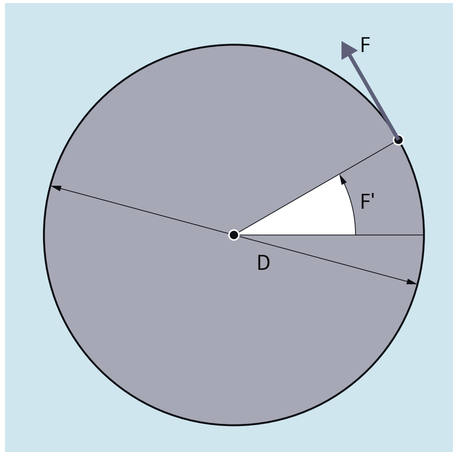
Unit for rotary and linear axes

For linear and rotary axes which are combined with FGROUP and traverse a path together, the feedrate is interpreted in the unit of the linear axes (depending on the default with G94/G95, in mm/min or inch/min and mm/rev or inch/rev).

The tangential velocity of the rotary axis in mm/min or inch/min is calculated according to the following formula:

F[mm/min] = F'[degrees/min] * π * D[mm]/360[degrees]

where: F: Tangential velocity
F': Angular velocity
π: Circle constant
D: Diameter



Traverse rotary axes with path velocity F (FGREF)

For machining operations in which the tool or the workpiece or both are moved by a rotary axis, the effective machining feedrate is to be interpreted as a path feed in the usual way by reference to the F value. This requires the specification of an effective radius (reference radius) for each of the rotary axes involved.

The unit of the reference radius depends on the `G70`/`G71`/`G700`/`G710` setting.

All axes involved must be included in the `FGROUP` command to be taken into account in the calculation of the path feedrate.

In order to ensure compatibility with the behavior with no `FGREF` programming, the factor 1 degree = 1 mm is activated on system power up and RESET. This corresponds to a reference radius of FGREF= 360 mm/(2π) = 57.296 mm.

Note

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 situations:

Program code
`N100 FGROUP(X,Y,Z,A)`
`N110 G1 G91 A10 F100`
`N120 G1 G91 A10 X0.0001 F100`

With this type of programming, the F value programmed in N110 is evaluated as the rotary axis feedrate in degrees/min, while the feedrate evaluation in N120 is either 100 inch/min or 100 mm/min, dependent upon the currently active G70/G71/G700/G710 setting.

NOTICE

Feedrate difference

FGREF evaluation 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

Read reference radius

The value of the reference radius of a rotary axis can be read using system variables:

- In synchronized actions or with preprocessing stop in the part program via system variable:

$AA_FGREF[<axis>]	Current main run value

- Without preprocessing stop in the part program via 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:

- In synchronized actions or with preprocessing stop in the part program via 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.

Path reference factors for orientation axes with FGREF

With orientation axes the mode of operation of the `FGREF[]` factors is dependent upon whether the change in the orientation of the tool is implemented by means of rotary axis or vector interpolation.

In the case of **rotary axis interpolation**, as is the case with rotary axes, the relevant `FGREF` factors of the orientation axes are calculated individually as reference radius for the axis paths.

In the case of **vector interpolation**, an effective `FGREF` factor, which is calculated as the geometric mean value of the individual `FGREF` factors, is applied.

FGREF[effective] = nth root of [(FGREF[A] * FGREF[B]...)]

where:	A:	Axis identifier of 1st orientation axis
	B:	Axis identifier of 2nd orientation axis
	C:	Axis identifier of 3rd orientation axis
	n:	Number of orientation axes

Example:

Since there are two orientation axes for a standard 5-axis transformation, the effective factor is, therefore, the root of the product of the two axial factors:

FGREF[effective] = square root of [(FGREF[A] * FGREF[B])]

Note

It is, therefore, possible to use the effective factor for orientation axes `FGREF` to define a reference point on the tool to which the programmed path feedrate refers.

3.7.2 Traverse positioning axes (POS, POSA, POSP, FA, WAITP, WAITMC)

Positioning axes are traversed independently of the path axes at a separate, axis-specific feedrate. There are no interpolation commands. With the POS/POSA/POSP commands, the positioning axes are traversed and the sequence of motions coordinated at the same time.

The following are typical examples of positioning axes:

- Pallet feed equipment
- Gauging stations

WAITP can be used to identify a position in the NC program where the program is to wait until an axis programmed with POSA in a previous NC block reaches its end position.

WAITMC loads the next NC block immediately when the specified wait marker is received.

Syntax

```
POS[<axis>]=<position>

POSA[<axis>]=<position>

POSP[<axis>]=(<end position>,<partial length>,<mode>)
```

```
FA[<axis>]=<value>
WAITP(<axis>) ; Programming in a separate NC block.
WAITMC(<wait marker>)
```

Meaning

`POS`/`POSA`:	Move positioning axis to specified position `POS` and `POSA` have the same functionality but differ in their block change behavior: • `POS` delays the enabling of the NC block until the position has been reached. • `POSA` enables the NC block even if the position has not been reached.			
	`<axis>`:	Name of the axis to be traversed (channel or geometry axis identifier)		
	`<position>`:	Axis position to be approached		
		Type:	REAL	
`POSP`:	Move positioning axis to specified end position in sections			
	`<end position>`:	Axis end position to be approached		
	`<partial length>`:	Length of a section		
	`<mode>`:	Approach mode		
		= `0`:	For the last two sections, the path remaining until the end position is split into two residual sections of equal size (preset).	
		= `1`:	The partial length is adjusted so that the total of all calculated partial lengths corresponds exactly to the path up to the end position.	
	Note: `POSP` is used specifically to program oscillating motion.			
`FA`:	Feedrate for the specified positioning axis			
	`<axis>`:	Name of the axis to be traversed (channel or geometry axis identifier)		
	`<Value>`:	Feedrate		
		Unit:	mm/min or inch/min or degrees/min	
	Note: Up to 5 `FA` values can be programmed for each NC block.			
`WAITP`:	Wait for a positioning axis to be traversed The subsequent blocks are not processed until the specified positioning axis programmed in a previous NC block with `POSA` has reached its end position (with exact stop fine).			
	`<axis>`:	Name of the axis (channel or geometry axis identifier) for which the `WAITP` command is to be applied		
	Note: With `WAITP`, an axis can be made available as an oscillating axis or for traversing as a concurrent positioning axis (via PLC).			
`WAITMC`:	Wait for the specified wait marker to be received When the wait marker is received, the next NC block is loaded immediately.			
	`<wait marker>`:	Number of the wait marker		

⚠ CAUTION

Travel with POSA

If a command, which implicitly causes a preprocessing stop, is read in a following block, this block is not executed until all other blocks which are already preprocessed and stored have been executed. The previous block is stopped in exact stop (as G9).

Examples

Example 1: Travel with POSA and access to machine status data

The control generates an internal preprocessing stop on access to machine status data ($A...). Machining is stopped until all preprocessed and saved blocks have been executed in full.

Program code	Comment
N40 POSA[X]=100	
N50 IF $AA_IM[X]==R100 GOTOF LABEL1	; Access to machine status data.
N60 G0 Y100	
N70 WAITP(X)	
N80 LABEL1:	
N...	

Example 2: Wait for end of travel with WAITP

Pallet feed equipment

Axis U:	Pallet store
	Transport of workpiece pallet to working area
Axis V:	Transfer line to a gauging station where spot checks are carried out to assist the process

Program code	Comment
N10 FA[U]=100 FA[V]=100	; Axis-specific feedrate specifications for the individual positioning axes U and V.
N20 POSA[V]=90 POSA[U]=100 G0 X50 Y70	; Traverse positioning and path axes.
N50 WAITP(U)	; Program execution does not resume until axis U reaches the end point programmed in N20.
…	

Further information

Travel with POSA

Block step enable or program execution is not affected by `POSA`. The movement to the end position can be performed during execution of subsequent NC blocks.

Travel with POS

The next block is not executed until all axes programmed under `POS` reach their end positions.

Wait for end of travel with WAITP

After a `WAITP`, assignment of the axis to the NC program is no longer valid; this applies until the axis is programmed again. This axis can then be operated as a positioning axis through the PLC, or as a reciprocating axis from the NC program/PLC or HMI.

Block change in the braking ramp with IPOBRKA and WAITMC

An axis is only decelerated if the wait marker has not yet been reached or if another end-of-block criterion is preventing the block change. After a `WAITMC`, the axis starts immediately if no other end-of-block criterion is preventing the block change.

3.7.3 Position-controlled spindle mode (SPCON, SPCOF)

The position-controlled operation of the spindle is explicitly activated or deactivated using the SPCON or SPCOF command.

Note

The switching on of the position control mode with SPCON requires a maximum of three position control cycles.

Syntax

```
SPCON
SPCON(<n>)
SPCON(<n>,<m>,...)
SPCOF
SPCOF(<n>)
SPCOF(<n>,<m>,...)
```

Meaning

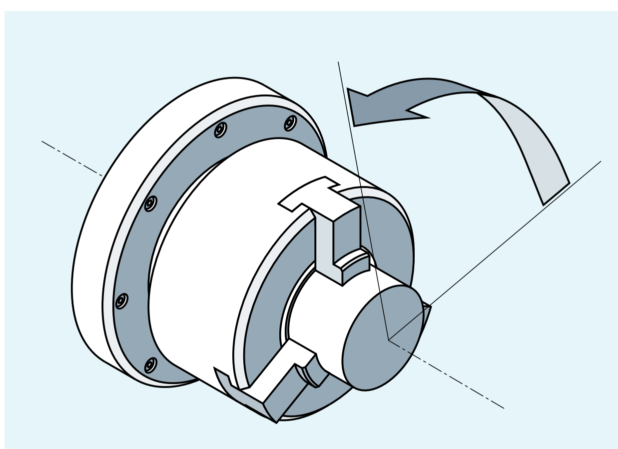
SPCON	Activate position-controlled mode The specified spindle is switched over from speed control to position control. SPCON s modal and is retained until SPCOF.
SPCOF	Deactivate position-controlled mode The specified spindle is switched over from position control to speed control.
<n>,<m>,...:	Spindle numbers Without specifying a spindle number: Master spindle of the channel

Note

For a synchronous spindle with setpoint coupling, it is not permissible that the leading spindle is switched into the speed-controlled mode using SPCOF.

3.7.4 Positioning spindles (SPOS, SPOSA, M19, M70, WAITS)

SPOS, SPOSA or M19 can be used to set spindles to specific angular positions, e.g. during tool change.



SPOS, SPOSA and M19 induce a temporary switchover to position-controlled mode until the next M3/M4/M5/M41 to M45.

Positioning in axis mode

The spindle can also be operated as a path axis, synchronized axis or positioning axis at the address defined in the machine data. When the axis identifier is specified, the spindle is in axis mode. M70 switches the spindle directly to axis mode.

End of positioning

The end-of-motion criterion when positioning the spindle can be programmed using FINEA, CORSEA, IPOENDA or IPOBRKA.

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.

Synchronization

In order to synchronize spindle movements, WAITS can be used to wait until the spindle position is reached.

Requirements

The spindle to be positioned must be capable of operation in position-controlled mode.

Syntax

Position spindle:

`SPOS=<value>` / `SPOS[<n>]=<value>`

`SPOSA=<value>` / `SPOSA[<n>]=<value>`

`M19` / `M<n>=19`

Switch spindle over to axis mode:

`M70` / `M<n>=70`

Define end-of-motion criterion:

`FINEA` / `FINEA[S<n>]`

`COARSEA` / `COARSEA[S<n>]`

`IPOENDA` / `IPOENDA[S<n>]`

`IPOBRKA` / `IPOBRKA(<axis>[,<instant in time>])` ; Programming in a separate NC block.

Synchronize spindle movements:

`WAITS` / `WAITS(<n>,<m>)` ; Programming in a separate NC block.

Meaning

<table>
<tr><td rowspan="14"><code>SPOS</code> / <code>SPOSA</code>:</td><td colspan="4">Set spindle to specified angle
SPOS and SPOSA have the same functionality but differ in their block change behavior:
• With SPOS, the NC block is only enabled once the position has been reached.
• With SPOSA, the block is enabled even if the position has not been reached.</td></tr>
<tr><td><code><n></code>:</td><td colspan="3">Number of the spindle to be positioned.
If a spindle number is not specified or if the spindle number is set to "0", SPOS or SPOSA will be applied to the master spindle.</td></tr>
<tr><td rowspan="12"><code><value></code>:</td><td colspan="3">Angular position to which the spindle is to be set.</td></tr>
<tr><td>Unit:</td><td colspan="2">Degrees</td></tr>
<tr><td>Type:</td><td colspan="2">REAL</td></tr>
<tr><td colspan="3">The following options are available for programming the position approach mode:</td></tr>
<tr><td><code>=AC(<value>)</code>:</td><td colspan="2">Absolute dimensions</td></tr>
<tr><td></td><td>Range of values:</td><td>0 ... 359.9999</td></tr>
<tr><td><code>=IC(<value>)</code>:</td><td colspan="2">Incremental dimensions</td></tr>
<tr><td></td><td>Range of values:</td><td>0 ... ±99 999.999</td></tr>
<tr><td><code>=DC(<value>)</code>:</td><td colspan="2">Approach absolute value directly</td></tr>
<tr><td><code>=ACN(<value>)</code>:</td><td colspan="2">Absolute dimension, approach in negative direction</td></tr>
<tr><td><code>=ACP(<value>)</code>:</td><td colspan="2">Absolute dimension, approach in positive direction</td></tr>
<tr><td><code>=<value></code>:</td><td colspan="2">as DC(<value>)</td></tr>
<tr><td><code>M<n>=19</code>:</td><td colspan="4">Set the master spindle (M19 or M0=19) or spindle with the number <n> (M<n>=19) to the angular position preset with SD43240 $SA_M19_SPOS with the position approach mode preset in SD43250 $SA_M19_SPOSMODE.
The NC block is not enabled until the position has been reached.</td></tr>
<tr><td><code>M<n>=70</code>:</td><td colspan="4">Switch the master spindle (M70 or M0=70) or spindle with the number <n> (M<n>=70) over to axis mode.
No defined position is approached. The NC block is enabled after the switchover has been performed.</td></tr>
<tr><td><code>FINEA</code>:</td><td colspan="4">Motion end when "Exact stop fine" reached</td></tr>
<tr><td><code>COARSEA</code>:</td><td colspan="4">Motion end when "Exact stop coarse" reached</td></tr>
<tr><td><code>IPOENDA</code>:</td><td colspan="4">End of motion on reaching "interpolator stop"</td></tr>
<tr><td rowspan="3"><code>S<n></code>:</td><td colspan="4">Spindle for which the programmed end-of-motion criterion is to be effective</td></tr>
<tr><td><code><n></code>:</td><td colspan="3">Spindle number</td></tr>
<tr><td colspan="4">If a spindle is not specified in [S<n>] or a spindle number of "0" is specified, the programmed end-of-motion criterion will be applied to the master spindle.</td></tr>
</table>

<table>
<tr><td rowspan="6"><code>IPOBRKA</code>:</td><td colspan="3">A block change is possible in the braking ramp.</td></tr>
<tr><td><code><axis></code>:</td><td colspan="2">Channel axis identifier</td></tr>
<tr><td rowspan="4"><code><instant in time></code>:</td><td colspan="2">Instant in time of the block change with reference to the braking ramp</td></tr>
<tr><td>Unit:</td><td>Percent</td></tr>
<tr><td>Range of values:</td><td>100 (application point of the braking ramp) to 0 (end of the braking ramp)</td></tr>
<tr><td colspan="2">If a value is not assigned to the <instant in time> parameter, the current value of the setting data is applied:
SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE
Note:
IBOBRKA with an instant in time of "0" is identical to IPOENDA.</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td rowspan="4"><code>WAITS</code>:</td><td colspan="3">Synchronization command for the specified spindle(s)
The subsequent blocks are not processed until the specified spindle(s) programmed in a previous NC block with SPOSA has (have) reached its (their) end position(s) (with exact stop fine).</td></tr>
<tr><td><code>WAITS</code> after <code>M5</code>:</td><td colspan="2">Wait for the specified spindle(s) to come to a standstill.</td></tr>
<tr><td><code>WAITS</code> after <code>M3</code>/<code>M4</code>:</td><td colspan="2">Wait for the specified spindle(s) to reach their setpoint speed.</td></tr>
<tr><td><code><n>,<m></code>:</td><td colspan="2">Numbers of the spindles to which the synchronization command is to be applied.
If a spindle number is not specified or if the spindle number is set to "0", WAITS will be applied to the master spindle.</td></tr>
</table>

Note

Three spindle positions are possible for each NC block.

Note

With incremental dimensions IC(<value>), spindle positioning can take place over several revolutions.

Note

If position control was activated with SPCON prior to SPOS, this remains active until SPCOF is issued.

Note

The control detects the transition to axis mode automatically from the program sequence. Explicit programming of M70 in the part program is, therefore, essentially no longer necessary. However, M70 can continue to be programmed, e.g. to increase the legibility of the part program.

More information

Positioning with SPOSA

The block step enable or program execution is not affected by SPOSA. Spindle positioning can be performed during execution of subsequent NC blocks. 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. The spindle positioning operation may be programmed over several blocks (see WAITS).

Note

If a command, which implicitly causes a preprocessing stop, is read in a following block, execution of this block is delayed until all positioning spindles are stationary.

Positioning with SPOS/M19

The block step enabling condition is met when all functions programmed in the block reach their end-of-block criterion (e.g. all auxiliary functions acknowledged by the PLC, all axes at their end point) and the spindle reaches the programmed position.

Velocity of the movements:

The velocity and the delay response for positioning are stored in the machine data. The configured values can be modified by programming or by synchronized actions, see:

- Feedrate for positioning axes / spindles (FA, FPR, FPRAON, FPRAOF) (Page 132)
- Programmable acceleration compensation (ACC) (Page 136)

Specification of spindle positions:

As the G90/G91 commands are not effective here, the corresponding dimensions apply explicitly, e.g. AC, IC, DC, ACN, ACP. If nothing is specified, traversing automatically takes place as for DC.

Synchronize spindle movements with WAITS

WAITS can be used to identify a point at which the NC program waits until one or more spindles programmed with SPOSA in a previous NC block reach their positions.

Example:

Program code	Comment
`N10 SPOSA[2]=180 SPOSA[3]=0`	
`...`	
`N40 WAITS(2,3)`	`; The block waits until spindles 2 and 3 have reached the positions specified in block N10.`

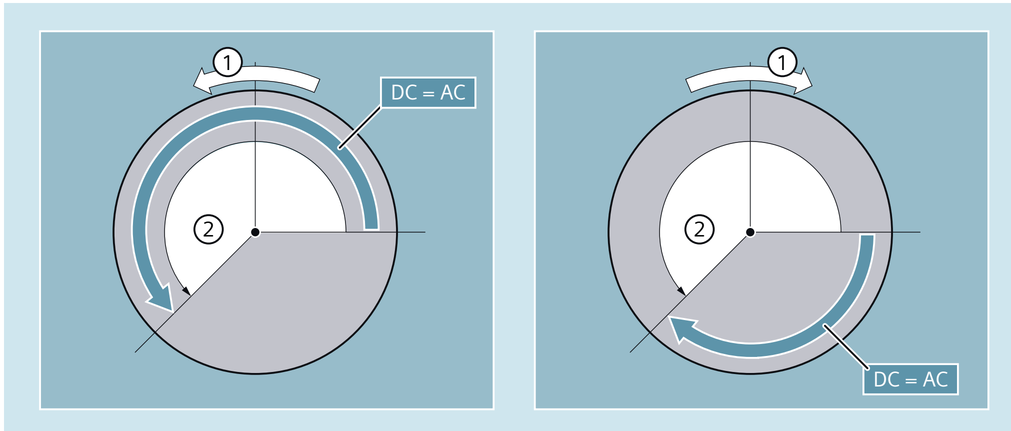
WAITS can be used after M5 to wait until the spindle(s) has (have) stopped. WAITS can be used after M3/M4 to wait until the spindle(s) has (have) reached the specified speed/direction of rotation.

Note

If the spindle has not yet been synchronized with synchronization marks, the positive direction of rotation is taken from the machine data (state on delivery).

Position spindle from rotation (M3/M4)

When M3 or M4 is active, the spindle comes to a standstill at the programmed value.



① Direction of rotation
② Programmed angle

There is no difference between DC and AC dimensioning. In both cases, rotation continues in the direction selected by M3/M4 until the absolute end position is reached. With ACN and ACP, deceleration takes place if necessary, and the appropriate approach direction is taken. With IC dimensioning, the spindle rotates additionally to the specified value starting at the current spindle position.

Position a spindle from standstill (M5)

The exact programmed distance is traversed from standstill (M5).

3.7.5 Feedrate for positioning axes / spindles (FA, FPR, FPRAON, FPRAOF)

It is also possible to derive the revolutional feedrate for path and synchronized axes or for individual positioning axes/spindles from another rotary axis or spindle.

Positioning axes such as workpiece transport systems, tool turrets and end supports are traversed independently of path and synchronized axes. A separate feedrate is therefore defined for each positioning axis.

A separate axial feedrate can also be programmed for spindles.

Syntax

Feedrate for positioning axis:
`FA[<axis>]=...`

Axis feedrate for spindle:
`FA[SPI(<n>)]=...`
`FA[S<n>]=...`

Derive revolutional feedrate for path/synchronized axes:

`FPR (<rotary axis>)`

`FPR(SPI(<n>))`

`FPR(S<n>)`

Derive rotational feedrate for positioning axes/spindles:

```
FPRAON(<axis>,<rotary axis>)
FPRAON(<axis>,SPI(<n>))
FPRAON(<axis>,S<n>)
FPRAON(SPI(<n>),<rotary axis>)
FPRAON(S<n>,<rotary axis>)
FPRAON(SPI(<n>),SPI(<n>))
FPRAON(S<n>,S<n>)
FPRAOF(<axis>,SPI(<n>), etc.)
FPRAOF(<axis>,S<n>, etc.)
```

Meaning

FA[...]=...:	Feedrate for the specified positioning axis or positioning speed (axial feedrate) for the specified spindle Unit: mm/min or inch/min or degrees/min Range of values: ... 999,999.999 mm/min, degrees/min ... 39 999.9999 inch/min
FPR(...):	FPR is used to identify the rotary axis (<rotary axis>) or spindle (SPI(<n>)/S<n>) from which the revolutional feedrate for the revolutional feedrate of the path and synchronized axes programmed under G95 is to be derived.
FPRAON(...):	Derive rotational feedrate for positioning axes and spindles The first parameter (<axis>/SPI(<n>)/S<n>) identifies the positioning axis/spindle to be traversed with revolutional feedrate. The second parameter (<rotary axis>/SPI(<n>)/S<n>) identifies the rotary axis/spindle from which the revolutional feedrate is to be derived. **Note:** The second parameter can be omitted, in which case the feedrate will be derived from the master spindle.
FPRAOF(...):	FPRAOF is used to deselect the derived revolutional feedrate for the specified axes or spindles.
<axis>:	Axis identifier (positioning or geometry axis)
SPI(<n>)/S<n>:	Spindle identifier SPI(<n>) and S<n> are identical in terms of function. <n>: Spindle number **Note:** SPI converts spindle numbers into axis identifiers. The transfer parameter (<n>) must contain a valid spindle number.

Note

The programmed feedrate FA[...] is modal.

Up to five feedrates for positioning axes or spindles can be programmed in each NC block.

Note

The derived feedrate is calculated according to the following formula:

Derived feedrate = programmed feedrate * absolute master feedrate

Examples

Example 1: Synchronous spindle coupling

With synchronous spindle coupling, the positioning speed of the following spindle can be programmed independently of the master spindle, e.g. for positioning operations.

Program code	Comment
...	
FA[S2]=100	; Positioning speed of the following spindle (spindle 2) = 100 degrees/min
...	

Example 2: Derived revolutional feedrate for path axes

Path axes X, Y must be traversed at the revolutional feedrate derived from rotary axis A:

Program code
...
N40 FPR(A)
N50 G95 X50 Y50 F500
...

Example 3: Derive revolutional feedrate for master spindle

Program code	Comment
N30 FPRAON(S1,S2)	; The revolutional feedrate for the master spindle (S1) must be derived from spindle 2.
N40 SPOS=150	; Position master spindle.
N50 FPRAOF(S1)	; Deselect derived revolutional feedrate for the master spindle.

Example 4: Derive revolutional feedrate for positioning axis

Program code	Comment
N30 FPRAON(X)	; The revolutional feedrate for positioning axis X must be derived from the master spindle.
N40 POS[X]=50 FA[X]=500	; The positioning axis is traversing at 500 mm/revolution of the master spindle.
N50 FPRAOF(X)	

Further information

FA[...]

The feedrate type is always G94. When G70/G71 is active, the unit is metric/inches according to the default setting in the machine data. G700/G710 can be used to modify the unit in the program.

Note

If no FA is programmed, the value defined in the machine data applies.

FPR(...)

As an extension of the G95command (revolutional feedrate referring to the master spindle), FPR allows the revolutional feedrate to be derived from any chosen spindle or rotary axis. G95 FPR(…) is valid for path and synchronized axes.

If the rotary axis/spindle specified in the FPR command is operating on position control, then the setpoint linkage is active. Otherwise the actual-value linkage is effective.

FPRAON(...)

FPRAON is used to derive the revolutional feedrate for positioning axes and spindles from the current feedrate of another rotary axis or spindle.

FPRAOF(...)

The revolutional feedrate can be deactivated for one or a number of axes/spindles simultaneously with the FPRAOF command.

3.7.6 Programmable feedrate override (OVR, OVRRAP, OVRA)

The velocity of path/positioning axes and speed of spindles can be modified in the NC program.

Syntax

```
OVR=<value>
OVRRAP=<value>
OVRA[<axis>]=<value>
OVRA[SPI(<n>)]=<value>
OVRA[S<n>]=<value>
```

Meaning

OVR:	Feedrate modification for path feedrate F
OVRRAP:	Feedrate modification for rapid traverse velocity
OVRA:	Feedrate modification for positioning feedrate FA or for spindle speed S
<axis>:	Axis identifier (positioning or geometry axis)

`SPI(<n>)`/`S<n>`:	Spindle identifier `SPI(<n>)` and `S<n>` are identical in terms of function. `<n>`: Spindle number **Note:** `SPI` converts spindle numbers into axis identifiers. The transfer parameter `(<n>)` must contain a valid spindle number.	
`<value>`:	Feedrate modification in percent The value refers to or is combined with the feedrate override set on the machine control panel. Range of values: 0 ... 200%, integral **Note:** With path and rapid traverse override, the maximum velocities set in the machine data is not overshot.	

3.7.7 Programmable acceleration compensation (ACC)

In critical program sections, it may be necessary to limit the acceleration to below the maximum values, e.g. to prevent mechanical vibrations from occurring.

The programmable acceleration override can be used to modify the acceleration for each path axis or spindle via a command in the NC program. The limit is effective for all types of interpolation. The values defined in the machine data apply as 100% acceleration.

Syntax

```
ACC[<axis>]=<value>
ACC[SPI(<n>)]=<value>
ACC(S<n>)=<value>
```

Deactivate:
```
ACC[...]=100
```

Syntax

`ACC`:	Acceleration change for the specified path axis or speed change for the specified spindle.
`<axis>`:	Channel axis name of path axis
`SPI(<n>)`/`S<n>`:	Spindle identifier `SPI(<n>)` and `S<n>` are identical in terms of function. `<n>`: Spindle number **Note:** `SPI` converts spindle numbers into axis identifiers. The transfer parameter `(<n>)` must contain a valid spindle number.

<value>:	Acceleration change in percent The value refers to or is combined with the feedrate override set on the machine control panel.	
	Range of values:	1 to 200%, integers

Note

With a greater acceleration rate, the values permitted by the manufacturer may be exceeded.

Example

Program code	Comment
N50 ACC[X]=80	; The axis slide in the X direction should only be traversed with 80% acceleration.
N60 ACC[SPI(1)]=50	; Spindle 1 should only accelerate or brake with 50% of the acceleration capacity.

Further information

Acceleration override programmed with ACC

The acceleration override programmed with ACC[...] is always taken into consideration for the output, the same as in system variables $AA_ACC. Readout in the part program and in synchronized actions takes place at different times in the NC processing run.

In the part program

The value written in the part program is then only taken into consideration in system variable $AA_ACC as written in the part program if ACC was not changed in the meantime by a synchronized action.

In synchronized actions

The following therefore applies: The value written to a synchronized action is then only considered in system variable $AA_ACC as written to the synchronized action if ACC was not changed in the meantime by a part program.

The specified acceleration can also be changed using synchronized actions.

Example:

Program code
...
N100 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140

The actual acceleration value can be called with system variable $AA_ACC[<Axis >] Machine data can be used to determine whether the ACC value last set should apply on RESET/parts program end or whether it should be set to 100%.

3.7.8 Velocity specification with handwheel override / path specification with handwheel (FD, FDA)

Via the address FD, a feedrate can be specified for the programmed traverse motions of the path axes. This feedrate is only effective in the current block and can be overridden by the handwheel. The handwheel of the first geometry axis of the channel is evaluated. Depending on the direction of rotation, the feedrate specified via FD is increased or decreased.

The FDA address can be used to activate the path specification by the handwheel for positioning axes or to specify a velocity for the axis that is effective in the current block and can be overridden by the handwheel. The handwheel assigned to the axis is evaluated.

The block change is performed as soon as the programmed target position is reached. This automatically deactivates the "Velocity specification with handwheel override" or "Path specification with handwheel" and further handwheel pulses have no effect.

Syntax

Note

Important information on programming FD and FDA:

- A **target position** must be programmed in the same NC block.
- Additional axes can be traversed simultaneously or using interpolation in the same NC block.
- F and FD or FA and FDA must not be programmed in the same NC block.

Path axes

```
X... Y... FD=<Velocity>
```

Positioning axes

```
POS[Axis]=.../POSA[Axis]=... FDA[<Axis>]=<Velocity>
```

Meaning

<table>
<tr><td>X... Y...</td><td colspan="3">Target position of the path axis</td></tr>
<tr><td rowspan="4">FD=<Velocity></td><td colspan="3">Specification of a feedrate with handwheel override</td></tr>
<tr><td rowspan="2"><Velocity></td><td>Value = 0</td><td>Not possible!</td></tr>
<tr><td>Value ≠ 0</td><td>Feedrate</td></tr>
<tr><td colspan="3">FD is only permitted if the following conditions are met:
• G1, G2, G3 or CIP active
• Exact stop G60 active
• Linear feedrate G94 active</td></tr>
<tr><td>POS[Axis]=...
POSA[Axis]=...</td><td colspan="3">Target position of the positioning axis
Target position of the positioning axis beyond block limit</td></tr>
</table>

<table>
<tr><td rowspan="4"><code>FDA[<Axis>]=<Velocity></code></td><td colspan="3">"Velocity specification with handwheel override" or "Path specification with handwheel" for positioning axis</td></tr>
<tr><td><code><Axis></code></td><td colspan="2">Axis identifier of positioning axis</td></tr>
<tr><td rowspan="2"><code><Velocity></code></td><td>Value = 0</td><td>Path specification with handwheel</td></tr>
<tr><td>Value ≠ 0</td><td>Axis velocity</td></tr>
</table>

Note

FD and FDA are **non-modal**.

Exception:

For **FDA** in connection with **POSA**, the handwheel override can also act modally because this positioning axis does not affect the block transition.

Examples

Example 1: Traversing path axes with velocity override

Program code	Description
`N10 X… Y… F500`	`; Feedrate = 500 mm/min`

`N20 X… Y… FD=700`	`; Feedrate = 700 mm/min and velocity override using handwheel.`
	`; Acceleration from 500 to 700 mm/min in N20.`
	`; Using the handwheel, it is possible to vary the path velocity between 0 and the maximum value (machine data) depending on the direction of rotation.`

Example 2: Traversing positioning axes with path specification

Program code	Description
`...`	

`N20 POS[V]=90 FDA[V]=0`	`; Target position = 90 mm, feedrate of the axis = 0 mm/min and path override by handwheel.`
	`; Velocity of axis V at start of block = 0 mm/min.`
	`; Path and speed defaults are set using handwheel pulses.`

In the NC block with programmed FDA[<Axis>]=0, the feedrate is set to zero so that the program cannot generate any traversing movement. The programmed traversing movement to the target position is now controlled exclusively by the operator rotating the handwheel.

Direction of motion, traversing velocity:

The axes follow the path set by the handwheel in the direction of the sign. Forward and backward traversing is possible dependent on the direction of rotation. The faster the handwheel rotates, the higher the traversing velocity.

Traversing range:
The traversing range is limited by the starting position and the programmed end point.

Example 3: Traversing positioning axes with velocity override

Program code	Description
`N10 POS[V]=… FA[V]=100`	`; Feedrate of the axis = 100 mm/min`

`N20 POS[V]=100 FAD[V]=200`	`; Target position of the axis = 100, feedrate of the axis = 200 mm/min and velocity override with handwheel.`
	`; Acceleration from 100 to 200 mm/min in N20.`
	`; Using the handwheel, it is possible to vary the velocity between 0 and the maximum value (machine data) depending on the direction of rotation.`

In NC blocks with programmed FDA[...]=..., the feedrate from the last programmed FA value is accelerated or decelerated to the value programmed under FDA. Starting from the current feedrate FDA, the handwheel can be turned to accelerate the programmed movement to the target position or to decelerate it to zero. The values set as parameters in the machine data serve as the maximum velocity.

Traversing range: The traversing range is limited by the starting position and the programmed end point.

3.7.9 Feedrate reduction with corner deceleration (FENDNORM, G62, G621)

With automatic corner deceleration the feed rate is reduced according to a bell-shaped curve before reaching the corner. It is also possible to parameterize the extent of the tool behavior relevant to machining via setting data.

- Start and end of feed rate reduction
- Override with which the feed rate is reduced
- Detection of a relevant corner

Relevant corners are those whose inside angle is less than the corner parameterized in the setting data.

Default value FENDNORM deactivates the function of the automatic corner override.

Syntax

```
FENDNORM
G62 G41
G621
```

Meaning

`FENDNORM`:	Automatic corner deceleration OFF
`G62`:	Corner deceleration **at inside corners** when tool radius compensation is active
`G621`:	Corner deceleration **at all corners** when tool radius compensation is active

G62 only acts on inner corners with active tool radius compensation G41/G42 (Page 254) and active continuous path mode G64/G641 (Page 294).

The corner is approached at a reduced feedrate, which is calculated as follows:

F * (override for feed rate reduction) * feed rate override

The maximum possible feed rate reduction is attained at the precise point where the tool is to change directions at the corner, with reference to the center path.

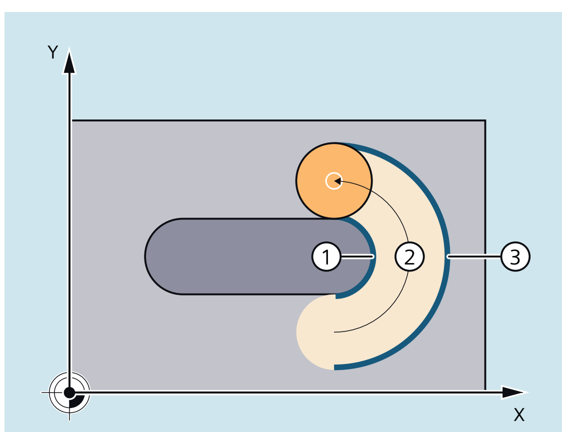
G621 acts analogously to G62 at each corner of the axes defined using FGROUP.

3.7.10 Feedrate optimization for curved path sections (CFTCP, CFC, CFIN)

With activated correction mode G41/G42, the programmed feedrate for the milling tool radius first refers to the milling tool center path (refer to Chapter "Coordinate transformations (frames) (Page 303)").

When you mill a circle (the same applies to polynomial and spline interpolation) the extent to which the feedrate varies at the cutter edge is so significant under certain circumstances that it can impair the quality of the machined part.

Example: Milling a small outside radius with a large tool. The path that the outside of the milling tool must travel is considerably longer than the path along the contour.



① Short path of the inner side of the milling tool along the contour
② Tool path
③ Long path of the outer side of the milling tool

Because of this, machining at the contour takes place with a very low feedrate. To prevent adverse effects, the feedrate needs to be controlled accordingly for curved contours.

Syntax

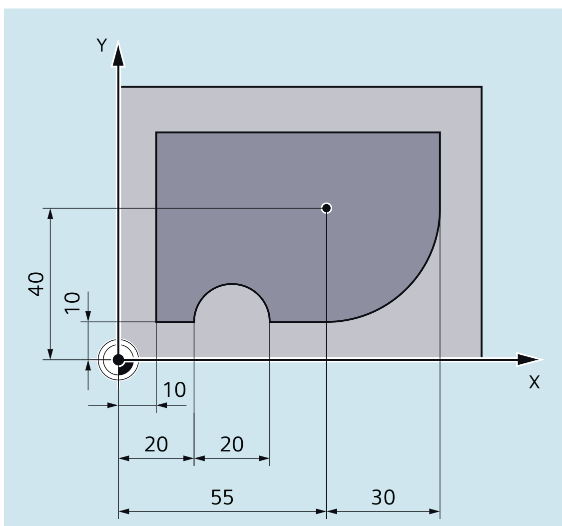
```
CFTCP
CFC
CFIN
```

Meaning

CFTCP	Constant feedrate on the milling cutter center path The control keeps the feedrate constant and feedrate overrides are deactivated.
CFC	Constant feedrate at the contour (tool cutting edge). This function is preset per default.
CFIN	Constant feedrate at the tool cutting edge only at concave contours, otherwise on the milling cutter center path. The feedrate is reduced for inside radii.

Example

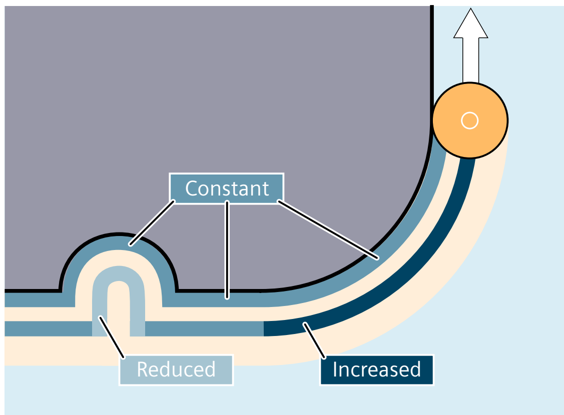
In this example, the contour is first produced with a CFC-corrected feedrate. During finishing, the cutting base is also machined with CFIN. This prevents the cutting base being damaged at the outside radii by an excessively high feedrate.



Program code	Comment
N10 G17 G54 G64 T1 M6	
N20 S3000 M3 CFC F500 G41	
N30 G0 X-10	
N40 Y0 Z-10	; Feed to first cutting depth
N50 CONTOUR1	; Subprogram call
N40 CFIN Z-25	; Feed to second cutting depth
N50 CONTOUR1	; Subprogram call
N60 Y120	
N70 X200 M30	

Further information

Constant feedrate along the contour with CFC



The feedrate is reduced for inside radii and increased for outside radii. This ensures a constant velocity at the tool cutting edge and thus at the contour.

3.7.11 Several feedrate values in one block (F, ST, SR, FMA, STA, SRA)

The "Multiple feedrates in one block" function can be used to activate different feedrate values for an NC block, a dwell time or a retraction motion-synchronously, dependent on external digital and/or analog inputs.

Syntax

Path motion

```
F=... F7=... F6=... F5=... F4=... F3=... F2=... ST=... SR=...
```

Axial motion:

```
FA[<Ax>]=... FMA[7,<Ax>]=... FMA[6,<Ax>]=... FMA[5,<Ax>]=...
FMA[4,<Ax>]=... FMA[3,<Ax>]=... FMA[2,<Ax>]=... STA[<Ax>]=...
SRA[<Ax>]=...
```

Meaning

`F=...`:	The path feedrate is programmed under the address `F` and remains valid during the absence of an input signal.	
	Effective:	Modal
`F2=...` to `F7=...`:	In addition to the path feedrate, up to six further feedrates can be programmed in the block. The numerical expansion indicates the bit number of the input that activates the feedrate when changed:	
	Effective:	Non-modal

ST=...:	Dwell time in s (for grinding technology: sparking-out time)	
	Input bit:	1
	Effective:	Non-modal
SR=...:	Retraction path The unit for the retraction path refers to the current valid unit of measurement (mm or inch).	
	Input bit:	0
	Effective:	Non-modal
FA[<Ax>]=...:	The axial feedrate is programmed under the address FA and remains valid during the absence of an input signal.	
	Effective:	Modal
FMA[2,<Ax>]=... to FMA[7,<Ax>]=...:	In addition to the axial feedrate FA up to six further feedrates per axis can be programmed in the block with FMA. The first parameter indicates the bit number of the input and the second the axis for which the feedrate is to apply.	
	Effective:	Non-modal
STA[<Ax>]=...:	Axial dwell time in s (for grinding technology: sparking-out time)	
	Input bit:	1
	Effective:	Non-modal
SRA[<Ax>]=...:	Axial retraction path	
	Input bit:	0
	Effective:	Non-modal
<Ax>:	Axis for which the feedrate is to apply	

Note

Priority of the signals

The signals are scanned in ascending order starting at input bit 0 (I0). Therefore, the retraction motion has the highest priority and the feedrate F7 the lowest priority. Dwell time and retraction motion end the feedrate motions that were activated with F2 to F7.

The signal with the highest priority determines the current feedrate.

Note

Delete distance-to-go

If input bit 1 is activated for the dwell time or bit 0 for the return path, the distance to go for the path axes or the relevant single axes is deleted and the dwell time or return started.

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, 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.

Examples

Example 1: Path motion

Program code	Comment
G1 X48 F1000 F7=200 F6=50 F5=25 F4=5 ST=1.5 SR=0.5	; Path feedrate = 1000
	; Additional path feedrate values: ; 200 (input bit 7) ; 50 (input bit 6) ; 25 (input bit 5) ; 5 (input bit 4)
	; Dwell time 1.5 s
	; Retraction 0.5 mm

Example 2: Axial motion

Program code	Comment
`POS[A]=300 FA[A]=800 FMA[7,A]=720 FMA[6,A]=640`	`; Feedrate for axis A = 800`

`FMA[5,A]=560 STA[A]=1.5 SRA[A]=0.5`	`; Additional feedrate values for axis A: 720 (input bit 7)`
	`; 640 (input bit 6)`
	`; 560 (input bit 5)`
	`; Axial dwell time: 1.5 s`
	`; Axial retraction: 0.5 mm`

Example 3: Multiple operations in one block

Program code	Comment
`N20 T1 D1 F500 G0 X100`	`Initial setting`

`N25 G1 X105 F=20 F7=5 F3=2.5 F2=0.5 ST=1.5 SR=0.5`	`; Normal feedrate with F,`
	`; roughing with F7,`
	`; finishing with F3,`
	`; smooth-finishing with F2,`
	`; dwell time 1.5 s,`
	`; retraction path 0.5 mm`

`...`	

3.7.12 Non-modal feedrate (FB)

The "Non-modal feedrate" function can be used to define a separate feedrate for a single block. After this block, the previous modal feedrate is active again.

Syntax

```
FB=<value>
```

Meaning

`FB:`	Feedrate for current block only
`<VALUE>:`	The programmed value must be greater than zero. Values are interpreted based on the active feedrate type: • G94: feedrate in mm/min or degrees/min • G95: feedrate in mm/rev or inch/rev • G96: constant cutting rate

Note

If no traversing motion is programmed in the block (e.g. computation block), the `FB` has no effect.

If no explicit feedrate for chamfering/rounding is programmed, then the value of `FB` also applies for any chamfering/rounding contour element in this block.

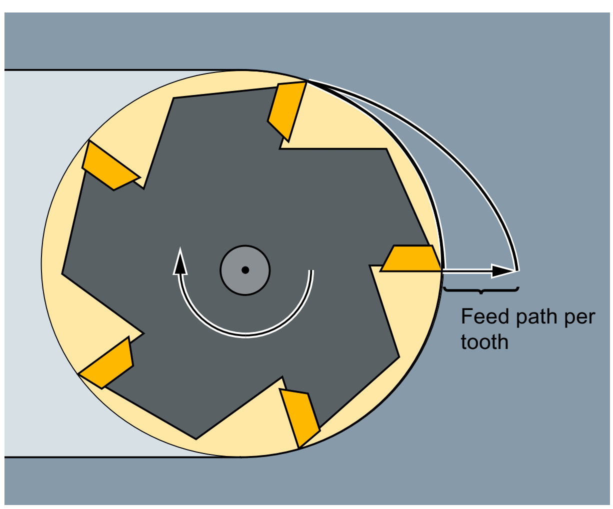
Feedrate interpolations `FLIN`, `FCUB`, etc. are also possible without restriction.

Simultaneous programming of `FB` and `FD` (handwheel travel with feedrate override) or `F` (modal path feedrate) is **not** possible.

Example

Program code	Comment
N10 G0 X0 Y0 G17 F100 G94	;Initial setting
N20 G1 X10	; Feedrate 100 mm/min
N30 X20 FB=80	; Feedrate 80 mm/min
N40 X30	; Feedrate is 100 mm/min again.
...	

3.7.13 Tooth feedrate (G95 FZ)



The control uses the $TC_DPNT (number of teeth) tool parameter associated with the active tool offset data block 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:	System variable tool parameter: Number of teeth/rev

The tool type ($TC_DP1) of the active tool is not taken into account.

The programmed tooth feedrate is independent of the tool change and the selection/deselection of a tool offset data block; it is retained in modal format.

A change to the $TC_DPNT tool parameter associated with the active tool cutting edge will be applied the next time a tool offset is selected or the next time the active offset data is updated.

Changing the tool or selecting/deselecting a tool offset data block generates a recalculation of the effective revolutional feedrate.

Note

The tooth feedrate refers only to the path (axis-specific programming is not possible).

Syntax

```
G95 FZ...
```

Meaning

`G95`:	Type of feedrate: Revolutional feedrate in mm/rev or inch/rev (dependent upon `G700`/`G710`) For `G95` see "Feedrate (G93, G94, G95, F, FGROUP, FL, FGREF) (Page 115)"	
`FZ`:	Tooth feedrate	
	Activation:	with `G95`
	Effectiveness:	Modal
	Unit:	mm/tooth or inch/tooth (dependent upon `G700`/`G710`)

NOTICE

Tool change/Changing the master spindle

A subsequent tool change or changing the master spindle must be taken into account by the user by means of corresponding programming, e.g. reprogramming `FZ`.

NOTICE
Tool operations undefined
Technological concerns such as climb milling or conventional milling, front face milling or peripheral face milling, etc., along with the path geometry (straight line, circle, etc.), are not taken into account automatically. Therefore, these factors have to be given consideration when programming the tooth feedrate.

Note

Switchover between G95 F... and G95 FZ...

With switchover between `G95 F...` (revolution feedrate) and `G95 FZ...` (tooth feedrate), the inactive feedrate value is deleted in each case.

Note

Derive feedrate with FPR

As is the case with the revolutional feedrate, `FPR` can also be used to derive the tooth feedrate of any rotary axis or spindle (see "Feedrate for positioning axes / spindles (FA, FPR, FPRAON, FPRAOF) (Page 132)").

Examples

Example 1: Milling cutter with 5 teeth ($TC_DPNT = 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.
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
...	

Example 2: Switchover between G95 F... and G95 FZ...

Program code	Comment
N10 G0 X100 Y50	
N20 G1 G95 F0.1	; Revolutional feedrate 0.1 mm/rev
N30 T1 M6	
N35 M3 S100 D1	
N40 X20	
N50 G0 X100 M5	

Program code	Comment
N60 M6 T3 D1	; Load tool with e.g. five teeth ($TC_DPNT = 5).
N70 X22 M3 S300	
N80 G1 X3 G95 FZ=0.02	; Change G95 F… to G95 FZ…, tooth feedrate active with 0.02 mm/tooth.
...	

Example 3: Derive tooth feedrate of a spindle (FBR)

Program code	Comment
…	
N41 FPR(S4)	; Tool in spindle 4 (not the master spindle).
N51 G95 X51 FZ=0.5	; Tooth feedrate 0.5 mm/tooth dependent upon spindle S4.
...	

Example 4: Subsequent tool change

Program code	Comment
N10 G0 X50 Y5	
N20 G1 G95 FZ=0.03	; Tooth feedrate 0.03 mm/tooth
N30 M6 T11 D1	; Load tool with e.g. seven teeth ($TC_DPNT = 7).
N30 M3 S100	
N40 X30	; Effective revolutional feedrate 0.21 mm/rev
N50 G0 X100 M5	
N60 M6 T33 D1	; Load tool with e.g. five teeth ($TC_DPNT = 5).
N70 X22 M3 S300	
N80 G1 X3	; Tooth feedrate modal 0.03 mm/tooth, effective revolutional feedrate 0.15 mm/rev
...	

Example 5: Changing the master spindle

Program code	Comment
N10 SETMS (1)	; Spindle 1 is the master spindle.
N20 T3 D3 M6	; Tool 3 is changed to spindle 1.
N30 S400 M3	; Speed S400 of spindle 1 (and therefore T3).
N40 G95 G1 FZ0.03	; Tooth feedrate 0.03 mm/tooth
N50 X50	; Path motion, the effective feedrate is dependent upon:
	- The tooth feedrate FZ
	- The speed of spindle 1
	- The number of teeth of the active tool T3
N60 G0 X60	
...	
N100 SETMS(2)	; Spindle 2 becomes the master spindle.
N110 T1 D1 M6	; Tool 1 is changed to spindle 2.
N120 S500 M3	; Speed S500 of spindle 2 (and therefore T1).

Program code	Comment
N130 G95 G1 FZ0.03 X20	; Path motion, the effective feedrate is dependent upon:
	- The tooth feedrate FZ
	- The speed of spindle 2
	- The number of teeth of the active tool T1

Note

Following the change in the master spindle (N100), a tool actuated by spindle 2 must be substituted (N110).

Further information

Changing between G93, G94 and G95

FZ can also be programmed when G95 is not active, although it will have no effect and is deleted when G95 is selected. In other words, when changing between G93, G94, and G95, in the same way as with F, the FZ value is also deleted.

Reselection of G95

Reselecting G95 when G95 is already active has no effect (unless a change between F and FZ has been programmed).

Non-modal feedrate (FB)

When G95 FZ... (modal) is active, a non-modal feedrate FB... is interpreted as a tooth feedrate.

SAVE mechanism

In subprograms with the SAVE attribute FZ is written to the value prior to the subprogram starting (in the same way as F).

Multiple feedrate values in one block

The "Multiple feedrate values in one block" function is not possible with tooth feedrate.

Synchronized actions

FZ cannot be programmed from synchronized actions.

Read tooth feedrate and path feedrate type

The tooth feedrate and the path feedrate type can be read using system variables.

- With preprocessing stop in the part program via system variables:

	$AC_FZ	Tooth feedrate effective when the current main run block was preprocessed.	
	$AC_F_TYPE	Path feedrate type effective when the current main run block was preprocessed.	
		Value:	**Meaning:**
		0	mm/min
		1	mm/rev
		2	inch/min
		3	inch/rev
		11	mm/tooth
		33	inch/tooth

- Without preprocessing stop in the part program via system variables:

	$P_FZ	Programmed tooth feedrate	
	$P_F_TYPE	Programmed path feedrate type	
		Value:	**Meaning:**
		0	mm/min
		1	mm/rev
		2	inch/min
		3	inch/rev
		11	mm/tooth
		33	inch/tooth

Note

If `G95` is not active, the $P_FZ and $AC_FZ variables will always return a value of zero.

3.8 Geometry settings

3.8.1 Settable work offset (G54 ... G59, G507 ... G599, G53, G500, SUPA, G153)

The G54 to G59 and G507 to G599 commands activate the settable work offsets for offsetting the workpiece coordinate system compared with the basic coordinate system set from the user interface.

Syntax

Switching on:
```
G54/.../G59/G507/.../G599
```

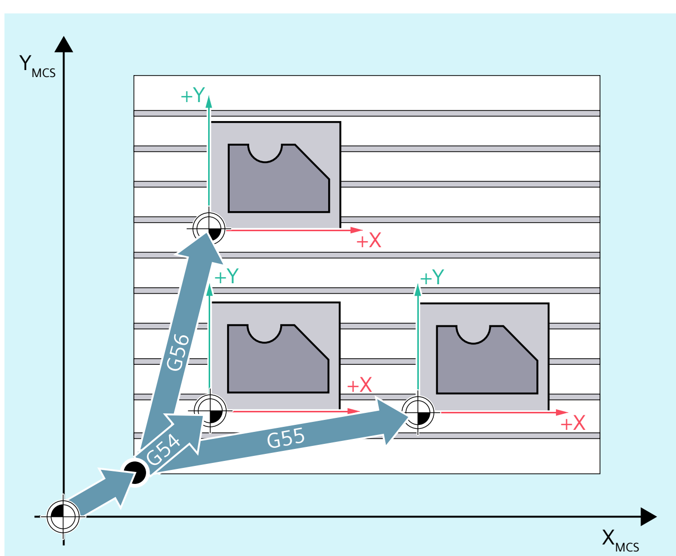
Switching off or suppressing:
`G500/G53/G153/SUPA`

Meaning

`G54 ... G59`	Call of the 1st to 6th settable work offset (WO)	
`G505 ... G599`	Call of the 5th to 99th settable work offset	
`G500`	Deactivation of the current settable work offset	
	G500=Zero frame: (default setting; contains no offset, rotation, mirroring or scaling)	Deactivation of the settable work offset until the next call, activation of the entire basic frame ($P_ACTBFRAME).
	G500 not equal to 0:	Activation of the first settable work offset ($P_UIFR[0]) and activation of the entire basic frame ($P_ACTBFRAME) or possibly a modified basic frame is activated.
`G53`	G53 suppresses the settable work offset and the programmable work offset non-modally.	
`G153`	G153 has the same effect as G53 and also suppresses the entire basic frame.	
`SUPA`	SUPA has the same effect as G153 and also suppresses: • Handwheel offsets (DRF) • Overlaid movements • External work offset • Preset offset	

Example

Three workpieces that are arranged on a palette according to the work offset values G54 to G56 are to be machined in succession. The machining sequence is programmed in subprogram L47.



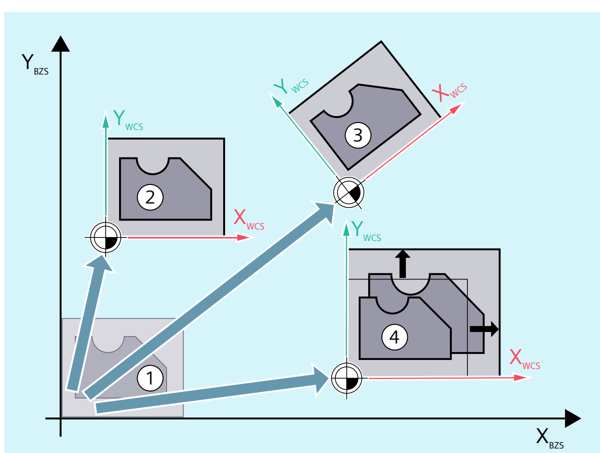
Program code	Comment
N10 G0 G90 X10 Y10 F500 T1	; Approach
N20 G54 S1000 M3	; Call of the first WO, spindle clockwise
N30 L47	; Program pass as subprogram
N40 G55 G0 Z200	; Call of the second WO, Z via obstruction
N50 L47	; Program pass as subprogram
N60 G56	; Call of the third WO
N70 L47	; Program pass as subprogram
N80 G53 X200 Y300 M30	; Suppress work offset, end of program

More information

Settable work offset

A settable work offset is in principle a set frame (Page 303). Consequently, the following components and frame values are also available for a settable work offset:

- Offset
- Rotation
- Scaling
- Scale



① Initial position in BZS
② Offset
③ Offset + rotation
④ Offset + scaling

The frame values for the settable work offsets are input from the user interface:

SINUMERIK Operate: "Parameters" > "Work offsets" > "Details" operating area

Parameterizable number of settable frames (G507 - G599)

The number of user-specific settable work offsets (G507 - G599) can be set for each specific channel via:

MD28080 $MC_MM_NUM_USER_FRAMES = <number>

3.8.2 Selection of the working plane (G17/G18/G19)

The specification of the working plane, in which the desired contour is to be machined, also defines the following parameters:

- Plane for tool radius compensation
- Infeed direction for the tool length compensation depending on the type of tool
- Plane for circular interpolation

Syntax

```
G17/G18/G19, etc.
```

Meaning

`G17`	Working plane X/Y	
	Infeed direction:	Z
	Plane selection:	1st - 2nd geometry axis
`G18`	Working plane Z/X	
	Infeed direction:	Y
	Plane selection:	3rd – 1st geometry axis
`G19`	Working plane Y/Z	
	Infeed direction:	X
	Plane selection:	2nd - 3rd geometry axis

Example

The "conventional" approach for milling is:

1. Define working plane (G17 default setting for milling).
2. Select tool type (T) and tool offset values (D).
3. Switch on path correction (G41).
4. Program traversing movements.

Program code	Comment
`N10 G17 T5 D8`	`; Call of working plane X/Y, tool call. Tool length offset is performed in the Z direction.`
`N20 G1 G41 X10 Y30 Z-5 F500`	`; Radius compensation is performed in the X/Y plane.`
`N30 G2 X22.5 Y40 I50 J40`	`; Circular interpolation / tool radius compensation in the X/Y plane.`

See also

Tool radius compensation (Page 254)

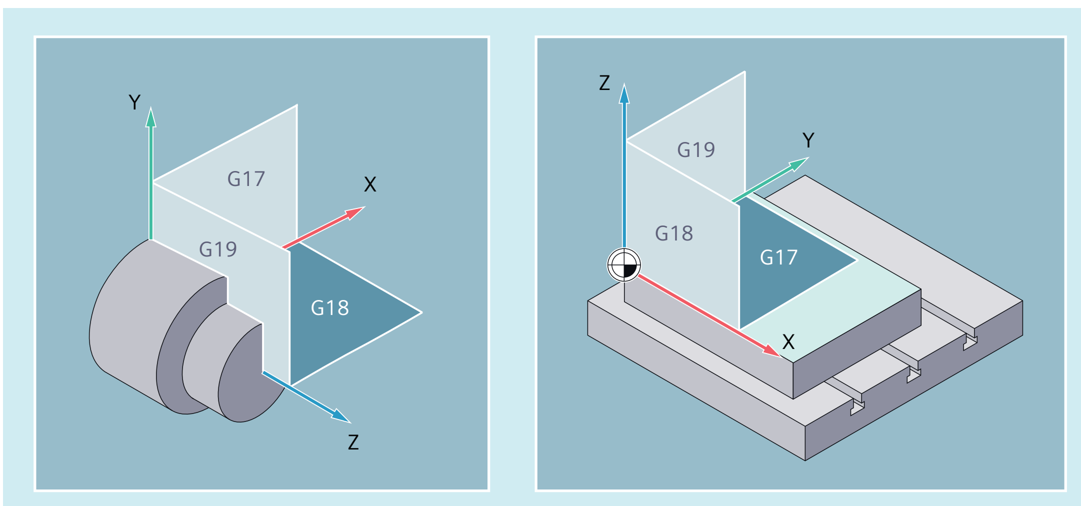
More information

Time of activation

It is recommended that the working plane G17 to G19 be selected at the start of the program.

Default setting

In the default setting, G18 (Z/X plane) is defined for turning and G17 (X/Y plane) is defined for milling:



Tool radius compensation

When calling the tool radius compensation G41/G42, the working plane must be defined so that the controller can correct the tool length and radius.

More information: → Chapter "Tool radius compensation (Page 254)"

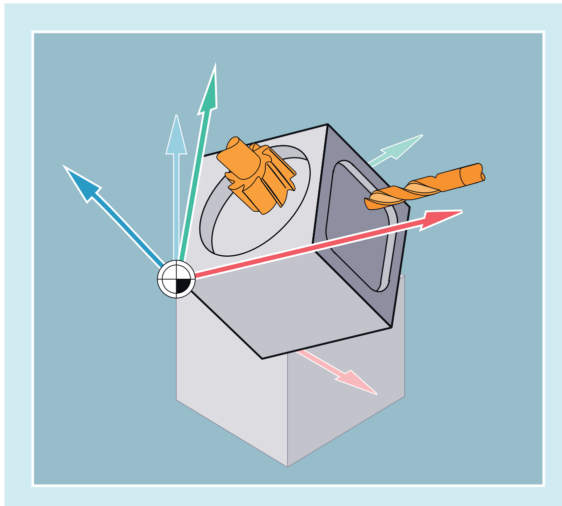
Circular interpolation

The controller requires the specification of the working plane for the calculation of the direction of rotation.

More information: → Chapter "Circular interpolation (Page 196)".

Machining on inclined planes

Rotate the coordinate system with ROT (Page 312) to position the coordinate axes on the inclined surface. The working planes rotate accordingly:



Tool length compensation on inclined planes

The calculation of the tool length compensation always refers to the non-rotated working plane that is fixed in space.

Note

The tool length components can be calculated according to the rotated working planes with the functions for "Tool length compensation for orientable tools".

The offset plane is selected with CUT2D/CUT2DF.

More information: → Chapter "Tool radius compensation (Page 254)"

The control provides convenient coordinate transformation functions for the spatial definition of the working plane.

More information: → Chapter "Coordinate transformations (frames) (Page 303)"

3.8.3 Dimensions

The basis of most NC programs is a workpiece drawing with specific dimensions.

These dimensions can be:

- In absolute dimensions or in incremental dimensions
- In millimeters or inches
- In radius or diameter (for turning)

Specific programming commands are available for the various dimension options so that the data from a dimension drawing can be transferred directly (without conversion) to the NC program.

3.8.3.1 Absolute dimensions (G90, AC)

With absolute dimensions, the position specifications always refer to the zero point of the currently valid coordinate system, i.e. the absolute position is programmed, on which the tool is to traverse.

Modal absolute dimensions

Modal absolute dimensions are activated with the G90 command. Generally it applies to all axes programmed in subsequent NC blocks.

Non-modal absolute dimensions

With preset incremental dimensions (G91), the AC command can be used to set non-modal absolute dimensions for individual axes.

Note

Non-modal absolute dimensions (AC) are also possible for spindle positioning (SPOS, SPOSA) and interpolation parameters (I, J, K).

Syntax

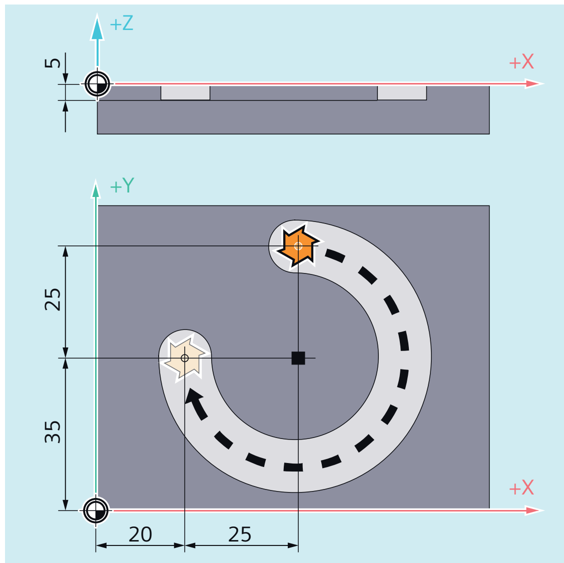
```
G90
<axis>=AC(<value>)
```

Meaning

G90:	Command for the activation of modal absolute dimensions
AC:	Command for the activation of non-modal absolute dimensions
<axis>:	Axis identifier of the axis to be traversed
<value>:	Position setpoint of the axis to be traversed in absolute dimensions

Examples

Example 1: Milling

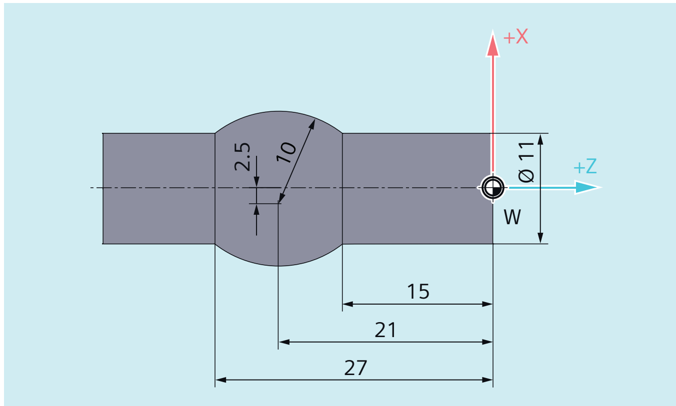


Program code	Comment
N10 G90 G0 X45 Y60 Z2 T1 S2000 M3	; Absolute dimension input, in rapid traverse to position XYZ, tool selection, spindle on with clockwise direction of rotation.
N20 G1 Z-5 F500	; Linear interpolation, feed of the tool.
N30 G2 X20 Y35 I=AC(45) J=AC(35)	; Clockwise circular interpolation, circle end point and circle center point in absolute dimensions.
N40 G0 Z2	; Traverse
N50 M30	; End of block

Note

For information on the input of the circle center point coordinates I and J, see Section "Circular interpolation".

Example 2: Turning



Program code	Comment
N5 T1 D1 S2000 M3	; Loading of tool T1, spindle on with clockwise direction of rotation.
N10 G0 G90 X11 Z1	; Absolute dimension input, in rapid traverse to position XZ.
N20 G1 Z-15 F0.2	; Linear interpolation, feed of the tool.
N30 G3 X11 Z-27 I=AC(-5) K=AC(-21)	; Counter-clockwise circular interpolation, circle end point and circle center point in absolute dimensions.
N40 G1 Z-40	; Traverse
N50 M30	; End of block

Note

For information on the input of the circle center point coordinates I and J, see Section "Circular interpolation".

See also

Absolute and incremental dimensions for turning and milling (G90/G91) (Page 164)

3.8.3.2 Incremental dimensions (G91, IC)

With incremental dimensions, the position specification refers to the last point approached, i.e. the programming in incremental dimensions describes by how much the tool is to be traversed.

Modal incremental dimensions

Modal incremental dimensions are activated with the G91 command. Generally it applies to all axes programmed in subsequent NC blocks.

Non-modal incremental dimensions

With preset absolute dimensions (G90), the IC command can be used to set non-modal incremental dimensions for individual axes.

Note

Non-modal incremental dimensions (IC) are also possible for spindle positioning (SPOS, SPOSA) and interpolation parameters (I, J, K).

Syntax

```
G91
<Axis>=IC(<Value>)
```

Meaning

`G91`:	Command for the activation of modal incremental dimensions
`IC`:	Command for the activation of non-modal incremental dimensions
`<Axis>`:	Axis identifier of the axis to be traversed
`<Value>`:	Position setpoint of the axis to be traversed in incremental dimensions

G91 extension

For certain applications, such as scratching, it is necessary that only the programmed distance is traversed in incremental dimensions. The active work offset or tool length compensation is not traversed.

This behavior can be set separately for the active work offset and tool length compensation via the following setting data:

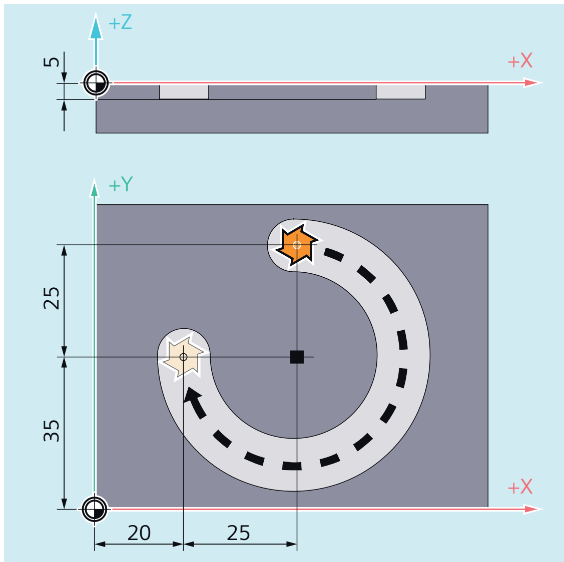
SD42440 $SC_FRAME_OFFSET_INCR_PROG (zero offsets in frames)

SD42442 $SC_TOOL_OFFSET_INCR_PROG (tool length compensations)

Value	Meaning
0	With incremental programming (incremental dimensions) of an axis, the work offset or the tool length compensation is **not** traversed.
1	With incremental programming (incremental dimensions) of an axis, the work offset or the tool length compensation is traversed.

Examples

Example 1: Milling

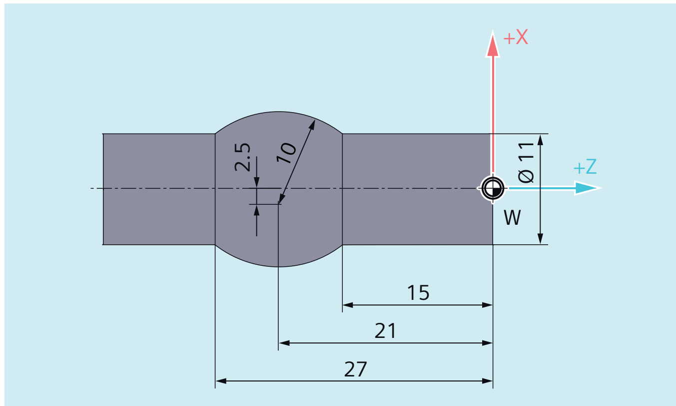


Program code	Comment
N10 G90 G0 X45 Y60 Z2 T1 S2000 M3	; Absolute dimension input, in rapid traverse to position XYZ, tool selection, spindle on with clockwise direction of rotation
N20 G1 Z-5 F500	; Linear interpolation, feed of the tool.
N30 G2 X20 Y35 I0 J-25	; Clockwise circular interpolation, circle end point in absolute dimensions, circle center point in incremental dimensions.
N40 G0 Z2	; Exit.
N50 M30	; End of block

Note

For information on the input of the circle center point coordinates I and J, see Section "Circular interpolation".

Example 2: Turning



Program code	Comment
N5 T1 D1 S2000 M3	; Loading of tool T1, spindle on with clockwise direction of rotation.
N10 G0 G90 X11 Z1	; Absolute dimension input, in rapid traverse to position XZ.
N20 G1 Z-15 F0.2	; Linear interpolation, feed of the tool.
N30 G3 X11 Z-27 I-8 K-6	; Counter-clockwise circular interpolation, circle end point in absolute dimensions, circle center point in incremental dimensions.
N40 G1 Z-40	; Exit.
N50 M30	; End of block

Note

For information on the input of the circle center point coordinates I and J, see Section "Circular interpolation".

Example 3: Incremental dimensions without traversing of the active work offset

Settings:

- G54 contains an offset in X of 25
- SD42440 $SC_FRAME_OFFSET_INCR_PROG = 0

Program code	Comment
N10 G90 G0 G54 X100	
N20 G1 G91 X10	; Incremental dimensions active, traversing in X of 10 mm (the work offset is not traversed).
N30 G90 X50	; Absolute dimensions active, traverse to position X75 (the work offset is traversed).

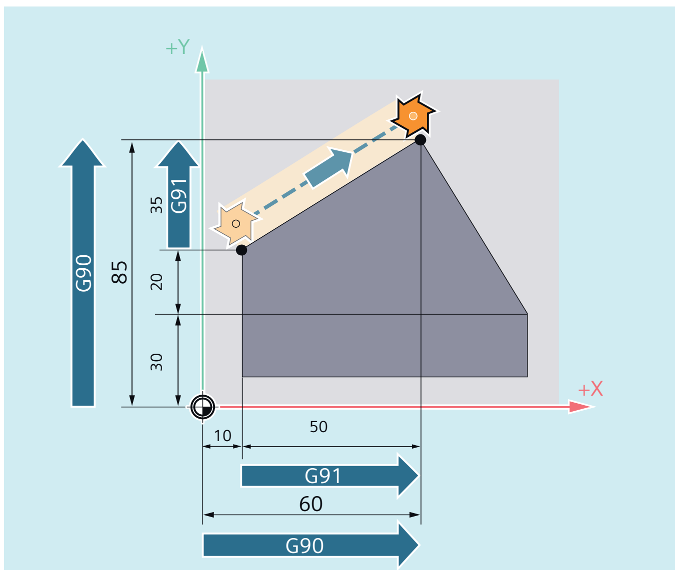
See also

Absolute and incremental dimensions for turning and milling (G90/G91) (Page 164)

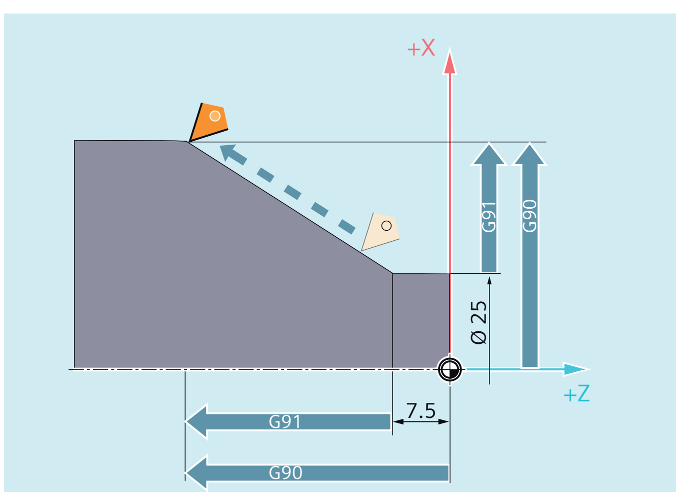
3.8.3.3 Absolute and incremental dimensions for turning and milling (G90/G91)

The two following figures illustrate the programming with absolute dimensions (`G90`) or incremental dimensions (`G91`) using turning and milling technology examples.

Milling:



Turning:



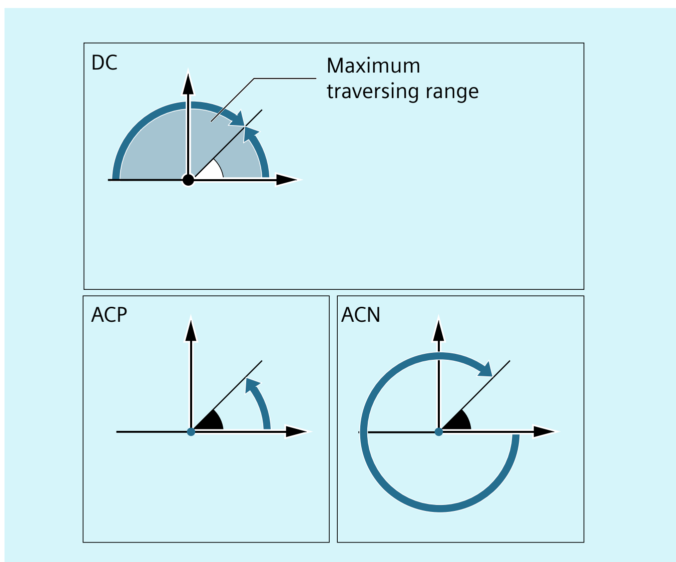
Note

On conventional turning machines, it is usual to consider incremental traversing blocks in the transverse axis as radius values, while diameter specifications apply for the reference dimensions. This conversion for `G90` is performed using the commands `DIAMON`, `DIAMOF` or `DIAM90`.

3.8.3.4 Absolute dimensions for rotary axes (DC, ACP, ACN)

The non-modal and G90/G91-independent commands DC, ACP and ACN are available for positioning rotary axes in absolute dimensions.

DC, ACP and ACN differ in the basic approach strategy:



Syntax

```
<Rotary axis>=DC(<Value>)
<Rotary axis>=ACP(<Value>)
<Rotary axis>=ACN(<Value>)
```

Meaning

`<Rotary axis>`:	Identifier of the rotary axis that is to be traversed (e.g. A, B or C)
`DC`:	Command to **directly** approach the position The rotary axis approaches the programmed position directly along the shortest path. The rotary axis traverses a maximum range of 180°.
`ACP`:	Command to approach the position in the **positive** direction The rotary axis traverses to the programmed position in the positive direction of axis rotation (counter-clockwise).

`ACN`:	Command to approach the position in the **negative** direction The rotary axis traverses to the programmed position in the negative direction of axis rotation (clockwise).	
`<Value>`:	Rotary axis position to be approached in absolute dimensions	
	Value range:	0 - 360 degrees

Note

The positive direction of rotation (clockwise or counter-clockwise) is set in the machine data.

Note

The traversing range between 0° and 360° must be set in the machine data (modulo behavior) for positioning where the direction is specified (ACP, ACN). G91 or IC must be programmed to traverse modulo rotary axes more than 360° in a block.

Note

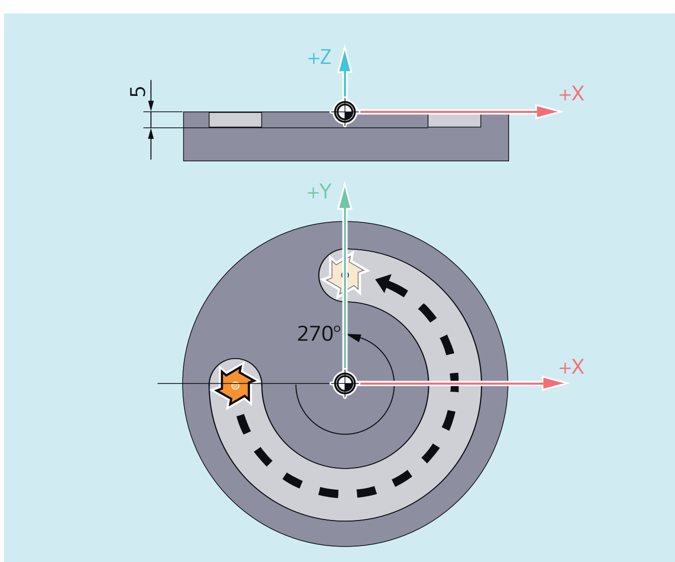
The commands DC, ACP and ACN can also be used for spindle positioning (SPOS, SPOSA) from standstill.

Example: SPOS=DC(45)

Example

Milling on a rotary table

The tool is stationary, the table turns to 270° **in a clockwise direction**. A circular groove/slot is machined.



Program code	Comment
`N10 SPOS=0`	`; Spindle in position control.`

Program code	Comment
N20 G90 G0 X-20 Y0 Z2 T1	; Absolute dimensions, feed in tool T1 in rapid traverse.
N30 G1 Z-5 F500	; Lower tool with the feedrate.
N40 C=ACP(270)	; Table turns clockwise to 270 degrees (positive), the tool mills a circular groove.
N50 G0 Z2 M30	; Retraction, end of program.

3.8.3.5 Metric/inch dimension system (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

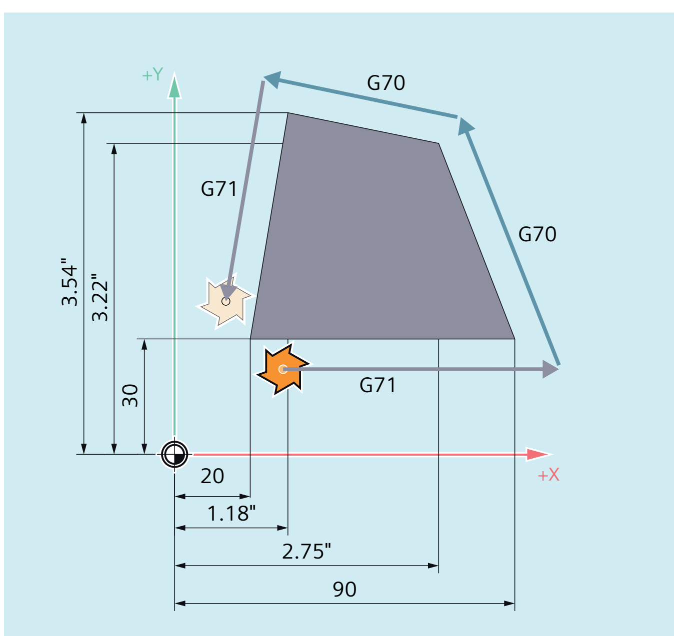
<table>
<tr><td rowspan="4">G70:</td><td colspan="2">Activating the inch system of units
The inch system of units is used to read and write geometrical data in units of length.
Technological data in units of length (e.g. feedrates, tool offsets, adjustable work offsets, machine data and system variables) is read and written using the parameterized basic system.</td></tr>
<tr><td>G group:</td><td>13</td></tr>
<tr><td>Initial setting:</td><td>Settable via MD20150 $MC_GCODE_RESET_VALUES</td></tr>
<tr><td>Effectiveness:</td><td>Modal</td></tr>
<tr><td rowspan="4">G71:</td><td colspan="2">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.</td></tr>
<tr><td>G group:</td><td>13</td></tr>
<tr><td>Initial setting:</td><td>Settable via MD20150 $MC_GCODE_RESET_VALUES</td></tr>
<tr><td>Effectiveness:</td><td>Modal</td></tr>
</table>

`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`

Further 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
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 areas	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

Data area	G70 / G71		G700 / G710	
	Read	Write	Read	Write
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

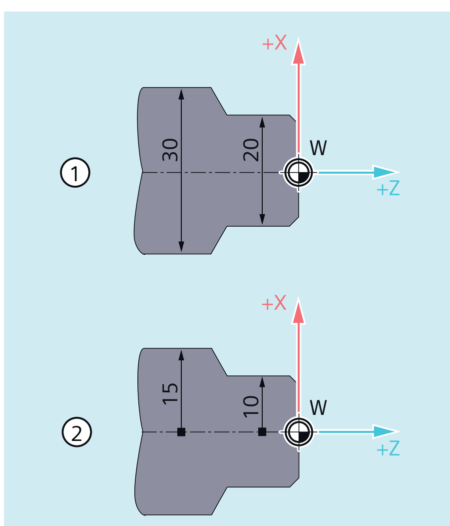
Reading position data in synchronized actions

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**.

Further information: Function Manual, Synchronized Actions

3.8.3.6 Channel-specific diameter/radius programming (DIAMON, DIAM90, DIAMOF, DIAMCYCOF)

During turning, the dimensions **for the transverse axis** can be specified in the diameter (①) or in the radius (②):



So that the dimensions from a technical drawing can be transferred directly (without conversion) to the NC program, channel-specific diameter or radius programming is activated using the modal commands `DIAMON`, `DIAM90`, `DIAMOF`, and `DIAMCYCOF`.

Note

The channel-specific diameter/radius programming refers to the geometry axis defined as transverse axis via MD20100 $MC_DIAMETER_AX_DEF (→ see machine manufacturer's specifications).

Only one transverse axis per channel can be defined via MD20100.

Syntax

```
DIAMON
DIAM90
DIAMOF
```

Meaning

`DIAMON`:	Command for the activation of the **independent** channel-specific diameter programming. The effect of `DIAMON` is independent of the programmed dimensions mode (absolute dimensions `G90` or incremental dimensions `G91`):	
	• For G90:	Dimensions in the diameter
	• For G91:	Dimensions in the diameter
`DIAM90`:	Command for the activation of the **dependent** channel-specific diameter programming. The effect of `DIAM90` depends on the programmed dimensions mode:	
	• For G90:	Dimensions in the diameter
	• For G91:	Dimensions in the radius
`DIAMOF`:	Command for the deactivation of the channel-specific diameter programming Channel-specific radius programming takes effect when diameter programming is deactivated. The effect of `DIAMOF` is independent of the programmed dimensions mode:	
	• For G90:	Dimensions in the radius
	• For G91:	Dimensions in the radius
`DIAMCYCOF`:	Command for the deactivation of channel-specific diameter programming during cycle processing. In this way, computations in the cycle can always be made in the radius. The last G command active in this group remains active for the position indicator and the basic block indicator.	

Note

With `DIAMON` or `DIAM90`, the transverse-axis actual values will always be displayed as a diameter. This also applies to reading of actual values in the workpiece coordinate system with `MEAS`, `MEAW`, `$P_EP[x]` and `$AA_IW[x]`.

Example

Program code	Comment
N10 G0 X0 Z0	; Approach starting point.
N20 DIAMOF	; Diameter programming off.
N30 G1 X30 S2000 M03 F0.7	; X axis = transverse axis, radius programming active; traverse to radius position X30.
N40 DIAMON	; The diameter programming is active for the transverse axis.
N50 G1 X70 Z-20	; Traverse to diameter position X70 and Z-20.
N60 Z-30	
N70 DIAM90	; Diameter programming for absolute dimensions and radius programming for incremental dimen-sions.
N80 G91 X10 Z-20	; Incremental dimensions active.
N90 G90 X10	; Absolute dimensions active.
N100 M30	; End of program

Additional information

Diameter values (DIAMON/DIAM90)

The diameter values apply for the following data:

- Actual value display of the transverse axis in the workpiece coordinate system
- JOG mode: Increments for incremental dimensions and manual handwheel travel
- Programming of end positions:
  Interpolation parameters `I`, `J`, `K` for `G2`/`G3`, if these have been programmed absolutely with `AC`.
  If `I`, `J`, `K` are programmed incrementally (`IC`), the radius is always calculated.
- Reading actual values in the workpiece coordinate system for:
  `MEAS`, `MEAW`, `$P_EP[X]`, `$AA_IW[X]`

3.8.3.7 Axis-specific diameter/radius programming (DIAMONA, DIAM90A, DIAMOFA, DIACYCOFA, DIAMCHANA, DIAMCHAN, DAC, DIC, RAC, RIC)

In addition to channel-specific diameter programming, the axis-specific diameter programming function enables the modal or non-modal dimensions and display in the diameter for one or more axes.

Note

The axis-specific diameter programming is only possible for axes that are permitted as further transverse axes for the axis-specific diameter programming via MD30460 $MA_BASE_FUNCTION_MASK (→ see machine manufacturer's specifications).

Syntax

Modal axis-specific diameter programming for several transverse axes in the channel:

```
DIAMONA[<axis>]
DIAM90A[<axis>]
DIAMOFA[<axis>]
DIACYCOFA[<axis>]
```

Acceptance of the channel-specific diameter/radius programming:

```
DIAMCHANA[<axis>]
DIAMCHAN
```

Non-modal axis-specific diameter/radius programming:

```
<axis>=DAC(<value>)
<axis>=DIC(<value>)
<axis>=RAC(<value>)
<axis>=RIC(<value>)
```

Meaning

Modal axis-specific diameter programming		
`DIAMONA`:	Command for the activation of the **independent** axis-specific diameter programming The effect of `DIAMONA` is independent of the programmed dimensions mode (`G90`/`G91` or `AC`/`IC`):	
	• For G90, AC:	Dimensions in the diameter
	• For G91, IC:	Dimensions in the diameter
`DIAM90A`:	Command for the activation of the **dependent** axis-specific diameter programming The effect of `DIAM90A` depends on the programmed dimensions mode:	
	• For G90, AC:	Dimensions in the diameter
	• For G91, IC:	Dimensions in the radius
`DIAMOFA`:	Command for the deactivation of the axis-specific diameter programming Axis-specific radius programming takes effect when diameter programming is deactivated. The effect of `DIAMOFA` is independent of the programmed dimensions mode:	
	• For G90, AC:	Dimensions in the radius
	• For G91, IC:	Dimensions in the radius
`DIACYCOFA`:	Command for the deactivation of axis-specific diameter programming during cycle processing. In this way, computations in the cycle can always be made in the radius. The last G command active in this group remains active for the position indicator and the basic block indicator.	

<table>
<tr><td rowspan="2"><axis>:</td><td colspan="2">Axis identifier of the axis for which the axis-specific diameter programming is to be activated.
Permitted axis identifiers are as follows:
• Geometry/channel axis name
or
• Machine axis name</td></tr>
<tr><td>Range of values:</td><td>The axis specified must be a known axis in the channel.
Other conditions:
• The axis must be permitted for the axis-specific diameter programming via MD30460 $MA_BASE_FUNCTION_MASK.
• Rotary axes are not permitted to serve as transverse axes.</td></tr>
<tr><td colspan="3">Acceptance of the channel-specific diameter/radius programming</td></tr>
<tr><td>DIAMCHANA:</td><td colspan="2">With the DIAMCHANA[<axis>] command, the specified axis accepts the channel status of the diameter/radius programming and is then assigned to the channel-specific diameter/radius programming.</td></tr>
<tr><td>DIAMCHAN:</td><td colspan="2">With the DIAMCHAN command, all axes permitted for the axis-specific diameter programming accept the channel status of the diameter/radius programming and are then assigned to the channel-specific diameter/radius programming.</td></tr>
<tr><td colspan="3">Non-modal axis-specific diameter/radius programming
The non-modal axis-specific diameter/radius programming specifies the dimension type as a diameter or radius value in the part program and synchronized actions. The modal status of diameter/radius programming remains unchanged.</td></tr>
<tr><td>DAC:</td><td colspan="2">The DAC command sets the following dimensions to non-modal for the specified axis:
Diameter in absolute dimensions</td></tr>
<tr><td>DIC:</td><td colspan="2">The DIC command sets the following dimensions to non-modal for the specified axis:
Diameter in incremental dimensions</td></tr>
<tr><td>RAC:</td><td colspan="2">The RAC command sets the following dimensions to non-modal for the specified axis:
Radius in absolute dimensions</td></tr>
<tr><td>RIC:</td><td colspan="2">The RIC command sets the following dimensions to non-modal for the specified axis:
Radius in incremental dimensions</td></tr>
</table>

Note

With `DIAMONA[<axis>]` or `DIAM90A[<axis>]`, the transverse-axis actual values are always displayed as a diameter. This also applies to reading of actual values in the workpiece coordinate system with `MEAS`, `MEAW`, `$P_EP[x]` and `$AA_IW[x]`.

Note

During the replacement of an additional transverse axis because of a `GET` request, the status of the diameter/radius programming in the other channel is accepted with `RELEASE[<axis>]`.

Examples

Example 1: Modal axis-specific diameter/radius programming

X is the transverse axis in the channel, axis-specific diameter programming is permitted for Y.

Program code	Comment
N10 G0 X0 Z0 DIAMON	; Channel-specific diameter programming active for X.
N15 DIAMOF	; Channel-specific diameter programming off.
N20 DIAMONA[Y]	; Modal axis-specific diameter programming active for Y.
N25 X200 Y100	; Radius programming active for X.
N30 DIAMCHANA[Y]	; Y accepts the status of the channel-specific diameter/ radius programming and is assigned to this.
N35 X50 Y100	; Radius programming active for X and Y.
N40 DIAMON	; Channel-specific diameter programming on.
N45 X50 Y100	; Diameter programming active for X and Y.

Example 2: Non-modal axis-specific diameter/radius programming

X is the transverse axis in the channel, axis-specific diameter programming is permitted for Y.

Program code	Comment
N10 DIAMON	; Channel-specific diameter programming on.
N15 G0 G90 X20 Y40 DIAMONA[Y]	; Modal axis-specific diameter programming active for Y.
N20 G01 X=RIC(5)	; Dimensions effective in this block for X: Radius in incremental dimensions.
N25 X=RAC(80)	; Dimensions effective in this block for X: Radius in absolute dimensions.
N30 WHEN $SAA_IM[Y]> 50 DO POS[X]=RIC(1)	; X is command axis. Dimensions effective in this block for X: Radius in incremental dimensions.
N40 WHEN $SAA_IM[Y]> 60 DO POS[X]=DAC(10)	; X is command axis. Dimensions effective in this block for X: Radius in absolute dimensions.
N50 G4 F3	

Further information

Diameter values (DIAMONA/DIAM90A)

The diameter values apply for the following data:

- Actual value display of the transverse axis in the workpiece coordinate system
- JOG mode: Increments for incremental dimensions and manual handwheel travel

- Programming of end positions:
  Interpolation parameters I, J, K for G2/G3, if these have been programmed absolutely with AC.
  If I, J, K are programmed incrementally (IC), the radius is always calculated.
- Reading actual values in the workpiece coordinate system for:
  MEAS, MEAW, $P_EP[X], $AA_IW[X]

Non-modal axis-specific diameter programming (DAC, DIC, RAC, RIC)

The statements DAC, DIC, RAC, RIC are permissible for any commands for which channel-specific diameter programming is relevant:

- Axis position: X..., POS, POSA
- Oscillation: OSP1, OSP2, OSS, OSE, POSP
- Interpolation parameters: I, J, K
- Contour definition: Straight line with angle specification
- Rapid retraction: POLF[AX]
- Traversing in the tool direction: MOVT
- Smooth approach and retraction:
  G140 to G143, G147, G148, G247, G248, G347, G348, G340, G341

3.8.4 Position of workpiece for turning

Axis identifiers

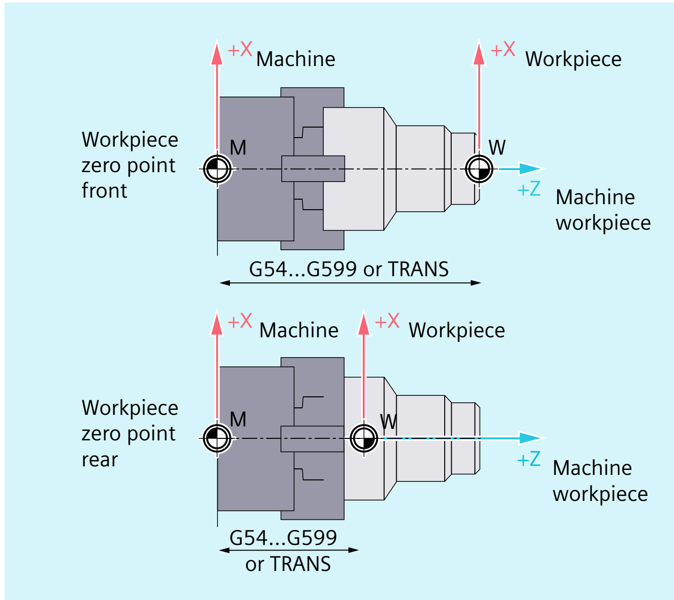
The two geometry axes perpendicular to one another are usually called:

Longitudinal axis	= Z axis (abscissa)
Transverse axis	= X axis (ordinate)

Workpiece zero

Whereas the machine zero is permanently defined, the workpiece zero can be freely selected on the longitudinal axis. Generally the workpiece zero is on the front or rear side of the workpiece.

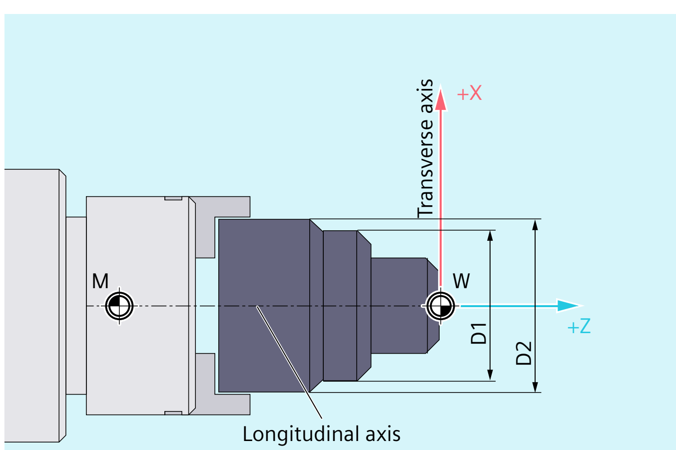
Both the machine and the workpiece zero are on the turning center. The settable offset on the X axis is therefore zero.



M	Machine zero
W	Workpiece zero
`Z`	Longitudinal axis
`X`	Transverse axis
`G54` to `G599` or `TRANS`	Call for the position of the workpiece zero

Transverse axis

Generally the dimensions for the transverse axis are diameter specifications (double path dimension compared to other axes):



The geometry axis that is to serve as transverse axis is defined in the machine data (→ machine manufacturer).

3.9 Motion commands

3.9.1 Introduction and overview

Contour elements

The programmed workpiece contour can be made up of the following contour elements:

- Straight lines
- Circular arcs
- Helical curves (through overlaying of straight lines and circular arcs)

Traversing commands

There are special motion commands for the production of the different contour elements.

Their description can be found in the following chapters:

- Linear interpolation (G1) (Page 194)
- Circular interpolation (Page 196)
- Helical interpolation (G2/G3, TURN) (Page 212)
- Contour definitions (Page 214)
- Thread cutting (Page 224)
- Tapping without compensating chuck (Page 243)
- Tapping with compensating chuck (Page 247)
- Chamfer, rounding (CHF, CHR, RND, RNDM, FRC, FRCM) (Page 248)

For fast positioning of the tool and for moving around the workpiece, the axes are moved in rapid traverse G0 (Page 187).

NOTICE
Tool operation undefined Before starting a machining process, the tool must be pre-positioned in such a way that damage to the tool and workpiece is excluded.

Target positions

A motion block contains the target positions for the axes to be traversed (path axes, synchronized axes, positioning axes).

Note

The axis address may only be programmed once per block.

The target positions can be programmed in Cartesian coordinates or in polar coordinates:

- Travel commands with Cartesian coordinates (G0, G1, G2, G3, X..., Y..., Z...) (Page 179)
- Travel commands with polar coordinates (Page 181)

Starting point - target point

The traversing motion is always for the last point reached to the programmed target position. This target position is then the starting position for the next travel command.

Workpiece contour

The motion blocks produce the workpiece contour when performed in succession:

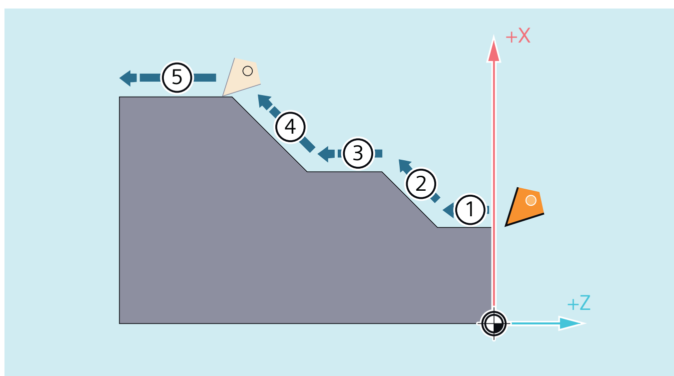


Figure 3-7 Motion blocks for turning

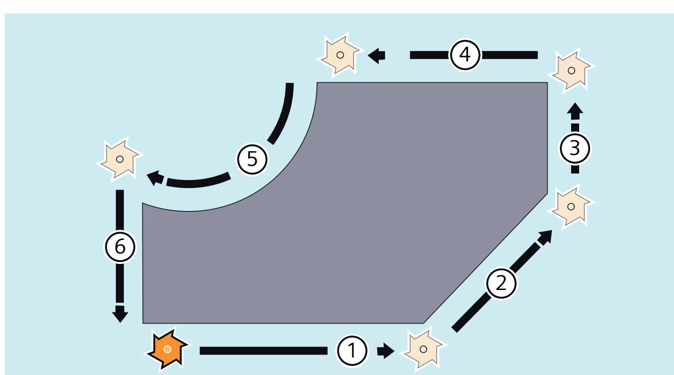


Figure 3-8 Motion blocks for milling

3.9.2 Travel commands with Cartesian coordinates (G0, G1, G2, G3, X..., Y..., Z...)

The position specified in the NC block with Cartesian coordinates can be approached with rapid traverse motion `G0`, linear interpolation `G1` or circular interpolation `G2` /`G3`.

Syntax

```
G0 X... Y... Z...
G1 X... Y... Z...
G2 X... Y... Z... ...
G3 X... Y... Z... ...
```

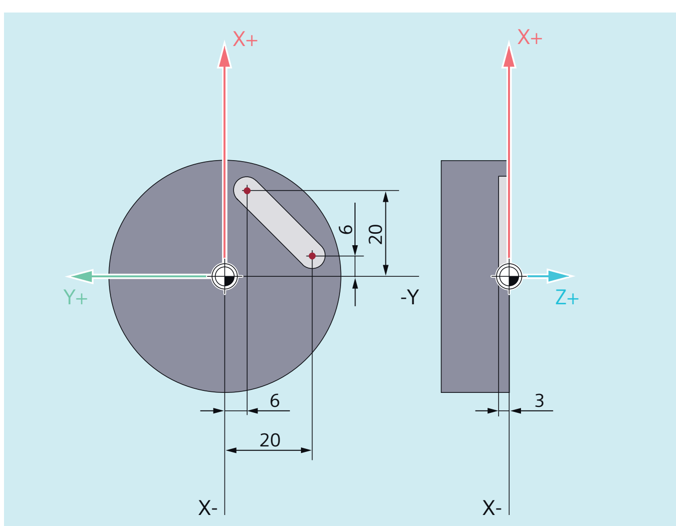
Meaning

`G0:`	Command for the activation of rapid traverse motion
`G1:`	Command for the activation of linear interpolation
`G2:`	Command for the activation of clockwise circular interpolation
`G3:`	Command for the activation of counter-clockwise circular interpolation
`X...:`	Cartesian coordinate of the target position in the X direction
`Y...:`	Cartesian coordinate of the target position in the Y direction
`Z...:`	Cartesian coordinate of the target position in the Z direction

Note

In addition to the coordinates of the target position `X...`, `Y...`, `Z...`, the circular interpolation `G2` / `G3` also requires further data (e.g. the circle center point coordinates; see "Introduction and overview (Page 196)").

Example



Program code	Comment
`N10 G17 S400 M3`	`; Selection of the working plane, spindle clockwise`
`N20 G0 X40 Y-6 Z2`	`; Approach of the starting position specified with Cartesian coordinates in rapid traverse`
`N30 G1 Z-3 F40`	`; Activation of the linear interpolation, feed of the tool`

Program code	Comment
N40 X12 Y-20	; Travel on an inclined line to an end position specified with Cartesian coordinates
N50 G0 Z100 M30	; Retraction in rapid traverse for tool change

3.9.3 Travel commands with polar coordinates

3.9.3.1 Reference point of the polar coordinates (G110, G111, G112)

The point from which the dimensioning starts is called the pole.

The pole can be specified in Cartesian or polar coordinates.

The reference point for the pole coordinates is clearly defined with the `G110` to `G112` commands. Absolute or incremental dimension inputs therefore have no effect.

Syntax

```
G110/G111/G112 X… Y… Z…
G110/G111/G112 AP=… RP=…
```

Meaning

<table>
<tr><td>G110 ...:</td><td colspan="3">With the command G110, the following pole coordinates refer to the last position reached.</td></tr>
<tr><td>G111 ...:</td><td colspan="3">With the command G111, the following pole coordinates refer to the zero point of the current workpiece coordinate system.</td></tr>
<tr><td>G112 ...:</td><td colspan="3">With the command G112, the following pole coordinates refer to the last valid pole.</td></tr>
<tr><td></td><td colspan="3">Note:
The commands G110...G112 must be programmed in a separate NC block.</td></tr>
<tr><td>X… Y… Z…:</td><td colspan="3">Specification of the pole in Cartesian coordinates</td></tr>
<tr><td rowspan="4">AP=… RP=…:</td><td colspan="3">Specification of the pole in polar coordinates</td></tr>
<tr><td rowspan="2">AP=…:</td><td colspan="2">Polar angle
Angle between the polar radius and the horizontal axis of the working plane (e.g. X axis for G17). The positive direction of rotation runs counter-clockwise.</td></tr>
<tr><td>Value range:</td><td>± 0...360°</td></tr>
<tr><td>RP=…:</td><td colspan="2">Polar radius
The specification is always in absolute positive values in [mm] or [inch].</td></tr>
</table>

Note

It is possible to switch block-by-block in the NC program between polar and Cartesian dimensions. It is possible to return directly to the Cartesian system by using Cartesian coordinate identifiers (X..., Y..., Z...). The defined pole is moreover retained up to program end.

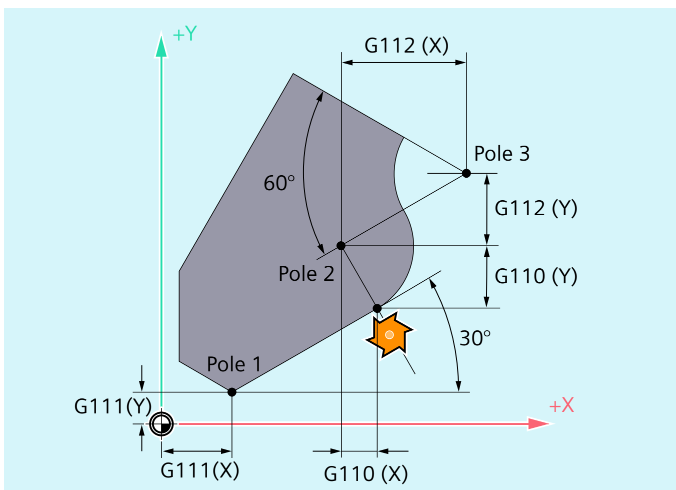
Note

If no pole has been specified, the zero point of the current workpiece coordinate system applies.

Example

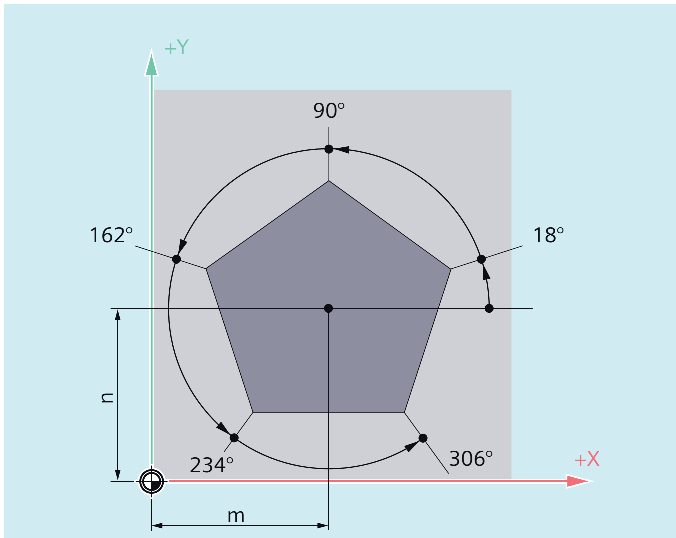
Poles 1 to 3 are defined as follows:

- Pole 1 with `G111 X… Y…`
- Pole 2 with `G110 X… Y…`
- Pole 3 with `G112 X… Y…`



3.9.3.2 Travel commands with polar coordinates (G0, G1, G2, G3, AP, RP)

Travel commands with polar coordinates are useful when the dimensions of a workpiece or part of the workpiece are measured from a central point and the dimensions are specified in angles and radii (e.g. for drilling patterns).



Syntax

```
G0/G1/G2/G3 AP=… RP=…
```

Meaning

`G0:`	Command for the activation of rapid traverse motion	
`G1:`	Command for the activation of linear interpolation	
`G2:`	Command for the activation of clockwise circular interpolation	
`G3:`	Command for the activation of counter-clockwise circular interpolation	
`AP:`	Polar angle Angle between the polar radius and the horizontal axis of the working plane (e.g. X axis for `G17`). The positive direction of rotation runs counter-clockwise.	
	Value range:	± 0...360°
	The angle can be specified either incremental or absolute:	
	`AP=AC(...):`	Absolute dimension input
	`AP=IC(...):`	Incremental dimensions input With incremental dimension input, the last programmed angle applies as reference.
	The polar angle remains stored until a new pole is defined or the working plane is changed.	
`RP:`	Polar radius The specification is **always in absolute positive values** in [mm] or [inch]. The polar radius remains stored until a new value is entered.	

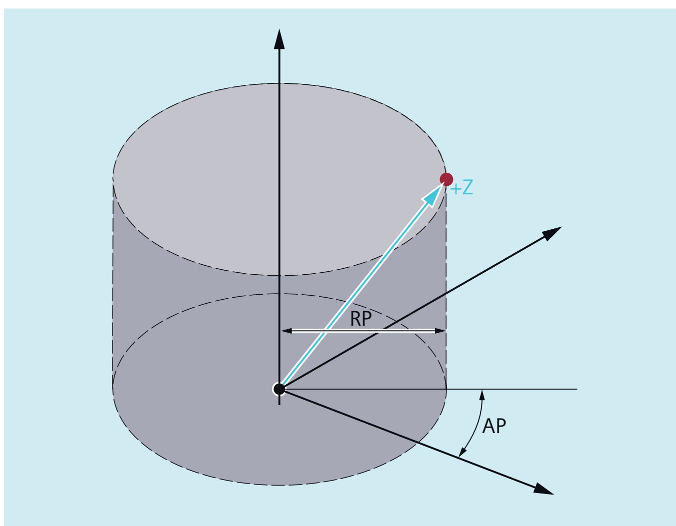
Note

The polar coordinates refer to the pole specified with `G110` ... `G112` and apply in the working plane selected with `G17` to `G19`.

Note

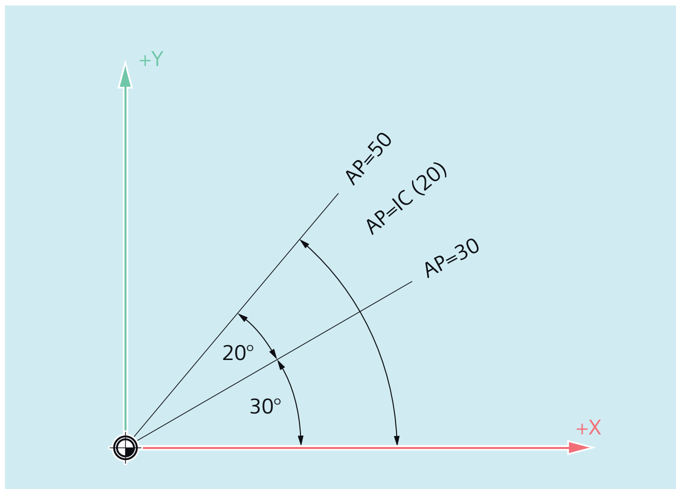
The 3rd geometry axis, which lies perpendicular to the working plane, can also be specified in Cartesian coordinates (see the following diagram). This enables spatial parameters to be programmed in cylindrical coordinates.

Example: `G17 G0 AP… RP… Z…`



Constraints

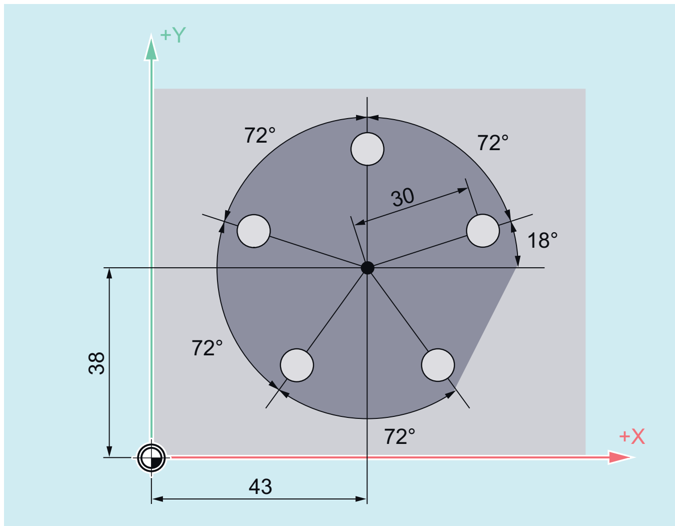
- No Cartesian coordinates such as interpolation parameters, axis addresses, etc. may be programmed for the selected working plane in NC blocks with polar end point coordinates.
- If a pole has not been defined with `G110` ... `G112`, then the zero point of the current workpiece coordinate system is automatically considered as the pole:



- Polar radius RP = 0
  The polar radius is calculated from the distance between the starting point vector in the pole plane and the active pole vector. The calculated polar radius is then saved as modal.
  This applies irrespective of the selected pole definition (`G110` ... `G112`). If both points have been programmed identically, this radius = 0 and alarm 14095 is generated.
- Only polar angle AP has been programmed
  If no polar radius RP has been programmed in the current block, but a polar angle AP, then when there is a difference between the current position and pole in the workpiece coordinates, this difference is used as polar radius and saved as modal. If the difference = 0, then the pole coordinates are specified again and the modal polar radius remains at zero.

Example

Creation of a drilling pattern



The positions of the holes are specified in polar coordinates. Each hole is machined with the same production sequence: Rough-drilling, drilling as dimensioned, reaming ...

The machining sequence is stored in the subprogram.

Program code	Comment
`N10 G17 G54`	`; Working plane X/Y, workpiece zero.`
`N20 G111 X43 Y38`	`; Specification of the pole.`
`N30 G0 RP=30 AP=18 Z5`	`; Approach starting point, specification in cylindrical coordinates.`
`N40 L10`	`; Subprogram call.`
`N50 G91 AP=72`	`; Approach next position in rapid traverse, polar angle in incremental dimensions, polar radius from block N30 remains saved and does not have to be specified.`
`N60 L10`	`; Subprogram call.`
`N70 AP=IC(72)`	`.`
`N80 L10`	…
`N90 AP=IC(72)`	
`N100 L10`	…
`N110 AP=IC(72)`	
`N120 L10`	…
`N130 G0 X300 Y200 Z100 M30`	`; Retract tool, end of program.`

See also

Introduction and overview (Page 196)

3.9.4 Rapid traverse movements

3.9.4.1 Activating rapid traverse (G0)

The traversing of the path axes at rapid traversing velocity is activated with the G command G0.

Syntax

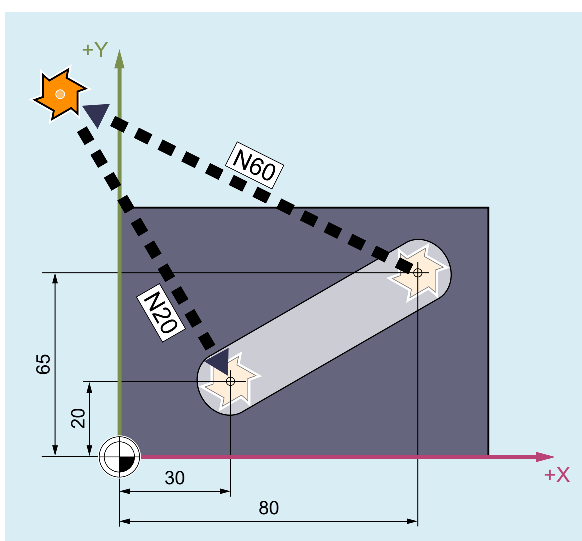
```
G0 X… Y… Z…
G0 RP=… AP=…
```

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	

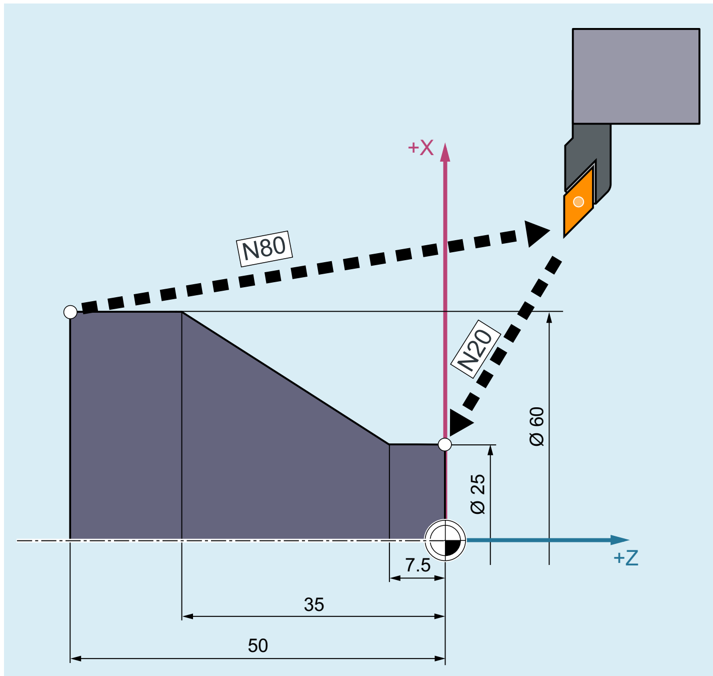
Examples

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

3.9.4.2 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 interpolation.
N50 G41 X20 Y20	; TRC active ⇒ G0 blocks are traversed using linear 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.

3.9.4.3 Adapt tolerances for rapid traverse motion (STOLF, CTOLG0, OTOLG0)

The tolerances for rapid traverse motion (G0 tolerances) configured using machine data can be temporarily adapted in the part program. In so doing, the settings in the machine data are not changed. After channel or end of program reset, the configured tolerances become effective again.

Requirements

G0 tolerances are only effective in compliance with the following conditions:

- One of the following functions is active:
  - Compressor function COMP...
  - 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 tolerances are not effective, as at the transition from non G0 motion to G0 motion (and vice versa), the "**lower tolerance**" always applies (workpiece processing tolerance)!

Syntax

Adaptation of the relative G0 tolerance

```
STOLF=<Value>
```

Adaptation of the absolute G0 tolerances

```
CTOLG0=<Value>
```

```
OTOLG0=<Value>
```

Meaning

STOLF:	Address for programming a temporarily effective tolerance factor for rapid traverse motion			
	<Value>:	G0 tolerance factor		
		Type:	REAL	
		Value:	≥ 0:	The G0 tolerance factor can be 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 motion as for non-rapid traverse motion. Normally, the tolerance factor is set to > 1.0. The programmed G0 tolerance factor remains effective until it is overwritten by renewed STOLF programming, replaced by CTOLG0/OTOLG0 programming or deleted by channel or end of program reset.
			< 0:	the programmed tolerance factor is deleted ⇒ The tolerance value preset in the machine data becomes effective again.
CTOLG0:	Address for programming a temporarily effective contour tolerance factor for rapid traverse motion			
	<Value>:	Absolute value for the contour tolerance		
		Type:	REAL	
		Value:	≥ 0:	The programmed absolute value for the contour tolerance remains effective until it is overwritten by renewed CTOLG0 programming, replaced by STOLF programming or deleted by channel or end of program reset.
			< 0:	the programmed tolerance value is deleted ⇒ The tolerance value preset in the machine data becomes effective again.
OTOLG0:	Address for programming a temporarily effective orientation tolerance factor for rapid traverse motion			
	<Value>:	Absolute value for the orientation tolerance		
		Type:	REAL	
		Value:	≥ 0:	The programmed absolute value for the orientation tolerance remains effective until it is overwritten by renewed OTOLG0 programming, replaced by STOLF programming or deleted by channel or end of program reset.
			< 0:	the programmed tolerance value is deleted ⇒ The tolerance value preset in the machine data becomes effective again.

Note

The last programmed address always has priority, as shown in the following examples:

- When CTOLG0 is programmed with active STOLF, the tolerance value programmed with CTOLG0 is applied to smooth the contour.
- When OTOLG0 is programmed with active STOLF, the tolerance value programmed with OTOLG0 is applied to smooth the orientation.
- After programming STO again, the tolerance factor for the contour and orientation tolerance is applied.

Examples

Example 1: Adaptation of the relative G0 tolerance

Program code	Comment
COMPCAD G645 G1 F10000	; Compressor function COMPCAD
X... Y... Z...	; The machine and setting data apply here.
X... Y... Z...	
X... Y... Z...	
G0 X... Y... Z...	
G0 X... Y... Z...	; Machine data $MC_G0_TOLERANCE_FACTOR (e.g. =3) is effective here, i.e. a smoothing tolerance of: $MC_G0_TOLERANCE_FACTOR * $MA_COMPRESS_POS_TOL
CTOL=0.02	
STOLF=4	
G1 X... Y... Z...	; A contour tolerance of 0.02 mm is applied starting from here.
X... Y... Z...	
X... Y... Z...	
G0 X... Y... Z...	
X... Y... Z...	; From here, a G0 tolerance factor of 4 applies, i.e. a contour tolerance of 0.08 mm.
...	

Example 2: Adaptation of the absolute G0 tolerances

The following absolute G0 tolerances should be preset in the machine data:

- G0 contour tolerance: 0.1
- G0 orientation tolerance: 1.0

These tolerances should be temporarily adapted in the part program:

Program code	Comment
COMPCAD G645 G1 F10000	; Compressor function COMPCAD
X... Y... Z...	; The configured workpiece machining tolerances apply from here.
X... Y... Z...	
X... Y... Z...	
G0 X... Y... Z...	
G0 X... Y... Z...	; The configured absolute G0 tolerances apply here.
CTOLG0=0.2 OTOLG0=2.0	; Programming the absolute G0 tolerances.
G1 X... Y... Z...	
X... Y... Z...	
X... Y... Z...	
G0 X... Y... Z...	
X... Y... Z...	; The programmed G0 tolerances apply from here.
...	

Further information

Reading the G0 tolerance factor

The currently active tolerance factor for rapid traverse motion 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 return the value configured in the machine data.

If no rapid traverse (G0) is active in a block, then these system variables always supply a value of 1.

Reading absolute G0 tolerances

The currently active absolute tolerances for rapid traverse motion can be read via system variables.

- In synchronized actions or with preprocessing stop in the part program via the system variables:

$AC_CTOL_G0_ABS	Active contour tolerance for G0 motion G0 contour tolerance that was active when the current main run block was preprocessed.
$AC_OTOL_ G0_ABS	Active orientation tolerance for G0 motion G0 orientation tolerance that was active when the current main run block was preprocessed.

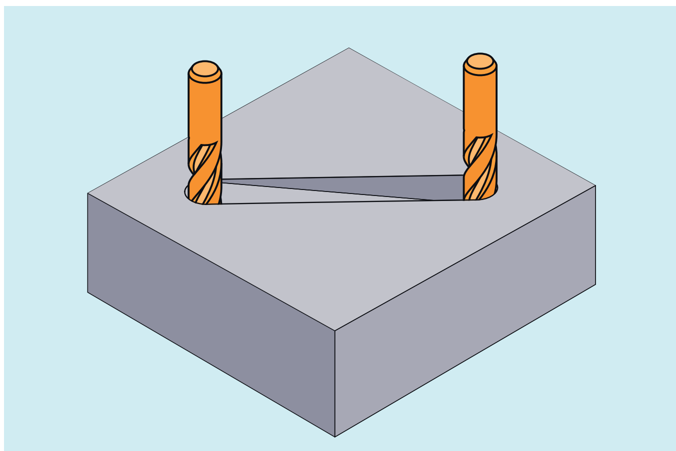
- Without preprocessing stop in the part program via system variables:

$P_CTOL_ G0_ABS	Programmed contour tolerance for G0 motion
$P_OTOL_ G0_ABS	Programmed orientation tolerance for G0 motion

If no absolute G0 tolerances with CTOLG0 and OTOLG0 are programmed in the active part program, then these system variables supply the values configured in the machine data.

3.9.5 Linear interpolation (G1)

With G1 the tool travels on paraxial, inclined or straight lines arbitrarily positioned in space. Linear interpolation permits machining of 3D surfaces, grooves, etc.



Syntax

```
G1 X… Y… Z … F…
G1 AP=… RP=… F…
```

Meaning

G1:	Linear interpolation with feedrate (linear interpolation)
X... Y... Z...:	End point in Cartesian coordinates
AP=...:	End point in polar coordinates, in this case polar angle
RP=...:	End point in polar coordinates, in this case polar radius
F...:	Feedrate speed in mm/min. The tool travels at feedrate F along a straight line from the current starting point to the programmed destination point. You can enter the destination point in Cartesian or polar coordinates. The workpiece is machined along this path. Example: G1 G94 X100 Y20 Z30 A40 F100 The end point on X, Y, Z is approached at a feedrate of 100 mm/min; the rotary axis A is traversed as a synchronized axis, ensuring that all four movements are completed at the same time.

Note

G1 is modal.

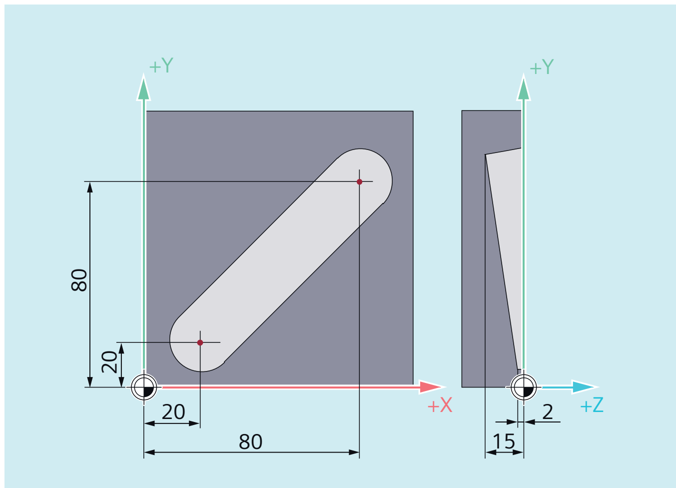
Spindle speed S and spindle direction M3/M4 must be specified for the machining.

Axis groups, for which path feedrate F applies, can be defined with FGROUP. You will find more information in the "Path behavior" section.

Examples

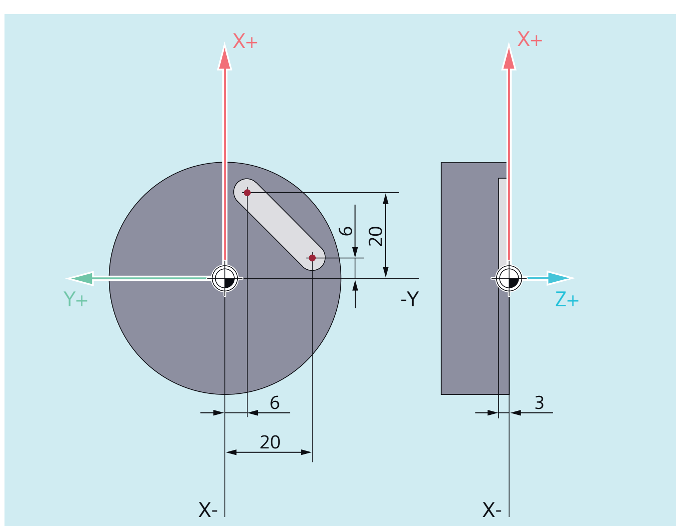
Example 1: Machining of a groove (milling)

The tool travels from the starting point to the end point in the X/Y direction. Infeed takes place simultaneously in the Z direction.



Program code	Comment
N10 G17 S400 M3	; Selection of the working plane, spindle clockwise
N20 G0 X20 Y20 Z2	; Approach the starting position
N30 G1 Z-2 F40	; Tool infeed
N40 X80 Y80 Z-15	; Travel on an inclined line
N50 G0 Z100 M30	; Retraction for tool change

Example 2: Machining of a groove (turning)



Program code	Comment
N10 G17 S400 M3	; Selection of the working plane, spindle clockwise
N20 G0 X40 Y-6 Z2	; Approach the starting position
N30 G1 Z-3 F40	; Tool infeed
N40 X12 Y-20	; Travel on an inclined line
N50 G0 Z100 M30	; Retraction for tool change

3.9.6 Circular interpolation

3.9.6.1 Introduction and overview

Circular interpolation enables the machining of full circles or arcs.

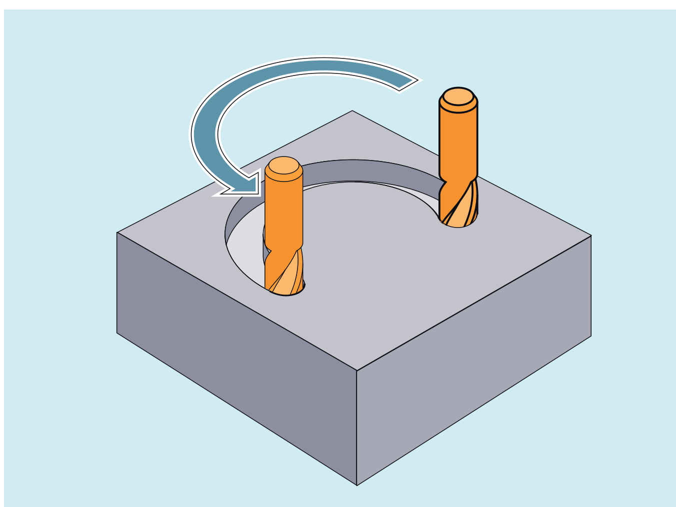


Figure 3-9 Application example: Milling a circular way

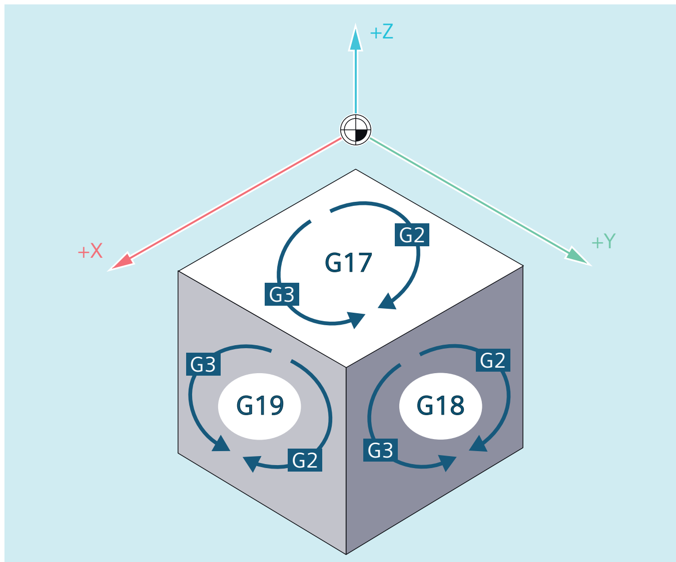
Programming options

The control system offers various options of programming circular movements. This allows the user to implement almost any type of drawing dimension directly.

- Circular interpolation with center point and end point (G2/G3, X... Y... Z..., I... J... K...) (Page 197)
- Circular interpolation with radius and end point (G2/G3, X... Y... Z..., CR) (Page 199)
- Circular interpolation with opening angle and end point / center point (G2/G3, X... Y... Z... / I... J... K..., AR) (Page 201)
- Circular interpolation with polar coordinates (G2/G3, AP, RP) (Page 203)
- Circular interpolation with intermediate point and end point (CIP, X... Y... Z..., I1... J1... K1...) (Page 205)
- Circular interpolation with tangential transition (CT, X... Y... Z...) (Page 208)

Plane for the circular interpolation

The control needs the working plane parameter (Page 155) to calculate the direction of rotation for the circle (G2 is clockwise or G3 is counter-clockwise).



Exception:

It is also possible to create circles outside the selected working plane (not if the opening angle is specified). In this case, the axis identifiers that the programmer specifies as circle end point determine the circle plane.

3.9.6.2 Circular interpolation with center point and end point (G2/G3, X... Y... Z..., I... J... K...)

Circular interpolation version, that uses the **center point** and **end point** of a circular contour element for the interpolation.

If the circle is programmed without an end point, the result is a full circle.

Syntax

```
G2/G3 X… Y… Z… I… J… K…
G2/G3 X… Y… Z… I=AC(…) J=AC(…) K=(AC…)
```

Meaning

G2:	Circular interpolation clockwise	
	Effective:	Modal
G3:	Circular interpolation counter-clockwise	
	Effective:	Modal

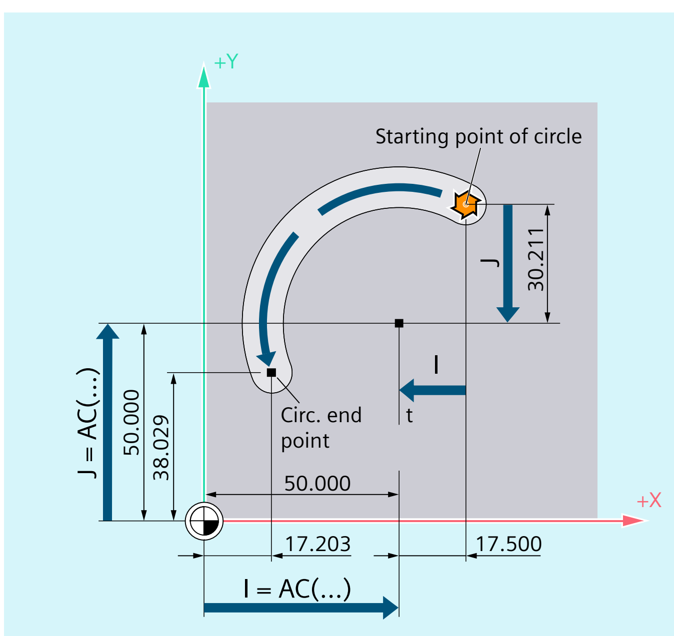
`X... Y... Z...:`	Circle end point in Cartesian coordinates. Depending on the currently valid dimensional notation setting `G90`/`G91` or `...=AC(...)` / `...=IC(...)`, the circle end point coordinates are interpreted either in the absolute dimension or in the incremental dimension.
`I... J... K...:`	Interpolation parameters to state the circle center point coordinates in the directions X, Y, Z Per default, the circle center point coordinates are stated in the incremental dimension in relation to the circle starting point. If the circle center point coordinates are stated in the absolute dimension in relation to workpiece zero, the interpolation parameters I, J, K must be programmed as follows: `I=AC(…) J=AC(…) K=AC(…)` **Note** An interpolation parameter with value 0 can be omitted, but the associated second parameter must always be specified.

Note

The default setting `G90`/`G91` absolute or incremental dimensions is only valid for the circle end point.

Examples

Example 1: Milling



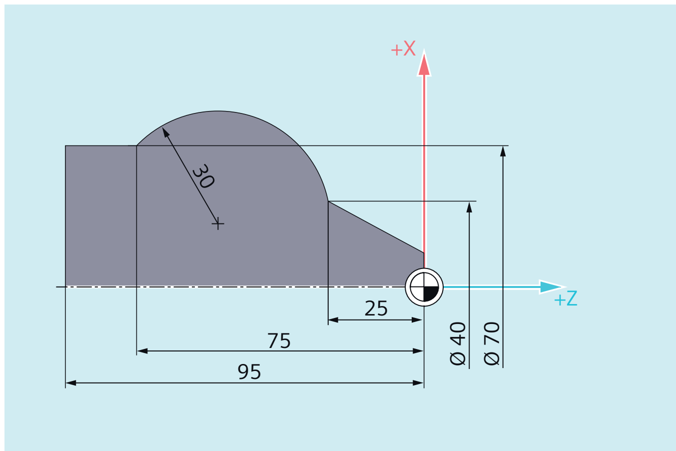
Center point data using incremental dimensions

```
N10 G0 X67.5 Y80.211
N20 G3 X17.203 Y38.029 I-17.5 J-30.211 F500
```

Center point data using absolute dimensions

```
N10 G0 X67.5 Y80.211
N20 G3 X17.203 Y38.029 I=AC(50) J=AC(50)
```

Example 2: Turning



Center point data using incremental dimensions

```
N120 G0 X12 Z0
N125 G1 X40 Z-25 F0.2
N130 G3 X70 Z-75 I-3.335 K-29.25
N135 G1 Z-95
```

Center point data using absolute dimensions

```
N120 G0 X12 Z0
N125 G1 X40 Z-25 F0.2
N130 G3 X70 Z-75 I=AC(33.33) K=AC(-54.25)
N135 G1 Z-95
```

3.9.6.3 Circular interpolation with radius and end point (G2/G3, X... Y... Z..., CR)

Circular interpolation version, that uses the **radius** and **end point** of a circular contour element for the interpolation.

Note

Full circles (traversing angle 360 °) can **not** be programmed with this version.

Syntax

```
G2/G3 X… Y… Z… CR=±...
```

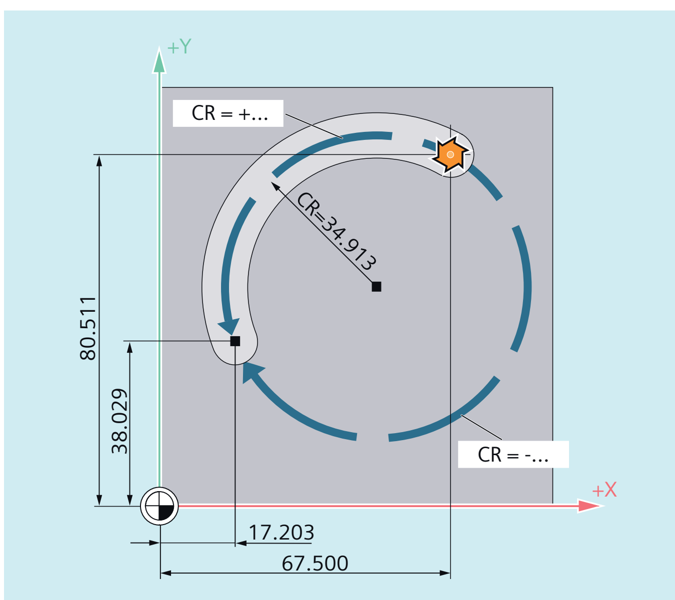
Meaning

G2:	Circular interpolation clockwise	
	Effective:	Modal

`G3:`	Circular interpolation counter-clockwise	
	Effective:	Modal
`X... Y... Z...:`	Circle end point in Cartesian coordinates. Depending on the currently valid dimensional notation setting `G90`/`G91` or `...=AC(...)` / `...=IC(...)`, the end point coordinates are interpreted either in the absolute dimension or in the incremental dimension.	
`CR=±...:`	Circle radius The sign indicates whether the traversing angle is to be greater than or less than 180°. A positive sign can be omitted.	
	`CR=+...:`	Traversing angle ≤ 180°
	`CR=-...:`	Traversing angle > 180°
	Note There is no practical limitation on the maximum size of the programmable radius.	

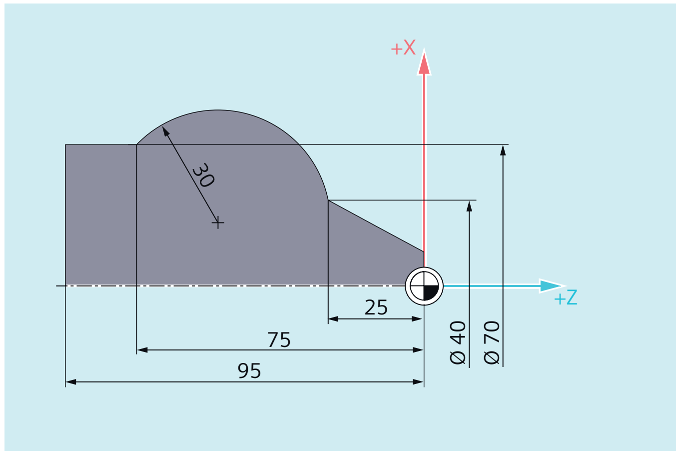
Examples

Example 1: Milling



Program code
`N10 G0 X67.5 Y80.511`
`N20 G3 X17.203 Y38.029 CR=34.913 F500`
`...`

Example 2: Turning



Program code
...
N125 G1 X40 Z-25 F0.2
N130 G3 X70 Z-75 CR=30
N135 G1 Z-95
...

3.9.6.4 Circular interpolation with opening angle and end point / center point (G2/G3, X... Y... Z... / I... J... K..., AR)

Circular interpolation version, that uses the **opening angle** and **center point** or **end point** of a circular contour element for the interpolation.

Note

Full circles (traversing angle 360 °) can **not** be programmed with this version.

Syntax

```
G2/G3 X… Y… Z… AR=...
G2/G3 I… J… K… AR=...
```

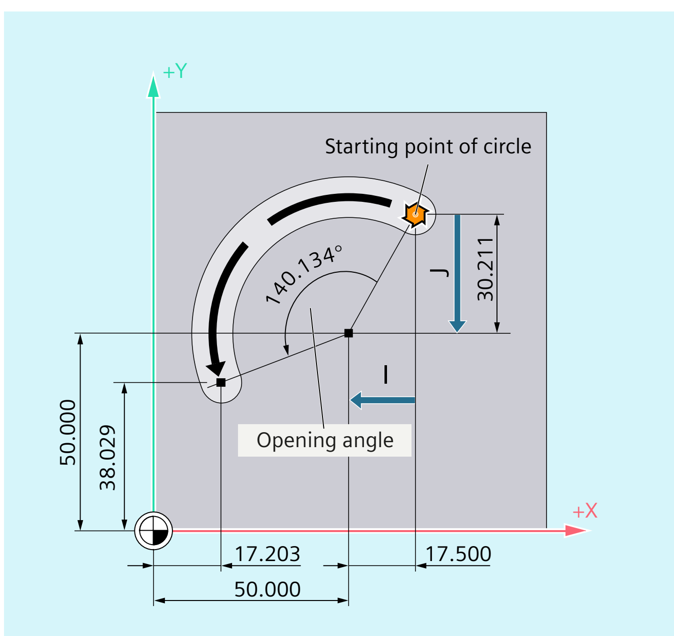
Meaning

G2:	Circular interpolation clockwise	
	Effective:	Modal
G3:	Circular interpolation counter-clockwise	
	Effective:	Modal

`X... Y... Z...:`	Circle end point in Cartesian coordinates. Depending on the currently valid dimensional notation setting `G90`/`G91` or `...=AC(...)` / `...=IC(...)`, the circle end point coordinates are interpreted either in the absolute dimension or in the incremental dimension.	
`I... J... K...:`	Interpolation parameters to state the circle center point coordinates in the directions X, Y, Z Per default, the circle center point coordinates are stated in the incremental dimension in relation to the circle starting point. If the circle center point coordinates are stated in the absolute dimension in relation to workpiece zero, the interpolation parameters I, J, K must be programmed as follows: `I=AC(…) J=AC(…) K=AC(…)` **Note** An interpolation parameter with value 0 can be omitted, but the associated second parameter must always be specified.	
`AR=...:`	Opening angle	
	Range of values:	0° ... 360°

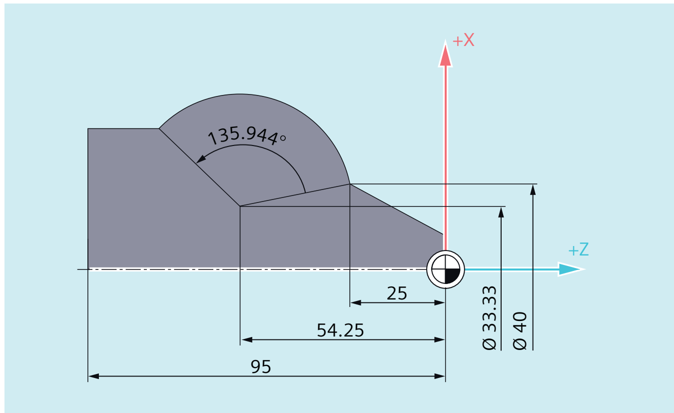
Examples

Example 1: Milling




```
Program code
N10 G0 X67.5 Y80.211
N20 G3 X17.203 Y38.029 AR=140.134 F500
N20 G3 I-17.5 J-30.211 AR=140.134 F500
```

Example 2: Turning



Program code
N125 G1 X40 Z-25 F0.2
N130 G3 X70 Z-75 AR=135.944
N130 G3 I-3.335 K-29.25 AR=135.944
N130 G3 I=AC(33.33) K=AC(-54.25) AR=135.944
N135 G1 Z-95

3.9.6.5 Circular interpolation with polar coordinates (G2/G3, AP, RP)

Circular interpolation version, that uses the **circle end point in polar coordinates** for the interpolation.

The following rule applies:

- The pole lies at the circle center.
- The polar radius corresponds to the circle radius.

Syntax

```
G2/G3 absolute pressure=... Recipe procedure=...
```

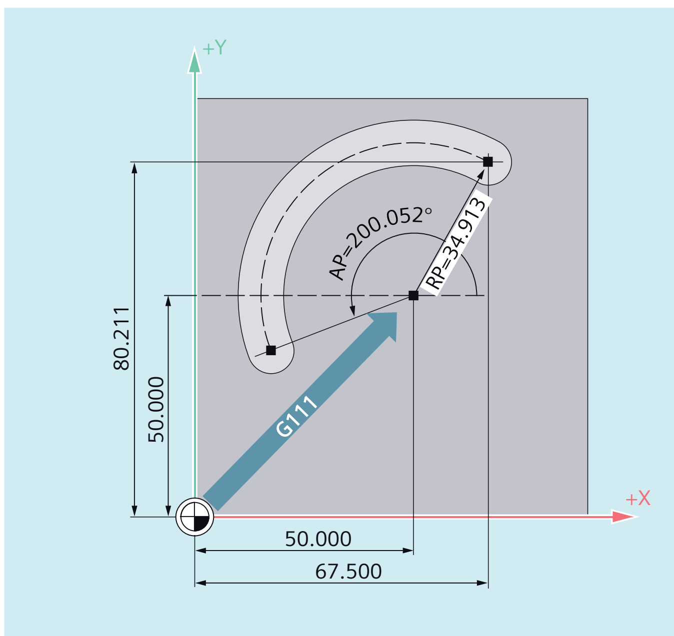
Meaning

G2:	Circular interpolation clockwise	
	Effective:	Modal
G3:	Circular interpolation counter-clockwise	
	Effective:	Modal

`Absolute pressure=... Recipe procedure=...:`	Circle end point in polar coordinates.	
	`Absolute pressure=...:`	Polar angle
	`Recipe procedure=...:`	Polar radius (≙ circle radius)

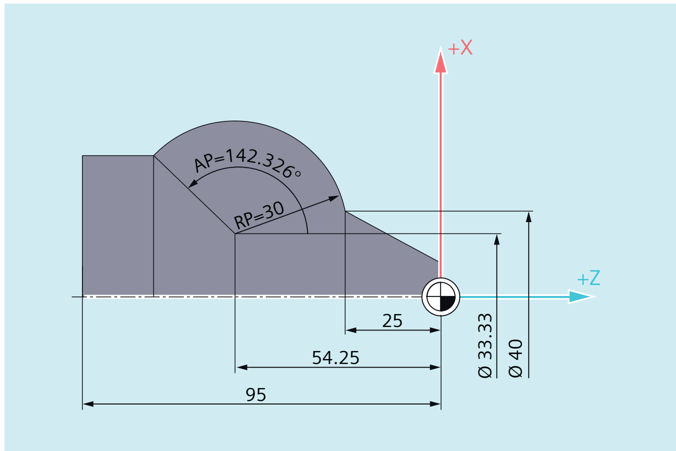
Examples

Example 1: Milling



Program code
`N10 G0 X67.5 Y80.211`
`N20 G111 X50 Y50`
`N30 G3 RP=34.913 AP=200.052 F500`

Example 2: Turning



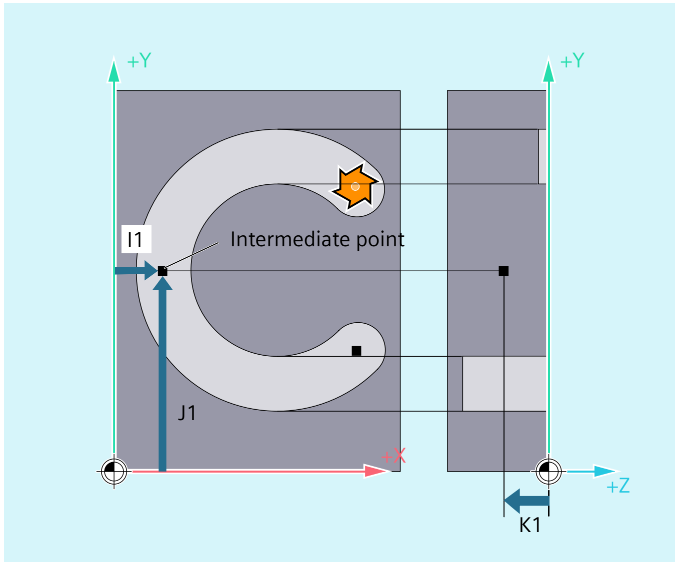
Program code
N125 G1 X40 Z-25 F0.2
N130 G111 X33.33 Z-54.25
N135 G3 RP=30 AP=142.326
N140 G1 Z-95

3.9.6.6 Circular interpolation with intermediate point and end point (CIP, X... Y... Z..., I1... J1... K1...)

The circular interpolation version programmed with the G command `CIP` allows the interpolation of arcs lying at an incline in the space.

The circular motion is described by the **intermediate point** and the **end point** of the circular contour.

The traversing direction is determined by the order of the starting point → intermediate point → end point.



Syntax

```
CIP X… Y… Z… I1=AC(…) J1=AC(…) K1=(AC…)
```

Meaning

`CIP`:	Circular interpolation through intermediate point	
	Effective:	Modal
`X... Y... Z...:`	Circle end point in Cartesian coordinates. Depending on the currently valid dimensional notation setting `G90`/`G91` or `...=AC(...)` / `...=IC(...)`, the circle end point coordinates are interpreted either in the absolute dimension or in the incremental dimension.	
`I1... J1... K1...:`	Interpolation parameters to state the circle intermediate point coordinates in the directions X, Y, Z Depending on the currently valid dimensional notation setting `G90`/`G91` or `...=AC(...)` / `...=IC(...)`, the circle intermediate point coordinates are interpreted either in the absolute dimension or in the incremental dimension. **Note** An interpolation parameter with value 0 can be omitted, but the associated second parameter must always be specified.	

Note

The default settings `G90`/`G91` (absolute or incremental dimensions) are only valid for the circle intermediate point and the circle end point.

With incremental dimensions `G91` or `...=IC(...)` active, the circle starting point is used as the reference for the intermediate point and the end point.

Note

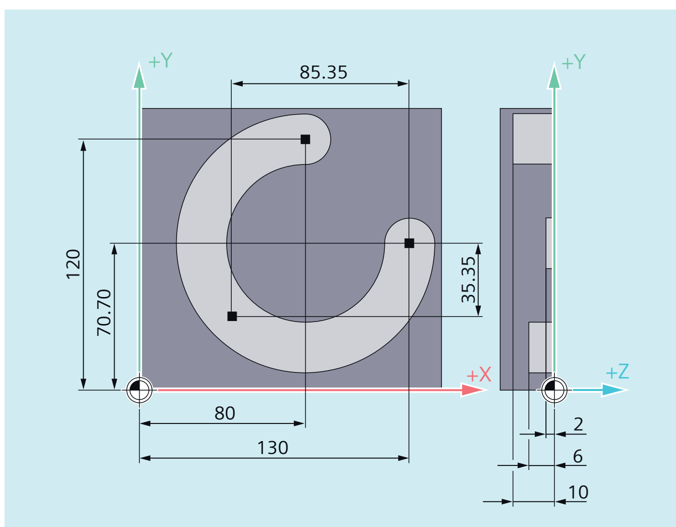
Turning technology

The diameter programming of the interpolation parameter for the transverse axis is not supported with `CIP` in the circular-path programming. The interpolation parameter for the transverse axis must therefore be programmed in the **radius**.

Examples

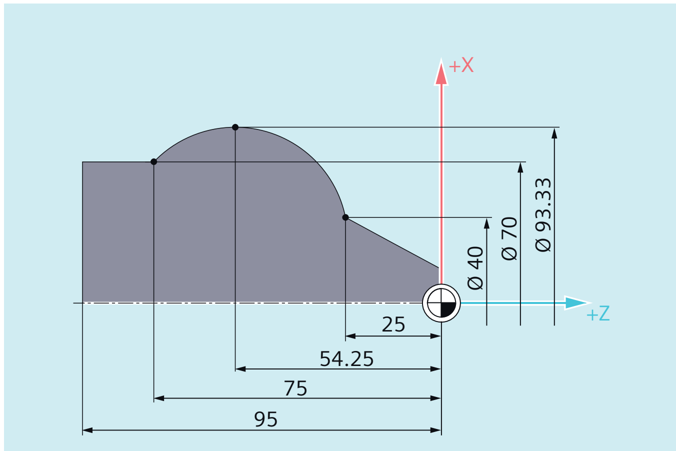
Example 1: Milling

In order to machine an inclined circular groove, a circle is described by specifying the intermediate point with three interpolation parameters, and the end point with three coordinates.



Program code	Comment
`N10 G0 G90 X130 Y70.70 S800 M3`	; Approach starting point.
`N20 G17 G1 Z-2 F100`	; Feed of the tool.
`N30 CIP X80 Y120 Z-10 I1=IC(-85.35) J1=IC(-35.35) K1=-6`	; Circle end point and intermediate point.
	; Coordinates for all three geometry axes.
`N40 M30`	; End of program

Example 2: Turning



Program code	Comment
...	
N125 G1 G90 X40 Z-25 F0.2	
N130 CIP X70 Z-75 **I1=IC(26.665)** K1=IC(-29.25)	; Interpolation parameter I1 for transverse axis must be programmed in the radius.
; or	
; N130 CIP X70 Z-75 **I1=46.665** K1=-54.25	
N135 G1 Z-95	

3.9.6.7 Circular interpolation with tangential transition (CT, X... Y... Z...)

The circular interpolation version programmed with the G command `CT` allows the interpolation of arcs that connect tangentially to the previously programmed contour element.

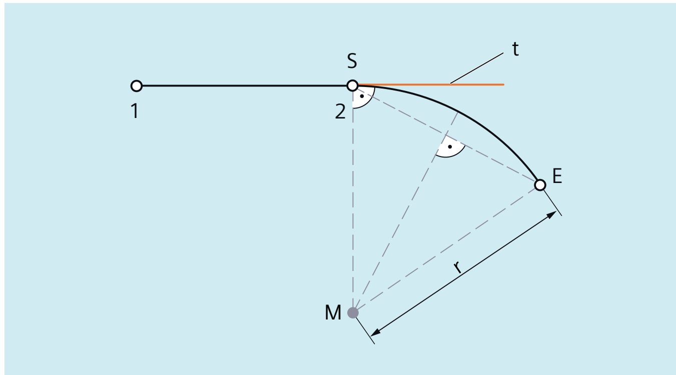
The circle is defined by the **start and end points**, and the **tangent direction at the start point**.

Note

Tangent direction at the start point.

The tangent direction in the starting point of a CT block is determined from the end tangent of the programmed contour of the last block with a traversing motion.

There can be any number of blocks without traversing information between this block and the current block.



S Start point

E End point

M Center of circle

r Circle radius

t End tangents of the programmed contour of the last block with a traversing movement.

Figure 3-10 Tangentially to the straight section 1-2 connecting circular path S-E

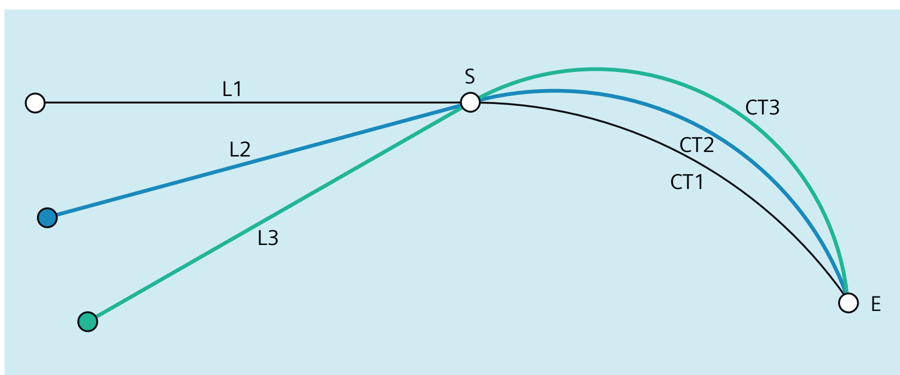


Figure 3-11 Tangentially connecting circular paths depend on the previous contour element

Syntax

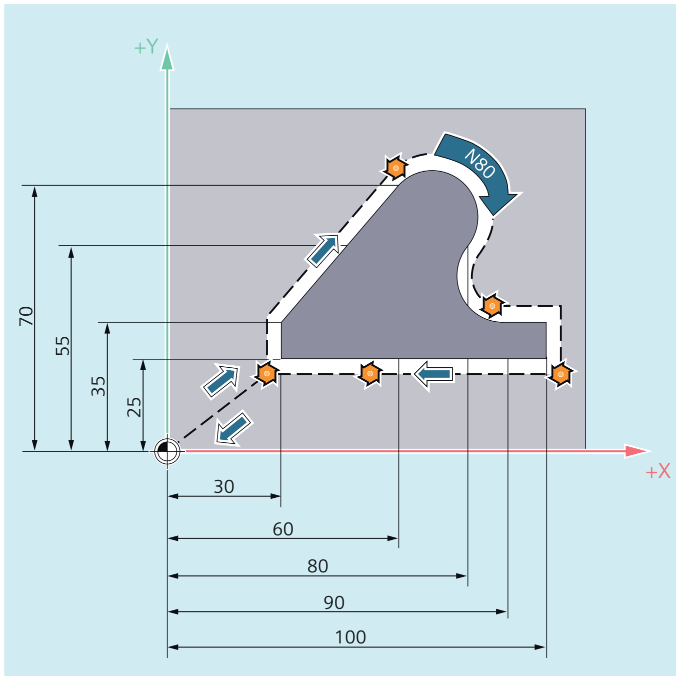
```
CT X… Y… Z…
```

Meaning

`CT:`	Circular interpolation with tangential transition	
	Effective:	Modal
`X... Y... Z...:`	Circle end point in Cartesian coordinates. Depending on the currently valid dimensional notation setting `G90`/`G91` or `...=AC(...)` / `...=IC(...)`, the circle end point coordinates are interpreted either in the absolute dimension or in the incremental dimension.	

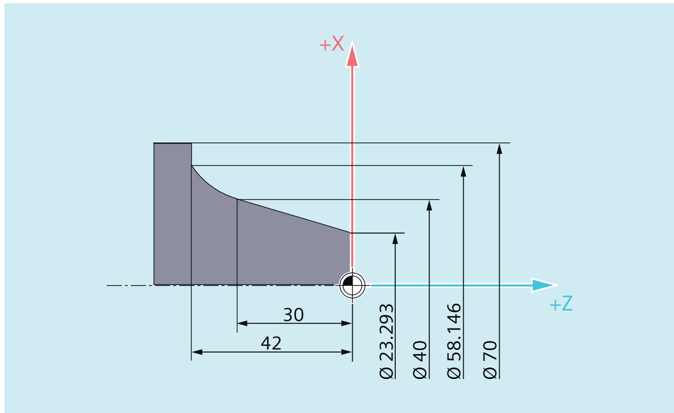
Examples

Example 1: Milling



Program code	Comment
N10 G0 Z100	
N20 G17 T1 M6	
N30 G0 X0 Y0 Z2 M3 S300 D1	
N40 Z-5 F1000	; Feed in tool.
N50 G41 X30 Y25 G1 F1000	; Switch on tool radius compensation.
N60 Y35	; Mill contour.
N70 X60 Y70	
N80 CT X80 Y55	**; Circular-path programming with tangential transition.**
N90 X90 Y35	
N100 G1 X100	
N110 Y25	
N120 X30	
N130 G0 G40 X0 Y0	; Switch off tool radius compensation.
N140 Z100	; Retract tool.
N140 M30	

Example 2: Turning



Program code	Comment
...	
N110 G1 X23.293 Z0 F10	
N115 X40 Z-30 F0.2	
N120 CT X58.146 Z-42	**; Circular-path programming with tangential transition.**
N125 G1 X70	
...	

Further information

Splines

In the case of splines, the tangential direction is defined by the straight line through the last two points. In the case of A and C splines with active ENAT or EAUTO, this direction is generally not the same as the direction at the end point of the spline.

The transition of B splines is always tangential, the tangent direction is defined as for A or C splines and active ETAN.

Frame change

If a frame change takes place between the block that defines the tangent and the CT block, the tangent is also subjected to this change.

Limit case

If the extension of the start tangent runs through the end point, a straight line is produced instead of a circle (limit case: circle with infinite radius). In this special case, TURN must either not be programmed or the value must be TURN=0.

Note

When the values tend towards this limit case, circles with an unlimited radius are produced and machining with TURN unequal to 0 is generally aborted with an alarm due to violation of the software limits.

Position of the circle plane

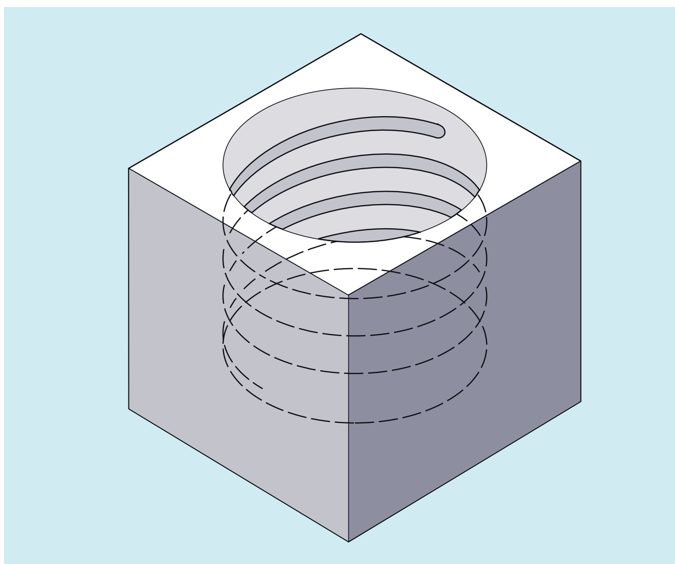
The position of the circle plane depends on the active plane (G17-G19).

If the tangent of the previous block does not lie in the active plane, its projection into the active plane is used.

If the start and end points do not have the same position components perpendicular to the active plane, a helix is produced instead of a circle.

3.9.7 Helical interpolation (G2/G3, TURN)

The helical interpolation enables, for example, the production of threads or oil grooves.



With helical interpolation, two motions are superimposed and executed in parallel:

- A plane circular motion on which
- A vertical linear motion is superimposed.

Syntax

```
G2/G3 X… Y… Z… I… J… K… TURN=
G2/G3 X… Y… Z… I… J… K… TURN=
G2/G3 AR=… I… J… K… TURN=
G2/G3 AR=… X… Y… Z… TURN=
G2/G3 AP… RP=… TURN=
```

Meaning

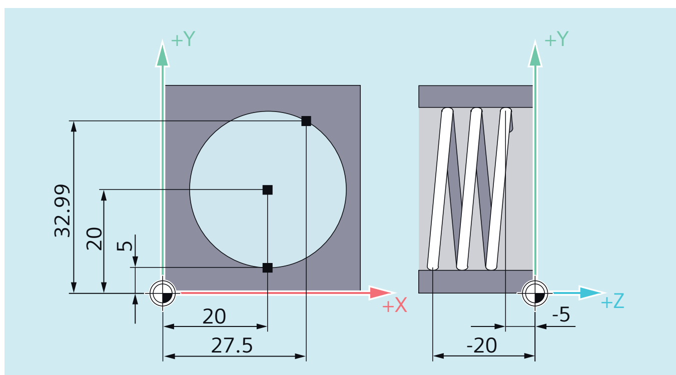
G2:	Travel on a circular path in clockwise direction
G3:	Travel on a circular path in counter-clockwise direction
X Y Z :	End point in Cartesian coordinates

`I J K`:	Circle center point in Cartesian coordinates
`AR`:	Opening angle
`TURN=`:	Number of additional circular passes in the range from 0 to 999
`AP=`:	Polar angle
`RP=`:	Polar radius

Note

G2 and G3 are modal.

Example



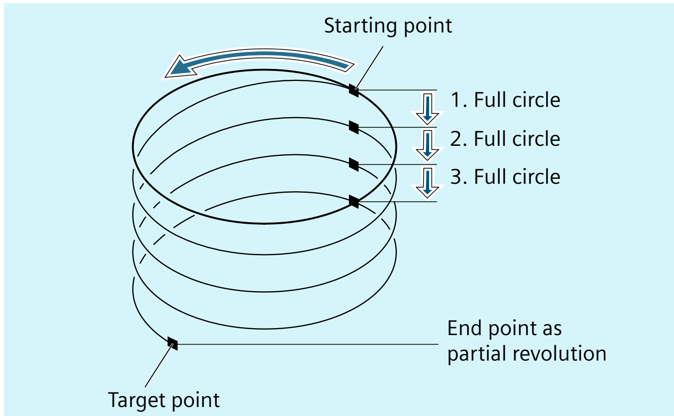
Program code	Comment
`N10 G17 G0 X27.5 Y32.99 Z3`	; Approach the starting position.
`N20 G1 Z-5 F50`	; Feed of the tool.
`N30 G3 X20 Y5 Z-20 I=AC(20) J=AC(20) TURN=2`	; Helix with the specifications: Execute two full circles after the starting position, then travel to end point.
`N40 M30`	; End of program

Additional information

Motion sequence

1. Approach starting point
2. Execute the full circles programmed with `TURN=`.
3. Approach circle end position, e.g. as part rotation.
4. Execute steps 2 and 3 across the infeed depth.

The pitch, with which the helix is to be machined is calculated from the number of full circles plus the programmed circle end position (executed across the infeed depth).



Programming the end point for helical interpolation

Please refer to circular interpolation for a detailed description of the interpolation parameters.

Programmed feedrate

For helical interpolation, it is advisable to specify a programmed feedrate override (`CFC`). `FGROUP` can be used to specify which axes are to be traversed with a programmed feedrate. For more information please refer to the Path behavior section.

3.9.8 Contour definitions

3.9.8.1 Contour definition programming

Function

The contour definition programming is used for the quick input of simple contours.

Programmable are contour definitions with one, two, three or more points with the transition elements chamfer or rounding, through specification of Cartesian coordinates and/or angles (ANG or ANG1 and ANG2).

Additional arbitrary NC addresses can be used, e.g. address letters for further axes (single axes or axis perpendicular to the machining plane), auxiliary function specifications, G commands, velocities, etc. in the blocks that describe contour definitions.

Note

Contour calculator

The contour definitions can be programmed easily with the aid of the contour calculator. This is a user interface tool that enables the programming and graphic display of simple and complex workpiece contours. The contours programmed using the contour calculator are transferred to the part program.

Further information: Operating Manual

Parameterizing

The identifiers for angle, radius and chamfer are defined via machine data:

MD10652 $MN_CONTOUR_DEF_ANGLE_NAME (name of the angle for contour definitions)

MD10654 $MN_RADIUS_NAME (name of the radius for contour definitions)

MD10656 $MN_CHAMFER_NAME (name of the chamfer for contour definitions)

Note

See the machine manufacturer's specifications.

3.9.8.2 Contour definitions: One straight line

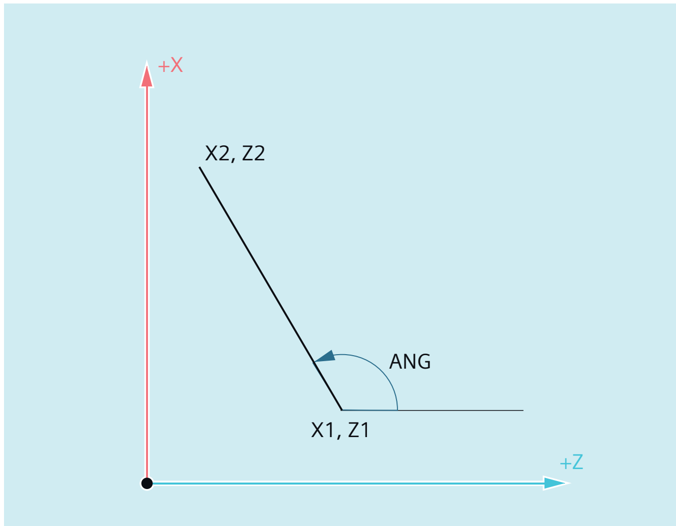
Note

In the following description it is assumed that:

- G18 is active (⇒ active working plane is the Z/X plane).
  (However, the programming of contour definitions is also possible without restrictions with G17 or G19.)
- The following identifiers have been defined for angle, radius and chamfer:
  - ANG (angle)
  - RND (radius)
  - CHR (chamfer)

The end point of the straight line is defined by the following specifications:

- Angle ANG
- **One** Cartesian end point coordinate (X2 or Z2)



ANG:	Angle of the straight line
X1, Z1:	Start coordinates
X2, Z2:	End point coordinates of the straight line

Syntax

```
X… ANG=…
Z… ANG=…
```

Meaning

`X...:`	End point coordinate in the X direction
`Z...:`	End point coordinate in the Z direction
`ANG:`	Identifier for angle programming The specified value (angle) refers to the abscissa of the active working plane (Z axis with `G18`).

Example

Program code	Comment
`N10 X5 Z70 F1000 G18`	`; Approach the starting position`
`N20 X88.8 ANG=110`	`; Straight line with angle specification`
`N30 ...`	

or

Program code	Comment
`N10 X5 Z70 F1000 G18`	`; Approach the starting position`
`N20 Z39.5 ANG=110`	`; Straight line with angle specification`
`N30 ...`	

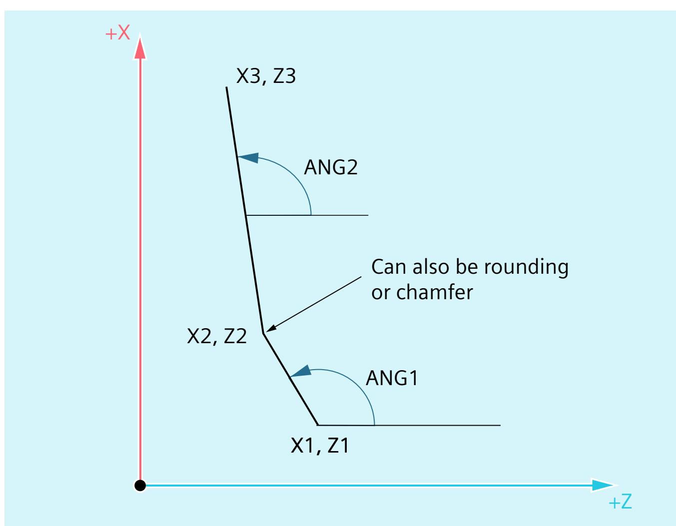
3.9.8.3 Contour definitions: Two straight lines

Note

In the following description it is assumed that:

- G18 is active (⇒ active working plane is the Z/X plane).
  (However, the programming of contour definitions is also possible without restrictions with G17 or G19.)
- The following identifiers have been defined for angle, radius and chamfer:
  - ANG (angle)
  - RND (radius)
  - CHR (chamfer)

The end point of the first straight line can be programmed by specifying the Cartesian coordinates or by specifying the angle of the two straight lines. The end point of the second straight line must always be programmed with Cartesian coordinates. The intersection of the two straight lines can be designed as a corner, curve or chamfer.



ANG1:	Angle of the first straight line
ANG2:	Angle of the second straight line
X1, Z1:	Start coordinates of the first straight line
X2, Z2:	End point coordinates of the first straight line or start coordinates of the second straight line
X3, Z3:	End point coordinates of the second straight line

Syntax

Programming of the end point of the first straight line by specifying the angle

- Corner as transition between the straight lines:

```
ANG=…
X… Z… ANG=…
```

- Rounding as transition between the straight lines:

```
ANG=… RND=...
X… Z… ANG=…
```

- Chamfer as transition between the straight lines:

```
ANG=… CHR=...
X… Z… ANG=…
```

Programming of the end point of the first straight line by specifying the coordinates

- Corner as transition between the straight lines:

```
X… Z…
X… Z…
```

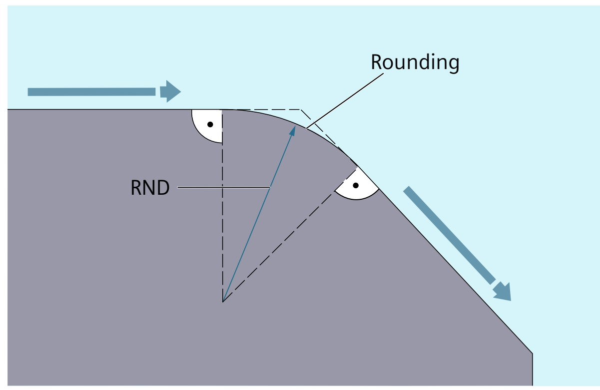
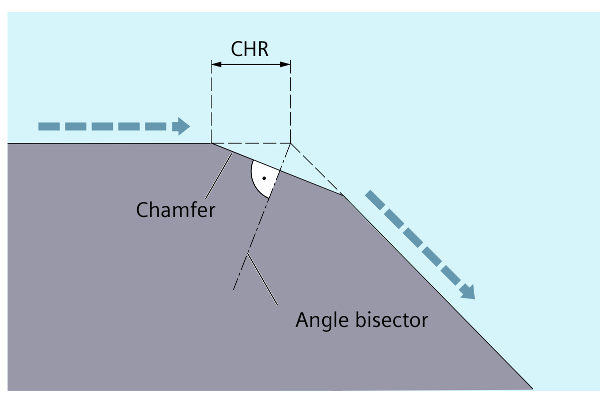
- Rounding as transition between the straight lines:

```
X… Z… RND=...
X… Z…
```

- Chamfer as transition between the straight lines:

```
X… Z… CHR=...
X… Z…
```

Meaning

`ANG=...:`	Identifier for angle programming The specified value (angle) refers to the abscissa of the active working plane (Z axis with `G18`).
`RND=...:`	Identifier for programming a rounding The specified value corresponds to the radius of the rounding: 
`CHR=...:`	Identifier for programming a chamfer The specified value corresponds to the width of the chamfer in the direction of motion: 
`X...:`	Coordinates in the X direction
`Z...:`	Coordinates in the Z direction

Note

For further information on the programming of a chamfer or rounding, see "Chamfer, rounding (CHF, CHR, RND, RNDM, FRC, FRCM) (Page 248)".

Example

Program code	Comment
`N10 X10 Z80 F1000 G18`	`; Approach the starting position.`
`N20 ANG=148.65 CHR=5.5`	`; Straight line with angle and chamfer specification.`
`N30 X85 Z40 ANG=100`	`; Straight line with angle and end point specification.`
`N40 ...`	

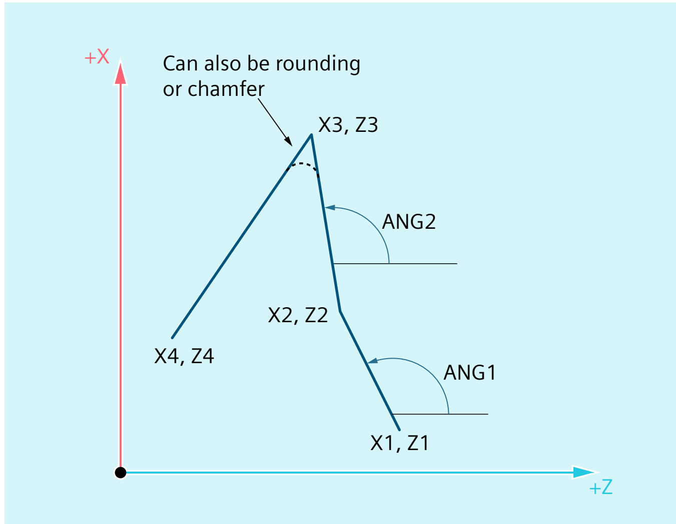
3.9.8.4 Contour definitions: Three straight lines

Note

In the following description it is assumed that:

- G18 is active (⇒ active working plane is the Z/X plane).
  (However, the programming of contour definitions is also possible without restrictions with G17 or G19.)
- The following identifiers have been defined for angle, radius and chamfer:
  - ANG (angle)
  - RND (radius)
  - CHR (chamfer)

The end point of the first straight line can be programmed by specifying the Cartesian coordinates or by specifying the angle of the two straight lines. The end point of the second and third straight lines must always be programmed with Cartesian coordinates. The intersection of the straight lines can be designed as a corner, a curve, or a chamfer.



ANG1:	Angle of the first straight line
ANG2:	Angle of the second straight line
X1, Z1:	Start coordinates of the first straight line
X2, Z2:	End point coordinates of the first straight line or start coordinates of the second straight line
X3, Z3:	End point coordinates of the second straight line or start coordinates of the third straight line
X4, Z4:	End point coordinates of the third straight line

Note

The programming described here for a three point contour definition can be expanded arbitrarily for contour definitions with more than three points.

Syntax

Programming of the end point of the first straight line by specifying the angle

- Corner as transition between the straight lines:

```
ANG=…
X… Z… ANG=…
X… Z…
```

- Rounding as transition between the straight lines:

```
ANG=… RND=...
X… Z… ANG=… RND=...
X… Z…
```

- Chamfer as transition between the straight lines:

```
ANG=… CHR=...
X… Z… ANG=… CHR=...
X… Z…
```

Programming of the end point of the first straight line by specifying the coordinates

- Corner as transition between the straight lines:

```
X… Z…
X… Z…
X… Z…
```

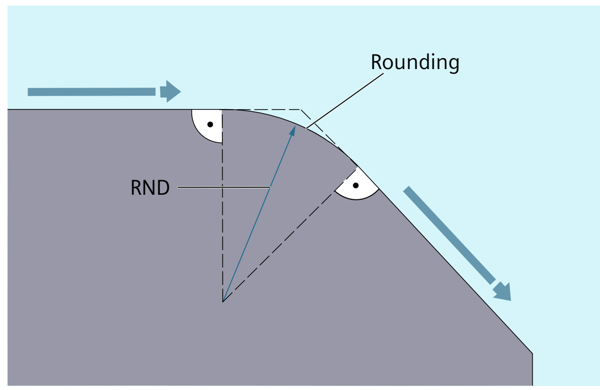
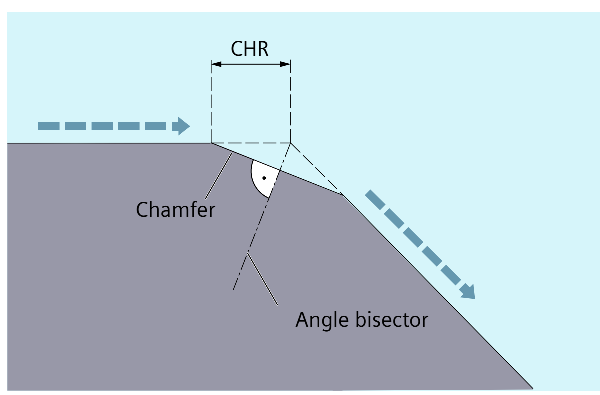
- Rounding as transition between the straight lines:

```
X… Z… RND=...
X… Z… RND=...
X… Z…
```

- Chamfer as transition between the straight lines:

```
X… Z… CHR=...
X… Z… CHR=...
X… Z…
```

Meaning

<table>
<tr><td><code>ANG=...:</code></td><td>Identifier for angle programming
The specified value (angle) refers to the abscissa of the active working plane (Z axis with <code>G18</code>).</td></tr>
<tr><td><code>RND=...:</code></td><td>Identifier for programming a rounding
The specified value corresponds to the radius of the rounding:
</td></tr>
<tr><td><code>CHR=...:</code></td><td>Identifier for programming a chamfer
The specified value corresponds to the width of the chamfer in the direction of motion:
</td></tr>
<tr><td><code>X...:</code></td><td>Coordinates in the X direction</td></tr>
<tr><td><code>Z...:</code></td><td>Coordinates in the Z direction</td></tr>
</table>

Note

For further information on the programming of a chamfer or rounding, see " Chamfer, rounding (CHF, CHR, RND, RNDM, FRC, FRCM) (Page 248) ".

Example

Program code	Comment
`N10 X10 Z100 F1000 G18`	`; Approach the starting position`
`N20 ANG=140 CHR=7.5`	`; Straight line with angle and chamfer specification.`
`N30 X80 Z70 ANG=95.824 RND=10`	`; Straight line to intermediate point with angle and chamfer specification.`
`N40 X70 Z50`	`; Straight line to end point.`

3.9.8.5 Contour definitions: End point programming with angle

Function

If the address letter A appears in an NC block, either none, one or both of the axes in the active plane may also be programmed.

Number of programmed axes

- If **no axis** of the active plane has been programmed, then this is either the first or second block of a contour definition consisting of two blocks.
  If it is the second block of such a contour definition, then this means that the starting point and end point in the active plane are identical. The contour definition is then at best a motion perpendicular to the active plane.
- If **exactly one axis** of the active plane has been programmed, then this is either a single straight line whose end point can be clearly defined via the angle and programmed Cartesian coordinate or the second block of a contour definition consisting of two blocks. In the second case, the missing coordinate is set to the same as the last (modal) position reached.
- If **two axes** of the active plane have been programmed, then this is the second block of a contour definition consisting of two blocks. If the current block has not been preceded by a block with angle programming without programmed axes of the active plane, then this block is not permitted.

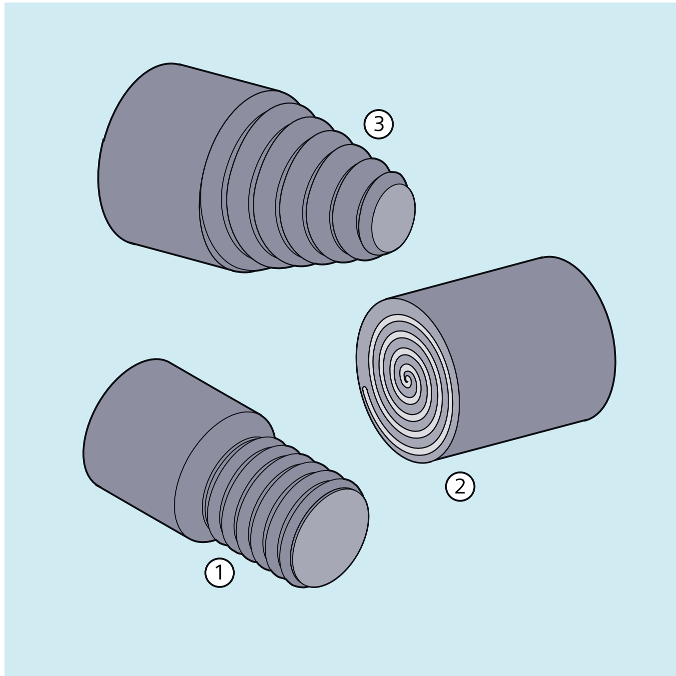
Angle A may only be programmed for linear or spline interpolation.

3.9.9 Thread cutting

3.9.9.1 Thread cutting with constant lead (G33, SF)

Threads with constant lead can be machined with `G33`:

- Cylindrical thread ①
- Face thread ②
- Taper thread ③

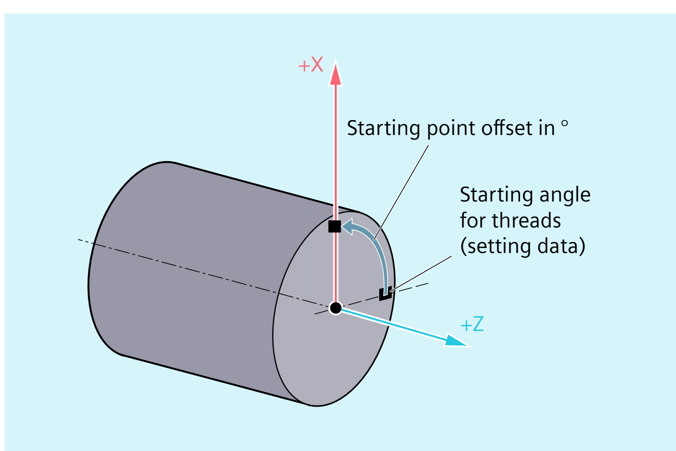


Note

Technical requirement for thread cutting with G33 is a variable-speed spindle with position measuring system.

Multiple thread

Multiple thread (thread with offset cuts) can be machined by specifying a starting point offset. The programming is performed in the G33 block at address SF.

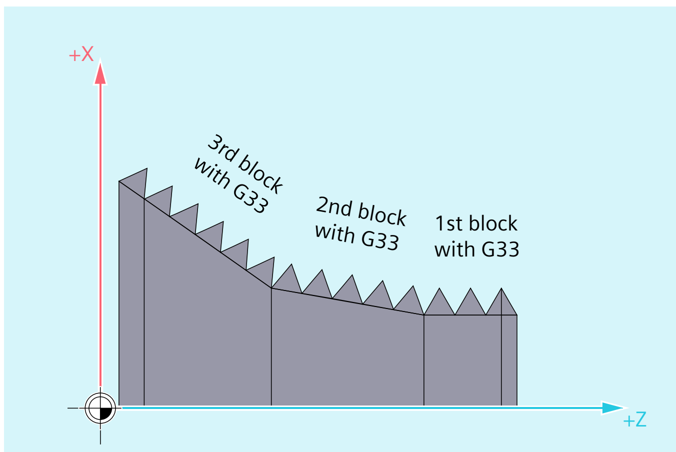


Note

If no starting point offset is specified, the "starting angle for thread" defined in the setting data is used.

Thread chain

A thread chain can be machined with several G33 blocks programmed in succession:



Note

With continuous-path mode G64, the blocks are linked by the look-ahead velocity control in such a way that there are no velocity jumps.

Direction of rotation of the thread

The direction of rotation of the thread is determined by the direction of rotation of the spindle:

- Clockwise with M3 produces a right-hand thread
- Counter-clockwise with M4 produces a left-hand thread

Syntax

Cylinder thread:

```
G33 Z… K…
G33 Z… K… SF=…
```

Face thread:

```
G33 X… I…
G33 X… I… SF=…
```

Tapered thread:

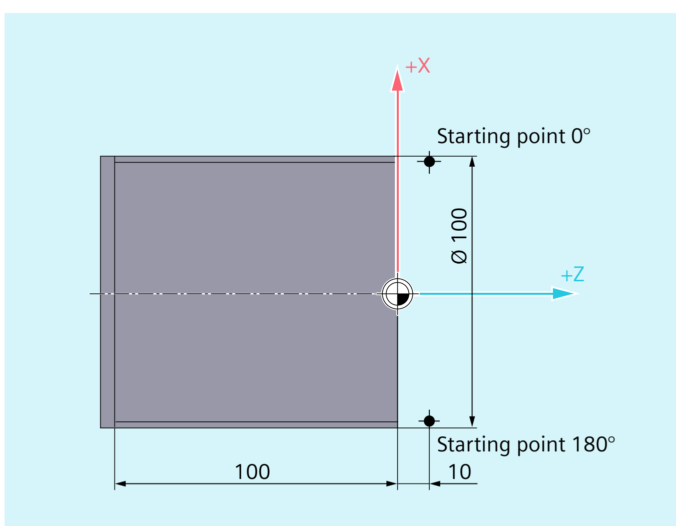
```
G33 X… Z… K…
G33 X… Z… K… SF=…
G33 X… Z… I…
G33 X… Z… I… SF=…
```

Meaning

`G33:`	Command for thread cutting with constant lead	
`X... Y... Z...:`	End point(s) in Cartesian coordinates	
`I...:`	Thread lead in X direction	
`J...:`	Thread lead in Y direction	
`K...:`	Thread lead in Z direction	
`Z:`	Longitudinal axis	
`X:`	Transverse axis	
`Z... K...:`	Thread length and lead for cylinder threads	
`X... I...:`	Thread diameter and thread lead for face threads	
`I...` or `K...:`	Thread lead for tapered threads The specification (`I...` or `K...`) refers to the taper angle:	
	< 45°:	The thread lead is specified with `K...` (thread lead in longitudinal direction).
	> 45°:	The thread lead is specified with `I..` (thread lead in transverse direction).
	= 45°:	The thread lead can be specified with `I...` or `K...`.
`SF=...:`	Starting point offset (only required for multiple threads) The starting point offset is specified as an absolute angle position.	
	Range of values:	0.0000 to 359.999 degrees

Examples

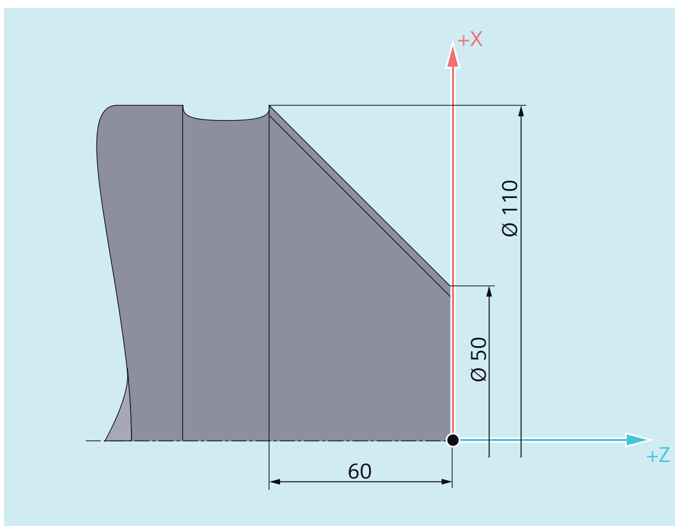
Example 1: Double cylinder thread with 180° starting point offset



Program code	**Comment**
`N10 G1 G54 X99 Z10 S500 F100 M3`	`; Work offset, approach starting point, activate spindle.`
`N20 G33 Z-100 K4`	`; Cylinder thread: End point in Z.`

Program code	Comment
N30 G0 X102	; Retraction to starting position.
N40 G0 Z10	
N50 G1 X99	
N60 G33 Z-100 K4 SF=180	; 2nd cut: Starting point offset 180°.
N70 G0 X110	; Retract tool.
N80 G0 Z10	
N90 M30	; End of program

Example 2: Tapered thread with angle less than 45°

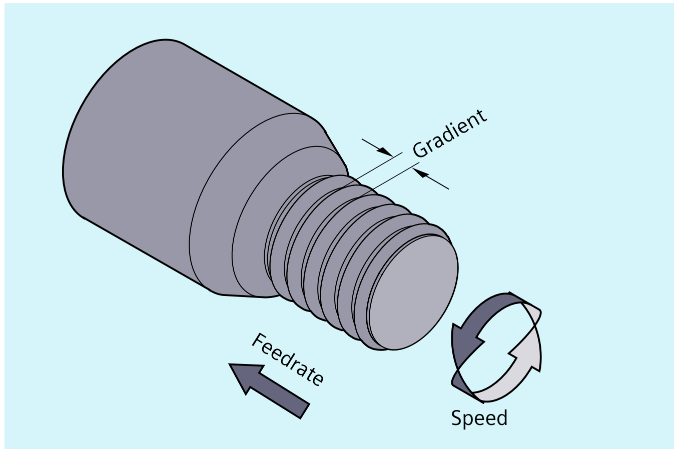


Program code	Comment
N10 G1 X50 Z0 S500 F100 M3	; Approach starting point, activate spindle.
N20 G33 X110 Z-60 K4	; Tapered thread: End point in X and Z, specification of thread lead with K... in Z direction (since angle < 45°).
N30 G0 Z0 M30	; Retraction, end of program.

Further information

Feedrate for thread cutting with G33

From the programmed spindle speed and the thread lead, the control calculates the required feedrate with which the turning tool is traversed over the thread length in the longitudinal and/or transverse direction. The feedrate `F` is not taken into account for `G33`, the limitation to maximum axis velocity (rapid traverse) is monitored by the control.



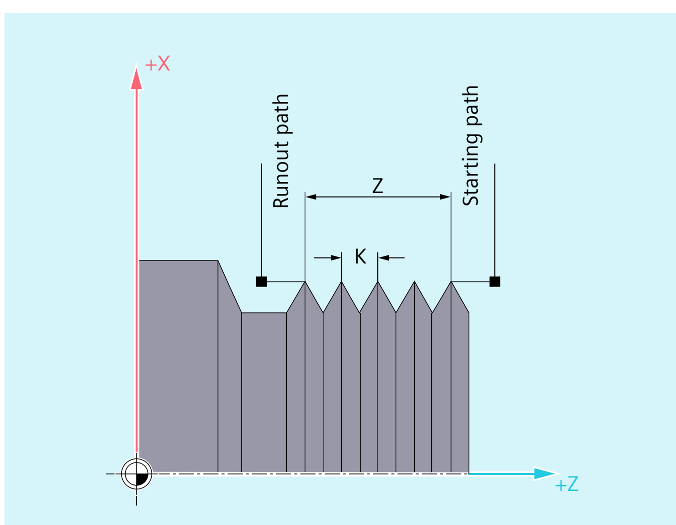
Cylinder thread

The cylinder thread is described by:

- Thread length
- Thread lead

The thread length is entered with one of the Cartesian coordinates X, Y or Z in absolute or incremental dimensions (for turning machines preferably in the Z direction). Allowance must also be made for the run-in and run-out paths, across which the feed is accelerated or decelerated.

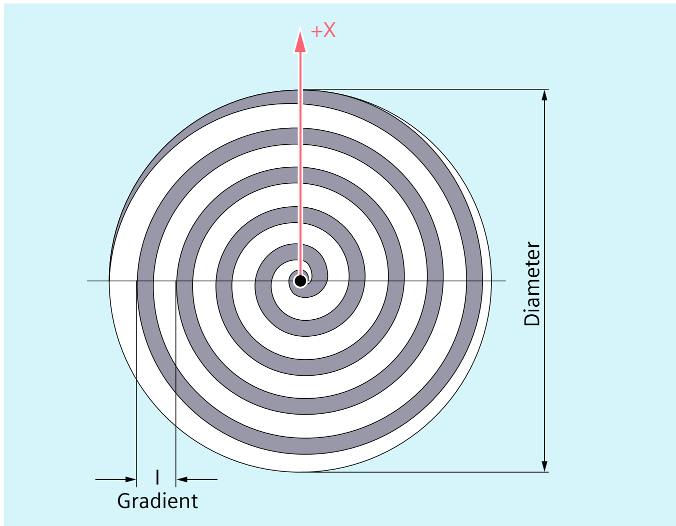
The thread lead is entered at addresses `I`, `J`, `K` (`K` is preferable for turning machines).



Face thread

The face thread is described by:

- Thread diameter (preferably in the X direction)
- Thread lead (preferably with `I`)



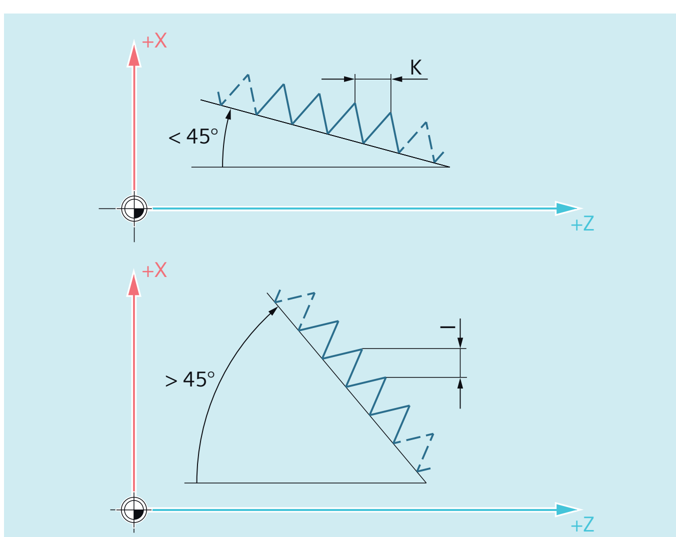
Tapered thread

The tapered thread is described by:

- End point in the longitudinal and transverse direction (taper contour)
- Thread lead

The taper contour is entered in Cartesian coordinates X, Y, Z in absolute or incremental dimensions - preferentially in the X and Z direction for machining on turning machines. Allowance must also be made for the run-in and run-out paths, across which the feed is accelerated or decelerated.

The specification of the lead depends on the taper angle (angle between the longitudinal axis and the outside of the taper):



3.9.9.2 Thread cutting with increasing or decreasing lead (G34, G35)

With the commands G34 and G35, the G33 functionality has been extended with the option of programming a change in the thread pitch at address F. With G34, this results in a linear increase and with G35 to a linear decrease of the thread pitch. Commands G34 and G35 can therefore be used for machining self-tapping threads.

Syntax

Cylinder thread with increasing pitch:

```
G34 Z… K… F...
```

Cylinder thread with decreasing pitch:

```
G35 Z… K… F...
```

Face thread with increasing pitch:

```
G34 X… I… F...
```

Face thread with decreasing pitch:

```
G35 X… I… F...
```

Taper thread with increasing pitch:

```
G34 X… Z… K… F...
G34 X… Z… I… F...
```

Taper thread with decreasing pitch:

```
G35 X… Z… K… F...
G35 X… Z… I… F...
```

Meaning

<table>
<tr><td><code>G34:</code></td><td colspan="2">Command for thread cutting with linear increasing pitch</td></tr>
<tr><td><code>G35:</code></td><td colspan="2">Command for thread cutting with linear decreasing pitch</td></tr>
<tr><td><code>X... Y... Z...:</code></td><td colspan="2">End point(s) in Cartesian coordinates</td></tr>
<tr><td><code>I...:</code></td><td colspan="2">Thread pitch in X direction</td></tr>
<tr><td><code>J...:</code></td><td colspan="2">Thread pitch in Y direction</td></tr>
<tr><td><code>K...:</code></td><td colspan="2">Thread pitch in Z direction</td></tr>
<tr><td rowspan="4"><code>F...:</code></td><td colspan="2">Thread pitch change
If you already know the starting and final pitch of a thread, you can calculate the thread pitch change to be programmed using the following equation:
$$F = \frac{k_e^2 - k_a^2}{2 * l_G} \text{ [mm/rev}^2\text{]}$$
The meanings are as follows:</td></tr>
<tr><td>k_e:</td><td>Final pitch thread (thread pitch of axis target point coordinate) [mm/rev]</td></tr>
<tr><td>k_a:</td><td>Starting thread pitch (programmed under I, J or K) [mm/rev]</td></tr>
<tr><td>l_G:</td><td>Thread length [mm]</td></tr>
</table>

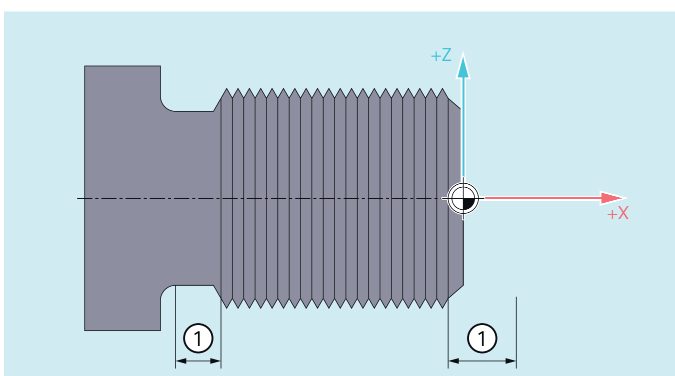
Example

Program code	Comment
N1608 M3 S10	; Spindle on.
N1609 G0 G64 Z40 X216	; Approach starting point.
N1610 G33 Z0 K100 SF=R14	; Thread cutting with constant pitch (100 mm/rev).
N1611 G35 Z-200 K100 F17.045455	; Pitch decrease: 17.0454 mm/rev2
	Pitch at end of block: 50 mm/rev
N1612 G33 Z-240 K50	; Traverse thread block without jerk.
N1613 G0 X218	
N1614 G0 Z40	
N1615 M17	

3.9.9.3 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.

3.9.9.4 Fast retraction during thread cutting (LFON, LFOF, DILF, ALF, LFTXT, LFWP, LFPOS, POLF, POLFMASK, POLFMLIN)

The "Rapid retraction during thread cutting (G33)" function can be used to interrupt thread cutting without causing irreparable damage in the following situations:

- NC stop initiated via DB320x.DBX7.3
- NC stop implicitly initiated via an alarm
- Switching a fast input (Page 535)

The retraction motion can be programmed via:

- Retraction path and retraction direction (relative)
- Retraction position (absolute)

Note

NC stop signals

During thread cutting, the following NC stop signals do not trigger a rapid retraction:
- DB320x.DBX7.4 (NC stop axes plus spindles)
- DB320x.DBX7.2 (NC stop at the block limit)

Tapping

The "Rapid retraction" function **cannot** be used with **tapping** (G331/G332).

Syntax

Enable rapid retraction, retraction motion via retraction path and retraction direction:
```
G33 ... LFON DILF=<value> LFTXT/LFWP ALF=<value>
```

Enable rapid retraction, retraction motion via retraction position:

```
POLF[<axis identifier>]=<value> LFPOS
POLFMASK/POLFMLIN(<axis 1 name>,<axis 2 name>, etc.)
```

```
G33 ... LFON
```

Disable rapid retraction during thread cutting:

```
LFOF
```

Meaning

`LFON`:	Enable rapid retraction during thread cutting (`G33`)
`LFOF`:	Disable rapid retraction during 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`. For `LFWP`, the direction in the machining plane is derived from 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 **For more information** on the programming options with `ALF` see Chapter "Traversing direction for fast retraction from the contour (Page 537)".
`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	
	Effective:	Modal
	=<value>:	In the case of geometry axes, the assigned value is interpreted as a position in the workpiece coordinate system. In the case of machine axes, it is interpreted as a position in the machine coordinate system. The values assigned can also be programmed as incremental dimensions: =IC<value>
<axis identifier>:	Identifier of a geometry axis or machine axis.	

Note

LFON or LFOF can always be programmed, but the evaluation is performed exclusively during thread cutting (G33).

Note

POLF with POLFMASK/POLFMLIN are not restricted to thread cutting applications.

Examples

Example 1: Enable rapid retraction during thread cutting

Program code	Comment
N55 M3 S500 G90 G18	; Active machining plane
...	; Approach the starting position
N65 MSG ("thread cutting")	; Tool infeed
MM_THREAD:	
N67 $AC_LIFTFAST=0	; Reset before starting the thread.
N68 G0 Z5	
N68 X10	
N70 G33 Z30 K5 LFON DILF=10 LFWP ALF=7	; Enable rapid retraction during 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	

Example 2: Switch off rapid retraction before tapping.

Program code	Comment
N55 M3 S500 G90 G0 X0 Z0	
...	
N87 MSG ("tapping")	
N88 LFOF	; Deactivate rapid retraction before tapping.
N89 CYCLE...	; Tapping cycle with G33.
N90 MSG ("")	
...	
N99 M30	

Example 3: Rapid retraction to absolute retraction position

Path interpolation of X is suppressed in the event of a stop and a motion executed to position POLF[X] at maximum velocity instead. The motion of the other axes continues to be determined by the programmed contour or the thread lead and the spindle speed.

Program code	Comment
N10 G0 G90 X200 Z0 S200 M3	
N20 G0 G90 X170	
N22 POLF[X]=210 LFPOS	
N23 POLFMASK(X)	; Activate (enable) rapid retraction from axis X.
N25 G33 X100 I10 LFON	
N30 X135 Z-45 K10	
N40 X155 Z-128 K10	
N50 X145 Z-168 K10	
N55 X210 I10	
N60 G0 Z0 LFOF	
N70 POLFMASK()	; Disable lift for all axes.
M30	

See also

Thread cutting with constant lead (G33, SF) (Page 224)

Thread cutting with increasing or decreasing lead (G34, G35) (Page 231)

3.9.9.5 Convex thread (G335, G336)

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.

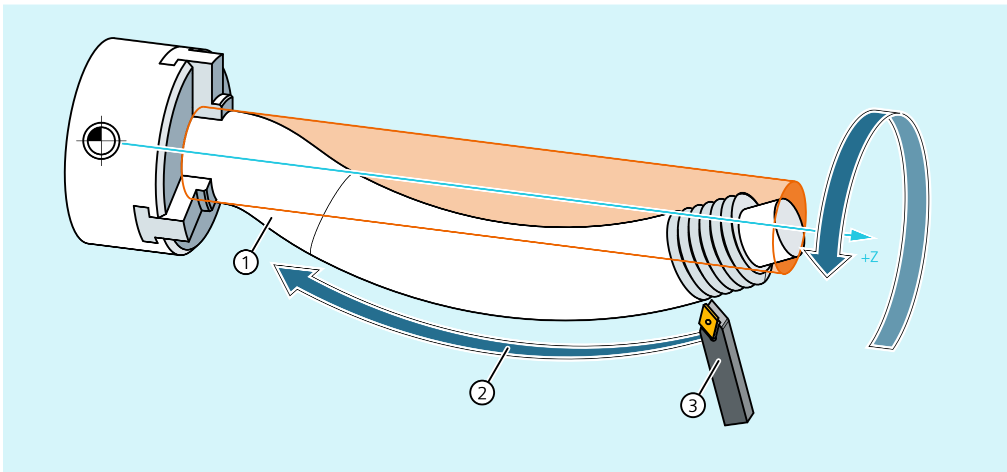


Figure 3-12 Turning a convex thread

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` (see "Thread cutting with constant lead (G33, SF) (Page 224)").

An arc is also specified. As for `G2`/`G3`, this can be programmed via the center point, radius, opening angle or intermediate point specification (see "Circular interpolation (Page 196)"). 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).

Differences to the default values must be taken from the specifications of the machine manufacturer.

Optionally, a starting point offset `SF` can also be specified (see "Thread cutting with constant lead (G33, SF) (Page 224)").

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>]
```

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 clockwise direction.

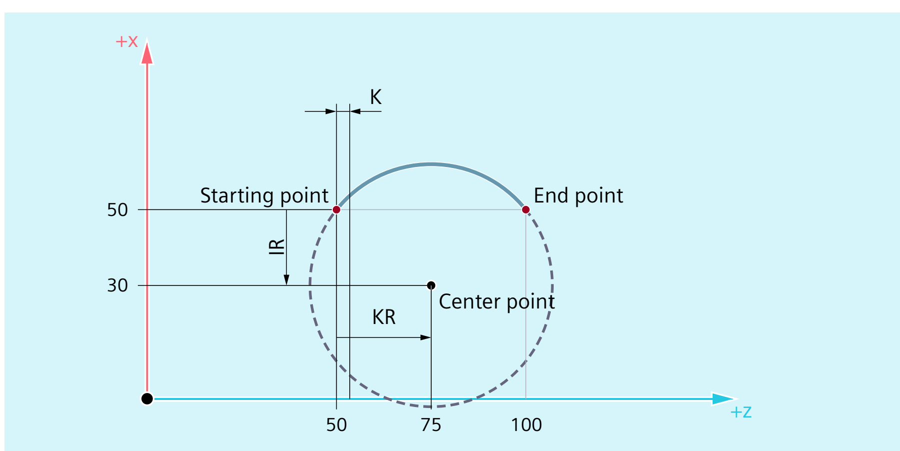


Figure 3-13 Convex thread in the clockwise direction with end and center point programming

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.

Program code	Comment
N10 **G336 Z100** K=3.5 **KR=25 IR=20** SF=90	; Turn convex thread in the counter-clockwise direction.

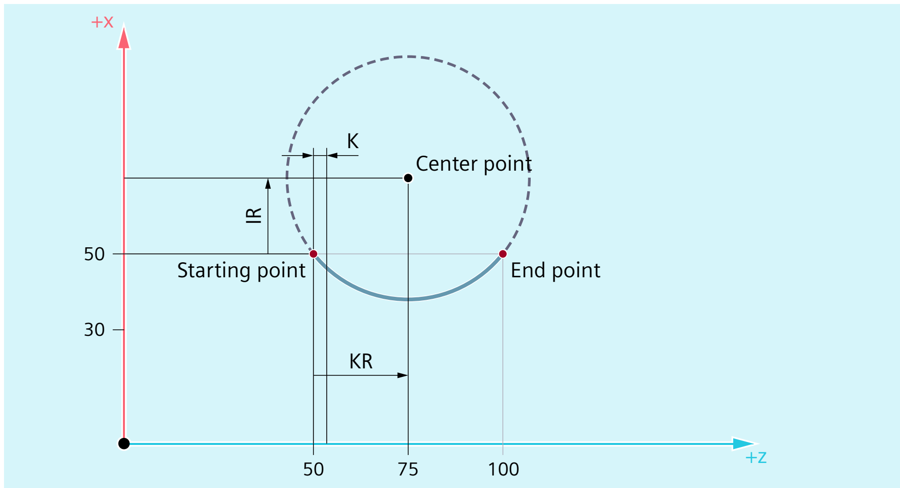


Figure 3-14 Convex thread in the counter-clockwise direction with end and center point programming

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

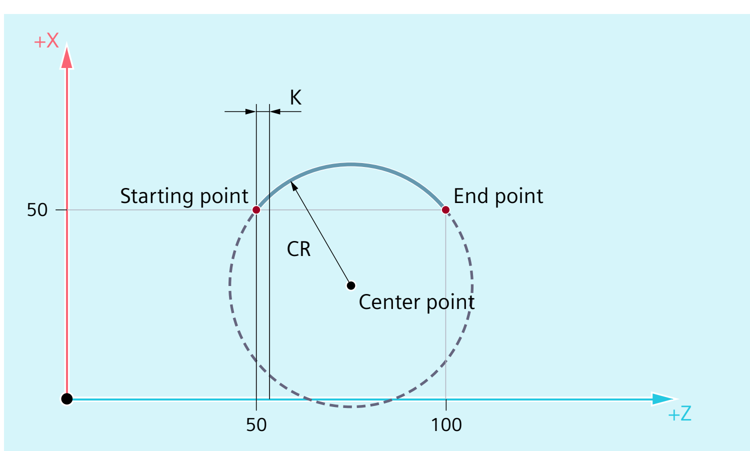


Figure 3-15 Convex thread in the clockwise direction with end point and radius programming

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

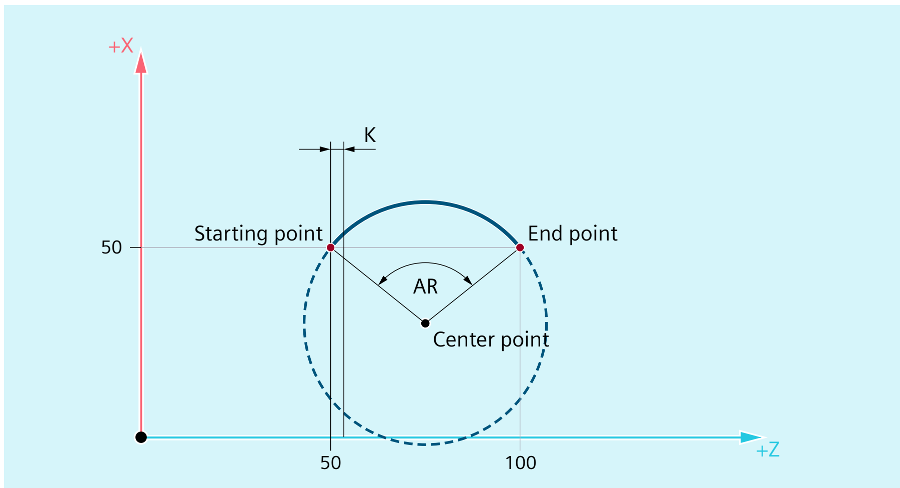


Figure 3-16 Convex thread in the clockwise direction with end point and opening angle programming

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

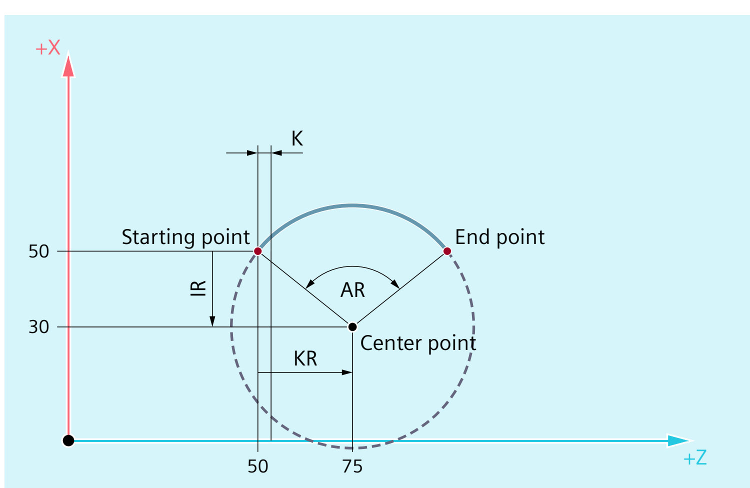


Figure 3-17 Convex thread in the clockwise direction with center point and opening angle programming

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**

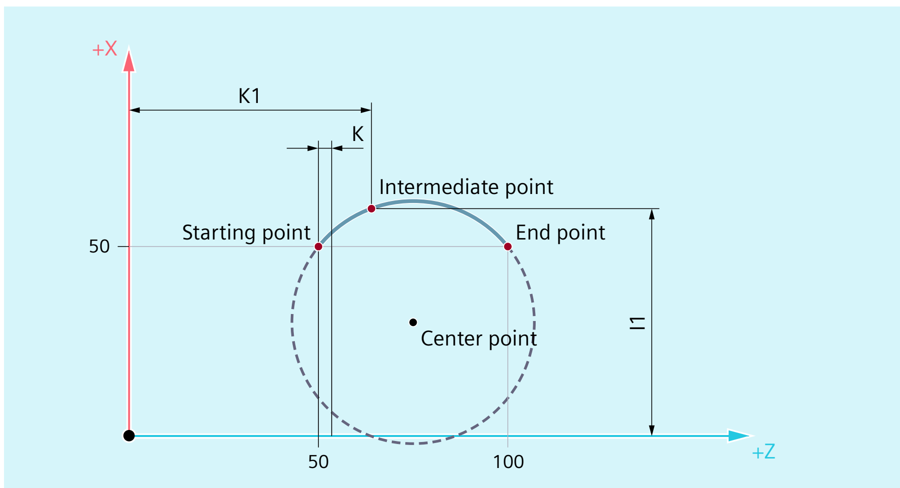
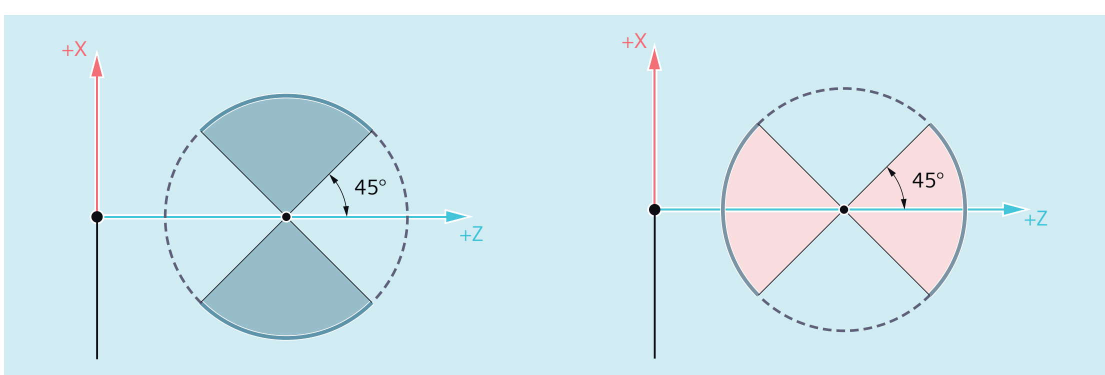


Figure 3-18 Convex thread in the clockwise direction with end and intermediate point programming

Further information

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:

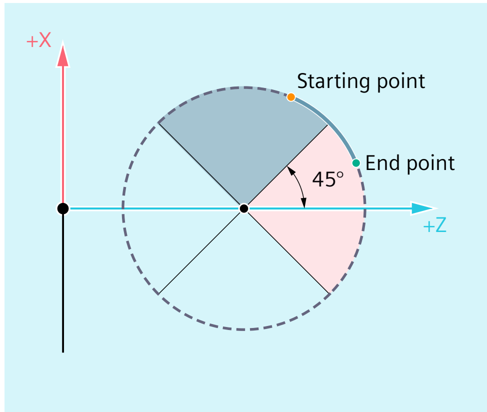


Figure 3-19 Convex thread: Area that is not permissible

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).

Supplementary conditions for the circular-path programming

The supplementary conditions described for the circular-path programming with G2/G3 apply for the circular-path programming under G335/G336 (see "Circular interpolation (Page 196)").

3.9.10 Tapping without compensating chuck

3.9.10.1 Tapping without compensating chuck and retraction motion (G331, G332)

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.	
	Effectiveness:	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.	
	Effectiveness:	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	
	Range of values:	±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.

For further information see the "Axes and Spindles" Function Manual.

Examples

- Example: Tapping with G331 / G332 (Page 244)
- Example: Output the programmed drilling speed in the current gear stage (Page 245)
- Example: Application of the second gear-stage data block (Page 246)
- Example: Speed is not programmed, the gearbox stage is monitored (Page 246)
- Example: Gearbox stage cannot be changed, gearbox stage monitoring (Page 246)
- Example: Programming without SPOS (Page 247)

3.9.10.2 Example: Tapping with G331 / G332

Program code	Comment
N10 SPOS[n]=0	; Spindle: Position control mode ; Start position 0 degrees

Program code	Comment
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

3.9.10.3 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).

3.9.10.4 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

3.9.10.5 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

3.9.10.6 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

3.9.10.7 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.

3.9.11 Tapping with compensating chuck

3.9.11.1 Tapping with compensating check and retraction motion (G63)

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`

3.9.12 Chamfer, rounding (CHF, CHR, RND, RNDM, FRC, FRCM)

Contour corners within the active working plane can be executed as roundings or chamfers.

For optimum surface quality, a separate feedrate can be programmed for chamfer/rounding. If a feedrate is not programmed, the standard path feedrate F will be applied.

The "Modal rounding" function can be used to round multiple contour corners in the same way one after the other.

Syntax

Chamfer the contour corner:

```
G... X... Z... CHR/CHF=<value> FRC/FRCM=<value>
G... X... Z...
```

Round the contour corner:
```
G... X... Z... RND=<value> FRC=<value>
G... X... Z...
```

Modal rounding:

```
G... X... Z... RNDM=<value> FRCM=<value>
...
RNDM=0
```

Note

The technology (feedrate, feedrate type, M commands, etc.) for chamfer/rounding is derived from either the previous or the next block dependent on the setting of bit 0 in machine data MD20201 $MC_CHFRND_MODE_MASK (chamfer/rounding behavior). The recommended setting is the derivation from the previous block (bit 0 = 1).

Meaning

CHF=... :	Chamfer the contour corner	
	<value>:	Length of the chamfer (unit corresponding to G70/G71)
CHR=... :	Chamfer the contour corner	
	<value>:	Width of the chamfer in the original direction of motion (unit corresponding to G70/G71)
RND=... :	Round the contour corner	
	<value>:	Radius of the rounding (unit corresponding to G70/G71)
RNDM=... :	Modal rounding (rounding multiple contour corners in the same way one after the other)	
	<value>:	Radius of the roundings (unit corresponding to G70/G71)
		Modal rounding is deactivated with RNDM=0.
FRC=... :	Non-modal feedrate for chamfer/rounding	
	<value>:	Feedrate in mm/min (with active G94) or mm/rev (with active G95)
FRCM=... :	Modal feedrate for chamfer/rounding	
	<value>:	Feedrate in mm/min (with active G94) or mm/rev (with active G95)
		FRCM=0 deactivates modal feedrate for chamfer/rounding and activates the feedrate programmed under F.

Note

Chamfer/rounding too high

If the values programmed for chamfer (CHF/CHR) or rounding (RND/RNDM) are too high for the contour elements involved, chamfer or rounding will automatically be adapted:

1. If MD11411 $MN_ENABLE_ALARM_MASK bit 4 is set, alarm 10833 "Chamfer or rounding must be reduces" is output (cancel alarm).
2. The chamfer/rounding is reduced until it fits in the contour corner. This results in at least one block without motion. At this block, the required motion is stopped.

Note

Chamfer/rounding not possible

No chamfer/rounding is performed if:

- No straight or circular contour is available in the plane
- A movement takes place outside the plane
- The plane is changed
- The number of blocks specified in the machine data that are not to contain any information about traversing (e.g. only command outputs) is exceeded

Note

FRC/FRCM

FRC/FRCM has no effect if a chamfer is traversed with G0; the command can be programmed according to the F value without error message.

FRC is only effective if a chamfer/rounding is programmed in the block or if RNDM has been activated.

FRC overwrites the F or FRCM value in the current block.

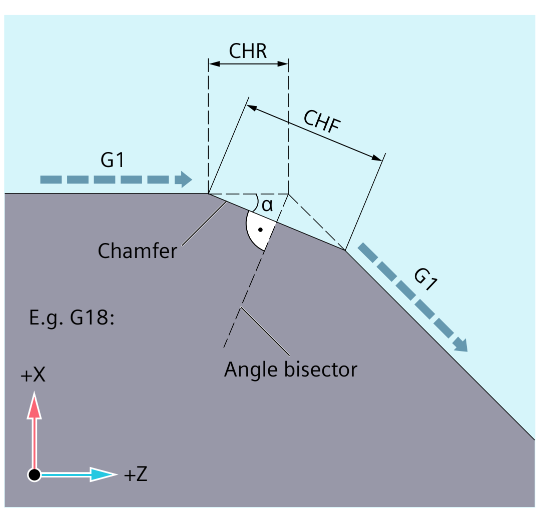
The feedrate programmed under FRC must be greater than zero.

FRCM=0 activates the feedrate programmed under F for chamfer/rounding.

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.

Examples

Example 1: Chamfer between two straight lines



- MD20201 Bit 0 = 1 (derived from previous block).
- G71 is active.
- The width of the chamfer in the direction of motion (CHR) should be 2 mm and the feedrate for chamfer 100 mm/min.

Programming can be performed in two ways:

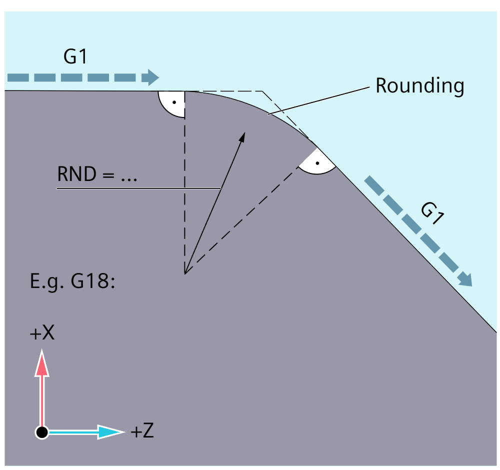
- Programming with CHR

Program code
...
N30 G1 Z… CHR=2 FRC=100
N40 G1 X…
...

- Programming with CHF

Program code
...
N30 G1 Z… CHF=2(cosα*2) FRC=100
N40 G1 X…
...

Example 2: Rounding between two straight lines

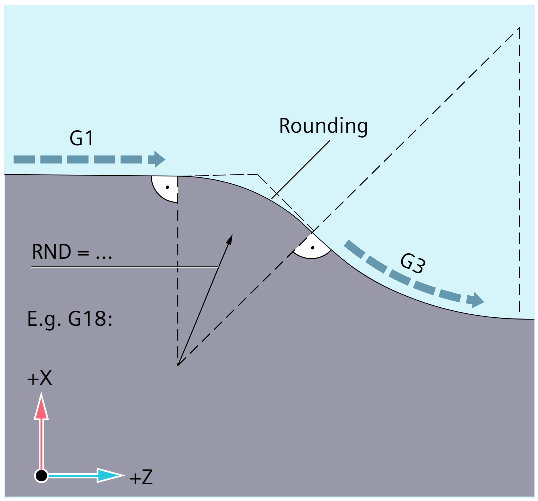


- MD20201 Bit 0 = 1 (derived from previous block).
- G71 is active.
- The radius of the rounding should be 2 mm and the feedrate for rounding 50 mm/min.

Program code
...
N30 G1 Z… RND=2 FRC=50
N40 G1 X…
...

Example 3: Rounding between straight line and circle

The RND function can be used to insert a circle contour element with tangential connection between the linear and circle contours in any combination.



- MD20201 Bit 0 = 1 (derived from previous block).
- G71 is active.
- The radius of the rounding should be 2 mm and the feedrate for rounding 50 mm/min.

Program code
...
N30 G1 Z… RND=2 FRC=50
N40 G3 X… Z… I… K…
...

Example 4: Modal rounding to deburr sharp workpiece edges

Program code	Comment
...	
N30 G1 X… Z… RNDM=2 FRCM=50	; Activate modal rounding.
	Radius of rounding: 2 mm
	Feedrate for rounding: 50 mm/min
N40...	
N120 RNDM=0	; Deactivate modal rounding.
...	

Example 5: Apply technology from following block or previous block

- MD20201 Bit 0 = 0: Derived 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	

Program code	Comment
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 (de-selection 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	

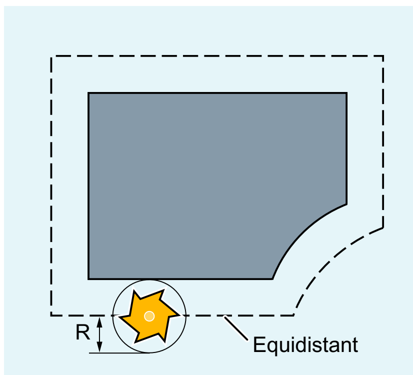
- MD20201 Bit 0 = 1: Derived from previous block (recommended setting!)

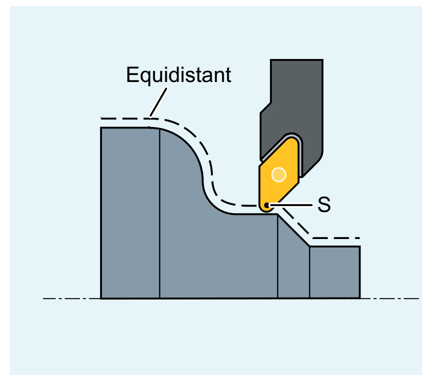
Program code	Comment
N10 G0 X0 Y0 G17 F100 G94	
N20 G1 X10 CHF=2	; Chamfer N20-N30 with F=100 mm/min
N30 Y10 CHF=4 FRC=120	; Chamfer N30-N40 with FRC=120 mm/min
N40 X20 CHF=3 FRC=200	; Chamfer N40-N60 with FRC=200 mm/min
N50 RNDM=2 FRCM=50	
N60 Y20	; Modal rounding N60-N70 with FRCM=50 mm/min
N70 X30	; 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	

3.10 Tool radius compensation

3.10.1 Activating/deactivating tool radius compensation (G40, G41, G42, OFFN):

When tool radius compensation (TRC) is active, the control automatically calculates the equidistant tool paths to the programmed workpiece contour for various tools.





R	Tool radius
S	Cutting edge center point

Commands of G Group 7 are used to activate and deactivate tool radius compensation.

Syntax

`G0/G1 X... Y… Z... `**`G41`**`/`**`G42`**` [`**`OFFN=<value>`**`]`	
`...`	
`G40`` X... Y… Z...`	

Meaning

`G41`	Activate TRC with machining direction **left** of the contour.
`G42`	Activate TRC with machining direction **right** of the contour.
`OFFN=<value>:`	Allowance on the programmed contour (normal contour offset) (optional), e.g. to generate equidistant paths for rough finishing.
`G40`	Deactivate TRC.

Note

In the NC block with G40/G41/G42, G0 or G1 must be active and at least one axis on the selected machining plane has to be specified.

If only one axis is specified on activation, the last position on the second axis is added automatically and traversed with **both** axes.

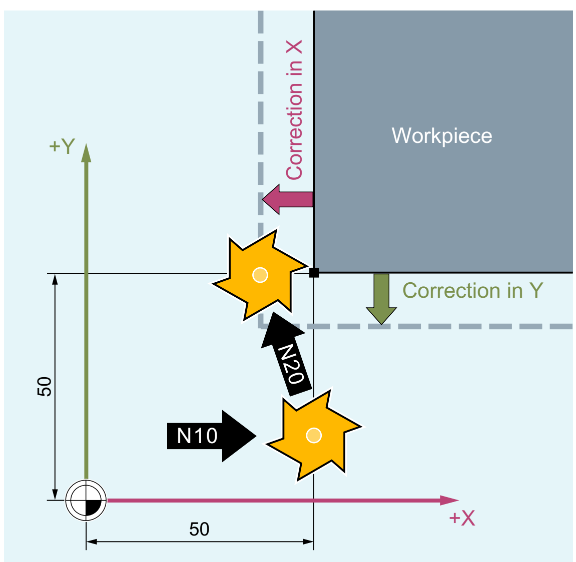
The two axes must be active as geometry axes in the channel. This can be ensured by programming GEOAX.

If no geometry axis is programmed for the current plane in the block with the tool radius compensation selection, then no selection is made.

If a geometry axis is programmed in the block with the tool radius compensation deselection, then compensation is deselected even if it is not on the current plane.

Examples

Example 1: Milling

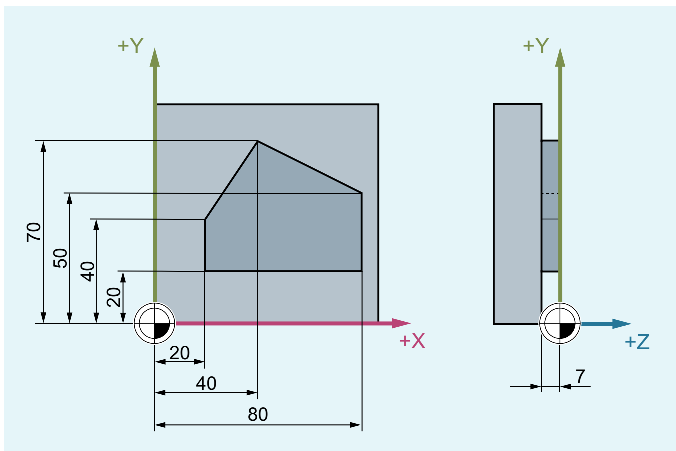


Program code	Comment
N10 G0 X50 T1 D1	; Only tool length compensation is activated. X50 is approached without compensation.
N20 G1 G41 Y50 F200	; Radius compensation is activated, point X50/Y50 is approached with compensation.
N30 Y100	
…	

Example 2: "Conventional" procedure using milling as an example

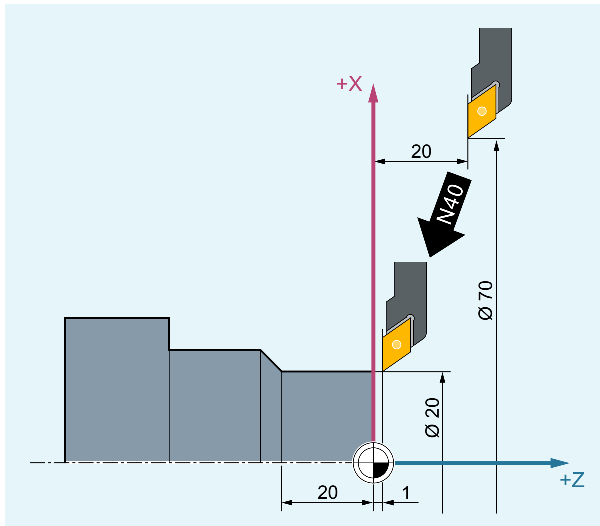
"Conventional" procedure:

1. Tool call.
2. Change tool.
3. Activate machining plane and tool radius compensation.



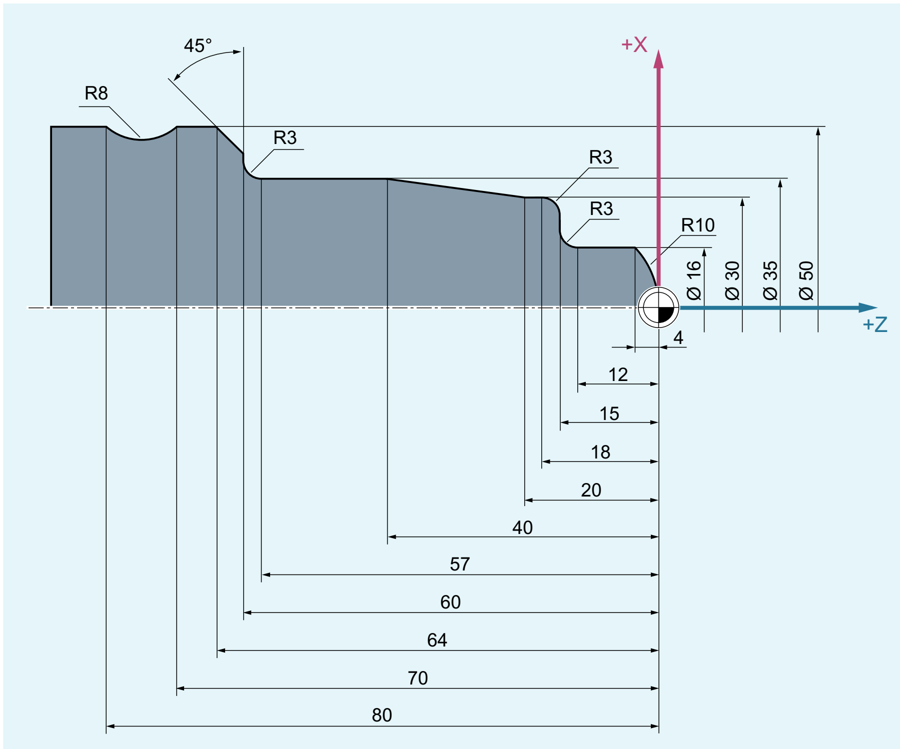
Program code	Comment
N10 G0 Z100	; Retraction for tool change.
N20 G17 T1 M6	; Tool change
N30 G0 X0 Y0 Z1 M3 S300 D1	; Call tool offset values, select length compensation.
N40 Z-7 F500	; Feed in tool.
N50 G41 X20 Y20	; Activate tool radius compensation, tool machines to the left of the contour.
N60 Y40	; Mill contour.
N70 X40 Y70	
N80 X80 Y50	
N90 Y20	
N100 X20	
N110 G40 G0 Z100 M30	; Retract tool, end of program.

Example 3: Turning



Program code	Comment
…	
N20 T1 D1	; Only tool length compensation is activated.
N30 G0 X70 Z20	; X70 Z20 is approached without compensation.
N40 G42 X20 Z1	; Radius compensation is activated, point X20/Z1 is approached with compensation.
N50 G1 Z-20 F0.2	
…	

Example 4: Turning



Program code	Comment
N5 G0 G53 X280 Z380 D0	; Starting point.
N10 TRANS X0 Z250	; Work offset.
N15 LIMS=4000	; Speed limitation (G96).
N20 G96 S250 M3	; Select constant feedrate
N25 G90 T1 D1 M8	; Select tool selection and offset.
N30 G0 G42 X-1.5 Z1	; Set tool with tool radius compensation.
N35 G1 X0 Z0 F0.25	
N40 G3 X16 Z-4 I0 K-10	; Turn radius 10.
N45 G1 Z-12	
N50 G2 X22 Z-15 CR=3	; Turn radius 3.
N55 G1 X24	
N60 G3 X30 Z-18 I0 K-3	; Turn radius 3.
N65 G1 Z-20	
N70 X35 Z-40	
N75 Z-57	
N80 G2 X41 Z-60 CR=3	; Turn radius 3.
N85 G1 X46	
N90 X52 Z-63	
N95 G0 G40 G97 X100 Z50 M9	; Deselect tool radius compensation and approach tool change location.
N100 T2 D2	; Call tool and select offset.

Program code	Comment
`N105 G96 S210 M3`	`; Select constant cutting rate.`
`N110 G0 G42 X50 Z-60 M8`	`; Set tool with tool radius compensation.`
`N115 G1 Z-70 F0.12`	`; Turn diameter 50.`
`N120 G2 X50 Z-80 I6.245 K-5`	`; Turn radius 8.`
`N125 G0 G40 X100 Z50 M9`	`; Retract tool and deselect tool radius compensation.`
`N130 G0 G53 X280 Z380 D0 M5`	`; Approach tool change location.`
`N135 M30`	`; End of program.`

More information

Calculation of the tool paths

The control requires the following information in order to calculate the tool paths:

Information	Meaning
Tool no. (T...), cutting edge no. (D...)	To calculate the distance between the tool path and the workpiece contour.
Machining direction (G41/G42)	To determine the direction in which the tool path should be shifted.
Machining plane (G17/G18/G19)	To determine the plane and therefore the axis directions in which compensation should be applied.

This is the reason that 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.

Negative compensation value

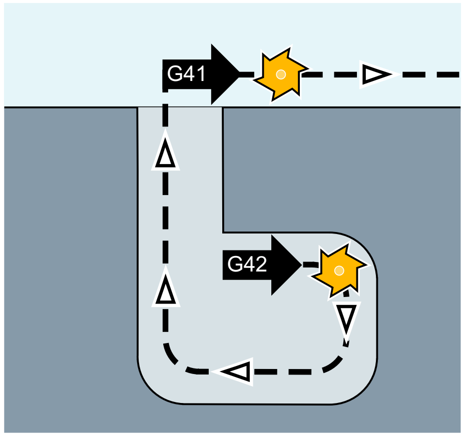
A negative compensation value has the same significance as a change of offset side (G41 ↔ G42).

Tool length compensation

The wear parameter assigned to the diameter axis on tool selection can be defined as the diameter value using an MD. This assignment is not automatically altered when the plane is subsequently changed. To do this, the tool must be selected again after the plane change.

Change in compensation direction (G41 ↔ G42)

A change in compensation direction (G41 ↔ G42) can be programmed without an intermediate G40.



Change of the machining plane

It is **not** possible to change the machining plane (G17/G18/G19) when G41/G42 is active.

Change of tool offset data block (D...)

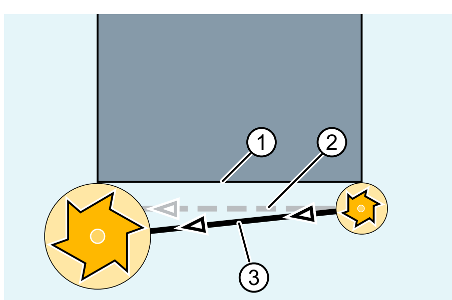
The tool offset data block can be changed in compensation mode.

A changed tool radius already becomes active as from the block containing the new D number.

Note

The radius change or compensation movement is performed across the entire block and only reaches the new equidistance at the programmed end point.

In the case of linear movements, the tool travels along an inclined path between the starting point and the end point:



① Programmed contour
② Tool path without changing the tool offset data block in the active block
③ Tool path when the tool offset data block in the active block changes and the tool radius changes as a result

Circular interpolation produces spiral movements.

Changing the tool radius

The change can be made, e.g. using system variables. The sequence is the same as when changing the tool offset data block (D...).

Note

The modified values only take effect the next time T or D is programmed. The change does not apply until the next block.

Compensation mode

Compensation mode may only be interrupted by a certain number of consecutive blocks or M functions which do not contain traversing commands or positional data in the compensation plane.

Note

The maximum number of consecutive interruption blocks or M commands can be set using machine data.

A block with a path distance of zero also counts as an interruption.

Setting data

The response of 2D tool radius compensation in certain machining situations can be set using the following setting data:

- SD42490 $SC_CUTCOM_G40_STOPRE (retraction response for a preprocessing stop before deselecting tool radius compensation)
- SD42496 $SC_CUTCOM_CLSD_CONT (tool radius compensation behavior with closed contour)

For details, see Parameter Manual Machine Data and Parameters.

See also

Adapting the approach/retract response (NORM, KONT, KONTC, KONTT) (Page 261)

3.10.2 Adapting the approach/retract response (NORM, KONT, KONTC, KONTT)

If tool radius compensation is active (G41/G42), G Group 17 commands (NORM, KONT, KONTC or KONTT) can be used to adapt the approach and retract paths of the tool to the required contour profile or the shape of the blank (unmachined part).

KONTC or KONTT ensure that continuity conditions are maintained in all three axes. It is, therefore, permissible to program a path component perpendicular to the offset plane simultaneously.

Note

A license is required for option "Polynomial interpolation" in order to use KONTC and KONTT.

Syntax

`G41/G42` **`NORM`**`/`**`KONT`**`/`**`KONTC`**`/`**`KONTT`** `X... Y... Z...`	
`...`	
`G40 X... Y... Z...`	

Meaning

`NORM`	Activate direct approach/retraction to/from a straight line. The tool is oriented perpendicular to the contour point.
`KONT`	Activate approach/retract with travel around the starting/end point The tool travels around the starting point either along a circular path or over the intersection of the equidistants depending on the programmed corner behavior (G450/G451).
`KONTC`	Activate approach/retraction with constant curvature. The contour point is approached/exited with constant curvature. There is no jump in the acceleration at the contour point.
`KONTT`	Activate approach/retraction with constant tangent. The contour point is approached/exited with constant tangent. A jump in the acceleration can occur at the contour point.

Note

Only G1 blocks are permissible as original approach/retract blocks for KONTC and KONTT. The control replaces these with polynomials for the appropriate approach/retract path.

Constraints

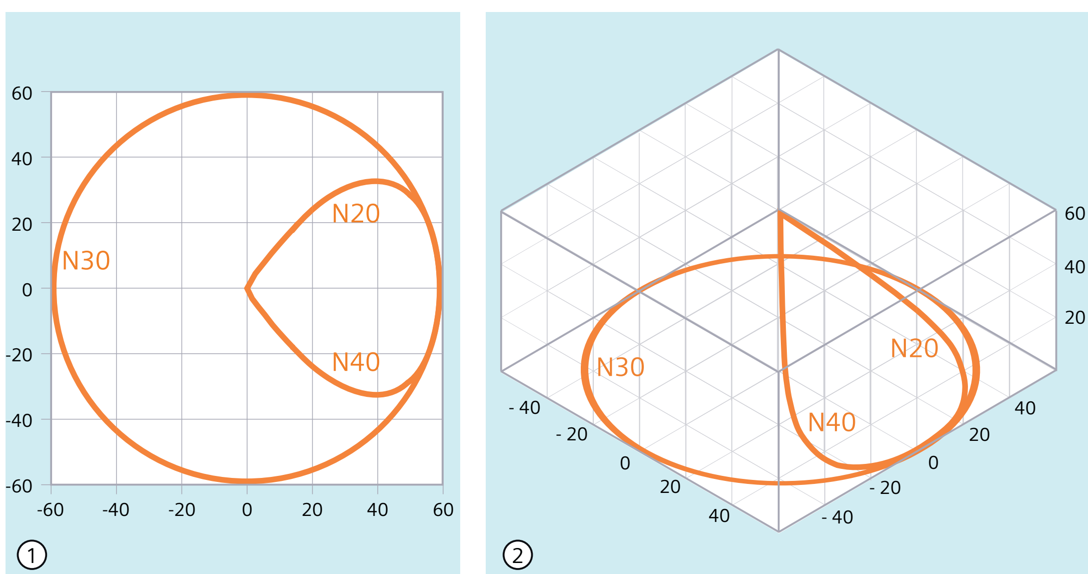
KONTT and KONTC are not available in 3D variants of tool radius compensation (CUT3DC, CUT3DCC, CUT3DF). If they are programmed, the control switches internally to NORM without an error message.

Example

In the following program example, a full circle with a radius of 70 mm is machined in the X/Y plane. The tool approaches/retracts with KONTC:

Program code	Comment
`$TC_DP1[1,1]=121`	; Milling tool
`$TC_DP6[1,1]=10`	; Radius 10 mm
`N10 G1 X0 Y0 Z60 G64 T1 D1 F10000`	
`N20 G41` **`KONTC`** `X70 Y0 Z0`	; Approach
`N30 G2 I-70`	; Full circle
`N40 G40 G1 X0 Y0 Z60`	; Retract
`N50 M30`	

As the tool has a radius of 10 mm, the resulting tool center point path describes a circle with a radius of 60 mm. Start and end point are at X0 Y0 Z60. When approaching the full-circle with KONTC (N20), the curvature is adapted to the circular path of the full circle. At the same time, the axis traverses from Z60 to the plane of circle Z0. The axis retracts (N40) in the same fashion.

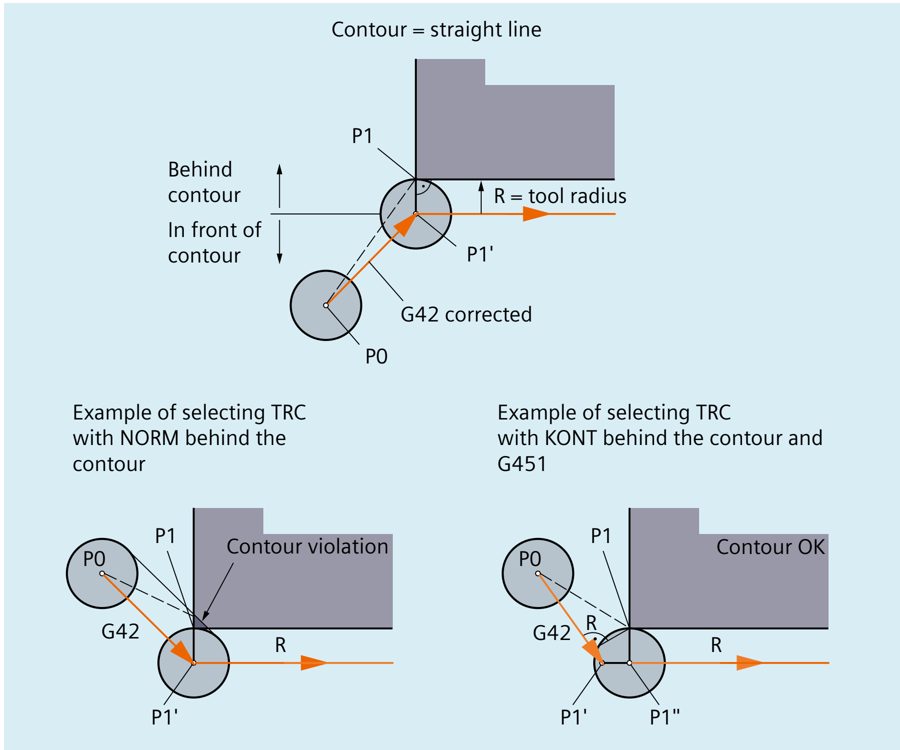


① Perpendicular projection
② Spatial representation
Figure 3-20 Tool path

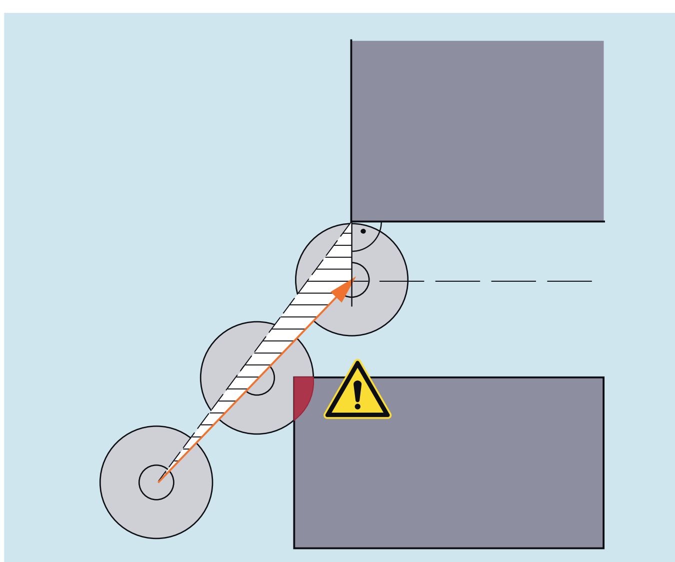
More information

Comparison of NORM and KONT

KONT only differs from NORM when the tool start position is behind the contour:



Modified approach/retract angle



NOTICE
Risk of collision Modified approach/retract angles as a result of the tool radius compensation must be taken into account during programming in order that collisions are avoided.

3.10.3 Defining the response when traveling around outside corners (G450, G451, DISC)

When tool radius compensation (G41/G42) is active, the compensated tool path when traveling around outside corners can be defined using commands of G Group 18 (G450/G451).

Note

G450/G451 is also used to define the approach path when KONT is active and the approach point behind the contour (see "Adapting the approach/retract response (NORM, KONT, KONTC, KONTT) (Page 261)").

Syntax

```
G450 [DISC=<value>]
G451
```

Meaning

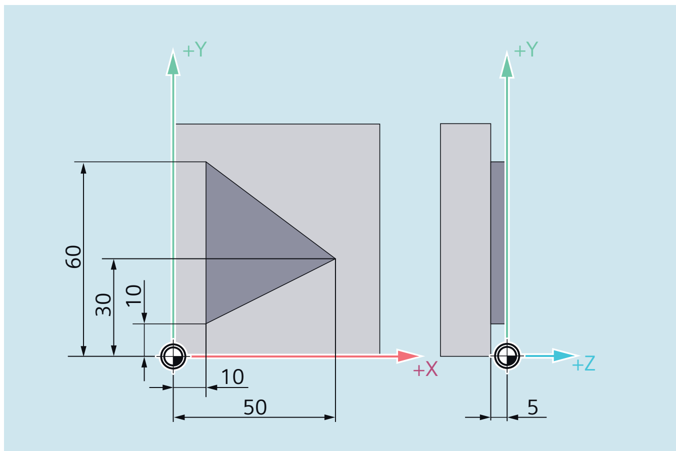
<table>
<tr><td>G450</td><td colspan="4">Activating travel around with transition circle
With G450, the tool center point travels around the workpiece outside corner along an arc with the tool radius.</td></tr>
<tr><td rowspan="5">DISC</td><td colspan="4">Flexible programming of the circular path with G450 (optional)
No sharp outside contour corners can occur with G450 as 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). If sharp outside corners are still to be machined with G450, the DISC instruction in the NC program can be used to program an overshoot. Thus, the transition circle becomes a conic section and the tool cutting edge retracts from the outside corner.</td></tr>
<tr><td rowspan="4"><value></td><td>Type:</td><td colspan="2">INT</td></tr>
<tr><td>Range of values:</td><td colspan="2">0, 1, 2, ... 100</td></tr>
<tr><td rowspan="2">Meaning:</td><td>0</td><td>Transition circle</td></tr>
<tr><td>100</td><td>Intersection of the equidistants (theoretical value)</td></tr>
<tr><td>G451</td><td colspan="4">Activate travel around with intersection point of the equidistants
With G451, the tool center point approaches the point of intersection of the two equidistants, which are located at a distance equivalent to the tool radius from the programmed contour. The tool backs off from the workpiece corner.
G451 only applies to straight lines and circles.</td></tr>
</table>

Note

DISC only applies with call of G450; however, it can be programmed in a previous block without G450. Both commands are modal.

Example

In the following example, a transition radius is programmed for all outside corners (corresponding to the programming of the corner behavior in block N30). This prevents the tool stopping and backing off at the change of direction.



Program code	Comment
`N10 G17 T1 G0 X35 Y0 Z0 F500`	`; Starting conditions.`
`N20 G1 Z-5`	`; Feed in tool.`
`N30 G41 KONT` **`G450`** `X10 Y10`	`;` **`Activate`** `TRC with KONT approach/retract mode and` **`corner behavior G450`**`.`
`N40 Y60`	`; Mill the contour.`
`N50 X50 Y30`	
`N60 X10 Y10`	
`N80 G40 X-20 Y50`	`; Deactivate compensation mode, retraction on transition circle.`
`N90 G0 Y100`	
`N100 X200 M30`	

More information

G450/G451

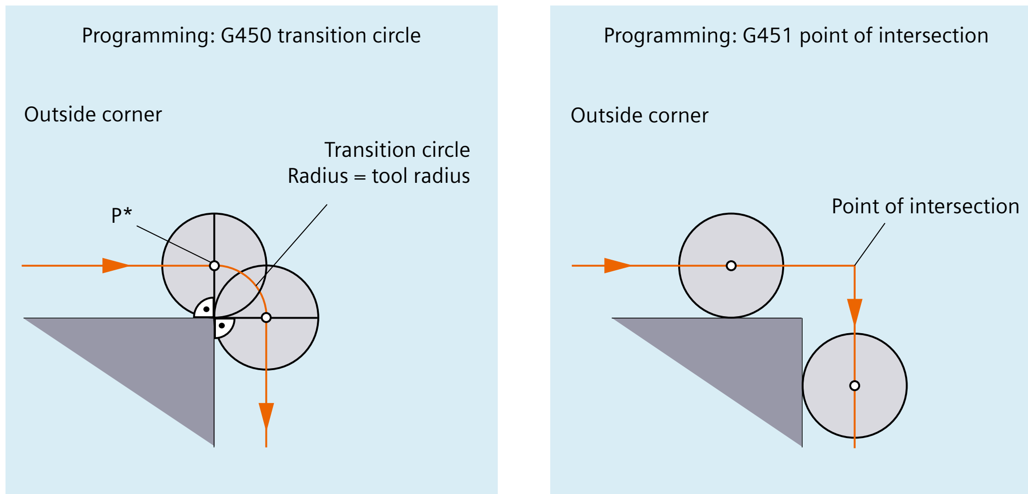


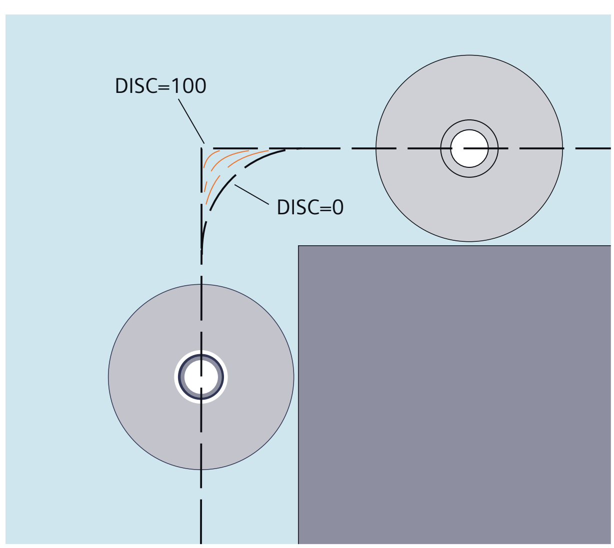
Figure 3-21 Traveling around a 90° outside corner with G450 or G451

At intermediate point P*, the control executes operations such as infeed movements or switching functions. These operations are programmed in blocks inserted between the two blocks forming the corner.

With respect to the data, for G450 the transition circle belongs to the next traversing command.

DISC

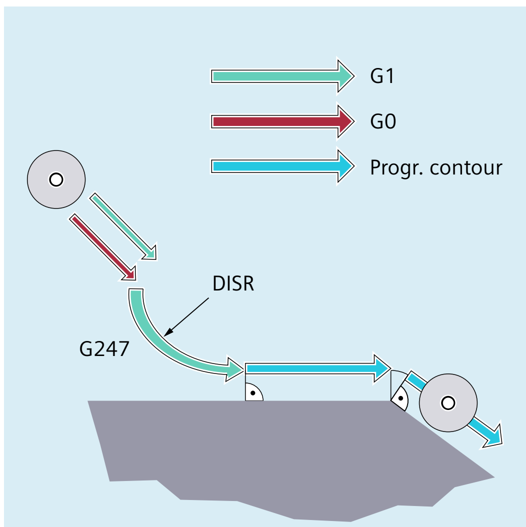
When DISC values greater than 0 are specified, transition circles are shown with an increased height – the result is transition ellipses or parabolas or hyperbolas.



3.10.4 Smooth approach and retraction

3.10.4.1 Soft approach and retraction (G140 to G143, G147, G148, G247, G248, G347, G348, G340, G341, DISR, DISCL, DISRP, FAD, PM, PR):

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.



This function is used preferably in conjunction with the tool radius compensation.

When the function is activated, the control calculates the intermediate points in such a way that the transition to the following block (or the transition from previous block during retraction) is performed in accordance with the specified parameters.

The approach movement consists of a maximum of four sub-movements. The starting point of the movement is called P_0, the end point P_4 in the following. Up to three intermediate points P_1, P_2 and P_3 can be between these points. 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.

Syntax

Smooth approach:

- With a straight line:
  `G147 G340/G341 ... DISR=..., DISCL=..., DISRP=... FAD=...`
- With a quadrant/semicircle:
  `G247/G347 G340/G341 G140/G141/G142/G143 ... DISR=... DISCL=... DISRP=... FAD=...`

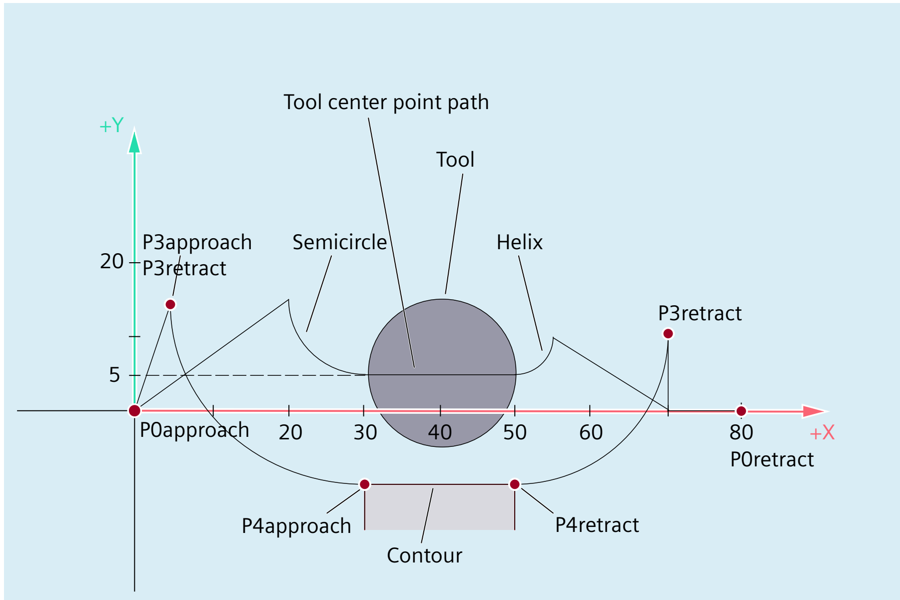
Smooth retraction:

- With a straight line:
  `G148 G340/G341 ... DISR=..., DISCL=..., DISRP=... FAD=...`
- With a quadrant/semicircle:
  `G248/G348 G340/G341 G140/G141/G142/G143 ... DISR=... DISCL=... DISRP=... FAD=...`

Meaning

`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
`G340:`	Approach and retraction in space (default setting)
`G341:`	Approach and retraction in the plane
`G140:`	Approach and retraction direction dependent on the current compensation side (default setting)
`G141:`	Approach from the left or retraction to the left
`G142:`	Approach from the right or retraction to the right
`G143:`	Approach and retraction direction dependent on the relative position of the start or end point to the tangent direction
`DISR=...:`	1. For approach and retraction with straight lines (G147/G148): Distance of the cutter edge from the starting point of the contour 2. For approach and retraction with circles (G247, G347/G248, G348): Radius of the tool center point path **Notice:** For REPOS with a semicircle, DISR is the circle diameter
`DISCL=...:`	Distance of the end point for the fast infeed motion from the machining plane DISCL=AC(...) Specification of the absolute position of the end point for the fast infeed motion
`DISCL=AC(...):`	Specification of the absolute position of the end point for the fast infeed motion
`DISRP:`	Distance of point P1 (retraction plane) from the machining plane
`DISRP=AC(...):`	Specification of the absolute position of point P1
`FAD=...:`	Speed of the slow feed movement The programmed value acts in accordance with the active feedrate type (G group 15).
`FAD=PM(...):`	The programmed value is interpreted as linear feedrate (like G94) irrespective of the active feedrate type.
`FAD=PR(...):`	The programmed value is interpreted as revolutional feedrate (like G95) irrespective of the active feedrate type.

Example



- Smooth approach (block N20 activated)
- Approach with quadrant (G247)
- Approach direction not programmed, G140 applies, i.e. TRC is active (G41)
- Contour offset OFFN=5 (N10)
- Current tool radius=10, and so the effective compensation radius for TRC=15, the radius of the SAR contour =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 to Z0 with FAD=200.
  - Approach circle in X-Y-plane and following blocks with F1500 (for this velocity to take effect in the following blocks, the active G0 in N30 must be overwritten with G1, otherwise the contour would be machined further with G0).
- Smooth retraction (block N60 activated)
- Retraction with quadrant (G248) and helix (G340)
- FAD not programmed, since irrelevant for G340
- Z=2 in the starting point; Z=8 in the end point, since DISCL=6
- When DISR=5, the radius of the SAR contour=20, the radius of the tool center point path=5

Retraction movements from Z8 to Z20 and the movement parallel to the X-Y plane to X70 Y0.

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`	`; (P0 app)`
`N20 G41 G247 G341 Z0 DISCL=AC(7) DISR=10 F1500 FAD=200`	`; Approach (P3 app)`
`N30 G1 X30 Y-10`	`; (P4 app)`
`N40 X40 Z2`	
`N50 X50`	`; (P4 ret)`
`N60 G248 G340 X70 Y0 Z20 DISCL=6 DISR=5 G40 F10000`	`; Retraction (P3 ret)`
`N70 X80 Y0`	`; (P0 ret)`
`N80 M30`	

Further information

Selecting the approach and retraction contour

The approach and retraction contour are selected with the appropriate G command from the 2nd G group:

G147:	Approach with a straight line
G247:	Approach with a quadrant
G347:	Approach with a semicircle
G148:	Retraction with a straight line
G248:	Retraction with a quadrant
G348:	Retraction with a semicircle

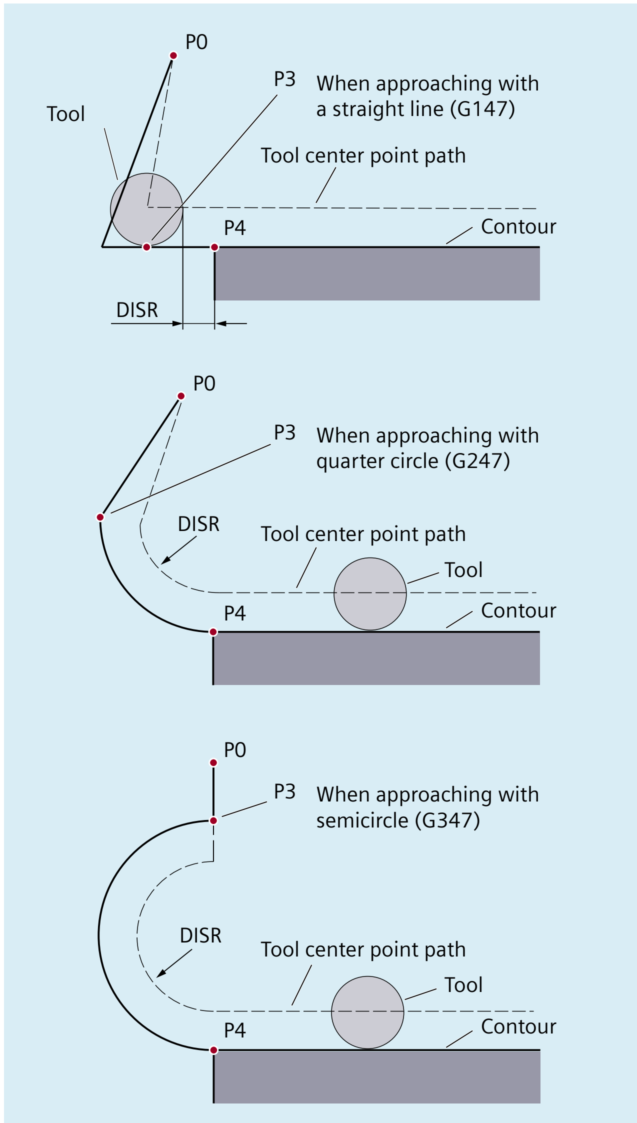


Figure 3-22 Approach movements with simultaneous activation of the tool radius compensation

Selecting the approach and retraction direction

Use the tool radius compensation (G140, default setting) to determine the approach and retraction direction with positive tool radius:

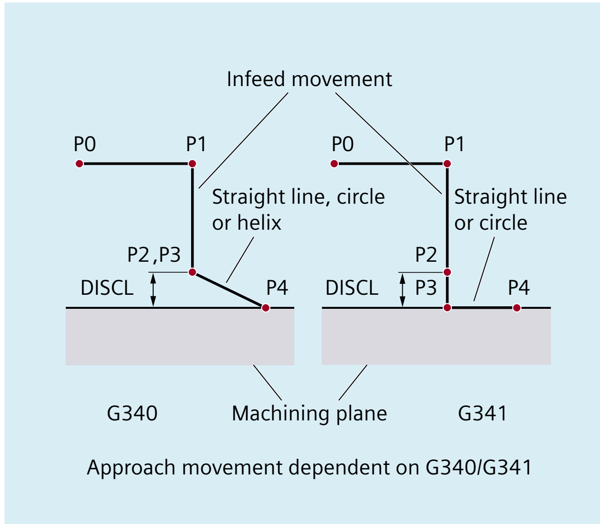
- G41 active → approach from left
- G42 active → approach from right

G141, G142 and G143 provide further approach options.

The G codes are only significant when the approach contour is a quadrant or a semicircle.

Motion steps between start point and end point (G340 and G341).

In all cases, the movements are made up of one or more straight lines and, depending on the G command for determining the approach contour, an additional straight line or a quadrant or semicircle. The two variants of the path segmentation are shown in the following figure:



G340:	Approach with a straight line from point P_0 to point P_1. This straight line is parallel to the machining plane, if parameter DISRP has not been programmed. Infeed perpendicular to the machining plane from point P_1 to point P_3 to the safety clearance to the machining plane defined by the DISCL parameter. Approach end point P4 with the curve determined by the G command of the second group (straight line, circle, helix). 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/G18/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:	Approach with a straight line from point P_0 to point P_1. This straight line is parallel to the machining plane, if parameter DISRP has not been programmed. Infeed perpendicular to the machining plane from point P_1 up to the safety clearance to the machining plane defined by the DISCL parameter in point P_2. Infeed perpendicular to the machining plane from point P_2 to point P_3. Approach end point with the curve determined by the G command of the second group. 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.

In all cases that include the position of the active plane G17/G18/G19 (circular plane, helical axis, infeed motion perpendicular to the active plane), any active rotating frame is taken into account.

Length of the approach straight line or radius for approach circles (DISR)

- Approach/retract with straight lines

  DISR specifies the distance of the cutter edge from the starting point of the contour, i.e. the length of the straight line when TRC is active is the sum of the tool radius and the programmed value of DISR. The tool radius is only taken into account when it is positive. The resulting straight line length must be positive, i.e. negative values for DISR are allowed provided that the absolute value of DISR is less than the tool radius.
- Approach/retract with circles
  DISR specifies the radius of the tool center point path. If TRC is activated, a circle is produced with a radius that results in the tool center point path with the programmed radius.

Distance of point P2 from the machining plane (DISCL)

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(...)`.

The following applies for DISCL=0:

- With G340: The whole of the approach motion now only consists of two blocks (P_1, P_2 and P_3 are combined). The approach contour is formed by P_1 to P_4.
- With G341: The whole approach contour consists of three blocks (P_2 and P_3 are combined). If P_0 and P_4 are on the same plane, only two blocks result (infeed movement from P_1 to P_3 is omitted).
- The point defined by DISCL is monitored to ensure that it is located between P_1 and P_3, i.e. the sign must be identical for the component perpendicular to the machining plane in all motions that possess such a component.
- On detection of a reversal of direction, a tolerance defined by the machine data MD20204 $MC_SAR_CLEARANCE_TOLERANCE is permitted.

Distance of point P1 (retraction plane) from the machining plane (DISRP)

If the position of point P_1 is to be specified by an absolute reference on the axis perpendicular to the machining plane, the value must be programmed in the form `DISRP=AC(...)`.

If this parameter is not programmed, point P_1 has the same distance to the machining plane as point P_0, i.e. the approach straight line $P_0 \rightarrow P_1$ is parallel to the machining plane.

The system checks that the point defined by DISRP lies between P_0 and P_2, i.e. in all movements that have a component perpendicular to the machining plane (e.g. infeed movements, approach movements from P_3 to P_4), this component must have the same leading sign. It is not permitted to change direction. An alarm is output if this condition is violated.

On detection of a reversal of direction, a tolerance defined by the machine data MD20204 $MC_SAR_CLEARANCE_TOLERANCE is permitted. However, if P_1 is outside the range defined by P_0 and P_2, but the deviation is less than or equal to this tolerance, it is assumed that P_1 is in the plane defined by P_0 or P_2.

Programming of the end point

The end point is generally programmed with X... Y... Z...

The programming of the contour end point when approaching differs greatly from that for retraction. Both cases are therefore treated separately here.

Programming of end point P4 for approach

End point P_4 can be programmed in the actual SAR block. Alternatively, P_4 can be determined by the end point of the next traversing block. More blocks can be inserted between an SAR block and the next traversing block without moving the geometry axes.

Example:

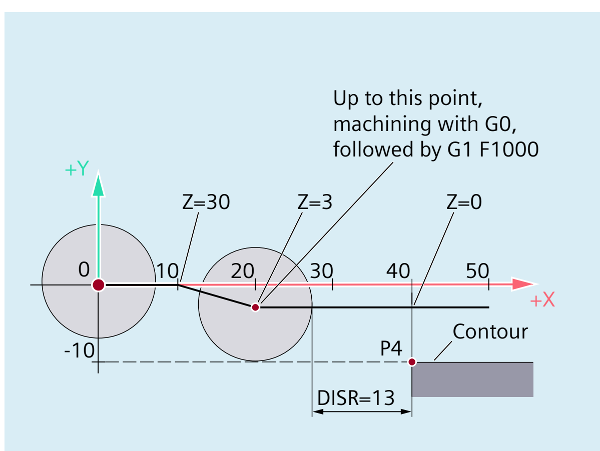
Program code	Comment
`$TC_DP1[1,1]=120`	`;Milling tool T1/D1`
`$TC_DP6[1,1]=7`	`;Tool with 7 mm radius`
`N10 G90 G0 X0 Y0 Z30 D1 T1`	
`N20 X10`	
`N30 G41 G147 DISCL=3 DISR=13 Z=0 F1000`	
`N40 G1 X40 Y-10`	
`N50 G1 X50`	
`...`	

`N30/N40` can be replaced by:

```
N30 G41 G147 DISCL=3 DISR=13 X40 Y-10 Z0 F1000
```

or

```
N30 G41 G147 DISCL=3 DISR=13 F1000
N40 G1 X40 Y-10 Z0
```



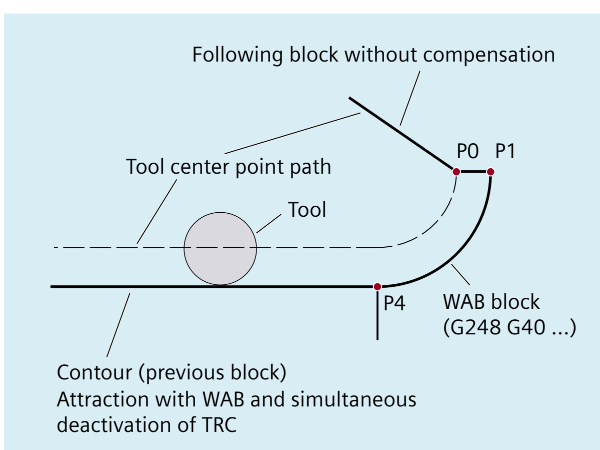
Programming of end point P0 for retraction

For retraction, the end point of the SAR contour cannot be programmed in a following block, i.e. the end position is always taken from the SAR block, irrespective of how many axes have been programmed. When determining the end point, a distinction is made between the following three cases:

1. No geometry axis is programmed in the SAR block. In this case, the contour ends at point P_1 (if DISRP has been programmed), at point P_2 (if DISCL, but not DISRP has been programmed) or point P_3 (if neither DICLS nor DISRP has been programmed).
   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 or DISPR. If in this case both DISCL=0 and DISRP=0, the motion is completely in the plane, i.e. points P_0 to P_3 coincide.
2. Only the axis perpendicular to the machining plane is programmed in the SAR block. In this case, the contour ends at point P_0. If DISRP has been programmed (i.e. points P_0 and P_1 do not coincide), the straight line $P_1 \rightarrow P_0$ is perpendicular to the machining plane. The positions of the two other axes are determined in the same way as in 1.
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. If the SAR retraction block is also used to deactivate the tool radius compensation, in the first two cases, an additional path component is inserted in the machining plane from P_1 to P_0 so that no movement is produced when the 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. In case 3, no special measures are required for deselection of the 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 behavior in cases 1 and 2, i.e. when an end point is not explicitly programmed in the machining plane with simultaneous deselection of the tool radius compensation, is shown in the following figure:

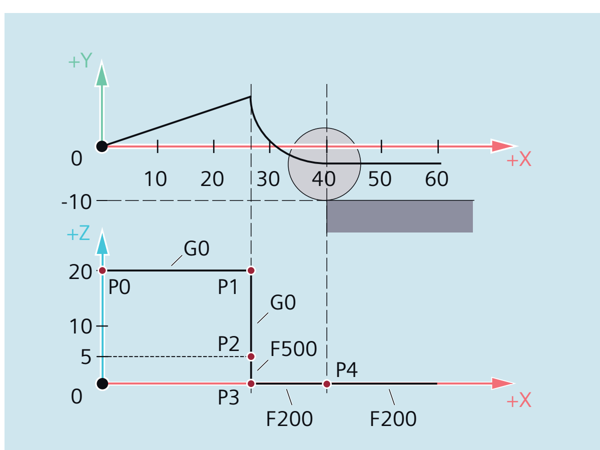


Approach and retraction velocities

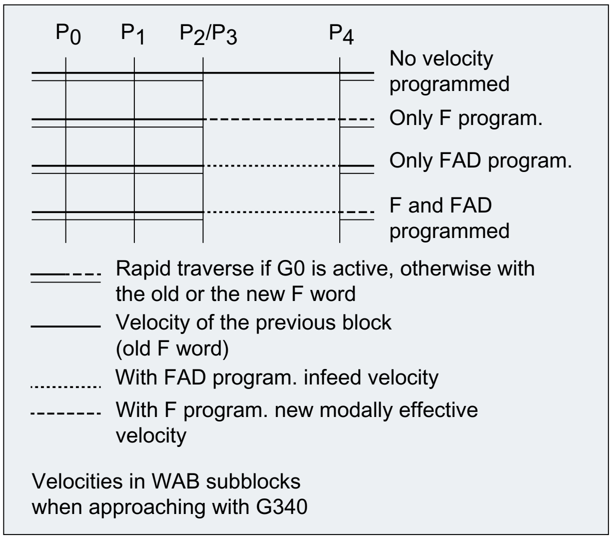
- Velocity of the previous block (G0)
  All motions from P_0 up to P_2 are executed at this velocity, i.e. the motion parallel to the machining plane and the part of the infeed motion up to the safety clearance.
- Programming with FAD
  Specification of the feedrate for
  - G341: Infeed movement perpendicular to the machining plane from P_2 to P_3
  - G340: From point P_2 or P_3 to P_4.
    If FAD is not programmed, this part of the contour is traversed at the speed which is active modally from the preceding block, in the event that no F command defining the speed is programmed in the SAR block.
- Programmed feedrate F
  This feedrate value is effective as of P_3 or P_2 if FAD is not programmed. If no F word is programmed in the SAR block, the speed of the previous block is active.

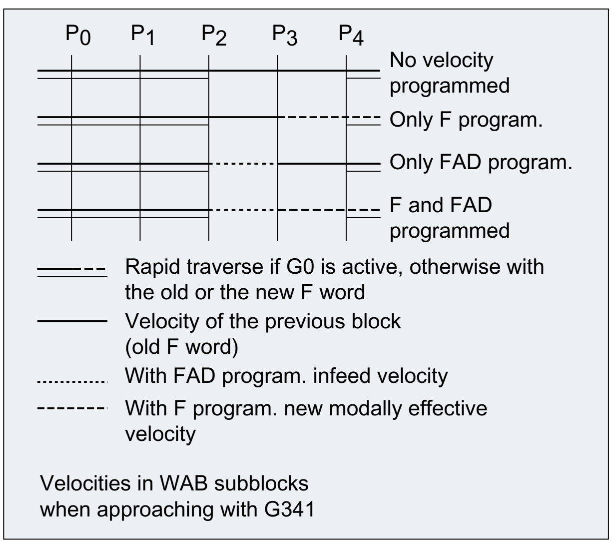
Example:

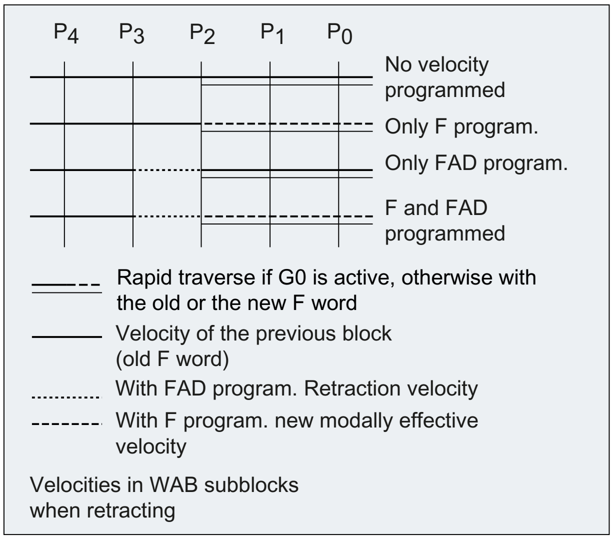
Program code	Comment
`$TC_DP1[1,1]=120`	`;Milling tool T1/D1`
`$TC_DP6[1,1]=7`	`;Tool with 7 mm radius`
`N10 G90 G0 X0 Y0 Z20 D1 T1`	
`N20 G41 G341 G247 DISCL=AC(5) DISR=13 FAD 500 X40 Y-10 Z=0 F200`	
`N30 X50`	
`N40 X60`	
`...`	



During retraction, the roles of the modally active feedrate from the previous block and the programmed feedrate value in the SAR block are reversed, i.e. the actual retraction contour is traversed with the old feedrate and a new speed programmed with the F word applies from P_2 up to P_0.


P0 P1 P2/P3 P4
No velocity programmed
Only F program.
Only FAD program.
F and FAD programmed
Rapid traverse if G0 is active, otherwise with the old or the new F word
Velocity of the previous block (old F word)
With FAD program. infeed velocity
With F program. new modally effective velocity
Velocities in WAB subblocks when approaching with G340


P0 P1 P2 P3 P4
No velocity programmed
Only F program.
Only FAD program.
F and FAD programmed
Rapid traverse if G0 is active, otherwise with the old or the new F word
Velocity of the previous block (old F word)
With FAD program. infeed velocity
With F program. new modally effective velocity
Velocities in WAB subblocks when approaching with G341



Reading positions

Points P_3 and P_4 can be read in the WCS as a system variable during approach.

- $P_APR: reading P
- $_3$ (initial point)
- $P_AEP: reading P
- $_4$ (contour starting point)
- $P_APDV: read whether $P_APR and $P_AEP contain valid data

3.10.4.2 Soft approach and retraction with extended retraction strategies (G460, G461, G462):

In certain special geometrical situations, special extended approach and retraction strategies, compared with the previous implementation with activated collision detection for the approach and retraction block, are required in order to activate or deactivate tool radius compensation. A collision detection can result, for example, in a section of the contour not being completely machined, see following figure:

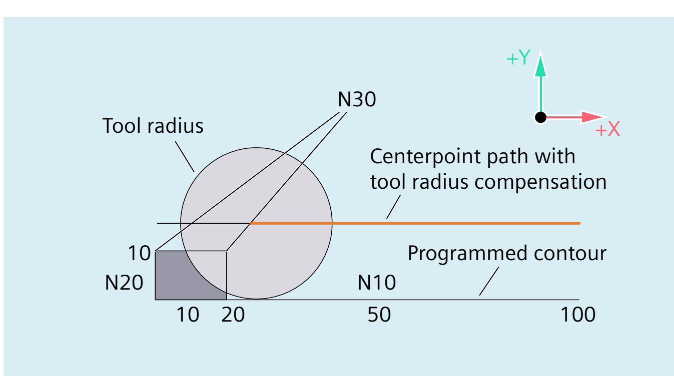


Figure 3-23 Retraction behavior with G460

Syntax

```
G460
G461
G462
```

Meaning

G460:	As previously (activation of the collision detection for the approach and retraction block).
G461:	Insertion of a circle in the TRC block, if it is not possible to have an intersection whose center point is in the end point of the uncorrected block, and whose radius is the same as the tool radius. Up to the intersection, machining is performed with an **auxiliary circle** around the contour end point (i.e. up to the end of the contour).
G462:	Insertion of a circle in the TRC block, if it is not possible to have an intersection; the block is extended by its end tangent (default setting). Machining is performed up to the **extension** of the last contour element (i.e. until shortly before the end of the contour).

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.

Examples

Example 1: Retraction behavior with G460

The following example describes only the situation for deactivation of tool radius compensation: The behavior for approach is exactly the same.

Program code	Comment
G42 D1 T1	; Tool radius 20 mm
...	
G1 X110 Y0	
N10 X0	
N20 Y10	
N30 G40 X50 Y50	

Example 2: Approach with G461

Program code	Comment
N10 $TC_DP1[1,1]=120	; Milling tool type
N20 $TC_DP6[1,1]=10	;Tool radius

Program code	Comment
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	

Further information

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.

The control attempts to cut this circle with one of the preceding blocks.

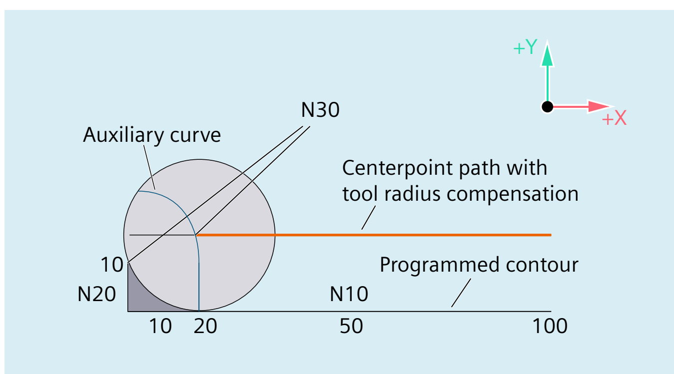


Figure 3-24 Retraction behavior with G461

Collision monitoring CDON, CDOF

If CDOF is active (see section Collision monitoring, CDON, CDOF), the search is aborted when an intersection is found, i.e., the system does not check whether further intersections with previous blocks exist.

If CDON is active, the search continues for further intersections after the first intersection is found.

An intersection point, which is found in this way, is the new end point of a preceding block and the start point of the deactivation block. The inserted circle is used exclusively to calculate the intersection and does not produce a traversing movement.

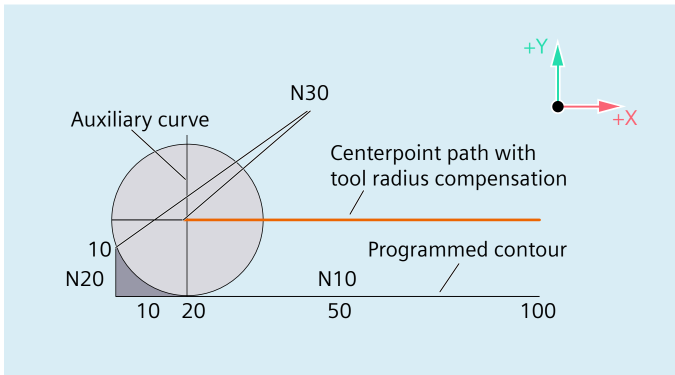
Note

If no intersection is found, alarm 10751 (collision danger) is output.

G462

If no intersection is possible between the last TRC block and a preceding block, a straight line is inserted, on retraction with G462 (initial setting), at the end point of the last block with tool radius compensation (the block is extended by its end tangent).

The search for the intersection is then identical to the procedure for G461.



Retraction behavior with G462 (see example)

With G462, the corner generated by N10 and N20 in the example 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.

Corner behavior with KONT

If KONT is active (travel round contour at start or end point), the behavior differs 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 in front of the contour, the retraction behavior is the same as with NORM. This property does not change even if the last contour block for 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, G460-462 has no effect. If the last traversing block in this situation has no intersection with a preceding block, an intersection with the inserted contour element or with the straight line of the end point of the bypass circle to the programmed endpoint can result.
  If the inserted contour element is a circle (G450), and this forms an interface with the preceding block, this is equal to the interface that would occur with NORM and G461. In general, however, a remaining section of the circle still has to be traversed. For the linear part of the retraction block, no further calculation of intersection is required.
  In the second case, if no interface of the inserted contour element with the preceding blocks is found, the intersection between the retraction straight line and a preceding block is traversed.
  Therefore, a behavior that deviates from G460 can only occur with active G461 or G462 either if NORM is active or the behavior with KONT is geometrically identical to that with NORM.

3.10.5 2 1/2 D tool offset (CUT2D, CUT2DD, CUT2DF, CUT2DFD)

The 2½ D tool radius compensation should be used if, when machining inclined surfaces, the **workpiece** is to be rotated, and not the tool alignment. This function is activated using commands `CUT2D`, `CUT2DD`, `CUT2DF` oder `CUT2DFD`.

Tool length offset

The tool length compensation is always taken into account referred to the machining plane that is not rotated and is fixed in space.

2½ D tool radius compensation for contour tools

2½ D tool radius compensation for contour tools is activated, if, together with `CUT2D`, `CUT2DD`, `CUT2DF` or `CUT2DFD`, one of the two commands `G41` (tool radius compensation left of the contour) or `G42` (tool radius compensation right of the contour) is programmed. It is used for automatic cutting-edge selection in the case of non-axially symmetrical tools that can be used for piece-by-piece machining of individual contour segments.

Note

If 2½ D tool radius compensation is not activated, a contour tool behaves like a standard tool, which only has the first cutting edge.

2½ tool radius compensation referred to a differential tool

2½ D tool radius compensation, referred to a differential tool, is activated using the`CUT2DD` or `CUT2DFD` commands. It should be applied if the programmed contour refers to the center point path of a differential tool, and a tool other than a differential tool is used for machining. When calculating the 2½ D tool radius compensation, only the wear of the radius of the active tool ($TC_DP_15) and the possibly programmed tool offsetOFFN (Page 254) and TOFFR (Page 94) are taken into account. The basic radius ($TC_DP6) of the active tool is **not** taken into account.

Syntax

```
CUT2D
CUT2DD
CUT2DF
CUT2DFD
```

Meaning

`CUT2D`:	Activating the 2½ D radius compensation
`CUT2DD`:	Activating the 2½ D radius compensation referred to a differential tool
`CUT2DF`:	Activating 2½ D radius compensation, tool radius compensation relative to the current frame and/or inclined plane
`CUT2DFD`:	Activating 2½ D radius compensation, tool radius compensation relative to the current frame and/or inclined plane

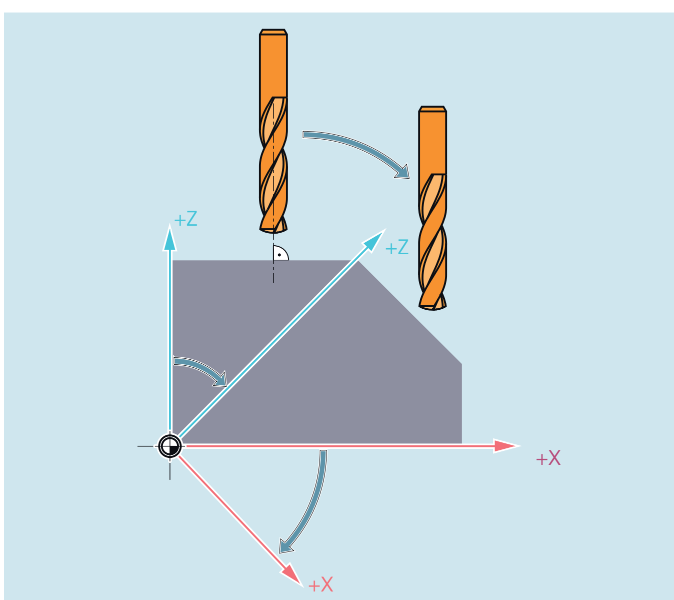
More information

Contour tools

- Enabling
  Tool radius compensation for contour tools is enabled on a channel-specific basis using:
  MD28290 $MC_MM_SHAPED_TOOLS_ENABLE
- Tool type
  Contour tool types are defined on a channel-specific basis using:
  MD20370 $MC_SHAPED_TOOL_TYPE_NO
- Cutting edge
  A number of cutting edges (D numbers) can be assigned to each contour tool in any sequence. The maximum number of cutting edges per tool is parameterized using:
  MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL
  The first cutting edge of a contour tool is the cutting edge, which is selected when activating the tool. If, e.g. in a program, using the commands T3 D5, the fifth cutting edge (D5) of the third tool (T3) is activated, then D5 and the following cutting edges define with one part, or altogether, the contour tool. The cutting edges located before D5 are ignored.

2½ D tool radius compensation without rotating the compensation plane (CUT2D, CUT2DD)

If a frame that contains a rotation is programmed, then for CUT2D or CUT2DD, the plane in which the tool radius compensation (compensation plane) takes place **is not rotated at the same time**. The tool radius compensation is taken into account, referred to the **non rotated** machining plane (G17, G18, G19). The tool length compensation acts relative to the compensation plane.

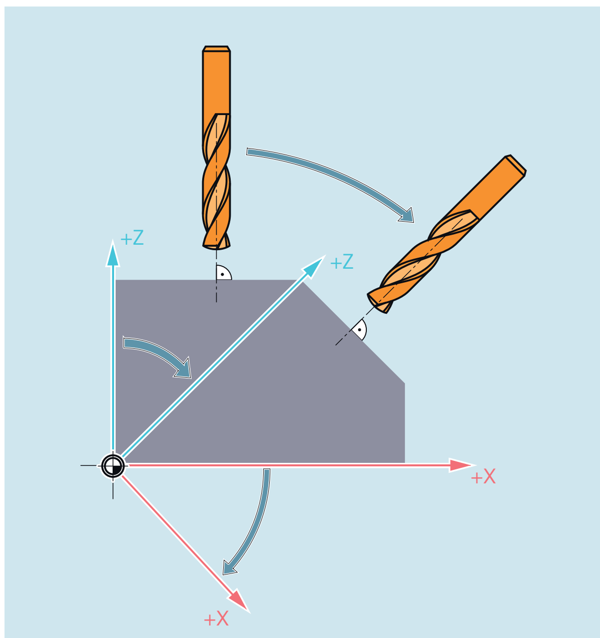


For machining inclined surfaces, the tool offsets must be appropriately defined or calculated based on the functions for "Tool length compensation for tools that can be orientated".

2½ D tool radius compensation with rotation of the compensation plane (CUT2DF, CUT2DFD)

If a frame that contains a rotation is programmed, then for CUT2DF or CUT2DFD, the plane in which the tool radius compensation takes place (compensation plane) **is also rotated**. The tool radius compensation is taken into account, referred to the **rotated** machining plane (G17, G18, G19). However, the tool length compensation still acts relative to the **non-rotated** machining plane.

Requirement: At the machine, the tool orientation must be able to be adjusted perpendicular to the rotated machining plane, and set for machining.



Note

The tool length compensation continues to be active relative to the non-rotated working plane.

Approach and retraction for 2½ D tool radius compensation

The approach/retract response when tool radius compensation is active for the cases where the activation and/or deactivation block does not contain any traversing information (only for 2½ D tool radius compensation with CUT2D or CUT2DF) is defined using the setting data:

SD42494 $SC_CUTCOM_ACT_DEACT_CTRL

The following decimal coding applies:

Thousands position (10³)	**Hundreds position (10²)**	**Tens position (10¹)**	**Units position (10⁰)**
Retraction response for tools without cutting edge position (milling tools)	Retraction response for tools with cutting edge position [1] (turning tools)	Approach response for tools without cutting edge position (milling tools)	Approach response for tools with cutting edge position[1] (turning tools)
[1] Tools with cutting edge position are tools with tool numbers between 400 and 599 (turning and grinding tools), whose cutting edge position has a value that lies between 1 and 8. Turning and grinding tools with cutting edge position 0 or 9 or other, undefined values, are treated in the same way as milling tools.			

If the relevant position has a 1, then approach and retraction are performed, even if only G41/G42 or G40 are present in the block.

Example:

```
N100 X10 Y0
N110 G41
N120 X20
```

If a tool radius of 10 mm is assumed in the example, then position X10 Y10 is approached in block N110.

If the relevant position has a 2, approach and retraction are only performed if at least one geometry axis is programmed in the activation/deactivation block. If the same result as in the example is to be achieved with this setting, then the program must be changed as follows, for example:

```
N100 X10 Y0
N110 G41 X10
N120 X20
```

If axis data X10 is not included in block N110, then activation of the tool radius compensation is delayed by one block, i.e. the activation block would then be block N120.

If the relevant position has a 3, then retraction is not performed in a deactivation block (G40) if only the geometry axis is programmed perpendicular with respect to the compensation plane. In this case, initially the axis moves perpendicular with respect to the compensation plane. The axis then retracts in the compensation plane. In this case, after G40, the block must contain motion information in the compensation plane. Approach movements for values 2 and 3 are identical.

If the relevant position has a 4 (only in the case of a tool without cutting edge position when retracting, i.e. the thousands digit), then for a deactivation block, the next programmed movement is performed if no motion was programmed in it.

Example:

```
N1040 G41 T1 D1
N1050 X20
N1060 X30
N1070 X50                    ; Offset active
```

```
N1080 UP
N1090 X70 F10000            ; Traverse offset with F10000
N1100 G01
N1110 G90
N1120 X90 F12000
N1130 X100
N1140 M30
```

```
N2000 PROC UP
N2010 G40
N2030 RET
```

If the relevant position has a number other than 1, 2 or 3, i.e. especially a value of 0, then in a block that does not include any traversing information, approach and retraction are not performed.

Note

Tools with cutting edge position

If the value of this setting data is changed in a program, it is recommended that a preprocessing stop (STOPRE) is programmed before writing, as otherwise there is a risk that the new value will be used in upstream parts of the program. The inverse case is not critical. This means that if the setting data is written to, subsequent NC blocks will definitely access the changed value.

3.10.6 Keep tool radius compensation constant (CUTCONON, CUTCONOF)

The "Keep tool radius compensation constant" function is used to suppress tool radius compensation for a number of blocks, whereby a difference between the programmed and the actual tool center path traveled set up by tool radius compensation in the previous blocks is retained as the compensation. It can be an advantage to use this method when several traversing blocks are required during line milling in the reversal points, but the contours produced by the tool radius compensation (follow strategies) are not wanted. It can be used independently of the type of tool radius compensation ($2^1/_2$D, 3D face milling, 3D circumferential milling).

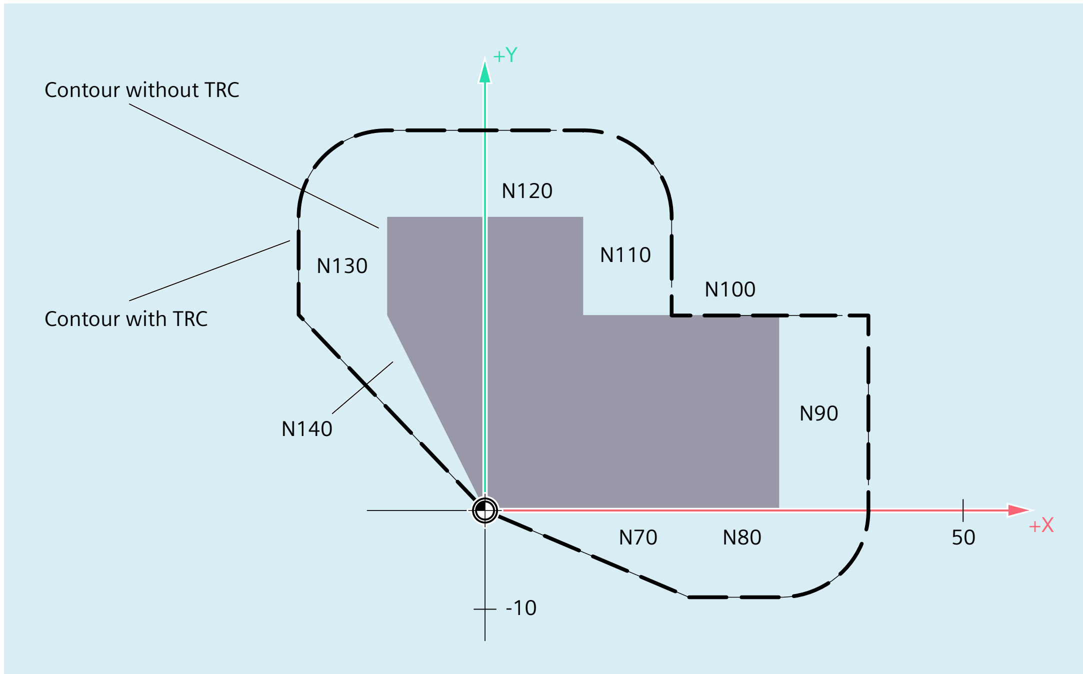
Syntax

```
CUTCONON
CUTCONOF
```

Meaning

`CUTCONON`:	Command to activate the "Keep tool radius compensation constant" function
`CUTCONOF`:	Command to deactivate the "Keep tool radius compensation constant" function

Example



Program code	Comment
N10	; Definition of tool d1.
N20 $TC_DP1[1,1] = 110	; Type
N30 $TC_DP6[1,1]= 10.	; 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	; If required, insert bypass circle when deactivating the compensation suppression.
N120 X-10 CUTCONOF	
N130 Y20 NORM	; No bypass circle when deactivating the TRC.
N140 X0 Y0 G40	
N150 M30	

Further information

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 offset suppression is active are traversed without offset. 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 starting 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 the offset suppression is automatically canceled in the first traversing block that contains a traversing motion in the offset plane and is not such a circle. Vertical circle in this sense can only occur during circumferential milling.

3.10.7 Tools with a relevant cutting edge position

In the case of tools with a relevant tool point direction (turning and grinding tools - tool types 400-599; see Section "Sign evaluation wear"), a change from G40 to G41/G42 or vice-versa is treated as a tool change. If a transformation is active (e.g., TRANSMIT), this leads to a preprocessing stop (decoding stop) and hence possibly to deviations from the intended part contour.

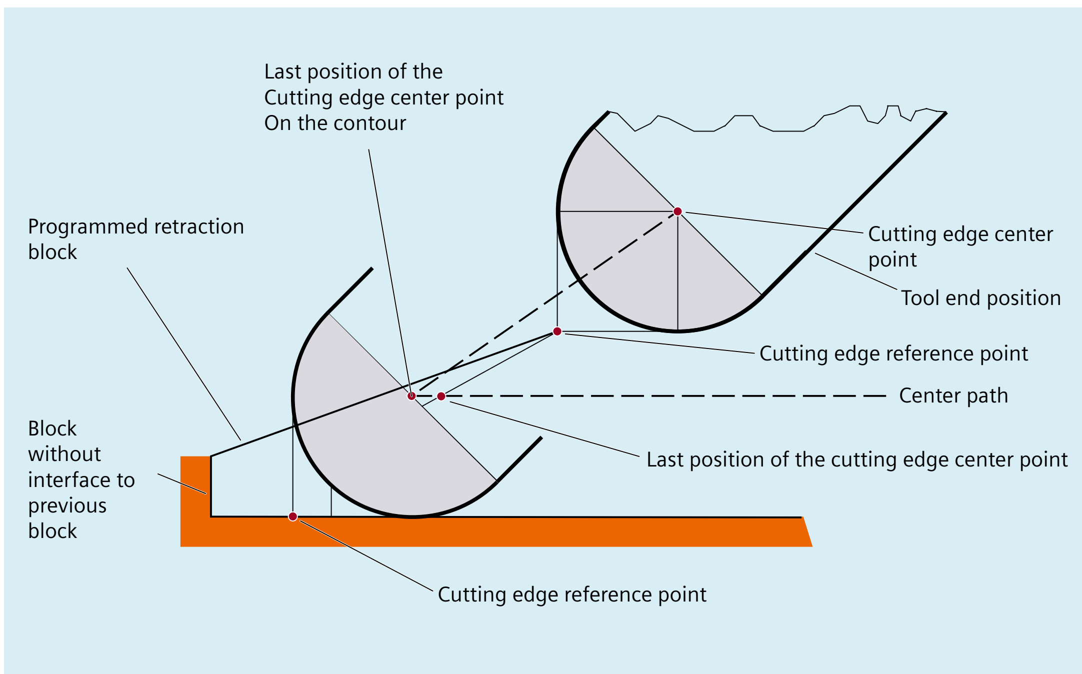
This original functionality changes with regard to:

1. Preprocessing stop on TRANSMIT
2. Calculation of intersection points at approach and retraction with KONT
3. Tool change with active tool radius compensation
4. Tool radius compensation with variable tool orientation at transformation

Further information

The original functionality has been modified as follows:

- A change from G40 to G41/G42 and vice-versa is no longer treated as a tool change. Therefore, a preprocessing stop no longer occurs with TRANSMIT.
- 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.
  On approach and retraction with KONT (tool circumnavigates the contour point, see above subsection "Contour approach and retraction"), superimposition takes place in the linear part block of the approach or retraction motion. The geometric conditions are therefore identical for tools with and without a relevant tool point direction. Deviations from the previous behavior occur only in relatively rare cases where the approach or retraction block does not intersect with an adjacent traversing block, see the following figure:



- 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 tool point direction are therefore not permitted for 3D peripheral milling (an alarm is output).

Note

The subject is irrelevant with respect to face milling as only defined tool types without relevant tool point direction are permitted for this operation anyway. (A tool with a type, which has not been explicitly approved, is treated as a ball end mill with the specified radius. A tool point direction parameter is ignored).

3.11 Path action

3.11.1 Exact stop (G60, G9, G601, G602, G603)

In exact stop traversing mode, all path axes and special axes involved in the traversing motion that are not traversed modally, are decelerated at the end of each block until they come to a standstill.

Activation

Activation takes place via commands from G groups 10, 11 and 12.

Application

Exact stop is used when sharp outside corners have to be machined or inside corners finished to exact dimensions.

Syntax

```
G60/G9 ...
G601/G602/G603, etc.
```

Meaning

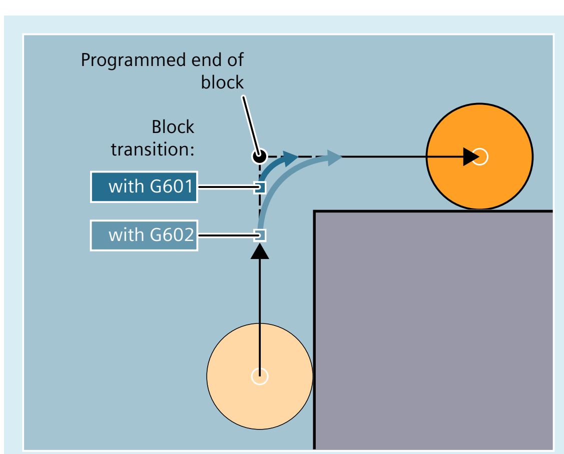
G60	Switch on **modally** effective exact stop G60 is active until it is deactivated by switching on the continuous-path mode (Page 294).
G9	Switch on **non-modally** effective exact stop
G601/G602/G603	Activate exact stop criterion G601 → Exact stop criterion "Exact stop fine" G602 → Exact stop criterion "Exact stop coarse" G603 → Exact stop criterion "Interpolator end" The exact stop criterion is only effective with active exact stop G60 or G9. It defines how exactly the corner point is to be approached and when the transition is to be made to the next block.

Example

```
Program code              Comment
N5 G602                   ; Criterion "Exact stop coarse" selected.
N10 G0 G60 Z...           ; Exact stop modal active.
N20 X... Z...             ; G60 continues to act.
...
N50 G1 G601               ; Criterion "Exact stop fine" selected.
N80 G64 Z...              ; Switchover to continuous-path mode.
...
N100 G0 G9                ; Exact stop acts only in this block.
N110 ...                  ; Continuous-path mode active again.
```

More information

G601 / G602



G601 activates the exact stop criterion "Exact stop fine": The block change is performed as soon as the axis-specific tolerance limits for "Exact stop fine" (MD36010 $MA_STOP_LIMIT_FINE[<Ax>]) are reached for all axes involved in the traversing motion.

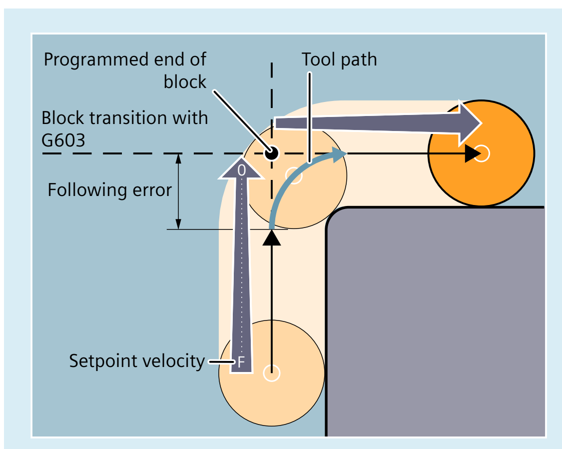
G602 activates the exact stop criterion "Exact stop coarse": The block change is performed as soon as the axis-specific tolerance limits for "Exact stop coarse" (MD36000 $MA_STOP_LIMIT_COARSE[<Ax>]) are reached for all axes involved in the traversing motion.

The movement is decelerated and stopped briefly at the corner point.

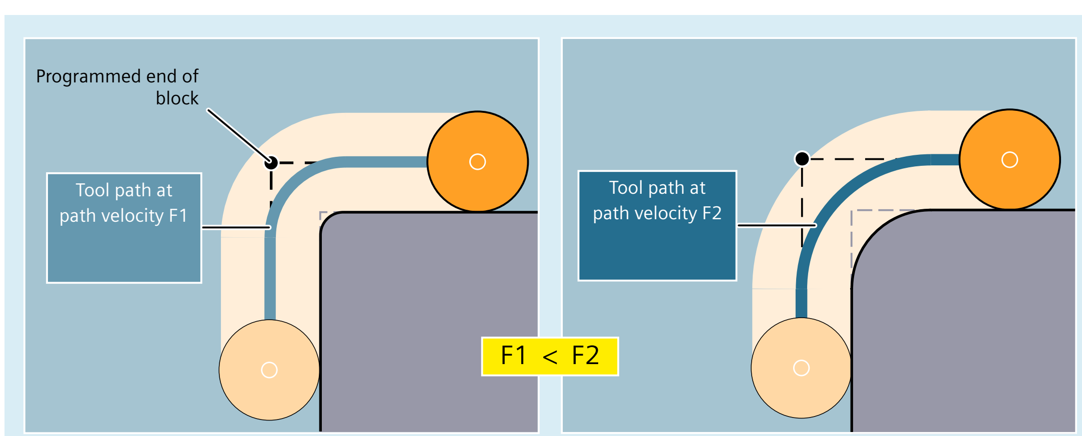
Note

Do not set the limits for the exact stop criteria any tighter than necessary. The tighter the limits, the longer it takes to position and approach the target position.

G603



The block change is initiated when the control has calculated a set velocity of zero for the axes involved. At this time, the actual value is behind by a lag portion. The workpiece corners can now be rounded. The effect depends on the dynamic response of the axes and the path velocity:



Configured exact stop criterion

A channel-specific setting can be made for G0 and the other commands in the first G group indicating that contrary to the programmed exact stop criterion, a preset criterion should be used automatically (see the machine manufacturer's specifications).

3.11.2 Continuous-path mode (G64, G641, G642, G643, G644, G645, G646, ADIS, ADISPOS)

In continuous-path mode, the path velocity at the end of the block (for the block change) is not decelerated to a level which would permit the fulfillment of an exact stop criterion. The objective of this mode is, in fact, 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 "Look-head" function is also activated when continuous-path mode is selected.

Continuous-path mode with smoothing facilitates the tangential shaping and/or smoothing of angular block transitions caused by local changes in the programmed contour.

Properties

Continuous-path mode causes:

- Rounding of the contour
- Shorter machining times
  (due to missing braking and acceleration processes that are required to fulfill the exact stop criterion)
- Better cutting conditions
  (due to the more constant velocity process)

Activation

The continuous-path mode is selected and activated via the commands of G group 10.

Application

Continuous-path mode is suitable:

- If a contour must be traversed as quickly as possible (e.g. with rapid traverse).
- If 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 needs to be traversed precisely.
- If an absolutely constant velocity is required.

Note

Continuous-path mode is interrupted by blocks which trigger a preprocessing stop implicitly, e.g. due to:

- Access to specific machine status data ($A...)
- Auxiliary function outputs

Syntax

```
G64/G646 ...
G641 ADIS=.../ADISPOS=...
G642 .../G643 .../G644 .../G645 ...
```

Meaning

<table>
<tr><td>G64</td><td colspan="2">Continuous-path mode with reduced velocity as per the overload factor
At non-tangential block transitions, the path velocity is reduced to such an extent that the non-tangential block transition can be bypassed in an interpolator cycle while maintaining the acceleration limit and taking an overload factor into account.</td></tr>
<tr><td>G641</td><td colspan="2">Continuous-path mode with smoothing as per distance criterion</td></tr>
<tr><td rowspan="4">ADIS=.../
ADISPOS=...</td><td colspan="2">Path criterion for G641</td></tr>
<tr><td>ADIS = ...</td><td>Path criterion for path functions G1, G2, G3, etc.</td></tr>
<tr><td>ADISPOS = ...</td><td>Path criterion for rapid traverse G0</td></tr>
<tr><td colspan="2">The path criterion (= smoothing clearance) ADIS or ADISPOS describes the maximum path the smoothing block may cover before the end of the block, or the distance after the end of block within which the smoothing block must be terminated respectively. If no ADIS/ADISPOS is programmed, then the value "zero" applies and thus the traversing behavior as with G64. The smoothing clearance is automatically reduced (by up to 36%) for short traversing distances.</td></tr>
<tr><td>G642</td><td colspan="2">Continuous-path mode with smoothing within the defined tolerances
In this mode, under normal circumstances smoothing takes place within the maximum permissible path deviation. However, instead of these axis-specific tolerances, observation of the maximum contour deviation (contour tolerance) or the maximum angular deviation of the tool orientation (orientation tolerance) can be configured.
Note:
Expansion to include contour and orientation tolerance is only supported on systems featuring the "Polynomial interpolation" option.</td></tr>
<tr><td>G643</td><td colspan="2">Continuous-path mode with smoothing within the defined tolerances (block-internal)
G643 differs from G642 in that is not used to generate a separate smoothing block; instead, axis-specific block-internal smoothing motions are inserted. The smoothing clearance can be different for each axis.</td></tr>
<tr><td>G644</td><td colspan="2">Continuous-path mode with smoothing with maximum possible dynamic response
Note:
G644 is not available with an active kinematic transformation. The system switches internally to G642.</td></tr>
</table>

`G645`	Continuous-path mode with smoothing and tangential block transitions within defined tolerances Smoothing of corners is the same as for G642. With G645, smoothing blocks are also only generated on tangential block transitions if the curvature of the original contour exhibits a jump in at least one axis.
`G646`	Extended continuous-path mode with reduced velocity as per the overload factor (option) Same as G64, except that the velocity reduction according to the overload factor is effective for several IPO cycles. This shortens the machining time when bypassing non-tangential block transitions. **Note:** If the option is available on the machine, depending on the setting in the machine data MD20492 $MC_EXTENDED_SMOOTHING_MODE, the extended continuous-path mode can also be effective for G64.

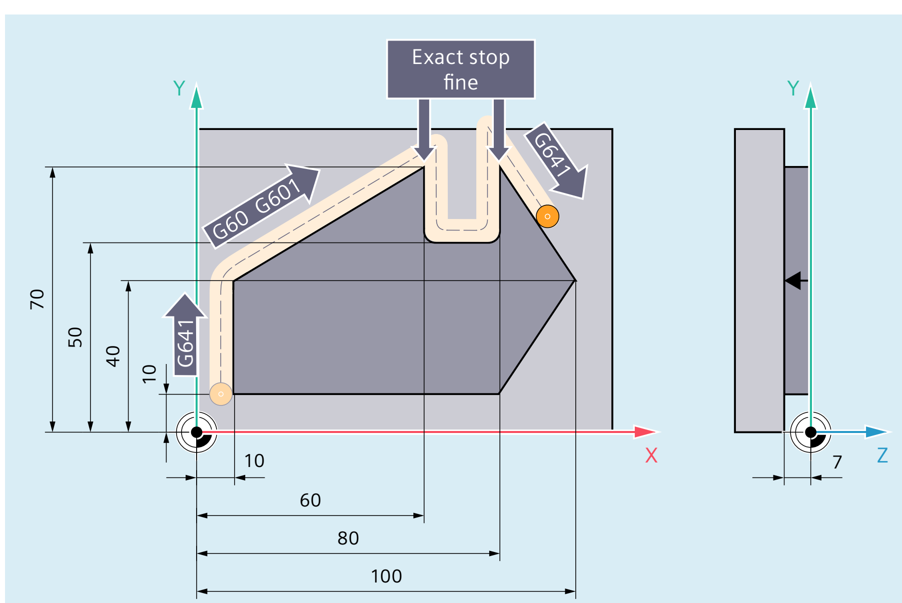
Note

Smoothing cannot be used as a substitute for corner rounding (RND). The user should not make any assumptions with respect to the appearance of the contour within the smoothing area. The type of smoothing can depend on dynamic conditions, e.g. the path velocity. Smoothing on the contour is therefore only practical with small ADIS values. RND must be used if a defined contour is to be traversed at the corner.

Note

If a smoothing motion initiated by G641, G642, G643, G644 or G645 is interrupted, the starting or end point of the original traversing block (as appropriate for REPOS mode) will be used for subsequent repositioning (REPOS), rather than the interruption point.

Example



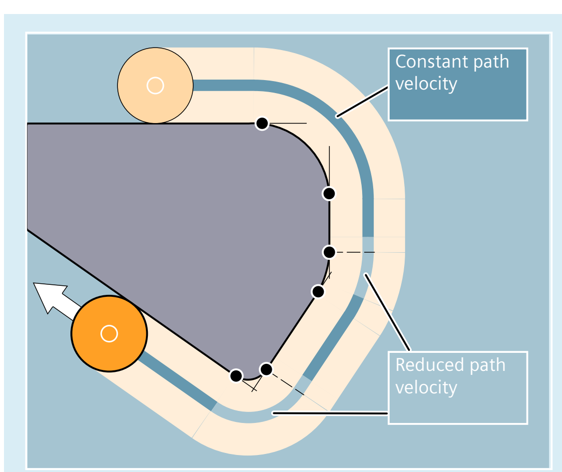
The two outside corners on the groove are to be approached exactly. Otherwise machining should be performed in continuous-path mode.

Program code	Comment
N05DIAMOF	; Radius as dimension
N10 G17 T1 G41 G0 X10 Y10 Z2 S300 M3	; Approach starting position, activate spindle, path compensation.
N20 G1 Z-7 F8000	; Feed in tool.
N30 G641 ADIS=0.5	; Contour transitions are smoothed.
N40 Y40	
N50 X60 Y70 G60 G601	; Approach position exactly with exact stop fine.
N60 Y50	
N70 X80	
N80 Y70	
N90 G641 ADIS=0.5 X100 Y40	; Contour transitions are smoothed.
N100 X80 Y10	
N110 X10	
N120 G40 G0 X-20	; Deactivate path compensation
N130 Z10 M30	; Retract tool, end of program.

More information

Continuous-path mode G64 and G646

In continuous-path mode, the tool travels across tangential contour transitions with as constant a path velocity as possible (no deceleration at block boundaries). LookAhead deceleration is applied before corners and blocks with exact stop.



Corners are also traversed at a constant velocity. In order to minimize the contour error, the velocity is reduced according to an acceleration limit and an overload factor.

Note

The extent of smoothing the contour transitions depends on the feedrate and the overload factor. The overload factor can be set in MD32310 $MA_MAX_ACCEL_OVL_FACTOR. In the case of extended continuous-path mode, additionally from the number of IPO cycles (MD20493 $MC_G64_NUM_IPO) in which the overload factor is effective.

Setting MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS means that block transitions will always be rounded irrespective of the set overload factor.

The following points should be noted in order to prevent an undesired stop in path motion (relief cutting):

- Auxiliary functions, which are enabled after the end of the motion or before the next motion, interrupt the continuous-path mode (exception: fast auxiliary functions).
- Positioning axes always traverse according to the exact stop principle, positioning window fine (as for G601). If an NC block has to wait for positioning axes, continuous-path mode is interrupted on the path axes.

However, intermediate blocks containing only comments, calculation blocks or subprogram calls do not affect continuous-path mode.

Note

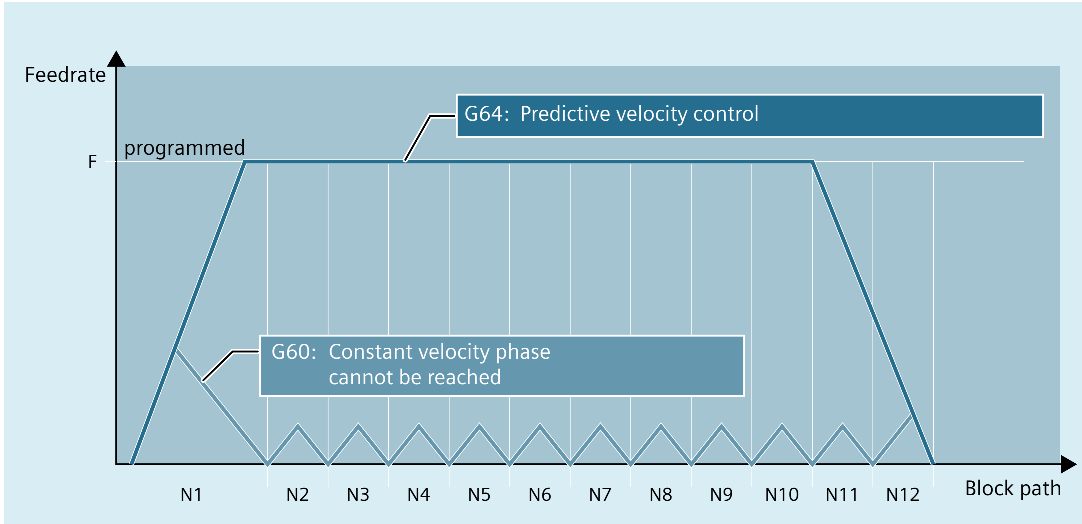
If FGROUP does not contain all the path axes, there is often a step change in the velocity at block transitions for those axes excluded from FGROUP; the control limits this change in velocity to the permissible values set in MD32300 $MA_MAX_AX_ACCEL and MD32310 $MA_MAX_ACCEL_OVL_FACTOR. This braking operation can be avoided through the application of a smoothing function, which smooths the specific positional interrelationship between the path axes.

LookAhead predictive velocity control

In continuous-path mode, the control automatically determines the velocity control for several NC blocks in advance. This enables acceleration and deceleration across multiple blocks with almost tangential transitions.

Look Ahead is particularly suitable for the machining of motion sequences comprising short traverse paths with high path feedrates.

The number of NC blocks included in the Look Ahead calculation can be defined in machine data.



Continuous-path mode with smoothing as per distance criterion (G641)

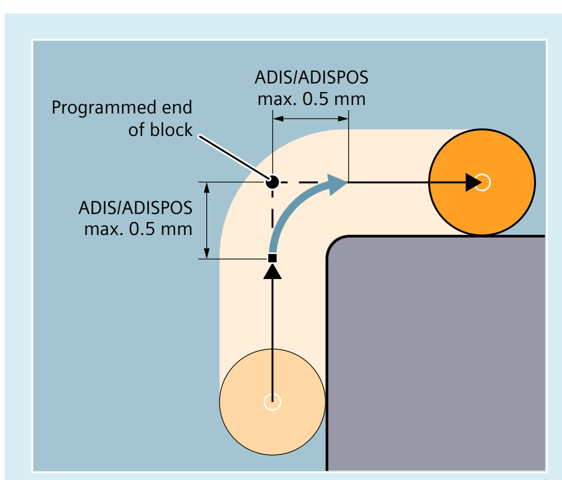
With G641, the control inserts transition elements at contour transitions. The smoothing clearance ADIS (or ADISPOS for G0) specifies the maximum extent to which corners can be smooth (rounded). Within this smoothing clearance, the control can ignore the configured path and replace it with a dynamically optimized path.

Disadvantage: Only one ADIS value is available for all axes.

The effect of G641 is similar to RNDM; however, it is not restricted to the axes of the working plane.

Just the same as G64, G641 works with lookahead predictive velocity control. Smoothing blocks with a high degree of curvature are approached with a reduced velocity.

Example:



Program code	Comment
N10 G641 ADIS=0.5 G1 X... Y...	; The smoothing block must begin no more than 0.5 mm before the programmed end of the block and must finish 0.5 mm after the end of the block. This setting remains modal.

Note

Smoothing cannot and should not replace the functions for defined smoothing (RND, RNDM, ASPLINE, BSPLINE, CSPLINE).

Smoothing with axial precision with G642

With G642, smoothing does not take place within a defined ADIS range; instead, the axial tolerances defined with MD33100 $MA_COMPRESS_POS_TOL are applied. The smoothing clearance is determined based on the shortest smoothing path of all axes. This value is taken into account when generating a smoothing block.

Block-internal smoothing with G643

The maximum deviations from the precise contour in the case of smoothing with G643 are defined for each axis using machine data MD33100 $MA_COMPRESS_POS_TOL.

G643 is not used to generate a separate smoothing block, but axis-specific block-internal smoothing motion is inserted. In the case of G643, the smoothing clearance of each axis can be different.

Smoothing with contour and orientation tolerance with G642/G643

MD20480 $MC_SMOOTHING_MODE can be used to configure smoothing with G642 and G643 so that instead of axis-specific tolerances, a contour tolerance and an orientation tolerance can be applied.

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 setting data can be programmed in the NC program; this means that it can be specified differently for each block transition. Very different specifications for the contour tolerance and the tolerance of the tool orientation can only take effect with G643.

Note

Expansion to include contour and orientation tolerance is only supported on systems featuring the "Polynomial interpolation" option.

Note

An orientation transformation must be active for smoothing within the orientation tolerance.

Smoothing with greatest possible dynamic response in G644

Smoothing with maximum possible dynamic response is configured in the thousands place with MD20480 $MC_SMOOTHING_MODE.

Value	Meaning
0	Specification of maximum axial deviations with: MD33100 $MA_COMPRESS_POS_TOL
1	Specification of maximum smoothing path by programming: ADIS=... or ADISPOS=...
2	Specification of the maximum possible frequencies of each axis occurring in the smoothing range with: MD32440 $MA_LOOKAH_FREQUENCY The smoothing range is defined such that no frequencies in excess of the specified maximum can occur during smoothing motion.
3	When smoothing with G644, neither the tolerance nor the smoothing clearance is monitored. Each axis traverses around a corner with the maximum possible dynamic response. With SOFT, both the maximum acceleration and the maximum jerk of each axis is maintained. With the BRISK command, the jerk is not limited; instead, each axis travels at the maximum possible acceleration.

Smoothing of tangential block transitions with G645

With G645, the smoothing motion is defined 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 smoothing behavior is the same as with G642.

No intermediate smoothing blocks

An intermediate smoothing block is not inserted in the following cases:

- The axis stops between the two blocks.
  This occurs when:
  - The following block contains an auxiliary function output before the motion.
  - The following block does not contain a path motion.
  - An axis is traversed for the first time as a path axis for the following block when it was previously a positioning axis.
  - An axis is traversed for the first time as a positioning axis for the following block when it was previously a path axis.
  - The previous block traverses geometry axes and the following block does not.
  - The following block traverses geometry axes and the previous block does not.
  - Before tapping, the following block uses G33 as preparatory function and the previous block does not.
  - A change is made between BRISK and SOFT.
  - Axes involved in the transformation are not completely assigned to the path motion (e.g. for oscillation, positioning axes).
- The smoothing block would slow down part program execution.
  This occurs:
  - Between very short blocks.
    Since each block requires at least one interpolator clock cycle, the added intermediate block would double the machining time.
  - If a block transition G64 (continuous-path mode without smoothing) can be traversed without a reduction in velocity.
    Smoothing would increase the machining time. This means that the value of the permitted overload factor (MD32310 $MA_MAX_ACCEL_OVL_FACTOR) affects whether a block transition is rounded or not. The overload factor is only taken into account for smoothing with G641 / G642. The overload factor has no effect in the case of smoothing with G643 (this behavior can also be set for G641 and G642 by setting MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS to TRUE).

- Smoothing is not parameterized.
  This occurs when:
  - for G641 in G0 blocks, ADISPOS=0 (preassignment!).
  - for G641 in non G0 blocks, ADISPOS=0 (preassignment!).
  - for G641, for transition from G0 to non-G0 or non-G0 to G0, the lower of the values from ADISPOS and ADIS applies.
  - for G642/G643, all axis-specific tolerances are zero.
- The block does not contain traversing motion (zero block).
  This occurs when:
  - Synchronized actions are active.
    Normally, the Interpreter eliminates zero blocks. However, if synchronized actions are active, this zero block is included and also executed. In so doing, an exact stop is initiated corresponding to the active programming. This allows the synchronized action to also switch.
  - Zero blocks are generated by program jumps.

Continuous-path mode in rapid traverse G0

Also for rapid traverse motion, one of the listed functions G60/G9 or G64 - respectively G641 - G645 - must be specified. Otherwise, the default in the machine data is used.

3.12 Coordinate transformations (frames)

3.12.1 Frames

Frame

The frame is a self-contained arithmetic rule that transforms one Cartesian coordinate system into another Cartesian coordinate system.

Basic frame (basic offset)

The basic frame describes coordinate transformation from the basic coordinate system (BCS) to the basic zero system (BZS) and has the same effect as settable frames.

See Basic coordinate system (BCS) (Page 38).

Settable frames

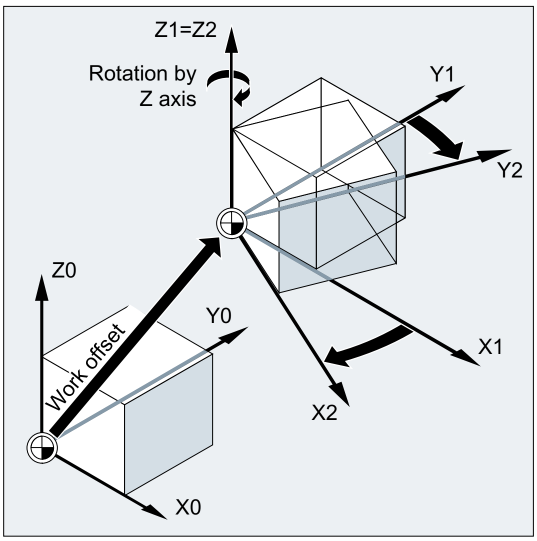
Settable frames are the configurable zero offsets which can be called from within any NC program with the `G54` to `G57` and `G505` to `G599` commands. The offset values are predefined by the user, and stored in the zero offset memory on the controller. They are used to define the settable zero system (SZS).

See:

- Settable zero system (SZS) (Page 40)
- Settable work offset (G54 ... G59, G507 ... G599, G53, G500, SUPA, G153) (Page 152)

Programmable frames

Sometimes it is useful or necessary within an NC program, to move the originally selected workpiece coordinate system (or the "settable zero system") to another position and, if required, to rotate it, mirror it and/or scale it. This can be achieved using programmable frames.



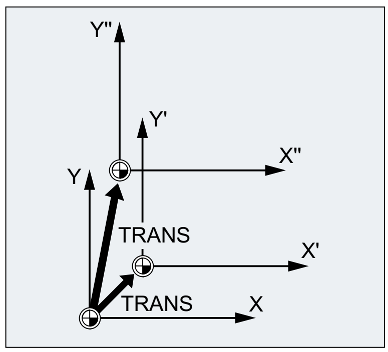
See Frame instructions (Page 305).

3.12.2 Frame instructions

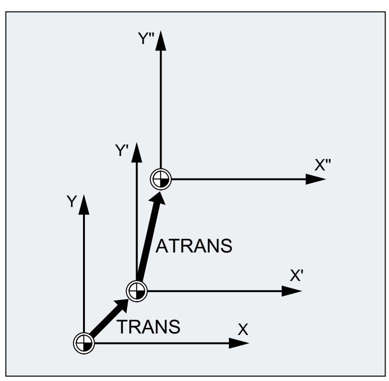
Function

The statements for programmable frames apply in the current NC program. They function as either additive or substitute elements:

- Substitute statement
  Deletes all previously programmed frame statements. The reference is provided by the last settable zero offset called (G54 to G57, G505 to G599).

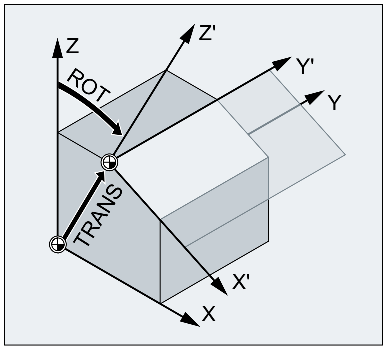


- Additive statement
  Appended to existing frames. The reference is provided by the currently set workpiece zero or the last workpiece zero programmed with a frame statement.



Application example

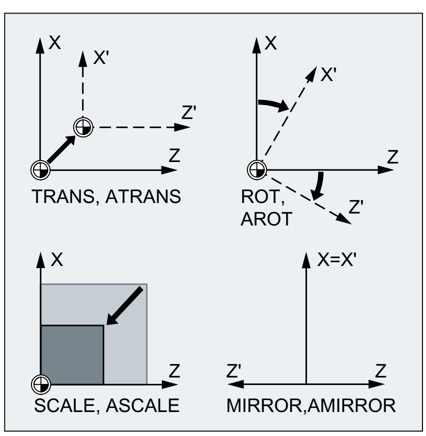
1. Move the zero point of the workpiece coordinate system (WCS).
2. Rotate the workpiece coordinate system (WCS) to orientate a plane parallel to the desired work plane.

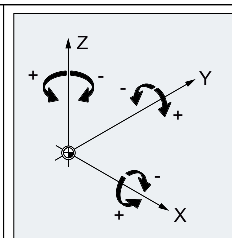


Syntax

Substituting statements	Additive statements
`TRANS X… Y… Z…`	`ATRANS X… Y… Z…`
`ROT X… Y… Z…`	`AROT X… Y… Z…`
`ROT RPL=…`	`AROT RPL=…`
`ROTS/CROTS X... Y...`	`AROTS X... Y...`
`SCALE X… Y… Z…`	`ASCALE X… Y… Z…`
`MIRROR X0/Y0/Z0`	`AMIRROR X0/Y0/Z0`

Meaning



<table>
<tr><td><code>TRANS/ATRANS</code>:</td><td colspan="3">Workpiece coordinate system offset in the direction of the specified geometry axis or axes</td></tr>
<tr><td rowspan="11"><code>ROT/AROT</code>:</td><td colspan="3">Workpiece coordinate system rotation:
• By linking individual rotations around the specified geometry axis or axes
or
• Around the angle <code>RPL=...</code> in the current working plane (<code>G17</code>/<code>G18</code>/<code>G19</code>)</td></tr>
<tr><td>Direction of rotation:</td><td colspan="2">
</td></tr>
<tr><td rowspan="2">Rotation sequence:</td><td>With RPY notation:</td><td>Z, Y', X''</td></tr>
<tr><td>With Euler angle:</td><td>Z, X', Z''</td></tr>
<tr><td rowspan="7">Range of values:</td><td colspan="2">The angles of rotation are only defined unambiguously in the following ranges:</td></tr>
<tr><td rowspan="3">With RPY notation:</td><td>-180 ≤ x ≤ 180</td></tr>
<tr><td>-90 < y < 90</td></tr>
<tr><td>-180 ≤ z ≤ 180</td></tr>
<tr><td rowspan="3">With Euler angle:</td><td>0 ≤ x < 180</td></tr>
<tr><td>-180 ≤ y ≤ 180</td></tr>
<tr><td>-180 ≤ z ≤ 180</td></tr>
<tr><td><code>ROTS/AROTS</code>:</td><td colspan="3">Workpiece coordinate system rotation by means of the specification of solid angles
The orientation of a plane in space is defined unambiguously by specifying two solid angles. Therefore, up to two solid angles may be programmed:
<code>ROTS/AROTS X... Y... / Z... X... / Y... Z...</code></td></tr>
<tr><td><code>CROTS</code>:</td><td colspan="3"><code>CROTS</code> works in the same way as <code>ROTS</code> but refers to the valid frame in the database.</td></tr>
<tr><td><code>SCALE/ASCALE</code>:</td><td colspan="3">Scaling in the direction of the specified geometry axis or axes to increase/reduce the size of a contour</td></tr>
<tr><td rowspan="2"><code>MIRROR/AMIRROR</code>:</td><td colspan="3">Workpiece coordinate system mirroring by means of mirroring (direction change) the specified geometry axis</td></tr>
<tr><td>Value:</td><td colspan="2">Freely selectable (in this case: "0")</td></tr>
</table>

Supplementary conditions

- Frame statements must be programmed in a separate NC block.
- Frame statements can be used individually or combined as required.

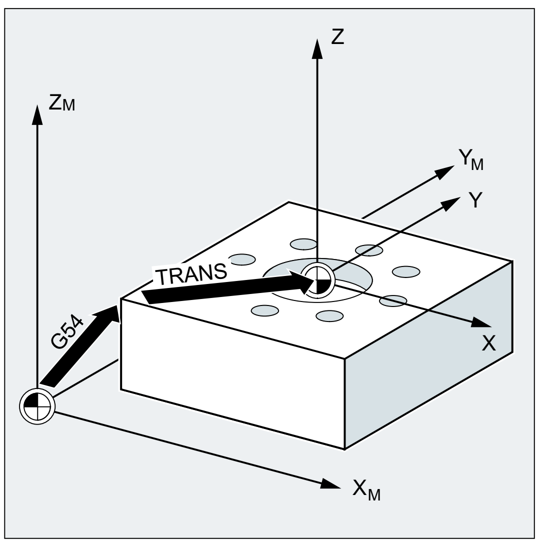
- Frame statements are executed in the programmed sequence.
- Additive statements are frequently used in subprograms. The basic statements defined in the main program are not lost after the end of the subprogram if the subprogram has been programmed with the SAVE attribute.

3.12.3 Programmable work offset (TRANS, ATRANS)

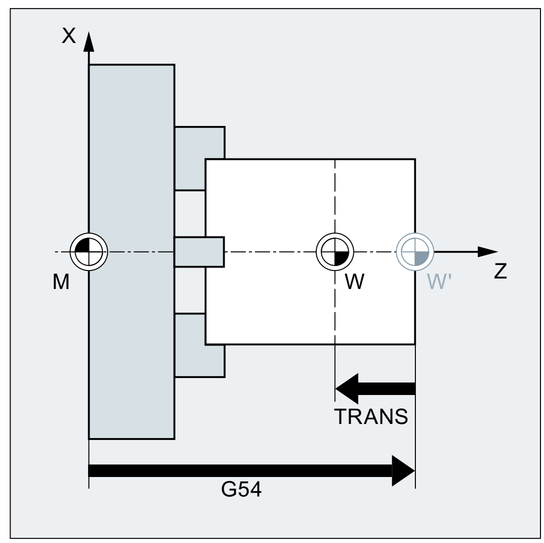
The `TRANS` command moves the WCS absolutely based on the SZS created with a settable work offset (G54 ... G57, G505 ... G599).

The `ATRANS` command moves additively the WCS created with `TRANS`.

Milling:



Turning:



Syntax

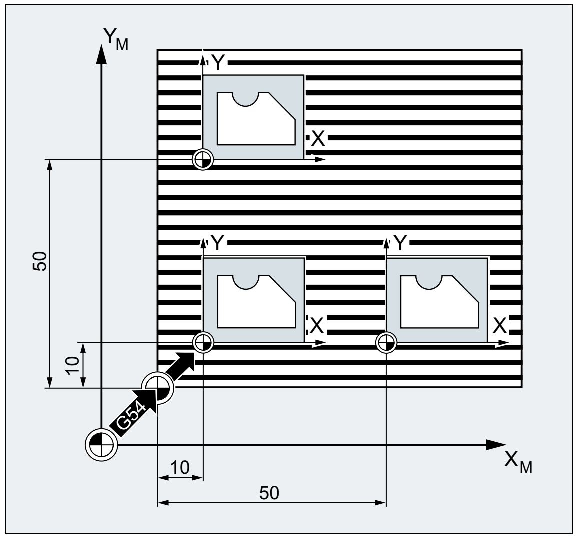
```
TRANS X… Y… Z…
ATRANS X… Y… Z…
```

Meaning

`TRANS:`	Absolute offset of the WCS with reference to the workpiece zero (SZS) set with a settable work offset (G54 ... G57, G505 ... G599).	
	Alone in the block:	yes
`ATRANS:`	Additive zero offset of the WCS with reference to the parameterized workpiece zero set with `TRANS`	
	Alone in the block:	yes
`X... Y... Z...:`	Offset values in the direction of the specified geometry axes	

Examples

Example 1: Milling



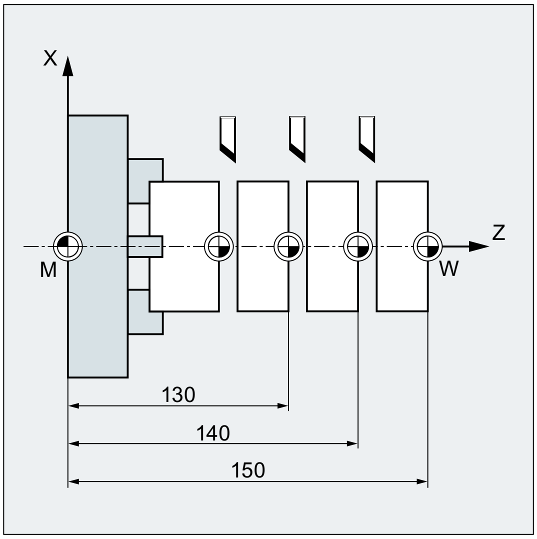
With this workpiece, the shapes shown recur in a program.

The machining sequence for this shape is stored in a subprogram.

Zero offset is used to set the workpiece zeros required in each case and then call the subprogram.

Program code	Comment
N10 G1 G54	; Working plane X/Y, workpiece zero
N20 G0 X0 Y0 Z2	; Approach starting point
N30 TRANS X10 Y10	; Absolute offset
N40 L10	; Subprogram call
N50 TRANS X50 Y10	; Absolute offset
N60 L10	; Subprogram call
N70 M30	; End of program

Example 2: Turning



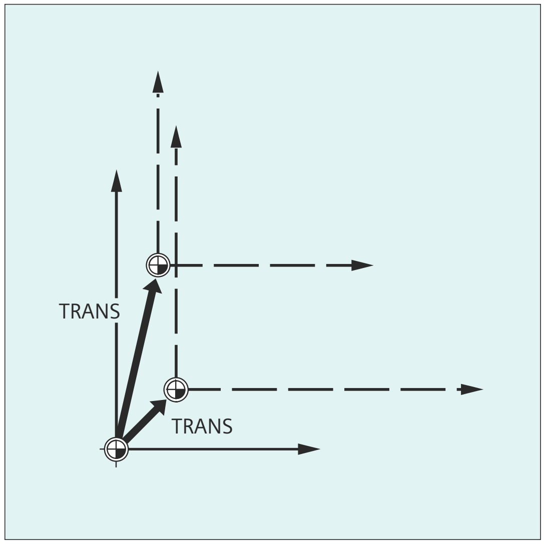
Program code	Comment
...	
N10 TRANS X0 Z150	; Absolute offset
N15 L20	; Subprogram call
N20 TRANS X0 Z140 (or ATRANS Z-10)	; Absolute offset
N25 L20	; Subprogram call
N30 TRANS X0 Z130 (or ATRANS Z-10)	; Absolute offset
N35 L20	; Subprogram call
...	

Further information

TRANS X... Y... Z...

Translation through the offset values programmed in the specified axis directions (path, synchronized axes and positioning axes). The reference is provided by the last settable work offset called (G54 to G57, G505 to G599).

NOTICE
No original frame
The `TRANS` command resets all frame components of the previously activated programmable frame.

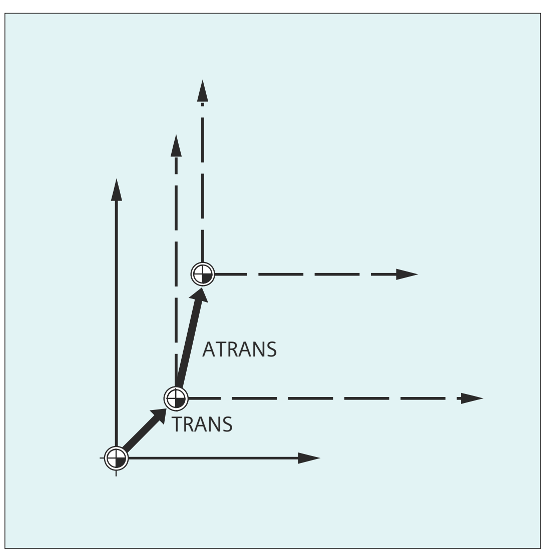


Note

ATRANS can be used to program an offset to be added to existing frames.

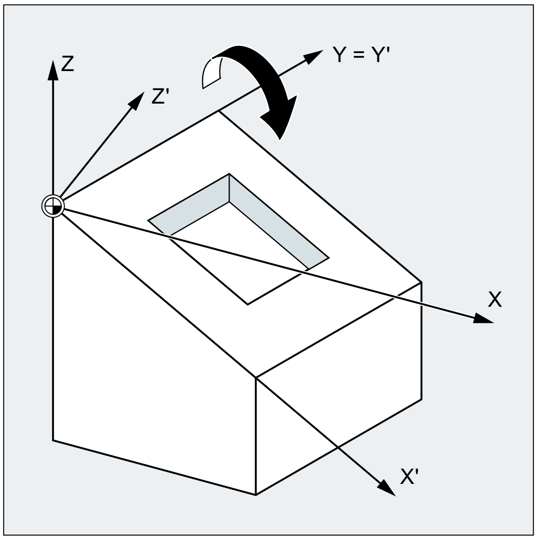
ATRANS X... Y... Z...

Translation through the offset values programmed in the specified axis directions. The currently set or last programmed zero point is used as the reference.



3.12.4 Programmable rotation (ROT, AROT, RPL)

The workpiece coordinate system can be rotated in space with the ROT/AROT statements. The statements refer exclusively to the programmable frame $P_PFRAME.



Syntax

```
ROT <1st GeoAx><angle> <2nd GeoAx><angle> <3rd GeoAx><angle>
ROT RPL=<angle>
AROT <1st GeoAx><angle> <2nd GeoAx><angle> <3rd GeoAx><angle>
AROT RPL=<angle>
```

Note

The rotations of the workpiece coordinate system are performed via Euler angles.

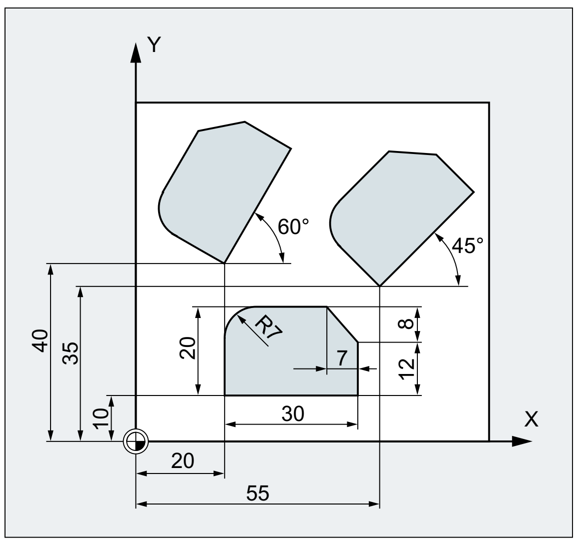
Meaning

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 GeoAx>:	Identifier of the nth geometry axis around which rotation is to be performed with the specified angle. The value 0° is implicitly set as angle of rotation for a geometry axis that has not been programmed.	

`RPL`:	Rotation around the geometry axis perpendicular to the active plane (`G17`, `G18`, `G19`) by the specified angle	
	Reference frame:	Programmable frame `$P_PFRAME`
	Reference point:	Zero point of the current workpiece coordinate system set with `G54` ... `G57`, `G505` ... `G599`
`<Angle>`	Angle specification in degrees.	
	Value range:	-360° ≤ angle ≤ 360°

Examples

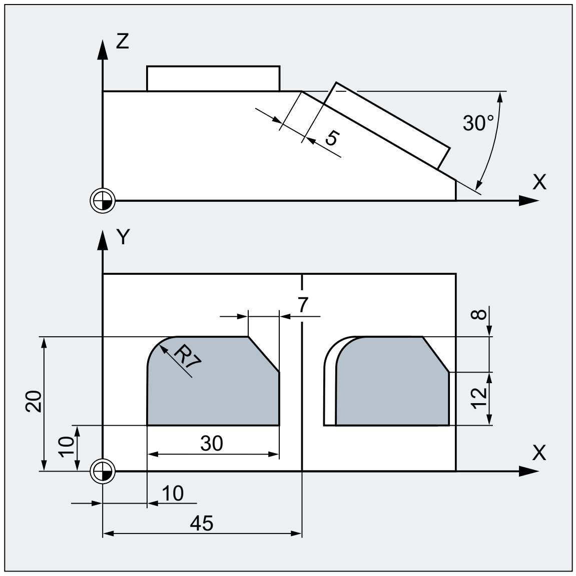
Example 1: Rotation in the G17 plane



With this workpiece, the shapes shown recur in a program. In addition to the work offset, rotations have to be performed, as the shapes are not arranged paraxially.

Program code	Comment
`N10 G17 G54`	`; Working plane X/Y, workpiece zero`
`N20 TRANS X20 Y10`	`; Absolute offset`
`N30 L10`	`; Subprogram call`
`N40 TRANS X55 Y35`	`; Absolute offset`
`N50 AROT RPL=45`	`; Additive rotation around the Z axis perpendicular`
	`to the G17 plane through 45°`
`N60 L10`	`; Subprogram call`
`N70 TRANS X20 Y40`	`; Absolute offset` `(resets all previous offsets)`
`N80 AROT RPL=60`	`; Additive rotation around the Z axis perpendicular`
	`; to the G17 plane through 60°`
`N90 L10`	`; Subprogram call`
`N100 G0 X100 Y100`	`; Retraction`
`N110 M30`	`; End of program`

Example 2: Spatial rotation around the Y axis

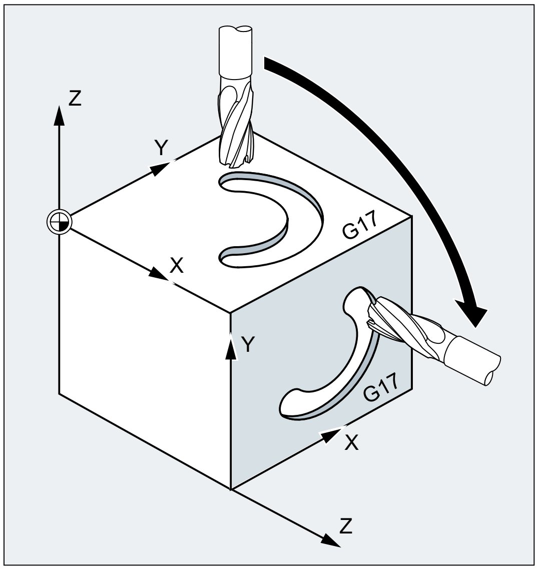


In this example, paraxial and inclined workpiece surfaces are to be machined in a clamping.

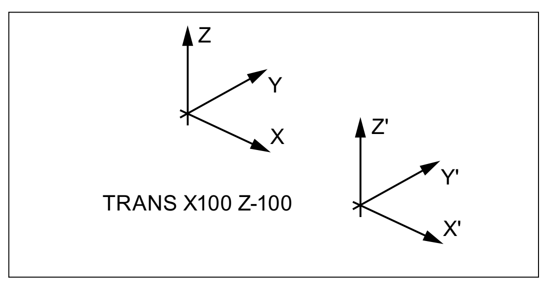
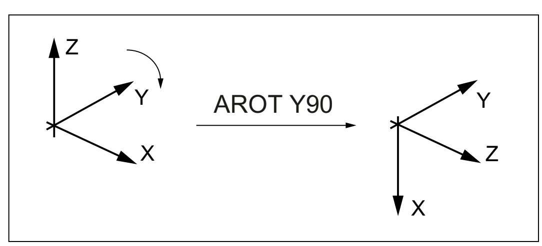
Condition:
The tool must be aligned perpendicular to the inclined surface in the rotated Z direction.

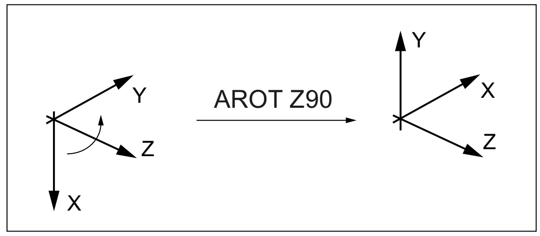
Program code	Comment
N10 G17 G54	; Working plane X/Y, workpiece zero
N20 TRANS X10 Y10	; Absolute offset
N30 L10	; Subprogram call
N40 ATRANS X35	; Additive offset
N50 AROT Y30	; Additive rotation around the Y axis
N60 ATRANS X5	; Additive offset
N70 L10	; Subprogram call
N80 G0 X300 Y100 M30	; Retraction, end of program

Example 3: Multi-face machining



In this example, identical shapes are machined in two workpiece surfaces perpendicular to one another via subprograms. In the new coordinate system on the right-hand workpiece surface, infeed direction, working plane and the zero point have been set up as on the top surface. Therefore, the conditions required for the subprogram execution still apply: Working plane `G17`, coordinate plane X/Y, infeed direction Z.

Program code	Comment
`N10 G17 G54`	`; Working plane X/Y, workpiece zero`
`N20 L10`	`; Subprogram call`
`N30 TRANS X100 Z-100`	`; Absolute offset of the WCS`
	
`N40 AROT Y90`	`; Additive rotation of the WCS around Y through 90°`
	

Program code	Comment
N50 AROT Z90	; Additive rotation of the WCS around Z through 90°
	
N60 L10	; Subprogram call
N70 G0 X300 Y100 M30	; Retraction, end of program

Further information

Rotation in the active plane

When programming using `RPL=...`, the WCS is rotated around the axis perpendicular to the active plane.

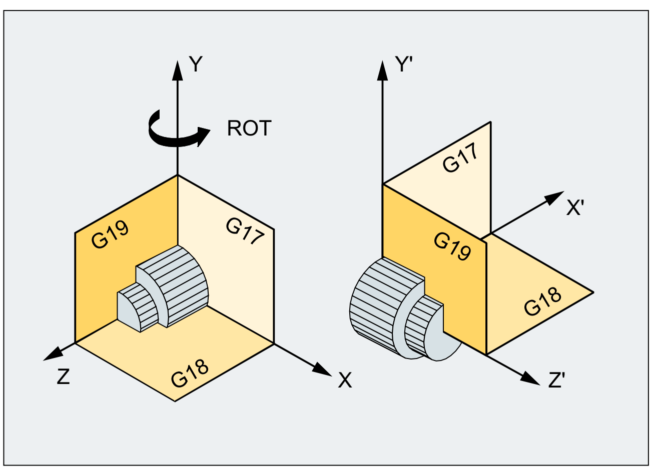
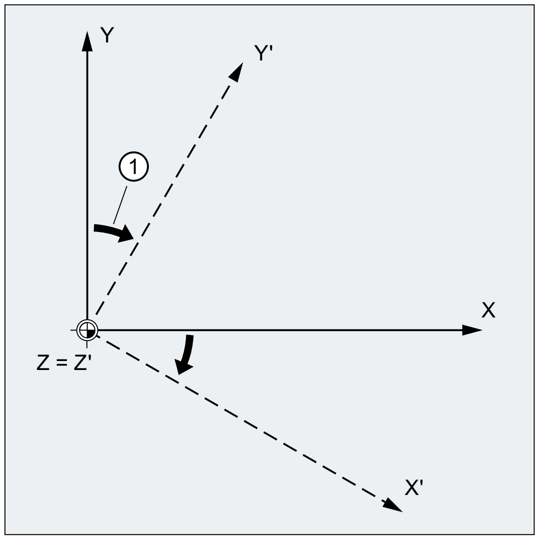


Figure 3-25 Rotation around the Y axis or in the G18 plane

> ⚠ **WARNING**
>
> **Plane change**
>
> If a plane change (`G17`, `G18`, `G19`) is programmed after a rotation, the current angles of rotation of the respective axes are retained and are also effective in the new plane. It is therefore strongly recommended that the current angles of rotation be reset to 0 before a plane change:
>
> - `N100 ROT X0 Y0 Z0 ; explicit angle programming`
> - `N100 ROT ; implicit angle programming`

Absolute rotation with ROT X... Y... Z...

The WCS is rotated around the specified axes to the programmed angles of rotation.

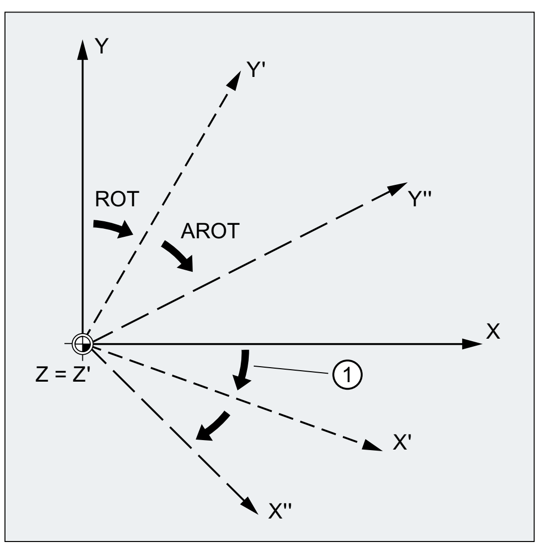


① Angle of rotation

Figure 3-26 Absolute rotation around the Z axis

Additive rotation with AROT X... Y... Z...

The WCS is rotated further around the specified axes through the programmed angles of rotation.



① Angle of rotation

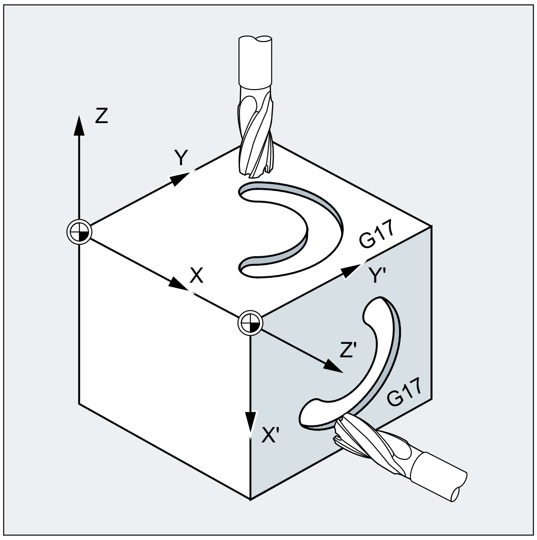
Figure 3-27 Absolute and additive rotation around the Z axis

Rotation of the working plane

During a rotation using `ROT`/`AROT`, the working plane (`G17`, `G18`, `G19`) also rotates.

Example: Working plane G17
The WCS is positioned on the top surface of the workpiece. Using offset and rotation, the coordinate system is moved to one of the side faces. Working plane G17 also rotates. In this way, traversing motions can still be programmed in the G17 plane via X and Y and infeeds via Z.

Requirement:
The tool must be perpendicular to the working plane and the positive direction of the infeed axis points in the direction of the tool base.
Specifying `CUT2DF` activates the tool radius compensation in the rotated plane.



3.12.5 Programmable frame rotations with solid angles (ROTS, AROTS, CROTS)

Rotations of the workpiece coordinate system can be specified in solid angles with the `ROTS`, `AROTS` and `CROTS` statements. Solid angles are the angles formed by the intersections of the plane rotated in space with the main planes of the not yet rotated WCS.

Note

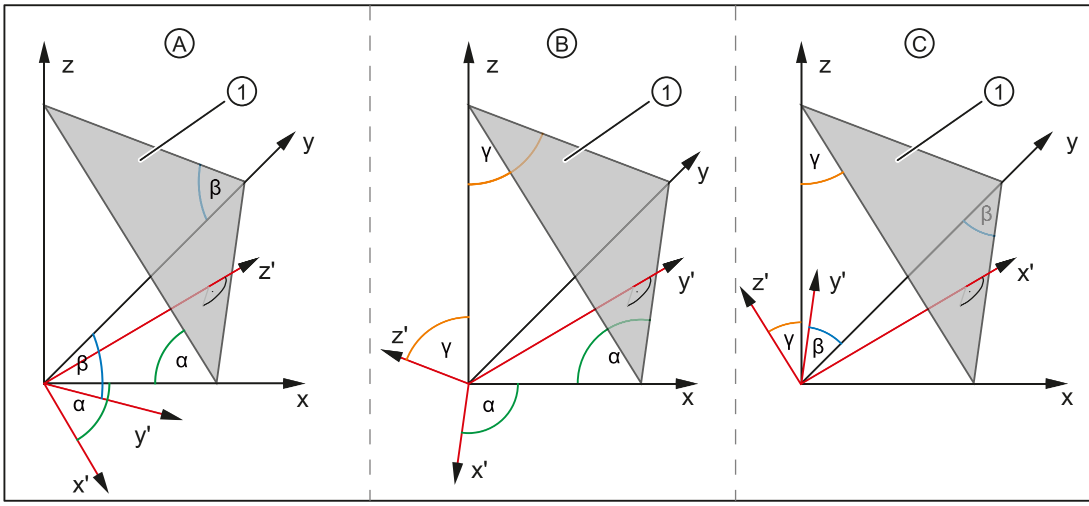
Geometry axis identifiers

The following definition is made as an example for the further description:

- 1st geometry axis: X
- 2nd geometry axis: Y
- 3rd geometry axis: Z

As shown in the following figure, the programming of `ROTS Xα Yβ` results in an alignment of the G17 plane of the WCS parallel to the displayed inclined plane. The position of the zero point of the WCS remains unchanged.

The orientation of the rotated WCS is defined so that the first rotated axis lies in the plane formed by this and the 3rd axis of the original coordinate system. In the example: X' is in the original X/Z plane.



①	Inclined plane
α, β, γ	Solid angle
A	Alignment of the G17 plane parallel to the inclined plane:

- 1st rotation
  Rotation of x around y through angle α ⇒
  x'-axis parallel to the inclined plane
- 2nd rotation
  Rotation of y' around x' through β ⇒
  y'-axis parallel to the inclined plane
  ⇒ z'-axis parallel to the inclined plane
  ⇒ G17 parallel to the inclined plane

B Alignment of the G18 plane parallel to the inclined plane:

- 1st rotation
  Rotation of z around x through the angle γ ⇒
  z'-axis parallel to the inclined plane
- 2nd rotation
  Rotation of x' around z' through angle α ⇒
  x'-axis parallel to the inclined plane
  ⇒ y'-axis parallel to the inclined plane
  ⇒ G18 parallel to the inclined plane

C Alignment of the G19 plane parallel to the inclined plane:

- 1st rotation
  Rotation of y around z through the angle β ⇒
  y'-axis parallel to the inclined plane
- 2nd rotation
  Rotation of z' around y' through angle γ ⇒
  z'-axis parallel to the inclined plane
  ⇒ x'-axis parallel to the inclined plane
  ⇒ G19 parallel to the inclined plane

Syntax

Requirements

The position of a plane in space is clearly defined by two solid angles. The plane would be "over-defined" by the specification of a third solid angle. It is therefore not permitted.

If only one solid angle is programmed, the rotation of the WCS is identical to `ROT`, `AROT` (see Section "Programmable rotation (ROT, AROT, RPL) (Page 312)").

Through the two programmed axes, a plane is specified according to the plane definitions for `G17`, `G18`, `G19`. This defines the sequence of the coordinate axes (1st axis / 2nd axis of the plane) or the sequence of the rotations through the solid angles:

Plane	1st axis	2nd axis
G17	X	Y
G18	Z	X
G19	Y	Z

Alignment of the G17 plane ⇒ solid angle for X and Y

- 1st rotation: X around Y through the angle α
- 2nd rotation: Y around X' through the angle β
- Orientation: X' is in the original Z/X plane.

```
ROTS  X<α> Y<β>
AROTS X<α> Y<β>
CROTS X<α> Y<β>
```

Alignment of the G18 plane ⇒ solid angle for Z and X

- 1st rotation: Z around X through the angle γ
- 2nd rotation: X around Z' through the angle α
- Orientation: Z' is in the original Y/Z plane

```
ROTS  Z<γ> X<α>
AROTS Z<γ> X<α>
CROTS Z<γ> X<α>
```

Alignment of the G19 plane ⇒ solid angle for Y and Z

- 1st rotation: Y around Z through the angle β
- 2nd rotation: Z around Y' through the angle γ
- Orientation: Y' is in the original X/Z plane.

```
ROTS  Y<β> Z<γ>
AROTS Y<β> Z<γ>
CROTS Y<β> Z<γ>
```

Meaning

`ROTS`:	Absolute frame rotations with solid angles, reference frame: Programmable frame $P_PFRAME
`AROTS`:	Additive frame rotations with solid angles, reference frame: Programmable frame $P_PFRAME
`CROTS`:	Absolute frame rotations with solid angles, reference frame: Programmed frame $P_ ...
`X`, `Y`, `Z`:	Geometry axis identifiers (see note above: Geometry axis identifiers)
A, β, γ:	Solid angle in relation to the appropriate geometry axis: • α → X • β → Y • γ → Z

3.12.6 Programmable scaling factor (SCALE, ASCALE)

`SCALE`/`ASCALE` can be used to program up or down scale factors for all path, synchronized, and positioning axes in the direction of the axes specified in each case. This makes it possible, therefore, to take geometrically similar shapes or different shrinkage allowances into account in the programming.

Syntax

```
SCALE X… Y… Z…
ASCALE X… Y… Z…
```

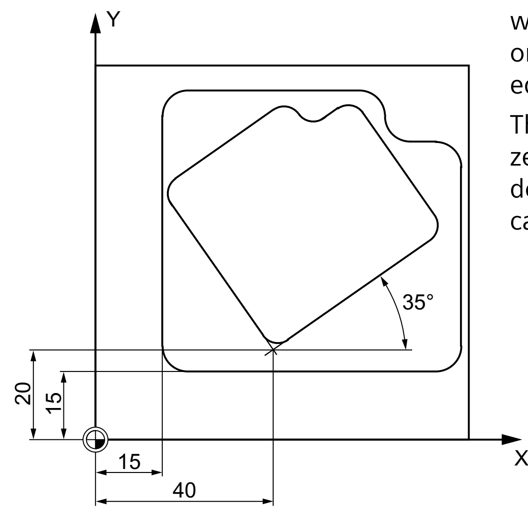
Note

Each frame operation is programmed in a separate NC block.

Meaning

`SCALE`:	Scale up/down absolute in relation to the currently valid coordinate system set with G54 to G57, G505 to G599.
`ASCALE`:	Scale up/down additive in relation to the currently valid set or programmed coordinate system.
`X… Y… Z…`:	Scale factors in the direction of the specified geometry axes.

Example



The pocket occurs twice on this workpiece, but with different sizes and rotated in relation to one another. The machining sequence is stored in the subprogram.

The required workpiece zeroes are set with zero offset and rotation, the contour is scaled down with scaling and the subprogram is then called again.

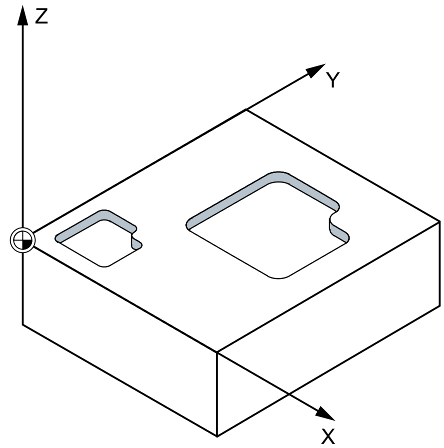
Program code	Comment
`N10 G17 G54`	`; Working plane X/Y, workpiece zero`
`N20 TRANS X15 Y15`	`; Absolute offset`
`N30 L10`	`; Machine large pocket`
`N40 TRANS X40 Y20`	`; Absolute offset`
`N50 AROT RPL=35`	`; Rotation in the plane through 35°`
`N60 ASCALE X0.7 Y0.7`	`; Scaling factor for the small pocket`
`N70 L10`	`; Machine small pocket`
`N80G0 X300 Y100 M30`	`; Retraction, end of program`

Further information

SCALE X... Y... Z...

You can specify an individual scale factor for each axis, by which the shape is to be reduced or enlarged. The scale refers to the workpiece coordinate system set with G54 to G57, G505 to G599.

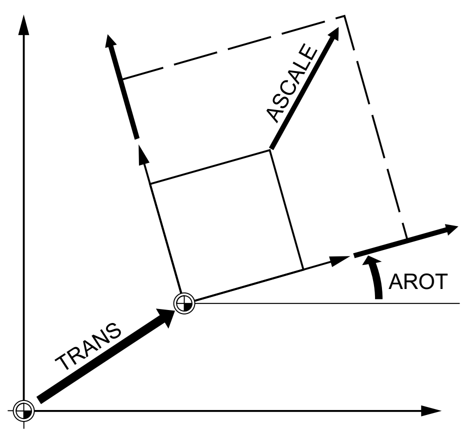
> **NOTICE**
>
> **No original frame**
>
> The `SCALE` command resets all frame components of the previously activated programmable frame.



ASCALE X... Y... Z...

The ASCALE command is used to program scale changes to be added to existing frames. In this case, the last valid scale factor is multiplied by the new one.

The currently set or last programmed coordinate system is used as the reference for the scale change.



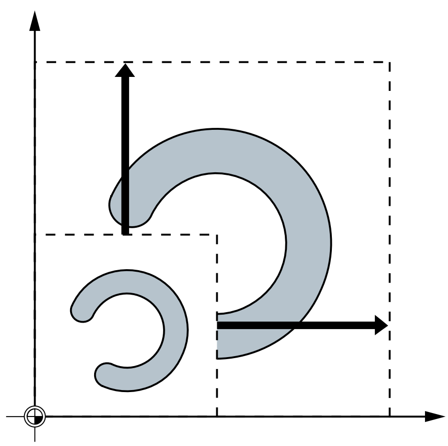
Scaling and offset

Note

If an offset is programmed with ATRANS after SCALE, the offset values will also be scaled.

Different scale factors

NOTICE
Risk of collision
Please take great care when using different scale factors! Circular interpolations can, for example, only be scaled using identical factors.



Note

However, different scale factors can be used specifically to program distorted circles.

3.12.7 Programmable mirroring (MIRROR, AMIRROR)

`MIRROR`/`AMIRROR` can be used to mirror workpiece shapes on coordinate axes. All traversing movements programmed after the mirror call (e.g. in the subprogram) are executed with mirroring.

Syntax

```
MIRROR X... Y... Z...
AMIRROR X... Y... Z...
```

Note

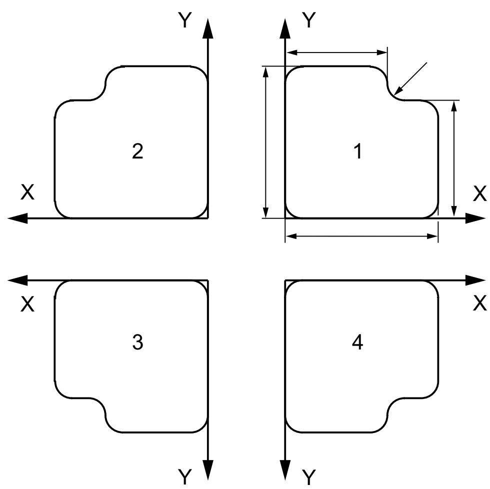
Each frame operation is programmed in a separate NC block.

Meaning

`MIRROR`:	Mirror absolute in relation to the currently valid coordinate system set with G54 to G57, G505 to G599.
`AMIRROR`:	Additive mirror image with reference to the currently valid set or programmed coordinate system.
`X... Y... Z...`:	Geometry axis whose direction is to be changed. The value specified here can be chosen freely, e.g. X0 Y0 Z0.

Examples

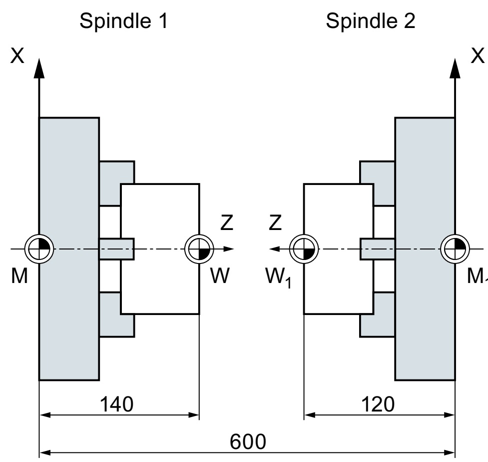
Example 1: Milling



The contour shown here is programmed once as a subprogram. The three other contours are generated using mirroring. The workpiece zero is located at the center of the contours.

Program code	Comment
N10 G17 G54	; Working plane X/Y, workpiece zero
N20 L10	; Machine first contour at top right
N30 MIRROR X0	; Mirror X axis (the direction is changed in X)
N40 L10	; Machine second contour at top left
N50 AMIRROR Y0	; Mirror Y axis (the direction is changed in Y)
N60 L10	; Machine third contour at bottom left
N70 MIRROR Y0	; MIRROR resets previous frames. Mirror Y axis (the direction is changed in Y)
N80 L10	; Machine fourth contour at bottom right
N90 MIRROR	;Deactivate mirroring
N100 G0 X300 Y100 M30	; Retraction, end of program

Example 2: Turning



The actual machining is stored as a subprogram and execution at the respective spindle is implemented by means of mirroring and offsets.

Program code	Comment
N10 TRANS X0 Z140	; Zero offset to W
...	; Machining of the first side with spindle 1
N30 TRANS X0 Z600	; Zero offset to spindle 2
N40 AMIRROR Z0	; Mirroring of the Z axis
N50 ATRANS Z120	; Zero offset to W1
...	; Machining of the second side with spindle 2

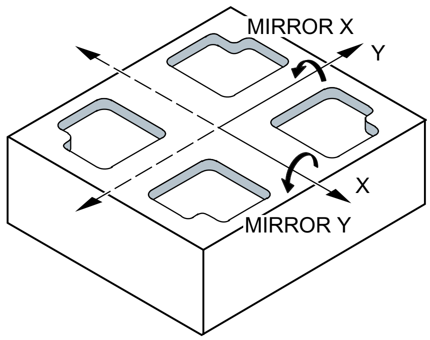
Further information

MIRROR X... Y... Z...

The mirror is programmed by means of an axial change of direction in the selected working plane.

Example: Working plane G17 X/Y

The mirror (on the Y axis) requires a direction change in X and, accordingly, is programmed with `MIRROR X0`. The contour is then mirrored on the opposite side of the mirror axis Y.

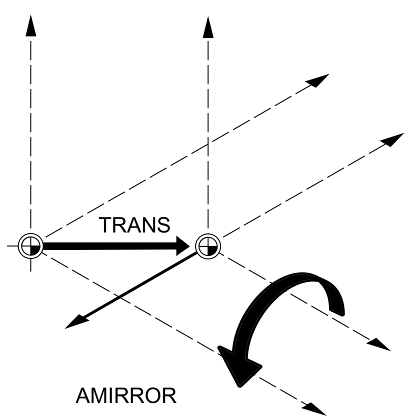


Mirroring is implemented in relation to the currently valid coordinate system set with G54 to G57, G505 to G599.

NOTICE
No original frame
The `MIRROR` command resets all frame components of the previously activated programmable frame.

AMIRROR X... Y... Z...

A mirror image, which is to be added to an existing transformation, is programmed with `AMIRROR`. The currently set or last programmed coordinate system is used as the reference.



Deactivate mirroring

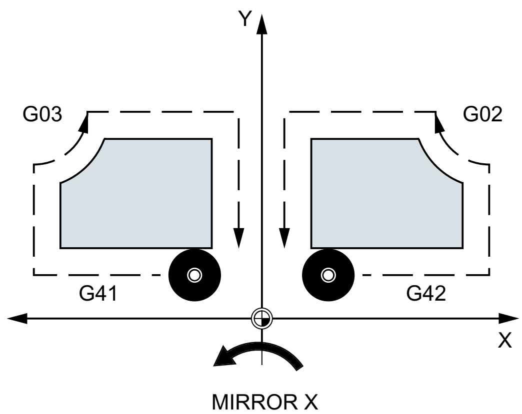
For all axes: `MIRROR` (without axis parameter)

All frame components of the previously programmed frame are reset.

Tool radius compensation

Note

The mirror command causes the control to automatically change the path compensation commands (`G41`/`G42` or `G42`/`G41`) according to the new machining direction.



The same applies to the direction of circle rotation (G2/G3 or G3/G2).

Note

If you program an additive rotation with `AROT` after `MIRROR`, you may have to work with reversed directions of rotation (positive/negative or negative/positive). Mirrors on the geometry axes are converted automatically by the control into rotations and, where appropriate, mirrors on the mirror axis specified in the machine data. This also applies to settable zero offsets.

Mirror axis

The axis to be mirrored can be set in machine data:

MD10610 $MN_MIRROR_REF_AX = <value>

Value	Meaning
0	Mirroring is performed around the programmed axis (negation of values).
1	The reference axis is the X axis.
2	The reference axis is the Y axis.
3	The reference axis is the Z axis.

Interpreting the programmed values

Machine data is used to specify how the programmed values are to be interpreted:

MD10612 $MN_MIRROR_TOGGLE = <value>

Value	Meaning
0	Programmed axis values are not evaluated.
1	Programmed axis values are evaluated: • For programmed axis values ≠ 0, the axis is mirrored if it has not yet been mirrored. • For a programmed axis value = 0, mirroring is deactivated.

3.12.8 Frame generation according to tool orientation (TOFRAME, TOROT, PAROT):

`TOFRAME` generates a rectangular frame whose Z axis coincides with the current tool orientation. This means that the user can retract the tool in the Z direction without risk of collision (e.g. after a tool break in a 5-axis program).

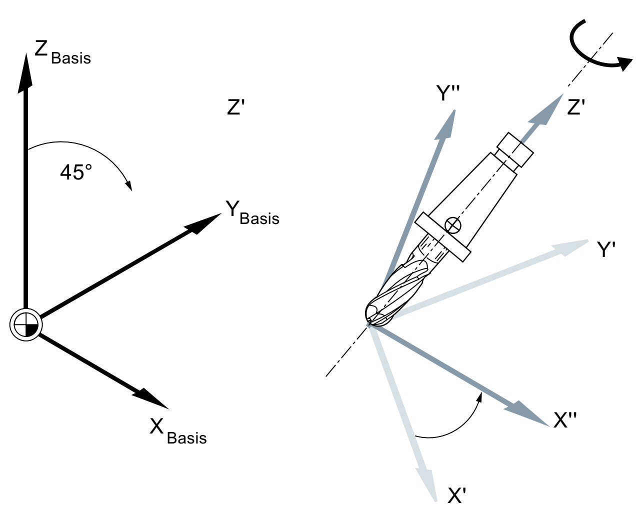
The position of the X and Y axes is determined by the setting in machine data MD21110 $MC_X_AXES_IN_OLD_X_Z_PLANE (coordinate system with automatic frame definition). The new coordinate system is either left as generated from the machine kinematics or is turned around the new Z axis additionally so that the new X axis lies in the old Z/X plane (see machine manufacturer's specifications).

The resulting frame describing the orientation is written in the system variable for the programmable frame ($P_PFRAME).

`TOROT` only overwrites the rotation component in the programmed frame. All other components remain unchanged.

`TOFRAME` and `TOROT` are designed for milling operations in which `G17` (working plane X/Y) is typically active. In the case of turning operations or generally when `G18` or `G19` is active, however, frames are needed where the X or Y axis matches the orientation of the tool. These frames are programmed with the `TOFRAMEX`/`TOROTX` or `TOFRAMEY`/`TOROTY` statements.

`PAROT` aligns the workpiece coordinate system on the workpiece.



Syntax

```
TOFRAME/TOFRAMEZ/TOFRAMEY/TOFRAMEX
...
TOROTOF
```

```
TOROT/TOROTZ/TOROTY/TOROTX
...
TOROTOF
```

```
PAROT
...
PAROTOF
```

Meaning

TOFRAME:	Align Z axis of the WCS by rotating the frame parallel to the tool orientation
TOFRAMEZ:	As TOFRAME
TOFRAMEY:	Align Y axis of the WCS by rotating the frame parallel to the tool orientation
TOFRAMEX:	Align X axis of the WCS by rotating the frame parallel to the tool orientation
TOROT:	Align Z axis of the WCS by rotating the frame parallel to the tool orientation The rotation defined with TOROT is the same as that defined with TOFRAME.
TOROTZ:	As TOROT
TOROTY:	Align Y axis of the WCS by rotating the frame parallel to the tool orientation
TOROTX:	Align X axis of the WCS by rotating the frame parallel to the tool orientation
TOROTOF:	Deactivate orientation parallel to tool orientation
PAROT:	Rotate frame to align workpiece coordinate system on workpiece Translations, scaling and mirroring in the active frame remain valid
PAROTOF:	The workpiece-specific frame rotation activated with PAROT is deactivated with PAROTOF.

Note

The TOROT statement ensures consistent programming with active orientable toolholders for each kinematic type.

Just as in the situation for rotatable toolholders, PAROT can be used to activate a rotation of the work table. This defines a frame which changes the position of the workpiece coordinate system in such a way that no compensatory movement is performed on the machine. Language command PAROT is not rejected if no toolholder with orientation capability is active.

Example

Program code	Comment
`N100 G0 G53 X100 Z100 D0`	
`N120 TOFRAME`	
`N140 G91 Z20`	`; TOFRAME is included in the calculation, all programmed geometry axis movements refer to the new coordinate system.`
`N160 X50`	
`...`	

Further information

Assigning axis direction

If one of the TOFRAMEX, TOFRAMEY, TOROTX, TOROTY statements is programmed instead of TOFRAME/TOFRAMEZ or TOROT/TOROTZ, the axis direction statements listed in this table will apply:

Statement	Tool direction (applicate)	Secondary axis (abscissa)	Secondary axis (ordinate)
TOFRAME / TOFRAMEZ / TOROT / TOROTZ	Z	X	Y
TOFRAMEY / TOROTY	Y	Z	X
TOFRAMEX / TOROTX	X	Y	Z

Separate system frame for TOFRAME or TOROT

The frames resulting from TOFRAME or TOROT can be written in a separate system frame $P_TOOLFRAME. For this purpose, bit 3 must be enabled in machine data MD28082 $MC_MM_SYSTEM_FRAME_MASK. The programmable frame remains unchanged. Differences occur when the programmable frame is processed further elsewhere.

3.12.9 Deselect frame (G53, G153, SUPA, G500)

When executing certain processes, such as approaching the tool change point, various frame components have to be defined and suppressed at different times.

Settable frames can either be deactivated modally or suppressed non-modally.

Programmable frames can be suppressed or deleted non-modally.

Syntax

```
G53
G153
SUPA
G500
TRANS
ROT
SCALE
```

```
MIRROR
```

Meaning

`G53`:	Non-modal suppression of all programmable and settable frames
`G153`:	`G153` has the same effect as `G53` and also suppresses the entire basic frame ($P_ACTBFRAME).
`SUPA`:	`SUPA` has the same effect as `G153` and also suppresses: • Handwheel offsets (DRF) • Overlaid movements • External zero offset • PRESET offset
`G500`:	Modal deactivation of all settable frames (G54 to G57, G505 to G599) if G500 does not contain a value.
`TRANS ROT SCALE MIRROR`:	Without axis details, a deletion of the programmable frames acts.

3.12.10 Deselect DRF offsets / position offsets (CORROF, DRFOF)

DRF offsets set via handwheel procedures and position offsets programmed via the $AA_OFF system variable can be deselected axis-specifically via the predefined CORROF procedure.

DRF offsets for all active axes in the channel can be deselected with the predefined procedure DRFOF.

A preprocessing stop is initiated through the deselection and the position component of the deselected DRF offset is transferred to the position in the basic coordinate system. No axis is traversed. The value of the $AA_IM system variable (current MCS setpoint of an axis) does not change; the value of the $AA_IW system variable (current WCS setpoint of an axis) changes because it now contains the deselected component from the overlaid movement.

Syntax

```
CORROF(<Axis>,"<String>"[,<Axis>,"<String>"]) / DRFOF
```

Meaning

<table>
<tr><td rowspan="8">CORROF</td><td colspan="4">Axis-specific deselection of a DRF offset or a position offset ($AA_OFF)</td></tr>
<tr><td>Effectiveness:</td><td colspan="3">Modal</td></tr>
<tr><td rowspan="2"><Axis></td><td colspan="3">Axis identifier (channel, geometry or machine axis identifier)</td></tr>
<tr><td>Data type:</td><td colspan="2">AXIS</td></tr>
<tr><td rowspan="4"><String></td><td colspan="3">This parameter is used to specify whether a DRF offset or a position offset ($AA_OFF) is to be deselected.</td></tr>
<tr><td>Data type:</td><td colspan="2">STRING</td></tr>
<tr><td rowspan="2">Value:</td><td>DRF</td><td>DRF offset</td></tr>
<tr><td>AA_OFF</td><td>Position offset ($AA_OFF)</td></tr>
</table>

DRFOF	Deselection of the DRF offsets for all active axes of the channel	
	Effectiveness:	Modal

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
N10 CORROF(X,"DRF")	; Only the DRF offset of the X axis is deselected; the DRF offset of the Y axis is retained (in the case of DRFOF both offsets would have been deselected).
...	

Example 3: 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
N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5	; A position offset == 10 is interpolated for the X axis.
...	
N70 CORROF(X,"DRF",X,"AA_OFF")	; Only the DRF offset and the position offset of the X axis are deselected; the DRF offset of the Y axis is retained.
...	

Example 4: 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
N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5	; A position offset == 10 is interpolated for the X axis.
...	

Program code	Comment
N70 CORROF(Y,"DRF",X,"AA_OFF")	; 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.
...	

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.

3.13 Auxiliary function outputs

Function

The auxiliary function output sends information to the PLC indicating when the NC program needs the PLC to perform specific switching operations on the machine tool. The auxiliary functions are output, together with their parameters, to the PLC interface. The transferred values and signals must be processed by the PLC user program.

Auxiliary functions

The following auxiliary functions can be transferred to the PLC:

Auxiliary Function	Address
Tool selection	T
Tool offset	D, DL
Feedrate	F/FA
Spindle speed	S
M functions	M
H functions	H

For each function group or single function, machine data is used to define whether the output is triggered **before**, **with** or **after** the traversing motion.

The PLC can be programmed to acknowledge auxiliary function outputs in various ways.

Properties

Important properties of the auxiliary function are shown in the following overview table:

Function	Address extension		Value			Explanations	Maximum number per block
	Meaning	Range	Range	Type	Meaning		
M	-	0 (implicit)	0 ... 99	INT	Function	The address extension is 0 for the range between 0 and 99. Mandatory without address extension: M0, M1, M2, M17, M30	5
	Spindle no.	1 - 12	1 ... 99	INT	Function	M3, M4, M5, M19, M70 with address extension spindle no. (e.g. `M2=5`; spindle stop for spindle 2). Without spindle number, the function applies for the master spindle.	
	Any	0 - 99	100 ... 2147483647	INT	Function	User M function*	
S	Spindle no.	1 - 12	0 ... $\pm 1.8*10^{308}$	REAL	Speed	Without spindle number, the function applies for the master spindle.	3
H	Any	0 - 99	0 ... ± 2147483647 $\pm 1.8*10^{308}$	 INT REAL	Any	Functions have no effect in the NC; only to be implemented on the PLC.*	3
T	Spindle no. (for active tool management)	1 - 12	0 - 32000 (or tool names with active tool management)	INT	Tool selection	Tool names are not passed to the PLC interface.	1
D	-	-	0 - 12	INT	Tool offset selection	D0: Deselection Default setting: D1	1

Function	Address extension		Value			Explanations	Maximum number per block
	Meaning	Range	Range	Type	Meaning		
DL	Location-dependent offset	1 - 6	0 ... ± $1.8*10^{308}$	REAL	Tool fine offset selection	Refers to previously selected D number.	1
F	-	-	0.001 - 999 999.999	REAL	Path feedrate		6
FA	Axis No.	1 - 31	0.001 - 999 999.999	REAL	Axial feedrate		
* The meaning of the functions is defined by the machine manufacturer (**see machine manufacturer's specifications**).							

Further information

Number of function outputs per NC block

Up to 10 function outputs can be programmed in one NC block. Auxiliary functions can also be output from the action component of **synchronized actions**.

Grouping

The functions described can be grouped together. Group assignment is predefined for some M commands. The acknowledgment behavior can be defined by the grouping.

High-speed function outputs (QU)

Functions, which have not been programmed as high-speed outputs, can be defined as high-speed outputs for individual outputs with the keyword `QU`. Program execution continues without waiting for the acknowledgment of the miscellaneous function (the program waits for the transport acknowledgment). This helps avoid unnecessary hold points and interruptions to traversing movements.

Note

The appropriate machine data must be set for the "High-speed function outputs" function (→ **machine manufacturer**).

Function outputs for travel commands

The transfer of information as well as waiting for the appropriate response takes time and therefore influences the traversing movements.

High-speed acknowledgment without block change delay

Block change behavior can be influenced by machine data. When the "without block change delay" setting is selected, the system response with respect to high-speed auxiliary functions is as follows:

Auxiliary function output	Response
Before the movement	The block transition between blocks with high-speed auxiliary functions occurs **without** interruption and **without** a reduction in velocity. The auxiliary function output takes place in the first interpolator clock cycle of the block. The following block is executed with no acknowledgment delay.
During the movement	The block transition between blocks with high-speed auxiliary functions occurs **without** interruption and **without** a reduction in velocity. The auxiliary function output takes place during the block. The following block is executed with no acknowledgment delay.
After the movement	The movement stops at the end of the block. The auxiliary function output takes place at the end of the block. The following block is executed with no acknowledgment delay.

CAUTION

Function outputs in continuous-path mode

Function outputs **before** traversing motion interrupt the continuous-path mode (G64 / G641) and generate an exact stop for the previous block.

Function outputs **after** the traversing movements interrupt the continuous-path mode (G64 / G641) and generate an exact stop for the current block.

Important: Waiting for a pending acknowledgment signal from the PLC can also interrupt the continuous-path mode, e.g. for M command sequences in blocks with extremely short path lengths.

3.13.1 M functions

The M functions initiate switching operations, such as "Coolant ON/OFF" and other functions on the machine.

Syntax

```
M<value>
M[<address extension>] = <value>
```

Meaning

`M:`	Address for the programming of the M functions.
`<address extension>:`	The extended address notation applies for some M functions (e.g. specification of the spindle number for spindle functions).

`<value>`:	Assignment is made to a certain machine function through the value assignment (M function number).	
	Type:	INT
	Range of values:	0 ... 2147483647 (max. INT value)

Predefined M functions

Certain important M functions for program execution are supplied as standard with the control:

M function	Meaning
M0*	Programmed stop
M1*	Optional stop
M2*	End of program, main program (as M30)
M3	Spindle clockwise
M4	Spindle counter-clockwise
M5	Spindle stop
M6	Tool change (default setting)
M17*	End of subprogram
M19	Spindle positioning
M30*	End of program, main program (as M2)
M40	Automatic gear change
M41	Gear stage 1
M42	Gear stage 2
M43	Gear stage 3
M44	Gear stage 4
M45	Gear stage 5
M70	Spindle is switched to axis mode

Note

Extended address notation cannot be used for the functions marked with *.

The functions M0, M1, M2, M17 and M30 are always triggered **after** the traversing movement.

M functions defined by the machine manufacturer

All free M function numbers can be used by the machine manufacturer, e.g. for switching functions to control the clamping devices or for the activation/deactivation of further machine functions.

Note

The functions assigned to the free M function numbers are machine-specific. A certain M function can therefore have a different functionality on another machine.

Refer to the machine manufacturer's specifications for the M functions available on a machine and their functions.

Examples

Example 1: Maximum number of M functions in the block

Program code	Comment
N10 S...	
N20 X... M3	; M function in the block with axis movement,
	; spindle accelerates prior to X axis movement.
N180 M789 M1767 M100 M102 M376	; Maximum of five M functions in the block.

Example 2: M function as high-speed output

Program code	Comment
N10 H=QU(735)	; Fast output for H735.
N10 G1 F300 X10 Y20 G64	
N20 X8 Y90 M=QU(7)	; Fast output for M7.

M7 has been programmed as fast output so that the continuous-path mode (G64) is not interrupted.

Note

Only use this function in special cases as, for example, the chronological alignment is changed in combination with other function outputs.

Further information about the predefined M commands

Programmed stop: M0

The machining is stopped in the NC block with `M0`. You can now remove chips, remeasure, etc.

Programmed stop 1 - optional stop: M1

M1 can be set via:

- HMI / dialog box "Program Control"
  or
- NC/PLC interface

The program execution of the NC is stopped by the programmed blocks.

Programmed stop 2 - an auxiliary function associated with M1 with stop in the program execution

Programmed stop 2 can be set via the HMI / dialog box "Program Control" and allows the technological sequences to be interrupted at any time at the end of the part to be machined. In this way, the operator can interrupt the production, e.g. to remove chip flows.

End of program: M2, M17, M30

A program is ended with `M2`, `M17` or `M30`. If the main program is called from another program (as subprogram), `M2`/`M30` has the same effect as `M17` and vice versa, i.e. `M17` has the same effect in the main program as `M2`/`M30`.

Spindle functions: M3, M4, M5, M19, M70

The extended address notation with specification of the spindle number applies for all spindles.

Example:

Program code	Comment
M2=3	; Clockwise spindle rotation for the second spindle

If an address extension has not been programmed, the function applies for the master spindle.

3.14 Supplementary commands

3.14.1 Output messages (MSG)

Using the `MSG()` statement, any character string from the part program can be output as message to the operator.

Syntax

```
MSG("<Message text>"[,<Execution>])
...
MSG ()
```

Meaning

MSG:	Predefined subprogram call for output of a message	
<message text>:	Any character string to be displayed as message	
	Type:	STRING
	Maximum length:	124 characters; the display takes up two lines (2*62 characters)
	By using the link operator "<<", variables can also be output in the message text.	

<Execution>:	Parameter to define the time when the message is written (optional)		
	Type:	INT	
	Value:	0 (basic setting)	To write the message, a dedicated main run block is not generated. This is realized in the next NC block that can be executed. Active continuous-path mode is not interrupted.
		1	To write the message, a dedicated main run block is generated. Active continuous-path mode is interrupted.
MSG():	The actual message can be deleted by programming MSG() without message text. If not deleted, the display remains until the next message is present.		

Note

If the message is to be output in the language active on the user interface, then the user requires information about the language that is currently set on the HMI. This information can be interrogated in the part program and in synchronized actions by the the system variable $AN_LANGUAGE_ON_HMI (Page 1227)

Examples

Example 1: Output/delete message

Program code	Comment
N10 G91 G64 F100	; Continuous path mode
N20 X1 Y1	
N... X... Y...	
N20 MSG ("Machining part 1")	; The message is first output with N30.
	; continuous-path mode is retained.
N30 X... Y...	
N... X... Y...	
N400 X1 Y1	
N410 MSG ("Machining part 2",1)	; The message is output with N410.
	; continuous-path mode is interrupted.
N420 X1 Y1	
N... X... Y...	
N900 MSG ()	; Delete message.

Example 2: Message text with variable

Program code	Comment
N10 R12=$AA_IW [X]	; Actual position of the X axis in R12.
N20 MSG ("Check position of X axis"<<R12<<)	; Output message with variable R12.

Program code	Comment
...	
N90 MSG ()	; Clear message from N20.

3.14.2 Writing string in OPI variable (WRTPR)

Using the `WRTPR()` function, it is possible to write any character string from the part program into the OPI variable progProtText.

Syntax

```
WRTPR(<String>[,<ExecTime>])
```

Meaning

WRTPR:	Function call for outputting a character string.		
<String>:	Any character string, which is written to the OPI variable progProtText.		
	Type:	STRING	
	Maximum length:	128 characters	
<ExecTime>:	Optional parameters to define the instant in time when the string is written.		
	Type:	INT	
	Range of values:	0, 1	
		0 (default)	No dedicated main run block is not generated to write the character string. This is realized in the next NC block that can be executed. Active continuous-path mode is not interrupted.
		1	A dedicated main run block is generated to write the character string. Active continuous-path mode is interrupted.

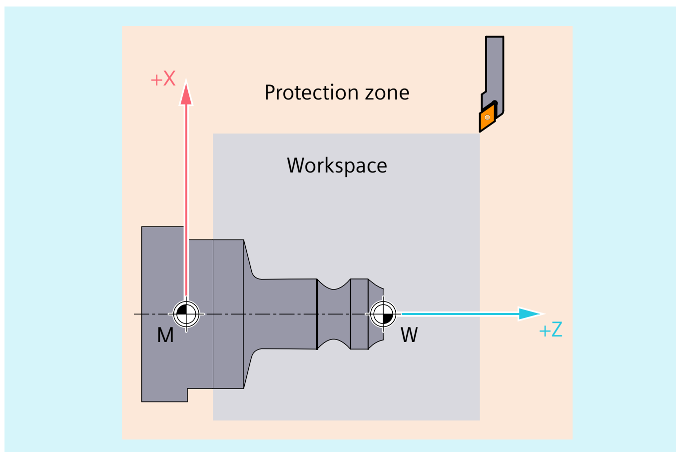
Examples

Program code	Comment
N10 G91 G64 F100	; continuous path mode
N20 X1 Y1	
N30 WRTPR("N30")	; The character string "N30" is first written to N40.
	; Continuous-path mode is kept.
N40 X1 Y1	
N50 WRTPR("N50",1)	; The character string "N50" is written to N50.
	; Continuous-path mode is interrupted.
N60 X1 Y1	

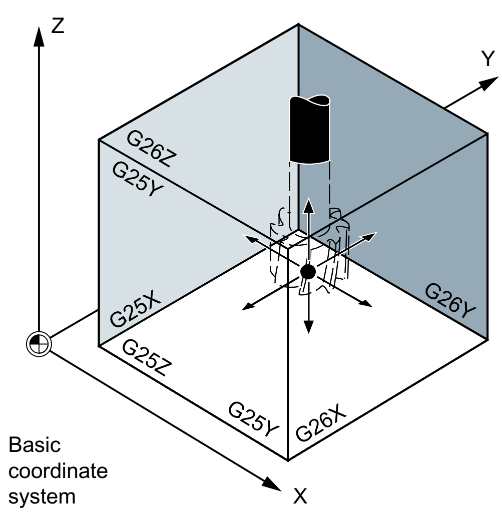
3.14.3 Working area limitation

3.14.3.1 Working area limitation in BCS (G25/G26, WALIMON, WALIMOF)

`G25`/`G26` limits the working area (working field, working space) in which the tool can traverse. The areas outside the working area limitations defined with `G25`/`G26` are inhibited for any tool motion.



The coordinates for the individual axes apply in the basic coordinate system:



The working area limitation for all validated axes must be programmed with the `WALIMON` command. The `WALIMOF` command deactivates the working area limitation. `WALIMON` is the default setting. Therefore, it only has to be programmed if the working area limitation has been disabled beforehand.

Syntax

```
G25 X…Y…Z…
G26 X…Y…Z…
WALIMON
...
WALIMOF
```

Meaning

`G25`:	**Lower** working area limitation Assignment of values in channel axes in the basic coordinate system
`G26`:	**Upper** working area limitation Assignment of values in channel axes in the basic coordinate system
`X… Y… Z…`:	Lower or upper working area limits for individual channel axes The limits specified refer to the basic coordinate system.
`WALIMON`:	Switch working area limitation **on** for all axes
`WALIMOF`:	Switch working area limitation **off** for all axes

In addition to programming values using `G25`/`G26`, values can also be entered using axis-specific setting data:

SD43420 $SA_WORKAREA_LIMIT_PLUS (Working area limitation plus)

SD43430 $SA_WORKAREA_LIMIT_MINUS (Working area limitation minus)

Activating and deactivating the working area limitation, parameterized using SD43420 and SD43430, are carried out for a specific direction using the axis-specific setting data that becomes immediately effective:

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)

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

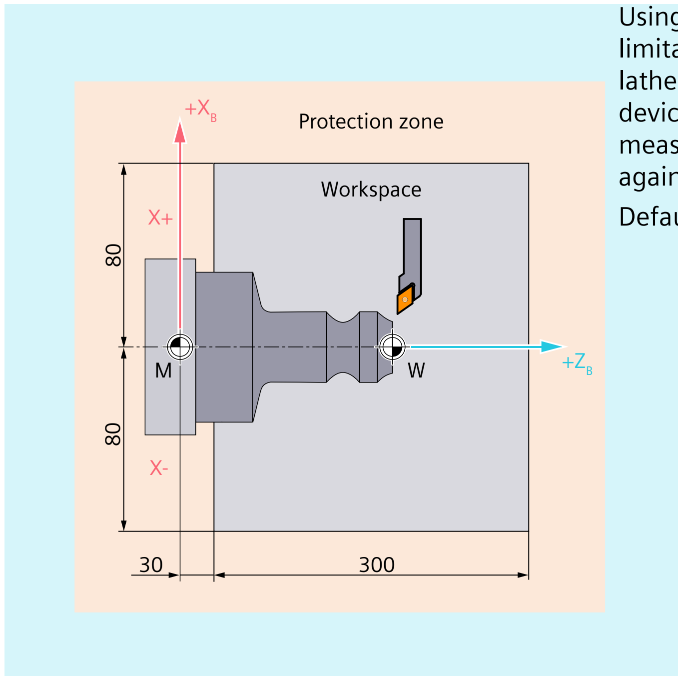
Note

The programmed working area limitation, programmed with `G25`/`G26`, has priority and overwrites the values entered in SD43420 and SD43430.

Note

`G25`/`G26` can also be used to program limits for spindle speeds at the address `S`. For more information see " Programmable spindle speed limitation (G25, G26) (Page 114) ".

Example



Using the working area limitation G25/26, the working area of a lathe is limited so that the surrounding devices and equipment - such as revolver, measuring station, etc. are protected against damage.

Default setting: WALIMON

Program code	Comment
N10 G0 G90 F0.5 T1	
N20 G25 X-80 Z30	; Define the lower limit for the individual coordinate axes
N30 G26 X80 Z330	;Define the upper limit
N40 L22	;Cutting program
N50 G0 G90 Z102 T2	;To tool change location
N60 X0	
N70 WALIMOF	;Deactivate working area limitation
N80 G1 Z-2 F0.5	;Drill
N90 G0 Z200	;Back
N100 WALIMON	; Switch on working area limitation
N110 X70 M30	; End of program

Further information

Reference point at the tool

When tool length offset is active, the tip of the tool is monitored as reference point, otherwise it is the toolholder reference point.

Taking into consideration the tool radius must be activated separately. This is done using channel-specific machine data:

MD21020 $MC_WORKAREA_WITH_TOOL_RADIUS

If the tool reference point lies outside the working area defined by the working area limitation, or if this area is exited, then the program sequence is stopped.

Note

If transformations are active, the tool data taken into consideration (tool length and tool radius) can deviate from the described behavior.

Programmable working area limitation, G25/G26

An upper (G26) and a lower (G25) working area limitation can be defined for each axis. These values are effective immediately and remain effective for the corresponding MD setting (→ MD10710 $MN_PROG_SD_RESET_SAVE_TAB) after RESET and after being powered-up again.

Note

Using the predefined CALCPOSI (Page 571) tag function, it can be checked as to whether the predicted path is moved through taking into account the working area limits and/or the protection areas.

3.14.3.2 Working area limitation in WCS/SZS (WALCS0 ... WALCS10)

The "working area limitation in 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 homed.

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

Syntax

```
...
$P_WORKAREA_CS_COORD_SYSTEM[<WALimNo>]=<Value>
$P_WORKAREA_CS_PLUS_ENABLE[<WALimNo>,<Ax>]=<Value>
$P_WORKAREA_CS_LIMIT_PLUS[<WALimNo>,<Ax>]=<Value>
$P_WORKAREA_CS_MINUS_ENABLE[<WALimNo>,<Ax>]=<Value>
```

```
$P_WORKAREA_CS_LIMIT_MINUS[<WALimNo>,<Ax>]=<Value>
...
WALCS<n>
...
WALCS0
```

Meaning

$P_WORKAREA_CS_COORD_SYSTEM[<WALimNo>]=<Value>		
The coordinate system to which the working area limitation group refers		
<WALimNo>:	Working area limitation group	
	Type:	INT
	Range of values:	0 (group 1) ... 9 (group 10)
<Value>:	Value of the type INT	
	1	Workpiece coordinate system (WCS)
	3	Settable zero system (SZS)

$$P_WORKAREA_CS_PLUS_ENABLE[<WALimNo>,<Ax>]=<Value>		
Enable the working area limitation in the **positive** axis direction for the specified channel axis		
<WALimNo>:	Working area limitation group	
	Type:	INT
	Range of values:	0 (group 1) ... 9 (group 10)
<Ax>:	Channel axis name	
<Value>:	Value of the type BOOL	
	0 (FALSE)	No release
	1 (TRUE)	Release

$P_WORKAREA_CS_MINUS_ENABLE[<WALimNo>,<Ax>]=<Value>		
Enable the working area limitation in the **negative** axis direction for the specified channel axis		
<WALimNo>:	Working area limitation group	
	Type:	INT
	Range of values:	0 (group 1) ... 9 (group 10)
<Ax>:	Channel axis name	
<Value>:	Value of the type BOOL	
	0 (FALSE)	This has not been released
	1 (TRUE)	Enable

$P_WORKAREA_CS_LIMIT_PLUS[<WALimNo>,<Ax>]=<Value>		
Working area limitation in the **positive** direction of the specified channel axis		
<WALimNo>:	Working area limitation group	
	Type:	INT
	Range of values:	0 (group 1) ... 9 (group 10)

`<Ax>:`	Channel axis name
`<Value>:`	Value of the type REAL

`$P_WORKAREA_CS_LIMIT_MINUS[<WALimNo>,<Ax>]=<Value>`		
Working area limitation in the **negative** direction of the specified channel axis		
`<WALimNo>:`	Working area limitation group	
	Type:	INT
	Range of values:	0 (group 1) ... 9 (group 10)
`<Ax>:`	Channel axis name	
`<Value>:`	Value of the type REAL	

`WALCS<n>:`	Activation of the working area limitations of a working area limitation group	
`<n>:`	Number of the working area limitation group	
	Range of values:	1 ... 10

`WALCS0:`	Deactivation of the working area limits active in the channel

---

Note

The actual available number of working area limitation groups depends on the configuration (→ see details of the machine manufacturer).

Example

Three axes are defined in the channel: X, Y and Z

A working area limitation group No. 2 is to be defined and then activated in which the axes are to be limited in the WCS according to the following specifications:

- 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

Program code	Comment
...	

Program code	Comment
N51 $P_WORKAREA_CS_COORD_SYSTEM[1]=1	; The working area limitation of working area limitation group 2 applies in the WCS.
N60 $P_WORKAREA_CS_PLUS_ENABLE[1,X]=TRUE	
N61 $P_WORKAREA_CS_LIMIT_PLUS[1,X]=10	
N62 $P_WORKAREA_CS_MINUS_ENABLE[1,X]=FALSE	
N70 $P_WORKAREA_CS_PLUS_ENABLE[1,Y]=TRUE	
N73 $P_WORKAREA_CS_LIMIT_PLUS[1,Y]=34	
N72 $P_WORKAREA_CS_MINUS_ENABLE[1,Y]=TRUE	
N73 $P_WORKAREA_CS_LIMIT_MINUS[1,Y]=-25	
N80 $P_WORKAREA_CS_PLUS_ENABLE[1,Z]=FALSE	
N82 $P_WORKAREA_CS_MINUS_ENABLE[1,Z]=TRUE	
N83 $P_WORKAREA_CS_LIMIT_PLUS[1,Z]=-600	
...	
N90 WALCS2	; Activate working area limitation group 2.
...	

Further information

Effectivity

The working area limitation with `WALCS1` - `WALCS10` acts independently of the working area limitation with `WALIMON`. If both functions are active, that limit becomes effective which the axis motion first reaches.

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 corresponds to the behavior for the working area limitation with `WALIMON`.

3.14.4 Reference point approach (G74)

When the machine has been powered up (where incremental position measuring systems are used), all of the axis slides must approach their reference mark. Only then can traversing movements be programmed.

The reference point can be approached in the NC program with `G74`.

Syntax

`G74 X1=0 Y1=0 Z1=0 A1=0` … ; Programmed in a separate NC block

Meaning

G74:	G command call reference point approach
X1=0 Y1=0 Z1=0 …:	The specified machine axis address X1, Y1, Z1 ... for **linear axes** is approached as the reference point.
A1=0 B1=0 C1=0 …:	The specified machine axis address A1, B1, C1 ... for **rotary axes** is approached as the reference point.

Note

A transformation must not be programmed for an axis which is to approach the reference point with G74.

The transformation is deactivated with command TRAFOOF.

Example

When the measuring system is changed, the reference point is approached and the workpiece zero point is set up.

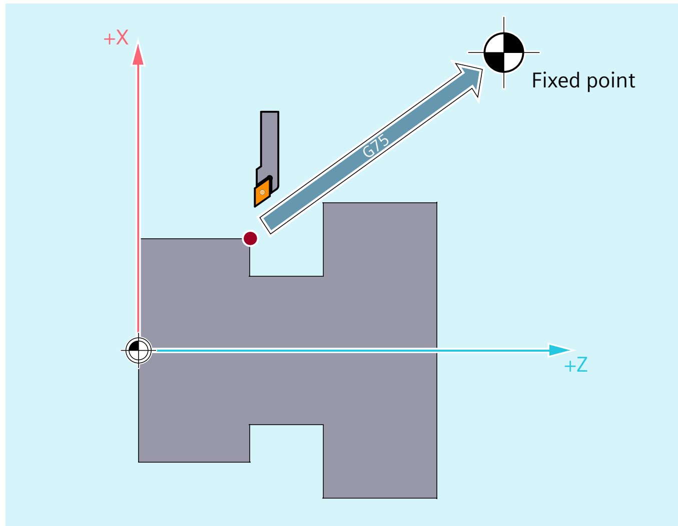
Program code	Comment
N10 SPOS=0	;Spindle in position control
N20 G74 X1=0 Y1=0 Z1=0 C1=0	;Reference point approach for linear axes and rotary axes
N30 G54	; Zero offset
N40 L47	;Cutting program
N50 M30	; End of program

3.14.5 Approaching a fixed point (G75)

The non-modal command G75 can be used to move axes individually and independently of one another to fixed points in the machine space, e.g. to tool change points, loading points, pallet change points, etc.

The fixed points are positions in the machine coordinate system which are stored in the machine data (MD30600 $MA_FIX_POINT_POS[n]). A maximum of four fixed points can be defined for each axis.

The fixed points can be approached from every NC program irrespective of the current tool or workpiece positions. An internal preprocessing stop is executed prior to moving the axes.



Requirements

The following requirements must be satisfied to approach fixed points with `G75`:

- The fixed-point coordinates must have been calculated exactly and written to machine data.
- The fixed points must be located within the valid traversing range (→ note the software limit switch limits!)
- The axes to be traversed must be referenced.
- No tool radius compensation must be active.
- A kinematic transformation may not be active.
- None of the axes to be traversed must be involved in active transformation.
- None of the axes to be traversed must be a following axis in an active coupling.
- None of the axes to be traversed must be an axis in a gantry grouping.
- Compile cycles must not activate motion components.

Syntax

```
G75 <axis name><axis position> ... FP=<n>
```

Meaning

`G75`:	Fixed point approach
`<axis name>`:	Name of the machine axis to be traversed to the fixed point All axis identifiers are permitted.
`<axis position>`:	The position value has no significance. A value of "0" is, therefore, usually specified.

<table>
<tr><td rowspan="4">FP=:</td><td colspan="3">Fixed point that is to be approached</td></tr>
<tr><td rowspan="2"><n>:</td><td colspan="2">Fixed point number</td></tr>
<tr><td>Range of values:</td><td>1, 2, 3, 4</td></tr>
<tr><td colspan="3">Note:
In the absence of FP=<n> or a fixed point number, or if FP=0 has been programmed, this is interpreted as FP=1 and fixed point 1 is approached.</td></tr>
</table>

Note

Multiple axes can be programmed in one G75 block. The axes are then traversed simultaneously to the specified fixed point.

Note

The value of the address FP must not be greater than the number of fixed points specified for each programmed axis (MD30610 $MA_NUM_FIX_POINT_POS).

Example

For a tool change, axes X (= AX1) and Z (= AX3) need to move to the fixed machine axis position 1 where X = 151.6 and Z = -17.3.

Machine data:

- MD30600 $MA_FIX_POINT_POS[AX1,0] = 151.6
- MD30600 $MA_FIX_POINT[AX3,0] = 17.3

NC program:

Program code	Comment
…	
N100 G55	; Activate settable zero offset.
N110 X10 Y30 Z40	; Approach positions in the WCS.
N120 G75 X0 Z0 FP=1 M0	; The X axis moves to 151.6
	; and the Z axis moves to 17.3 (in the MCS).
	; Each axis travels at its maximum velocity.
	; No additional movements are permitted to be active in this block.
	; A stop is inserted here so that after reaching
	; the end positions,
	; no additional motion takes place.
N130 X10 Y30 Z40	; The position of N110 is approached again.
	; The zero offset is reactivated.
…	

Note

If the "Tool management with magazines" function is active, the auxiliary function `T…` or `M...` (typically `M6`) will not be sufficient to trigger a block change inhibit at the end of `G75` motion.

Reason: If "Tool management with magazines" is active, auxiliary functions for tool change are not output to the PLC.

Further information

G75

The axes are traversed as machine axes in rapid traverse. The motion is mapped internally using the "SUPA" (suppress all frames) and "G0 RTLIOF" (rapid traverse motion with single-axis interpolation) functions.

If the conditions for "RTLIOF" (single-axis interpolation) are not met, the fixed point is approached as a path.

When the fixed point is reached, the axes come to a standstill within the "Exact stop fine" tolerance window.

Parameterizable dynamic response for G75

The required dynamic response mode can be set via the following machine data for positioning movements to fixed-point positions (G75):

MD18960 $MN_POS_DYN_MODE (type of positioning axis dynamic response)

Additional axis movements

The following additional axis movements are taken into account at the instant in time at which the G75 block is interpreted:

- External work offset
- DRF
- Synchronization offset ($AA_OFF)

After this, the additional axis movements are not permitted to change until the end of traversing is reached by the G75 block.

Additional motion following interpretation of the G75 block will offset the approach to the fixed point accordingly.

The following additional movements are not taken into account, irrespective of the point at which interpolation takes place, and will offset the target position accordingly:

- Online tool offset
- Additional movements from compile cycles in the BCS and machine coordinate system

Active frames

All active frames are ignored. Traversing is performed in the machine coordinate system.

Working area limitation in the workpiece coordinate system/SZS

Coordinate-system-specific working area limitation (WALCS0 ... WALCS10) is not effective in the block with G75. The destination point is monitored as the starting point of the following block.

Axis/Spindle movements with POSA/SPOSA

If programmed axes/spindles were previously traversed with POSA or SPOSA, these movements will be completed first before the fixed point is approached.

Spindle functions in the G75 block

If the spindle is excluded from "fixed-point approach", then additional spindle functions (e.g. positioning with SPOS/SPOSA) can be programmed in the G75 block.

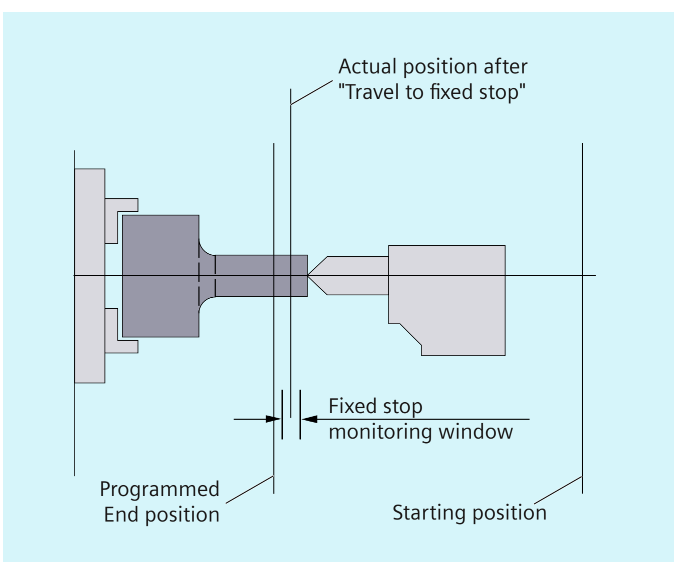
Modulo axes

In the case of modulo axes, the fixed point is approached along the shortest distance.

3.14.6 Travel to fixed stop (FXS, FXST, FXSW)

Function

The "Travel to fixed stop" function can be used to establish defined forces for clamping workpieces, such as those required for tailstocks, quills and grippers. The function can also be used for the approach of mechanical reference points.



With sufficiently reduced torque, it is also possible to perform simple measurement operations without connecting a probe. The "travel to fixed stop" function can be implemented for axes as well as for spindles with axis-traversing capability.

Syntax

```
FXS[<axis>]=…
FXST[<axis>]=…
FXSW[<axis>]=…
FXS[<axis>]=… FXST[<axis>]=…
FXS[<axis>]=… FXST[<axis>]=… FXSW[<axis>]=…
```

Meaning

`FXS`:	Command for activation and deactivation of the "Travel to fixed stop" function	
	`FXS[<axis>]=1`:	Activate function
	`FXS=[<axis>]=0`:	Deactivate function
`FXST`:	Optional command for setting the clamping torque Specified as % of the maximum drive torque	
`FXSW`:	Optional command for setting the window width for the fixed stop monitoring Specified in mm, inches or degrees	
`<axis>`:	Machine axis name Machine axes (X1, Y1, Z1, etc.) are programmed	

Note

The commands `FXS`, `FXST` and `FXSW` are modal.

The programming of `FXST` and `FXSW` is optional: If no parameter is specified, the last programmed value or the value set in the relevant machine data applies.

Activate travel to fixed stop: FXS[<axis>] = 1

The movement to the destination point can be described as a path or positioning axis movement. With positioning axes, the function can be performed across block boundaries.

Travel to fixed stop can be performed simultaneously for several axes and parallel to the movement of other axes. The fixed stop must be located between the start and end positions.

NOTICE

Risk of collision

It is not permissible to program a new position for an axis if the "Travel to fixed stop" function has already been activated for an axis/spindle.

Spindles must be switched to position-controlled mode before the function is selected.

Example:

Program code	Comment
X250 Y100 F100 FXS[X1]=1 FXST[X1]=12.3 FXSW[X1]=2	; Axis X1 travels with feedrate F100 (specification optional) to target position X=250 mm. The clamping torque is 12.3% of the maximum drive torque, monitoring is performed in a 2 mm wide window.
...	

Deactivate travel to fixed stop: FXS[<axis>] = 0

Deselection of the function triggers a preprocessing stop.

The block with `FXS[<axis>]=0` may and should contain traversing movements.

NOTICE

Risk of collision

The traversing movement to the retraction position must move away from the fixed stop, otherwise damage to the stop or to the machine may result.

The block change takes place when the retraction position has been reached. If no retraction position is specified, the block change takes place immediately after the torque limit has been deactivated.

Example:

Program code	Comment
X200 Y400 G01 G94 F2000 FXS[X1]=0	; Axis X1 is retracted from the fixed stop to position X = 200 mm. All other parameters are optional.
...	

Clamping torque (FXST) and monitoring window (FXSW)

Any programmed torque limiting `FXST` is effective from the block start, i.e. the fixed stop is also approached at a reduced torque. `FXST` and `FXSW` can be programmed and changed in the part program at any time. The changes take effect before traversing movements in the same block.

NOTICE
Risk of collision Programming of a new fixed stop monitoring window causes a change not only in the window width, but also in the reference point for the center of the window if the axis has moved prior to reprogramming. The actual position of the machine axis when the window is changed is the new window center point. The window must be selected such that only a breakaway from the fixed stop causes the fixed stop monitoring to respond.

Further information

Rise ramp

A rate of rise ramp for the new torque limit can be defined in MD to prevent any abrupt changes to the torque limit setting (e.g. insertion of a quill).

Alarm suppression

The fixed stop alarm can be suppressed for applications by the part program by masking the alarm in a machine data item and activating the new MD setting with NEW_CONF.

Activation

The commands for travel to fixed stop can be called from synchronized actions or technology cycles. They can be activated without initiation of a motion, the torque is limited instantaneously. As soon as the axis is moved via a setpoint, the limit stop monitor is activated.

Activation from synchronized actions

Example:

If the expected event ($R1) occurs and travel to fixed stop is not yet running, `FXS` should be activated for axis Y. The torque must correspond to 10% of the rated torque value. The width of the monitoring window is set to the default.

Program code

```
N10 IDS=1 WHENEVER (($R1=1) AND ($AA_FXS[Y]==0)) DO $R1=0 FXS[Y]=1
FXST[Y]=10
```

The normal part program must ensure that $R1 is set at the desired point in time.

Deactivation from synchronized actions

Example:

If an anticipated event ($R3) has occurred and the status "Limit stop contacted" (system variable $AA_FXS) is reached, then FXS must be deselected.

Program code
`IDS=4 WHENEVER (($R3==1) AND ($AA_FXS[Y]==1)) DO FXS[Y]=0` `FA[Y]=1000 POS[Y]=0`

Fixed stop reached

When the fixed stop has been reached:

- The distance-to-go is deleted and the set position is tracked.
- The drive torque increases up to the programmed limit value `FXSW`, and then remains constant.
- Fixed stop monitoring is activated within the specified window width.

Supplementary conditions

- Measurement with delete distance-to-go
  "Measurement with delete distance-to-go" (`MEAS` command) and "Travel to fixed stop" cannot be programmed at the same time in one block.
  Exception: One function acts on a path axis and the other on a positioning axis or both act on positioning axes.
- Contour monitoring
  Contour monitoring is not performed while "Travel to fixed stop" is active.
- Positioning axes
  For "Travel to fixed stop" with positioning axes, the block change is performed irrespective of the fixed stop movement.
- Travel to fixed stop is **not** possible:
  - With gantry axes
  - For competing positioning axes that are controlled exclusively from the PLC (`FXS` must be selected from the NC program).
- If the torque limit is reduced too far, the axis will not be able to follow the specified setpoint; the position controller then goes to the limit and the contour deviation increases. In this operating state, an increase in the torque limit may result in sudden, jerky movements. To ensure that the axis can follow the setpoint, check the contour deviation to make sure it is not greater than the deviation with an unlimited torque.

3.14.7 Dwell time (G4)

With the command `G4`, a time (dwell time) is programmed in a block that expires as soon as the block is executed in the main run. The block change to the following block is performed as soon as the time has completely expired.

Note

`G4` interrupts continuous-path mode.

Syntax

```
G4 F<Time>
G4 S<NumSpi>
G4 S<n> = <NumSpi>
```

Meaning

G4:	Activate dwell time	
	Alone in the block:	Yes
F<Time>:	The dwell time <Time> in seconds is specified at address F.	
S<NumSpi>:	The dwell time is programmed at address S in spindle revolutions <NumSpi> with reference to the current main spindle.	
S<n>=NumSpi>:	The dwell time is programmed at address S in spindle revolutions <NumSpi> with reference to the spindle addressed with the address extension <n>.	

Note

The addresses F and S used for the time specified in the dwell block G4 do not influence the feedrates F... and the spindle speeds S... of the program.

Supplementary conditions

Synchronized actions

Two synchronized actions are programmed in one program in such a way that the following block with the dwell time becomes the action block in which the synchronized actions are performed. One synchronized action is a modal synchronized action. The other synchronized action is a non-modal synchronized action. If the non-modal synchronized action is intended to influence the model synchronized action, e.g. release it for execution with UNLOCK, **at least two interpolation cycles** e.g. G4 F<interpolator_cycle * 2> must be provided as the effective dwell time.

The effective dwell time depends on the setting in the machine data MD10280 $MN_PROG_FUNCTION_MASK, Bit 4 = <value>

Value	Meaning
0	The effective dwell time is equal to the programmed dwell time
1	The effective dwell time is equal to the programmed dwell time rounded to the next largest multiple of the interpolator cycle (MD10071 $MN_IPO_CYCLE_TIME)

Program example:

- MD10071 $MN_IPO_CYCLE_TIME == 8 ms
- MD10280 $MN_PROG_FUNCTION_MASK, Bit 4 = 1

Program code	Comment
N10 WHEN TRUE DO LOCK(1)	; Non-modal SynAct: LOCK of the
	; modal SynAct. ID=1

Program code	Comment
N20 G4 F2	; Action block for SynAct from N10
N30 WHEN TRUE DO UNLOCK(1)	; Non-modal SynAct: UNLOCK
	; of the modal SynAct. ID=1
N40 ID=1 WHENEVER TRUE DO $R0=1 RDISABLE	; Modal SynAct ID=1
	; R parameter R0=1
	; Set the read-in disable
N50 G4 F0.012	; Action block for SynAct from N40 and N50
	; See paragraph "Description" below
N60 G4 F10	

Description

The desired behavior is that the modal synchronized action from N30 cancels the active lock (LOCK) of the modal synchronized action with ID=1 from N40, causing the R parameter to be written in N50 and the read-in disable to become active. This behavior is only achieved if the active dwell time is at least two interpolation cycles long.

The active dwell time results from the programmed dwell time, the interpolation cycle, and the setting in MD10280 $MN_PROG_FUNCTION_MASK, Bit 4. To ensure that the active dwell time is at least two interpolation cycles long, the following dwell time must be programmed:

- Bit 4 == 0: Programmed dwell time ≥ 2 * interpolator cycle
- Bit 4 == 1: Programmed dwell time ≥ 1.5 * interpolator cycle

If the active dwell time is shorter than two interpolation cycles, writing the R parameter and read-in disable will not be executed until block N60.

Example

Program code	Comment
N10 G1 F200 Z-5 S300 M3	;Feed F; spindle speed S
N20 G4 F3	; Dwell time: 3 s
N30 X40 Y10	
N40 G4 S30	; Dwelling 30 revolutions of the spindle (at S=300 rpm and 100% speed override, corresponds to: t = 0.1 min).
N50 X...	; The feedrate and spindle speed programmed in N10 continue to apply.

3.14.8 Internal preprocessing stop

Function

The control generates an internal preprocessing stop on access to machine status data ($A...). The following block is not executed until all preprocessed and saved blocks have been executed in full. The previous block is stopped in exact stop (as `G9`).

Example

Program code	Comments
...	
N40 POSA[X]=100	
N50 IF $AA_IM[X]==R100 GOTOF MARKE1	; Access to machine status data ($A...), the control generates an internal pre-processing stop.
N60 G0 Y100	
N70 WAITP(X)	
N80 LABEL1:	
...	

3.15 Other information

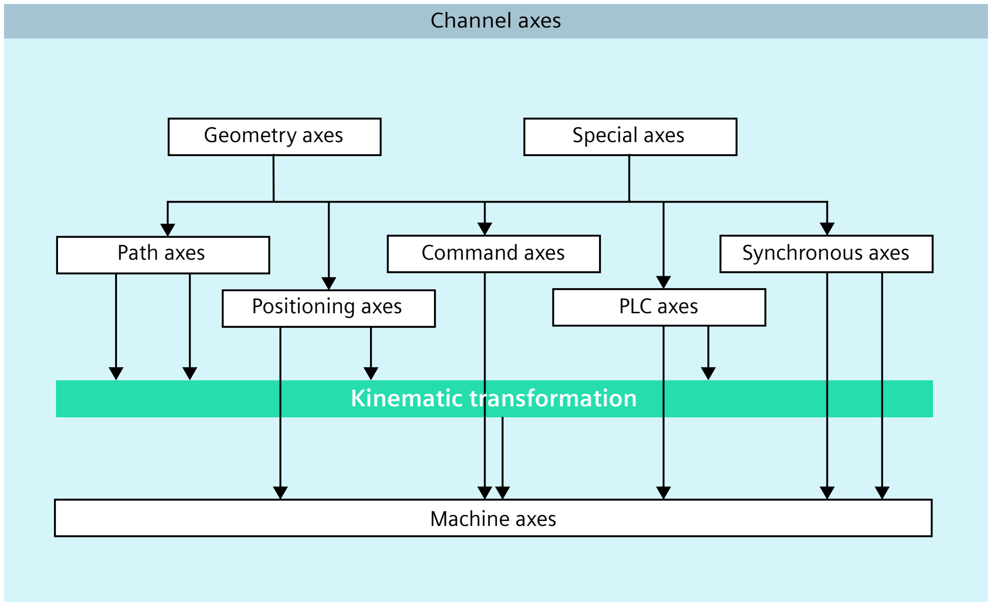
3.15.1 Axes

3.15.1.1 Axes (overview)

Axis types

A distinction is made between the following types of axis types when programming:

- Main axes / geometry axes
- Special axes
- Main spindle, master spindle
- Machine axes
- Channel axes
- Path axes
- Positioning axes
- Synchronized axes
- Command axes
- PLC axes / competing positioning axes



3.15.1.2 Main axes/Geometry axes

The main axes define a right-angled, right-handed coordinate system. Tool movements are programmed in this coordinate system.

In NC technology, the main axes are called geometry axes. This term is also used in this Programming Guide.

Replaceable geometry axes

The "Replaceable geometry axes" function (see Function Manual, Job Planning) can be used to alter the geometry axes grouping configured using machine data from the part program. Here any geometry axis can be replaced by a channel axis defined as a synchronous special axis.

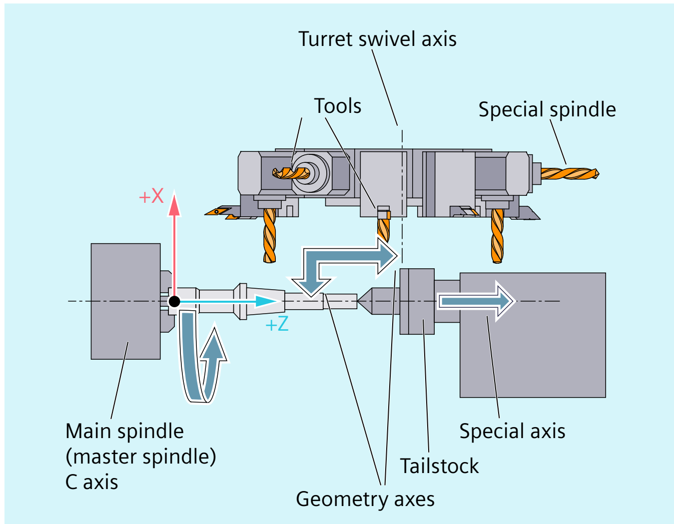
Axis identifier

The name/identifier of a geometry axis can be defined using the following machine data:

MD20060 $MC_AXCONF_GEOAX_NAME_TAB (name of the geometry axis in the channel)

Standard identifier for turning machines:

1. Geometry axis: X
2. Geometry axis: Z



Standard identifier for milling machines:

1. Geometry axis: X
2. Geometry axis: Y
3. Geometry axis: Z

Further information

A maximum of three geometry axes are used for programming frames and the workpiece geometry (contour).

The identifiers for geometry and channel axes may be the same, provided a reference is possible.

Geometry and channel axis names must be the same in all channels. This means that a program can be executed in any channel.

3.15.1.3 Special axes

In contrast to the geometry axes, no geometrical relationship is defined between the special axes.

Typical special axes are:

- Tool revolver axes
- Swivel table axes
- Swivel head axes
- Loader axes

Axis identifier

On a turning machine with circular magazine, for example:

- Revolver position U
- Tailstock V

Programming example

Program code	Comment
N10 G1 X100 Y20 Z30 A40 F300	; Path axis movements
N20 POS[U]=10POS[X]=20 FA[U]=200 FA[X]=350	; Positioning axis movements.
N30 G1 X500 Y80 POS[U]=150FA[U]=300 F550	; Path and positioning axis.
N40 G74 X1=0 Z1=0	; Approach reference point.

3.15.1.4 Main spindle, master spindle

The machine kinematics determine, which spindle is the main spindle. This spindle is usually declared as the master spindle in the machine data.

This assignment can be changed with the `SETMS(<spindle number>)` program command. `SETMS` can be used without specifying a spindle number to switch back to the master spindle defined in the machine data.

Special functions such as thread cutting are supported by the master spindle.

Spindle identifier

S or S0

3.15.1.5 Machine axes

Machine axes are the axes physically existing on a machine.

The programmed motion of a path or additional axis can act on several machine axes due to transformation (TRANSMIT, TRACYL or TRAORI) active in the channel.

Machine axes are only directly addressed in the program in special circumstances (e.g. for reference point or fixed point approach).

Axis identifier

The name/identifier of a machine axis can be defined using the following NC-specific machine data:

MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)

Default setting: X1, Y1, Z1, A1, B1, C1, U1, V1

Further, machine axes have fixed axis identifiers, which can always be used, independent of the names set in the machine data:

AX1, AX2, ..., AX<n>

3.15.1.6 Channel axes

All geometry, special and machine axes, which are assigned to a channel, are called channel axes.

Axis identifier

The channel-specific name/identifier of a geometry and special axis can be defined using the following machine data:

MD20080 $MC_AXCONF_CHANAX_NAME_TAB (channel axis name)

Default setting: X, Y, Z, A, B, C, U, V

The assignment regarding on which machine axis a geometry or special axis is emulated in the channel, is defined in the following machine data:

MD20070 $MC_AXCONF_MACHAX_USED (machine axes used)

3.15.1.7 Path axes

Path axes define the path and therefore the movement of the tool in space.

The programmed feed is active for this path. The axes involved in this path reach their position at the same time. As a rule, these are the geometry axes.

However, default settings define, which axes are the path axes, and therefore determine the velocity.

Path axes can be specified in the NC program with `FGROUP`.

For more information about `FGROUP`, see "Feedrate (G93, G94, G95, F, FGROUP, FL, FGREF) (Page 115)".

3.15.1.8 Positioning axes

Positioning axes are interpolated separately; in other words, each positioning axis has its own axis interpolator and its own feedrate. Positioning axes do not interpolate with the path axes.

Positioning axes are traversed by the NC program or the PLC. If an axis is to be traversed simultaneously by the NC program and the PLC, an error message appears.

Typical positioning axes are:

- Loaders for moving workpieces to the machine
- Loaders for moving workpieces away from the machine
- Tool magazine/turret

Types

A distinction is made between positioning axes with synchronization at the block end or over several blocks.

POS axes

Block change occurs at the end of the block when all the path and positioning axes programmed in this block have reached their programmed end point.

POSA axes

The movement of these positioning axes can extend over several blocks.

POSP axes

The movement of these positioning axes for approaching the end position takes place in sections.

Note

Positioning axes become synchronized axes if they are traversed without the special POS/POSA identifier.

Continuous-path mode (G64) for path axes is only possible if the positioning axes (POS) reach their final position before the path axes.

Path axes programmed with `POS`/`POSA` are removed from the path axis grouping for the duration of this block.

For more information about `POS`, `POSA`, and `POSP`, see "Traverse positioning axes (POS, POSA, POSP, FA, WAITP, WAITMC) (Page 123)".

3.15.1.9 Synchronized axes

Synchronized axes traverse synchronously to the path from the start position to the programmed end position.

The feedrate programmed in `F` applies to all the path axes programmed in the block, but does not apply to synchronized axes. Synchronized axes take the same time as the path axes to traverse.

A synchronized axis can be a rotary axis, which is traversed synchronously to the path interpolation.

3.15.1.10 Command axes

Command axes are started from synchronized actions in response to an event (command). They can be positioned, started and stopped fully asynchronous to the part program. An axis cannot be moved from the part program and from synchronized actions simultaneously.

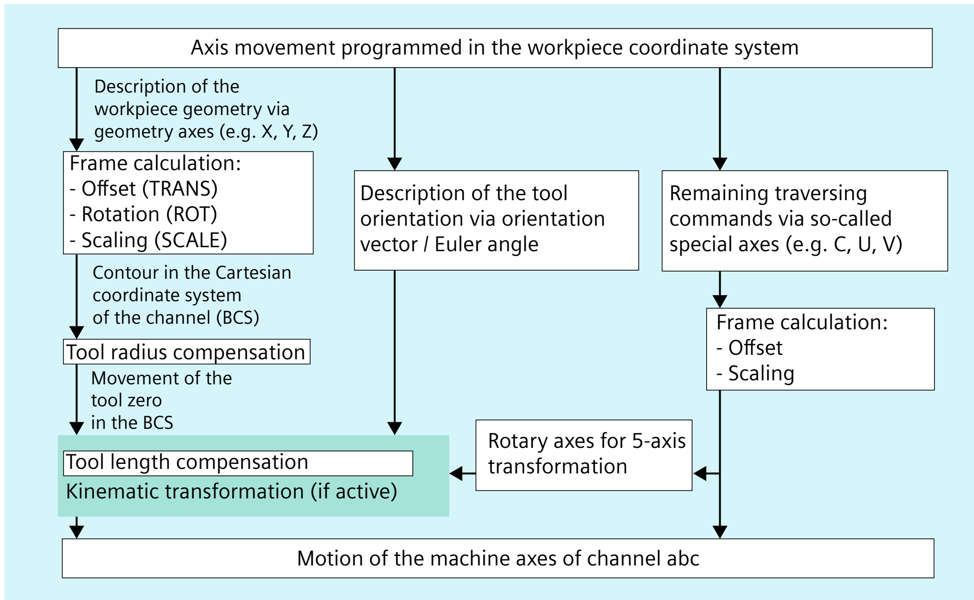
Command axes are interpolated separately; in other words, each command axis has its own axis interpolator and its own feedrate.

3.15.1.11 PLC axes

PLC axes are traversed by the PLC via special function blocks in the basic program; their movements can be asynchronous to all other axes. Traversing movements take place independently of path and synchronized movements.

3.15.2 From travel command to machine movement

The relationship between the programmed axis movements (travel commands) and the resulting machine movements is illustrated in the following figure:

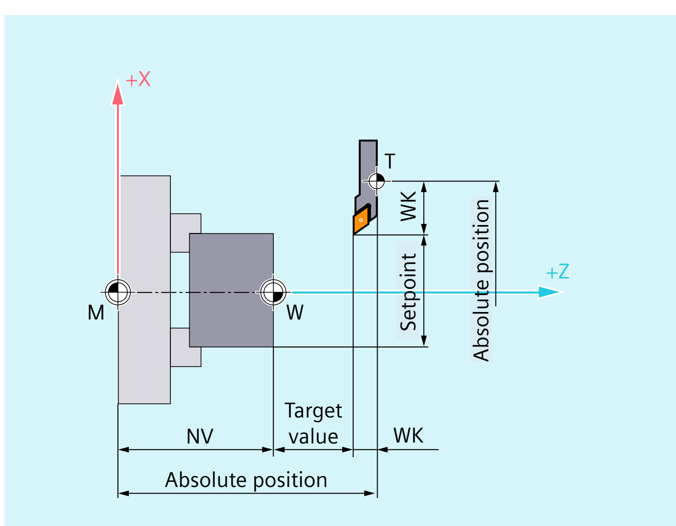


3.15.3 Path calculation

The path calculation determines the distance to be traversed in a block, taking into account all offsets and compensations.

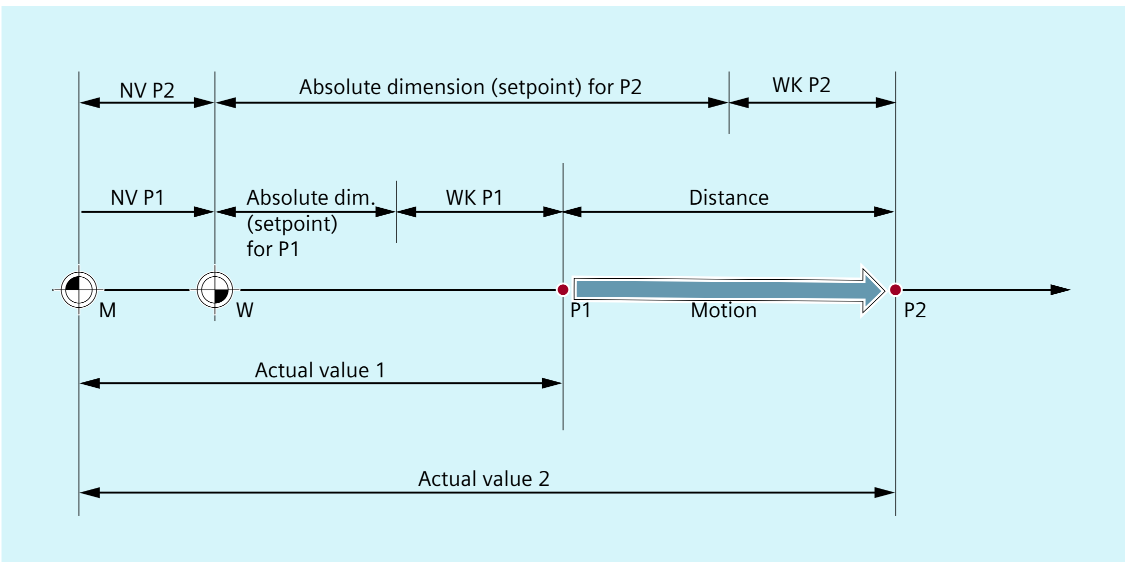
In general:

Path =
setpoint - actual value + zero offset (ZO) + tool offset (TO)



If a new zero offset and a new tool offset are programmed in a new program block, the following applies:

- With absolute dimensioning:
  Path = (absolute dimension P2 - absolute dimension P1) + (WO P2 - WO P1) + (TO P2 - TO P1).
- With incremental dimensioning:
  Path = incremental dimension + (WO P2 - WO P1) + (TO P2 - TO P1).



3.15.4 Addresses

Fixed addresses

These addresses are permanently set, that is the address characters cannot be changed.

A list can be found in Table "Fixed addresses (Page 1166)".

Settable addresses

The machine manufacturer may assign another name to these addresses via machine data.

Note

Settable addresses must be unique within the control, i.e. the same address name must not be used for different address types (axis values and end points, tool orientation, interpolation parameters, etc.).

A list can be found in Table "Settable addresses (Page 1171)".

Modal/non-modal addresses

Modal addresses remain valid with the programmed value (in all subsequent blocks) until a new value is programmed at the same address.

Non-modal addresses only apply in the block, in which they were programmed.

Example:

Program code	Comment
N10 G01 F500 X10	
N20 X10	; Feedrate F from N10 remains active until a new feedrate is entered.

Addresses with axial extension

In addresses with axial extension, an axis name is inserted in square brackets after the address. The axis name assigns the axis.

Example:

Program code	Comment
FA[U]=400	; Axis-specific feedrate for U axis.

See also Table "Fixed addresses (Page 1166)".

Extended address notation

Extended address notation enables a larger number of axes and spindles to be organized in a system.

An extended address consists of a numeric extension and an arithmetic expression assigned with an "=" character. The numeric extension has one or two digits and is always positive.

The extended address notation is only permitted for the following direct addresses:

Address	Meaning
X, Y, Z, ...	Axis addresses
I, J, K	Interpolation parameters
S	Spindle speed
SPOS, SPOSA	Spindle position
M	Special functions
H	Auxiliary functions
T	Tool number
F	Feedrate

Examples:

Program code	Comment
X7	; No "=" required, 7 is a value, but the "=" character can also be used here

Program code	Comment
X4=20	; Axis X4; "=" is required
CR=7.3	; Two letters; "=" are required
S1=470	; Speed for 1st spindle: 470 rpm
M3=5	; Spindle stop for 3rd spindle

The numeric extension can be replaced by a variable for addresses M, H, S and for SPOS and SPOSA. The variable identifier is enclosed in square brackets.

Examples:

Program code	Comment
S[SPINU]=470	; Speed for the spindle whose number is stored in the SPINU variable.
M[SPINU]=3	; Clockwise rotation for the spindle whose number is stored in the SPINU variable.
T[SPINU]=7	; Selection of the tool for the spindle whose number is stored in the SPINU variable.

3.15.5 Names

The commands according to DIN 66025 are supplemented with named objects, etc. by the NC high-level language.

Examples of named objects:

- System variables
- User-defined variables
- Axes/spindles
- Subprograms
- Keywords
- Jump markers
- Macros

Note

Identifiers must be unique. It is **not** permissible to use the same identifier for different objects.

Naming rules

A name can be chosen freely providing the following rules are observed:

- Permissible characters:
  – Letters: A ... Z, a ... z
  – Numbers: 0 ... 9
  – Underscore: _
- The first two characters should be letters or underscores.
- Maximum length:
  – Program names (Page 42): 24 characters
  – Axis names: 8 characters
  – Variable names: 31 characters

Note

Reserved keywords must not be used as identifiers.

Cycles

To prevent name conflicts, we recommend that the following specification for the assignment of names for user cycles is observed:

Character string	Reserved for names for
• CYCLE • CUST_ • GROUP_ • _ • S_ • E_ • F_	SIEMENS cycles
• CCS_	SIEMENS compile cycles
• CC_	User compile cycles

User cycles

We recommend that the names of user cycles begin with U_.

Variables

Information relating to the name assignment for variables is provided in the following chapters:

- System data (Page 378)
- Definition of user variables (DEF) (Page 383)

3.15.6 Constants

Constant (general)

A constant is a data element whose value does not change during the execution of a program, e.g. a value assignment to an address.

Decimal constant

The numeric value of a decimal constant is displayed in the decimal system.

INTEGER constant

An INTEGER constant is an integer value, i.e. a sequence of digits without decimal point, with or without sign.

Examples:

`X10`	Assignment of the value +10 to address X
`X-35`	Assignment of the value -35 to address X
`X0`	Assignment of the value 0 to address X **Note:** X0 cannot be replaced by X.

REAL constant

A REAL constant is a sequence of digits with decimal point, with or without sign and with or without exponent.

Examples:

`X10.25`	Assignment of the value +10.25 to address X
`X-10.25`	Assignment of the value -10.25 to address X
`X0.25`	Assignment of the value +0.25 to address X
`X.25`	Assignment of the value +0.25 to address X without leading "0"
`X=-.1EX-3`	Assignment of the value $-0.1*10^{-3}$ to address X

Note

If, in an address, which permits decimal point input, more decimal places are specified than actually provided for the address, then they are rounded to fit the number of places provided.

Hexadecimal constant

Constants can also be interpreted as hexadecimal format, i.e. based on 16. The letters A to F are hexadecimal digits with the decimal values 10 to 15.

Hexadecimal constants are enclosed in single quotation marks and start with the letter "H", followed by the value in hexadecimal notation. Separators are permitted between the letters and digits.

Example:

Program code	Comment
`$MC_TOOL_MANAGEMENT_MASK='H7F'`	`; By assigning the hexadecimal constant, bits 0 to 7 are set in the machine data.`

Note

The maximum number of characters is limited by the value range of the integer data type.

Binary constant

Constants can also be interpreted in binary format. In this case, only the digits "0" and "1" are used.

Binary constants are enclosed in single quotation marks and start with the letter "B", followed by the binary value. Separators are permitted between the digits.

Example:

Program code	Comment
`$MN_AUXFU_GROUP_SPEC='B10000001'`	`; By assigning the binary constant, bit 0 and bit 7 are set in the machine data.`

Note

The maximum number of characters is limited by the value range of the integer data type.

3.15.7 Operators and arithmetic functions

Operators

Arithmetic operators

System variables of the REAL and INT type can be linked by the following operators:

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication

Operator	Meaning
/	• Division in synchronous actions: INT / INT ⇒ **INT** • Division in synchronous actions with REAL result by using the function ITOR(): ITOR(INT) / ITOR(INT) ⇒ **REAL** • Division in NC programs: INT / INT ⇒ **REAL**
DIV	Integer division: INT / INT ⇒ **INT**
MOD	Modulo division (only for type INT) supplies remainder of an INT division Example: 3 MOD 4 = 3

Note

Only variables of the same type may be linked by these operations.

Relational operators

Operator	Meaning
==	Equal to
>	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Boolean operators

Operator	Meaning
NOT	NOT
AND	AND
OR	OR
XOR	Exclusive OR

Bit logic operators

Operator	Meaning
B_OR	Bit-by-bit OR
B_AND	Bit-by-bit AND
B_XOR	Bit-by-bit exclusive OR
B_NOT	Bit-by-bit negation

Priority of the operators

The operators have the following priorities for execution in the synchronous action (highest priority: 1):

Priority	Operators	Meaning
1	NOT, B_NOT	Negation, bit-by-bit negation
2	*, /, DIV, MOD	Multiplication, division
3	+, -	Addition, subtraction
4	B_AND	Bit-by-bit AND
5	B_XOR	Bit-by-bit exclusive OR
6	B_OR	Bit-by-bit OR
7	AND	AND
8	XOR	Exclusive OR
9	OR	OR
10	<<	Concatenation of strings, result type STRING
11	==, <>, <, >, >=, <=	Relational operators

Note

It is strongly recommended that the individual operators are clearly prioritized by setting parentheses "(...)" when several operators are used in an expression.

Example of a condition with an expression with several operators:

```
Program code
... WHEN ($AA_IM[X] > VALUE) AND ($AA_IM[Y] > VALUE1) DO ...
```

Arithmetic functions

Operator	Meaning
SIN()	Sine
COS()	Cosine
TAN()	Tangent
ASIN()	Arc sine
ACOS()	Arc cosine
ATAN2()	Arc tangent 2
SQRT()	Square root
ABS()	Absolute value
POT()	2nd power (square)
TRUNC()	Integer component The accuracy for comparison commands can be set using TRUNC
ROUND()	Round to an integer
LN()	Natural logarithm
EXP()	Exponential function

Indexing

The index of a system variable of type "Array of ..." can in turn be a system variable. The index is also evaluated in the main run in the interpolator clock cycle.

Example

Program code
`... WHEN … DO $AC_PARAM[$AC_MARKER[1]]=3`

Restrictions

- It is not permissible to nest indices with further system variables.
- The index must not be formed via preprocessing variables. The following example is therefore **not** permitted since $P_EP is a preprocessing variable:
  $AC_PARAM[1] = $P_EP[$AC_MARKER[0]]

See also

Arithmetic functions (Page 427)

Instructions (Page 427)

Work preparation 4

4.1 Flexible NC programming

4.1.1 Variables

The use of variables from the system data and user data areas, especially in conjunction with arithmetic functions and check structures, enables highly flexible NC programs and cycles to be written.

⚠ WARNING
Material damage and personal injuries caused by changed variables
When using variables in the NC program it must be taken into account that machine operators or unauthorized persons with corresponding access rights can change the variables and thus affect the program run. This can result in material damage and personal injuries.
• In order to avoid negative effects on the program run caused by changed variables, appropriate data checks ("input validation") are to be provided in the NC program.

- System data
  The system data contains the variables predefined in the system. These variables have a defined meaning. They are primarily used by the system software. The user can read and write these variables in NC programs and cycles. Example: Machine data, setting data, system variables.
  Although the meaning of a system data item is fixed, the user can modify its properties within certain limits by redefinition.
  See "Redefinition of system data, user data, and NC commands (REDEF) (Page 388)"
- User data
  The user data contains those variables defined by the user with meanings defined exclusively by the user. They are not evaluated by the system.
  The user data is divided into:
  - Predefined user variables
    Predefined user variables are variables that have already been defined in the system and whose number is parameterized in the machine data. The user can change the properties of these variables.
    See "Redefinition of system data, user data, and NC commands (REDEF) (Page 388)".
  - User-defined variables
    User-defined variables are variables that are defined by the user and are not created by the system until runtime. Their number, data type, visibility, and all other properties are defined exclusively by the user.
    See "Definition of user variables (DEF) (Page 383)"

4.1.1.1 System data

The system data contain the variables that are predefined in the system and enable access to the current parameter settings of the control, as well as to machine, control, and process states, in NC programs and cycles.

Preprocessing variables

Preprocessing variables are system data that are read and written during preprocessing, in other words, at the instant at which the block containing the variable is interpreted. Preprocessing variables do not trigger preprocessing stops.

Main run variables

Main run variables are system data that are read and written during the main run, in other words, at the instant at which the block containing the variable is executed. The following are main run variables:

- Variables that can be programmed in synchronized actions (read/write)
- Variables that can be programmed in the NC program and trigger preprocessing stops (read/write)
- Variables that can be programmed in the NC program and whose value is calculated during preprocessing but not written until the main run (main run synchronized: write only)

Prefix system

To distinguish system data from other data, their names are usually preceded by a prefix comprising the $ sign followed by one or two letters and an underscore.

$ + 1. Letter	Meaning: Data type
Preprocessing data (system data that are read/written during preprocessing)	
$M	Machine data 1)
$S	Setting data, protection areas 1)
$T	Tool management data
$P	Programmed values
$C	Cycle variables of ISO envelope cycles
$O	Option data
R	R-parameters (arithmetic parameters) 2)
Main run data (system data that are read/written during the main run)	
$$M	Machine data 1)
$$S	Setting data 1)
$A	Current main run data
$V	Position controller data

$ + 1. Letter	Meaning: Data type
$R	R-parameters (arithmetic parameters) [2)]
[1)] Whether machine and setting data is treated as preprocessing or main run variables depends on whether they are written with one or two $ characters. The notation is freely selectable for the specific application. [2)] When an R-parameter is used in the part program/cycle as a preprocessing variable, the prefix is omitted, e.g. R10. When it is used in a synchronized action as a main run variable, a $ sign is written as a prefix, e.g. $R10.	

2nd letter	Meaning: Visibility
N	NC global variable (**N**C)
C	Channel-specific variable (**C**hannel)
A	Axis-specific variable (**A**xis)

Supplementary conditions

Exceptions in the prefix system

The following system of variables deviate from the prefix system specified above:

- $TC_...: Here, the 2nd letter C does not refer to channel-specific system variables but to toolholder-specific system variables (TC= tool carrier).
- $P_ ...: Channel-specific system variables

Use of machine and setting data in synchronized actions

When machine and setting data is used in synchronized actions, the prefix can be used to define whether the machine or setting data will be read/written synchronous to the preprocessing run or the main run.

If the data remains unchanged during machining, it can be read synchronous to the preprocessing run. For this purpose, the machine or setting data prefix is written with a $ sign:

```
ID=1 WHENEVER $AA_IM[z] < $SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
```

If the data changes during machining, it must be read/written synchronous to the main run. For this purpose, the machine or setting data prefix is written with two $ signs:

```
ID=1 WHENEVER $AA_IM[z] < $$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
```

Note

Writing machine and setting data

When writing an item of machine or setting data, it is important to ensure that the access level which is active when the part program/cycle is executed permits write access and that the data is set to take "IMMEDIATE" effect.

See also

Variables (Page 377)

4.1.1.2 Predefined user variables: Channel-specific arithmetic parameters (R)

Channel-specific arithmetic parameters or R parameters are predefined user variables with the designation R, defined as an array of the REAL data type. For historical reasons, notation both with array index, e.g. `R[10]`, and without array index, e.g. `R10`, is permitted for R parameters.

When using synchronized actions, the $ sign must be included as a prefix, e.g. `$R10`.

Syntax

When used as a preprocessing variable:

```
R<n>
R[<expression>]
```

When used as a main run variable:

```
$R<n>
$R[<expression>]
```

Meaning

`R`:	Identifier when used as a preprocessing variable, e.g. in the part program	
`$R`:	Identifier when used as a main run variable, e.g. in synchronized actions	
	Type:	REAL
	Range of values:	For a non-exponential notation: ± (0.000 0001 ... 9999 9999) **Note:** A maximum of 8 decimal places are permitted
		For an exponential notation: $\pm (1*10^{-300} ... 1*10^{+300})$ **Note:** • Notation: <Mantissa>EX<Exponent> e.g. 8.2EX-3 • A maximum of 10 characters are permitted including sign and decimal point.
`<n>`:	Number of the R parameter	
	Type:	INT
	Range of values:	0 - MAX_INDEX **Note** MAX_INDEX is calculated from the parameterized number of R-parameters: MAX_INDEX = (MD28050 $MN_MM_NUM_R_PARAM) - 1
`<expression>`:	Array index Any expression can be used as an array index, as long as the result of the expression can be converted to the INT data type (INT, REAL, BOOL, CHAR).	

Example

Assignments to R-parameters and use of R-parameters in mathematical functions:

Program code	Comment
R0=3.5678	; Assignment in preprocessing
R[1]=-37.3	; Assignment in preprocessing
R3=-7	; Assignment in preprocessing
$R4=-0.1EX-5	; Assignment in the main program run: R4 = -0.1 * 10^-5
$R[6]=1.874EX8	; Assignment in the main program run: R6 = 1.874 * 10^8
R7=SIN(25.3)	; Assignment in preprocessing
R[R2]=R10	; Indirect addressing using R-parameter
R[(R1+R2)*R3]=5	; Indirect addressing using math. expression
X=(R1+R2)	; Traverse axis X to the position resulting from the sum of R1 and R2
Z=SQRT(R1*R1+R2*R2)	; Traverse axis Z to the square root position (R1^2 + R2^2)

See also

Variables (Page 377)

4.1.1.3 Predefined user variables: Global arithmetic parameters (RG)

Function

In addition to the channel-specific R parameters, the user has access to global R parameters. They exist once within the control unit and can be read and written from all channels.

Global R parameters are used, for example, to transfer information from one channel to the next. Another example concerns global settings that should be evaluated for all channels, such as the overhang of the raw part from the spindle.

The global R parameters are read and written from the user interface or in the NC program during the preprocessing. Synchronous actions and technology cycles cannot be used.

Note

No synchronization between the channels when reading and writing global R parameters.

Because the reading and writing is performed during the preprocessing, the point in time when a written value from one channel becomes active in another channel is not defined.

Example:

In channel 1, a loop runs with a global R parameter as loop counter. Channel 2 writes a value to this global R parameter; this causes a loop abort in channel 1. All loops that can be interpreted in the preprocessing in channel 1 are however still executed. The number of loops is not defined and depends on the channel loading, etc.

The user must implement a synchronization between the channels as application, e.g. with WAIT flags!

Syntax

Writing in the NC program

```
RG[<n>]=<value>
RG[<expression>]=<value>
```

Reading in the NC program

```
R...=RG[<n>]
R...=RG[<expression>]
```

Meaning

`RG :`	Default name of the NC address for **global** R parameters **Note:** The name of the NC address can be set via MD15800 $MN_R_PARAM_NCK_NAME	
`<n>:`	Number of the global R parameter	
	Type:	INT
	Range of values:	0 ... MAX_INDEX **Note** MAX_INDEX is calculated from the parameterized number of global R parameters: MAX_INDEX = (MD18156 $MN_MM_NUM_R_PARAM_NCK) - 1
`<expression>:`	Any expression can be used as an array index, as long as the result of the expression can be converted to the INT data type (INT, REAL, BOOL, CHAR).	

`<value>`:	Value of the global R parameter	
	Type:	REAL
	Range of values:	For a non-exponential notation: ± (0.000 0001 ... 9999 9999) **Note:** A maximum of eight decimal places are permitted
		For an exponential notation: ± ($1*10^{-300}$ ... $1*10^{+300}$) **Note:** • Notation: <mantissa>EX<exponent> e.g. 8.2EX-3 • A maximum of ten characters are permitted including sign and decimal point.

4.1.1.4 Definition of user variables (DEF)

With the `DEF` command, you can define user-specific variables, or user variables (user data), and assign values to them.

According to the range of validity (in other words, the range in which the variable is visible) there are the following categories of user variables:

- Local user variables (LUD)
  Local user variables (LUD) are variables defined in an NC program that is not the main program at the time of execution. They are created when the NC program is called, and deleted with an end of program reset – or the next time that the control system powers up. Local user variables can only be accessed within the NC program in which they are defined.
- Program-global user variables (PUD)
  Program-global user variables (PUD) are user variables defined in an NC program used as the main program. They are created when the NC program is called, and deleted with an end of program reset – or the next time that the control system powers up. It is possible to access PUD in the main program and in all subprograms of the main program.

  Note

  Availability of program-global user variables (PUD)

  Program-global user variables (PUD) defined in the main program are only available in subprograms if the following machine data is set:

  MD11120 $MN_LUD_EXTENDED_SCOPE = 1

  If MD11120 = 0 the program-global user variables defined in the main program will only be available in the main program.

- Global user variables (GUD)
  Global user variables (GUD) are NC or channel-global variables which are defined in a data block (SGUD, MGUD, UGUD, GUD4 to GUD9) and are kept even after an end of program reset or the next time that the control system powers up. GUD can be accessed in all NC programs.

User variables must be defined before they can be used (read/write). The following rules must be observed in this context:

- GUDs must be defined in a definition file, e.g. _N_DEF_DIR/_N_UGUD_DEF.
- PUDs and LUDs must be defined in the definition section of the NC program.
- The data must be defined in a dedicated block.
- Only one data type may be used for each data definition.
- Several variables of the same data type can be defined for each data definition.

Syntax

LUD and PUD

```
DEF <type> <phys_unit> <limit values> <name>[<value_1>, <value_2>,
<value_3>]=<init_value>
```

GUD

```
DEF <range> <pp_stop> <access_rights> <data class> <type>
<phys_unit> <limit values> <name>[<value_1>, <value_2>,
<value_3>]=<init_value>
```

Meaning

DEF:	Command for defining GUD, PUD, LUD user variables	
<range>:	Range of validity, only relevant for GUD:	
	NC:	NC-global user variable
	CHAN:	Channel-global user variable
<PP_stop>:	Preprocessing stop, only relevant for GUD (optional)	
	SYNR:	Preprocessing stop when reading
	SYNW:	Preprocessing stop when writing
	SYNRW:	Preprocessing stop when reading/writing
<access rights>:	Protection level for reading/writing GUD via NC program or OPI (optional)	
	APRP <protection level>:	Read: NC program
	APWP <protection level>:	Write: NC program
	APRB <protection level>:	Read: OPI
	APWB <protection level>:	Write: OPI
	<protection level>:	Range of values: 0 ... 7
	See "Attribute: Access rights (APR, APW, APRP, APWP, APRB, APWB) (Page 399)"	
<data class>:	Data class assignment (only SINUMERIK 828D) (Page 403)	
	DCM:	Data class M (= Manufacturer)
	DCI:	Data class I (= Individual)
	DCU:	Data class U (= User)

`<type>:`	Data type:	
	`INT`:	Integer with sign
	`REAL`:	Real number (LONG REAL to IEEE)
	`BOOL`:	Truth value TRUE (1)/FALSE (0)
	`CHAR`:	ASCII character
	`STRING[<MaxLength>]`:	Character string of a defined length
	`AXIS`:	Axis/spindle identifier
	`FRAME`:	Geometric data for a static coordinate transformation
	See "Data types (Page 411)"	
`<phys_unit>:`	Physical unit (optional)	
	`PHU` <unit>:	Physical unit
	See "Attribute: Physical unit (PHU) (Page 396)"	
`<limit values>:`	Lower/upper limit value (optional)	
	`LLI` <limit value>:	Lower limit value (lower limit)
	`ULI` <limit value>:	Upper limit value (upper limit)
	See "Attribute: Limit values (LLI, ULI) (Page 395)"	
`<name>:`	Name of variable **Note** • Maximum 31 characters • The first two characters must be a letter and/or an underscore. • The $ sign is reserved for system variables and must not be used.	
`[<value_1>, <value_2>, <value_3>]:`	Specification of array sizes for 1- to max. 3-dimensional array variables (optional) For the Initialization of array variables see "Definition and initialization of array variables (DEF, SET, REP) (Page 405)"	
`<init_value>:`	Initialization value (optional) See "Attribute: Initialization value (Page 392)" For the Initialization of array variables see "Definition and initialization of array variables (DEF, SET, REP) (Page 405)"	

Examples

Example 1: Definition of user variables in the data block for machine manufacturers

```
Program code                                        Comment
%_N_MGUD_DEF                                        ; GUD block: Machine manufacturer
$PATH=/_N_DEF_DIR
DEF CHAN REAL PHU 24 LLI 0 ULI 10 CURRENT_1, CURRENT_2
;Description
;Definition of two GUD items: CURRENT_1, CURRENT_2
;Range of validity: Throughout the channel
;Data type: REAL
```

Program code	Comment
PP stop: Not programmed => default value = no PP stop	
; phys. unit: 24 = [A]	
;Limit values: Low = 0.0, high = 10.0	
;Access rights: Not programmed => default value = 7 = key-operated switch position 0	
;Initialization value: Not programmed => default value = 0.0	
DEF NCK REAL PHU 13 LLI 10 APWP 3 APRP 3 APWB 0 APRB 2 TIME_1=12, TIME_2=45	
;Description	
;Definition of two GUD items: TIME_1, TIME_2	
;Range of validity: Throughout NC	
;Data type: REAL	
PP stop: Not programmed => default value = no PP stop	
; phys. unit: 13 = [s]	
;Limit values: low = 10.0, high = not programmed => upper definition range limit	
;Access rights:	
; NC program: Write/read = 3 = user	
; OPI: Write = 0 = Siemens, read = 3 = user	
;Initialization value: TIME_1 = 12.0, TIME_2 = 45.0	
DEF NCK APWP 3 APRP 3 APWB 0 APRB 3 STRING[5] GUD5_NAME = "COUNTER"	
;Description	
; Definition of one GUD item: GUD5_NAME	
;Range of validity: Throughout NC	
;Data type: STRING, max. 5 characters	
PP stop: Not programmed => default value = no PP stop	
; phys. unit: Not programmed => default value = 0 = no phys. unit	
;Limit values: Not programmed => definition range limits: Low = 0, high = 255	
;Access rights:	
; NC program: Write/read = 3 = user	
; OPI: Write = 0 = Siemens, read = 3 = user	
;Initialization value: "COUNTER"	
M30	

Example 2: Global program and local user variables (PUD/LUD)

Program code	Comment
PROC MAIN	; Main program
DEF INT VAR1	; PUD definition
...	
SUB2	; Subprogram call
...	
M30	

Program code	Comment
PROC SUB2	; Subprogram SUB2
DEF INT VAR2	; LUD DEFINITION
...	
IF (VAR1==1)	; Read PUD
VAR1=VAR1+1	; Read & write PUD
VAR2=1	; Write LUD
ENDIF	
SUB3	; Subprogram call
...	
M17	

Program code	Comment
PROC SUB3	; Subprogram SUB3
...	
IF (VAR1==1)	; Read PUD
VAR1=VAR1+1	; Read & write PUD
VAR2=1	; Error: LUD from SUB2 not known
ENDIF	
...	
M17	

Example 3: Definition and use of user variables of data type AXIS

Program code	Comment
DEF AXIS ABSCISSA	; 1st geometry axis
DEF AXIS SPINDLE	; Spindle
...	
IF ISAXIS(1) == FALSE GOTOF CONTINUE	
ABSCISSA = $P_AXN1	
CONTINUE:	
...	
SPINDLE=(S1)	; 1st Spindle
OVRA[SPINDLE]=80	; Spindle override = 80%
SPINDLE=(S3)	; 3rd Spindle

Supplementary conditions

Global user variables (GUD)

In the context of the definition of global user variables (GUD), the following machine data has to be taken into account:

No.	Identifier: $MN_	Meaning
11140	GUD_AREA_ SAVE_TAB	Additional save for GUD blocks
18118 1)	MM_NUM_GUD_MODULES	Number of GUD files in the active file system
18120 1)	MM_NUM_GUD_NAMES_NCK	Number of global GUD names
18130 1)	MM_NUM_GUD_NAMES_CHAN	Number of channel-specific GUD names
18150 1)	MM_GUD_VALUES_MEM	Memory location for global GUD values
18660 1)	MM_NUM_SYNACT_GUD_REAL	Number of configurable GUD of the REAL data type
18661 1)	MM_NUM_SYNACT_GUD_INT	Number of configurable GUD of the INT data type
18662 1)	MM_NUM_SYNACT_GUD_BOOL	Number of configurable GUD of the BOOL data type
18663 1)	MM_NUM_SYNACT_GUD_AXIS	Number of configurable GUD of the AXIS data type
18664 1)	MM_NUM_SYNACT_GUD_CHAR	Number of configurable GUD of the CHAR data type
18665 1)	MM_NUM_SYNACT_GUD_STRING	Number of configurable GUD of the STRING data type

1) For SINUMERIK 828D, MD can only be read!

Cross-channel use of an NC-global user variable of the AXIS data type

An NC-global user variable of the `AXIS` data type initialized during definition in the data block with an axis identifier can then only be used in other NC channels if the axis has the same channel axis number in these channels.

If this is not the case, the variable has to be loaded at the beginning of the NC program or, as in the following example, the AXNAME(...) function (see "Axis functions (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Page 791)") has to be used.

Program code	Comment
`DEF NCK STRING[5] ACHSE="X"`	`;Definition in the data block`
`...`	
`N100 AX[AXNAME(AXIS)]=111 G00`	`; Use in the NC program`

4.1.1.5 Redefinition of system data, user data, and NC commands (REDEF)

The `REDEF` command changes the attributes of system data, user data, and NC commands. A fundamental condition of redefinition is that it has to post-date the corresponding definition.

Multiple attributes cannot be changed simultaneously during redefinition. A separate `REDEF` command must be programmed for each attribute to be changed.

If several concurrent attribute changes are programmed, the last change is always active.

Resetting attribute values

The attributes for access rights and initialization time change with `REDEF` can be reset to their default values by reprogramming `REDEF`, followed by the name of the variable or the NC language command:

- Access rights: Protection level 7
- Initialization time: No initialization or retention of the current value

Redefinable attributes

See "Overview of definable and redefinable attributes (Page 404)".

Local user variables (PUD/LUD)

Redefinitions are not permitted for local user variables (PUD/LUD).

Syntax

```
REDEF <name> <PP_stop>
REDEF <name> <phys_unit>
REDEF <name> <limit_values>
REDEF <name> <access_rights>
REDEF <name> <init_time>
REDEF <name> <init_time> <init_value>
REDEF <name> <data class>
REDEF <name>
```

Meaning

`REDEF`:	Command for redefinition of a certain attribute or to reset the "Access rights" and/or "Initialization time" attributes of system variables, user variables and NC language commands	
`<name>`:	Name of an already defined variable or an NC language command	
`<PP stop>`:	Preprocessing stop	
	`SYNR`:	Preprocessing stop when reading
	`SYNW`:	Preprocessing stop when writing
	`SYNRW`:	Preprocessing stop when reading/writing
`<phys_unit>`:	Physical unit	
	`PHU` <unit>:	Physical unit
	See "Attribute: Physical unit (PHU) (Page 396)". **Note** Cannot be redefined for: • System variables • Global user data (GUD) of the data types: `BOOL`, `AXIS`, `STRING`, `FRAME`	

`<limit values>`:	Lower/upper limit	
	`LLI` <limit value>:	Lower limit value (lower limit)
	`ULI` <limit value>:	Upper limit value (upper limit)
	See "Attribute: Limit values (LLI, ULI) (Page 395)". **Note** Cannot be redefined for: • System variables • Global user data (GUD) of the data types: `BOOL`, `AXIS`, `STRING`, `FRAME`	
`<access rights>`:	Access rights for reading/writing via part program or OPI	
	`APX` <protection level>:	Execute: NC language element
	`APRP` <protection level>:	Read: Part program
	`APWP` <protection level>:	Write: Part program
	`APRB` <protection level>:	Read: OPI
	`APWB` <protection level>:	Write: OPI
	<protection level>:	Range of values: 0 ... 7
	See "Attribute: Access rights (APR, APW, APRP, APWP, APRB, APWB) (Page 399)".	
`<init_time>`:	Point in time at which the variable is reinitialized	
	`INIPO`:	Power On
	`INIRE`:	End of main program, NC reset or Power On
	`INICF`:	NEWCONF or main program end, NC reset or Power On
	`PRLOC`:	End of main program, NC reset following local change or Power On
	See "Attribute: Initialization value (Page 392)".	
`<init_value>`:	Initialization value When redefining the initialization value, an initialization time always has to be specified also (see <init_time>). See "Attribute: Initialization value (Page 392)". For the Initialization of array variables, see "Definition and initialization of array variables (DEF, SET, REP) (Page 405)". **Note** Cannot be redefined for system variables, except setting data.	
`<data class>`:	Data class assignment (only SINUMERIK 828D) (Page 403)	
	`DCM`:	Data class M (= Manufacturer)
	`DCI`:	Data class I (= Individual)
	`DCU`:	Data class U (= User)

Example

Redefinitions of system variable $TC_DPCx in the data block for machine manufacturers

```
Program code
%_N_MGUD_DEF                                  ; GUD block: Machine manufacturer
N100 REDEF $TC_DPC1 APWB 2 APWP 3
N200 REDEF $TC_DPC2 PHU 21
N300 REDEF $TC_DPC3 LLI 0 ULI 200
N400 REDEF $TC_DPC4 INIPO (100, 101, 102, 103)
N800 REDEF $TC_DPC1
N900 REDEF $TC_DPC4
M30
```

regarding N100:	Write access: OPI = protection level 2, part program = protection level 3
regarding N200:	Physical unit [%]
regarding N300:	Lower limit value = 0, upper limit value = 200
regarding N400:	The array variable is initialized with the four values at POWER ON.
regarding N800 / N900	Reset of the "Access rights" and/or "Initialization time" attribute values

Note

Use of ACCESS files

If ACCESS files are used, the redefinition of access rights has to be relocated from _N_MGUD_DEF to _N_MACCESS_DEF.

Supplementary conditions

Granularity

A redefinition is always applied to the entire variable which is uniquely identified by its name. Array variables do not, for example, support the assignment of different attributes to individual array elements.

4.1.1.6 Attribute: Initialization value

Definition of user variables (DEF)

During definition, an initialization value can be preassigned for the following user variables:

- Global user data (GUD)
- Program-global user variables (PUD)
- Local user variables (LUD)

Redefinition of system and user variables (REDEF)

During redefinition, an initialization value can be preassigned for the following variables:

- System data
  - Setting data
- User data
  - R parameters
  - Synchronized action variables ($AC_MARKER, $AC_PARAM, $AC_TIMER)
  - Synchronized action GUD (SYG_xy[ ], where x=R, I, B, A, C, S and y=S, M, U, 4 to 9)
  - EPS parameters
  - Tool data OEM
  - Magazine data OEM
  - Global user data (GUD)

Reinitialization time

During redefinition, the point in time can be specified at which the variable should be reinitialized, i.e. reset to the initialization value:

- `INIPO` (POWER ON)
  The variable is reinitialized at Power On.
- `INIRE` (reset)
  The variable is reinitialized on NC reset, mode group reset, at the end of the part program (M02/M30) or at Power On.

- `INICF` (NEWCONF)
  For the function "Set machine data active", the variable is reinitialized via HMI, part program command NEWCONF or NC reset, mode group reset, part program end (M02 / M30) or a Power On.
- `PRLOC` (program-local change)
  The PRLOC attribute may only be changed in conjunction with programmable setting data (see the table below).
  A programmable setting data is only reinitialized at NC reset, mode group reset or part program end (M02 / M30) if its value was changed in the current part program by programming the corresponding NC language command (see "NC language command" table column). If the value of the setting data is changed by programming the identifier (see "Identifier" table column), the setting date is not reinitialized at NC reset, mode group reset or part program end (M02 / M30).

Programmable setting data		
Number	**Identifier**	**NC language command** [1)]
42000	$SC_THREAD_START_ANGLE	SF
42010	$SC_THREAD_RAMP_DISP	DITS/DITE
42400	$SA_PUNCH_DWELLTIME	PDELAYON
42800	$SA_SPIND_ASSIGN_TAB	SETMS
43210	$SA_SPIND_MIN_VELO_G25	G25
43220	$SA_SPIND_MAX_VELO_G26	G26
43230	$SA_SPIND_MAX_VELO_LIMS	LIMS
43300	$SA_ASSIGN_FEED_PER_REV_SOURCE	FPRAON
43420	$SA_WORKAREA_LIMIT_PLUS	G26
43430	$SA_WORKAREA_LIMIT_MINUS	G25
43510	$SA_FIXED_STOP_TORQUE	FXST
43520	$SA_FIXED_STOP_WINDOW	FXSW
43700	$SA_OSCILL_REVERSE_POS1	OSP1
43710	$SA_OSCILL_REVERSE_POS2	OSP2
43720	$SA_OSCILL_DWELL_TIME1	OST1
43730	$SA_OSCILL_DWELL_TIME2	OST2
43740	$SA_OSCILL_VELO	FA
43750	$SA_OSCILL_NUM_SPARK_CYCLES	OSNSC
43760	$SA_OSCILL_END_POS	OSE
43770	$SA_OSCILL_CTRL_MASK	OSCTRL
43780	$SA_OSCILL_IS_ACTIVE	OS
43790	$SA_OSCILL_START_POS	OSB
[1)] This NC language command or G command addresses the setting data		

Constraints

Initialization value: Global user data (GUD)

- Only INIPO (Power On) can be defined as the initialization time for global user data (GUD) with the NC range of validity.
- In addition to INIPO (Power On), INIRE (reset) or INICF (NEWCONF) can be defined as the initialization time for global user data (GUD) with the CHAN range of validity.
- In the case of global user variables (GUD) with the CHAN range of validity and INIRE (reset) or INICF (NEWCONF) initialization time, for an NC reset, mode group reset and "Activate machine data", the variables are only reinitialized in the channels in which the named events were triggered.

Initialization value: FRAME data type

It is not permitted to specify an initialization value for variables of the FRAME data type. Variables of the FRAME data type are initialized implicitly and always with the default frame.

Initialization value: CHAR data type

For variables of the CHAR data type, instead of the ASCII code (0...255), the corresponding ASCII character can be programmed in quotation marks, e.g. "A"

Initialization value: Data type STRING

In the case of variables of the STRING data type, the character string must be enclosed in quotation marks, e.g. ...= "MACHINE_1"

Initialization value: AXIS data type

In the case of variables of the AXIS data type, for an extended address notation, the axis identifier must be enclosed in brackets, e.g. ...=(X3)

Initialization value: System variable

For system variables, redefinition cannot be used to define user-specific initialization values. The initialization values for the system variables are specified by the system and cannot be changed. However, redefinition can be used to change the point in time (INIRE, INICF) at which the system variable is reinitialized.

Implicit initialization value: AXIS data type

For variables of the AXIS data type the following implicit initialization value is used:

- System data: "First geometry axis"
- Synchronized action GUD (designation: SYG_A*), PUD, LUD:
  axis designation from the machine data: MD20082
  $MC_AXCONF_CHANAX_DEFAULT_NAME

Implicit initialization value: Tool and magazine data

Initialization values for tool and magazine data can be defined using the following machine data: MD17520 $MN_TOOL_DEFAULT_DATA_MASK

Note

Synchronization

The synchronization of events triggering the reinitialization of a global variable when this variable is read in a different location is the sole responsibility of the user / machine manufacturer.

See also

Variables (Page 377)

4.1.1.7 Attribute: Limit values (LLI, ULI)

An upper and a lower limit of the definition range can only be defined for the following data types:

- INT
- REAL
- CHAR

Definition (DEF) of user variables: Limit values and implicit initialization values

If no explicit initialization value is defined when defining a user variable of one of the above data types, the variable is set to the data type's implicit initialization value.

- INT: 0
- REAL: 0.0
- CHAR: 0

If the implicit initialization value is outside the definition range specified by the programmed limit values, the variable is initialized with the limit value which is closest to the implicit initialization value:

- Implicit initialization value < lower limit value (LLI) ⇒
  initialization value = lower limit value
- Implicit initialization value > upper limit value (ULI) ⇒
  initialization value = upper limit value

Examples:

Program code	Comment
DEF REAL GUD1	; Lower limit value = definition range limit ; Upper limit value = definition range limit ; No initialization value programmed ; => Implicit initialization value = 0.0
DEF REAL LLI 5.0 GUD2	; Lower limit value = 5.0 ; Upper limit value = definition range limit ; => Initialization value = 5.0
DEF REAL ULI -5 GUD3	; Lower limit value = definition range limit ; Upper limit value = -5.0 ; => Initialization value = -5.0

Redefinition (REDEF) of user variables: Limit values and current actual values

If the limit values of a user variable are redefined, they change to the extent that the current actual value is outside the new definition range, an alarm will be issued and the limit values will be rejected.

Note

Redefinition (REDEF) of user variables

If the limit values of a user variable are redefined, care must be taken to ensure that the following values are changed consistently:

- Limit values
- Actual value
- Initialization value on redefinition and automatic reinitialization on the basis of INIPO, INIRE or INICF

See also

Variables (Page 377)

4.1.1.8 Attribute: Physical unit (PHU)

A physical unit can only be specified for variables of the following data types:

- INT
- REAL

Programmable physical units (PHU)

The physical unit is specified as fixed point number: `PHU <unit>`

The following physical units can be programmed:

<unit>	Meaning	Physical unit
0	Not a physical unit	-
1	Linear or angular position 1)2)	[mm], [inch], [degree]
2	Linear position 2)	[mm], [inch]
3	Angular position	[degree]
4	Linear or angular velocity 1)2)	[mm/min], [inch/min], [rpm]
5	Linear velocity 2)	[mm/min]
6	Angular velocity	[rpm]
7	Linear or angular acceleration 1)2)	[m/s^2], [$inch/s^2$], [rev/s^2]
8	Linear acceleration 2)	[m/s^2], [$inch/s^2$]
9	Angular acceleration	[rev/s^2]
10	Linear or angular jerk 1)2)	[m/s^3], [$inch/s^3$], [rev/s^3]
11	Linear jerk 2)	[m/s^3], [$inch/s^3$]
12	Angular jerk	[rev/s^3]
13	Time	[s]
14	Position controller gain	[16.667/s]
15	Revolutional feedrate 2)	[mm/rev], [inch/rev]
16	Temperature compensation 1)2)	[mm], [inch]
18	Force	[N]
19	Mass	[kg]
20	Moment of inertia 3)	[kgm^2]
21	Percent	[%]
22	Frequency	[Hz]
23	Voltage	[V]
24	Current	[A]
25	Temperature	[°C]
26	Angle	[degree]
27	KV	[1000/min]
28	Linear or angular position 3)	[mm], [inch], [degree]
29	Cutting rate 2)	[m/min], [feet/min]
30	Peripheral speed 2)	[m/s], [feet/s]
31	Resistance	[ohm]
32	Inductance	[mH]
33	Torque 3)	[Nm]
34	Torque constant 3)	[Nm/A]
35	Current controller gain	[V/A]
36	Speed controller gain 3)	[Nm/(rad*s)]
37	Speed	[rpm]
42	Power	[kW]
43	Current, low	[μA]
46	Torque, low 3)	[μNm]
48	Per mil	-

<unit>	Meaning	Physical unit
49	-	[Hz/s]
65	Flow rate	[l/min]
66	Pressure	[bar]
67	Volume 3)	[cm^3]
68	Controlled-system gain 3)	[mm/(V*min)]
69	Force controller controlled-system gain	[N/V]
155	Thread lead 3)	[mm/rev], [inch/rev]
156	Change in thread lead 3)	[mm/rev / rev], [inch/rev / rev]
1) The physical unit depends on the axis type: Linear or rotary axis		
2) System of units changeover G70/G71(inch/metric) After changing over the basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) with G70/G71, for read/write operations to system and user variables involving a length, then the values are **not** converted (actual value, default value and limit values) G700/G710(inch/metric) After changing over the basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) with G700/G710, for read/write operations to system and user variables involving a length, then the values **are** converted (actual value, default value and limit values)		
3) The variable is **not** converted to the NC's current measuring system (inch/metric) automatically. Conversion is the sole responsibility of the user/machine manufacturer.		

Note

Level overflow due to format conversion

The internal storage format for all user variables (GUD/PUD/LUD) with physical units of length is metric. Excessive use of these types of variable in the NCK's main run, e.g. in synchronized actions, can lead to a CPU time overflow at interpolation level when the measuring system is switched over, generating alarm 4240.

Note

Compatibility of units

When using variables (assignment, comparison, calculation, etc.) the compatibility of the units involved is not checked. Should conversion be required, this is the sole responsibility of the user / machine manufacturer.

See also

Variables (Page 377)

4.1.1.9 Attribute: Access rights (APR, APW, APRP, APWP, APRB, APWB)

Designation

The designation of the access attribute AP... comprises:

1. A: **A**ccess
2. P: **P**rotection
3. R / W: **R**ead / **W**rite
4. P / O: **P**rogram / **B**TSS (OPI)

Access rights / access levels

The following access levels, which have to be specified during programming, correspond to the access rights:

Access right	Protection level
System password	0
Machine manufacturer password	1
Service password	2
User password	3
Key-operated switch position 3	4
Key-operated switch position 2	5
Key-operated switch position 1	6
Key-operated switch position 0	7

Definition (DEF) of user data

Access rights (APR.../APW...) can be defined for the following data:

- Global user data (GUD)

Redefinition (REDEF) of system and user data

Access rights (APR.../APW...) can be redefined for the following data:

- System data
  - Machine data

    Note

    Redefinition of reading rights of machine data

    The protection level for reading machine data can only be set with keyword APR for the part program and OPI.

    The keywords APRP and APRB are not supported by the redefinition of the reading rights, and result in alarm 12490 "Access right APRP/APRB <protection level> was not set".

  - Setting data
  - System variable
  - Process data
  - Magazine data
  - Tool data
- User data
  - R parameters
  - Synchronized action variables ($AC_MARKER, $AC_PARAM, $AC_TIMER)
  - Synchronized action GUD (SYG_xy[ ], where x=R, I, B, A, C, S and y=S, M, U, 4 to 9)
  - EPS parameters
  - Tool data OEM
  - Magazine data OEM
  - Global user variables (GUD)

    Note

    During redefinition the access right can be freely assigned to a variable between the lowest protection level 7 and the dedicated protection level, e.g. 1 (machine manufacturer).

Redefinition (REDEF) of NC language commands

The access or execution right (APX) can be redefined for the following NC language commands:

- G commands / preparatory functions (Page 178)
- Predefined functions (Page 1214)
- Predefined subprogram calls
- DO operation with synchronized actions
- Cycles program identifier
  The cycle must be saved in a cycle directory and must contain a PROC operation.

Access rights in relation to NC programs and cycles (APRP, APWP)

The various access rights facilitate the following with regard to access from an NC program or cycle:

- APRP 0/APWP 0
  - During NC program processing the system password has to be set.
  - The cycle has to be stored in the _N_CST_DIR directory (system).
  - The execution right must be set to system for the _N_CST_DIR directory in MD11160 $MN_ACCESS_EXEC_CST.
- APRP 1/APWP 1 or APRP 2/APWP 2
  - During NC program processing the machine manufacturer or service password has to be set.
  - The cycle has to be stored in the _N_CMA_DIR (machine manufacturer) or _N_CST_DIR directory.
  - The execution rights must be set to at least machine manufacturer for the _N_CMA_DIR or _N_CST_DIR directories in machine data MD11161 $MN_ACCESS_EXEC_CMA or MD11160 $MN_ACCESS_EXEC_CST respectively.
- APRP 3/APWP 3
  - During NC program execution, the user password must be set.
  - The cycle has to be stored in the _N_CUS_DIR (user), _N_CMA_DIR or _N_CST_DIR directory.
  - The execution rights must be set to at least user for the _N_CUS_DIR, _N_CMA_DIR or _N_CST_DIR directories in machine data MD11162 $MN_ACCESS_EXEC_CUS, MD11161 $MN_ACCESS_EXEC_CMA or MD11160 $MN_ACCESS_EXEC_CST respectively.
- APRP 4...7 / APWP 4...7
  - During NC program processing the key-operated switch must be set to 3 ... 0.
  - The cycle has to be stored in directory _N_CUS_DIR, _N_CMA_DIR or in directory _N_CST_DIR.
  - The execution rights must be set to at least the corresponding key-operated switch position for the _N_CUS_DIR, _N_CMA_DIR or _N_CST_DIR directories in machine data MD11162 $MN_ACCESS_EXEC_CUS, MD11161 $MN_ACCESS_EXEC_CMA or MD11160 $MN_ACCESS_EXEC_CST respectively.

Access rights in relation to OPI (APRB, APWB)

The access rights (APRB, APWB) restrict access to system and user variables via the OPI equally for all system components (HMI, PLC, external computers, EPS services, etc.).

Note

Local HMI access rights

When changing access rights to system data, care must be taken to ensure that such changes are consistent with the access rights defined using HMI mechanisms.

APR/APW access attributes

For compatibility reasons, attributes APR and APW are implicitly mapped to the attributes APRP / APRB and APWP / APWB:

- APR x ⇒ APRP x APRB x
- APW y ⇒ APWP y APWB y

Access rights using ACCESS files

When using ACCESS files to assign access rights, access rights for system data, user data, and NC language commands must only be redefined in ACCESS files. Global user data (GUD) is an exception. For this data, access rights still have to be redefined in the corresponding definition files *_DEF.

For continuous access protection, the machine data for the execution rights and the access protection for the corresponding directories have to be modified consistently.

In principle, the procedure is as follows:

1. Creation of the necessary definition files:
   - _N_DEF_DIR/_N_SACCESS_DEF
   - _N_DEF_DIR/_N_MACCESS_DEF
   - _N_DEF_DIR/_N_UACCESS_DEF
2. Setting of the write right for the definition files to the value required for redefinition:
   - MD11170 $MN_ACCESS_WRITE_SACCESS = <protection level>
   - MD11171 $MN_ACCESS_WRITE_MACCESS = <protection level>
   - MD11172 $MN_ACCESS_WRITE_UACCESS = <protection level>
3. For access to protected elements from cycles, the execution and write rights for cycle directories _N_CST_DIR, _N_CMA_DIR, and _N_CST_DIR have to be modified.
   Execution rights
   - MD11160 $MN_ACCESS_EXEC_CST = <protection level>
   - MD11161 $MN_ACCESS_EXEC_CMA = <protection level>
   - MD11162 $MN_ACCESS_EXEC_CUS = <protection level>

   Write rights
   - MD11165 $MN_ACCESS_WRITE_CST = <protection level>
   - MD11166 $MN_ACCESS_WRITE_CMA = <protection level>
   - MD11167 MN_ACCESS_WRITE_CUS = <protection level>

   The execution right has to be set to at least the same protection level as the highest protection level of the element used.
   The write right must be set to at least the same protection level as the execution right.
4. The write rights of the local HMI cycle directories must be set to the same protection level as the local NC cycle directories.

Subprogram calls in ACCESS files

To structure access protection further, subprograms (SPF or MPF identifier) can be called in ACCESS files. The subprograms inherit the execution rights of the calling ACCESS file.

Note

Only access rights can be redefined in the ACCESS files. All other attributes have to continue to be programmed/redefined in the corresponding definition files.

See also

Variables (Page 377)

4.1.1.10 Attribute: Data class (DCM, DCI, DCU)

To simplify the data handling during the commissioning, series start-up and upgrade of machines and machine series, all system and user data of the NC is divided into data classes.

Data class	Data
S = System	System data provided by Siemens, such as machine and setting data, standard and measuring cycles, definitions (SGUD) and macros (SMAC), etc.
M = Manufacturer (machine manufacturer)	Machine series-specific commissioning data such as manufacturer cycles, definitions (MGUD) and macros (MMAC) and machine data that defines the functional scope of the machine.
I = Individual (machine-specific)	Machine-specific commissioning data such as compensation data reference point offsets.
U = User	Machine-specific data generated during operation of the machine such as tool data, setting data, part programs, user cycles, definitions (UGUD) and macros (UMAC).

References:
SINUMERIK 828D Commissioning Manual, Turning and Milling; Section "Introduction and use of data classes"

Definition (DEF) of user data

The data class of the data item is implicitly specified through the data class of the file or directory in which the user data is defined. The data class of the data item cannot be changed.

However, for the definition (`DEF`) of the user data, a different data class to that of the data item can be specified for the **data value**.

The following must apply for the data class of the data item:

Priority of the data class of the data value ≤ priority of the data class of the data item

Example:

The definition of the GUD, which defines a probe, should be in data class M (= Manufacturer) because it is required to run the manufacturer cycles. However, the value of the data item should belong to data class I (= Individual) because the probe type can differ from machine to machine.

MGUD.DEF (data class M)

```
...
DEF CHAN DCI INT CALIPER
...
```

Redefinition (REDEF) of system data

The data class of the system data can be changed through redefinition (REDEF). The redefinition must be performed in a definition file with data class S or M.

When using ACCESS files, the redefinitions may only be performed with the ACCESS files.

The respective data class of the machine, setting and option data as well as the system variables can be found in the

- List Manual, Detailed Machine Data Description, parameter: "Class"
- List Manual, System Variables

4.1.1.11 Overview of definable and redefinable attributes

The following tables show which attributes can be defined (DEF) and/or redefined (REDEF) for which data types.

System data

Data type	Init. value	Limit values	Physical unit	Access rights	Data class (only 828D)
Machine data	---	---	---	REDEF	REDEF
Setting data	REDEF	---	---	REDEF	---
FRAME data	---	---	---	REDEF	---
Process data	---	---	---	REDEF	---
Leadscrew error comp. (EEC)	---	---	---	REDEF	---
Sag compensation (CEC)	---	---	---	REDEF	---
Quadrant error compensation (QEC)	---	---	---	REDEF	---
Magazine data	---	---	---	REDEF	---
Tool data	---	---	---	REDEF	---
Protection areas	---	---	---	REDEF	---
Toolholder, with orientation capability	---	---	---	REDEF	---
Kinematic chains	---	---	---	REDEF	---
3D protection areas	---	---	---	REDEF	---
Working area limitation	---	---	---	REDEF	---

User data

Data type	Init. value	Limit values	Physical unit	Access rights	Data class
R-parameters	REDEF	REDEF	REDEF	REDEF	---
Synchronized action variable ($AC_...)	REDEF	REDEF	REDEF	REDEF	---
Synchronized action GUD (SYG_...)	REDEF	REDEF	REDEF	REDEF	---
EPS parameters	REDEF	REDEF	REDEF	REDEF	---
Tool data OEM	REDEF	REDEF	REDEF	REDEF	---
Magazine data OEM	REDEF	REDEF	REDEF	REDEF	---
Global user variables (GUD)	DEF/REDEF	DEF	DEF	DEF/REDEF	DEF/REDEF
Local user variables (PUD/LUD)	DEF	DEF	DEF	---	---

See also

Variables (Page 377)

4.1.1.12 Definition and initialization of array variables (DEF, SET, REP)

A user variable can be defined as a 1- up to a maximum of a 3-dimensional array.

- 1-dimensional: `DEF <Data type> <Variable name>[<n>]`
- 2-dimensional: `DEF <Data type> <Variable name>[<n>,<m>]`
- 3-dimensional: `DEF <Data type> <Variable name>[<n>,<m>,<o>]`

Note

STRING data type user variables can be defined as up to a maximum of 2-dimensional arrays.

Data types

User variables can be defined as arrays for the following data types: BOOL, CHAR, INT, REAL, STRING, AXIS, FRAME

Assignment of values to array elements

Values can be assigned to array elements at the following points in time:

- During array definition (initialization values)
- During program execution

Values can be assigned by means of:

- Explicit specification of an array element
- Explicit specification of an array element as a starting element and specification of a value list (`SET`)
- Explicit specification of an array element as a starting element and specification of a value and the frequency at which it is repeated (`REP`)

Note

FRAME data type user variables cannot be assigned initialization values.

Syntax (DEF)

```
DEF <Data type> <Variable name>[<n>,<m>,<o>]
DEF  STRING[<String length>] <Variable name>[<n>,<m>]
```

Syntax (DEF...=SET...)

Using a value list:

- During definition:
  ```
  DEF <Data type> <Variable name>[<n>,<m>,<o>] =
  SET(<Value1>,<Value2>,...)
  ```
  Equivalent to:
  ```
  DEF <Data type> <Variable name>[<n>,<m>,<o>] =
  (<Value1>,<Value2>,...)
  ```

  Note

  `SET` does not have to be specified for initialization via a value list.

- During value assignment:
  ```
  <Variable name>[<n>,<m>,<o>] = SET(<VALUE1>,<VALUE2>,...)
  ```

Syntax (DEF...=REP...)

Using a value with repetition

- During definition:
  ```
  DEF <Data type> <Variable name>[<n>,<m>,<o>] = REP(<Value>)
  DEF <Data type> <Variable name>[<n>,<m>,<o>] =
  REP(<Value>,<Number_of_array_elements>)
  ```
- During value assignment:
  ```
  <Variable name>[<n>,<m>,<o>] = REP(<Value>)
  <Variable name>[<n>,<m>,<o>] =
  REP(<Value>,<Number_of_array_elements>)
  ```

Meaning

DEF:	Command to define variables
<Data type>:	Data type of variables Value range: • for system variables: BOOL, CHAR, INT, REAL, STRING, AXIS • for GUD or LUD variables: BOOL, CHAR, INT, REAL, STRING, AXIS, FRAME
<String length>:	Maximum number of characters for a STRING data type
<Variable name>:	Variable name
[<n>,<m>,<o>]:	Array sizes or array indices
<n>:	Array size or array index for 1st dimension Type: INT (for system variables, also AXIS) Value range: Max. array size: 65535 Array index: 0 ≤ n ≤ 65534
<m>:	Array size or array index for 2nd dimension Type: INT (for system variables, also AXIS) Value range: Max. array size: 65535 Array index: 0 ≤ m ≤ 65534
<o>:	Array size or array index for 3rd dimension Type: INT (for system variables, also AXIS) Value range: Max. array size: 65535 Array index: 0 ≤ o ≤ 65534
SET:	Value assignment using specified value list
(<value1>,<value2>, etc.):	Value list
REP:	Value assignment using specified <Value>
<Value>:	Value, which the array elements should be written when initializing with REP.
<Number_of_array_elements>:	Number of array elements to be written with the specified <Value>. The following apply to the remaining array elements, dependent on the point in time: • Initialization when defining the array: → Zero is written to the remaining array elements. • Assignment during program execution: → The actual values of the array elements remain unchanged. If the parameter is not programmed, all array elements are written with <Value>. If the parameter equals zero, the following apply dependent on the point in time: • Initialization when defining the array: → All elements are pre-assigned zero • Assignment during program execution: → The actual values of the array elements remain unchanged.

Array index

The implicit sequence of the array elements, e.g. in the case of value assignment using SET or REP, is right to left due to iteration of the array index.

Example: Initialization of a 3-dimensional array with 24 array elements:

```
DEF INT FELD[2,3,4] = REP(1,24)
  FELD[0,0,0] = 1       1st array element
  FELD[0,0,1] = 1       2nd array element
  FELD[0,0,2] = 1       3rd array element
  FELD[0,0,3] = 1       4th array element
  ...
  FELD[0,1,0] = 1       5th array element
  FELD[0,1,1] = 1       6th array element
  ...
  FELD[0,2,3] = 1       12th array element
  FELD[1,0,0] = 1       13th array element
  FELD[1,0,1] = 1       14th array element
  ...
  FELD[1,2,3] = 1       24th array element
```

corresponding to:

```
FOR n=0 TO 1
  FOR m=0 TO 2
     FOR o=0 TO 3
       FELD[n,m,o] = 1
     ENDFOR
  ENDFOR
ENDFOR
```

Example: Initializing complete variable arrays

For the actual assignment, refer to the diagram.

Program code
N10 DEF REAL FELD1[10,3]=SET(0,0,0,10,11,12,20,20,20,30,30,30,40,40,40,)
N20 FELD1[0,0]=REP(100)
N30 FELD1[5,0]=REP(-100)
N40 FELD1[0,0]=SET(0,1,2,-10,-11,-12,-20,-20,-20,-30, , , ,-40,-40,-50,-60,-70)
N50 FELD1[8,1]=SET(8.1,8.2,9.0,9.1,9.2)

Array index 2

1

[1.2]	N10: Initialization at definition		
	0	1	2
0	0	0	0
1	10	11	12
2	20	20	20
3	30	30	30
4	40	40	40
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
	The array elements [5.0] to [9.2] were initialized with the default value (0.0).		

N20/N30: Initialization with identical value		
0	1	2
100	100	100
100	100	100
100	100	100
100	100	100
100	100	100
-100	-100	-100
-100	-100	-100
-100	-100	-100
-100	-100	-100
-100	-100	-100

N40/N50: Initialization with different values		
0	1	2
0	1	2
-10	-11	-12
-20	-20	-20
-30	0	0
0	-40	-40
-50	-60	-70
-100	-100	-100
-100	-100	-100
-100	8.1	8.2
9.0	9.1	9.2
The array elements [3.1] to [4.0] were initialized with the default value (0.0). The array elements [6.0] to [8.0] were not changed.		

See also

Definition and initialization of array variables (DEF, SET, REP) (Page 405)

Variables (Page 377)

Additional information (SET)

initialization for the definition

- Starting with the 1st array element, as many array elements are assigned with the values from the value list as there are elements programmed in the value list.
- A value of 0 is assigned to array elements without explicitly declared values in the value list (gaps in the value list).
- For variables of the AXIS data type, gaps in the value list are not permitted.
- An alarm is displayed if the value list contains more values than there are array elements defined.

Value assignment in program execution

In the case of value assignment in program execution, the rules described above for the definition apply. The following options are also supported:

- Expressions are also permitted as elements in the value list.
- Value assignment starts with the programmed array index. Values can be assigned selectively to subarrays.

Example:

Program code	Comment
DEF INT ARRAY[5,5]	; Array definition
ARRAY[0,0]=SET(1,2,3,4,5)	; Value assignment to the first 5 array elements [0,0] - [0,4]
ARRAY[0,0]=SET(1,2, , ,5)	; Value assignment with gap to the first 5 array elements [0,0] - [0,4], array elements[0,2] and [0,3] = 0
ARRAY[2,3]=SET(VARIABLE,4*5.6)	; Value assignment with variable and expression starting at array index [2,3]: [2,3] = VARIABLE [2,4] = 4 * 5.6 = 22.4

Further information (REP)

initialization for the definition

- All or the optionally specified number of array elements are initialized with the specified value (constant).
- Variables of the FRAME data type cannot be initialized.

Example:

Program code	Comment
DEF REAL varName[10]=REP(3.5,4)	; Initialize array definition and array elements [0] to [3] with value 3.5.

Value assignment in program execution

In the case of value assignment in program execution, the rules described above for the definition apply. The following options are also supported:

- Expressions are also permitted as elements in the value list.
- Value assignment starts with the programmed array index. Values can be assigned selectively to subarrays.

Examples:

Program code	Comment
DEF REAL varName[10]	; Array definition
varName[5]=REP(4.5,3)	; Array elements [5] to [7] = 4.5
R10=REP(2.4,3)	; R-parameters R10 to R12 = 2.4
DEF FRAME FRM[10]	; Array definition

Program code	Comment
`FRM[5]=REP(CTRANS(X,5))`	`; Array elements [5] to [9] = CTRANS(X,5)`

4.1.1.13 Data types

The following data types are available in the NC:

Data type	Meaning	Value range
INT	Integer with sign	-2147483648 ... +2147483647
REAL	Real number (LONG REAL to IEEE)	$\pm(\sim 2.2*10^{-308} \ldots \sim 1.8*10^{+308})$
BOOL	Truth value TRUE (1) and FALSE (0)	1, 0
CHAR	ASCII character	ASCII code 0 to 255
STRING	Character string of a defined length	Maximum of 400 characters (no special characters)
AXIS	Axis/spindle identifier	Channel axis identifier
FRAME	Geometric parameters for static coordinate transformation (translation, rotation, scaling, mirroring)	---

Implicit data type conversions

The following data type conversions are possible and are performed implicitly during assignments and parameter transfers:

from ↓/ to →	REAL	INT	BOOL
REAL	x	o	&
INT	x	x	&
BOOL	x	x	x
x : Possible without restrictions o: Data loss possible due to the range of values being overshot ⇒ alarm; rounding: decimal place value ≥ 0.5 ⇒ round up, decimal place value < 0.5 ⇒ round down &: value ≠ 0 ⇒ TRUE, value== 0 ⇒ FALSE			

See also

Variables (Page 377)

4.1.1.14 Variable minimum, maximum and range (MINVAL, MAXVAL and BOUND)

The `MINVAL` and `MAXVAL` commands compare the values of two variables. The smaller value (in the case of `MINVAL`) or the larger value (in the case of `MAXVAL`) respectively is delivered as a result.

The `BOUND` command tests whether the value of a test variable falls within a defined range of values.

Syntax

```
<smaller value>=MINVAL(<variable1>,<variable2>)
<larger value>=MAXVAL(<variable1>,<variable2>)
<return value>=<BOUND>(<minimum>,<maximum>,<test variable>)
```

Meaning

`MINVAL`:	Obtains the **smaller** value of two variables (`<variable1>`, `<variable2>`)
`<smaller value>`:	Result variable for the `MINVAL` command Set to the smaller variable value.
`MAXVAL`:	Obtains the **larger** value of two variables (`<variable1>`, `<variable2>`)
`<larger value>`:	Result variable for the `MAXVAL` command Set to the larger variable value.
`BOUND`:	Tests whether a variable (`<test variable`) is within a defined range of values.
`<minimum>`:	Variable which defines the minimum value of the range of values.
`<maximum>`:	Variable which defines the maximum value of the range of values.
`<return value>`:	Result variable for the `BOUND` command If the value of the test variable is within the defined range of values, the result variable is set to the value of the test variable. If the value of the test variable is greater than the maximum value, the result variable is set to the maximum value of the definition range. If the value of the test variable is less than the minimum value, the result variable is set to the minimum value of the definition range.

Note

`MINVAL`, `MAXVAL`, and `BOUND` can also be programmed in synchronized actions.

Note

Behavior if values are equal

If the values are equal, `MINVAL/MAXVAL` are set to this equal value. In the case of `BOUND` the value of the variable to be tested is returned again.

Example

Program code	Comment
DEF REAL rVar1=10.5, rVar2=33.7, rVar3, rVar4, rVar5, rValMin, rValMax, rRetVar	
rValMin=MINVAL(rVar1,rVar2)	; rValMin is set to value 10.5.
rValMax=MAXVAL(rVar1,rVar2)	; rValMax is set to value 33.7.
rVar3=19.7	
rRetVar=BOUND(rVar1,rVar2,rVar3)	; rVar3 is within the limits, rRetVar is set to 19.7.
rVar3=1.8	

Program code	Comment
`rRetVar=BOUND(rVar1,rVar2,rVar3)`	`; rVar3 is below the minimum limit, rRetVar is set to 10.5.`
`rVar3=45.2`	
`rRetVar=BOUND(rVar1,rVar2,rVar3)`	`; rVar3 is above the maximum limit, rRetVar is set to 33.7.`

4.1.1.15 Check availability of a variable (ISVAR)

The predefined ISVAR function can be used to check whether a system/user variable (e.g. machine data, setting data, system variable, general variables such as GUD) is known in the NC.

Variable

The variables to be queried have the following structure:

Dimensionless variable:	<Variable>
One-dimensional variable without array index:	<Variable>[ ]
One-dimensional variable with array index n:	<Variable>[<n>]
Two-dimensional variable without array index:	<Variable>[,]
Two-dimensional variable with array indices n and m:	<Variable>[<n>,<m>]

Syntax

```
<Result>=ISVAR(<Variable>[<n>,<m>])
```

Meaning

`<result>`:	Return value		
	Data type:	BOOL	
	Range of values:	1	Variable available
		0	Variable unknown
`ISVAR`:	Checks whether the specified system/user variable is known in the NC.		
`<Variable>`:	Name of the system/user variable		
	Data type:	STRING	
<n>:	Array index of the first dimension (**optional**)		
	Data type:	INT	
<m>:	Array index of the second dimension (**optional**)		
	Data type:	INT	

The following checks are made in accordance with the transfer parameter:

- Is the name known?
- Is the variable an array?
- Is it a one- or two-dimensional array?
- Is the respective array index in the permissible range?

Only if all checks are positive, TRUE (1) is returned.

If a check is negative or a syntax error has occurred, FALSE (0) is returned.

Examples

Program code	Comment
DEF INT VAR1	
DEF BOOL IS_VAR=FALSE	
N10 IS_VAR=ISVAR("VAR1")	; IS_VAR is in this case TRUE.

Program code	Comment
DEF REAL VARARRAY[10,10]	
DEF BOOL IS_VAR=FALSE	
N10 IS_VAR=ISVAR("VARARRAY[,]")	; IS_VAR is in this case TRUE, is a two-dimensional array.
N20 IS_VAR=ISVAR("VARARRAY")	; IS_VAR is TRUE, variable exists.
N30 IS_VAR=ISVAR("VARARRAY[8,11]")	; IS_VAR is FALSE, array index is not permitted.
N40 IS_VAR=ISVAR("VARARRAY[8,8")	; IS_VAR is FALSE, "]" missing (syntax error).
N50 IS_VAR=ISVAR("VARARRAY[,8]")	; IS_VAR is TRUE, array index is permitted.
N60 IS_VAR=ISVAR("VARARRAY[8,]")	; IS_VAR is TRUE, array index is permitted.

Program code	Comment
DEF BOOL IS_VAR=FALSE	
N100 IS_VAR=ISVAR("$MC_GCODE_RESET_VALUES[1]"	; Transfer parameter is a machine data item, IS_VAR is TRUE.

Program code	Comment
DEF BOOL IS_VAR=FALSE	
N10 IS_VAR=ISVAR("$P_EP")	; IS_VAR is in this case TRUE.
N20 IS_VAR=ISVAR("$P_EP[X]")	; IS_VAR is in this case TRUE.

4.1.1.16 Reading attribute values / data type (GETVARPHU, GETVARAP, GETVARLIM, GETVARDIM, GETVARDFT, GETVARTYP)

The attribute values of system/user variables can be read with the predefined GETVARPHU, GETVARAP, GETVARLIM, GETVARDIM and GETVARDFT functions, the data type of a system/user variable with GETVARTYP.

Read physical unit

Syntax:

```
<Result>=GETVARPHU(<name>)
```

Meaning:

<result>:	Numeric value of the physical unit		
	Data type:	INT	
	Range of values:	See Table in "Attribute: Physical unit (PHU) (Page 396)"	
		In case of fault	
		- 2	The specified variable name has not been assigned to a system parameter or a user variable.
GETVARPHU:	Reading of the physical unit of a system/user variable		
<name>:	Name of the system/user variables		
	Data type:	STRING	

Example:

The NC contains the following GUD variables:

DEF CHAN REAL PHU 42 LLI 0 ULI 10000 electric

Program code	Comment
`DEF INT result=0`	
`result=GETVARPHU("electric")`	`; Determine the physical unit of the GUD variables.`
`IF (result < 0) GOTOF error`	

The value 42 is returned as result. This corresponds to the physical unit [kW].

Note

GETVARPHU can be used, for example, to check whether both variables have the expected physical units in a variable assignment a = b.

Read access right

Syntax:
`<Result>=GETVARAP(<name>,<access>)`

Meaning:

<result>:	Protection level for the specified <access>		
	Data type:	INT	
	Range of values:	0 ... 7	See "Attribute: Access rights (APR, APW, APRP, APWP, APRB, APWB) (Page 399)".
		In case of fault	
		- 1	Cannot be written (only relevant for access types "WP" and "WB")
		- 2	The specified variable name has not been assigned to a system parameter or a user variable.
		- 3	Incorrect value for the parameter <access>

`GETVARAP`:	Reading of the access right to a system/user variable		
`<name>`:	Name of the system/user variables		
	Data type:	STRING	
`<access>`:	Type of access		
	Data type:	STRING	
	Range of values:	`"RP"`	Read via part program
		`"WP"`	Write via part program
		`"RB"`	Read via OPI
		`"WB"`	Write via OPI

Example:

Program code	Comment
`DEF INT result=0`	
`result=GETVARAP("$TC_MAP8","WB")`	`; Determine the access protection for the system parameter "magazine position" with regard to writing via OPI.`
`IF (result < 0) GOTOF error`	

The value 7 is returned as result. This corresponds to the key switch position 0 (= no access protection).

Note

GETVARAP can be used, for example, to implement a checking program that checks the access rights expected by the application.

Read limit values

Syntax:

```
<Status>=GETVARLIM(<name>,<limit value>,<result>)
```

Meaning:

`<Status>`:	Function status		
	Data type:	INT	
	Range of values:	1	OK
		-1	No limit value defined (for variables of type AXIS, STRING, FRAME)
		-2	The specified variable name has not been assigned to a system parameter or a user variable.
		-3	Incorrect value for the parameter <limit value>
`GETVARLIM`:	Reading of the lower/upper limit value of a system/user variable		
`<name>`:	Name of the system/user variables		
	Data type:	STRING	

<table>
<tr><td rowspan="4"><limit value>:</td><td colspan="3">specifies which limit value should be read out</td></tr>
<tr><td>Data type:</td><td colspan="2">CHAR</td></tr>
<tr><td rowspan="2">Range of values:</td><td>"L"</td><td>Lower limit value</td></tr>
<tr><td>"U":</td><td>Upper limit value</td></tr>
<tr><td rowspan="2"><result>:</td><td colspan="3">Return of the limit value</td></tr>
<tr><td>Data type:</td><td colspan="2">VAR REAL</td></tr>
</table>

Example:

Program code	Comment
`DEF INT state=0`	
`DEF REAL result=0`	
`state=GETVARLIM("$MA_MAX_AX_VELO","L",result)`	`; Determine the lower limit value for MD32000 $MA_MAX_AX_VELO.`
`IF (result < 0) GOTOF error`	

Read dimension value

Syntax:
`<Result>=GETVARDIM(<Name>,<Index>)`

Meaning:

<table>
<tr><td rowspan="2"><result>:</td><td colspan="3">Size of the dimension specified by the parameter <Index></td></tr>
<tr><td>Data type:</td><td colspan="2">INT</td></tr>
<tr><td>GETVARDIM:</td><td colspan="3">Reading the size of the first, second or third dimension of the field of a system/user variable</td></tr>
<tr><td rowspan="2"><name>:</td><td colspan="3">Name of the system/user variables</td></tr>
<tr><td>Data type:</td><td colspan="2">STRING</td></tr>
<tr><td rowspan="6"><Index>:</td><td colspan="3">Array index</td></tr>
<tr><td>Data type:</td><td colspan="2">INT</td></tr>
<tr><td rowspan="4">Range of values:</td><td colspan="2">1 ... 3</td></tr>
<tr><td>1</td><td>Index for first Dimension of the field</td></tr>
<tr><td>2</td><td>Index for 2nd Dimension of the field</td></tr>
<tr><td>3</td><td>Index for third Dimension of the field</td></tr>
</table>

Example:

Program code	Comment
`N5 DEF REAL myReal[5,4]`	
`N10 R1=GETVARDIM("myReal",1)`	`; Determine the size of the first dimension of the field.`
	`; Result: R1 = 5`
`N15 R2=GETVARDIM("myReal",2)`	`; Determine the size of the second dimension of the field.`
	`; Result: R2 = 4`

Read default value

Syntax:

```
<Status>=GETVARDFT(<Name>,<Result>[,<Index_1>,<Index_2>, <Index_3>])
```

Meaning:

`<Status>`:	Function status		
	Data type:	INT	
	Range of values:	1	OK
		-1	No default value available (e.g. because the data type definition of the result variables does not match the data type of the system/user variables).
		-2	The specified variable name has not been assigned to a system parameter or a user variable.
		-3	Incorrect value for parameter <Index_1>, dimension smaller than one (⇒ no field but scalar variable)
		-4	Incorrect value for the parameter <Index_2>
		-5	Incorrect value for the parameter <Index_3>
`GETVARDFT`:	Reading of the default value of a system/user variable		
`<name>`:	Name of the system/user variables		
	Data type:	STRING	
`<result>`:	Return of the default value		
	Data type:	VAR REAL (when reading the default value of variables of the types INT, REAL, BOOL, AXIS)	
		VAR STRING (when reading the default value of variables of the types STRING and CHAR)	
		VAR FRAME (when reading the default value of variables of the type FRAME)	
`<index_1>`:	Index for first dimension of the field (optional)		
	Data type:	INT	
	Not programmed means = 0		
`<index_2>`:	Index for 2nd dimension of the field (optional)		
	Data type:	INT	
	Not programmed means = 0		
`<index_3>`:	Index for third dimension of the field (optional)		
	Data type:	INT	
	Not programmed means = 0		

Example:

Program code	Comment
`DEF INT state=0`	
`DEF REAL resultR=0`	`; Variable to accept the default values of the types INT, REAL, BOOL, AXIS.`
`DEF FRAME resultF=0`	`; Variable to accept the default values of the type FRAME`

Program code	Comment
`IF (GETVARTYP("$MA_MAX_AX_VELO") <> 4) GOTOF error`	
`state=GETVARDFT("$MA_MAX_AX_VELO",resultR, AXTOINT(X))`	`; Determine the default value of the "X" axis.`
`IF (result < 0) GOTOF error`	
`IF (GETVARTYP("$TC_TP8") <> 3) GOTOF error`	
`state=GETVARDFT("$TC_TP8", resultR)`	
`IF (GETVARTYP("$P_UBFR")  <> 7) GOTOF error`	
`state=GETVARDFT("$P_UBFR", resultF )`	

Read data type

Syntax:
```
<Result>=GETVARTYP(<name>)
```

Meaning:

`<result>:`	Data type of the specified system/user variables		
	Data type:	INT	
	Range of values:	1	= BOOL
		2	= CHAR
		3	= INT
		4	= REAL
		5	= STRING
		6	= AXIS
		7	= FRAME
		In case of fault	
		< 0	The specified variable name has not been assigned to a system parameter or a user variable.
`GETVARTYP:`	Reading of the data type of a system/user variable		
`<name>:`	Name of the system/user variables		
	Data type:	STRING	

Example:

Program code	Comment
`DEF INT result=0`	
`DEF STRING name="R"`	
`result=GETVARTYP(name)`	`; Determine the type of the R parameter.`
`IF (result < 0) GOTOF error`	

The value 4 is returned as result. This corresponds to the REAL data type.

4.1.1.17 Possible type conversions

The constant numeric value, the variable, or the expression assigned to a variable must be compatible with the variable type. If this is the case, the type is automatically converted when the value is assigned.

Possible type conversions

to	REAL	INT	BOOL	CHAR	STRING	AXIS	FRAME
from							
REAL	yes	Yes*	Yes 1)	Yes*	–	–	–
INT	yes	yes	Yes 1)	Yes 2)	–	–	–
BOOL	yes	yes	yes	yes	yes	–	–
CHAR	yes	yes	Yes 1)	yes	yes	–	–
STRING	–	–	Yes 4)	Yes 3)	yes	–	–
AXIS	–	–	–	–	–	yes	–
FRAME	–	–	–	–	–	–	yes

Explanation

* At type conversion from REAL to INT, fractional values that are >=0.5 are rounded up, others are rounded down (cf. ROUND function).

1) Value <> 0 is equivalent to TRUE; value == 0 is equivalent to FALSE

2) If the value is in the permissible range

3) If only 1 character

4) String length 0 = >FALSE, otherwise TRUE

Note

If conversion produces a value greater than the target range, an error message is output.

If mixed types occur in an expression, type conversion is automatic. Type conversions are also possible in synchronous actions, see Chapter "Motion-synchronous actions, implicit type conversion".

4.1.2 Indirect programming

4.1.2.1 Indirectly programming addresses

When indirectly programming addresses (Page 1165), the extended address (<Index>) is replaced by the suitable type of variable.

Note

It is not possible to indirectly program addresses for:

- N (block number)
- L (subprogram)
- Settable addresses
  (e.g. X[1] instead of X1 is not permissible)

Syntax

```
<ADDRESS>[<Index>]
```

Meaning

`<ADDRESS>[...]`:	Fixed address with extension (index)
`<index>`:	Variable, e.g. for spindle number, axis, ...

Examples

Example 1: Indirectly programming a spindle number

Direct programming:

Program code	Comment
`S1=300`	`; Speed 300 rpm for the spindle number 1.`

Indirect programming:

Program code	Comment
`DEF INT SPINU=1`	`; Defining variables, type INT and value assignment.`
`S[SPINU]=300`	`; Speed 300 rpm for the spindle, whose number is saved in the SPINU variable (in this example 1, the spindle with the number 1).`

Example 2: Indirectly programming an axis

Direct programming:

Program code	Comment
`FA[U]=300`	`; Feedrate 300 for axis "U".`

Indirect programming:

Program code	Comment
DEF AXIS AXVAR2=U	; Defining a variable, type AXIS and value assignment.
FA[AXVAR2]=300	; Feedrate of 300 for the axis whose address name is saved in the variables with the name AXVAR2.

Example 3: Indirectly programming an axis

Direct programming:

Program code	Comment
$AA_MM[X]	; Read probe measured value (MCS) of axis "X".

Indirect programming:

Program code	Comment
DEF AXIS AXVAR3=X	; Defining a variable, type AXIS and value assignment.
$AA_MM[AXVAR3]	; Read probe measured value (MCS) whose name is saved in the variables AXVAR3.

Example 4: Indirectly programming an axis

Direct programming:

Program code
X1=100 X2=200

Indirect programming:

Program code	Comment
DEF AXIS AXVAR1 AXVAR2	; Defining two type AXIS variables.
AXVAR1=(X1) AXVAR2=(X2)	; Assigning the axis names.
AX[AXVAR1]=100 AX[AXVAR2]=200	; Traversing the axes whose address names are saved in the variables with the names AXVAR1 and AXVAR2

Example 5: Indirectly programming an axis

Direct programming:

Program code
G2 X100 I20

Indirect programming:

Program code	Comment
DEF AXIS AXVAR1=X	; Defining a variable, type AXIS and value assignment.
G2 X100 IP[AXVAR1]=20	; Indirect programming the center point data for the axis, whose address name is saved in the variable with the name AXVAR1.

Example 6: Indirectly programming array elements

Direct programming:

Program code	Comment
DEF INT ARRAY1[4,5]	; Defining array 1

Indirect programming:

Program code	Comment
DEFINE DIM1 AS 4	; For array dimensions, array sizes must be specified as fixed values.
DEFINE DIM2 AS 5	
DEF INT ARRAY[DIM1,DIM2]	
ARRAY[DIM1-1,DIM2-1]=5	

Example 7: Indirect subprogram call

Program code	Comment
CALL "L" << R10	; Call the program, whose number is located in R10 (string cascading).

4.1.2.2 Indirectly programming G commands

Indirectly programming G commands (Page 1176) permits cycles to be effectively programmed.

Syntax

```
G[<group>]=<number>
```

Meaning

G[...]:	G command with extension (index)	
<group>:	Index parameter: G group	
	Type:	INT
<number>:	Variable for the G command number	
	Type:	INT or REAL

Note

Generally, only G commands that do not determine the syntax can be indirectly programmed.

Only G group 1 is possible from the G commands that determine the syntax.
The syntax-determining G commands of G groups 2, 3 and 4 are not possible.

Note

Arithmetic functions are not permitted in the indirect G command programming. If it is necessary to calculate the G command number, this must be done in a separate part program line before the indirect G command programming.

Examples

Example 1: Adjustable work offset (G group 8)

Program code	Comment
N1010 DEF INT INT_VAR	
N1020 INT_VAR=2	
...	
N1090 G[8]=INT_VAR G1 X0 Y0	;G54
N1100 INT_VAR=INT_VAR+1	; G command calculation
N1110 G[8]=INT_VAR G1 X0 Y0	;G55

Example 2: Level selection (G group 6)

Program code	Comment
N2010 R10=$P_GG[6]	; Read active G command of G group 6
...	
N2090 G[6]=R10	

4.1.2.3 Indirectly programming position attributes (GP)

Position attributes, e.g. the incremental or absolute programming of the axis position, can be indirectly programmed as variables in conjunction with the key word GP.

Application

Indirectly programming position attributes is used in **replacement cycles**, as in this case, the following advantage exists over programming position attributes as keyword (e.g. IC, AC, ...):

As a result of the indirect programming as variable, **no** CASE statement is required, which would otherwise branch over all possible position attributes.

Syntax

```
<POSITIONING COMMAND>[<axis/spindle>]=
GP(<position>,<position attribute)
<axis/spindle>=BP(<position>,<position attribute)
```

Meaning

`<POSITIONING COMMAND>[]:`	The following positioning commands can be programmed together with the key word GP: `POS`, `POSA`, `SPOS`, `SPOSA` Also possible: • All axis and spindle identifiers present in the channel: `<axis/spindle>` • Variable axis/spindle identifier `AX`
`<axis/spindle>:`	Axis/spindle that is to be positioned

`GP():`	Key word for positioning
`<position>:`	Parameter 1 Axis/spindle position as constant or variable
`<position attribute>:`	Parameter 2 Position attribute (e.g. position approach mode) as a variable (e.g. $P_SUB_SPOSMODE) or as key word (IC, AC, ...)

The values supplied from the variables have the following significance:

Value	Meaning	Permissible for:
0	No change to the position attribute	
1	AC	POS, POSA,SPOS, SPOSA,AX, axis address
2	IC	POS, POSA,SPOS, SPOSA,AX, axis address
3	DC	POS, POSA,SPOS, SPOSA,AX, axis address
4	ACP	POS, POSA,SPOS, SPOSA,AX, axis address
5	ACN	POS, POSA,SPOS, SPOSA,AX, axis address
6	OC	-
7	PC	-
8	DAC	POS, POSA,AX, axis address
9	DIC	POS, POSA,AX, axis address
10	RAC	POS, POSA,AX, axis address
11	RIC	POS, POSA,AX, axis address
12	CAC	POS, POSA
13	CIC	POS, POSA
14	CDC	POS, POSA
15	CACP	POS, POSA
16	CACN	POS, POSA

Example

For an active synchronous spindle coupling between the leading spindle S1 and the following spindle S2, the following replacement cycle to position the spindle is called using the `SPOS` command in the main program.

Positioning is realized using the statement in N2230:

```
SPOS[1]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
SPOS[2]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
```

The position to be approached is read from the system variable $P_SUB_SPOSIT; the position approach mode is read from the system variable $P_SUB_SPOSMODE.

Program code	Comment
`N1000 PROC LANG_SUB DISPLOF SBLOF`	
`...`	
`N2100 IF($P_SUB_AXFCT==2)`	
`N2110`	`; Replacement of the SPOS / SPOSA / M19 command for an active synchronous spindle coupling`

Program code	Comment
N2185 DELAYFSTON	; Start of stop delay area
N2190 COUPOF(S2,S1)	; Deactivate synchronous spindle coupling
N2200	; Position leading and following spindles
N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)	
N2220	; Positioning the spindle with SPOS:
N2230 SPOS[1]=GP($P_SUB_SPOSIT, $P_SUB_SPOSMODE)	
SPOS[2]=GP($P_SUB_SPOSIT, $P_SUB_SPOSMODE)	
N2250 ELSE	
N2260	; Positioning the spindle using M19:
N2270 M1=19 M2=19	; Position leading and following spindles
N2280 ENDIF	
N2285 DELAYFSTOF	; End of stop delay area
N2290 COUPON(S2,S1)	; Activate synchronous spindle coupling
N2410 ELSE	
N2420	; Query on further replacements
...	
N3300 ENDIF	
...	
N9999 RET	

Supplementary conditions

The indirect programming of position attributes is not possible in synchronized actions.

4.1.2.4 Indirectly programming part program lines (EXECSTRING)

Using the part program command EXECSTRING, it is possible to execute a previously generated string variable as part program line.

Syntax

EXECSTRING is programmed in a separate part program line:

```
EXECSTRING (<string_variable>)
```

Meaning

EXECSTRING:	Command to execute a string variable as part program line
<string variable>:	Type STRING variable, that includes the actual part program line to be executed

---

Note

With `EXECSTRING`, all part program constructions that can be programmed in the **program section of a part program**, with the exception of control structures (Page 464), can be extracted. This means that `PROC` and `DEF` statements are excluded as well as the general use in INI and DEF files.

Example

Program code	Comment
N100 DEF STRING[100] MY_BLOCK	; Definition of string variables to accept the part program line to be executed.
N110 DEF STRING[10] MFCT1="M7"	
...	
N200 EXECSTRING(MFCT1 << "M4711")	; Execute part program line "M7 M4711".
...	
N300 R10=1	
N310 MY_BLOCK="M3"	
N320 IF(R10)	
N330 MY_BLOCK = MY_BLOCK << MFCT1	
N340 ENDIF	
N350 EXECSTRING(MY_BLOCK)	; Execute part program line "M3 M7".

4.1.3 Instructions

4.1.3.1 Arithmetic functions

Operator / arithmetic function	Meaning
+	Addition
–	Subtraction
*	Multiplication
/ [1)]	Division [1)]
DIV [1)]	Integer number division [1)]
MOD [1)]	Modulo division (supplies the remainder of the integer number division) [1)]
:	Chain operator for FRAME variables
SIN()	Sine
COS()	Cosine
TAN()	Tangent
ASIN()	Arc sine

`ACOS()`	Arc cosine
`ATAN2(,)` 1)	Arc tangent2 1)
`SQRT()`	Square root
`ABS()`	Absolute value
`POT()`	Power function (Page 432)
`TRUNC()`	Precision correction for comparison errors (Page 435)
`ROUND()`	Round to an integer number
`ROUNDUP()`	Round up (Page 434)
`LN()`	Natural logarithm
`EXP()`	Exponential function
`MINVAL ()`	Lower value of two variables (Page 411)
`MAXVAL ()`	Higher value of two variables (Page 411)
`BOUND()`	Variable value within the defined value range (Page 411)
`CTRANS()`	Value assignments to frames: Offset (Page 617)
`CROT()`	Value assignments to frames: Rotation (Page 617)
`CSCALE()`	Value assignments to frames: Change of scale (Page 617)
`CMIRROR()`	Value assignments to frames: Mirroring (Page 617)
1) See the paragraph, "Examples"	

Programming

The usual mathematical notation is used for arithmetic functions. Priorities for execution are indicated by parentheses. Angles are specified for trigonometry functions and their inverse functions (right angle = 90°).

Examples

Division: /

(type REAL) = type INT or type REAL) / (type INT or type REAL);

Example: 3 / 4 = 0.75

Integer number division: DIV

(type INT) = (type INT or REAL) / (type INT or REAL);

Example: 7 DIV 4.1 = 1

Modulo division (supplies the remainder of the integer number division): MOD

(type REAL) = (type INT or REAL) MOD (type INT or REAL);

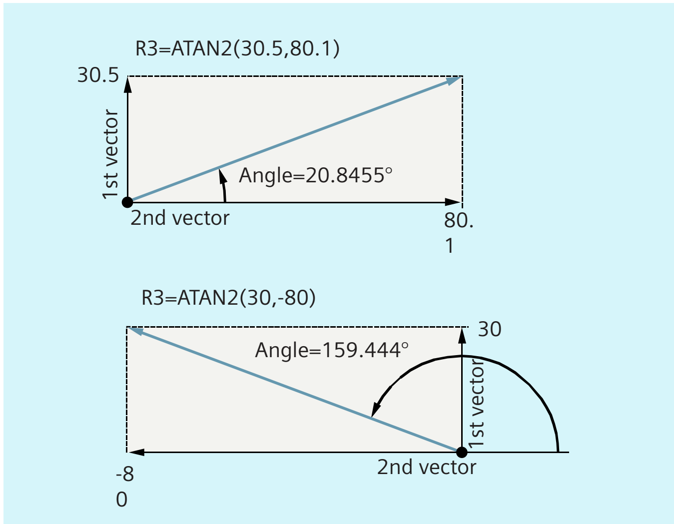
Example: 7 MOD 4.1 = 2.9

Arc tangent 2: ATAN2

The arithmetic function ATAN2 calculates the angle of the total vector from two mutually perpendicular vectors.

The result is in one of four quadrants (-180° < 0 < +180°).

The angular reference is always based on the 2nd value in the positive direction.



Programming examples

Program code	Comment
`R1=R1+1`	`; New R1 = old R1 + 1`
`R1=R2+R3  R4=R5-R6  R7=R8*R9`	
`R10=R11/R12  R13=SIN(25.3)`	
`R14=R1*R2+R3`	`; Multiplication or division takes precedence over addition or subtraction.`
`R14=(R1+R2)*R3`	`; Expressions and parentheses are calculated first.`
`R15=SQRT(POT(R1)+POT(R2))`	`; Inner parentheses are resolved first:`
	`R15 = square root of ( (R1^2 + R2^2) )`
`RESFRAME=FRAME1:FRAME2`	`; FRAME logic operation with chain operator`
`FRAME3=CTRANS(…):CROT(…)`	`Value assignment at a FRAME component`

4.1.3.2 Comparison and logic operations

Comparison operations can be used, for example, to formulate a jump condition. Complex expressions can also be compared.

The comparison operations are applicable to variables of type `CHAR`, `INT`, `REAL` and `BOOL`. The code value is compared with the `CHAR` type.
For types `STRING`, `AXIS` and `FRAME`, the following are possible: == and <>, which can be used for `STRING` type operations, even in synchronous actions.

The result of comparison operations is always of `BOOL` type.

Logic operators are used to link truth values.

The logical operations can only be applied to type `BOOL` variables. However, they can also be applied to the `CHAR`, `INT` and `REAL` data types via internal type conversion.

For the logic (Boolean) operations, the following applies to the BOOL, CHAR, INT and REAL data types:

- 0 corresponds to: FALSE
- Not equal to 0 means: TRUE

Bit-by-bit logic operators

Logic operations can also be applied to single bits of types CHAR and INT. Type conversion is automatic.

Programming

Relational operator	Meaning
==	Equal to
<>	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Logic operator	Meaning
AND	AND
OR	OR
NOT	Negation
XOR	Exclusive OR

Bit-by-bit logic operator	Meaning
B_AND	Bit-by-bit AND
B_OR	Bit-by-bit OR
B_NOT	Bit-by-bit negation
B_XOR	Bit-by-bit exclusive OR

Note

In arithmetic expressions, the execution order of all the operators can be specified by parentheses, in order to override the normal priority rules.

Note

Spaces must be left between BOOLEAN operands and operators.

Note

The operator B_NOT only refers to one operand. This is located after the operator.

Examples

Example 1: Comparison operators

```
IF R10>=100 GOTOF DEST
```

or

```
R11=R10>=100
IF R11 GOTOF DEST
```

The result of the `R10>=100` comparison is first buffered in `R11`.

Example 2: Logic operators

```
IF (R10<50) AND ($AA_IM[X]>=17.5) GOTOF DESTINATION
```

or

```
IF NOT R10 GOTOB START
```

`NOT` only refers to one operand.

Example 3: Bit-by-bit logic operators

```
IF $MC_RESET_MODE_MASK B_AND 'B10000' GOTOF  ACT_PLANE
```

4.1.3.3 Priority of the operations

Each operator is assigned a priority. When an expression is evaluated, the operators with the highest priority are always applied first. Where operators have the same priority, the evaluation is from left to right.

In arithmetic expressions, the execution order of all the operators can be specified by parentheses, in order to override the normal priority rules.

Order of operators

From the highest to lowest priority

1.	NOT, B_NOT	Negation, bit-by-bit negation
2.	*, /, DIV, MOD	Multiplication, division
3.	+, –	Addition, subtraction
4.	B_AND	Bit-by-bit AND
5.	B_XOR	Bit-by-bit exclusive OR
6.	B_OR	Bit-by-bit OR
7.	AND	AND
8.	XOR	Exclusive OR
9.	OR	OR
10.	<<	Concatenation of strings, result type STRING
11.	==, <>, >, <, >=, <=	Comparison operators

Note

The concatenation operator ":" for Frames must not be used in the same expression as other operators. A priority level is therefore not required for this operator.

Example: IF statement

```
If (otto==10) and (anna==20) gotof end
```

4.1.3.4 Power function (POT)

With the predefined function POT() it is possible to calculate any power for an input value.

POT() is programmable in the NC program and in synchronous actions.

Syntax

NC program:

```
POT(<x>,[<p>])
```

Synchronous action:

```
DO POT(<x>,[<p>])
```

Meaning

`POT()`	Function call for calculating a power	
`<x>`	Parameter 1: Input value (basis of the power function)	
	Type:	REAL

<p>	Parameter 2 (optional): Exponent of the power function (≙ degree of the power function)		
	Type:	REAL	
	Value:	$p \geq 0$ $p \in \mathbb{N}_0$	In the simplest case, the exponent is a non-negative integer. The formula for power calculation is then as follows: $f(x) = x^p$
			Restriction of the value range of the base <x>: **None**
		$p \leq -1$ $p \in \mathbb{Z}$	For negative integer exponents, the formula can be rearranged as follows: $f(x) = \frac{1}{x^p}$
			Restriction of the value range of the base <x>: Since division by 0 is not permitted, the value for the base must always be unequal to zero for negative exponents: **$x \neq 0$**
		$p = n/m$ $n/m > 0$ $n \in \mathbb{N}_0$ $m \in \mathbb{N}$	If the exponent is a non-negative rational number of the form n/m, then the formula can also be represented as a root function: $f(x) = x^p = x^{n/m} = \sqrt[m]{x^n}$
			Restriction of the value range of the base <x>: Since the root of a negative number cannot be mapped, the value for the base must always be greater than (or equal to) zero for root functions with odd exponents: **$x \geq 0$ for odd values of n**
		$p = -n/m$ $n/m < 0$ $n \in \mathbb{N}_0$ $m \in \mathbb{N}$	If the exponent is a negative rational number of the form n/m, then the formula can also be represented as a root function in a fraction: $f(x) = \frac{1}{\sqrt[m]{x^n}}$
			Restriction of the value range of the base <x>: Since the root of a negative number cannot be mapped and division by 0 is not allowed, the following is mandatory: • For root functions in the denominator of a fraction and with odd exponents, the value for the base must be greater than zero: **$x > 0$ for odd values of n** • For root functions in the denominator of a fraction and with even exponents, the value for the base must not be zero: **$x \neq 0$ for even values of n**
	Without specifying an exponent, the function calculates the power of degree 2 (square function): POT(<x>) → x^2		

Note

When entering the base <x> and the optional exponent <p>, applicable arithmetic laws must be observed. Not possible are, for example, divisions by zero or extracting a root with a negative base. In such cases, the function call leads to the output of an alarm.

Examples

Program block	Result	Note
`POT(2)`	Return value = $2^2 = 4$	Power of degree 2 to base 2
`POT(2,3)`	Return value = $2^3 = 8$	Power of degree 3 to base 2
`POT(0,-1)`	Alarm	Division by 0 not allowed
`POT(-1,0.5)`	Alarm	Root of a negative number not permissible

4.1.3.5 Roundup (ROUNDUP)

Input values, type REAL (fractions with decimal point) can be rounded up to the next higher integer number using the "ROUNDUP" function.

Syntax

```
ROUNDUP(<x>)
```

Meaning

`ROUNDUP`:	Command to roundup an input value
`<x>`:	Input value, type REAL

Note

Input value, type INTEGER (an integer number) is returned unchanged.

Examples

Example 1: Various input values and their rounding results

Example	Rounding up result
`ROUNDUP(3.1)`	4.0
`ROUNDUP(3.6)`	4.0
`ROUNDUP(-3.1)`	-3.0
`ROUNDUP(-3.6)`	-3.0
`ROUNDUP(3.0)`	3.0
`ROUNDUP(3)`	3.0

Example 2: ROUNDUP in the NC program

Program code
`N10 X=ROUNDUP(3.5) Y=ROUNDUP(R2+2)`
`N15 R2=ROUNDUP($AA_IM[Y])`
`N20 WHEN X=100 DO Y=ROUNDUP($AA_IM[X])`
`...`

4.1.3.6 Precision correction on comparison errors (TRUNC)

The TRUNC command truncates the operand multiplied by a precision factor.

Settable precision for comparison commands

Program data of type REAL is displayed internally with 64 bits in IEEE format. This display format can cause decimal numbers to be displayed imprecisely and lead to unexpected results when compared with the ideally calculated values.

Relative equality

To prevent the imprecision caused by the display format from interfering with program flow, the comparison commands do not check for absolute equality, but rather for relative equality.

Syntax

Precision correction on comparison errors

```
TRUNC (R1*1000)
```

Meaning

`TRUNC:`	Truncate decimal places

Relative quality of 10^{-12} taken into account for

- Equality: (==)
- Inequality: (<>)
- Greater than or equal to: (>=)
- Less than or equal to: (<=)
- Greater/less than: (><) with absolute equality
- Greater than: (>)
- Less than: (<)

Compatibility

For compatibility reasons, the check for relative quality for (>) and (<) can be deactivated by setting machine data MD10280 $MN_ PROG_FUNCTION_MASK Bit0 = 1.

Note

Comparisons with data of type REAL are subject to a certain imprecision for the above reasons. If deviations are unacceptable, use INTEGER calculation by multiplying the operands by a precision factor and then truncating with TRUNC.

Synchronized actions

The response described for the comparison commands also applies to synchronized actions.

Examples

Example 1: Precision considerations

Program code	Comments
`N40 R1=61.01 R2=61.02 R3=0.01`	`;Assignment of initial values`
`N41 IF ABS(R2-R1) > R3 GOTOF ERROR`	`; Jump would have been executed up until now`
`N42 M30`	`; End of program`
`N43 ERROR: SETAL(66000)`	
`R1=61.01 R2=61.02 R3=0.01`	`;Assignment of initial values`
`R11=TRUNC(R1*1000) R12=TRUNC(R2*1000)`	`; Accuracy correction`
`R13=TRUNC(R3*1000)`	
`IF ABS(R12-R11) > R13 GOTOF ERROR`	`; Jump is no longer executed`
`M30`	`; End of program`
`ERROR: SETAL(66000)`	

Example 2: Calculate and evaluate the quotient of both operands

Program code	Comments
`R1=61.01 R2=61.02 R3=0.01`	`;Assignment of initial values`
`IF ABS((R2-R1)/R3)-1) > 10EX-5 GOTOF ERROR`	`; Jump is not executed`
`M30`	`; End of program`
`ERROR: SETAL(66000)`	

4.1.4 String operations

Sting operations

In addition to the classic operations "assign" and "comparison" the following string operations are possible:

- Type conversion to STRING (AXSTRING) (Page 437)
- Type conversion from STRING (NUMBER, ISNUMBER, AXNAME) (Page 438)
- Concatenation of strings (<<) (Page 439)
- Conversion to lower/upper case letters (TOLOWER, TOUPPER) (Page 440)
- Determine length of string (STRLEN) (Page 441)
- Search for character/string in the string (INDEX, RINDEX, MINDEX, MATCH) (Page 441)
- Selection of a substring (SUBSTR) (Page 442)
- Reading and writing of individual characters (Page 443)
- Formatting a string (SPRINT) (Page 445)

Special significance of the 0 character

Internally, the 0 character is interpreted as the end identifier of a string. If a character is replaced with the 0 character, the string is truncated.

Example:

Program code	**Comment**
`DEF STRING[20] STRG="axis . stationary"`	
`STRG[6]="X"`	
`MSG(STRG)`	`; Supplies the message "axis X stationary".`
`STRG[6]=0`	
`MSG(STRG)`	`; Supplies the message "axis".`

4.1.4.1 Type conversion to STRING (AXSTRING)

The function "type conversion to STRING" allows variables of different types to be used as a component of a message (MSG).

When using the << operator this is realized implicitly for data types INT, REAL, CHAR and BOOL (see " Concatenation of strings (<<) (Page 439) ").

An INT value is converted to normal readable format. REAL values convert with up to 10 decimal places.

Type AXIS variables can be converted to STRING using the `AXSTRING` command.

Syntax

```
<STRING_RES> = << <any_type>
```

```
<STRING_RES> = AXSTRING(<axis identifier>)
```

Meaning

<STRING_RES>:	Variable for the result of the type conversion	
	Type:	STRING
<any_type>:	Variable types INT, REAL, CHAR, STRING and BOOL	
AXSTRING:	The AXSTRING command supplies the specified axis identifier as string.	
<axis identifier>:	Variable for axis identifier	
	Type:	AXIS

Note

FRAME variables cannot be converted.

4.1.4.2 Type conversion from STRING (NUMBER, ISNUMBER, AXNAME)

A conversion is made from STRING to REAL using the NUMBER command. The ability to be converted can be checked using the ISNUMBER command.

A string is converted into the axis data type using the AXNAME command.

Syntax

```
<REAL_RES>=NUMBER("<string>")
<BOOL_RES>=ISNUMBER("<string>")
<AXIS_RES>=AXNAME("<string>")
```

Meaning

NUMBER:	The NUMBER command returns the number represented by the <string> as REAL value.		
<string>:	Type STRING variable to be converted		
<REAL_RES>:	Variable for the result of the type conversion with NUMBER		
	Type:	REAL	
ISNUMBER:	The ISNUMBER command checks whether the <string> can be converted into a valid number.		
<BOOL_RES>:	Variable for the result of the interrogation with ISNUMBER		
	Type:	BOOL	
	Value:	TRUE	ISNUMBER supplies the value TRUE, if the <string> represents a valid REAL number in compliance with the language rules.
		FALSE	If ISNUMBER supplies the value FALSE, when calling NUMBER with the same <string>, an alarm is initiated.

AXNAME:	The AXNAME command converts the specified <string> into an axis identifier. **Note:** If the <string> cannot be assigned a configured axis identifier, an alarm is initiated.	
<AXIS_RES>:	Variable for the result of the type conversion with AXNAME	
	Type:	AXIS

Example

Program code	Comment
DEF BOOL BOOL_RES	
DEF REAL REAL_RES	
DEF AXIS AXIS_RES	
REAL_RES == 1234.9876Ex-7	; BOOL_RES == TRUE
BOOL_RES=ISNUMBER("1234XYZ")	; BOOL_RES == FALSE
REAL_RES=NUMBER("1234.9876Ex-7")	; REAL_RES == 1234.9876Ex-7
AXIS_RES=AXNAME("X")	; AXIS_RES == X

4.1.4.3 Concatenation of strings (<<)

The function "concatenation strings" allows a string to be configured from individual components.

The concatenation is realized using the operator "<<". This operator has STRING as the target type for all combinations of basic types CHAR, BOOL, INT, REAL, and STRING. Any conversion that may be required is carried out according to existing rules.

Syntax

```
<any_type> << <any_type>
```

Meaning

<any_type>:	Variable, type CHAR, BOOL, INT, REAL or STRING
<< :	Operator to chain variables (<any_type>) to configure a character string (type STRING). This operator is also available alone as a so-called "unary" variant. This can be used for explicit type converter to STRING (not for FRAME and AXIS): `<< <any_type>`

For example, such a message or a command can be configured from text lists and parameters can be inserted (for example a block name):
`MSG(STRG_TAB[LOAD_IDX]<<BLOCK_NAME)`

Note

The intermediate results of string concatenation must not exceed the maximum string length.

Note

The FRAME and AXIS types cannot be used together with the operator "<<".

Examples

Example 1: Concatenation of strings

Program code	Comment
`DEF INT IDX=2`	
`DEF REAL VALUE=9.654`	
`DEF STRING[20] STRG="INDEX:2"`	
`IF STRG=="Index:"<<IDX GOTOF NO_MSG`	
`MSG("Index:"<<IDX<<"/value:"<<VALUE)`	`; Display: "Index:2/value:9.654"`
`NO_MSG:`	

Example 2: Explicit type conversion with <<

Program code	Comment
`DEF REAL VALUE=3.5`	
`<<VALUE`	`; The specified REAL type variable is converted into a STRING type.`

4.1.4.4 Conversion to lower/upper case letters (TOLOWER, TOUPPER)

The "conversion to lowercase/uppercase letters" function allows all of the letters of a string to be converted into a standard representation.

Syntax

```
<STRING_RES>=TOUPPER("<string>")
<STRING_RES>=TOLOWER("<string>")
```

Meaning

`TOUPPER:`	Using the `TOUPPER` command, all of the letters in a character string are converted into **uppercase** letters.	
`TOLOWER:`	Using the `TOLOWER` command, all of the letters in a character string are converted into **lowercase** letters.	
`<string>:`	Character string that is to be converted	
	Type:	STRING
`<STRING_RES>:`	Variable for the result of the conversion	
	Type:	STRING

Example

Because user inputs can be initiated on the user interface, they can be given standard capitalization (uppercase or lowercase):

Program code

```
DEF STRING [29] STRG
...
IF "LEARN.CNC"==TOUPPER(STRG) GOTOF LOAD_LEARN
```

4.1.4.5 Determine length of string (STRLEN)

The `STRLEN` command determines the length of a character string.

Syntax

```
<INT_RES>=STRLEN("<STRING>")
```

Meaning

`STRLEN`:	The `STRLEN` command determines the length of the specified character string. The number of characters that are not the 0 character, counting from the beginning of the string is returned.	
`<string>`:	Character string whose length is to be determined	
	Type:	STRING
`<INT_RES>`:	Variable for the result of the determination	
	Type:	INT

Example

In conjunction with the single character access, this function allows the end of a character string to be determined:

Program code

```
IF (STRLEN(BLOCK_NAME)>10) GOTOF ERROR
```

4.1.4.6 Search for character/string in the string (INDEX, RINDEX, MINDEX, MATCH)

This functionality searches for single characters or a string within a string. The function results specify where the character/string is positioned in the string that has been searched.

Syntax

`INT_RES=INDEX(STRING,CHAR)` **; Result type: INT**

`INT_RES=RINDEX(STRING,CHAR)` **; Result type: INT**

`INT_RES=MINDEX(STRING,STRING)` **; Result type: INT**

`INT_RES=MINDEX(STRING,STRING)` **; Result type: INT**

Semantics

Search functions: It supplies the position in the string (first parameter) where the search has been successful. If the character/string cannot be found, then the value -1 is returned. The first character has position 0.

Meaning

`INDEX`:	Searches for the character specified as second parameter (from the beginning) in the first parameter.
`RINDEX`:	Searches for the character specified as second parameter (from the end) in the first parameter.
`MINDEX`:	Corresponds to the INDEX function, except for the case that a list of characters is transferred (as string) in which the index of the first found character is returned.
`MATCH`:	Searches for a string in a string.

This allows strings to be broken up according to certain criteria, for example, at positions with blanks or path separators ("/").

Example

Breaking up an input into path and block names

Program code	Comment
`DEF INT PFADIDX, PROGIDX`	
`DEF STRING[26] INPUT`	
`DEF INT LISTIDX`	
`INPUT = "/_N_MPF_DIR/_N_EXECUTE_MPF"`	
`LISTIDX = MINDEX (INPUT, "M,N,O,P") + 1`	`; The value returned in LISTIDX is 3; because "N" is the first character in the parameter INPUT from the selection list starting from the beginning.`
`PFADIDX = INDEX (INPUT, "/") +1`	`; Therefore the following applies: PFADIDX = 1`
`PROGIDX = RINDEX (INPUT, "/") +1`	`; Therefore the following applies: PROGIDX = 12`
	`; The SUBSTR function introduced in the next section can be used to break-up variable INPUT into the components "path" and "module":`
`VARIABLE = SUBSTR (INPUT, PFADIDX, PROGIDX-PFADIDX-1)`	`; Then returns "_N_MPF_DIR"`
`VARIABLE = SUBSTR (INPUT, PROGIDX)`	`; Then returns "_N_EXECUTE_MPF"`

4.1.4.7 Selection of a substring (SUBSTR)

Arbitrary parts within a string can be read with the SUBSTRING function.

Syntax

```
<STRING_RES>=SUBSTR(<string>,<index>,<length>)
<STRING_RES>=SUBSTR(<string>,<index>)
```

Meaning

`SUBSTR`:	This function returns a substring from <string>, starting with <index> with the specified <length>. If the parameter <length> is not specified, the function returns a substring starting with <index> until the end of the string.
`<index>`:	Start position of the substring within the string. If the start position is after the end of the string, an empty string (" ") is returned. First character of the string: Index = 0 Range of values: 0 ... (string length - 1)
`<length>`:	Length of the substring. If too long a length is specified, only the substring up to the end of the string is returned. Range of values: 1 ... (string length - 1)

Example

Program code	**Comment**
`DEF STRING[29] RES`	
`;                       1`	
`;             0123456789012345678`	
`RES = SUBSTR("QUITTUNG: 10 to 99", 10, 2)`	`; RES == "10"`
`RES = SUBSTR("QUITTUNG: 10 to 99", 10)`	`; RES == "10 to 99"`

4.1.4.8 Reading and writing of individual characters

Individual characters can be read and written within a string.

The following supplementary conditions must be observed:

- Only possible with user-defined variables, not with system variables
- Individual characters of a string are only transferred "call by value" for subprogram calls

Syntax

```
<Character>=<string>[<index>]
<Character>=<string_array>[<array_index>,<index>]
<String>[<index>]=<character>
<String_array>[<array_index>,<index>]=<character>
```

Meaning

`<string>:`	Any string
`<character>:`	Variable of type CHAR
`<index>:`	Position of the character within the string. First character of the string: Index = 0 Range of values: 0 ... (string length - 1)

Examples

Example 1: Variable message

Program code	Comment
`;                          0123456789`	
`DEF STRING [50] MESSAGE = "Axis n has reached position"`	
`MESSAGE [6] = "X"`	
`MSG (MESSAGE)`	`; "Axis X has reached position"`

Example 2: Evaluating a system variable

Program code	Comment
`DEF STRING[50] STRG`	`; Buffer for system variable`
`...`	
`STRG = $P_MMCA`	`; Load system variable`
`IF STRG[0] == "E" GOTO ...`	`; Evaluating the system variable`

Example 3: Parameter transfer "call by value" and "call by reference"

Program code	Comment
`;                       0123456`	
`DEF STRING[50] STRG = "Axis X"`	
`DEF CHAR CHR`	
`...`	
`EXTERN UP_VAL(ACHSE)`	`; Definition of subprogram with "call by value" parameters`
`EXTERN UP_REF(VAR ACHSE)`	`; Definition of subprogram with "call by reference" parameters`
`...`	
`UP_VAL(STRG[6])`	`; Parameter transfer "by value"`
`...`	
`CHR = STRG[6]`	`; Buffer`
`UP_REF(CHR)`	`; Parameter transfer "by reference"`

4.1.4.9 Formatting a string (SPRINT)

Using the pre-defined SPRINT function, character strings can be formatted and e.g. prepared for output on external devices (also see "Process DataShare - output to an external device/file (EXTOPEN, WRITE, EXTCLOSE) (Page 866)").

Syntax

```
"<Result_string>"=SPRINT("<Format_string>",<value_1>,<value_2>,...,
<value_n>)
```

Meaning

`SPRINT:`	Identifier for a pre-defined function that supplies a value, type STRING.
`"<Format_String>":`	Character string that contains fixed and variable elements. The variable elements are defined using the format control character % and a subsequent format description.
`< value_1>,< value_2>,…,< value_n>:`	Value in the form of a constant or NC variables, which is inserted at the location where the nth format control character % is located, corresponding to the format description in the <format_string>.
`"<result_string>":`	Formatted character string (maximum 400 bytes)

Format descriptions available

`%B:`	Conversion into the "TRUE" string, if the value to be converted:
	• Is not equal to 0.
	• Is not an empty string (for string values).
	Conversion into the "FALSE" string, if the value to be converted:
	• Is equal to 0.
	• Is an empty string.
	Example:
	`N10 DEF BOOL BOOL_VAR=1`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF BOOL_VAR:%B", BOOL_VAR)`
	Result: The character string "CONTENT OF BOOL_VAR:TRUE" is written to the RESULT string variable.

`%C:`	Conversion into an ASCII character.
	Example:
	`N10 DEF CHAR CHAR_VAR="X"`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF CHAR_VAR:%C",CHAR_VAR)`
	Result: The character string "CONTENT OF CHAR_VAR:X is written to the string variable RESULT.

%D:	Conversion into a string with an integer value (INTEGER). **Example:** N10 DEF INT INT_VAR=123 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF INT_VAR:%D",INT_VAR) Result: The character string "CONTENT OF INT_VAR:123" is written to the string variable RESULT.
%<m>D:	Conversion into a string with an integer value (INTEGER). The string has a minimum length of <m> characters. The missing locations are filled with spaces, left-justified. **Example:** N10 DEF INT INT_VAR=-123 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF INT_VAR:%6D",INT_VAR) Result: The character string "CONTENT OF INT_VAR:xx-123" is written to string variable RESULT ("x" in the example represents spaces).
%F:	Conversion into a string with a decimal number with 6 decimal places. Where relevant, the decimal places are rounded-off or filled with 0. **Example:** N10 DEF REAL REAL_VAR=-1.2341234EX+03 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%F",REAL_VAR) Result: The string variable RESULT is written with the character string "CONTENT OF REAL_VAR: -1234.123400".
%<m>F:	Conversion into a string with a decimal number with 6 decimal places and a total length of at least <m> characters. Where relevant, the decimal places are rounded-off or filled with 0. Missing characters are filled up to the total length <m> using spaces, left-justified. **Example:** N10 DEF REAL REAL_VAR=-1.23412345678EX+03 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%15F",REAL_VAR) Result: The string variable RESULT is written with the character string "CONTENT OF REAL_VAR: xxx-1234.123457" (where "x" is a placeholder for space).
%.<n>F:	Conversion into a string with a decimal number with <n> decimal places. Where relevant, the decimal places are rounded-off or filled with 0. **Example:** N10 DEF REAL REAL_VAR=-1.2345678EX+03 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%.3F",REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:-1234.568" is written to the string variable RESULT.
%<m>.<n>F:	Conversion into a string with a decimal number with <n> decimal places and a total length of at least <m> characters. Where relevant, the decimal places are rounded-off or filled with 0. Missing characters are filled up to the total length <m> using spaces, left-justified. **Example:** N10 DEF REAL REAL_VAR=-1.2341234567890EX+03 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%10.2F",REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:xx-1234.12" is written to the string variable RESULT ("x" in the example represents spaces).

%E:	Conversion into a string with a decimal number in the exponential representation. The mantissa is saved, normalized with one pre-decimal place and 6 decimal places. Where relevant, the decimal places are rounded-off or filled with 0. The exponent starts with the keyword "EX". It is followed by the sign ("+" or "-") and a two or three-digit number. **Example:** N10 DEF REAL REAL_VAR=-1234.567890 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%E",REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:-1.234568EX+03" is written to the string variable RESULT.
%<m>E:	Conversion into a string with a decimal number in the exponential representation and a total length of at least <m> characters. The missing characters are filled with spaces, left-justified. The mantissa is saved, normalized with one pre-decimal place and 6 decimal places. Where relevant, the decimal places are rounded-off or filled with 0. The exponent starts with the keyword "EX". It is followed by the sign ("+" or "-") and a two or three-digit number. **Example:** N10 DEF REAL REAL_VAR=-1234.5 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%20E",REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:xxxxxx-1.234500EX+03" is written to the string variable RESULT ("x" in the example represents spaces).
%.<n>E:	Conversion into a string with a decimal number in the exponential representation. The mantissa is saved, normalized with one pre-decimal place and <n> decimal places. Where relevant, the decimal places are rounded-off or filled with 0. The exponent starts with the keyword "EX". It is followed by the sign ("+" or "-") and a two or three-digit number. **Example:** N10 DEF REAL REAL_VAR=-1234.5678 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%.2E",REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:-1.23EX+03" is written to the string variable RESULT.
%<m>.<n>E:	Conversion into a string with a decimal number in the exponential representation and a total length of at least <m> characters. The missing characters are filled with spaces, left-justified. The mantissa is saved, normalized with one pre-decimal place and <n> decimal places. Where relevant, the decimal places are rounded-off or filled with 0. The exponent starts with the keyword "EX". It is followed by the sign ("+" or "-") and a two or three-digit number. **Example:** N10 DEF REAL REAL_VAR=-1234.5678 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%12.2E", REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:xx-1.23EX+03" is written to the string variable RESULT ("x" in the example represents spaces).

`%G`:	Conversion into a string with a decimal number – depending on the value range – in a decimal or exponential representation: If the absolute value to be represented is less than 1.0EX-04 or greater than/equal to 1.0EX+06, then the exponential notation is selected, otherwise the decimal notation. A maximum of six significant places are displayed or if required, rounded-off.
	Example with decimal notation:
	`N10 DEF REAL REAL_VAR=1.234567890123456EX-04`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%G",REAL_VAR)`
	Result: The character string "CONTENT OF REAL_VAR:0.000123457" is written to the string variable RESULT.
	Example with exponential notation:
	`N10 DEF REAL REAL_VAR=1.234567890123456EX+06`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%G",REAL_VAR)`
	Result: The character string "CONTENT OF REAL_VAR:1.23457EX+06" is written to the string variable RESULT.

`%<m>G`:	Conversion into a string with a decimal number – depending on the value range – in a decimal or exponential notation (like `%G`). The string has a total length of at least <m> characters. The missing characters are filled with spaces, left-justified.
	Example with decimal notation:
	`N10 DEF REAL REAL_VAR=1.234567890123456EX-04`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%15G",REAL_VAR)`
	Result: The character string "CONTENT OF REAL_VAR:xxxx0.000123457" is written to the string variable RESULT ("x" in the example represents spaces).
	Example with exponential notation:
	`N10 DEF REAL REAL_VAR=1.234567890123456EX+06`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%15G",REAL_VAR)`
	Result: The character string "CONTENT OF REAL_VAR:xxx1.23457EX+06" is written to the string variable RESULT ("x" in the example represents spaces).

`%.<n>G`:	Conversion into a string with a decimal number – depending on the value range – in a decimal or exponential representation. A maximum of <n> significant places are displayed or if required, rounded-off. If the absolute value to be represented is less than 1.0EX-04 or greater than/equal to 1.0EX(+<n>), then the exponential notation is selected, otherwise the decimal notation.
	Example with decimal notation:
	`N10 DEF REAL REAL_VAR=1.234567890123456EX-04`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%.3G",REAL_VAR)`
	Result: The character string "CONTENT OF REAL_VAR:0.000123" is written to the string variable RESULT.
	Example with exponential notation:
	`N10 DEF REAL REAL_VAR=1.234567890123456EX+03`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT = SPRINT("CONTENT OF REAL_VAR:%.3G",REAL_VAR)`
	Result: The character string "CONTENT OF REAL_VAR:1.23EX+03" is written to the string variable RESULT.

%<m>.<n>G:	Conversion into a string with a decimal number – depending on the value range – in a decimal or exponential notation (like %.<n>G). The string has a total length of at least <m> characters. The missing characters are filled with spaces, left-justified. **Example with decimal notation:** N10 DEF REAL REAL_VAR=1.234567890123456EX-04 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%12.4G",REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:xxx0.0001235" is written to the string variable RESULT ("x" in the example represents spaces). **Example with exponential notation:** N10 DEF REAL REAL_VAR=1.234567890123456EX+04 N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF REAL_VAR:%12.4G",REAL_VAR) Result: The character string "CONTENT OF REAL_VAR:xx1.235EX+06" is written to the string variable RESULT ("x" in the example represents spaces).
%.<n>P:	Converting a REAL value into an INTEGER value taking into account <n> decimal places. The INTEGER value is output as a 32-bit binary number. If the value to be converted cannot be represented with 32 bits, then processing is interrupted with an alarm. As a byte sequence generated using the format statement %.<n>P can also contain binary zeroes, then the total string that is generated in this way no longer corresponds to the conventions of the NC data type STRING. As a consequence, it can neither be saved in a variable, type STRING, nor be further processed using the string commands of the NC language. The only possible use is to transfer the parameter to the WRITE command with output at an appropriate external device (see the following example). As soon as the <Format_String> contains a format description, type %P then the complete string, with the exception of the binary number generated with %.<n>P, is output corresponding to the MD10750 $MN_SPRINT_FORMAT_P_CODE in the ASCII character code, ISO (DIN6024) or EIA (RS244). If a character that cannot be converted is programmed, then processing is interrupted with an alarm. **Example:** N10 DEF REAL REAL_VAR=123.45 N20 DEF INT ERROR N30 DEF STRING[20] EXT_DEVICE="/ext/dev/1" ... N100 EXTOPEN(ERROR,EXT_DEVICE) N110 IF ERROR <> 0 ... ; error handling N200 WRITE(ERROR,EXT_DEVICE,SPRINT("INTEGER BINARY CODED:%.3P",REAL_VAR) N210 IF ERROR <> 0 … ; error handling Result: The string "INTEGER BINARY CODED: 'H0001E23A'" is transferred to the output device /ext/dev/1. The hexadecimal value 0x0001E23A corresponds to the decimal value 123450.

`%<m>.<n>P:`	Conversion of a REAL value corresponding to the setting in machine data MD10751 $MN_SPRINT_FORMAT_P_DECIMAL into a string with:
	• An integer of <m> + <n> places **or**
	• A decimal number with a maximum of <m> pre-decimal places and precisely <n> decimal places.
	Just the same as for the format description `%.<n>P`, the complete string is saved in the character code defined by MD10750 $MN_SPRINT_FORMAT_P_CODE.
	Conversion for MD10751 = 0:
	The REAL value is converted into a string with an integer number of <m> + <n> places. If required, decimal places are rounded-off to <n> places or filled with 0. The missing pre-decimal places are filled with spaces. The minus sign is attached, left-justified; a space is entered instead of the plus sign.
	Example:
	`N10 DEF REAL REAL_VAR=-123.45`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("PUNCHED TAPE FORMAT:%5.3P",REAL_VAR)`
	Result: The character string "PUNCHED TAPE FORMAT:-xx123450" is written to the string variable RESULT ("x" in the example represents spaces).
	Conversion for MD10751 = 1:
	The REAL value is converted into a string with a decimal number with a maximum of <m> pre-decimal places and precisely <n> decimal places. Where necessary, the pre-decimal places are cut-off and the decimal places are rounded-off or filled with 0. If <n> is equal to 0, then the decimal point is also omitted.
	Example:
	`N10 DEF REAL REAL_VAR1=-123.45`
	`N20 DEF REAL REAL_VAR2=123.45`
	`N30 DEF STRING[80] RESULT`
	`N40 RESULT=SPRINT("PUNCHED TAPE FORMAT:%5.3P VAR2:%2.0P",`
	`REAL_VAR1,REAL_VAR2)`
	Result: The character string "PUNCHED TAPE FORMAT:-123.450 VAR2:23" is written to the string variable RESULT.

`%S:`	Inserting a string.
	Example:
	`N10 DEF STRING[16] STRING_VAR="ABCDEFG"`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF STRING_VAR:%S",STRING_VAR)`
	Result: The character string "CONTENT OF STRING_VAR:ABCDEFG" is written to the string variable RESULT.

`%<m>S:`	Inserting a string with a minimum of <m> characters. The missing places are filled with spaces.
	Example:
	`N10 DEF STRING[16] STRING_VAR="ABCDEFG"`
	`N20 DEF STRING[80] RESULT`
	`N30 RESULT=SPRINT("CONTENT OF STRING_VAR:%10S",STRING_VAR)`
	Result: The character string "CONTENT OF STRING_VAR:xxxABCDEFG" is written to the string variable RESULT ("x" in the example represents spaces).

%.<n>S:	Inserting <n> characters of a string (starting with the first character). **Example:** N10 DEF STRING[16] STRING_VAR="ABCDEFG" N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF STRING_VAR:%.3S",STRING_VAR) Result: The character string "CONTENT OF STRING_VAR:ABC" is written to the string variable RESULT.
%<m>.<n>S:	Inserting <n> characters of a string (starting with the first character). The total length of the generated string has at least <m> characters. The missing places are filled with spaces. **Example:** N10 DEF STRING[16] STRING_VAR="ABCDEFG" N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("CONTENT OF STRING_VAR:%10.5S", STRING_VAR) Result: The character string "CONTENT OF STRING_VAR:xxxxxABCDE" is written to the string variable RESULT ("x" in the example represents spaces).
%X:	Converting an INTEGER value into a string with the hexadecimal notation. **Example:** N10 DEF INT INT_VAR='HA5B8’ N20 DEF STRING[80] RESULT N30 RESULT=SPRINT("INTEGER HEXADECIMAL:%X",INT_VAR) Result: The character string "INTEGER HEXADECIMAL:A5B8" is written to the string variable RESULT.

Note

A property of the NC language, where a distinction is not made between uppercase and lowercase letters for identifiers and keywords, also applies to the format descriptions. As a consequence, you can program using either lowercase or uppercase letters without any functional difference.

Combination options

The following table provides information as to which NC data types can be combined with which format description. The rules regarding implicit data type conversion apply (see "Data types (Page 411)").

	NC data types						
	BOOL	**CHAR**	**INT**	**REAL**	**STRING**	**AXIS**	**FRAME**
%B	+	+	+	+	+	-	-
%C	-	+	-	-	+	-	-
%D	+	+	+	+	-	-	-
%F	-	-	+	+	-	-	-
%E	-	-	+	+	-	-	-
%G	-	-	+	+	-	-	-
%S	-	+	-	-	+	-	-
%X	+	+	+	-	-	-	-
%P	-	-	+	+	-	-	-

Note

The table indicates that the NC data types AXIS and FRAME cannot be directly used in the SPRINT function. However it is possible:

- To convert the AXIS data type into a string using the AXSTRING function – which can then be processed with SPRINT.
- To read the individual values of the FRAME data type per frame component access. As a consequence, a REAL data type is obtained, which can be processed with SPRINT.

4.1.5 Program jumps and branches

4.1.5.1 Return jump to the start of the program (GOTOS)

The GOTOS command can be used to jump back to the beginning of a main or subprogram in order to repeat the program.

Machine data can be used to set that for every return jump is made to the program start:

- The program runtime is set to "0".
- Workpiece counting is incremented by the value "1".

Syntax

```
GOTOS
```

Meaning

GOTOS:	Jump statement where the destination is the beginning of the program.	
	The execution is controlled via the NC/PLC interface signal: DB3200.DBX16.0 (control program branch)	
	Value	**Meaning**
	0	No return jump to the beginning of the program. Program execution is resumed with the next part program block after GOTOS.
	1	Return jump to the beginning of the program. The part program is repeated.

Constraints

- GOTOS internally initiates a STOPRE (preprocessing stop).
- For a part program with data definitions (local user variables (LUD)) with the GOTOS, a jump is made to the first program block after the definition section, i.e. data definitions are not executed again. The defined variables therefore retain the value reached in the GOTOS block and are not reset to the default values programmed in the definition section.

- If programs with jump instructions are to be executed from an external program memory via the "Execution from external source" function, the jump destinations must be located within the reload memory.
  This condition is especially problematic regarding jump instructions to the start of the program (GOTOS), since the programs are typically much too large to fit entirely in the reload memory. When reloading for the first time, the start of the program is removed from the reload memory. If a jump instruction to the beginning of the program is executed, the function is no longer able to find the jump destination. The program is aborted and alarm 14000 is output.

  Note

  To be able to execute external programs without restrictions with regard to the programmed jump instructions, it is recommended to use the option "Execution from External Storage (EES)" instead of the function "Execution from external source".

- The GOTOS command is not available in synchronized actions and technology cycles.

Example

Program code	Comment
N10 ...	; Start of the program.
...	
N90 GOTOS	; Jump to beginning of the program.
...	

See also

Program jumps to jump markers (GOTOB, GOTOF, GOTO, GOTOC) (Page 453)

Program branch (CASE ... OF ... DEFAULT ...) (Page 456)

4.1.5.2 Program jumps to jump markers (GOTOB, GOTOF, GOTO, GOTOC)

Jump labels can be set in a program, which can be jumped to from another location within the same program using the commands GOTOF, GOTOB, GOTO, or GOTOC. Program execution is resumed with the statement that immediately follows the jump label. This means that branches can be realized within the program.

In addition to jump labels, main and sub-block numbers are possible as jump designation.

If a jump condition (IF ...) is formulated before the jump statement, the program jump is only executed if the jump condition is fulfilled.

Syntax

```
GOTOB <jump destination>
IF <jump condition> == TRUE GOTOB <jump destination>

GOTOF <jump destination>
IF <jump condition> == TRUE GOTOF <jump destination>
```

```
GOTO <jump destination>
IF <jump condition> == TRUE GOTO <jump destination>

GOTOC <jump destination>
IF <jump condition> == TRUE GOTOC <jump destination>
```

Meaning

`GOTOB`:	Jump statement with jump destination toward the beginning of the program.	
`GOTOF`:	Jump statement with jump destination toward the end of the program.	
`GOTO`:	Jump statement with jump destination search. The search is first made in the direction of the end of the program, then in the direction of the beginning of the program.	
`GOTOC`:	Same effect as for GOTO with the difference that Alarm 14080 "Jump designation not found" is suppressed. This means that program execution is not interrupted in the case that the jump destination search is unsuccessful – but is continued with the program line following the GOTOC command.	
`<jump destination>`:	Jump destination parameter Possible data include:	
	<jump label>:	Jump destination is the jump label set in the program with a user-defined name:`<jump label>:`
	<block number>:	Jump destination is main block or sub-block number (e.g.: `200`, `N300`)
	STRING type variable:	Variable jump destination. The variable stands for a jump label or a block number.
`IF`:	Keyword to formulate the jump condition. The jump condition permits all comparison and logical operations (result: TRUE or FALSE). The program jump is executed if the result of this operation is TRUE.	

Note

Jump labels

Jump labels are always located at the beginning of a block. If a program number exists, the jump label is located immediately after the block number.

The following rules apply when naming jump labels:

- Number of characters:
  - Minimum 2
  - Maximum 32
- Permissible characters are:
  - Letters
  - Numbers
  - Underscores
- The first two characters must be letters or underscores.
- The name of the jump label is followed by a colon (":").

Constraints

- The jump destination can only be a block with jump label or block number that is located **within** the program.
- A jump statement without jump condition must be programmed in a separate block. This restriction does not apply to jump statements with jump conditions. In this case, several jump statements can be formulated in a block.
- For programs with jump instructions without jump conditions, the end of the program M2/M30 does not necessarily have to be at the end of the program.
- If programs with jump instructions are to be executed from an external program memory via the "Execution from external source" function, the jump destinations must be located within the reload memory. Otherwise the jump destination is not found and the program aborts and alarm 14000 is output.

Note

To be able to execute external programs without restrictions with regard to the programmed jump instructions, it is recommended to use the option "Execution from External Storage (EES)" instead of the function "Execution from external source".

Examples

Example 1: Jumps to jump labels

Program code	Comment
N10 …	
N20 GOTOF Label_1	; Jump toward end of program to
	; jump label "Label_1".
N30 …	
N40 Label_0: R1=R2+R3	; Jump label "Label_0" set.
N50 …	
N60 Label_1:	; Jump label "Label_1" set.
N70 …	
N80 GOTOB Label_0	; Jump toward beginning of program
	; to the jump label "Label_0".
N90 …	

Example 2: Indirect jump to the block number

Program code	Comment
IF <condition> == TRUE	
R10=100	; Assign jump destination
ELSE	
R10=110	; Assign jump destination
ENDIF	
; Jump toward end of program to the block whose block number is located in R10	
N10 GOTOF "N"<<R10	
...	

Program code	Comment
N90 ...	
N100 ...	; Jump destination
N110 ...	
...	

Example 3: Jump to variable jump destination

Program code	Comment
DEF STRING[20] DESTINATION	
IF <condition> == TRUE	
DESTINATION = "Label1"	; Assign jump destination
ELSE	
DESTINATION = "Label2"	; Assign jump destination
ENDIF	
; Jump toward end of program to the variable jump destination "Content of DESTINATION."	
GOTOF DESTINATION	
Label1: T="Drill1"	; Jump destination 1
...	
Label2: T="Drill2"	; Jump destination 2
...	

Example 4: Jump with jump condition

Program code	Comment
N40 R1=30 R2=60 R3=10 R4=11 R5=50 R6=20	; Assignment of the initial values
N41 LA1: G0 X=R2*COS(R1)+R5 Y=R2*SIN(R1)+R6	; Jump label LA1
N42 R1=R1+R3 R4=R4-1	
; IF jump condition == TRUE	
; THEN jump toward beginning of program to the jump label LA1	
N43 IF R4>0 GOTOB LA1	
N44 M30	; End of program

4.1.5.3 Program branch (CASE ... OF ... DEFAULT ...)

The CASE function provides the possibility of checking the actual value (type: INT) of a variable or an arithmetic function and, depending on the result, to jump to different positions in the program.

Syntax

```
CASE(<expression>) OF <constant_1> GOTOF <jump destination_1>
<constant_2> GOTOF <jump destination_2> ... DEFAULT GOTOF <jump
destination_n>
```

Meaning

`CASE`:	Jump instruction
`<expression>`:	Variable or arithmetic function
`OF`:	Keyword to formulate conditional program branches
`<constant_1>`:	First specified constant value for the variable or arithmetic function Type: INT
`<constant_2>`:	Second specified constant value for the variable or arithmetic function Type: INT
`DEFAULT`:	For the cases where the variable or arithmetic function does not assume any of the specified constant values, the DEFAULT statement can be used to determine the jump destination. **Note:** If the DEFAULT statement is not programmed, then in these cases, the block following the CASE statement is the jump destination.
`GOTOF`:	Jump statement with jump destination toward the end of the program. Instead of `GOTOF` all other GOTO commands can be programmed (refer to the subject "Program jumps to jump markers").
`<jump destination_1>`:	A branch is made to this jump destination if the value of the variable or arithmetic function corresponds to the first specific constant. The jump destination can be specified as follows: <jump label>: Jump destination is the jump label set in the program with a user-defined name:`<jump label>:` <block number>: Jump destination is main block or sub-block number (e.g.: `200`, `N300`) STRING type variable: Variable jump destination. The variable stands for a jump label or a block number.
`<jump destination_2>`:	A branch is made to this jump destination if the value of the variable or arithmetic function corresponds to the second specified constant.
`<jump destination_n>`:	A branch is made to this jump destination if the value of the variable does not assume the specified constant value.

Constraints

Process from external source

If programs with the CASE function are to be executed from an external program memory via the "Execution from external source" function, the jump destinations must be located within the reload memory. Otherwise the jump destination is not found and the program aborts and alarm 14000 is output.

Note

To be able to execute external programs without restrictions with regard to the programmed jump instructions, it is recommended to use the option "Execution from External Storage (EES)" instead of the function "Execution from external source".

Example

Program code
...
N20 DEF INT VAR1 VAR2 VAR3
N30 CASE(VAR1+VAR2-VAR3) OF 7 GOTOF Label_1 9 GOTOF Label_2 DEFAULT GOTOF Label_3
N40 Label_1: G0 X1 Y1
N50 Label_2: G0 X2 Y2
N60 Label_3: G0 X3 Y3
...

The CASE statement from N30 defines the following program branch possibilities:

1. If the value of the arithmetic function VAR1+VAR2-VAR3 = 7, then jump to the block with the jump marker definition "Label_1" (→ N40).
2. If the value of the arithmetic function VAR1+VAR2-VAR3 = 9, then jump to the block with the jump marker definition "Label_2" (→ N50).
3. If the value of the arithmetic function VAR1+VAR2-VAR3 is neither 7 nor 9, then jump to the block with the jump marker definition "Label_3" (→ N60).

4.1.6 Repeat program section (REPEAT, REPEATB, ENDLABEL, P)

Program section repetition allows you to repeat existing program sections within a program in any order. The program sections are called with the keywords REPEAT or REPEATB. The program lines or program sections to be repeated are identified by labels. The number of repeats is programmable.

The program line containing the jump label can be before or after the REPEAT/REPEATB statement. The search initially commences towards the start of the program. If the label is not found in this direction, the search continues working toward the end of the program.

If the program line containing the jump label contains further operations, these are executed again on each repetition.

Syntax

Repeat individual program line:

```
<Label>: ...
...
REPEATB <Label> P=<n>
...
```

Repeat program section between label and REPEAT statement:

```
<Label>: ...
...
REPEAT <Label> P=<n>
```

```
...
```

Note

If the program section between the jump label and the REPEAT statement needs to be repeated, the program line with the jump label must be **before** the REPEAT statement because, in this case, the search runs **only** toward the beginning of the program.

---

3. Repeat section between two labels:

```
<Label_1>: ...
...
<Label_2>: ...
...
REPEAT <Label_1> <Label_2> P=<n>
...
```

Note

It is not possible to nest the REPEAT statement with the two jump labels within parentheses. If the first jump label is found before the REPEAT statement and the second jump label is not reached before the REPEAT statement, the repetition is performed between the first jump label and the REPEAT statement.

---

4. Repeat section between label and ENDLABEL:

```
<Label>: ...
...
ENDLABEL: ...
...
REPEAT <Label> P=<n>
...
```

Note

It is not possible to bracket the REPEAT statement with the jump label and the ENDLABEL. If the jump label is found before the REPEAT statement and ENDLABEL is not reached before the REPEAT statement, the repetition is performed between the jump label and the REPEAT statement.

Meaning

`REPEATB`	Command for repeating a program line
`REPEAT`	Command for repeating a program section

<table>
<tr><td><Label></td><td colspan="2">Jump label
The jump label denotes either the program line to be repeated (in the case of REPEATB) or the start of the program section to be repeated (in the case of REPEAT).
Jump labels are always located at the beginning of a block. If a block number exists, the jump label is located immediately after the block number.
The following rules apply when naming jump labels:
• Number of characters:
  – Minimum 2
  – Maximum 32
• Permissible characters are:
  – Letters
  – Numbers
  – Underscores
• The first two characters must be letters or underscores.
• The name of the jump label is followed by a colon (":").</td></tr>
<tr><td rowspan="3"><Label_1>
<Label_2></td><td colspan="2">Jump labels that identify the program section to be repeated (in the case of REPEAT) if the repetition is not to extend as far as the REPEAT statement</td></tr>
<tr><td><Label_1></td><td>The first jump label marks the start of a program section to be repeated.</td></tr>
<tr><td><Label_2></td><td>The second jump label marks the end of a program section to be repeated.</td></tr>
<tr><td>ENDLABEL</td><td colspan="2">The end of a program section to be repeated can also be identified with the keyword ENDLABEL.
If the program line with ENDLABEL contains further operations, these are executed again on each repetition.
ENDLABEL can be used more than once in the program.</td></tr>
<tr><td>P</td><td colspan="2">Address for specifying the number of repetitions</td></tr>
<tr><td rowspan="3"><n></td><td colspan="2">Number of program section repetitions</td></tr>
<tr><td>Type:</td><td>INT</td></tr>
<tr><td colspan="2">The program section to be repeated is repeated <n> times. After the last repetition, the program is continued at the line following the REPEAT/REPEATB line.
Note:
If no number is specified for P=<n>, the program section is repeated just once.</td></tr>
</table>

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 ...`	

Program code	Comment
N70 M30	

Example 2: Repeat program section between label and REPEAT statement:

Program code	Comment
N5 R10=15	
N10 Begin: R10=R10+1	;Width
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	;Width
N20 Z=10-R10	
N30 G1 X=R10 F200	
N40 Y=R10	
N50 X=-R10	
N60 Y=-R10	
N70 END: Z=10	
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:	
N30 X10	
N40 Y10	
N50 BEGIN2:	
N60 X20	
N70 Y30	
N80 ENDLABEL: Z10	
N90 X0 Y0 Z0	

Program code	Comment
N100 Z-10	
N110 BEGIN3: X20	
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:	; Drilling positions 1
N30 X1 Y1	
N40 X2	
N50 Y2	
N60 X3 Y3	
N70 ENDLABEL:	
N80 POS_2:	; Drilling positions 2
N90 X10 Y5	
N100 X9 Y-5	
N110 X3 Y3	
N120 ENDLABEL:	
N130 DRILL()	; Change drill and drilling cycle.
N140 THREAD(6)	; Load tap M6 and threading cycle.
N150 REPEAT POS_1	; Repeat program section once from POS_1 up to ENDLABEL.
N160 DRILL()	; Change drill and drilling cycle.
N170 THREAD(8)	; Load tap M8 and threading cycle.
N180 REPEAT POS_2	; Repeat program section once from POS_2 up to ENDLABEL.
N190 M30	

More information

Nesting

Program section repetitions can be nested. Each call uses a subprogram level.

M17 / RET

If `M17` or `RET` is programmed during processing of a program section repetition, the repetition is canceled. The program is resumed at the block following the `REPEAT` line.

Program display

In the current program display, the program section repetition is displayed as a separate subprogram level.

Cancel level

If the level is canceled during the program section repetition, 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	

Control structures and program section repetition

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.

Jumps and 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

Program code
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 instruction should be placed behind the traversing blocks.

Execution from external source and program section repetition

For external programs with program section repetition, the program line to be repeated (for REPEATB) or the start of the program section to be repeated (for REPEAT) must be within the reload memory. Otherwise the jump destination is not found and the program aborts and alarm 14000 is output.

Note

To be able to execute external programs without restrictions with regard to program section repetitions, it is recommended to use the function "Execution from External Storage (EES)" instead of the function "Execution from external source".

Variable jump destinations

Variable jump destinations like those used inGOTO commands (Page 453) are **not** permissible for program section repetition with REPEAT/REPEATB.

4.1.7 Check structures

The control processes the NC blocks as standard in the programmed sequence.

This sequence can be variable by programming alternative program blocks and program loops. These check structures are programmed using the key words IF, ELSE, ENDIF, LOOP, FOR, WHILE and REPEAT.

NOTICE

Programming error

Check structures may only be inserted in the statement section of a program. Definitions in the program header may not be executed conditionally or repeatedly.

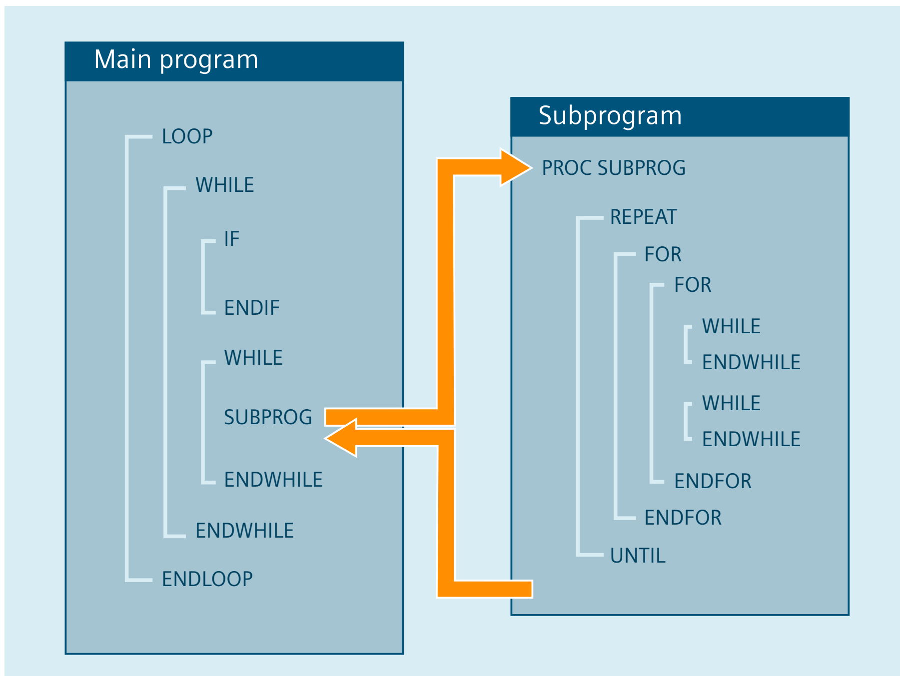
It is not permissible to superimpose macros on keywords for control structures or on jump destinations. No such check is made when the macro is defined.

Effectiveness

The check structure cannot be used program-wide.

Nesting depth

A nesting depth of up to 16 check structures can be set up on each subprogram level.



Runtime response

In interpreter mode (active as standard), it is possible to shorten program processing times more effectively by using program branches than can be obtained with check structures.

There is no difference between program branches and check structures in precompiled cycles.

Current block display for program loops

If only selected blocks are executed within a program loop, the last main run block **before** the program loop is shown in the current block display.

So that the processed selected blocks are also visible in the current block display, e.g. for diagnostic purposes, the decoding single block SBL2 must be activated.

Grinding without main run block

If, within a program loop, no main run block has been programmed, then the loop is pre-processed until the loop condition is satisfied.

As a consequence, a high level of utilization can occur and this can have a negative impact on the display.

The STOPRE command or a dwell time G04 of 0 seconds can be inserted **in** the loop as countermeasure.

Constraints

Block display

Blocks with check structure elements cannot be suppressed.

Control structures and program jumps

Jumper markers (labels) are not permitted in blocks with check structure elements.

Note

It is not generally advisable to use a mixture of check structures and program branches.

Process from external source

For external programs with control structures, the start of the loop must be within the reload memory. Otherwise the jump destination is not found and the program aborts and alarm 14000 is output.

Note

To be able to execute external programs without restrictions with regard to control structures, it is recommended to use the function "Execution from External Storage (EES)" instead of the function "Execution from external source".

Interpreter mode

Check structures are processed interpretively. When a loop end is detected, a search is made for the loop beginning, allowing for the check structures found in the process. For this reason, the block structure of a program is not checked completely in interpreter mode.

Preprocessing of cycles

A check can be made to ensure that check structures are nested correctly when cycles are preprocessed.

4.1.7.1 Conditional statement and branch (IF, ELSE, ENDIF)

Conditional statement: IF - program block - ENDIF

With a conditional statement, the program block between `IF` and `ENDIF` is only executed when the condition is satisfied.

Branch: IF - program block_1 - ELSE - program block_2 - ENDIF

With a branch, one of two program blocks is always executed.

If the condition is satisfied, program block_1 between `IF` and `ELSE` is executed.

If the condition is **not** satisfied, program block_2 between `ELSE` and `ENDIF` is executed.

Note

ELSE in synchronized actions

The keyword ELSE can also be programmed in synchronized actions. Thus a synchronized action can be expanded by actions that are to be executed if the condition is not fulfilled.

Syntax

Conditional statement

```
IF <condition>
   Program block                          ; Execution with: <Condition> == TRUE
ENDIF
```

Branch

```
IF <condition>
   Program block_1                        ; Execution with: <Condition> == TRUE
ELSE
   Program block_2                        ; Execution with: <Condition> == FALSE
ENDIF
```

Meaning

`IF`:	Introduces the conditional statement or branch.
`ELSE`:	Introduces the alternative program block.
`ENDIF`:	Marks the end of the conditional statement or branch.
`<condition>`:	Logical expression that is evaluated as TRUE or FALSE.

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`	
`N530 IF $P_ISTEST`	In the program test mode ...
`N540    TNR_AKTUELL = $P_TOOLNO`	... the "current" tool is read from the program context.
`N550 ELSE`	Otherwise ...
`N560    TNR_AKTUELL = $TC_MPP6[9998,1]`	... the tool of the spindle is read-out.

Program code	Comment
N570 ENDIF	
N580 GETSELT(TNR_VORWAHL)	Read the T number of the pre-selected tool in the spindle.
N590 IF TNR_AKTUELL <> TNR_VORWAHL	If the pre-selected tool is still not the current tool, then ...
N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0	... Approach tool change position ...
N610 M206	... and perform a tool change.
N620 ENDIF	
N630 M17	

4.1.7.2 Continuous program loop (LOOP, ENDLOOP)

Endless loops are used in endless programs. At the end of the loop, there is always a branch back to the beginning.

Syntax

```
LOOP
...
ENDLOOP
```

Meaning

LOOP:	Initiates the endless loop.
ENDLOOP:	Marks the end of the loop and results in a return jump to the beginning of the loop.

Example

```
Program code
...
LOOP
MSG ("no tool cutting edge active")
M0
STOPRE
ENDLOOP
...
```

4.1.7.3 Count loop (FOR ... TO ..., ENDFOR)

The count loop is used if an operation must be repeated with a fixed number of runs.

Syntax

```
FOR <variable> = <initial value> TO <end value>
...
ENDFOR
```

Meaning

`FOR`:	Initiates the count loop.	
`ENDFOR`:	Marks the end of the loop and results in a return jump to the beginning of the loop, as long as the end value of the count has still not been reached.	
`<variable>`:	Count variable, which is incremented from the initial to the end value and is increased by the value "1" at each run.	
	Type	INT or REAL **Note:** The REAL type is used if R parameters are programmed for a count loop, for example. If the count variable is of the REAL type, its value is rounded to an integer.
`<initial value>`:	Initial value of the count Condition: The start value must be lower than the end value.	
`<full-scale value>`:	End value of the count	

Examples

Example 1: INTEGER variable or R parameter as count variable

INTEGER variable as count variable:

Program code	Comment
`DEF INT iVARIABLE1`	
`R10=R12-R20*R1 R11=6`	
`FOR iVARIABLE1 = R10 TO R11`	`; Count variable = INTEGER variable`
` R20=R21*R22+R33`	
`ENDFOR`	
`M30`	

R parameter as count variable:

Program code	Comment
`R11=6`	
`FOR R10=R12-R20*R1 TO R11`	`; Count variable = R parameter (real variable)`
` R20=R21*R22+R33`	
`ENDFOR`	
`M30`	

Example 2: Production of a fixed quantity of parts

Program code	Comment
DEF INT WKPCCOUNT	; Defines type INT variable with the name "WKPCCOUNT".
FOR WKPCCOUNT = 0 TO 100	; Initiates the count loop. The "WKPCCOUNT" variable increments from the initial value "0" to the end value "100".
G01 …	
ENDFOR	; End of count loop
M30	

4.1.7.4 Program loop with condition at start of loop (WHILE, ENDWHILE)

For a WHILE loop, the condition is at the beginning of the loop. The WHILE loop is executed as long as the condition is fulfilled.

Syntax

```
WHILE <condition>
...
ENDWHILE
```

Meaning

WHILE:	Initiates the program loop.
ENDWHILE:	Marks the end of the loop and results in a return jump to the beginning of the loop.
<condition>:	The condition must be fulfilled so that the WHILE loop is executed.

Example

Program code	Comment
...	
WHILE $AA_IW[DRILL_AXIS] > -10	; Call the WHILE loop under the following condition: The actual WCS setpoint for the drilling axis must be greater than -10.
G1 G91 F250 AX[DRILL_AXIS] = -1	
ENDWHILE	
...	

4.1.7.5 Program loop with condition at the end of the loop (REPEAT, UNTIL)

For a REPEAT loop, the condition is at the end of the loop. The REPEAT loop is executed once and repeated continuously until the condition is fulfilled.

Syntax

```
REPEAT
...
UNTIL <significance>
```

Meaning

REPEAT:	Initiates the program loop.
UNTIL:	Marks the end of the loop and results in a return jump to the beginning of the loop.
<condition>:	The condition that must be fulfilled so that the REPEAT loop is no longer executed.

Example

```
Program code                 Comment
...
REPEAT                       ; Call the REPEAT loop.
...
UNTIL ...                    ; Check whether the condition is fulfilled.
...
```

4.1.7.6 Program example with nested check structures

```
Program code                                  Comment
LOOP
IF NOT $P_SEARCH                              ; IF no block search
      G1 G90 X0 Z10 F1000
      WHILE $AA_IM[X] <= 100                  ; WHILE (setpoint X axis <= 100)
         G1 G91 X10 F500                      ; Drilling pattern
         Z-5 F100
         Z5
      ENDWHILE
   ELSE                                       ; ELSE block search
      MSG("No drilling during block search")
   ENDIF                                      ; ENDIF
   $A_OUT[1] = 1                              ; Next drilling plate
   G4 F2
ENDLOOP
M30
```

4.1.8 Cross-channel program coordination (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)

In principle, a channel of the NC can execute the program started in it independently of other channels in its mode group. If, however, several programs in several channels of the mode group are involved in machining a workpiece, the program sequences in the various channels must be coordinated with the following coordination commands.

Requirement

All the channels involved in the program coordination must belong to the **same** mode group:

MD10010 $MC_ASSIGN_CHAN_TO_MODE_GROUP[<Channel>] = <Mode group number>

Channel name instead of channel number

Instead of the channel numbers, the channel names entered in MD20000 $MC_CHAN_NAME[<Channel index>] can be used as parameters of the predefined procedures for the program coordination. The use of the channel names in the NC programs first has to be enabled:

MD10280 $MN_PROG_FUNCTION_MASK, bit 1 = TRUE

Note

Minimum distance between commands

At least two traversing block distances must be maintained between the commands `INIT`, `START`, `WAITE`, `WAITM`, `SETM`, `CLEARM` and the command `WAITMC`. WAITMC is an executable block, but is moved into the previous block for optimization, and then deleted as a block. `SETM` for example is not an executable block, and is moved into the next block so that if there were a distance of one block between two commands, both commands would be in the middle block. As only one block is possible, optimization is not performed with a one block distance for `WAITMC`.

This stops the program, and processing is briefly interrupted.

Syntax

```
INIT(<ChanNo>, <Prog>, <AckMode>)
START(<ChanNo>, <ChanNo>, ...)
WAITM(<MarkNo>, <ChanNo>, <ChanNo>, ...)
WAITE(<ChanNo>, <ChanNo>, ...)
WAITMC(<MarkNo>, <ChanNo>, <ChanNo>, ...)
SETM(<MarkNo>, <MarkNo>, ...)
CLEARM(<MarkNo>, <MarkNo>, ...)
```

Meaning

`INIT():`	Predefined procedure for selecting the NC program that is to be executed in the specified channel
`START():`	Predefined procedure for starting the selected program in the respective channel

`WAITM()`:	Predefined procedure to wait for a wait marker to be reached in the specified channels The specified wait marker is set by `WAITM` in the same channel. The previous block is terminated with exact stop. The wait marker is deleted after synchronization. A maximum of 10 markers can be set simultaneously in each channel.
`WAITE()`:	Predefined procedure to wait for the end of program in one or more other channels
`WAITMC()`:[1)]	Predefined procedure to wait for a wait marker to be reached in the specified channels In contrast to `WAITM`, the braking of the axes to exact stop is only initiated if the other channels have not yet reached the wait marker.
`SETM()`:[1)]	Predefined procedure to set one or more wait markers for the channel coordination The processing in own channel is not affected by this. `SETM` remains valid after a channel reset and NC start.
`CLEARM()`:[1)]	Predefined procedure to delete one or more wait markers for the channel coordination The processing in own channel is not affected by this. `CLEARM()` deletes all wait markers in the channel. `CLEARM(0)` only deletes wait marker "0". `CLEARM` remains valid after a channel reset and NC start.
`<ChanNo>`:	Channel number The number of the own channel does not have to be specified. Type: INT
`<Prog>`:	Absolute or relative path specification (**optional**) + program name Type: STRING Information relating to path data, see the Programming Manual NC Programming, Chapter "Addressing program memory files" (Page 544).
`<AckMode>`:	Acknowledgment mode (**optional**) Type: CHAR Values: "N": Without acknowledgment The program execution is continued after the command has been sent. The sender is not informed if the command cannot be executed successfully. "S": Synchronous acknowledgment The program execution is stopped until the receiving component has acknowledged the command. If the acknowledgment is positive, the next command is executed. If the acknowledgment is negative, an error message is output.
`<MarkNo>`:	Number of the wait marker **Note** In a multi-channel system, a maximum of 100 wait markers are available (wait markers 0 ... 99). Only wait marker 0 is available in a single-channel system.
1) For user-specific communication and/or coordination of channels, wait markers can be deployed using `SETM` / `CLEARM` – and also without using the conditional wait command `WAITMC`. The wait markers retain their values, even after a channel reset and NC start.	

Examples

START using channel names from MD20000

- Parameter assignment
  `MD10280 $MN_PROG_FUNCTION_MASK, bit 1 = TRUE`
  `$MC_CHAN_NAME[ 0 ] = "MACHINING"; Name of channel 1`
  `$MC_CHAN_NAME[ 1 ] = "INFEED"; Name of channel 2`
- Programming

Program code	Comment
START(MACHINING)	; Start of channel 1
START(INFEED)	; Start of channel 2

START using local "channel names" and user variables

Program code	Comment
DEF INT MACHINE = 1	; Definition of user variable for channel 1
DEF INT LOADER = 2	; Definition of user variable for channel 2
...	
START(MACHINE)	; Start of channel 1
START(LOADER)	; Start of channel 2

START using local "channel names", user variables and parameterized channel names

Program code	Comment
DEF INT chanNo1	; Definition of user variable for channel 1
DEF INT chanNo2	; Definition of user variable for channel 2
chanNo1 = CHAN_1	; Assignment of parameterized channel name channel 1
chanNo2 = CHAN_2	; Assignment of parameterized channel name channel 2
...	
START(ChanNo1)	; Start of channel 1
START(ChanNo2)	; Start of channel 2

INIT command with absolute path specification

Selection of program /_N_MPF_DIR/_N_ABSPAN1_MPF in channel 2.

Program code
```
INIT(2,"/_N_WCS_DIR/_N_SHAFT1_WPD/_N_CUT1_MPF")
```

INIT command with program name

Selection of the program with the name "MYPROG". The control searches for the program using the search path.

Program code
```
INIT(2,"MYPROG")
```

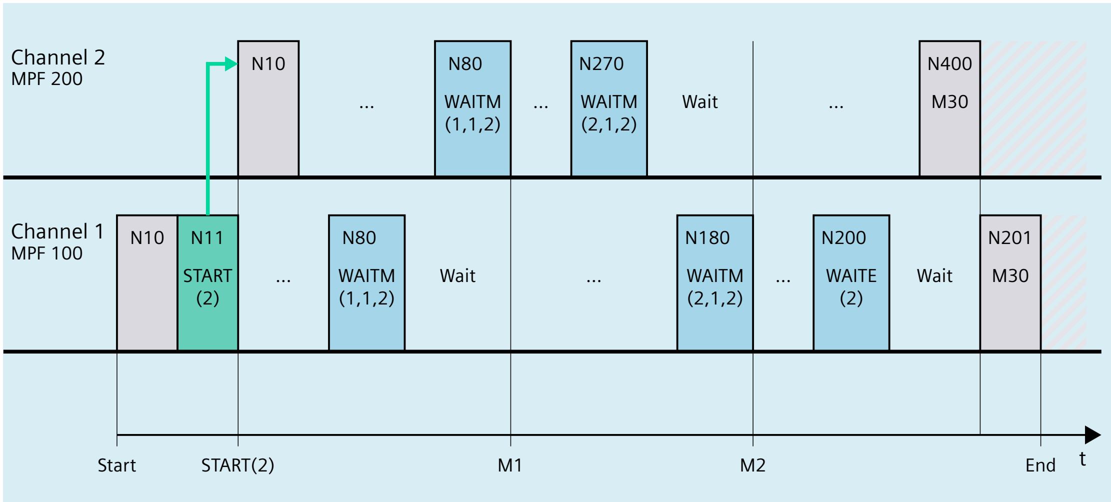
Program coordination with WAITM

- Channel 1: The program /_N_MPF_DIR/_N_MPF100_MPF has already been selected and started.

Program code	Comment
	; Program MPF100
N10 INIT(2,"MPF200","N")	; Selection of program MPF200, channel 2
N11 START(2)	; Start of channel 2
...	
N80 WAITM(1,1,2)	; Wait for WAIT marker 1 in channels 1 and 2
N81 ...	; Channel 1, N81 and channel 2, N71 are ; started synchronously
...	
N180 WAITM(2,1,2)	; Wait for WAIT marker 2 in channels 1 and 2
N181 ...	; Channel 1, N181 and channel 2, N271 are ; started synchronously
...	
N200 WAITE(2)	; Wait for end of program in channel 2
N201 ...	; N201 is not started until the end of program ; MPF200 started in channel 2
N201 M30	; End of program channel 1

- Channel 2: In channel 1, the program MPF200_MPF is selected and started for channel 2 using blocks N10 and N20.

Program code	Comment
;$PATH=/_N_MPF_DIR	; Program MPF200
...	
N70 WAITM(1,1,2)	Wait for WAIT marker 1 in channels 1 and 2
N71 ...	; Channel 1, N81 and channel 2, N71 are ; started synchronously
...	
N270 WAITM(2,1,2)	Wait for WAIT marker 2 in channels 1 and 2
N271 ...	; Channel 1, N181 and channel 2, N271 are ; started synchronously
...	
N400 M30	End of program channel 2



Supplementary conditions

Non-synchronous start of execution of following blocks after WAIT markers

In the case of channel coordination using WAIT markers, execution of the following blocks may start non-synchronously. This behavior occurs if an action is triggered in one of the channels to be synchronized immediately before reaching the common WAIT marker; the consequence of which is implicit repositioning (REPOSA) in this delete distance-to-go.

Assumption: Current axis assignment in channels 1 and 2

- Channel 1: Axes X1 and U
- Channel 2: Axis X2

Table 4-1 Time sequence in channels 1 and 2

Channel 1	Channel 2	Description
. . .	. . .	Arbitrary processing in channels 1 and 2
`N100 WAITM(20,1,2)`		Channel 1: reaches the `WAIT` marker and waits for synchronization with channel 2
Start of the GETD(U) processing: • *Axis interchange* • *Delete distance-to-go* • *REPOSA* *End*	`N200 GETD(U)`	Channel 2: Requests axis U from channel 1 Channel 1: Processing of `GET(U)` in the background
	`N210 WAITM(20,1,2)`	Channel 2: reaches the `WAIT` marker. ⇒ This completes the synchronization of channels 1 and 2
	`N220 G0 X2=100`	Channel 2: Start of processing of N220
`N110 G0 X1=100`		Channel 1: Staggered start of processing of N110

4.1.9 Macro technique (DEFINE ... AS)

NOTICE
Macro technology increases the complexity of the programming
Macros can significantly alter the control's programming language. Macro technology may only be used with great care.

A macro is a sequence of individual statements which have together been assigned a name of their own. When a macro is called during a program run, the statements programmed under the program name are executed one after the other.

Macro types

According to the range of validity (in other words, the range in which the macro definition is active), there are the following macro categories:

- Local macros
  Local macros are macros that are defined at the beginning of an NC program, which at the time of execution is not the main program. They are created when the NC program is called, and deleted with an end of program reset – or the next time that the control system powers up. Local macros can only be accessed within the NC program in which they are defined.
- Program-global macros
  Program-global macros are macros that are defined at the beginning of an NC program, which is used as a main program. They are created when the NC program is called, and deleted with an end of program reset – or the next time that the control system powers up. Program-global macros can be accessed in the main program and in all subprograms.

  Note

  Availability of program-global macros

  Program-global macros defined in the main program are only available in subprograms if the following machine data is set:

  MD11120 $MN_LUD_EXTENDED_SCOPE = 1

  If MD11120 = 0, the program-global macros defined in the main program will only be available in the main program.

- Global macros
  Global macros are NC or channel-global macros, which are defined in a definition file (macro file) – and are retained even after an end of program reset or the next time that the control system powers up. Global macros can be called in any main program or subprogram and executed.

  Note

  In order to use the macros of an **external** macro file in the NC program, the macro file must be downloaded to the NC.

Macro definition rules

Macros must be defined before they can be used. The following rules must be observed in this context:

- Any identifier, G, M, H functions and L subprogram names can be defined in a macro.
- The macro can be defined at the beginning of the program or in a dedicated definition file (macro file).
- Local and program-global macros are defined at the beginning of the program.
- Global macros must be defined in a macro file, e.g._N_DEF_DIR/_N_UMAC_DEF.
- G command macros can only be defined as global macros.
- H and L functions can be programmed with 2 digits.
- M and G commands can be programmed with 3 digits.

Nesting of macros

It is possible to program macros within macros. This is used for the following purposes, for example:

- To increase the flexibility of programming.
- To implement a simpler cycle programming.

A maximum of 3 macro levels are possible within a nesting.

Syntax

Macro definition without nesting:
`DEFINE <Macro_name> AS <Operation_1> <Operation_2> ...`

Macro definition with nesting:

```
DEFINE <Macro_Name1> AS <Operation_1> <Operation_2> ...
DEFINE <Macro_Name2> AS <Macro_Name1> <Operation_1> ...
DEFINE <Macro_Name3> AS <Macro_Name2> <Operation_2> ...
```

Call in the NC program:
`<Macro_name>`

Meaning

DEFINE ... AS	Keyword combination to define a macro
<Macro_name>	Macro name Only identifiers are permissible as macro names. The macro is called from the NC program by the macro name.
<Operation_1>	First programming instruction in the macro
<Operation_2>	Second programming instruction in the macro

Note

Keywords and reserved names may not be overwritten by macros. This also applies to all jump destinations within a GOTO command, and to the keywords in program loops, such as FOR, WHILE, LOOP, REPEAT.

Examples

Example 1: Macro definition at the beginning of the program

Program code	Comment
DEFINE LINE AS G1 G94 F300	; Macro definition
...	
N70 LINE X10 Y20	; Macro call
...	

Example 2: Macro definitions in a macro file

Program code	Comment
DEFINE M6 AS L6	; A subprogram is called at tool change to handle the necessary data transfer. The actual tool change M function is output in the subprogram (e.g. M106).
DEFINE G81 AS DRILL(81)	; Emulation of the DIN-G command.
DEFINE G33 AS M333 G333	; During thread cutting, synchronization is requested with the PLC. The original G command G33 was renamed to G333 by machine data so that the programming remains identical for the user.

Example 3: External macro file

After reading the external macro file into the control, the macro file must be downloaded into the NC. Only then can macros be used in the NC program.

Program code	Comment
%_N_UMAC_DEF	
;$PATH=/_N_DEF_DIR	; Customer-specific macros
DEFINE PI AS 3.14	
DEFINE TC1 AS M3 S1000	
DEFINE M13 AS M3 M7	; Spindle clockwise, coolant on
DEFINE M14 AS M4 M7	; Spindle counter-clockwise, coolant on
DEFINE M15 AS M5 M9	; Spindle stop, coolant off
DEFINE M6 AS L6	; Call tool change program
DEFINE G80 AS MCALL	; Deselect drilling cycle
M30	

Example 4: Nesting of macros

```
Program code
%_N_MGUD_DEF
;$PATH=/_N_DEF_DIR
DEFINE _TOP_SURFACE_SMOOTH_ON AS 10
DEFINE ROUGH AS CYCLE832( 0.1,_TOP_SURFACE_SMOOTH_ON,1)
...
```

After the macro file has been loaded into the controller, the NC program is processed. With the ROUGH call, CYCLE832 starts as a nested macro.

4.2 Subprogram technique

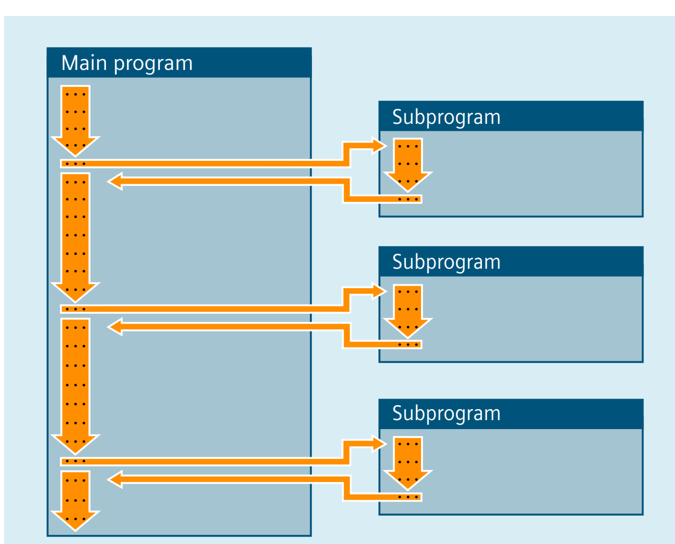
4.2.1 Fundamentals

4.2.1.1 Subprogram

The term "subprogram" has its origins during the time when part programs were split strictly into main and subprograms. Main programs were the part programs selected for processing on the control and then launched. Subprograms were the part programs called from within the main program.

This strict division no longer exists with today's SINUMERIK NC language. In principle, each part program can be selected as a main program and launched or called from another part program as a subprogram.

Accordingly, the subprogram can then be used to refer to a part program called from within another part program.



Application

As in all high-level programming languages, in the NC language, subprograms swaps out program sections used more than once to independent, self-contained programs.

Subprograms have the following advantages:

- Better transparency and readability of programs
- Higher quality due to reuse of tested program parts
- Possibility of creating specific machining libraries
- More efficient use of memory space

4.2.1.2 Subprogram names

Naming rules

The subprogram name can be chosen freely providing the following rules are observed:

- Permissible characters:
  - Letters: A ... Z, a ... z
  - Numbers: 0 ... 9
  - Underscore: _
- The first two characters should either be two letters or an underscore followed by a letter.

  Note

  If this condition is satisfied, then an NC program can be called as subprogram from another program just by specifying the program name. However, if the program name starts with digits, the subprogram call is then only possible via the CALL statement.

- Maximum length: 24 characters

Note

Uppercase/lowercase letters

The SINUMERIK NC language does **not** distinguish between uppercase and lowercase letters.

Note

Impermissible program names

To avoid problems with Microsoft Windows applications, the following program names may **not** be used:

- CON, PRN, AUX, NUL
- COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9
- LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9

Control-internal extensions

The program name assigned when the subprogram is created is expanded within the control with the addition of a prefix and a suffix:

- Prefix: `_N_`
- Postfix: `_SPF`

Using the program name

When using the program name, e.g. in the context of a subprogram call, all combinations of prefix, program name, and suffix are possible.

Example:

The subprogram with the program name `SUB_PROG` can be started using the following identifiers:

1. `SUB_PROG`
2. `_N_SUB_PROG`
3. `SUB_PROG_SPF`
4. `_N_SUB_PROG_SPF`

Main programs and subprograms with the same name

If a main program (.MPF) and a subprogram (.SPF) exist with the same program name, the appropriate file extension for the unique identification must be specified when the program name in the NC program is used. Otherwise the program found first in the search path with the specified name is used.

File name and program name

The file name and the name of the program contained in this file must always be identical.

One file therefore always contains one program.

4.2.1.3 Nesting of subprograms

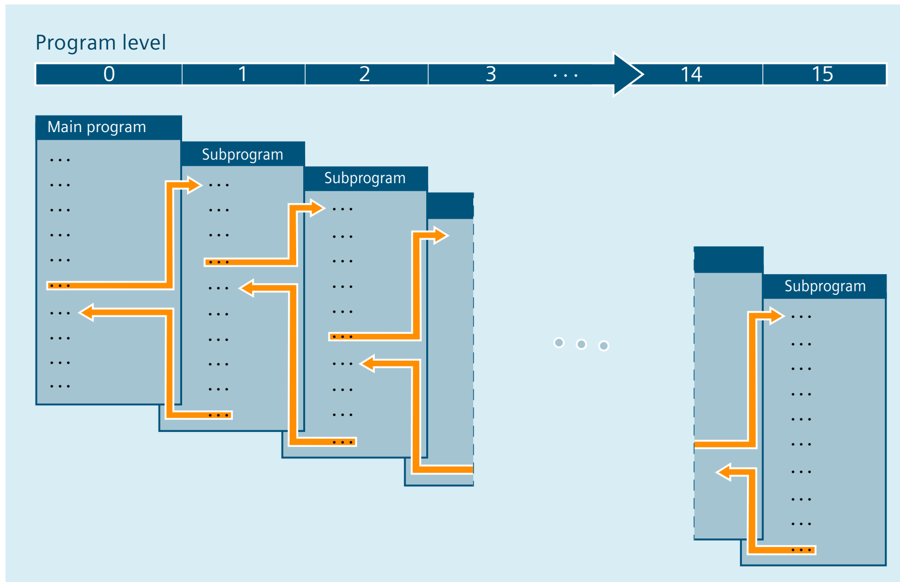
A main program can call subprograms which in turn call more subprograms. As such, the sequences of the programs are nested within each other. Each program runs on a dedicated program level.

Nesting depth

The NC language currently provides 16 program levels. The main program always runs at the uppermost program level, 0. A subprogram always runs at the next lowest program level following the call. Program level 1 is, therefore, the first subprogram level.

Division of program levels:

- Program level 0: Main program level
- Program level 1 to 15: Subprogram level 1 to 15



Interrupt routines (ASUB)

If a subprogram is called in the context of an interrupt routine, this will not be executed at the program level currently active in the channel (n) but at the next lowest program level (n+1). So that this remains possible even at the lowest program level, 2 additional program levels (16 and 17) are available in conjunction with interrupt routines.

If more than 2 program levels are required, this has to be taken into account explicitly in the structuring of the part program executed in the channel. In other words, only a maximum of as many program levels may be used in order to leave sufficient program levels available for interrupt processing.

If interrupt processing needs 4 program levels for example, the part program must be structured so that it uses a maximum of up to program level 13. In the event of an interrupt, the 4 program levels it requires (14 to 17) will be available to it.

Siemens cycles

Siemens cycles need 3 program levels. Therefore, a Siemens cycle must be called at the latest in:

- Part program processing: program level 12
- interrupt routine: program level 14

4.2.1.4 Search path

When a subprogram without path details is called, the control system searches the available program memory using a predefined search sequence (see "Search path for subprogram call (Page 549)").

4.2.1.5 Formal and actual parameters

Formal and actual parameters occur in conjunction with the definition and calling of subprograms with parameter transfer.

Formal parameter

When a subprogram is defined, the parameters to be transferred to it (known as the formal parameters) have to be defined with type and parameter name.

The formal parameters define, therefore, the interface of the subprogram.

Example:

Program code	Comment
`PROC CONTOUR (REAL X, REAL Y)`	`; Formal parameters: X and Y, both REAL type`
`N20 X1=X Y1=Y`	`; Traversing of axis X1 to position X and axis Y1 to position Y`
`...`	
`N100 RET`	

Actual parameters

When a subprogram is called, absolute values or variables (known as actual parameters) have to be transferred to it.

As such, the actual parameters assign up-to-date values to the interface of the subprogram when the latter is called.

Example:

Program code	Comment
`N10 DEF REAL WIDTH`	`; Variable definition`
`N20 WIDTH=20.0`	`; Variable assignment`
`N30 CONTOUR(5.5, WIDTH)`	`; Subprogram call with actual parameters: 5.5 and WIDTH`
`...`	
`N100 M30`	

4.2.1.6 Parameter transfer

Definition of a subprogram with parameter transfer

A subprogram with parameter transfer is defined using the `PROC` keyword and a complete list of all the parameters expected by the subprogram.

Call of a subprogram with parameter transfer

When the subprogram is called, not all the parameters defined in the subprogram interface have to be transferred explicitly. If a parameter is omitted, the default value "0" is transferred for it.

So that the parameter sequence can be uniquely identified, however, the commas used as parameter separators always have to be included. The last parameter is an exception. If it is omitted from the call, the last comma can also be left out.

Example:

Subprogram:

Program code	Comment
PROC SUB_PROG (REAL X, REAL Y, REAL Z)	; Formal parameters: X, Y, and Z
...	
N100 RET	

Main program:

Program code	Comment
PROC MAIN_PROG	
...	
N30 SUB_PROG(1.0,2.0,3.0)	; Subprogram call with complete parameter transfer: X=1.0, Y=2.0, Z=3.0
…	
N100 M30	

Examples for the subprogram call in N30 with incomplete parameter transfer:

```
N30 SUB_PROG( ,2.0,3.0)         ; X=0.0, Y=2.0, Z=3.0
N30 SUB_PROG(1.0, ,3.0)         ; X=1.0, Y=0.0, Z=3.0
N30 SUB_PROG(1.0,2.0)           ; X=1.0, Y=2.0, Z=0.0
N30 SUB_PROG( , ,3.0)           ; X=0.0, Y=0.0, Z=3.0
N30 SUB_PROG( , , )             ; X=0.0, Y=0.0, Z=0.0
```

Note

Call-by-reference parameter transfer

Parameters transferred using call-by-reference must not be left out of the subprogram call.

Note

AXIS data type

AXIS data type parameters must not be left out of the subprogram call.

Checking the transfer parameters

System variable $P_SUBPAR [n] where n = 1, 2, etc., can be used to check whether a parameter has been transferred explicitly or left out in the subprogram. The index n refers to the sequence of the formal parameters. Index n = 1 refers to the first formal parameter, index n = 2 to the second formal parameter, and so on.

The following program excerpt shows an example of how a check can be performed based on the first formal parameter:

Programming	Comment
PROC SUB_PROG (REAL X, REAL Y, REAL Z)	; Formal parameters: X, Y, and Z
N20 IF $P_SUBPAR[1]==TRUE	; Check of the first formal parameter X.
...	; These actions are taken if the formal parameter X has been transferred explicitly.
N40 ELSE	
...	; These actions are taken if the formal parameter X has not been transferred.
N60 ENDIF	
...	; General actions
N100 RET	

4.2.2 Definition of a subprogram

4.2.2.1 Subprogram without parameter transfer

When defining subprograms without parameter transfer, the definition line at the beginning of the program can be omitted.

Syntax

```
[PROC <ProgName>]
...
```

Meaning

PROC	Definition operation at the beginning of a program
<ProgName>	Name of the program

Examples

Example 1: Subprogram with PROC statement

Program code	Comment
PROC SUB_PROG	; Definition line
N10 G01 G90 G64 F1000	
N20 X10 Y20	
...	
N100 RET	; Subprogram return

Example 2: Subprogram without PROC statement

Program code	Comment
N10 G01 G90 G64 F1000	
N20 X10 Y20	
...	
N100 RET	; Subprogram return

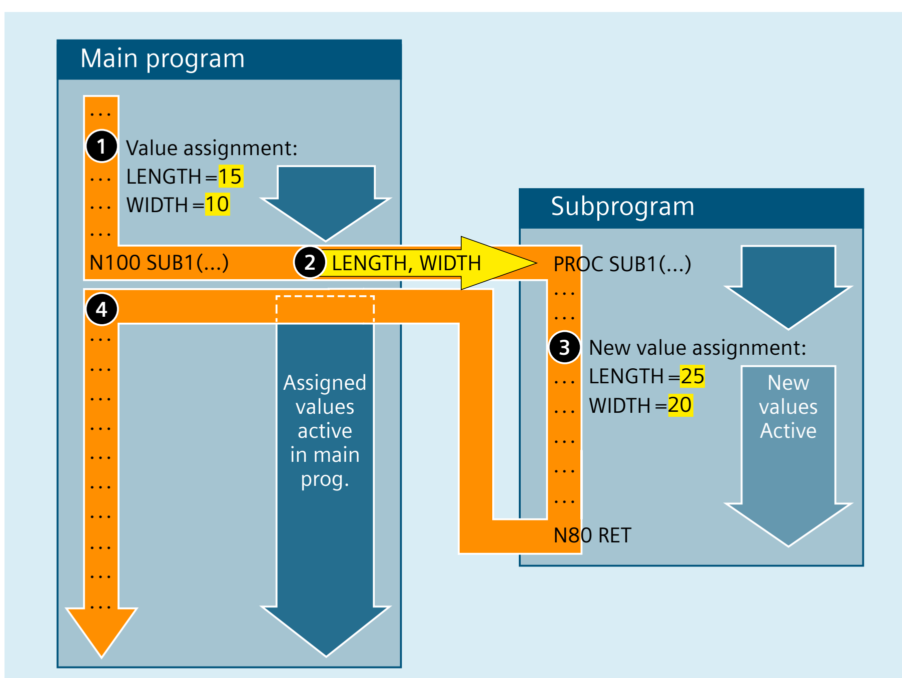
See also

Subprogram call without parameter transfer (Page 512)

4.2.2.2 Subprogram with call-by-value parameter transfer (PROC)

In a call-by-value parameter transfer, the calling program only transfers the value of a variable to the subprogram. Thus the subprogram is not given direct access to the variable. Only the value visible in the subprogram is modified when the parameter value is changed, while the value of the variables defined in the calling program remains unchanged. As a consequence, the call-by-value parameter transfer does not affect the calling program.

The following figure illustrates call-by-value parameter transfer using an example:



① Value assignment to the variables LENGTH and WIDTH in the calling program.

② When the SUB1 subprogram is called, the values of the two variables LENGTH and WIDTH are transferred to the subprogram.

③ If the values transferred change due to a new value assignment, the new values only apply to the subprogram.

④ After return from the subprogram, the values last assigned in the calling program are valid again.

Syntax

Definition

A subprogram with call-by-value parameter transfer is defined using the PROC keyword followed by the name of the program and a complete list of all the parameters with their type and name. The definition operation must appear in the first program line:

```
PROC <ProgName>(<Par1Type> <Par1Name>[=<InitValue>],<Par2Type>
<Par2Name>[=<InitValue>],...)
```

Call

Subprograms with parameter transfer must be declared with the keyword EXTERN (external) before they are called in the main program.

→ See Chapter "Subprogram call with parameter transfer (EXTERN) (Page 515)".

The actual subprogram call is made via the program name and specification of the transfer parameters.

→ See Chapter "Parameter transfer (Page 484)".

Meaning

PROC	Definition operation at the beginning of a program
<ProgName>	Name of the program
<Par1Type>	Data type of the 1st transfer parameter (e.g. REAL, INT, BOOL)
<Par2Type>	Data type of the 2nd transfer parameter
...	...
<Par1Name>	Name the 1st transfer parameter
<Par2Name>	Name the 2nd transfer parameter
...	...
<InitValue>	Value for the initialization of the parameter (optional) Parameters that are not specified in the call of the subprogram are then assigned the initialization value defined in the subprogram definition instead of "0" as the default value.

Example

Definition of a subprogram with three parameters of type REAL and with default values:

Program code

```
PROC SUB_PROG(REAL LENGTH=10.0, REAL WIDTH=20.0, REAL HEIGHT=30.0)
...
```

More information

Maximum number of transfer parameters

Up to 127 transfer parameters can be defined in a PROC statement.

When you define such a large number of transfer parameters, note that the PROC statement must not exceed the maximum permissible line length of 900 characters (or 512 characters when editing in the program editor of SINUMERIK Operate). Otherwise, alarm 6560 "Data format not allowed" will be output when the program is loaded into the NC (or edited in the program editor). It is therefore advisable to reduce the number of characters required for each parameter definition by using macros (Page 477).

Example:

A subprogram is to be defined with 127 transfer parameters. It is also assumed that all 127 transfer parameters are programmed according to the following pattern:

```
REAL PA1,
```

In this case, 9 characters would be required for each transfer parameter. For 127 transfer parameters that would therefore be far more than 900 characters, which is more than the maximum permitted.

To be able to define all 127 transfer parameters nevertheless, macros are used as follows:

```
DEFINE RL AS REAL
```

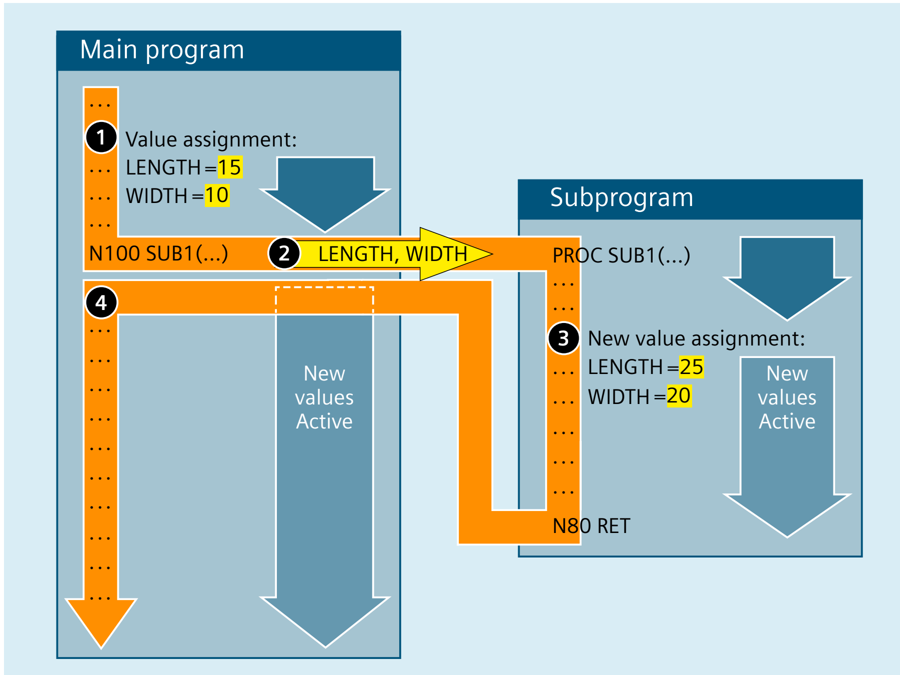
This reduces the number of characters required for each transfer parameter by 2 characters:
`RL PA1,`

The PROC statement thus remains below the maximum permissible line length and the program can be loaded into the NC.

4.2.2.3 Subprogram with call-by-reference parameter transfer (PROC, VAR)

The calling program transfers not the value of a variable to the subprogram on a call-by-reference parameter transfer, but a reference (pointer) to the variable. This gives the subprogram direct access to the variable. In this way, not only the value visible in the subprogram is modified when a parameter value is changed, but also the value of the variables defined in the calling program. Call-by-reference parameter transfer therefore affects the calling program, even after the subprogram has ended.

The following figure illustrates call-by-reference parameter transfer using an example:



① Value assignment to the variables LENGTH and WIDTH in the calling program.
② When the SUB1 subprogram is called, references to the two variables LENGTH and WIDTH are transferred to the subprogram.
③ New value assignment to the variables LENGTH and WIDTH in the subprogram.
④ Because of the new value assignment in the subprogram, the values of the variables defined in the calling program change. The new values are therefore active even after the return from the subprogram.

Note

The call-by-reference parameter transfer is then only necessary if the transferred variable was defined locally in the calling program (LUD). Channel-global or NC-global variables do not have to be transferred, since these cannot be accessed directly from within the subprogram.

Syntax

Definition

A subprogram with call-by-reference parameter transfer is defined using the PROC keyword followed by the name of the program and a complete list of all the parameters with the VAR keyword, type, and name. The definition operation must appear in the first program line. As parameters, references to arrays can also be transferred:

```
PROC <ProgName> (VAR <Par1Type> <Par1Name>, VAR <Par2Type>
<Par2Name>, ...)
PROC <ProgName> (VAR <Array1Type> <Array1Name> [<m>,<n>,<o>], VAR
<Array2Type> <Array2Name> [<m>,<n>,<o>], ...)
```

Call

Subprograms with parameter transfer must be declared with the keyword EXTERN (external) before they are called in the main program.

→ See Chapter "Subprogram call with parameter transfer (EXTERN) (Page 515)".

The actual subprogram call is made via the program name and specification of the transfer parameters.

→ See Chapter "Parameter transfer (Page 484)".

Meaning

<table>
<tr><td>PROC</td><td colspan="2">Definition operation at the beginning of a program</td></tr>
<tr><td>VAR</td><td colspan="2">Keyword for call-by-reference parameter transfer</td></tr>
<tr><td><ProgName></td><td colspan="2">Name of the program</td></tr>
<tr><td><Par1Type></td><td colspan="2">Data type of the 1st parameter (e.g. REAL, INT, BOOL)</td></tr>
<tr><td><Par2Type></td><td colspan="2">Data type of the 2nd parameter</td></tr>
<tr><td>...</td><td colspan="2">...</td></tr>
<tr><td><Par1Name></td><td colspan="2">Name of the 1st parameter</td></tr>
<tr><td><Par1Name></td><td colspan="2">Name of the 2nd parameter</td></tr>
<tr><td>...</td><td colspan="2">...</td></tr>
<tr><td><Array1Type></td><td colspan="2">Data type of the elements of array 1 (e.g. REAL, integer, BOOL)</td></tr>
<tr><td><Array2Type></td><td colspan="2">Data type of the elements of array 2</td></tr>
<tr><td>...</td><td colspan="2">...</td></tr>
<tr><td><Array1Name></td><td colspan="2">Name of array 1</td></tr>
<tr><td><Array2Name></td><td colspan="2">Name of array 2</td></tr>
<tr><td>...</td><td colspan="2">...</td></tr>
<tr><td rowspan="4">[<m>,<n>,<o>]</td><td colspan="2">Array size
Currently, up to 3-dimensional arrays are possible:</td></tr>
<tr><td><m></td><td>Array size for 1st dimension</td></tr>
<tr><td><n></td><td>Array size for 2nd dimension</td></tr>
<tr><td><o></td><td>Array size for 3rd dimension</td></tr>
</table>

Note

The program name specified after the PROC keyword must match the program name assigned on the user interface.

Example

Definition of a subprogram with two parameters as reference:

- Parameter 1: Reference to variable of type REAL with the name LENGTH
- Parameter 2: Reference to variable of type REAL with the name WIDTH

Program code

```
PROC SUB_PROG(VAR REAL LENGTH, VAR REAL WIDTH)
...
```

More information

Maximum number of transfer parameters

Up to 127 transfer parameters can be defined in a PROC statement.

When you define such a large number of transfer parameters, note that the PROC statement must not exceed the maximum permissible line length of 900 characters (or 512 characters when editing in the program editor of SINUMERIK Operate). Otherwise, alarm 6560 "Data format not allowed" will be output when the program is loaded into the NC (or edited in the program editor). It is therefore advisable to reduce the number of characters required for each parameter definition by using macros (Page 477).

For an example, see Chapter "Subprogram with call-by-value parameter transfer (PROC) (Page 487)".

Arrays with variable length

With arrays of an undefined array length, subprograms can process arrays of variable length as formal parameter. When defining a two-dimensional array as a formal parameter, for example, the length of the 1st dimension is not specified. However, the comma must be written.

Example:

```
PROC <ProgName> (VAR REAL <ArrayName>[ ,5])
```

4.2.2.4 Save modal G functions (SAVE)

If the subprogram definition contains the SAVE attribute, the modal G commands that were active before the subprogram call are saved and reactivated after the end of the subprogram.

Syntax

```
PROC <ProgName> SAVE
```

Meaning

PROC	Definition operation at the beginning of a program
SAVE	Saves the modal G commands before the subprogram call and restores them after the end of the subprogram.
<ProgName>	Name of the program

Example

In the CONTOUR subprogram, the modal G command G91 applies (incremental dimension). The modal G command G90 is effective in the main program (absolute dimension). G90 is again effective in the main program after the end of the subprogram as a result of the subprogram definition with SAVE.

Subprogram definition:

Program code	Comment
PROC CONTOUR (REAL VALUE1) SAVE	; Subprogram definition with the SAVE parameter
N10 G91 ...	; Modal G command G91 (incremental dimensioning)
N100 M17	; End of subprogram

Main program:

Program code	Comment
N10 G0 X... Y... G90	; Modal G command G90 (absolute dimensioning)
N20 ...	
...	
N50 CONTOUR (12.4)	;Subprogram call
N60 X... Y...	; Modal G command G90 reactivated using SAVE

More information

Continuous-path mode

If, for active continuous-path mode, a subprogram is called with the SAVE attribute, the continuous-path mode is interrupted at the end of the subprogram (return jump).

Frames

The behavior of frames regarding subprograms with the SAVE attribute depends on the frame time and can be set using machine data.

4.2.2.5 Suppress single block execution (SBLOF, SBLON)

Even with active single block machining, the user can completely or partly process an NC program without interruption. Single block machining is suppressed via the SBLOF command, and reactivated via the SBLON command.

Suppressing single block machining for the complete NC program

If the deactivation of single block machining (SBLOF) is programmed in the first line (PROC ...) of a **main program**, 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 line (PROC ...) of a **subprogram**, 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 with M17: Stop at the end of the subprogram
- Return with RET: **No** stop at end of subprogram

Suppressing 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

Suppressing single block machining for the complete NC program:

```
PROC ... SBLOF
...
```

Suppressing single block machining within the NC program:

```
...
SBLOF
...
SBLON
...
```

Meaning

PROC:	First operation in a program	
SBLOF:	Predefined procedure to deactivate single block machining	
	Alone in the block:	Yes, possible in the PROC block
	Effectiveness:	Modal
SBLON:	Predefined procedure to activate single block machining	
	Alone in the block:	yes
	Effectiveness:	Modal

Special aspects

- **Block display with suppressed single block machining**
  The current block display can be suppressed in subprograms using DISPLOF. If DISPLOF is programmed together with SBLOF, for single block stops within the subprogram, the subprogram call is displayed.
- **Suppression of single block machining with asynchronous subprograms (ASUB)**
  In order to execute an ASUB with active single block machining in one step, a PROC statement must be programmed in the ASUB with SBLOF. This also applies to the function "Editable system ASUB" (MD11610 $MN_ASUP_EDITABLE).
  If the single block stop in the system or user ASUB is suppressed by programming SBLOF in the PROC line or using the settings in machine data
  MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (bit0 = 1 or bit1 = 1), then the single block stop can be reactivated by programming SBLON in the ASUB.
  If the single block stop in the user ASUB is suppressed using the setting in machine data MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP, the single block stop **cannot** be reactivated by programming SBLON in the ASUB.
- **Special features for various single block machining types**
  - "SB2: Arithmetic block" AND MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, Bit 12 = 1:
    → The program is not stopped in the SBLON block.
  - "SB3: single block fine":
    → The SBLOF command is suppressed.
- **Suppression of single block machining in nested programs**
  If SBLOF was programmed in the PROC statement in a subprogram, then execution is stopped at the subprogram return with M17. That prevents the next block in the calling program from already running. If, in a subprogram with SBLOF, without SBLOF in the PROC statement, a suppression of single block machining is activated, execution is only stopped after the next machine function block of the calling program. If that is not wanted, SBLON must be programmed in the subprogram before the return (M17). Execution does not stop for a return to a higher-level program with RET.

Examples

Example 1: Suppressing single block machining within the NC program

Initial situation: Single block machining is active.

Program code	Comment
N10 G1 X100 F1000	
N20 **SBLOF**	; Switch off single block machining
N30 Y20	
N40 M100	
N50 R10=90	
N60 **SBLON**	; Reactivate single block machining
N70 M110	
N80 ...	

The area between N20 and N60 is executed as one step in single block mode.

Example 2: A cycle is to act like a command for a user

Initial situation: Single block machining is active.

Main program:

Program code
...
N100 G1 X10 G90 F200
N120 X-4 Y6
N130 **CYCLE1**
N140 G1 X0
N150 M30

Cycle CYCLE1:

Program code	Comment
N100 **PROC** CYCLE1 DISPLOF **SBLOF**	; Suppress single block machining for the complete program.
N110 R10=3*SIN(R20)+5	
N120 IF (R11 <= 0)	
N130 SETAL(61000)	
N140 ENDIF	
N150 G1 G91 Z=R10 F=R11	
N160 M17	

The cycle CYCLE1 is processed with active single block machining. CYCLE1 is processed for active single block machining, i.e. the Start key must be pressed once to process CYCLE1.

Example 3: An ASUB, which is started by the PLC in order to activate a modified zero offset and tool offsets, is to be executed invisibly

Program code	Comment
N100 **PROC** NV **SBLOF DISPLOF**	; Suppress single block machining and block display.
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	

Program code	Comment
N200 END: D=$P_TOOL T=$P_TOOLNO	
N210 RET	

Example 4: Specific stopping in the subprogram

Initial situation:

- Single block machining 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	

Cycle CYCLE:

Program code	Comment
PROC CYCLE **SBLOF**	; Suppress single block machining
N100 R0=1	
N110 **SBLON**	; No single block stop due to MD10702 bit12 = 1
N120 X1	; single block stop
N140 **SBLOF**	
N150 R0=2	
RET	

Example 5: Suppression of single block machining with program nesting

Initial situation:

- Single block machining type 2 is active.
- The program execution shall not be interrupted in the SBLON block (MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, Bit12 = 1)

Program code	Comment
N10 X0 F1000	; single block stop
N20 UP1(0)	
PROC UP1(INT _NR) **SBLOF**	; Deactivate single block machining for UP1
N100 X10	
N110 UP2(0)	
PROC UP2(INT _NR)	
N200 X20	
N210 **SBLON**	; Activate single block machining
N220 X22	; single block stop

Program code	Comment
N230 UP3(0)	
PROC UP3(INT _NR)	
N300 **SBLOF**	; Switch off single block machining
N305 X30	
N310 **SBLON**	; Activate single block machining
N320 X32	; single block stop
N330 **SBLOF**	; Switch off single block machining
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

4.2.2.6 Suppress current block display (DISPLOF, DISPLON, ACTBLOCNO)

The current program block is displayed as standard in the block display. The display of the current block can be suppressed in cycles and subprograms using the `DISPLOF` command. Instead of the current block, the call of the cycle or the subprogram is displayed. The `DISPLON` command revokes suppression of the block display.

`DISPLOF` and `DISPLON` are programmed in the program line with the `PROC` operation and are effective for the entire subprogram and implicitly for all subprograms called from it which do not contain a `DISPLON` or `DISPLOF` command. This is true for all ASUBs.

Syntax

```
PROC … DISPLOF
PROC … DISPLOF ACTBLOCNO
PROC … DISPLON
```

Meaning

`DISPLOF`:	Command to suppress the current block display.	
	Location:	At the end of the program line with the `PROC` operation
	Effective:	Up to the return jump from the subprogram or end of program.
	Note: If further subprograms are called from the subprogram using the `DISPLOF` command, then the current block display is also suppressed in these subprograms unless `DISPLON` is explicitly programmed in them.	

DISPLON:	Command for revoking suppression of the display of the current block	
	Location:	At the end of the program line with the PROC operation
	Effective:	Up to the return jump from the subprogram or end of program.
	Note: If further subprograms are called from the subprogram using the DISPLON command, then the current block will also be displayed in these subprograms unless DISPLOF is explicitly programmed in them.	
ACTBLOCNO:	DISPLOF together with the ACTBLOCNO attribute means that in the case of an alarm, the number of the actual block is output in which the alarm occurred. This also applies if only DISPLOF is programmed in a lower program level. On the other hand, for DISPLOF without ACTBLOCNO, the block number of the cycle or subprogram call from the last program level not designated with DISPLOF is displayed.	

Examples

Example 1: Suppress current block display in the cycle

Program code	Comment
PROC CYCLE (AXIS TOMOV, REAL POSITION) SAVE DISPLOF	; Suppress current block display Instead, the cycle call should be displayed, e.g.: CYCLE(X,100.0)
DEF REAL DIFF	;Cycle contents
G01 ...	
...	
RET	; Subprogram return jump. The block following the cycle call is displayed in the block display.

Example 2: Block display for alarm output

Subprogram SUBPROG1 (with ACTBLOCNO):

Program code	Comment
PROC SUBPROG1 DISPLOF ACTBLOCNO	
N8000 R10 = R33 + R44	
...	
N9040 R10 = 66 X100	; Trigger alarm 12080
...	
N10000 M17	

Subprogram SUBPROG2 (without ACTBLOCNO):

Program code	Comment
PROC SUBPROG2 DISPLOF	
N5000 R10 = R33 + R44	

Program code	Comment
...	
N6040 R10 = 66 X100	; Trigger alarm 12080
...	
N7000 M17	

Main program:

Program code	Comment
N1000 G0 X0 Y0 Z0	
N1010 ...	
...	
N2050 SUBPROG1	; Alarm output = "12080 channel K1 block N9040 syntax error for text R10="
N2060 ...	
N2350 SUBPROG2	; Alarm output = "12080 channel K1 block N2350 syntax error for text R10="
...	
N3000 M30	

Example 3: Revoke suppression of the current block display

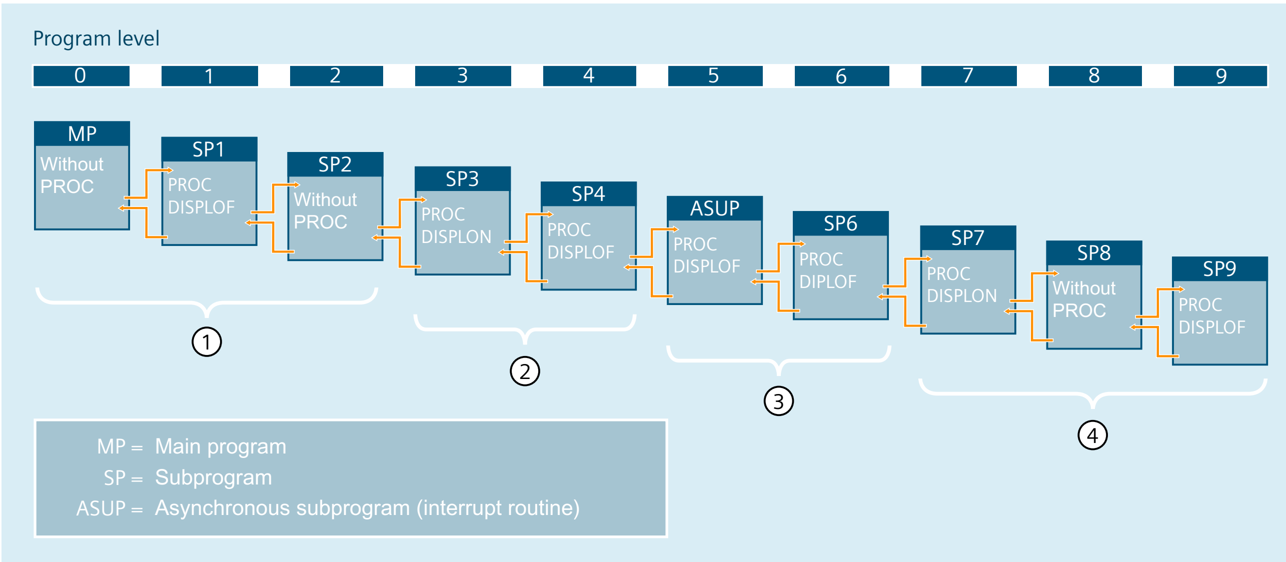
Subprogram SUB1 with suppression:

Program code	Comment
PROC SUB1 DISPLOF	; Suppress current block display in SUB1 subprogram. Instead, the block is to be displayed with the SUB1 call.
...	
N300 SUB2	; Call subprogram SUB2.
...	
N500 M17	

Subprogram SUB2 without suppression:

Program code	Comment
PROC SUB2 DISPLON	; Revoke suppression of the current block display in subprogram SUB2.
...	
N200 M17	; Return to subprogram SUB1. Suppression of the current block display is restored in SUB1.

Example 4: Display response for different DISPLON/DISPLOF combinations



① The part program lines from program level 0 are displayed in the current block display.
② The part program lines from program level 3 are displayed in the current block display.
③ The part program lines from program level 3 are displayed in the current block display.
④ The part program lines from program level 7/8 are displayed in the current block display.

4.2.2.7 Identifying subprograms with preparation (PREPRO)

All files can be identified with the `PREPRO` keyword at the end of the `PROC` operation line during power up.

Note

This type of program preprocessing depends on the setting of machine data MD10700 $MN_PREPROCESSING_LEVEL. Please follow the manufacturer's instructions.

Syntax

```
PROC … PREPRO
```

Meaning

`PREPRO:`	Keyword for identifying all files (of the NC programs stored in the cycle directories) prepared during power up

Read subprogram with preparation and subprogram call

The cycle directories are processed in the same order both for subprograms preprocessed with parameters during power up and during subprogram call.

1. _N_CUS_DIR user cycles
2. _N_CMA_DIR manufacturer cycles
3. _N_CST_DIR standard cycles

In the case of NC programs sharing the same name but having different characteristics, the first `PROC` operation found is activated and the other `PROC` operation is overlooked without an alarm message.

4.2.2.8 Setting a return program jump (overview)

One of the following return jump commands or end of program command M30 is located at the end of a subprogram:

- M17 / M30 (Page 502)
- RET (Page 503)
- RET(...) / RETB(...) (Page 504)

4.2.2.9 Program return jump M17 or end of program M30

Just like the end of program command M30, M17 results in a return jump to the calling program to the block after the subprogram call.

Note

M17 and M30 are treated as equivalents in the NC language.

Syntax

```
PROC <ProgName>
...
M17/M30
```

Constraints

Impact on the continuous-path mode

An active continuous-path mode in the channel is interrupted if M17 (or M30) appears on its own in the part program block.

To avoid this, M17 (or M30) should also be written in the last traversing block.

Furthermore, the following machine data must be set to "0":

MD20800 $MC_SPF_END_TO_VDI = 0 (no M30/M17 output to the NC/PLC interface)

Examples

Example 1: Subprogram with M17 in a separate block

Program code	Comment
N10 G64 F2000 G91 X10 Y10	
N20 X10 Z10	
N30 M17	; Return jump with interruption of continuous-path mode.

Example 2: Subprogram with M17 in the last traversing block

Program code	Comment
N10 G64 F2000 G91 X10 Y10	
N20 X10 Z10 M17	; Return jump without interruption of continuous-path mode.

4.2.2.10 RET program return

Just like M17/M30, RET results in a return jump to the calling program to the block after the subprogram call. However, contrary to M17/M30, an active continuous-path mode in the channel is not interrupted.

Parameters can be programmed to adapt the return jump behavior of RET (see "Parameterizable program return jump RET(...) / RETB(...) (Page 504)").

Note

RET can only be used in subprograms, which were not defined with the SAVE attribute.

Syntax

RET **must** be located in a dedicated block:

```
PROC <ProgName>
...
RET
```

Example

Main program

Program code	Comment
PROC MAIN_PROGRAM	; Start of the program
...	
N50 SUB_PROG	; Subprogram call: SUB_PROG
N60 ...	
...	

Program code	Comment
N100 M30	; End of program

Subprogram

Program code	Comment
PROC SUB_PROG	
...	
N100 **RET**	; Return jump to block N60 in the main program.

4.2.2.11 Parameterizable program return jump RET(...) / RETB(...)

Generally, a return jump is made from a subprogram into the calling program. Processing is then continued with the program line following the subprogram call.

In addition, there are also applications where a different return jump behavior is required, for example:

- Resume program execution after calling the stock removal cycles in the ISO dialect mode (after describing the contour).
- Return to the main program from any subprogram level (even after ASUB) for error handling
- Return jump across several program levels for special applications in compile cycles and in the ISO dialect mode

In cases such as these, predefined procedures RET(...) and RETB(...) are applied, which facilitate return jumps across program levels to the program block defined as destination.

Syntax

```
RET/RETB("<Target>")
RET/RETB("<Target>",<NextBlock>)
RET/RETB("<Target>",<NextBlock>,<NrRetJumpLevels>)
RET/RETB("<Target>", ,<NrRetJumpLevels>)
RET/
RETB("<Target>",<NextBlock>,<NrRetJumpLevels>,<RetJumpToProgStart>)
RET/RETB( , ,<NrRetJumpLevels>,<RetJumpToProgStart>)
```

Meaning

RET(...)	RET(...) first searches the specified return jump target in the program end direction. A search is made toward the program start if the search was not successful.
RETB(...)	RETB(...) only searches the specified return jump target in the program start direction.

<table>
<tr><td rowspan="4"><Target></td><td colspan="3">1st parameter: Return jump target
Specifies the block where program execution should be resumed.
If parameter <NrRetJumpLevels> is not programmed, then the return jump target is located in the program from which the current subprogram was called.</td></tr>
<tr><td>Type:</td><td colspan="2">STRING</td></tr>
<tr><td>Value:</td><td colspan="2">The following can be specified as return jump target:
• Block number
• Jump marker
• Character string, for example program or variable name
The character string must satisfy the following conditions:
– Blank at the end (contrary to the jump marker, which is identified by a ":" at the end).
– Before the character string only one block number and/or a jump marker may be set, no program commands.</td></tr>
<tr><td colspan="3">Note:
The specified jump destination must exist in the program to which the return jump is made. Otherwise, the search does not deliver a result, and alarm 14080 ("Jump destination not found") is output. This is also the case if, for a return jump with RETB, the jump destination is not in the program start direction, but in the program end direction.</td></tr>
<tr><td rowspan="4"><NextBlock></td><td colspan="3">2nd parameter: Following block
This parameter should be programmed if program execution should not be continued with the destination block, but only with the following block.</td></tr>
<tr><td>Type:</td><td colspan="2">INT</td></tr>
<tr><td rowspan="2">Value:</td><td>0</td><td>Continue program execution with the block that is specified in parameter <Target> as return jump destination.</td></tr>
<tr><td>> 0</td><td>Continue program execution with the subsequent block.</td></tr>
<tr><td rowspan="7"><NrRetJumpLevels></td><td colspan="3">3rd parameter: Number of program levels to be skipped
Specifies the number of program levels to be skipped to access the program level with the destination block</td></tr>
<tr><td>Type:</td><td colspan="2">INT</td></tr>
<tr><td rowspan="4">Value:</td><td>1</td><td>The program is resumed at the "current program level -1" (just like RET without parameter).</td></tr>
<tr><td>2</td><td>The program is resumed at the "current program level -2", i.e. one level is skipped.</td></tr>
<tr><td>3</td><td>The program is resumed at the "current program level -3", i.e. two levels are skipped.</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>Value range:</td><td colspan="2">1 ... 15</td></tr>
</table>

`<RetJumpToProgStart>`	**4th parameter: Return jump to the program start** This parameter should be programmed, if for a return jump into the main program, and where the **ISO dialect mode** is active, the program should be continued at the program start.		
	Type:	BOOL	
	Value:	1	If the return jump is made into the main program and an **ISO dialect mode** is active there, then the program branches to the beginning of the program.

Note

If a character string, which is not a block number, is specified as return jump destination, when making a search, it is initially assumed that it involves a jump marker. As a consequence, a program or variable name used as return jump destination must not match the name of a jump marker, as otherwise the return jump is always made to the block with the jump marker, and not to the block with the program or variable name (see example 2).

Constraints

Return jump across several program levels

When making a return jump through several program levels, the SAVE instructions of the individual program levels are evaluated.

If, for a return jump over several program levels, a modal subprogram is active and if in one of the skipped programs the deselection command MCALL is programmed for the modal subprogram, then the modal subprogram remains active.

NOTICE

The return jump can impact modal settings

For a return jump across several program levels, it is the user's responsibility to ensure that processing is continued with the necessary modal settings. This can be achieved, e.g. by programming an appropriate main block.

Process from external source

If main programs and subprograms are executed from an external program memory via the "Execution from external source" function, the return destination programmed with RET(...) must be within the reload memory. Otherwise, the jump destination is not found and the program aborts and alarm 14000 ("Illegal end of file") is output.

Note

To be able to execute external programs without restrictions with regard to the programmed jump instructions, it is recommended to use the option "Execution from External Storage (EES)" instead of the function "Execution from external source".

RET(...) examples

Example 1: Resuming in the main program after ASUB execution

Programming	Comment
N10010 CALL "UP1"	; Program level 0 (main program)
N11000 PROC UP1	; Program level 1
N11010 CALL "UP2"	
N12000 PROC UP2	; Program level 2
...	
N19000 PROC ASUP	; Program level 3 (ASUB execution)
...	
N19100 RET("N10900", ,$P_STACK)	; Subprogram return jump into the main program
	; $P_STACK: actual program level
N10900	; Target block in the main program
N10910 MCALL	; Deactivate the modal subprogram call
N10920 G0 G60 G40 M5	; Initialize additional modal settings

Example 2: Character string as return jump destination

Main program:

Program code	Comment
PROC MAIN_PROGRAM	
N1000 DEF INT iVar1=1, iVar2=4	
N1010 ...	
N1200 subProg1	; Calls subprogram "subProg1"
N1210 M2 S1000 X10 F1000	
N1220	
N1400 subProg2	; Calls subprogram "subProg2"
N1410 M3 S500 Y20	
N1420 ..	
N1500 lab1: iVar1=R10*44	
N1510 F500 X5	
N1520 ...	
N1550 subprog1: G1 X30	; "subProg1" is defined here as jump marker.
N1560 ...	
N1600 subProg3	; Calls subprogram "subProg3"
N1610 ...	
N1900 M30	

Subprogram subProg1:

Program code	Comment
PROC subProg1	

Program code	Comment
N2000 R10=R20+100	
N2010 ...	
N2200 RET("subProg2")	; Return jump into the main program at block N1400

Subprogram subProg2:

Program code	Comment
PROC subProg2	
N2000 R10=R20+100	
N2010 ...	
N2200 RET("iVar1")	; Return jump into the main program at block N1500

Subprogram subProg3:

Program code	Comment
PROC subProg3	
N2000 R10=R20+100	
N2010 ...	
N2200 RET("subProg1")	; Return jump into the main program at block N1550

RETB(...) example

Program code	Comment
EXAMPLE.MPF	
…	
N3000 START_CYC(parm1, param2, …)	
N3010 TECH_CYC1(param1, param2, …)	
N3020 TECH_CYC2(param1, param2, …)	
N3030 TECH_CYC3(param1, param2, …)	
N3040 **END_CYC**(param1, param2, …)	
N3050 …	
N4500 START_CYC(param11, param12, …)	
N4510 …	
N4590 END_CYC(param11, param12, ..)	
N5000 …	
…	
N6000 M30	

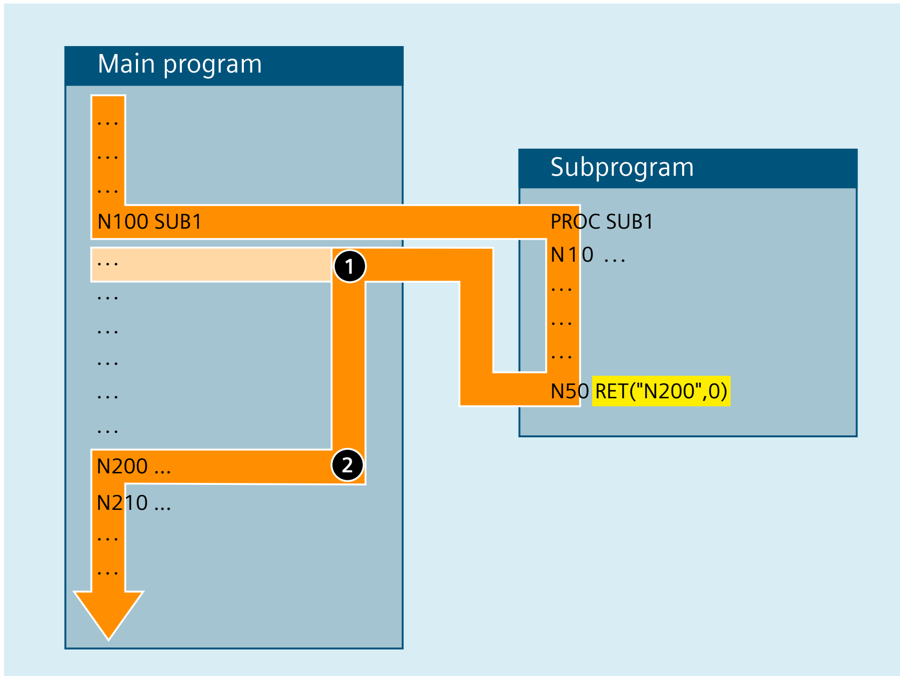
Program code	Comment
PROC END_CYC(…)	; Call in the main program, line N3040

Program code	Comment
N10000 …	
N15000 if status == 1	
N15010 RETB("START_CYC")	; Return jump to the calling program EXAMPLE.MPF
	; Search for character string "START_CYC"
	; Search direction: backward in the program start direction
	; Program processing is continued with line N3000
N15020 endif	
N15030 if status == 0	
N15040 RET	; Return jump to the calling program EXAMPLE.MPF
	; Program processing is continued with line N3050
N15050 endif	
N16000 RET("START_CYC")	; Return jump to the calling program EXAMPLE.MPF
	; Search for character string "START_CYC"
	; Search direction: forward in the program end direction
	; Program processing is continued with line N4500
N17060 RETB	; Return jump to the calling program EXAMPLE.MPF
	; Program processing is continued with line N3050
	; RETB without parameter is identical to RET

More information

The following examples for RET(...) illustrate how the return jump behavior can be adapted by programming the parameter.

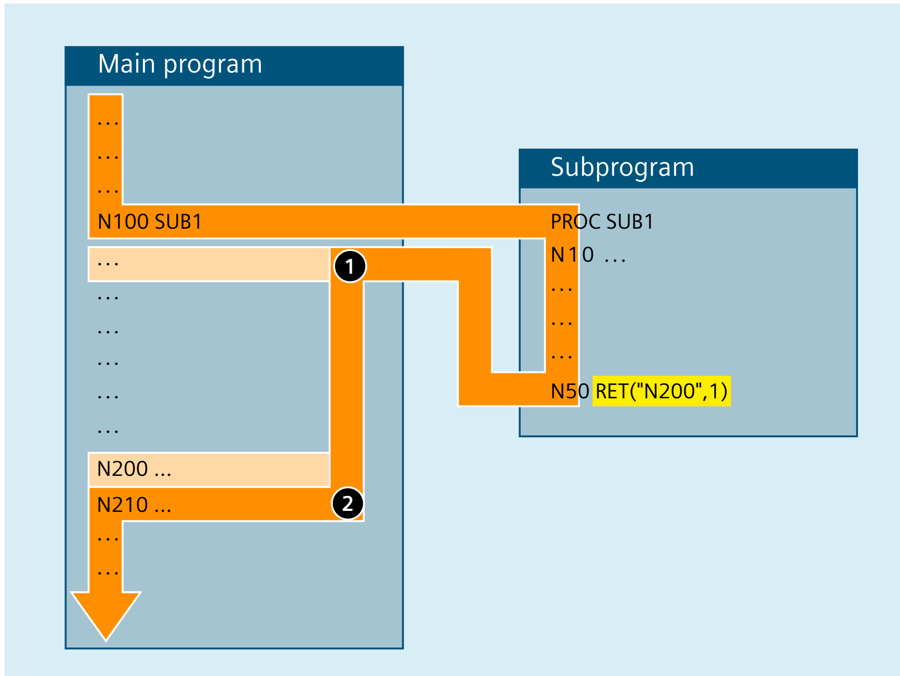
RET("N200",0)



① After the RET command, the return jump to the main program is first made to the block after the calling block.

② A search is then made for the destination in the direction of the program end and program execution continued with block N200.

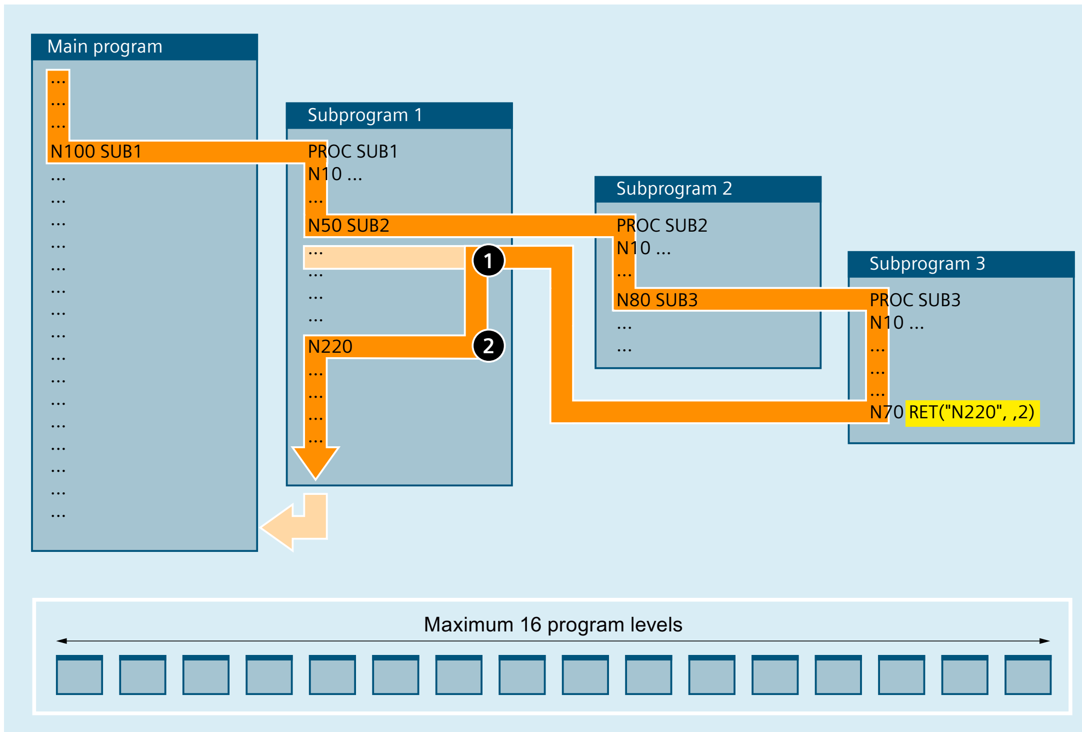
RET("N200",1)



① After the RET command, the return jump to the main program is first made to the block after the calling block.

② A search is then made for the destination in the direction of the program end and program execution continued after the block that follows destination block N200.

RET("N220", ,2)



① After the RET command, two program levels are jumped back.

② A search is then made for the destination in the direction of the program end and program execution continued with block N220.

4.2.3 Subprogram call

4.2.3.1 Subprogram call without parameter transfer

A subprogram is called either with address L and subprogram number or by specifying the program name.

A main program can also be called as a subprogram. The end of program `M2` or `M30` set in the main program is evaluated as `M17` in this case (end of program with return to the calling program).

Note

Accordingly, a subprogram can also be started as a main program.

Search strategy of the control:

Are there any *_MPF?

Are there any *_SPF?

This means, if the name of the subprogram to be called is identical to the name of the main program, the main program that issued the call is called again. This is generally an undesirable effect and must be avoided by assigning unique names to subprograms and main programs.

Note

Subprograms not requiring parameter transfer can also be called from an initialization file.

Syntax

```
L<number>/<program name>
```

Note

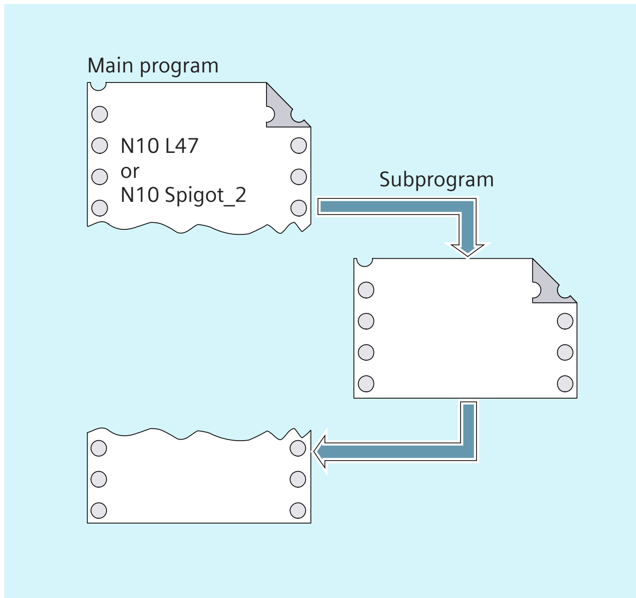
The subprogram call must always be programmed in a separate NC block.

Meaning

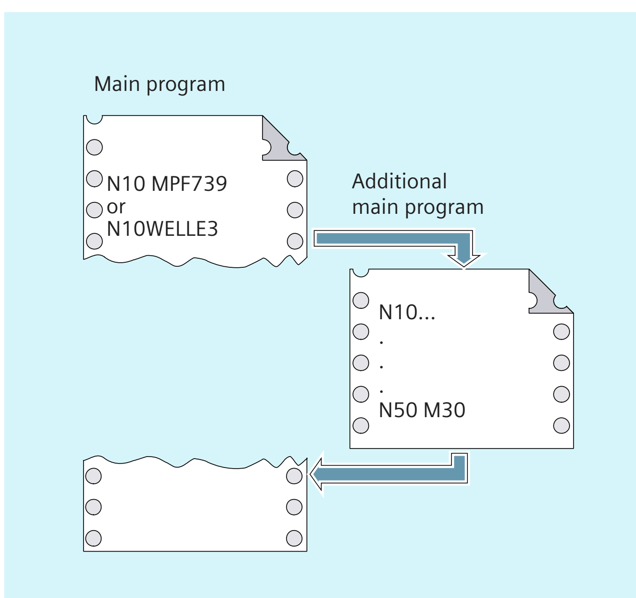
`L`:	Address for the subprogram call	
`<number>`:	Name of the subprogram	
	Type:	INT
	Value:	Maximum 7 decimal places **Notice:** Leading zeros are significant in names (⇒ L123, L0123 and L00123 are three different subprograms).
`<program name>`:	Name of the subprogram (or main program)	

Examples

Example 1: Subprogram call without parameter transfer



Example 2: Calling a main program as a subprogram



See also

Subprogram without parameter transfer (Page 486)

4.2.3.2 Subprogram call with parameter transfer (EXTERN)

For a subprogram call with parameter transfer, variables or values can be transferred directly (only with call-by-value parameter transfer).

Subprograms with parameter transfer must be declared with keyword EXTERN (external) in the main program before they are called in the main program (e.g. at the beginning of the program). The name of the subprogram and the variable types are thereby specified in the sequence in which they are transferred.

NOTICE

Risk of confusion

Both the variable types and the sequence of the transfer must match the definitions declared under PROC in the subprogram. The parameter names can be different in the main program and the subprogram.

Syntax

```
EXTERN <ProgName>(<Par1Type>,<Par2Type>,<Par3Type>)
...
<ProgName>(<Par1Value>,<Par2Value>,<Par3Value>)
```

Note

The subprogram call must always be programmed in a separate NC block.

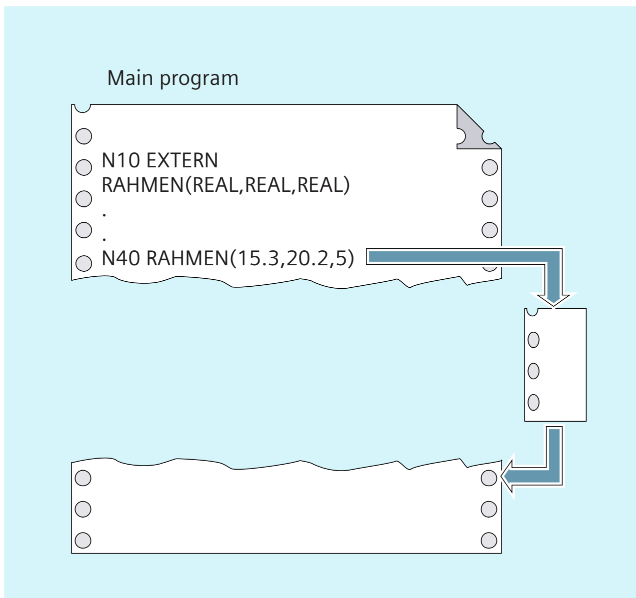
Meaning

`<ProgName>`	Name of subprogram
`EXTERN`	Keyword to declare a subprogram with parameter transfer. **Note:** You only have to specify EXTERN (external) if the subprogram is in the workpiece or in the global subprogram directory. Cycles do not have to be declared as EXTERN (external).
`<Par1Type>,<Par2Type>,<Par3Type>`	Variable types of the parameters to be transferred in the sequence of the transfer
`<Par1Value>,<Par2Value>,<Par3Value>`	Variable values for the parameters to be transferred

Examples

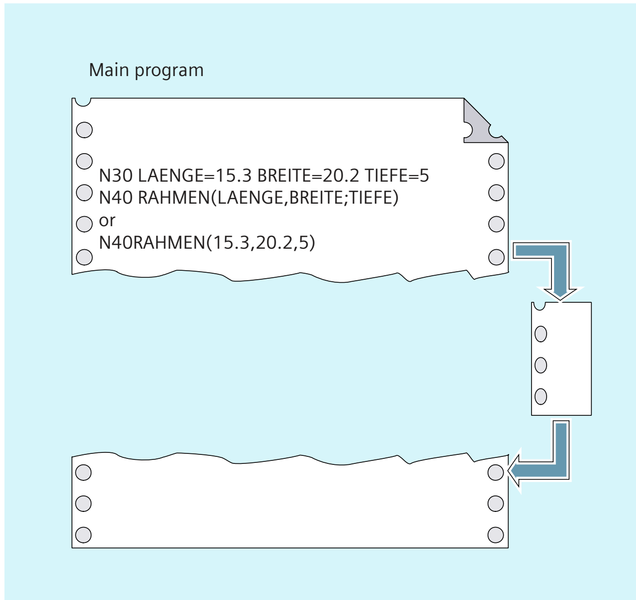
Example 1: Subprogram call preceded by declaration

Program code	Comment
N10 EXTERNAL BORDERS(REAL,REAL,REAL)	; Specify the subprogram.
...	
N40 BORDER(15.3,20.2,5)	; Call the subprogram with parameter transfer.



Example 2: Subprogram call without declaration

Program code	Comment
N10 DEF REAL LENGTH, WIDTH, DEPTH	
N20 …	
N30 LENGTH=15.3 WIDTH=20.2 DEPTH=5	
N40 BORDER(LENGTH,WIDTH,DEPTH)	; or: N40 BORDER(15.3,20.2,5)



See also

Subprogram with call-by-value parameter transfer (PROC) (Page 487)

Subprogram with call-by-reference parameter transfer (PROC, VAR) (Page 489)

4.2.3.3 Number of program repetitions (P)

If a subprogram is to be executed several times in succession, the desired number of program repetitions can be entered at address `P` in the block with the subprogram call.

> **CAUTION**
>
> **Subprogram call with program repetition and parameter transfer**
>
> Parameters are transferred only when the program is called, i.e., on the first run. The parameters remain unchanged for the remaining repetitions. If you want to change the parameters during program repetitions, you must make the appropriate provision in the subprogram.

Syntax

```
<program name> P<value>
```

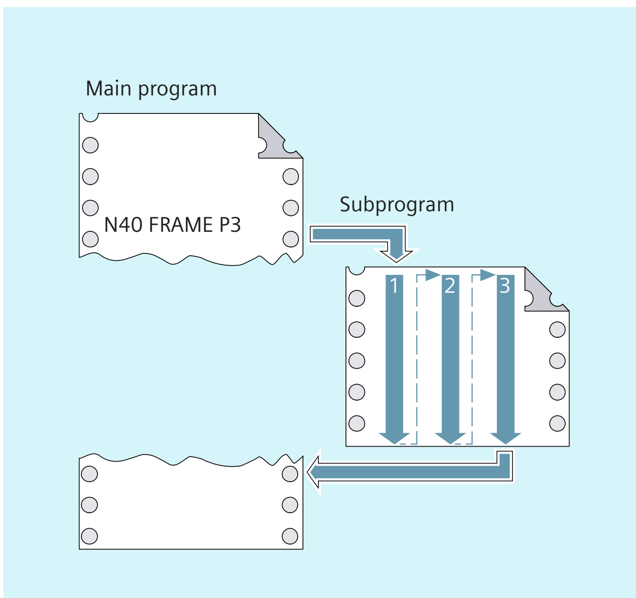
Meaning

`<program name>:`	Subprogram call
`P:`	Address to program program repetitions

`<value>`:	Number of program repetitions	
	Type:	INT
	Range of values:	1 ... 9999 (unsigned)

Example

Program code	Comment
...	
N40 FRAME P3	; The BORDER subprogram is to be executed three times one after the other.
...	



4.2.3.4 Modal subprogram call (MCALL, MCALLOF)

With the MCALL subprogram call, the user can activate a modally effective subprogram within a part program, which is automatically selected and processed after each traversing block with path movement following activation, even across program levels.

In the event that path movements occur in the part program after which the subprogram is not to be executed, automatic selection and execution of the specified subprogram can be suppressed block by block by programming the MCALLOF keyword in the relevant traverse blocks.

Note

In a program sequence, only the last modal subprogram call is ever effective. The current modal subprogram call replaces the one that has been active up until then.

Note

LUD definition and value assignment

The data definition area at the start of the subprogram is only executed once when executing the block with the programmed MCALL call. For the following subprogram calls, after the traversing blocks, the data definition area is no longer executed. This means that a value assigned in the definition of a local user variable (LUD) is no longer available after the first call, but the value that was last written in the cycle. The variable is not recreated after the first call but keeps the last value from the previous call. To work around this response, we recommend that the LUD definition and the value assignment are separated from one another, and the value assignment is programmed in a dedicated block after the data definition area.

NOTICE

Modal subprogram calls without path motion

In the following situations the modal subprogram is also called without programming path motion:

- Programming addresses S or F if G0 or G1 is active.
- If G0 or G1 were programmed alone in the block or with additional G commands.

Syntax

```
...
MCALL <ProgName> [(Par1, Par2, ...)]


...
MCALLOF ...
...
MCALL
...
```

Meaning

MCALL <ProgName>	Modally effective subprogram call	
<ProgName>	Name of subprogram	
	Type:	STRING
(Par1, Par2, etc.)	Transfer parameters (optional) **Note:** If parameters are transferred to the subprogram, the parameters are only transferred with call!	
MCALLOF	Suppress modally effective subprogram call block by block	
MCALL	MCALL without specification of a program name deactivates the modally effective subprogram call.	

Constraints

ASUB

If the part program processing is interrupted by an ASUB (see Chapter "Interrupt routine (ASUB) (Page 529)"), then no modal subprogram calls are executed in this ASUB.

If an ASUB is started in the "Reset" channel state, then it behaves just like a normal part program with regard to the modal subprogram calls.

Tool change cycle

If the modally effective subprogram call is deselected during the tool change cycle, note that the tool change cycle is called implicitly, even after a block search, via the search ASUB, or manually via overstore. In this situation, the modally effective subprogram call must not be deselected because otherwise the search result is falsified. It is therefore recommended that the deselection of the modally effective subprogram call in the tool change cycle is programmed as follows:

Program code	Comment
...	
IF $AC_ASUB == 0	; Call is not performed via search ASUB or overstore.
MCALL	; Deactivate modally effective subprogram call.
ENDIF	
...	

Examples

Example 1: Activate/deactivate modally effective subprogram call

Program code	Comment
N10 G0 X0 Y0	
N20 **MCALL L70**	; Activate modally effective call for subprogram L70.
N30 X10 Y10	; X10 Y10 is approached, and then L70 is called.
N40 X20 Y20	; X20 Y20 is approached, and then L70 is called.
...	
N100 **MCALL**	; Deactivate modally effective subprogram call.
N110 X0 Y0	; X0 Y0 is approached, L70 is **not** called.

Example 2: Modally effective subprogram call across program levels

Program code
N10 G0 X0 Y0
N20 **MCALL L70**
N30 L80

In this example, the following NC blocks with programmed path axes are in subprogram L80. L70 is called by L80.

Example 3: Suppress modally effective subprogram call block by block

Program code	Comment
N10 **MCALL L70 (Par1, Par2, …)**	; Activate modally effective call for subprogram L70 with parameter transfer.
N20 X=0	; Traversing set with subsequent execution of L70.
N30 X=100 **MCALLOF**	; The execution of L70 after this traversing set is suppressed.
N40 X=150	; Traversing set with subsequent execution of L70.
...	

4.2.3.5 Indirect subprogram call (CALL)

Depending on the prevailing conditions at a particular point in the program, different subprograms can be called. The name of the subprogram is stored in a variable of the STRING type. The subprogram call is realized with `CALL` and the variable name.

Note

The indirect subprogram call is only possible for subprograms without parameter transfer. For a direct subprogram call, save the name in a STRING constant.

Syntax

```
CALL <program name>
```

Meaning

`CALL`:	Command for the indirect subprogram call.	
`<program name>`:	Name of the subprogram (variable or constant)	
	Type:	STRING

Example

Direct call with STRING constant:

Program code	Comment
…	
CALL "/_N_WKS_DIR/_N_SUBPROG_WPD/_N_PART1_SPF"	; Direct call to subprogram PART1 with CALL.
…	

Indirect call via variable:

Program code	Comment
…	
DEF STRING[100] PROGNAME	: Define variable.
PROGNAME="/_N_WKS_DIR/_N_SUBPROG_WPD/_N_PART1_SPF"	; Assign subprogram PART1 to the PROGNAME variable.
CALL PROGNAME	; Indirect call to subprogram PART1 via CALL and the PROG-NAME variable.
…	

4.2.3.6 Indirect subprogram call with specification of the calling program part (CALL BLOCK ... TO ...)

`CALL` and the keyword combination `BLOCK ... TO` is used to call a subprogram indirectly and execute the program section designated by the start and end labels.

Syntax

```
CALL <program name> BLOCK <start label> TO <end label>
CALL BLOCK <start label> TO <end label>
```

Meaning

`CALL`:	Command for the indirect subprogram call.	
`<program name>`:	Name of the subprogram (variable or constant) that contains the program section to be executed (**specification optional**).	
	Type:	STRING
	Note: If a `<program name>` has not been programmed, the program section designated by `<start label>` and `<end label>` is searched for in the current program and executed.	
`BLOCK ... TO ... :`	Keyword combination for indirect program section execution	
`<start label>`:	Variable that refers to the start of the program section to be executed.	
	Type:	STRING
`<end label>`:	Variable that refers to the end of the program section to be executed.	
	Type:	STRING

Example

Main program:

Program code	Comment
...	
DEF STRING[20] STARTLABEL, ENDLABEL	; Variable definition for the start and end labels.

Program code	Comment
STARTLABEL="LABEL_1"	
ENDLABEL="LABEL_2"	
...	
CALL "CONTUR_1" BLOCK STARTLABEL TO ENDLABEL	; Indirect subprogram call and identifier associated with the calling program section.
...	

Subprogram:

Program code	Comment
PROC CONTUR_1 ...	
LABEL_1	; Start label: Start of program section execution.
N1000 G1 ...	
...	
LABEL_2	; End label: End of program section execution.
...	

4.2.3.7 Indirect call of a program programmed in ISO language (ISOCALL)

A program programmed in an ISO language can be called using the indirect program call ISOCALL. The ISO mode set in the machine data is then activated. The original execution mode becomes effective again at the end of the program. If no ISO mode is set in the machine data, the subprogram is called in Siemens mode.

Additional information about the ISO mode: Function Manual ISO Dialects

Syntax

```
ISOCALL <program_name>
```

Meaning

ISOCALL:	Keyword for an indirect subprogram call with which the ISO mode set in the machine data is activated.
<program name>:	Name of the program programmed in an ISO language (variable or constant, type STRING)

Example: Calling a contour with cycle programming from ISO mode

Program code	Comment
0122_SPF	; Contour description in ISO mode
N1010 G1 X10 Z20	
N1020 X30 R5	
N1030 Z50 C10	
N1040 X50	

Program code	Comment
N1050 M99	
N0010 DEF STRING[5] PROGNAME = "0122"	; Siemens part program (cycle)
...	
N2000 R11 = $AA_IW[X]	
N2010 ISOCALL PROGNAME	
N2020 R10 = R10+1	; Execute program 0122.spf in ISO mode
...	
N2400 M30	

4.2.3.8 Call subprogram with path specification and parameters (PCALL)

With PCALL, you can call subprograms with the absolute path and parameter transfer.

Syntax

```
PCALL <path/program name>(<parameter 1>,…,<parameter n>)
```

Meaning

PCALL:	Keyword for subprogram call with absolute path name
<path/program name>:	Absolute path data including subprogram names. Rules regarding path data, see "Addressing program memory files (Page 544)". If no absolute path name is specified, PCALL behaves like a standard subprogram call with a program identifier. The program name is specified without prefix and without file identifier. If the program name is to be programmed with prefix and file identifier, then it must be explicitly declared with prefix and file identifier using the EXTERN command.
<parameter 1>, ...:	Actual parameters in accordance with the PROC operation of the subprogram.

Example

Program code
PCALL/_N_WKS_DIR/_N_SHAFT_WPD/SHAFT(parameter1,parameter2,…)

4.2.3.9 Extend search path for subprogram calls (CALLPATH)

The search path for subprogram calls can be extended using the CALLPATH command. This means that also subprograms can be called from a non-selected workpiece directory without having to specify the complete, absolute path name of the subprogram.

Another application option is possible in the EES mode "EES without GDIR", if another directory is used on an external program memory to save global subroutines. In this case, using CALLPATH the search path can be extended by this subprogram directory.

The search path extension is made before the entry for user cycles (_N_CUS_DIR).

The search path extension is deselected again as a result of the following events:

- CALLPATH with blanks
- CALLPATH without parameter
- End of part program
- Reset

Syntax

```
CALLPATH("<path name>")
```

Meaning

CALLPATH:	Keyword for the programmable search path extension. Is programmed in a separate part program line.
<path name>:	Constant or variable, STRING type. Contains the absolute path name of the directory by which the search path should be extended. Rules regarding path data, see "Addressing program memory files (Page 544)".

Example

The search path should be extended by a certain workpiece directory:

Program code
...
CALLPATH ("/_N_WKS_DIR/_N_MYWPD_WPD")
...

This means that the following search path is set (position 5. is new):

1. Actual directory/*name*
2. Actual directory/*name_SPF*
3. Actual directory/*name_MPF*
4. //NC:/_N_SPF_DIR / *name_SPF*
5. **/_N_WKS_DIR/_N_MYWPD_WPD/name_SPF**
6. /N_CUS_DIR/*name_SPF*
7. /_N_CMA_DIR/*name_SPF*
8. /_N_CST_DIR/*name_SPF*

Supplementary conditions

- CALLPATH checks whether the programmed path name actually exists. In the case of an error, part program execution is interrupted with correction block alarm 14009.
- CALLPATH can also be programmed in INI files. It is only effective for the time it takes to process the INI file (WPD-INI file or initialization program for NC active data, e.g. frames in the 1st channel _N_CH1_UFR_INI). The search path is again reset.

4.2.3.10 Execute external subroutine (EXTCALL)

A part program can be loaded from an external memory and executed with the EXTCALL command.

The following are available as external memory:

- User CF card
- Network drive
- USB drive

Note

As the interface for the execution of an external program located on a USB drive, only the USB interface of the operator panel front (PPU) may be used.

NOTICE
Tool/workpiece damage when using a USB flash drive
It is recommended that a USB flash drive is not used to execute an external subprogram. A communication interruption to the USB flash drive while executing the subprogram as a result of contact problems, drop out, interruption through a knock or accidental unplugging stops the machining immediately. The tool and/or workpiece could be damaged.

Default setting of the external program path

The path for the external program directory can be preset with the setting data:

SD42700 $SC_EXT_PROG_PATH

Together with the program path and identifier specified with the EXTCALL call, this forms the entire path for the subprogram to be called.

Note

If the program path is specified only via the EXTCALL call, then SD42700 must be empty.

Note

Parameters

When an external program is called, no parameters can be transferred to it.

Syntax

```
EXTCALL("<Path/><Program name>")
```

Meaning

EXTCALL:	Command for calling an external subprogram.	
"<Path/><Program name>":	Constant/variable of type STRING	
	<Path/>:	Absolute or relative path specification (**optional**)
	<Program name>:	The program name is specified without prefix "_N_". The file extensions ("MPF", "SPF") can be attached to program names using the "_" or "." character (**optional**). Example: "SHAFT" "SHAFT_SPF" "SHAFT.SPF"

Path specification: Short designations

The following short designations can be used to specify the path:

- User CF card: "**CF_CARD:**"
- USB drive (operator panel front): "**USB:**"

Example

The "MAIN.MPF" main program is stored in the NC memory and is selected for execution.

Subprogram "SP_1"

The external subprogram "SP_1.SPF " or "SP_1.MPF " is on the user CF card in the "/WKS.DIR/WST1.WPD" directory.

The path for the external program directory is set with:

SD42700 $SC_EXT_PROG_PATH = CF_CARD:WKS.DIR/WST1.WPD

Note

Specification of the path for calling the external subprogram:
- Without the default setting: "CF_CARD:WKS.DIR/WST1.WPD/SP_1"
- With the default setting: "SP_1"

Subprogram "SP_2"

The external subprogram "SP_2.SPF " or "SP_2.MPF " is in the WKS.DIR/WST1.WPD directory of the USB drive. The default setting of the path to the external program directory is used for the path of subprogram "SP_1" and is also not rewritten in the main program. Therefore, the complete path has to be specified when subprogram "SP_2" is called.

Main program "MAIN"

Program code
N010 PROC MAIN

Program code
`N020 ...`
`N030 EXTCALL("SP_1")`
`N030 EXTCALL("USB:WKS.DIR/WST1.WPD/SP_2")`
`N050 ...`
`N060 M30`

More information

EXTCALL call with absolute path name

If the subprogram exists under the specified path, it is executed with the EXTCALL call. If the subprogram does not exist under the specified path, the program execution is aborted with the EXTCALL call.

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:

1. 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 is obtained from linking the following characters:
   - Default path specification in SD42700 $SC_EXT_PROG_PATH
   - Separator "/"
   - Path specification and subprogram name in the EXTCALL command
2. If the subprogram was not found under 1., the directories of the user memory are searched.

The search ends when the subprogram is found for the first time. If the subprogram is not found, the program execution is aborted with the EXTCALL call.

Adjustable reload memory (FIFO buffer)

A reload memory is required for the execution of an external subprogram. The size of the reload memory is preset (see MD18360 MM_EXT_PROG_BUFFER_SIZE).

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 reload memory.

This condition is especially problematic regarding jump instructions to the start of the program (GOTOS), since the programs are typically much too large to fit entirely in the reload memory. When reloading for the first time, the start of the program is removed from the reload memory. If a jump instruction to the beginning of the program is executed, the function is no longer able to find the jump destination. The program is aborted and alarm 14000 is output.

To be able to execute external subprograms without restrictions with regard to programmed jump instructions, it is recommended to use the function "Execution from External Storage (EES)" instead of the function "Execution from external subprograms (EXTCALL)".

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.

A separate reload memory is required for external subprograms executed in parallel.

Reset / end of program / POWER ON

Reset and POWER ON cause external subprogram calls to be interrupted and the associated load memory to be deleted.

A program selected for "Execution from external source" remains selected for "Execution from external source" after a reset / end of program. The behavior does not differ from internally selected programs, assuming that the external program memory is still available.

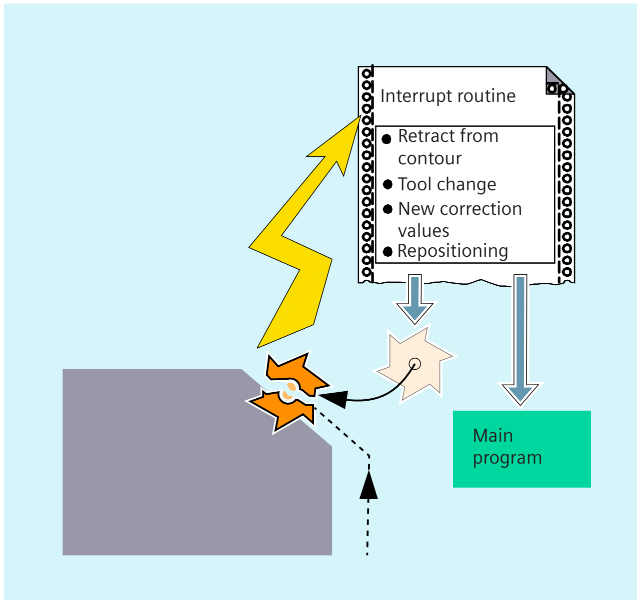
4.3 Interrupt routine (ASUB)

4.3.1 Function of an interrupt routine

Note

The terms "asynchronous subprogram (ASUB)" and "interrupt routine" are used interchangeably in the description below to refer to the same functionality.

A typical example should clarify the function of an interrupt routine:



The tool breaks during machining. This triggers a signal that stops the current machining process and simultaneously starts a subprogram – known as an interrupt routine. The interrupt routine contains all the statements which are to be executed in this case.

When the interrupt routine has finished being executed and the machine is ready to continue operation, the control jumps back to the main program and continues machining at the point of interruption – depending on the REPOS command (see "Repositioning at the contour (Page 770)").

> ⚠ **CAUTION**
>
> **Risk of collision**
>
> If a REPOS command has not been programmed in the subprogram, then the control goes to the end point of the block that follows the interrupted block.

4.3.2 Creating an interrupt routine

Create interrupt routine as subprogram

The interrupt routine is identified as a subprogram in the definition.

Example:

Program code	Comment
PROC LIFT_Z	; Program name "ABHEB_Z"

Program code	Comment
N10 ...	; The NC blocks then follow:
...	
N50 M17	; Finally, end the program and return to the main program.

Saving modal G commands (SAVE)

The interrupt routine can be designated by defining with SAVE.

The SAVE attribute means that the active modal G commands are saved before calling the interrupt routine and are reactivated after the end of the interrupt routine (see " Subprograms with SAVE mechanism (SAVE) (Page 492) ").

This means that it is possible to resume processing at the interruption point after the interrupt routine has been completed.

Example:

Program code
PROC LIFT_Z SAVE
N10 ...
...
N50 M17

Assign additional interrupt routines (SETINT)

SETINT statements can be programmed within the interrupt routine (see "Assign and start interrupt routine (SETINT)" (Page 531)) therefore activating additional interrupt routines. They are triggered via the input.

4.3.3 Assign and start interrupt routine (SETINT, PRIO, BLSYNC)

The control has several fast inputs (inputs 1 ... 8), which initiate an interrupt (1 ... 8). Each interrupt can be assigned a priority and an interrupt routine using the `SETINT` command. If the interrupt is initiated by setting the fast input, then processing in the channel is interrupted and the interrupt routine started.

Interrupt priority

If, in a part program, several inputs are assigned interrupts, then the interrupts must be assigned different priorities.

An interrupt can be assigned a priority value from 1 ... 128. Priority value 1 corresponds to the highest priority and 128 the lowest.

Syntax

```
SETINT(<n>) <NAME>
SETINT(<n>) PRIO=<value> <NAME>
SETINT(<n>) PRIO=<value> <NAME> BLSYNC
SETINT(<n>) PRIO=<value> <NAME> LIFTFAST
```

Meaning

`SETINT(<n>):`	Input <n> is assigned the interrupt routine <Name>. The assigned interrupt routine is started as soon as input <n> == 1 is detected. **Note:** If an already programmed input <n> is assigned another interrupt routine, then the previous assignment is no longer effective.	
`<n>:`	Input number	
	Type:	INT
	Range of values:	1 ... 8
`PRIO= :`	Priority of the interrupt (optional)	
`<value>:`	Priority value (optional)	
	Type:	INT
	Range of values:	1 ... 128 (1 ⇒ highest priority)
`<NAME>:`	Name of the interrupt routine (subprogram)	
`BLSYNC:`	`BLSYNC` ensures that after initiating the interrupt, the system first waits until the actual block has been completed. Only then is the interrupt routine executed. (optional)	
`LIFTFAST:`	`LIFTFAST` ensures that after initiating the interrupt, initially a fast retraction is realized (see Chapter "Fast retraction from the contour (SETINT LIFTFAST, ALF) (Page 535)"). Only then is the interrupt routine executed. (optional)	

Supplementary conditions

Interrupt rules

1. For every interrupt that cannot be immediately executed, or is presently already being processed, an additional interrupt request is saved. All other interrupt requests for this interrupt are lost.
2. If an interrupt is currently being processed and an additional interrupt with higher priority initiated, then this interrupts the lower-priority interrupt. The lower priority interrupt is continued after the higher priority interrupt has been completed. If, while the higher priority interrupt is being processed, additional requests are received for the lower-priority interrupt, then one request is saved. All others are lost.
3. If an interrupt is currently being processed and an additional interrupt with higher priority initiated, then this interrupts the lower-priority interrupt. The higher priority interrupt is processed. If a higher priority interrupt is initiated, the actual interrupt is interrupted and the higher priority interrupt processed. A maximum of six active interrupt levels are possible. One interrupt level presently being processed and five waiting interrupt levels. For each active interrupt level, a maximum of one additional interrupt request is saved. All other interrupt requests are lost. Interrupt requests are also lost if these are requested for additional interrupt levels (interrupt level ≥ 7).

Examples

Example 1: Assign interrupt routines and define the priority

Program code	Comment
...	
N20 SETINT(3) PRIO=1 ABHEB_Z	; IF input 3 == 1 THEN start interrupt routine "ABHEB_Z"
N30 SETINT(2) PRIO=2 ABHEB_X	; IF input 2 == 1 THEN start interrupt routine "ABHEB_X".
...	

The interrupt routines are executed in the sequence of the priority values if the inputs become available simultaneously (are energized simultaneously): First "ABHEB_Z", then "ABHEB_X".

Example 2: Newly assign an interrupt routine

Program code	Comment
...	
N20 SETINT(3) PRIO=2 ABHEB_Z	; IF input 3 == 1 THEN start interrupt routine **"ABHEB_Z"**
...	
N80 SETINT(3) PRIO=1 ABHEB_X	; IF input 3 == 1 THEN start interrupt routine **"ABHEB_X"**
...	

4.3.4 Deactivating/reactivating the assignment of an interrupt routine (DISABLE, ENABLE)

A `SETINT` statement can be deactivated with `DISABLE` and reactivated with `ENABLE` without losing the input → interrupt routine assignment.

Syntax

```
DISABLE(<n>)
ENABLE(<n>)
```

Meaning

DISABLE(<n>):	Command: **Deactivating** the interrupt routine assignment of input <n>	
ENABLE(<n>):	Command: **Reactivating** the interrupt routine assignment of input <n>	
<n>:	Parameter: Number of the interrupt signal	
	Type:	INT
	Range of values:	1 ... 32

Example

Program code	Comment
N20 SETINT(3) PRIO=1 ABHEB_Z	; If input 3 switches, then interrupt
	; routine "ABHEB_Z" should start.
...	
N90 DISABLE(3)	; The SETINT statement from N20 is deactivated.
...	
N130 ENABLE(3)	; The SETINT statement from N20 is reactivated.
...	

4.3.5 Delete assignment of interrupt routine (CLRINT)

An interrupt signal assignment defined with `SETINT` for an NC program (ASUP) can be deleted with `CLRINT`.

Syntax

```
CLRINT(<n>)
```

Meaning

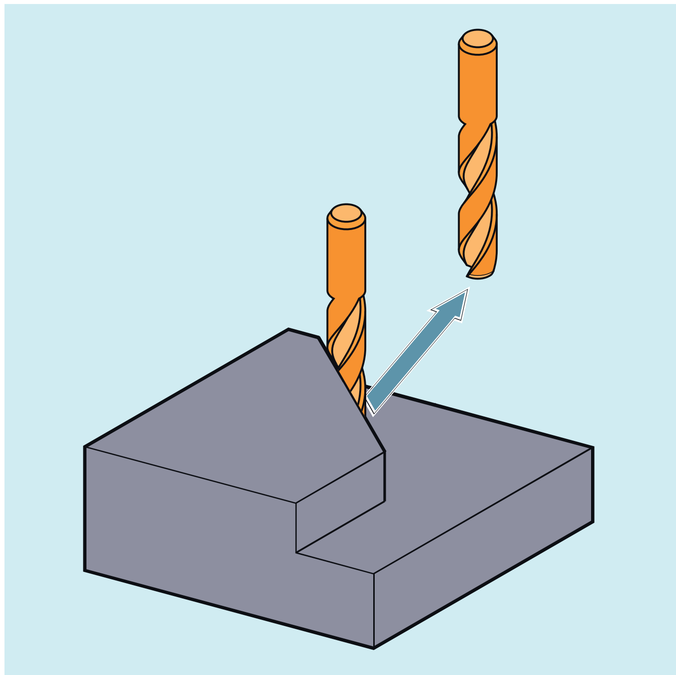
`CLRINT(<n>)`:	Command: Delete assignment of the interrupt signal <n> to the NC program (ASUP) defined with SETINT <n>	
`<n>`:	Parameter: Number of the interrupt signal	
	Type:	INT
	Range of values:	1 ... 32

Example

Program code	Comment
N20 SETINT(3) PRIO=2 ABHEB_Z	
...	
N50 CLRINT(3)	; The assignment between input "3" and interrupt routine "ABHEB_Z" is deleted.

4.3.6 Fast retraction from the contour (SETINT LIFTFAST, ALF)

For a `SETINT` statement with `LIFTFAST`, when the input is switched, the tool is moved away from the workpiece contour using fast retraction.



The further sequence is then dependent on whether the `SETINT` statement includes an interrupt routine in addition to `LIFTFAST`:

With interrupt routine:	**After** the fast retraction, the interrupt routine is executed.
Without interrupt routine:	Machining is stopped after fast retraction and an alarm is output.

Syntax

```
SETINT(<n>) PRIO=1 LIFTFAST
SETINT(<n>) PRIO=1 <NAME> LIFTFAST
```

Meaning

`SETINT(<n>):`	Command: Assign input <n> to an interrupt routine. The assigned interrupt routine starts when input <n> switches.	
`<n>:`	Parameter: Input number	
	Type:	INT
	Range of values:	1 ... 8
`PRIO= :`	Defining the priority	
`<value>:`	Priority value	
	Range of values:	1 ... 128
	Priority 1 corresponds to the highest priority.	

<NAME>:	Name of the subprogram (interrupt routine) that is to be executed.
LIFTFAST:	Command: Fast retraction from the contour
ALF=… :	Command: Programmable traverse direction (in motion block) Regarding the possibilities of programming with ALF, refer to the subject "Traversing direction for fast retraction from the contour (Page 537)".

Supplementary conditions

Behavior for active frame with mirroring

When determining the retraction direction, a check is performed to see whether a frame with mirror is active. In this case, for the retraction direction, right and left are interchanged referred to the tangential direction. The direction components in tool direction are not mirrored. This behavior is activated with the MD setting:

MD21202 $MC_LIFTFAST_WITH_MIRROR = TRUE

Example

A broken tool should be automatically replaced by a daughter tool. Machining is then continued with the new tool.

Main program:

Main program	Comment
N10 SETINT(1) PRIO=1 W_WECHS LIFTFAST	; When input 1 is switched, the tool is immediately retracted from the contour with fast retraction (code no. 7 for tool radius compensation G41). Then interrupt routine "W_WECHS" is executed.
N20 G0 Z100 G17 T1 ALF=7 D1	
N30 G0 X-5 Y-22 Z2 M3 S300	
N40 Z-7	
N50 G41 G1 X16 Y16 F200	
N60 Y35	
N70 X53 Y65	
N90 X71.5 Y16	
N100 X16	
N110 G40 G0 Z100 M30	

Subprogram:

Subprogram	Comment
PROC W_CHANGE SAVE	; Subprogram where the actual operating state is saved
N10 G0 Z100 M5	;Tool changing position, spindle stop
N20 T11 M6 D1 G41	;Change tool

Subprogram	Comment
N30 REPOSL RMBBL M3	; Reposition at the contour and return jump into the main program (this is programmed in a block)

4.3.7 Traversing direction for fast retraction from the contour

Retraction movement

The following G commands define the retraction movement plane:

- LFTXT
  The retraction movement plane is defined by the path tangent and the tool direction (default setting).
- LFWP
  The plane of the retraction movement is the active working plane selected with G commands G17, G18 or G19. The direction of the retraction movement is not dependent on the path tangent. This allows a fast retraction to be programmed parallel to the axis.
- LFPOS
  Retraction of the axis declared using POLFMASK/POLFMLIN to the absolute axis position programmed with POLF.
  ALF has no influence on the retraction direction for several axes and for several axes in a linear system.

Programmable traversing direction (ALF=...)

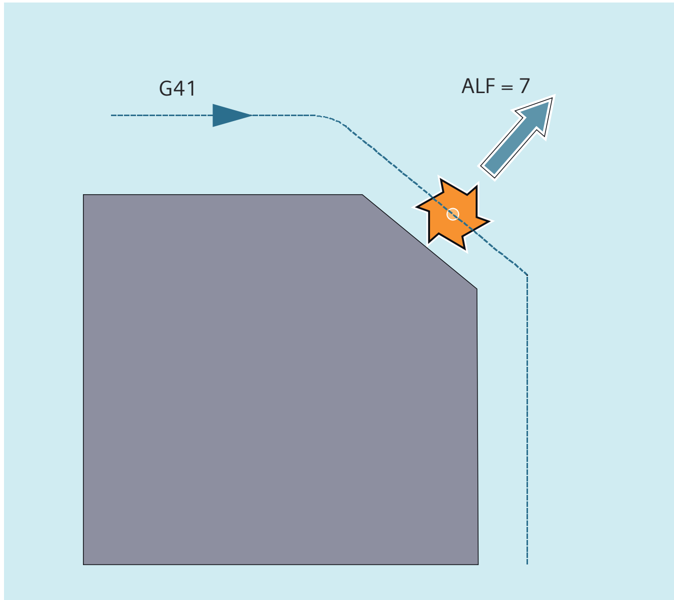
The direction is programmed in discrete steps of 45 degrees with ALF in the plane of the retraction movement.

The possible traversing directions are stored in special code numbers on the control and can be called up using these numbers.

Example:

Program code
N10 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST
ALF=7

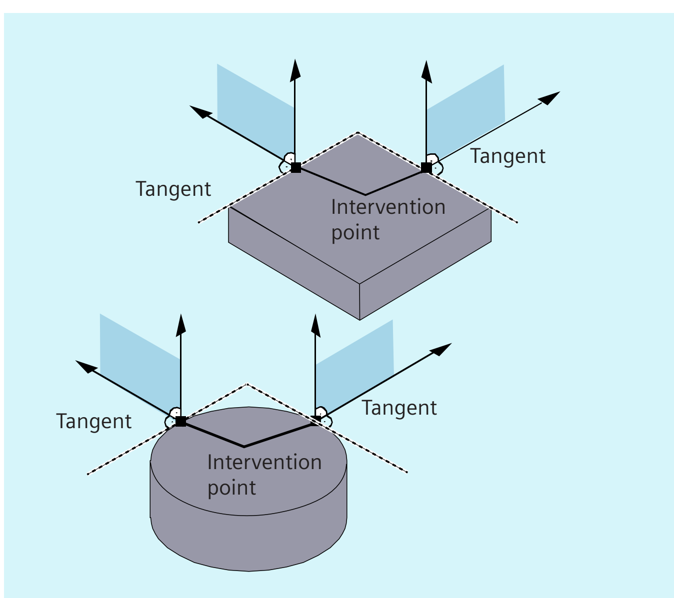
With G41 activated (machining direction to the left of the contour) the tool vertically moves away from the contour.



Reference plane for defining the traversing direction for LFTXT

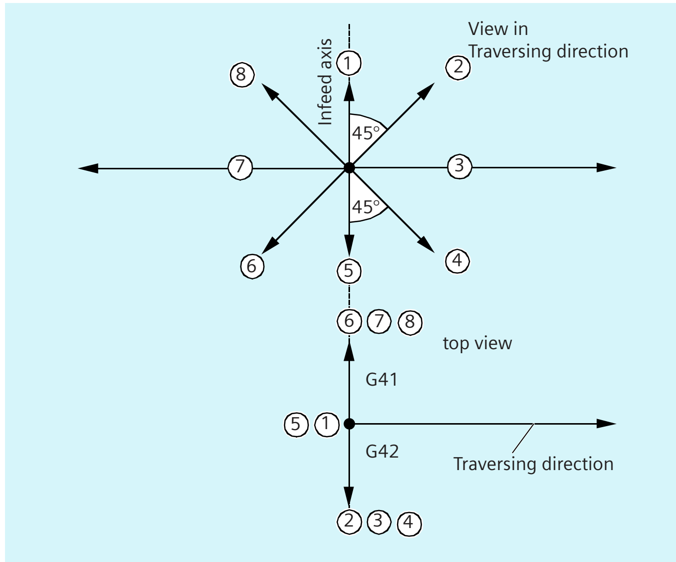
At the point of application of the tool to the programmed contour, the tool is clamped at a plane which is used as a reference for specifying the retraction movement with the corresponding code number.

The reference plane is derived from the longitudinal tool axis (infeed direction) and a vector positioned perpendicular to this axis and perpendicular to the tangent at the point of application of the tool.



Code numbers with traversing direction for LFTXT

Starting from the reference plane, you will find the code numbers with traversing directions in the following diagram.



The retraction in the tool direction is defined for `ALF=1`.

The "fast retraction" function is deactivated with `ALF=0`.

CAUTION

Risk of collision

When the tool radius compensation is activated, then:

- For `G41` codes 2, 3, 4
- For `G42` codes 6, 7, 8

should not be used, as in these cases, the tool would move to the contour and would collide with the workpiece.

Code numbers with traversing directions for LFWP

For `LFWP`, the direction in the machining plane is derived from 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

4.3.8 Motion sequence for interrupt routines

Interrupt routine without LIFTFAST

Axis motion is braked along the path down to standstill (zero speed). The interrupt routine then starts.

The standstill position is saved as interrupt position and is approached at the end of the interrupt routine for `REPOS` with `RMIBL`.

Interrupt routine with LIFTFAST

Axis motion is braked along the path. The `LIFTFAST` motion is simultaneously executed as superimposed motion. If the path motion and `LIFTFAST` motion have come to a standstill (zero speed), the interrupt routine is started.

The position on the contour is saved as interrupt position where the `LIFTFAST` motion is started and therefore the path was left.

The interrupt routine with `LIFTFAST` and `ALF=0` behaves in precisely the same way as the interrupt routine without `LIFTFAST`.

Note

The absolute value through which the geometry axes move when quickly retracting from the contour can be set using machine data.

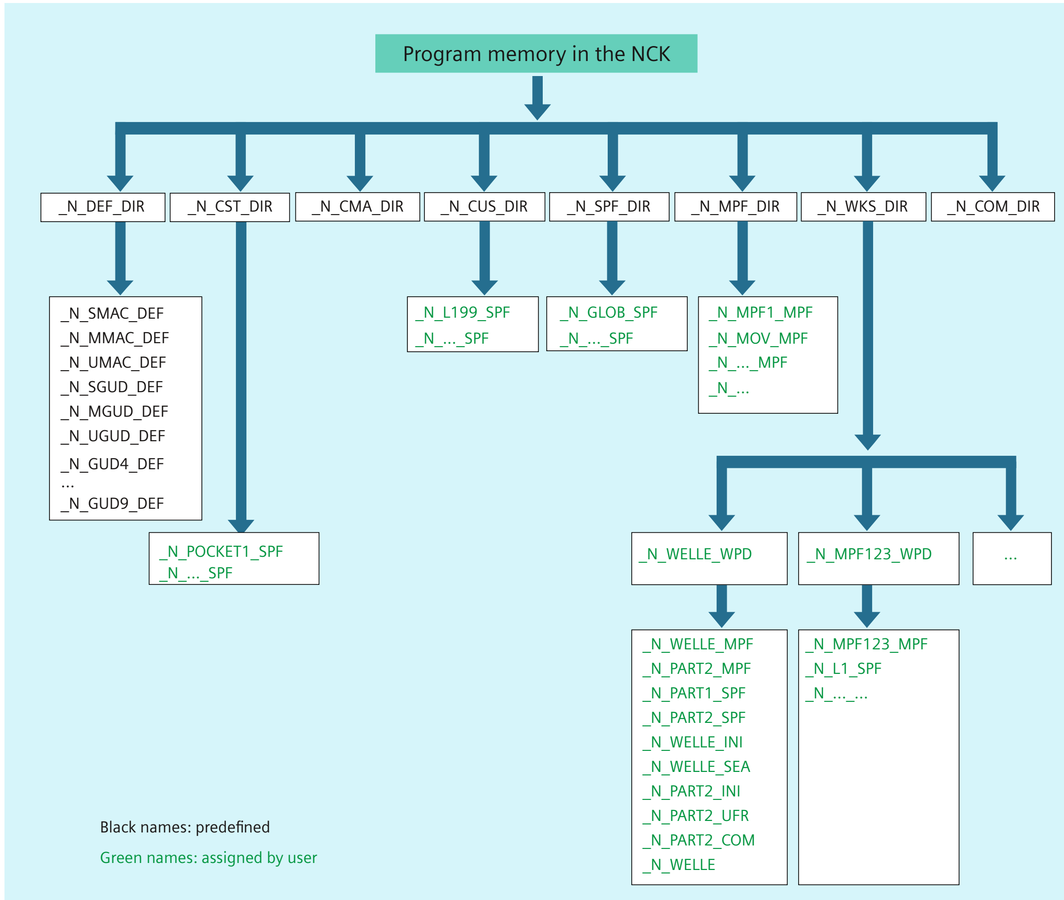
4.4 File and Program Management

4.4.1 Program memory

4.4.1.1 Program memory in the NCK

Files and programs (e.g. main programs and subprograms, macro definitions) are saved in the non-volatile program memory (→ passive file system).

A number of file types are also stored here temporarily; these can be transferred to the work memory as required (e.g. for initialization purposes when machining a specific workpiece).



Standard directories

The following standard directories are available:

Directory	Content
_N_DEF_DIR	Data modules and macro modules
_N_CST_DIR	Standard cycles
_N_CMA_DIR	Manufacturer cycles
_N_CUS_DIR	User cycles
_N_WKS_DIR	Workpieces
_N_SPF_DIR	Global subprograms
_N_MPF_DIR	Main programs
_N_COM_DIR	Comments

File types

The following file types can be stored in the main memory:

File type	Description
<name>_MPF	Main program
<name>_SPF	Subprogram
<name>_TEA	Machine data
<name>_SEA	Setting data
<name>_TOA	Tool offsets
<name>_UFR	Work offsets/frame
<name>_INI	Initialization files
<name>_GUD	Global user data
<name>_RPA	R parameters
<name>_COM	Comment
<name>_DEF	Definitions for global user data and macros

Workpiece main directory (_N_WKS_DIR)

The workpiece main directory exists in the standard setup of the program memory under the name _N_WKS_DIR. The workpiece main directory contains all the workpiece directories for the workpieces that you have programmed.

Workpiece directories (..._WPD)

A workpiece directory contains all files required for machining a workpiece. These can be main programs, subprograms, any initialization programs and comment files.

The first time a part program is started, initialization programs are executed once, depending on the selected program (in accordance with machine data MD11280 $MN_WPD_INI_MODE).

Example:

The workpiece directory _N_SHAFT_WPD, created for SHAFT workpiece contains the following files:

File	Description
_N_SHAFT_MPF	Main program
_N_PART2_MPF	Main program
_N_PART1_SPF	Subprogram
_N_PART2_SPF	Subprogram
_N_SHAFT_INI	General initialization program for the data of the workpiece
_N_SHAFT_SEA	Setting data initialization program
_N_PART2_INI	General initialization program for the data for the Part 2 program
_N_PART2_UFR	Initialization program for the frame data for the Part 2 program
_N_SHAFT_COM	Comment file

Data can also be stored in the workpiece directory which is not directly required by the NC for the machining. In addition to ASCII files, this can be binary files, such as images in JPG format or descriptions in PDF format. In order that these can be interpreted as binary files by the NC, the file extensions must be known in the NC (setting during commissioning via MD17000 $MN_ EXTENSIONS_OF_BIN_FILES; the following file extensions are preset in the basic setting: JPG, GIF, PNG, BMP, PDF, ICO, HTM).

Select workpiece for machining

A workpiece directory can be selected for execution in a channel. If a main program with the **same name** or only a single main program (_MPF) is stored in this directory, this is automatically selected for execution.

4.4.1.2 External program memory

In addition to the passive file system in the NC, external program memories can also be available at the machine (e.g. on the local drive or on a network drive).

Using the functions "Execute from external" or "EES (Execution from External Storage)" part programs can be **directly** executed from external program memories.

Further information: Function Manual Basic Functions

Global part program memory (GDIR)

When declaring the drives, one of the drives can be designated the global part program memory (GDIR).

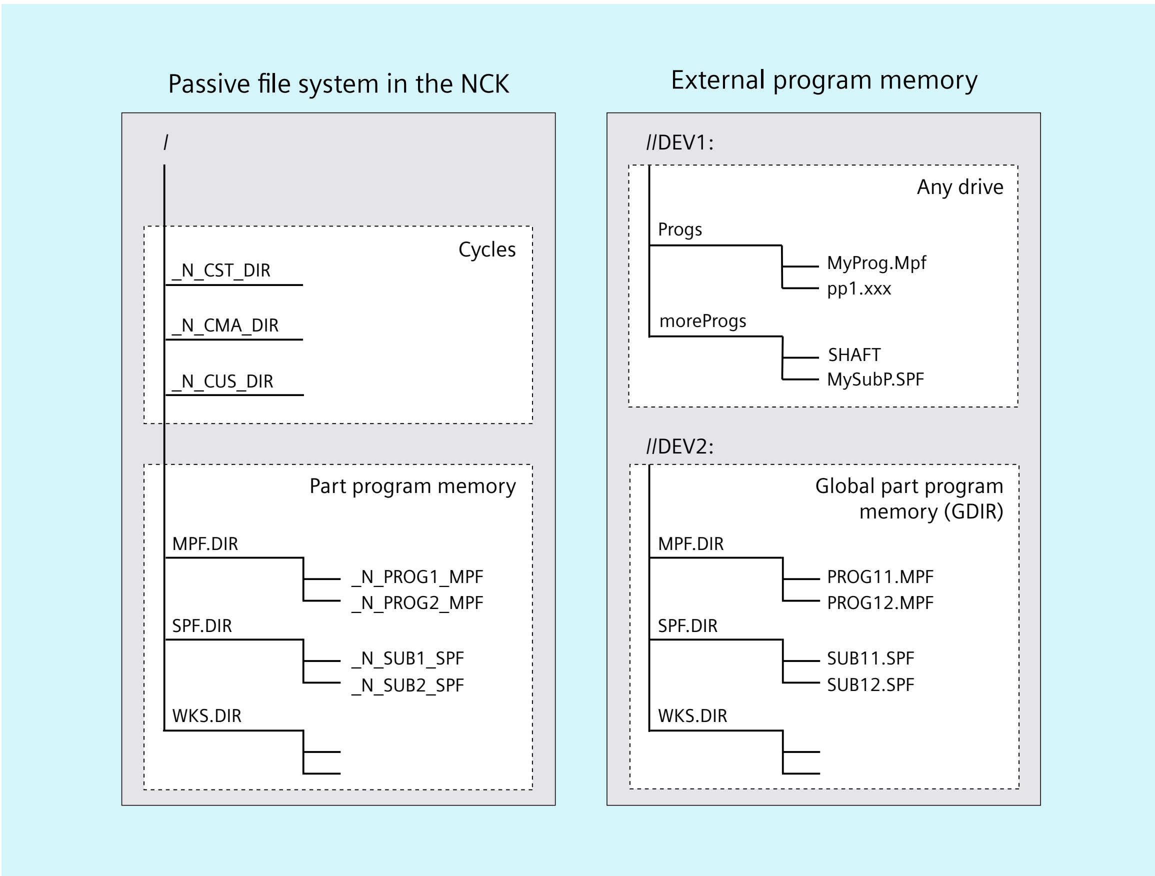
The system automatically creates the MPF.DIR, SPF.DIR and WKS.DIR directories on the drive. These three directories form the GDIR.

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. The creation of a GDIR is, however, not essential for 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. This permits a compatible transfer of an NC program with path details from the passive file system to the GDIR. The directory SPF.DIR of the GDIR is contained in the search path for subprograms.

Program organization

The program organization on external program memories is shown in the following diagram:



Case-insensitive file systems

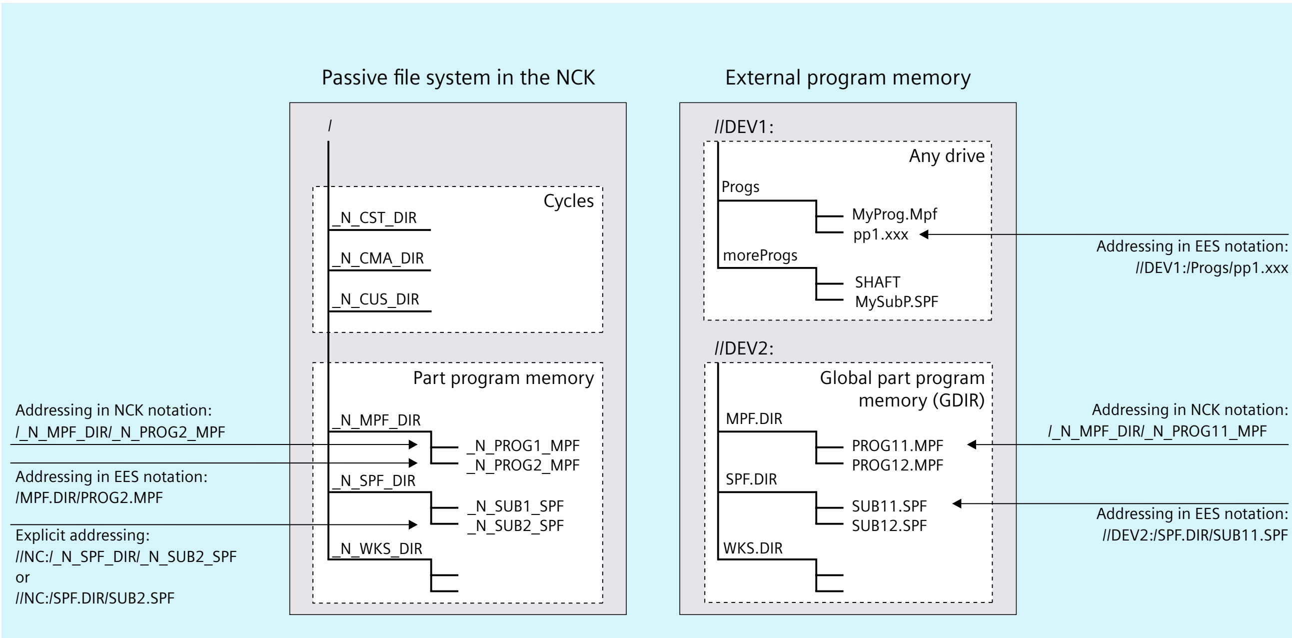
Note

To avoid problems with case-sensitivity for the file addressing (see "Addressing program memory files (Page 544)"), **case-insensitive** file systems should be used as external program memory.

4.4.1.3 Addressing program memory files

A file in the program memory, which is addressed with a file handling command (Page 554) (e.g. WRITE, DELETE, READ, ISFILE, FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO), is referenced with an absolute path plus file names or only with the file names. In the second case, the path of the selected program is used as file path.

Addressing in the NC/EES notation



Addressing files of the passive file system

Files of the passive file system are generally addressed in the **NC notation** (directory and file names begin with the domain identifier "_N_", "_" is the separator for the file identifier) without specifying the drive name. An addressing in **EES notation** (without domain identification "_N_", separator for the directory/file extension is ".") is, however, also permitted.

Example:

- NC notation: "/_N_SPF_DIR/_N_SUB1_SPF"
- EES notation: "/SPF.DIR/SUB1.SPF"

Note

The addressing schemes for files of the passive file system in EES notation are converted internally into NC notation in accordance with the following rules:

- Directory and file names are extended with the domain identification "_N_".
- If the fourth-last character in the directory or file name is a period ("."), it will be converted into an underscore ("_").

The passive file system can also be explicitly addressed using the predefined drive names "//NC:".

Example:

- NC notation: "//NC:/_N_SPF_DIR/_N_SUB1_SPF"
- EES notation: "//NC:/SPF.DIR/SUB1.SPF"

Addressing files of an external program memory

Files of an external program memory not recorded as GDIR must be addressed in EES notation. The drive name (e.g. "//DEV1:") must be specified at the start of the addressing path. All symbolic device names configured in /user/sinumerik/hmi/cfg/logdrive.ini are permissible.

Example:

- EES notation: "//DEV1:/MyProgDir/pp1.xxx"
- NC notation: Not permissible

Addressing files of the global part program memory (GDIR)

When addressing files of the GDIR, in addition to specifying the path in the EES notation, it is also permissible to specify the path in the NC notation.

Example:

- EES notation: "//DEV2:/MPF.DIR/PROG11.MPF"
- NC notation: "/_N_MPF_DIR/_N_PROG11_MPF"

Note

The addressing schemes for files of the GDIR in NC notation are converted internally into EES notation in accordance with the following rules:

- The domain identification "_N_" in directory and file names is removed.
- If the fourth-last character in the directory or file name is an underscore ("_"), it will be converted into a period (".").

Rules for the path specification

A complete path specification consists of drive name, directory path and file name.

Drive name

The following rules govern the specification of the drive name:

- All symbolic device names configured in /user/sinumerik/hmi/cfg/logdrive.ini are permissible.
- The character "//" is at the beginning, followed by at least one letter or one digit.
- The following characters can be any combination of letters, digits, "_" and spaces.
- The name is ended with a letter or a digit, followed by a ":".
- Other special characters are not permitted.

Note

The drive name "//NC:" is predefined for the passive file system.

Examples:

- External program memory:
  - //Drive1:
  - //Drive_1:
  - //Drive 1:
  - //A B:
  - //1 B C 2:

Directory path

The following rules govern the specification of the directory path:

- A "/" is located at the start and end of the directory path and as separator for the individual path sections.

  Note

  A double slash ("//") within the directory path is **not** permitted!

- Directory names:
  - Directory names must begin with a letter or a digit. Only for addressing in the NC notation do directory names begin with the domain identification "_N_".
  - The following characters can be any combination of letters, digits and "_".

    Note

    Spaces in directory names are also permitted for external program memories. This is not true, however, when the external program memory is created as global part program memory (GDIR).

  - Other special characters are not permitted.
- Directory extensions:
  - Directory extensions must consist of three letters/digits.
  - They are separated with "_" (NC notation) or "." (EES notation) from the directory name.

  Note

  The passive file system has only the directory extensions _DIR and _WPD.

Examples:

- Passive file system or GDIR:
  - NC notation: _N_WKS_DIR/_N_MYNCPROGS_WPD/...
  - EES notation: WKS.DIR/MYPROGS.WPD/...
- External program memory:
  - /abc
  - /ab_c.def
  - /ab c1.def
  - /a b c .d11
  - /abc.def/ghi.klm

File name

The following rules apply to the file names:

- Only for addressing in NC notation do file names begin with the domain identification "_N_".
- The next two characters should be either two letters or an underscore followed by a letter.

  Note

  If this condition is satisfied, then an NC program can be called as subprogram from another program just by specifying the program name. However, if the program name starts with digits, the subprogram call is then only possible via the CALL statement.

- The following characters can be any combination of letters, digits and "_".
- File extension:
  - The file extension must consist of three letters/digits.

    Note

    Permitted file extensions in the passive file system, see "Program memory in the NCK (Page 540)".

  - They are separated with "_" (NC notation) or "." (EES notation) from the file names.

Examples:

- Passive file system or GDIR:
  - NC notation: _N_SUB1_SPF
  - EES notation: SUB1.SPF
- External program memory:
  - Part1
  - _Part1
  - Part_1.spf
  - Part1.mpf

DIN subprogram name

The following rules apply to DIN subprogram names:

- The first character must be the letter "L".
- The following characters are digits (at least one).
- File extension:
  – The file extension must consist of three letters.
  – They are separated with "_" (NC notation) or "." (EES notation) from the file names.

Examples:

- L123
- L1_SPF (NC notation) or L1.SPF (EES notation)

Maximum path length

Maximum 128 bytes are available for specifying the drive name and the directory path; the maximum length of the file name is 31 bytes. The maximum length of the complete path is 159 bytes.

4.4.1.4 Search path for subprogram call

For subprogram calls without path data, the absolute path is determined by processing a fixed search path.

A search is then made in the program memory in the following sequence:

		Directory	Description
1		current directory / *name*	The current directory is the directory in which the program is selected. This can be: • A workpiece directory or the standard directory _N_MPF_DIR in the NC part program memory or global part program memory or • Any directory of an external program memory
2		current directory / *name_SPF*	
3		current directory / *name_MPF*	
4	a	//NC:/_N_SPF_DIR / *name_SPF*	Subprogram directory in the NC part program memory
	b	//DEV2:/_N_SPF_DIR / name_SPF [1]	Subprogram directory in the global part program memory **Note:** This search step is not executed if a global part program memory has not been created, or the program is selected in the NC part program memory.
5		Search path extension programmed with `CALLPATH` (see "Extend search path for subprogram calls (CALLPATH) (Page 524)"). **Note:** This search step is not executed if `CALLPATH` has not been programmed.	
6		/_N_CUS_DIR / *name_SPF*	User cycle directory

	Directory	Description
7	/_N_CMA_DIR / *name_SPF*	Manufacturer cycle directory
8	/_N_CST_DIR / *name_SPF*	Standard cycle directory

1) //DEV2:" For example represents the drive on which the global part program memory has been created.

The following rules apply for the search:

- The search path is run through for each individual subprogram call, this means that it is irrelevant where the higher-level program is located.
- Depending on the directory, different file types are taken into account.
- A search is made in a directory, and not in lower-level, i.e. nested directories.

4.4.1.5 Interrogating the path and file name

The following system variables, which can be read in the part program, are available to interrogate the path and file name of an NC program:

System variable	Type	Meaning
$P_STACK	INT	Supplies the program level in which the current NC program is executed.
$P_PATH[<n>]	STRING	Supplies the path of the NC program, which is processed at the program level selected using field index <n>. Examples: $P_PATH[0] supplies the path for the main program, e.g. "/_N_WKS_DIR/_N_WELLE_WPD/". $P_PATH[$P_STACK - 1] supplies the path of the calling program. If the path refers to an NC program, which is saved in the passive file system of the NC or in the global part program memory (GDIR), then the path is supplied in the NC notation. If the path refers to an NC program, which is executed by an external program memory other than the global part program memory then $P_PATH supplies the path in the EES notation.
$P_PROG[<n>]	STRING	Supplies the name of the NC program, which is processed at the program level selected using field index <n>. If the NC program is saved in the passive file system of the NC or in the global part program memory, then the program name is supplied in the NC notation. If the NC program is executed by an external drive other than the global part program memory, then $P_PROG supplies the name in the EES notation.
$P_PROGPATH	STRING	Supplies the path of the NC program that is presently being processed. Calling $P_PROGPATH is identical to $P_PATH[$P_STACK].

System variable	Type	Meaning	
$P_IS_EES_PATH[<n>]	BOOL	Interrogates whether the path supplied by $P_PATH[<n>] or the program name supplied by $P_PROG[<n>] corresponds to the NC notation or the EES notation.	
		= FALSE	$P_PATH[<n>] and $P_PROG[<n>] supply a NC notation. This means that each identifier has the prefix "_N_". The separator for the file identifier is "_". Examples: • Path in the NC notation: "/_N_WKS_DIR/_N_MYWPD_WPD/" • Program name in the NC notation:"_N_MYPROG_MPF" A path in the NC notation can refer to the passive file system in the NC as well as also the global part program memory.
		= TRUE	$P_PATH[<n>] and $P_PROG[<n>] supply an EES notation. This means that the identifiers do not have the "_N_" prefix. The separator for the file identifier is ".". Examples: • Path in the EES notation: "//DEV1:/WKS.DIR/MYWPD.WPD/" • Program name in the EES notation: "MYPROG.MPF"

<n>: Index <n> defines the program level, from which the path information should be read (value range: 0 ... 17)

Note

In the EES mode, outside the global part program memory (GDIR), system variables $P_PROG, $P_PATH and $P_PROGPATH path names in the EES notation. For the EES mode, user programs that evaluate and process these path names must be extended so that they can also process pathnames in the EES notation.

4.4.2 Working memory (CHANDATA, COMPLETE, INITIAL)

Function

The working memory contains the current system and user data with which the control is operated (active file system), e.g.:

- Active machine data
- Tool offset data
- Zero offsets
- ...

Initialization programs

These are programs with which the working memory data is initialized. The following file types can be used for this:

File type	Description
name_TEA	Machine data
name_SEA	Setting data
name_TOA	Tool offsets
name_UFR	Zero offsets/frames
name_INI	Initialization files
name_GUD	Global user data
name_RPA	R-parameters

Data areas

The data can be organized in different areas in which they are to apply. For example, a control can have several channels or, as is commonly the case, several axes at its disposal.

There are:

Identifier	Data areas
NC	NC-specific data
CH<n>	Channel-specific data (<n> specifies the channel name)
AX<n>	Axis-specific data (<n> specifies the number of the machine axis)
TO	Tool data
COMPLETE	All data

Create initialization program at an external PC

The data area identifier and the data type identifier can be used to determine the areas which are to be treated as a unit when the data is saved:

_N_AX5_TEA_INI	Machine data for axis 5
_N_CH2_UFR_INI	Frames of channel 2
_N_COMPLETE_TEA_INI	All machine data

When the control is started up initially, a set of data is automatically loaded to ensure proper operation of the control.

Procedure for multi-channel controls (CHANDATA)

`CHANDATA(<channel number>)` for several channels is only permissible in the file _N_INITIAL_INI. This is the commissioning file with which all data of the control is initialized.

Program code	Comment
`%_N_INITIAL_INI`	
`CHANDATA(1)`	

Program code	Comment
	; Machine axis assignment, channel 1:
`$MC_AXCONF_MACHAX_USED[0]=1`	
`$MC_AXCONF_MACHAX_USED[1]=2`	
`$MC_AXCONF_MACHAX_USED[2]=3`	
`CHANDATA(2)`	
	; Machine axis assignment, channel 2:
`$MC_AXCONF_MACHAX_USED[0]=4`	
`$MC_AXCONF_MACHAX_USED[1]=5`	
`CHANDATA(1)`	
	; Axial machine data:
	; Exact stop window coarse:
`$MA_STOP_LIMIT_COARSE[AX1]=0.2`	; Axis 1
`$MA_STOP_LIMIT_COARSE[AX2]=0.2`	; Axis 2
	; Exact stop window fine:
`$MA_STOP_LIMIT_FINE[AX1]=0.01`	; Axis 1
`$MA_STOP_LIMIT_FINE[AX1]=0.01`	; Axis 2

NOTICE

CHANDATA statement

In the part program, the `CHANDATA` statement may only be set for that channel in which the NC program is executed. This means the statement can be used to protect NC programs so that they are not executed in the wrong channel.

Program processing is aborted if an error occurs.

Note

INI files in job lists do not contain any `CHANDATA` statements.

Save initialization program (COMPLETE, INITIAL)

The files of the working memory can be saved on an external PC and then read in again from there.

- The files are saved with `COMPLETE`.
- `INITIAL` is used to create an INI file (_N_INITIAL_INI) over all areas.

Read-in initialization program

NOTICE
Data loss If the file is read-in with the name "INITIAL_INI", then all data that is not supplied in the file is initialized using standard data. Only machine data is an exception. This means that **setting data, tool data, ZO, GUD values, ...** are supplied with standard data (normally "ZERO").

For example, the file COMPLETE_TEA_INI is suitable for reading-in individual machine data. The control only expects machine data in this file. This is the reason that the other data areas remain unaffected in this case.

Loading initialization programs

The INI programs can also be selected and called as part programs if they only use data of one channel. This means that it is also possible to initialize program-controlled data.

4.5 File handling

4.5.1 Write file (WRITE)

The WRITE command is used to write blocks/data from the NC program at the end of a file (log file) in the passive file system or to external program memory. This can also be the program that is presently being executed. If the file to be written to does not exist in the program memory specified as the target directory, a new file is created.

Requirements

Protection level

The currently set protection level must be equal to or greater than the WRITE right of the file. If this is not the case, access is denied with an error message (return value of error variable = 13).

Access to EES drives

External program memory on drives for which use with the function "Execution from External Storage (EES)" was set up during commissioning, can only be addressed by file handling commands if the license-only option to use this function is set.

Syntax

```
DEF INT <Error>
...
WRITE(<Error>,"<FileName>"/"<ExtG>","<Set/Data>")
```

Meaning

WRITE	Command for appending a block or data to the end of the specified file.		
<Error>	**Parameter 1:** Variable for returning the error value.		
	Type:	INT	
	Value:	0	No error
		1	Path not allowed
		2	Path not found
		3	File not found
		4	Incorrect file type
		10	File is full
		11	The file is in use
		12	No resources available
		13	No access rights
		14	Missing or unsuccessful EXTOPEN for the output device
		15	Error when writing to an external device
		16	Invalid external path has been programmed
<FileName>	**Parameter 2:** The name of the file in which the specified block or specified data is to be added.		
	Type:	STRING	
	The absolute path can be specified before the actual file name. If a path is not specified, the file is searched for in the current directory (= directory of selected program). Rules regarding path data, see "Addressing program memory files (Page 544)".		
<ExtG>	If the data is to be output to an external device/file using the "Process DataShare" function, then the symbolic identifiers for the external device/file to be opened must be specified instead of the file name.		
	Type:	STRING	
	For further information, see "Process DataShare - output to an external device/file (EXTOPEN, WRITE, EXTCLOSE) (Page 866)". **Note:** The identifier must be identical to the identifier specified in the EXTOPEN command.		
<Set/Data>:	**Parameter 3:** The block or data to be added to the specified file.		
	Type:	STRING	

Note

When writing to the passive file system or to an external program memory, the WRITE command implicitly inserts an "LF" character (LINE FEED = new line) at the end of the output string.

This behavior does not apply for output to an external device/file using the "Process DataShare" function. If an "LF" is also to be output, then this must be explicitly specified in the output string.

→ also refer to example 3: Implicit/explicit "LF"!

Constraints

Maximum file size (→ machine manufacturer)

The maximum possible file size of log files in the passive file system is set with the machine data:

MD11420 $MN_LEN_PROTOCOL_FILE

The maximum file length is applicable for all files created using the WRITE command in the passive file system. If it is exceeded, an error message is output and the block or data is not saved. If there is sufficient free memory, a new file can be created.

Examples

Example 1: WRITE command into the passive file system without absolute path data

Program code	Comment
N10 DEF INT ERROR	; Definition of error variables.
N20 WRITE(ERROR,"PROT","LOG FROM 7.2.97")	; Write the text "LOG FROM 7.2.97" to file _N_PROT_MPF.
N30 IF ERROR	; Error evaluation.
N40 MSG ("Error with WRITE command:" << ERROR)	
N50 M0	
N60 ENDIF	
...	

Example 2: WRITE command into the passive file system with absolute path data

```
Program code
...
WRITE(ERROR,"/_N_WKS_DIR/_N_PROT_WPD/_N_PROT_MPF","LOG FROM 7.2.97")
...
```

Example 3: Implicit/explicit "LF"

a) Write to the passive file system with implicitly generated "LF"

```
Program code
...
N110 DEF INT ERROR
N120 WRITE(ERROR,"/_N_MPF_DIR/_N_MYPROTFILE_MPF","MY_STRING")
N130 WRITE(ERROR,"/_N_MPF_DIR/_N_MYPROTFILE_MPF","MY_STRING")
N140 M30
```

Output result:

MY_STRING

MY_STRING

b) Write to an external file without implicitly generated "LF"

```
Program code
...
N200 DEF STRING[30] DEV_1
N210 DEF INT ERROR
N220 DEV_1="LOCAL_DRIVE/myprotfile.mpf"
N230 EXTOPEN(ERROR,DEV_1)
N240 WRITE(ERROR,DEV_1,"MY_STRING")
N250 WRITE(ERROR,DEV_1,"MY_STRING")
N260 EXTCLOSE(ERROR,DEV_1)
N270 M30
```

Output result:

MY_STRINGMY_STRING

c) Write to an external file with explicitly generated "LF"

The following must be programmed in order to achieve the same result as under a:

```
Program code
...
N200 DEF STRING[30] DEV_1
N210 DEF INT ERROR
N220 DEV_1="LOCAL_DRIVE/myprotfile.mpf"
N230 EXTOPEN(ERROR,DEV_1)
N240 WRITE(ERROR,DEV_1,"MY_STRING'H0A'")
N250 WRITE(ERROR,DEV_1,"MY_STRING'H0A'")
N260 EXTCLOSE(ERROR,DEV_1)
N270 M30
```

Output result:

MY_STRING

MY_STRING

4.5.2 Delete file (DELETE)

All files can be deleted by means of the DELETE command, irrespective of whether these were created using the WRITE command or not. Files that were created using a higher access level can also be deleted with DELETE.

Requirements

Access to EES drives

External program memory on drives for which use with the function "Execution from External Storage (EES)" was set up during commissioning, can only be addressed by file handling commands if the license-only option to use this function is set.

Syntax

```
DEF INT <Error>
DELETE(<Error>,"<FileName>")
```

Meaning

<table>
<tr><td>DELETE</td><td colspan="3">Command for deleting the specified file.</td></tr>
<tr><td rowspan="10"><Error></td><td colspan="3">Variable for returning the error value.</td></tr>
<tr><td>Type.</td><td colspan="2">INT</td></tr>
<tr><td rowspan="8">Value:</td><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>Path not allowed</td></tr>
<tr><td>2</td><td>Path not found</td></tr>
<tr><td>3</td><td>File not found</td></tr>
<tr><td>4</td><td>Incorrect file type</td></tr>
<tr><td>11</td><td>The file is in use</td></tr>
<tr><td>12</td><td>No resources available</td></tr>
<tr><td>20</td><td>Other error</td></tr>
<tr><td rowspan="3"><FileName></td><td colspan="3">Name of the file to be deleted</td></tr>
<tr><td>Type:</td><td colspan="2">STRING</td></tr>
<tr><td colspan="3">The absolute path can be specified before the actual file name. If a path is not specified, the file is searched for in the current directory (= directory of selected program).
Rules regarding path data, see "Addressing program memory files (Page 544)".</td></tr>
</table>

Example

Program code	Comment
`N10 DEF INT ERROR`	`; Definition of error variables.`
`N15 STOPRE`	`; Preprocessing stop.`
`N20 DELETE(ERROR,"/_N_SPF_DIR/_N_TEST1_SPF")`	`; Deletes file TEST1 in the sub-program directory.`
`N30 IF ERROR`	`; Error evaluation.`
`N40 MSG("Error for DELETE command:" <<ERROR)`	
`N50 M0`	
`N60 ENDIF`	

4.5.3 Read lines in the file (READ)

The READ command reads one or several lines in the file specified and stores the information read in an array of type STRING. In this array, each read line occupies an array element.

Requirements

Protection level

The currently set protection level must be equal to or greater than the READ right of the file. If this is not the case, access is denied with an error message (return value of error variable = 13).

Access to EES drives

External program memory on drives for which use with the function "Execution from External Storage (EES)" was set up during commissioning, can only be addressed by file handling commands if the license-only option to use this function is set.

Syntax

```
DEF INT <Error>
DEF STRING[<StringLength>] <Result>[<n>,<m>]
READ(<Error>,"<FileName>",<StartLine>,<NumLines>,<Result>)
```

Meaning

`READ`	Command for reading lines from the specified file and storing these lines in a variable array.		
`<Error>`	Variable for returning the error value (call-by-reference parameter)		
	Type.	INT	
	Value:	0	No error
		1	Path not allowed
		2	Path not found
		3	File not found
		4	Incorrect file type
		11	The file is in use
		13	Insufficient access rights
		21	Line does not exist (<StartLine> or <NumLines> parameter exceeds the number of lines in the specified file).
		22	Array length of the result variable (<Result>) is too small.
		23	Line range too large (<NumLines> parameter selected so large that reading would go beyond the end of the file).
`<FileName>`	Name of the file to be read (call-by-value parameter)		
	Type:	STRING	
	The absolute path can be specified before the actual file name. If a path is not specified, the file is searched for in the current directory (= directory of selected program). Rules regarding path data, see "Addressing program memory files (Page 544)".		

`<StartLine>`	Start line of the file section to be read (call-by-value parameter)		
	Type:	INT	
	Value:	0	Reads the number of lines specified with the <NumLines> parameter before the end of the file.
		1 to n	Number of the first line to be read.
`<NumLines>`	Number of lines to be read (call-by-value parameter)		
	Type:	INT	
`<Result>`	Result variable (call-by-reference parameter) Variable array in which the read text is stored.		
	Type:	STRING (max. length: 255)	
	If fewer lines are specified in the <NumLines> parameter than the array size [<n>,<m>] of the result variable, the remaining array elements will not be modified. Termination of a line by means of the control characters "LF" (Line Feed) or "CR LF" (Carriage Return Line Feed) is **not** stored in the result variable. Read lines are cropped if the line is longer than the defined string length. An error message is not output.		

Note

Binary files cannot be read in. The "incorrect data type" error is output (return value of the error variable = 4). The following file types are not readable: _BIN, _EXE, _OBJ, _LIB, _BOT, _TRC, _ACC, _CYC, _NCK.

Example

Program code	Comment
`N10 DEF INT ERROR`	`; Definition of error variables.`
`N20 DEF STRING[255] RESULT[5]`	`; Definition of result variables.`
`N30 READ(ERROR,"/_N_CST_DIR/_N_TEST-FILE_MPF",1,5,RESULT)`	`;File name with domain and file identifier` `and path name.`
`N40 IF ERROR <>0`	`; Error evaluation.`
`N50 MSG("ERROR"<<ERROR<<"ON READ COMMAND")`	
`N60 M0`	
`N70 ENDIF`	
`...`	

4.5.4 Check for presence of file (ISFILE)

The ISFILE command checks whether a file exists in the program memory.

Requirements

Access to EES drives

External program memory on drives for which use with the function "Execution from External Storage (EES)" was set up during commissioning, can only be addressed by file handling commands if the license-only option to use this function is set.

Syntax

```
<Result>=ISFILE("<FileName>")
```

Meaning

<table>
<tr><td>ISFILE</td><td colspan="3">Command to check the availability of a file</td></tr>
<tr><td rowspan="4"><FileName></td><td colspan="3">Name of the file whose availability is to be checked.</td></tr>
<tr><td>Type:</td><td colspan="2">STRING</td></tr>
<tr><td colspan="3">The absolute path can be specified before the actual file name. If a path is not specified, the file is searched for in the current directory (= directory of selected program).
Rules regarding path data, see "Addressing program memory files (Page 544)".</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="4"><Result></td><td colspan="3">Result variable to which the result of the check is assigned.</td></tr>
<tr><td>Type.</td><td colspan="2">BOOL</td></tr>
<tr><td rowspan="2">Value:</td><td>TRUE</td><td>File exists</td></tr>
<tr><td>FALSE</td><td>File does not exist</td></tr>
</table>

Examples

Example 1

Program code	Comment
`N10 DEF BOOL RESULT`	`; Definition of result variables.`
`N20 RESULT=ISFILE("TESTFILE")`	
`N30 IF(RESULT==FALSE)`	
`N40    MSG("FILE DOES NOT EXIST")`	
`N50    M0`	
`N60 ENDIF`	
`...`	

Example 2

Program code	Comment
`N10 DEF BOOL RESULT`	`; Definition of result variables.`
`N20 RESULT=ISFILE("TESTFILE")`	
`N30 IF(NOT ISFILE("TESTFILE"))`	
`N40    MSG("FILE DOES NOT EXIST")`	
`N50    M0`	
`N60 ENDIF`	

Program code	Comment
...	

4.5.5 Read out file information (FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO)

The FILEDATE, FILETIME, FILESIZE, FILESTAT, and FILEINFO commands read out specific file information such as date/time of the last write access, current file size, file status or all of this information.

Requirements

Protection level

The currently set protection level must be equal to or greater than the show right of the superordinate directory. If this is not the case, access is denied with an error message (return value of error variable = 13).

Access to EES drives

External program memory on drives for which use with the function "Execution from External Storage (EES)" was set up during commissioning, can only be addressed by file handling commands if the license-only option to use this function is set.

Syntax

```
FILEx(<Error>,"<FileName>",<Result>)
```

Meaning

FILEDATE	Returns the **date** of the last write access to a file
FILETIME	Returns the **time** of the last write access to a file
FILESIZE	Returns the **current size** of a file
FILESTAT	Returns a file with regard to the following **rights** for the **status**: • Read (r: read) • Write (w: write) • Execute (x: execute) • Show (s: show) • Delete (d: delete) **Note:** These protection levels are specific properties of the passive file system. When accessing an external program memory, FILESTAT therefore supplies default access rights (77777).
FILEINFO	Returns the **sum of the information** for a file that can be read out via FILEDATE, FILETIME, FILESIZE and FILESTAT

<Error>	Variable for returning the error value (call-by-reference parameter)			
	Type.	VAR INT		
	Value:	0	No error	
		1	Path not allowed	
		2	Path not found	
		3	File not found	
		4	Incorrect file type	
		13	Insufficient access rights	
		22	String length of the result variable (<Result>) is too small.	
<FileName>	Name of the file from which the file information is to be read out			
	Type:	CHAR[160]		
	The absolute path can be specified before the actual file name. If a path is not specified, the file is searched for in the current directory (= directory of selected program). Rules regarding path data, see "Addressing program memory files (Page 544)".			
<Result>	Result variable (Call-By-Reference parameter) Variable in which the requested file information is stored.			
	Type:	VAR CHAR[8]	at	FILEDATE Format: "dd.mm.yy"
		VAR CHAR[8]	at	FILETIME Format: "hh.mm.ss"
		VAR INT	at	FILESIZE The file size is output in bytes.
		VAR CHAR[5]	at	FILESTAT Format: "rwxsd" (r: **r**ead, w: **w**rite, x: e**x**ecute, s: **s**how, d: **d**elete)
		VAR CHAR[32]	at	FILEINFO Format: "rwxsd nnnnnnnn dd.mm.yy hh:mm:ss"

Example

Program code	Comment
`N10 DEF INT ERROR`	; Definition of error variables.
`N20 STRING[32] RESULT`	; Definition of result variables.
`N30 FILEINFO(ERROR,"/_N_MPF_DIR/_N_TESTFILE_MPF",RESULT)`	; File name with domain, file ID and path data.
`N40 IF ERROR <> 0`	; Error evaluation
`N50    MSG("ERROR"<<ERROR<<"ON FILEINFO COMMAND")`	
`N60    M0`	
`N70 ENDIF`	
...	

In the result variables RESULT, the example could supply the following result:

"77777 12345678 26.05.00 13:51:30"

4.6 Protection zones

4.6.1 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 567)".

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

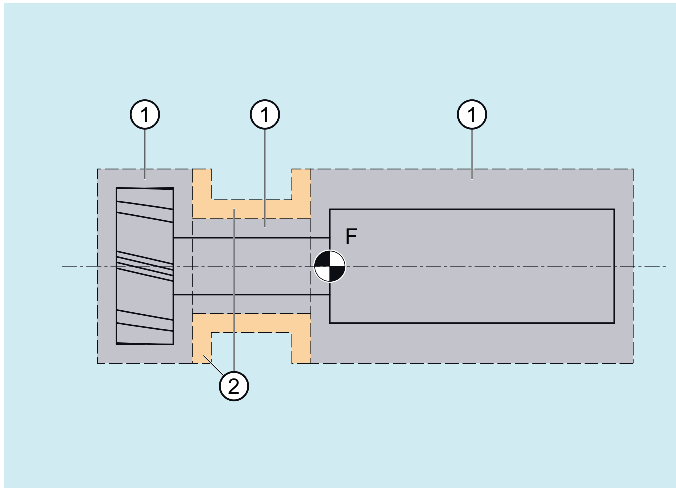
- Tool-related protection zones
  Coordinates for **tool**-related protection zones must be specified as absolute values referred to the **tool holder reference point F**.
- Workpiece-related protection zones
  Coordinates for **workpiece**-related protection zones must be specified as absolute values 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`

4.6.2 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

<table>
<tr><td>CPROT:</td><td colspan="3">Predefined procedure to activate a channel-specific protection zone</td></tr>
<tr><td>NPROT:</td><td colspan="3">Predefined procedure to activate a machine-specific protection zone</td></tr>
<tr><td rowspan="2"><n>:</td><td colspan="3">Number of the protection zone</td></tr>
<tr><td>Data type:</td><td colspan="2">INT</td></tr>
<tr><td rowspan="6"><Status>:</td><td colspan="3">The channel-specific activation status is set using this parameter</td></tr>
<tr><td>Data type:</td><td colspan="2">INT</td></tr>
<tr><td rowspan="4">Value:</td><td>0</td><td>Deactivate protection zone</td></tr>
<tr><td>1</td><td>Preactivate protection zone</td></tr>
<tr><td>2</td><td>Activate protection zone</td></tr>
<tr><td>3</td><td>Preactivate protection zone with conditional stop</td></tr>
<tr><td rowspan="2"><XMov>,<YMov>,<ZMov>:</td><td colspan="3">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</td></tr>
<tr><td>Data type:</td><td colspan="2">REAL</td></tr>
</table>

Example

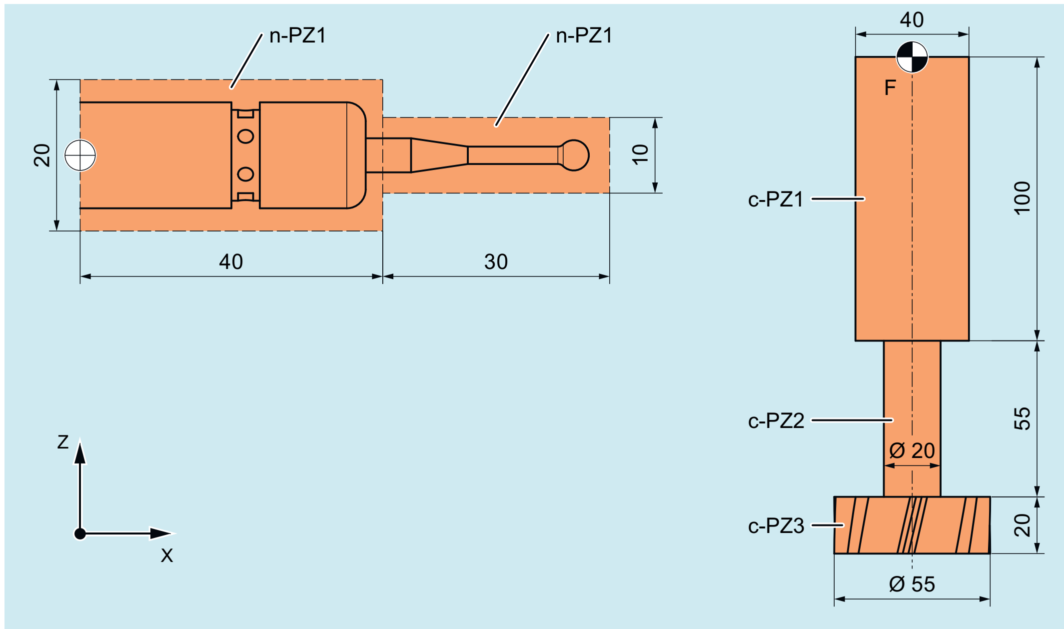
Possible collision of a milling cutter with the measuring probe is to be monitored on a milling machine. The position of the measuring probe is to be defined by an offset when the function is activated.

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`	

Program code	Comment
Y20	
X-20	
Y-20	
EXECUTE(PROTZONE)	
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.

4.6.3 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

<table>
<tr><td rowspan="14"><status>:
(Part 1)</td><td colspan="2">Function return value. Negative values indicate error states.</td></tr>
<tr><td>Data type:</td><td>INT</td></tr>
<tr><td>Value range:</td><td>-8 ≤ x ≤ 100000</td></tr>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>0</td><td>The distance can be traversed completely.</td></tr>
<tr><td>-1</td><td>At least one component is negative in <Limit>.</td></tr>
<tr><td>-2</td><td>Error in a transformation calculation.
Example: The traversing distance passes through a singularity so that the axis positions cannot be defined.</td></tr>
<tr><td>-3</td><td>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.</td></tr>
<tr><td>-4</td><td>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.</td></tr>
<tr><td>-5</td><td>In <TestLim>, bit 4 == 1 AND bit 5 == 1</td></tr>
<tr><td>-6</td><td>At least one machine axis that has to be considered for checking the traversing limits has not been referenced.</td></tr>
<tr><td>-7</td><td>Collision avoidance function: Invalid definition of the kinematic chain or the protection zones.</td></tr>
<tr><td>-8</td><td>Collision avoidance function: This command cannot be executed because of insufficient memory.</td></tr>
<tr><td colspan="2" style="display:none"></td></tr>
<tr><td rowspan="11"><status>:
(Part 2)</td><td colspan="2">Units digit
Note
If several limits are violated simultaneously, the limit with the greatest restriction on the specified traversing distance is signaled.</td></tr>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Software limit switches are limiting the traversing distance</td></tr>
<tr><td>2</td><td>Working area limits are limiting the traversing distance</td></tr>
<tr><td>3</td><td>Protection areas are limiting the traversing path</td></tr>
<tr><td>4</td><td>Collision avoidance function: Protection areas are limiting the traversing path</td></tr>
<tr><td colspan="2">Tens digit</td></tr>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1x</td><td>The initial value violates the limit</td></tr>
<tr><td>2x</td><td>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).</td></tr>
</table>

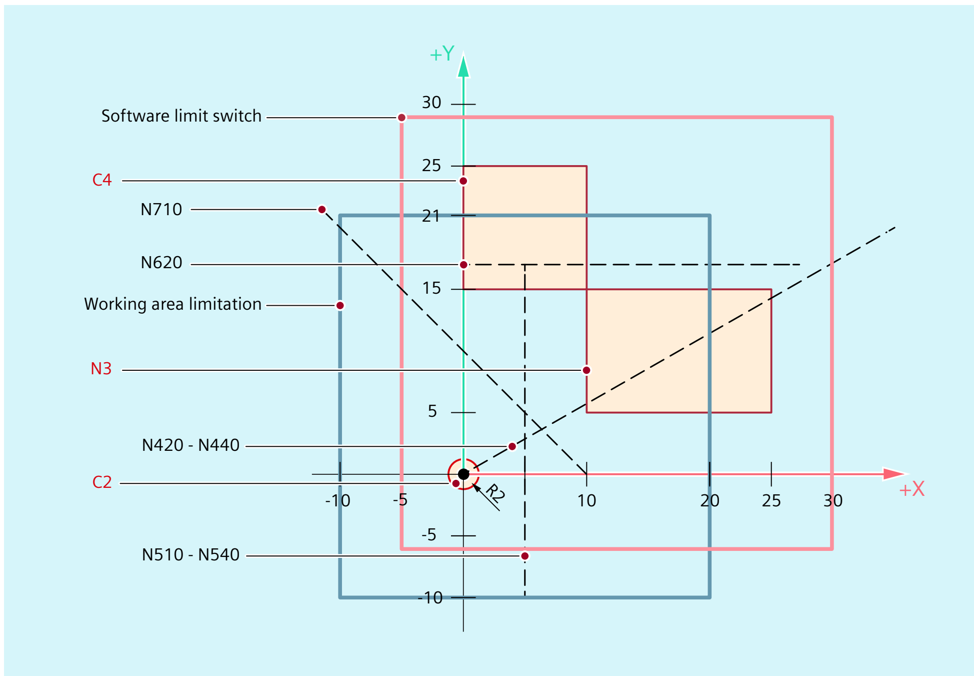
<status>: (Part 3)	Hundreds digit	
	Value	**Meaning**
	1xx	AND units digit == 1 or 2: The positive limit value has been violated.
		AND units digit == 3 [1)]: An NC-specific protection area has been violated.
	2xx	AND units digit == 1 or 2: The negative limit value has been violated.
		AND units digit == 3 [1)]: A channel-specific protection zone is violated.
<status>: (Part 4)	Thousands digit	
	Value	**Meaning**
	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	**Meaning**
	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	**Meaning**
	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:	Bit 0, 1, 2, 3, 6, 7 = 1 (207)
	Value	**Meaning**
	1	Software limit switch
	2	Working area limitation
	4	Activated conventional protection zones
	8	Preactivated conventional protection zones
	16	With violated software limit switches or working area limits in <Dist>, return the traversing direction as in **Case 1** (see above).
	32	With violated software limit switches or working area limits in <Dist>, return the traversing direction as in **Case 2** (see above).
	64	Activated collision avoidance protection zones
	128	Preactivated collision avoidance protection zones
	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
: 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)
```

Program code

```
; 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.
N540 _STATUS = CALCPOSI(_START, _DIST, _LIMIT, _MAXDIST,, 6)
N550 _START[0] = 27.
N560 _START[1] = 17.1
N570 _START[2] =  0.
```

Program code

```
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.
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.

N...	<status>	<MaxDist>[0] ≙ X	<MaxDist>[1] ≙ Y	Remarks
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>.

More 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 DB380x.DBX1000.2 / 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.

4.7 Special motion commands

4.7.1 Approaching coded positions (CAC, CIC, CDC, CACP, CACN)

Using the path commands to "Approach coded positions", it is possible to traverse linear and rotary axes to fixed axis positions saved in the machine data table by specifying position numbers.

Syntax

```
CAC(<n>)
CIC(<n>)
CACP(<n>)
CACN(<n>)
```

Meaning

CAC(<n>):	Approach coded position from position number n
CIC(<n>):	Starting from the actual position number, approach the coded position n position locations before (+n) or back (−n)
CDC(<n>):	Approach the position from position number n along the shortest path (only for rotary axes)
CACP(<n>):	Approach coded position from position number n in the positive direction (only for rotary axes)
CACN(<n>):	Approach coded position from position number n in the negative direction (only for rotary axes)
<n>:	Position number within the machine data table Value range: 0, 1, ... (max. number of table locations - 1)

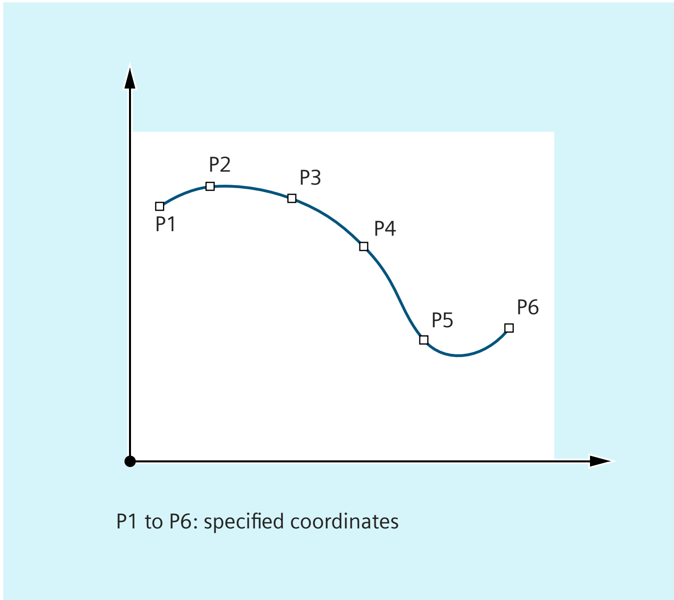
Example: Approach coded positions of a positioning axis

Programming code	Comment
N10 FA[B]=300	; Feedrate for positioning axis B
N20 POS[B]=CAC(10)	; Approach coded position from position number 10
N30 POS[B]=CIC(-4)	; Approach coded position from "current position number" - 4

4.7.2 Spline interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL)

Randomly curved workpiece contours cannot be precisely defined in an analytic form. This is the reason why these type of contours are approximated using a limited number of points along curves, e.g. when digitizing surfaces. The points along the curve must be connected to define a contour in order to generate the digitized surface of a workpiece. Spline interpolation permits this.

A spline defines a curve which is formed from polynomials of 2nd or 3rd degree. The characteristics of the points along the curve of a spline can be defined **depending on the spline type being used**.



For SINUMERIK solution line, the following spline types are available:

- A spline
- B spline
- C spline

Syntax

General:
```
ASPLINE X... Y... Z... A... B... C...
BSPLINE X... Y... Z... A... B... C...
CSPLINE X... Y... Z... A... B... C...
```

For a B spline, the following can be additionally programmed:
```
PW=<n>
SD=2
PL=<value>
```

For A and C splines, the following can be additionally programmed:

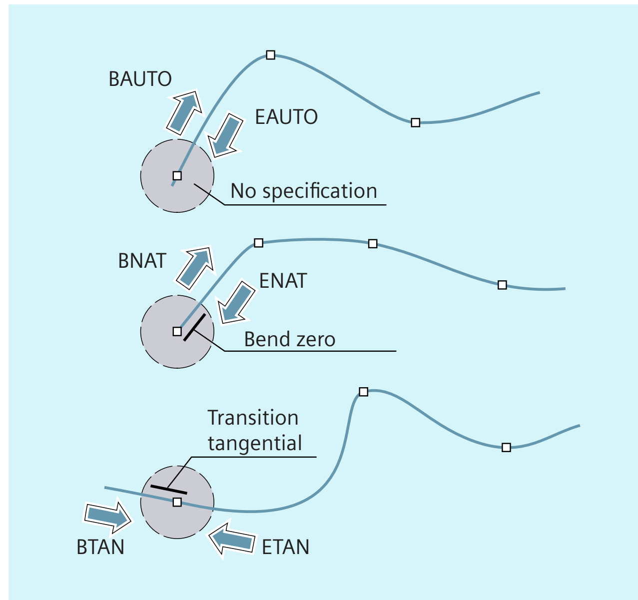
`BAUTO` / `BNAT` / `BTAN`

`EAUTO` / `ENAT` / `ETAN`

Meaning

Spline interpolation type:	
`ASPLINE`:	Command to activate A spline interpolation
`BSPLINE`:	Command to activate B spline interpolation
`CSPLINE`:	Command to activate C spline interpolation
	The `ASPLINE`, `BSPLINE` and `CSPLINE` commands are modally effective and belong to the group of motion commands.

Points along a curve and check points:			
`X... Y... Z...` `A... B... C...`	Positions in Cartesian coordinates		
Point weight (only B spline):			
`PW`:	Using the `PW` command, a so-called "point weight" can be programmed for every point along the curve.		
`<n>`:	"Point weight"		
	Range of values:	$0 \leq n \leq 3$	
	Increment:	0.0001	
	Effect:	$n > 1$	The checkpoint attracts the curve more significantly.
		$n < 1$	The checkpoint attracts the curve less significantly.
Spline degree (only B spline):			
`SD`:	A third degree polygon is used as standard, However, a second degree polygon can also be used by programming `SD=2`.		
Distance between nodes (only B spline):			
`PL` :	The distances between nodes are suitably calculated internally. The control can also machine pre-defined node clearances that are specified in the so-called parameter-interval-length using the `PL` command.		
`<value>`:	Parameter interval length		
	Range of values:	As for path dimension	
Transitional behavior at the start of the spline curve (only A or C spline):			
`BAUTO`:	No specifications for the transitional behavior. The start is determined by the position of the first point.		
`BNAT`:	Zero curvature		
`BTAN`:	Tangential transition to the previous block (delete position)		
Transitional behavior at the end of the spline curve (only A or C spline):			
`EAUTO`:	No specifications for the transitional behavior. The end is determined by the position of the last point.		
`ENAT`:	Zero curvature		

`ETAN`:	Tangential transition to the previous block (delete position)
	

Note

The programmable transitional behavior has no influence on the B spline. The B spline is always tangential to the check polygon at its start and end points.

Supplementary conditions

- Tool radius compensation may be used.
- Collision monitoring is carried out in the projection in the plane.

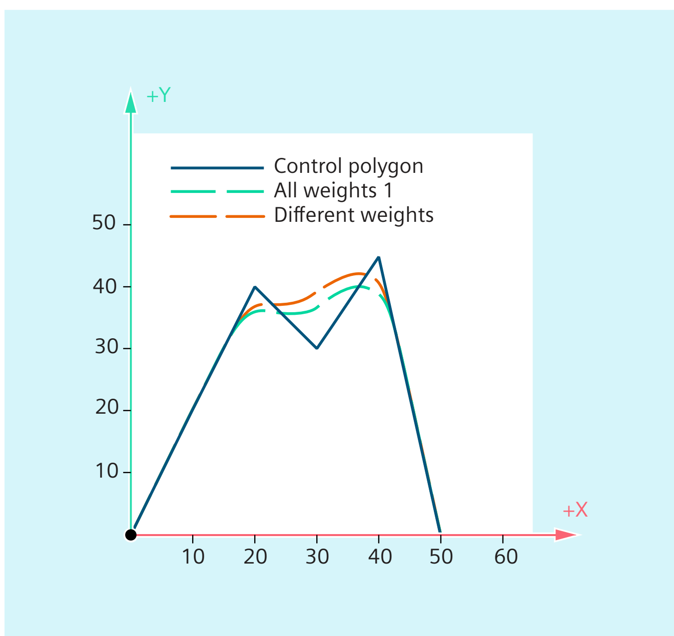
Examples

Example 1: B spline

```
Program code 1 (all weights 1)
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20
N40 X20 Y40
N50 X30 Y30
N60 X40 Y45
N70 X50 Y0
```

```
Program code 2 (different weights)
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20 PW=2
N40 X20 Y40
N50 X30 Y30 PW=0.5
N60 X40 Y45
N70 X50 Y0
```

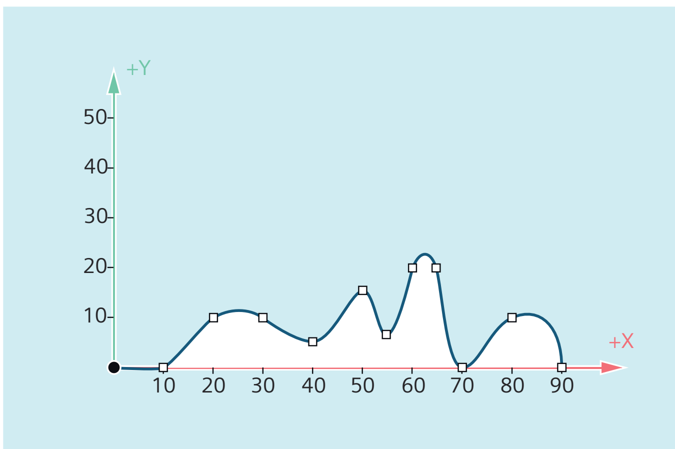
Program code 3 (check polygon)	Comment
N10 G1 X0 Y0 F300 G64	
N20	; n.a.
N30 X10 Y20	
N40 X20 Y40	
N50 X30 Y30	
N60 X40 Y45	
N70 X50 Y0	



Example 2: C spline, zero curvature at the start and at the end

```
Program code
N10 G1 X0 Y0 F300
N15 X10
N20 BNAT ENAT
```

```
Program code
N30 CSPLINE X20 Y10
N40 X30
N50 X40 Y5
N60 X50 Y15
N70 X55 Y7
N80 X60 Y20
N90 X65 Y20
N100 X70 Y0
N110 X80 Y10
N120 X90 Y0
N130 M30
```



Example 3: Spline interpolation (A spline) and coordinate transformation (ROT)

Main program:

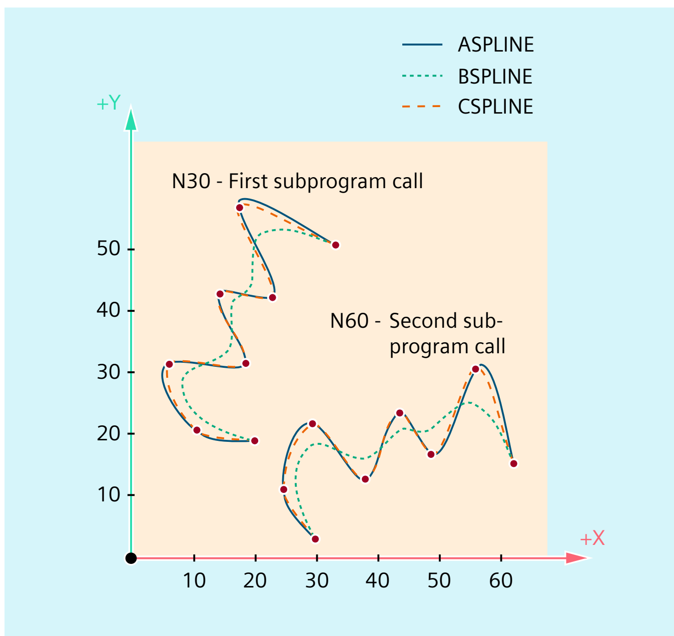
Program code	Comment
N10 G00 X20 Y18 F300 G64	; Approach starting point.
N20 ASPLINE	; Activate interpolation type A spline.
N30 CONTOUR	; First subprogram call.
N40 ROT Z-45	; Coordinate transformation: Rotation of the WCS through -45° around the Z axis.
N50 G00 X20 Y18	; Approach contour starting point.
N60 CONTOUR	; Second subprogram call.
N70 M30	; End of program

Subprogram "contour" (includes the coordinates of the points along the curve):

Program code
N10 X20 Y18

Program code
N20 X10 Y21
N30 X6 Y31
N40 X18 Y31
N50 X13 Y43
N60 X22 Y42
N70 X16 Y58
N80 X33 Y51
N90 M1

In addition to the spline curve, resulting from the example program (ASPLINE), the following diagram also contains the spline curves that would have been obtained when activating either B or C spline interpolation (BSPLINE, CSPLINE):



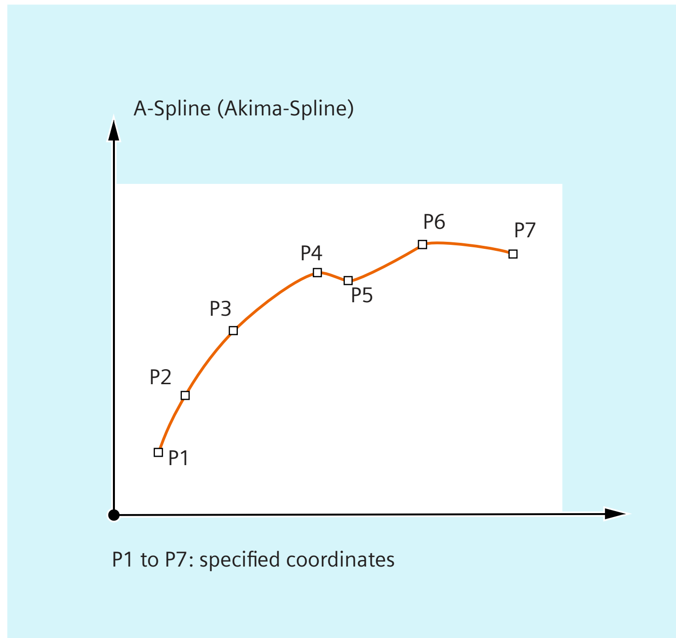
Further information

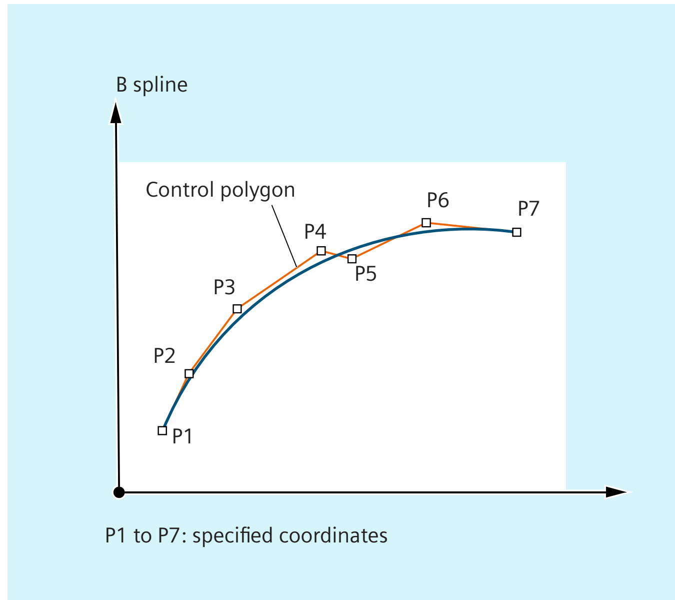
Advantages of spline interpolation

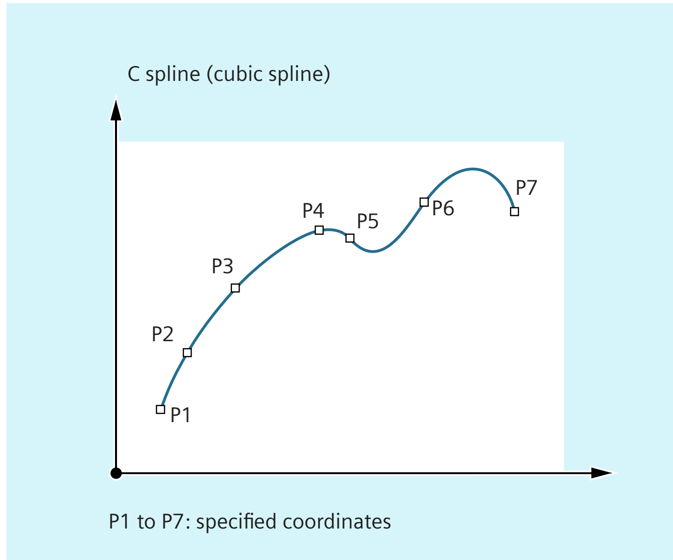
With spline interpolation, the following advantages can be obtained contrary to using straight line blocks G01:

- The number of part program blocks required to describe the contour are reduced
- Soft, curve characteristics that reduce the stress on the mechanical system at transitions between part program blocks.

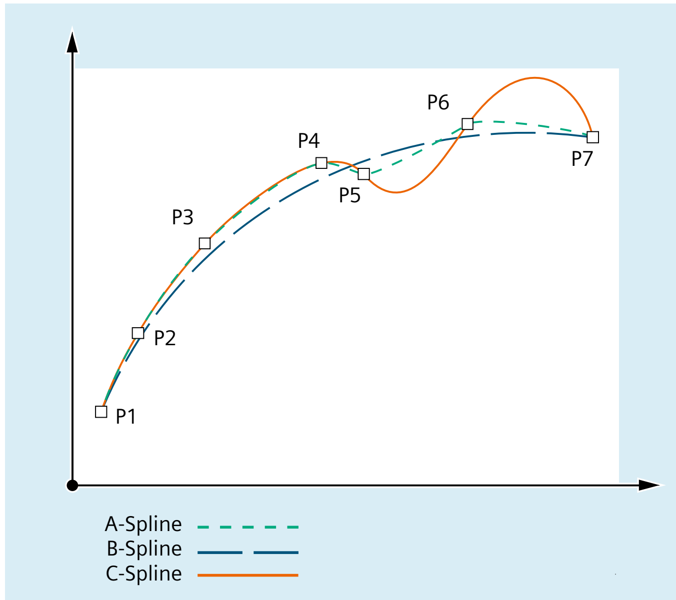
Properties and use of the various spline types

Spline type	Properties and use
A spline	 **Properties:** • Passes exactly through the specified intermediate points along the curve. • The curve characteristic is tangential, but does not have continuous curvature. • Produces hardly any undesirable oscillations. • The area of influence of changes to intermediate points along the curve is local. This means that a change to an intermediate point along the curve only affects up to max. 6 adjacent intermediate points. **Application:** The A spline is especially suitable for interpolating curves with large changes in the gradient (e.g. staircase-type curves and characteristics).

Spline type	Properties and use
B spline	 **Properties:** • Does not run through the specified intermediate points along the curve, but only close to them. The intermediate points to not attract the curve. The curve characteristic can be additionally influenced by weighting the intermediate points using a factor. • The curve characteristic is tangential with continuous curvature. • Does not generate any undesirable oscillations. • The area of influence of changes to intermediate points along the curve is local. This means that a change to an intermediate point along the curve only affects up to max. 6 adjacent intermediate points. **Application:** Their B spline is primarily intended as interface to CAD systems.

Spline type	Properties and use
C spline	 **Properties:** • Passes exactly through the specified intermediate points along the curve. • The curve characteristic is tangential with continuous curvature. • Frequently generates undesirable oscillations, especially at positions where the gradient changes significantly. • The area of influence of changes to the intermediate points is global. This means that if an intermediate point is changed then this influences the complete curved characteristic. **Application:** The C spline can be well used when the intermediate points lie on a curve defined analytically (circle, parabola, hyperbola).

Comparison of three spline types with identical interpolation points



Minimum number of spline blocks

The G codes ASPLINE, BSPLINE and CSPLINE link block end points with splines. For this purpose, a series of blocks (end points) must be simultaneously calculated. The buffer size for calculations is ten blocks as standard. Not every piece of block information is a spline end point. However, the control needs a certain number of spline end-point blocks for every ten blocks:

Spline type	Minimum number of spline blocks
A spline:	At least **4** blocks out of every 10 must be spline blocks. These do not include comment blocks or parameter calculations.
B spline:	At least **6** blocks out of every 10 must be spline blocks. These do not include comment blocks or parameter calculations.
C spline:	The required minimum number of spline blocks is the result of the following sum: Value of MD20160 $MC_CUBIC_SPLINE_BLOCKS + 1 The number of points to calculate the spline segment is entered in MD20160. The default setting is 8%. At least **9** blocks out of every 10 must be spline blocks.

Note

An alarm is output if the tolerated value is undershot and likewise when one of the axes involved in the spline is programmed as a positioning axis.

Combine short spline blocks

Spline interpolation can result in short spline blocks, which reduce the path velocity unnecessarily. The "Combine short spline blocks" function allows you to combine these blocks such that the resulting block length is sufficient and does not reduce the path velocity.

The function is activated via the channel-specific machine data:

MD20488 $MC_SPLINE_MODE (setting for spline interpolation).

Further information: Function Manual Basic Functions

4.7.3 Spline group (SPLINEPATH)

The axes to be interpolated in the spline group are selected using the `SPLINEPATH` command. Up to eight path axes can be involved in a spline interpolation grouping.

Note

If `SPLINEPATH` is not explicitly programmed, then the first three axes of the channel are traversed as spline group.

Syntax

The spline group is defined in a separate block:

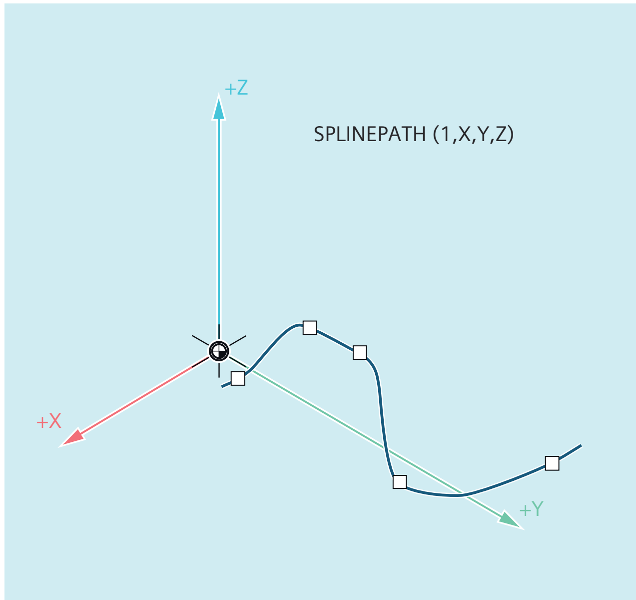
`SPLINEPATH(n,X,Y,Z,…)`

Meaning

`SPLINEPATH`:	Command to define a spline group
`n`:	=1 (fixed value)
`X,Y,Z,…` :	Identifier of the path axes to be interpolated in the spline group

Example: Spline group with three path axes

Program code	Comment
`N10 G1 X10 Y20 Z30 A40 B50 F350`	
`N11 SPLINEPATH(1,X,Y,Z)`	`; Spline group`
`N13 CSPLINE BAUTO EAUTO X20 Y30 Z40 A50 B60`	`; C spline`
`N14 X30 Y40 Z50 A60 B70`	`; Intermediate points`
`…`	
`N100 G1 X… Y…`	`; Deselect spline interpolation`



4.7.4 Activating/deactivating NC block compression (COMPON, COMPCURV, COMPCAD, COMPSURF, COMPPATH, 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 / COMCURV / COMPCAD / COMPSURF / COMPPATH
...
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
COMPPATH	Activating compressor function COMPPATH
COMPOF	Deactivating the currently active compressor function

Note

The smoothing function G642 and jerk limitation SOFT further improve the surface quality. These commands must be written at the beginning of the program.

Examples

Example 1: COMPSURF

Program code	Comment
N10 G00 X30 Y6 Z40	
N20 G1 F10000 G642	; Activation: Smoothing function G642
N30 SOFT CTOL=0.01	; Activation: Jerk limitation SOFT, contour tolerance = 0.01
N40 **COMPSURF**	; Activation: Compressor function COMPSURF
N50 FIFOCTRL	
N24050 Z32.499	; 1. traversing block
N24051 X41.365 Z32.500	; 2. traversing block
...	
N99999 X... Z...	; last traversing block
COMPOF	; compressor function deactivated
...	

Example 2: COMPPATH

Program code	Comment
N100 G64	
N110 CTOL=10	; contour tolerance = 10
N120 **COMPPATH**	; activation: Compressor function COMPPATH
N130 G1 X0. Y0. F10000	
N140 G1 X10	
N150 G0 X100	; COMPPATH active
N160 G0 Y300	; COMPPATH active
N170 G0 X10	; COMPPATH active
N180 G1 X0	
N190 **COMPOF**	; compressor function deactivated
...	

4.7.5 Settable path reference (SPATH, UPATH)

For polynomial interpolation (POLY, ASPLINE, BSPLINE, CSPLINE, COMP...), the positions of the path axes i are specified as polynomials $p_i(U)$. The curve parameter U moves from 0 to 1 within an NC block.

FGROUP selects the axes (FGROUP axes) to which the path feedrate F applies. An interpolation with constant speed on the path S of the FGROUP axes means during the polynomial interpolation normally a non-constant change of curve parameter U. Consequently, two possibilities are available for selecting the axes not contained in FGROUP on how they should follow the FGROUP axes:

- Synchronous to path S (SPATH)
- Synchronous to curve parameter U (UPATH)

Syntax

```
SPATH
UPATH
```

Meaning

SPATH:	The axes not contained in FGROUP are traversed with reference to path S
UPATH:	The axes not contained in FGROUP are traversed with reference to curve parameter U

Note

UPATH and SPATH also define the interrelationship of the F word polynomial (FPOLY, FCUB, FLIN) with path motion.

Boundary conditions

SPATH and UPATH have no meaning for:

- Linear interpolation (G1)
- Circular interpolation (G2, G3)
- Thread blocks (G33, G34, G35, G33x, G63)
- All path axes are contained in FGROUP

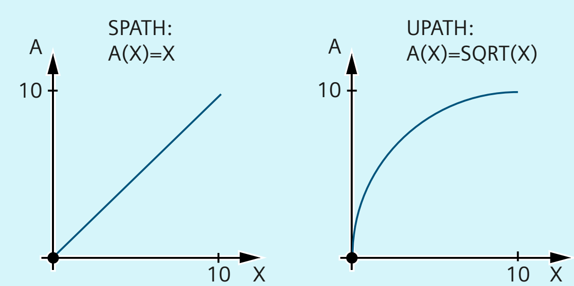
Example

The following example shows the difference between both types of motion control.

```
Program code
N10 FGROUP(X,Y,Z)
N15 G1 X0 A0 F1000 SPATH                 ; SPATH
N20 POLY PO[X]=(10,10) A10
```

Program code	
N10 FGROUP(X,Y,Z)	
N15 G1 X0 A0 F1000 UPATH	; UPATH
N20 POLY PO[X]=(10,10) A10	

In both program sections, the path S of the FGROUP axes in `N20` is dependent on the square of curve parameter U. Therefore, different position arise for synchronized axis A along path X, according to whether `SPATH` or `UPATH` is active.



Different geometric relations between axes for SPATH and UPATH

Further information

Control behavior for reset and machine/option data

The G command, defined with MD20150 $MC_GCODE_RESET_VALUES[44], is effective after a reset (45th. G group).

The initial state for the type of smoothing is defined with
MD20150 $MC_GCODE_RESET_VALUES[9] (10th G group).

The axis-specific machine data MD33100 $MA_COMPRESS_POS_TOL[<n>] has an extended significance: It contains the tolerances for the compressor function and for smoothing with G642.

4.7.6 Channel-specific measuring (MEAS, MEASF, MEAW)

In the case of channel-specific measuring, the measuring process for an NC channel is always activated from the part program running in the relevant channel.

As soon as a measuring block becomes active, the probe approaches the workpiece and the path and positioning axis movements programmed in the block start.

When the trigger event programmed in the measuring block occurs, the current positions of all involved path and positioning axes are recorded and stored in system variables.

Depending on the function variant (MEAS/MEASF or MEAW), the traversing motions are decelerated in a defined manner after the trigger event occurs (measurement with delete distance-to-go) or continued until the end (measurement without delete distance-to-go).

Requirements

MEASF

To use the MEASF function variant, the "Measuring stage 2" option, which requires a license, is required.

Syntax

```
MEAS=<TE> G... F... X... Y... Z...
MEASF=<TE> G... F... X... Y... Z...
MEAW=<TE> G... F... X... Y... Z...
```

Note

MEAS, MEASF and MEAW are active block-by-block and are programmed together with motion operations. The feedrate and interpolation type as well as the number of axes must be adapted for the respective measuring task.

Meaning

<table>
<tr><td><code>MEAS</code></td><td colspan="3">Channel-specific measurement with delete distance-to-go</td></tr>
<tr><td><code>MEASF</code></td><td colspan="3">Channel-specific high-speed measurement with delete distance-to-go
If the "Measuring stage 2" option is set, the MEASF variant is also available for channel-specific measurement with delete distance-to-go.
With MEASF, special internal control measures for preparing the measurement ensure that the measurement process is optimized and can deliver a measurement result as quickly as possible.
MEASF is therefore used for time-critical measuring tasks.</td></tr>
<tr><td><code>MEAW</code></td><td colspan="3">Channel-specific measurement without delete distance-to-go</td></tr>
<tr><td rowspan="8"><code><TE></code></td><td colspan="3">Trigger event to initiate measurement</td></tr>
<tr><td>Type:</td><td colspan="2">INT</td></tr>
<tr><td colspan="2">Value range:</td><td>-2, -1, 1, 2</td></tr>
<tr><td rowspan="4">Value:</td><td>(+)1</td><td>Rising edge of probe 1 (on measuring input 1)</td></tr>
<tr><td>-1</td><td>Falling edge of probe 1 (on measuring input 1)</td></tr>
<tr><td>(+)2</td><td>Rising edge of probe 2 (on measuring input 2)</td></tr>
<tr><td>-2</td><td>Falling edge of probe 2 (on measuring input 2)</td></tr>
<tr><td colspan="3">Note:
There is a maximum of 2 probes (dependent on configuration level).</td></tr>
<tr><td><code>G...:</code></td><td colspan="3">Type of interpolation (e.g. G0, G1, G2 or G3)</td></tr>
<tr><td><code>F...:</code></td><td colspan="3">Feedrate</td></tr>
<tr><td><code>X... Y... Z...:</code></td><td colspan="3">End points in Cartesian coordinates</td></tr>
</table>

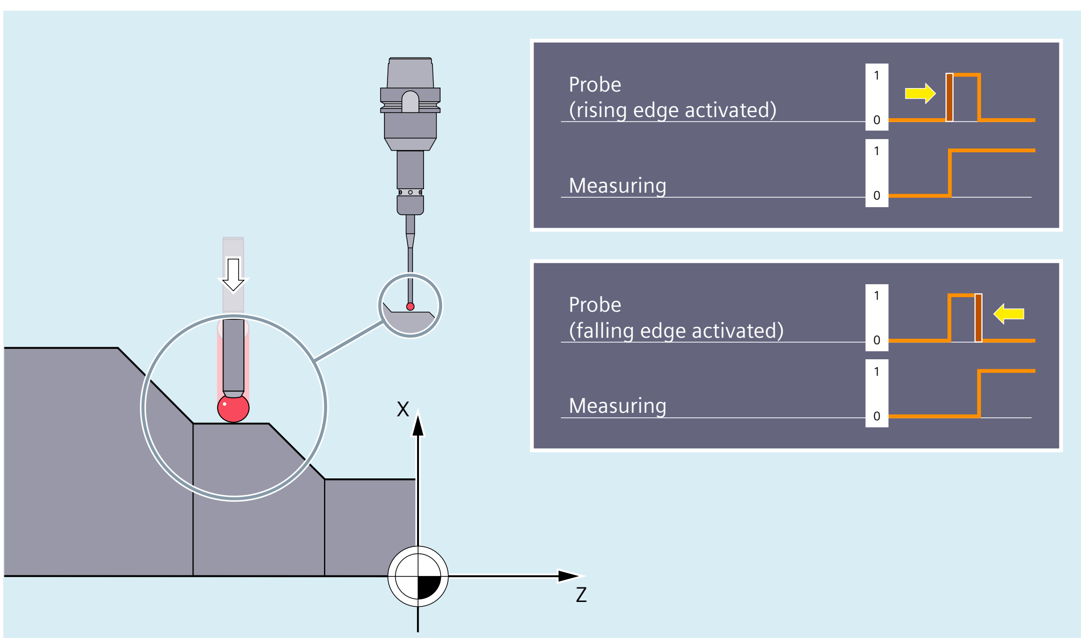
Example

Program code	Comment
...	
N10 **MEAS=1** G1 F1000 X100 Y730 Z40	; Measuring block with probe at first measuring input and linear interpolation. ; The rising edge of the probe signal triggers the measurement. ; Preprocessing stop is automatically generated.
...	

More information

Measuring process

The trigger for the measurement is the trigger event programmed in the measuring block, i.e. either the rising (0 → 1) or falling (1 → 0) edge of probe 1 or 2:



When the trigger event occurs, the positions of all traversed path and positioning axes of the block are recorded and stored in system variables.

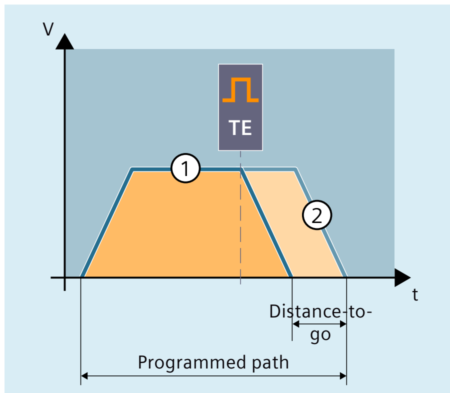
Note

If a geometry axis is programmed in a measuring block, the measured values are stored for all current geometry axes.

If an axis participating in a transformation is programmed in a measuring block, the measured values are stored for all axes participating in this transformation.

Measuring with and without delete distance-to-go

With MEAS or MEASF, the traversing motions of the block are decelerated in a defined manner after the trigger event occurs (measuring with delete distance-to-go), with MEAW they are continued until the end (measuring without delete distance-to-go):



①	Measuring **with** delete distance-to-go (MEAS/MEASF)
②	Measurement **without** delete distance-to-go (MEAW)

Reading measurement results

The measured values of the axes acquired by probes can be read through the following system variables in the part program and in synchronized actions:

System variable	Meaning
$AA_MM[<Axis>]	Probe measured value in the machine coordinate system
$AA_MW[<Axis>]	Probe measured value in the workpiece coordinate system

Query status

If an evaluation is required in the program, whether a probe has been deflected or has switched, the status can be queried through the following system variables:

System variable	Meaning	Data type	Value	
$A_PROBE[<n>]	Deflection state of the probe	INT	0	Probe not deflected.
			1	Probe deflected.
$AC_MEA[<n>]	Switching status of the probe $AC_MEA[<n>] is automatically reset at the beginning of a measurement.	INT	0	Probe has not switched
			1	Probe has switched.

<n> = number of the probe

4.7.7 Axis-specific measurement (MEASA, MEAWA, MEAC) (option)

With axis-specific measuring, activation of the measuring process can take place in the part program **or** in synchronized actions. Only one measuring job can be active per axis at any given time.

The trigger for the measurement are the trigger events programmed in the measuring block. Up to 4 trigger events can be programmed per axis. A trigger event is determined by the probe number or measurement input number (1 or 2) and by the trigger criterion (rising or falling edge of the probe signal).

The measuring mode can be used to set whether the trigger events are to be evaluated simultaneously in a position controller cycle in the order in which they occur or one after the other in the programmed order.

If two measuring systems are available for an axis, both can be used for the measurement.

Depending on the function variant, the measurement job is completed with the occurrence of the last trigger event (axis-specific measurement with or without delete distance-to-go) or the trigger events are activated again after each occurrence and evaluated repeatedly (axis-specific continuous measurement without delete distance-to-go).

The measurement results are stored in system variables or, in the case of continuous measurement, in FIFO variables.

Syntax

```
MEASA[<Axis>]=(<Mode>,<TE1>,...,<TE4>)
MEAWA[<Axis>]=(<Mode>,<TE1>,...,<TE4>)
MEAC[<Axis>]=(<Mode>,<MeasMem>,<TE1>,...,<TE4>)
```

Positioning axes

In axis-specific measurement of a positioning axis, the program line with the measuring task has the following general form:

```
MEAx[<Axis>]=(<Mode>,<TE1>,...,<TE4>) POS[<Axis>]=... FA[<Axis>]=...
```

Geometry axes/transformations

If axial measurement is to be started for a geometry axis or for an axis involved in a transformation, the same measuring job must be programmed explicitly for all remaining geometry axes or for all other axes involved in the transformation.

Example: Axis-specific measurement for the X axis. Axes x, y and z are geometry axes.

```
MEAx[X]=(<Mode>,<TE1>,...,<TE4>) MEAx[Y]=(<Mode>,<TE1>,...,<TE4>)
MEAx[Z]=(<Mode>,<TE1>,...,<TE4>) G... X... F...
```

Note

MEASA and MEAWA are non-modal; and can be programmed together in one block. If, on the contrary, MEASA/MEAWA are programmed with MEAS/MEAW in one block, there is an error message.

Meaning

<table>
<tr><td><code>MEASA</code></td><td colspan="2">Axis-specific measurement with delete distance-to-go</td></tr>
<tr><td><code>MEAWA</code></td><td colspan="2">Axes-specific measurement without delete distance-to-go</td></tr>
<tr><td><code>MEAC</code></td><td colspan="2">Axis-specific, continuous measurement without delete distance-to-go</td></tr>
<tr><td><code><Axis></code></td><td colspan="2">Name of channel axis used for measurement</td></tr>
<tr><td rowspan="12"><code><Mode></code></td><td colspan="2">Two-digit (xx) number indicating the operating mode (measuring mode and measuring system)</td></tr>
<tr><td colspan="2">Units decade: Measuring mode</td></tr>
<tr><td>x0</td><td>Cancel measuring job.</td></tr>
<tr><td>x1</td><td>Up to 4 different trigger events active simultaneously in one position controller cycle
Trigger events are evaluated in the order of their occurrence (= in chronological order).
Note:
In this mode, the programmed trigger events must be different, otherwise the measuring job will be aborted with an error message.</td></tr>
<tr><td>x2</td><td>Up to 4 sequentially active trigger events
Trigger events are evaluated in the programmed sequence:
<TE1> → <TE2> → <TE3> → <TE4>
Monitoring of the probe takes place in this mode. If, at the start of the measuring job, the deflection state of the probe is identical to the switching edge of the first programmed trigger event, the measuring job is aborted and alarm 21703 is output.</td></tr>
<tr><td>x3</td><td>As x2, but no monitoring of trigger event 1 at the start (alarm 21703 is suppressed).
Note:
MEAC does not support this mode.</td></tr>
<tr><td colspan="2">Tens decade: Measuring system
Specifies the measuring system with which measuring is to be performed.</td></tr>
<tr><td>0x (or no specification)</td><td>Active measuring system</td></tr>
<tr><td>1x</td><td>Measuring system 1</td></tr>
<tr><td>2x</td><td>Measuring system 2</td></tr>
<tr><td>3x</td><td>Both measuring systems</td></tr>
<tr><td colspan="2">Note:
If only one measuring system is available, this measuring system is automatically used (regardless of programming).</td></tr>
</table>

`<TE1>,...,<TE4>`	Trigger events to initiate measurement		
	Type:	INT	
	Value range:		-2, -1, 1, 2
	(+)1	Rising edge of probe 1	
	-1	Falling edge of probe 1	
	(+)2	Rising edge of probe 2	
	-2	Falling edge of probe 2	
	Note: If the measuring operation is performed with two measuring systems, a maximum of two trigger events can be programmed (rising or falling edge). The measured values of both measuring systems are acquired for both of the trigger events.		
`<MeasMem>`	Number of FIFO (circular buffer)		
	Type:	INT	

Examples

Example 1:

Axis-specific measurement of a positioning axis with delete distance-to-go in mode 1 (evaluation in chronological sequence)

Program code	Comment
...	
N100 **MEASA[Q]=(1,1,-1) POS[Q]=100 FA[Q]=1000**	
	; Measuring in mode 1 with active measuring system. Wait for measuring signal with positive/negative edge from probe 1 for travel path to Q=100.
N110 IF $AC_MEA[1]==FALSE GOTOF END	; Check that the measurement was successful.
N120 R10=$AA_MM1[Q]	; Save measured value acquired at the first programmed trigger event (rising edge).
N130 R11=$AA_MM2[Q]	; Save measured value acquired at the second programmed trigger event (falling edge).
N140 END:	

Example 2:

Axis-specific measurement of a geometry axis with delete distance-to-go in mode 1 (evaluation in chronological sequence)

Axis-specific measurement for the X axis. Axes x, y and z are geometry axes.

a) Measuring with one measuring system

Program code	Comment
...	
N100 **MEASA[X]=(1,1,-1) MEASA[Y]=(1,1,-1) MEASA[Z]=(1,1,-1)** G01 X100 F100	

Program code	Comment
	; Measuring in mode 1 with active measuring system. Wait for measuring signal with positive/negative edge from probe 1 for travel path to X=100.
N110 IF $AC_MEA[1]==FALSE GOTOF END	; Check that the measurement was successful.
N120 R10=$AA_MM1[X]	; Save measured value acquired at the first programmed trigger event (rising edge).
N130 R11=$AA_MM2[X]	; Save measured value acquired at the second programmed trigger event (falling edge).
N140 END:	

b) Measuring with two measuring systems

Program code	Comment
...	
N200 **MEASA[X]=(31,1,-1) MEASA[Y]=(31,1,-1) MEASA[Z]=(31,1,-1)** G01 X100 F100	
	; Measuring in mode 1 with active measuring system. Wait for measuring signal with positive/negative edge from probe 1 for travel path to X=100.
N210 IF $AC_MEA[1]==FALSE GOTOF END	; Check that the measurement was successful.
N220 R10=$AA_MM1[X]	; Save measured value of measuring system 1 at rising edge.
N230 R11=$AA_MM2[X]	; Save measured value of measuring system 2 at rising edge.
N240 R12=$AA_MM3[X]	; Save measured value of measuring system 1 at falling edge.
N250 R13=$AA_MM4[X]	; Save measured value of measuring system 2 at falling edge.
N260 END:	

Example 3:

Axis-specific measurement of a geometry axis with delete distance-to-go in mode 2 (evaluation in programmed sequence)

Axis-specific measurement for the X axis. Axes x, y and z are geometry axes.

Program code	Comment
...	
N100 **MEASA[X]=(2,1,-1,2,-2) MEASA[Y]=(2,1,-1,2,-2) MEASA[Z]=(2,1,-1,2,-2)** G01 X100 F100	

Program code	Comment
	; Measuring in mode 2 with active measuring system. ; Wait for measuring signal in the sequence rising edge probe 1, falling edge probe 1, rising edge probe 2, falling edge probe 2 on the traversing path to X=100.
N110 IF $AC_MEA[1]==FALSE GOTOF PROBE2	; Check that the measurement with probe 1 is successful.
N120 R10=$AA_MM1[X]	; Save measured value acquired at the first programmed trigger event (rising edge of probe 1).
N130 R11=$AA_MM2[X]	; Save measured value acquired at the second programmed trigger event (falling edge probe 1).
N140, PROBE2:	
N150 IF $AC_MEA[2]==FALSE GOTOF END	; Check that the measurement with probe 2 is successful.
N160 R12=$AA_MM3[X]	; Save measured value acquired at the third programmed trigger event (rising edge of probe 2).
N170 R13=$AA_MM4[X]	; Save measured value acquired at the fourth programmed trigger event (falling edge of probe 2).
N180 END:	

Example 4:

Axis-specific, continuous measurement of a geometry axis in mode 1 (evaluation in chronological sequence)

Axis-specific measurement for the X axis. Axes x, y and z are geometry axes.

a) Measurement of up to 100 measured values

Program code	Comment
...	
N110 DEF REAL MEASVALUE[100]	
N120 DEF INT loop=0	
N130 **MEAC[X]=(1,1,-1) MEAC[Y]=(1,1,-1) MEAC[Z]=(1,1,-1)** G01 X1000 F100	
	; Measuring in mode 1 with active measuring system. Save the measured values under $AC_FIFO1. Wait for measuring signal with falling edge from probe 1 on the traversing path to X=1000.
N135 STOPRE	
N140 MEAC[X]=(0) MEAC[Y]=(0) MEAC[Z]=(0)	; Terminate measurement when axis position is reached.
N150 R1=$AC_FIFO1[4]	; Save number of accumulated measured values in parameter R1.
N160 FOR loop=0 TO R1-1	

Program code	Comment
N170 MEASURED VALUE[loop]=$AC_FIFO1[0]	; Read-out measured values from $AC_FIFO1 and save.
N180 ENDFOR	

b) Measurement with delete distance-to-go after 10 measured values

Program code	Comment
...	
N10 WHEN $AC_FIFO1[4]>=10 DO MEAC[x]=(0) DELDTG(x)	
	; Delete distance-to-go after 10 measured values.
N20 **MEAC[X]=(1,1,1,-1) MEAC[Y]=(1,1,1,-1) MEAC[Z]=(1,1,1,-1)** G01 X100 F500	
	; Measuring in mode 1 with active measuring system. Save the measured values under $AC_FIFO1. Wait for measuring signal with positive/negative edge from probe 1 for travel path to X=100.
N30 MEAC[X]=(0) MEAC[Y]=(0) MEAC[Z]=(0)	
N40 R1 = $AC_FIFO1[4]	; Number of measured values.
...	

c) Measurement of a positive/negative tooth flank with 2 probes

Program code	Comment
...	
N110 DEF REAL MEASVALUE[16]	
N120 DEF INT loop=0	
N130 **MEAC[X]=(1,1,-1,2) MEAC[Y]=(1,1,-1,2) MEAC[Z]=(1,1,-1,2)** G01 X100 F100	
	; Measuring in mode 1 with active measuring system. Save the measured values under $AC_FIFO1. Waiting for measuring signal with falling edge probe 1/rising edge probe 2 on the traversing path to X=100.
N140 STOPRE	; Preprocessing stop.
N150 MEAC[X]=(0) MEAC[Y]=(0) MEAC[Z]=(0)	
	; Terminate measurement when axis position is reached.
N160 R1=$AC_FIFO1[4]	; Save number of accumulated measured values in parameter R1.
N170 FOR loop=0 TO R1-1	
N180 MEASURED VALUE[loop]=$AC_FIFO1[0]	; Read-out measured values from $AC_FIFO1 and save.
N190 ENDFOR	

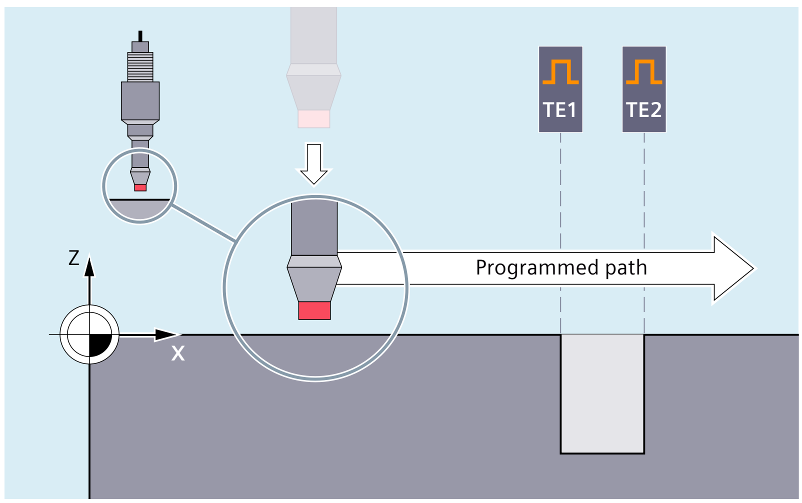
4.7.8 Axis-specific measurement (MEASA, MEAWA, MEAC) (option) Further information

More information

MEASA or MEAWA

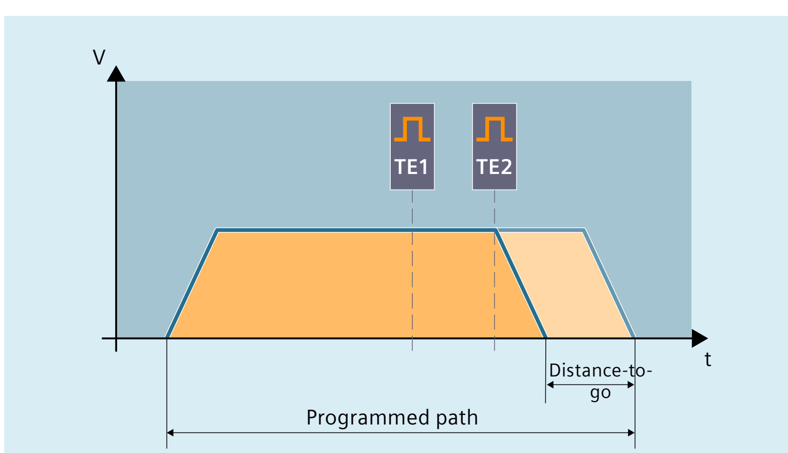
With MEASA or MEAWA for the programmed axis, up to four measured values are acquired for each measurement and the measurement results are then saved in system variables.

The following diagram shows the principle of operation of MEASA/MEAWA:



MEASA or MEAWA is used in this example to measure one drilled hole for the X axis along a programmed traversing path. The two trigger events TE1 and TE2 required for this are evaluated in the mode "one after the other in the programmed order". A contact-free switching probe is used (e.g. inductive probe).

When using function variant "Measurement with delete distance-to-go (MEASA)", axis motion is stopped after **all** programmed trigger events:



MEAWA is used for special measuring tasks in which a programmed position always has to be approached.

Note

MEASA cannot be programmed in synchronized actions. As an alternative, MEAWA plus delete distance-to-go can be programmed as a synchronized action.

If the measuring job with MEAWA is started from synchronized actions, the measured values will only be available in the machine coordinate system.

Reading measurement results (MEASA/MEAWA)

The probe measured values for MEASA/MEAWA can be read via the following system variables in the part program and in synchronized actions:

System variable	Meaning
$AA_MM1[<Axis>] ... $AA_MM4[<Axis>]	Probe measured value for trigger event 1 in the machine coordinate system ... Probe measured value for trigger event 4 in the machine coordinate system
$AA_MW1[<Axis>] ... $AA_MW4[<Axis>]	Probe measured value for trigger event 1 in the workpiece coordinate system ... Probe measured value for trigger event 4 in the workpiece coordinate system

<Axis> = measuring axis

If a measuring job is executed by two measuring systems, each of the two possible trigger events is acquired from both measuring systems.

The assignment of system variables is then as follows:

System variable	Meaning
$AA_MM1[<Axis>] or $AA_MW1[<Axis>]	Measured value from measuring system 1 on trigger event 1
$AA_MM2[<Axis>] or $AA_MW2[<Axis>]	Measured value from measuring system 2 on trigger event 1
$AA_MM3[<Axis>] or $AA_MW3[<Axis>]	Measured value from measuring system 1 on trigger event 2
$AA_MM4[<Axis>] or $AA_MW4[<Axis>]	Measured value from measuring system 2 on trigger event 2

<Axis> = measuring axis

MEAC

During continuous measurement (MEAC), the programmed trigger events are reactivated after each occurrence. This results in cyclically repeating switching edge programming and evaluation.

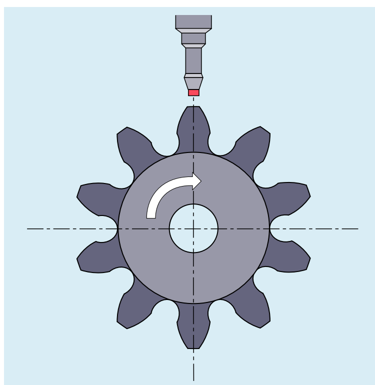
The measured values for MEAC are in the machine coordinate system and stored in the programmed FIFO[<n>] memory. If two probes are configured for the measurement, the measured values of the second probe are stored separately in the FIFO[<n>+1] memory configured specifically for this purpose.

The FIFO memory is a circular buffer in which measured values are written to $AC_FIFO variables according to the circular principle. Contents can be read only once from the circular buffer. If this measured data is to be used several times, it must be buffered in the net data.

If the number of measured values for the FIFO memory exceeds the maximum value defined in machine data, the measurement is automatically terminated.

An endless measuring process can be implemented by reading out measured values cyclically. In this case, data must be read out at least with the same frequency as new measured values are input.

A typical application example for MEAC is the measurement of toothed workpieces:



More information: Function Manual Synchronized Actions

Feedrate

The feedrate must be adjusted to suit the particular measuring task. The permissible feedrate depends on the number of programmed trigger events and the ratio of the interpolation cycle to the position controller cycle.

In the case of MEASA and MEAWA, the correctness of results can be only guaranteed for feedrates at which no more than one trigger event of the same type and no more than 4 trigger events of different types occur in each position control cycle.

In the case of continuous measurement with MEAC, the ratio between interpolation cycle and the position control cycle must not exceed 1:8.

Query status

If an evaluation is required in the program, whether a probe has been deflected or has switched, the status can be queried through the following system variables:

System variable	Meaning	Data type	Value	Meaning
$A_PROBE[<n>]	Deflection state of the probe	INT	0	Probe not deflected.
			1	Probe deflected.

System variable	Meaning	Data type	Value	Meaning
$AC_MEA[<n>]	Switching status of the probe $AC_MEA[<n>] is automatically reset at the beginning of a measurement.	INT	0	Probe has not switched
			1	Probe has switched (all trigger events programmed in the measuring block have taken place).

<n> = number of the probe

Note

If measurement is started from synchronized actions, $AC_MEA is no longer updated. In this case, the NC/PLC interface signal DB390x.DBX2.3 or the equivalent system variable $AA_MEAACT[<Axis>] must be queried:

$AA_MEAACT==1: Measurement active

$AA_MEAACT==0: Measurement not active

Probe limitation

In the NC program or in synchronized actions, when PROFIBUS telegram 395 is used, the probe limiting status can be read using system variable A_PROBE_LIMITED:

$A_PROBE_LIMITED[<n>] == 0: Probe limitation inactive/reset

$A_PROBE_LIMITED[<n>] == 1: Probe limitation active

<n> = probe number

Protection against programming errors

The following programming errors are detected and indicated as errors:

Description	Example
MEASA/MEAWA together with MEAS/MEAW in the same block	`N01 MEAS=1 MEASA[X]=(1,1) G01 F100 POS[X]=100`
MEASA/MEAWA with number of parameters <2 or >5	`N01 MEAWA[X]=(1) G01 F100 POS[X]=100`
MEASA/MEAWA with trigger event not equal to 1/ -1/ 2/ -2	`N01 MEASA[B]=(1,1,3) B100`
MEASA/MEAWA with invalid mode	`N01 MEAWA[B]=(4,1) B100`
MEASA/MEAWA with measurement mode 1 and trigger event programmed twice	`N01 MEASA[B]=(1,1,-1,2,-1) B100`
MEASA/MEAWA and missing geometry axis	`N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) G01 X50 Y50 Z50 F100` `; GEO axis X/Y/Z`
Inconsistent measuring job with geometry axes	`N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) MEASA[Z]=(1,1,2) G01 X50 Y50 Z50 F100`

4.7.9 Programmable end of motion criteria (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA)

Similar to the block change criterion for path interpolation (G601, G602, and G603) it is also possible to program the end-of-motion criterion for single-axis interpolation in a part program or in synchronized actions for command/PLC axes.

The end-of-motion criterion set will affect how quickly or slowly part program blocks and technology cycle blocks with single-axis movements are completed. The same applies for PLC via FC15/16/18.

Syntax

```
FINEA[<axis>]
COARSEA[<axis>]
IPOENDA[<axis>]
IPOBRKA(<axis>[,<instant in time>])
ADISPOSA[<axis>]=(<mode>,<window size>)
```

Meaning

`FINEA`:	End-of-motion criterion: "Exact stop fine"		
	Effective:	Modal	
`COARSEA`:	End-of-motion criterion: "Exact stop coarse"		
	Effective:	Modal	
`IPOENDA`:	End-of-motion criterion: "Interpolator stop"		
	Effective:	Modal	
`IPOBRKA`:	Block change criterion: Braking ramp		
	Effective:	Modal	
`ADISPOSA`:	Tolerance window for end-of-motion criterion		
	Effective:	Modal	
`<axis>`:	Channel axis name (X, Y, ...)		
`<instant in time>`:	Time of the block change, referred to the braking ramp as a %: • 100% = start of the braking ramp • 0% = end of the braking ramp, the same significance as `IPOENDA`		
	Type:	REAL	
`<mode>`:	Reference of the tolerance window		
	Range of values:	0	Tolerance window not active
		1	Tolerance window with respect to set position
		2	Tolerance window with respect to actual position
	Type:	INT	
`<window size>`:	Size of the tolerance window		
	Type:	REAL	

Examples

Example 1: End-of-motion criterion: "Interpolator stop"

Program code
; traverse positioning axis X to 100, velocity 200 m/min, acceleration 90%,
; end-of-motion criterion: Interpolator stop
N110 G01 POS[X]=100 FA[X]=200 ACC[X]=90 IPOENDA[X]
; Synchronized action:
; ALWAYS IF: Input 1 is set
; THEN traverse positioning axis X to 50, velocity 200 m/min, acceleration 140%,
;   end-of-motion criterion: Interpolator stop
N120 EVERY $A_IN[1] DO POS[X]=50 FA[X]=200 ACC[X]=140 IPOENDA[X]

Example 2: Block change criterion: "Braking ramp"

Program code	Comment
	; Default setting is effective
N40 POS[X]=100	; Positioning motion from X to position 100.
	Block change criterion: Exact stop fine
N20 IPOBRKA(X,100)	; Block change criterion: "Braking ramp",
	100% = start of the braking ramp
N30 POS[X]=200	; The block is changed as soon as the X axis starts to brake
N40 POS[X]=250	; X axis no longer brakes at position 200, but rather continues to traverse to position 250.
	As soon as the axis starts to brake, the block changes.
N50 POS[X]=0	; Axis X brakes and returns to position 0.
	The block change takes place at position 0 and "exact stop fine"
N60 X10 F100	; Axis X traverses as path axis to position 10

Further information

System variable for end-of-motion criterion

The effective end-of-motion criterion can be read using the system variable $AA_MOTEND.

Block-change criterion: "Braking ramp" (IPOBRKA)

If, when activating the block change criterion "brake ramp", a value is programmed for the optional block change instant in time, then this becomes effective for the next positioning motion and is written into the setting data synchronized to the main run. If no value is specified for the block change instant in time, then the actual value of the setting data is effective.

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE

IPOBRKA is deactivated for the corresponding axis when an axis end-of-motion criterion (FINEA, COARSEA, IPOENDA) is next programmed for the corresponding axis.

Additional block-change criterion: "Tolerance window" (ADISPOSA)

Using ADISPOSA, a tolerance window around the end of block (either as actual or setpoint position) can be defined as additional block change criterion. Then, two conditions must be fulfilled for the block change:

- Block-change criterion: "Braking ramp"
- Block-change criterion: "Tolerance window"

4.8 Coordinate transformations (frames)

4.8.1 Coordinate transformation via frame variables

In addition to frame instructions (Page 305), such as e.g. ROT, AROT, SCALE, etc., the workpiece coordinate system (WCS) can also be transformed using frame variables $P_...FR (data management frames) and $P_...FRAME (active frames).

The following diagram provides an overview of structuring frame variables:

- Data management frames
- Active frames
- Active total frame: Chain of all active frames
- NCU global frames
- Channel-specific frames

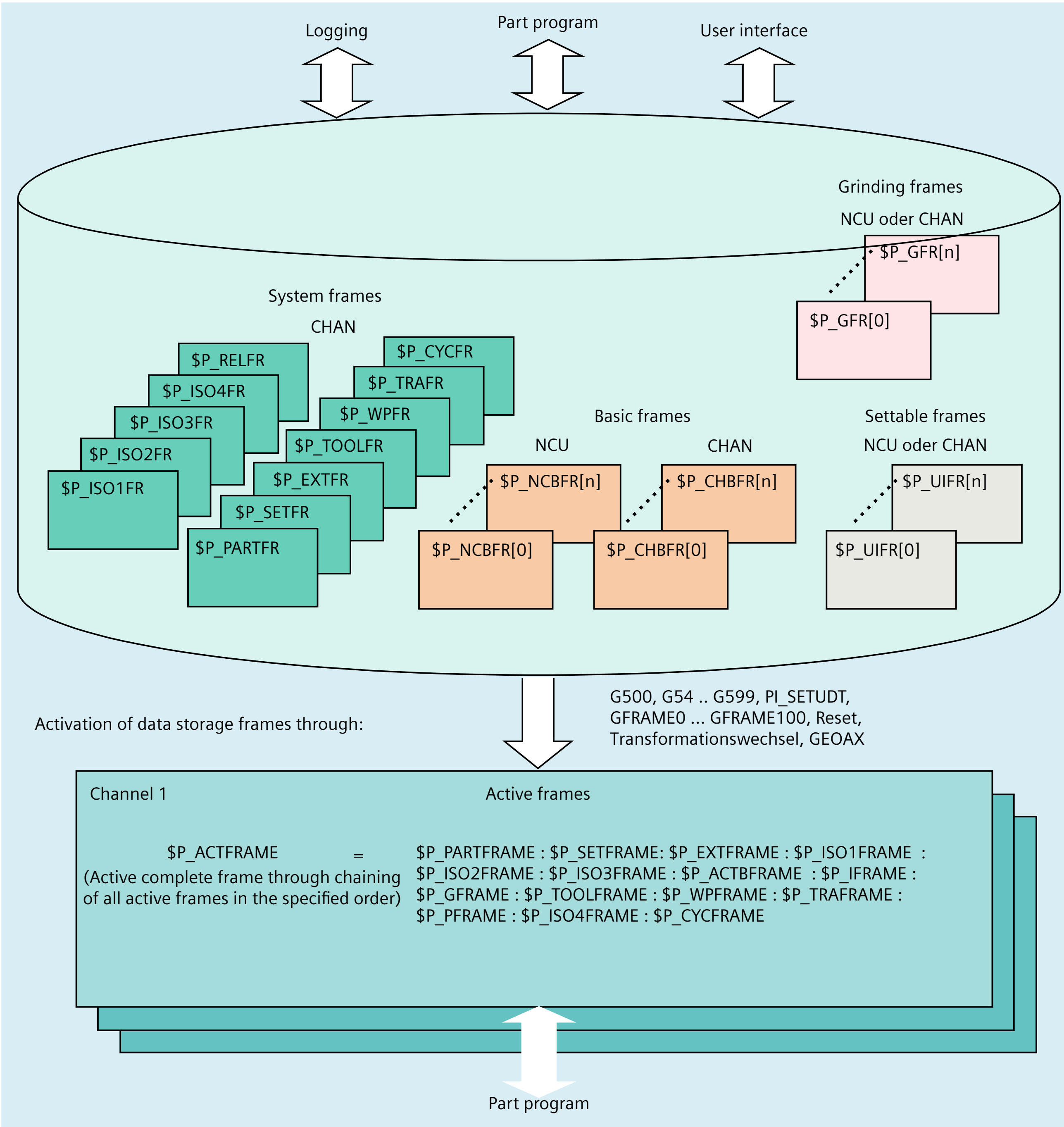


Figure 4-1 Overview of the frame variables

4.8.1.1 Predefined frame variable ($P_CHBFRAME, $P_IFRAME, $P_PFRAME, $P_ACTFRAME)

Active: channel-specific base frames $P_CHBFRAME[<n>] ($P_BFRAME)

Note

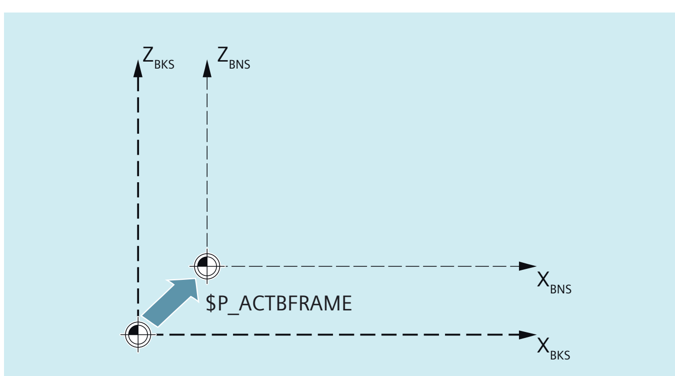
The current base frame `$P_BFRAME` and the data storage base frame `$P_UBFR` are retained for compatibility reasons.

- `$P_BFRAME ≙ $P_CHBFRAME[0]`
- `$P_UBFR ≙ $P_CHBFR[0].`

The frame variables $P_CHBFRAME[<n>] define the reference between the basic coordinate system (BCS) and the basic origin system (BOS).

If the current channel-specific base frame $P_CHBFRAME[`<n>`] should be active immediately in the NC program, the following possibilities are available.

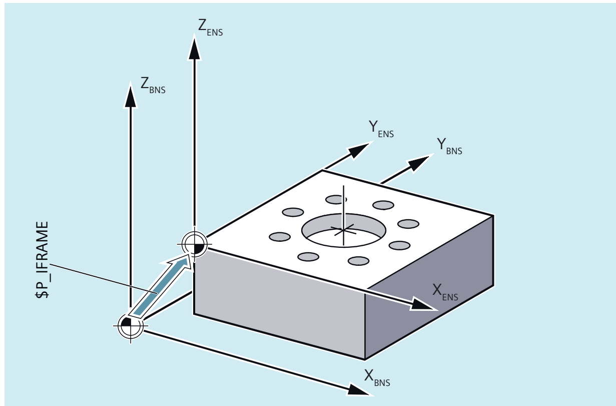
- Commands:
  - `G500` (deactivate all settable frames, the base frames remain active)
  - `G54` to `G599` (settable zero offsets)
- Assignment of a channel-specific base frames of the data storage to a current channel-specific base frame:
  `$P_CHBFRAME[<n>] = $P_CHBFR[<m>]`



Active: Channel-specific settable frame $P_IFRAME

The frame variable $P_IFRAME defines the reference between the basic origin system (BOS) and the settable zero system (SZS).

- `$P_IFRAME` corresponds to `$P_UIFR[$P_IFRNUM]`
- After `G54` is programmed, for example, `$P_IFRAME` contains the translation, rotation, scaling and mirroring defined by `G54`.

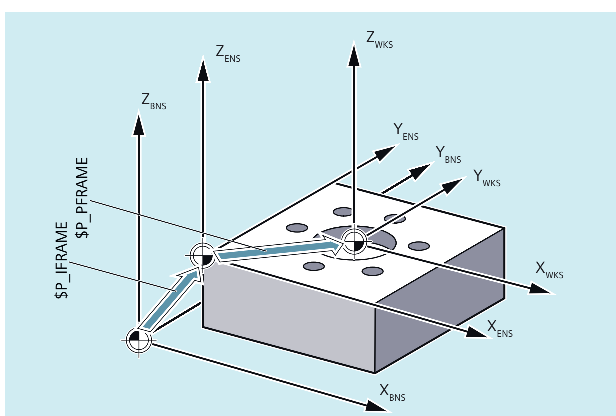


Active: Channel-specific programmable frame $P_PFRAME

The $P_PFRAME frame variable defines the reference between the settable zero system (SZS) and the workpiece coordinate system (WCS).

`$P_PFRAME` contains the resulting frame, that results

- **From the programming** of `TRANS/ATRANS`, `ROT/AROT`, `SCALE/ASCALE`, `MIRROR/AMIRROR` or
- **From the assignment** of `CTRANS`, `CROT`, `CMIRROR`, `CSCALE` to the programmed FRAME



Active: Total frame $P_ACTFRAME

The total frame active in the channel results from the chaining of all frames acting in 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
```

`$P_ACTFRAME` describes the currently valid workpiece coordinate system.

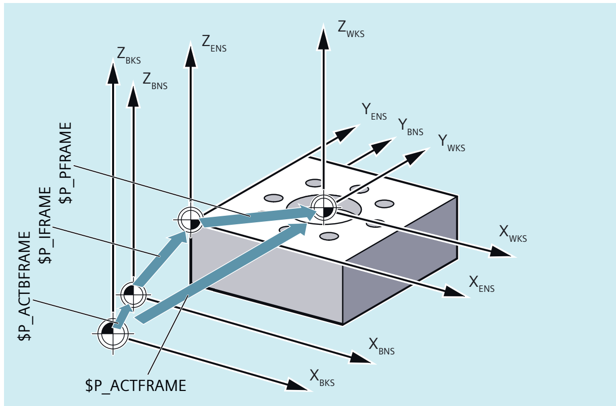
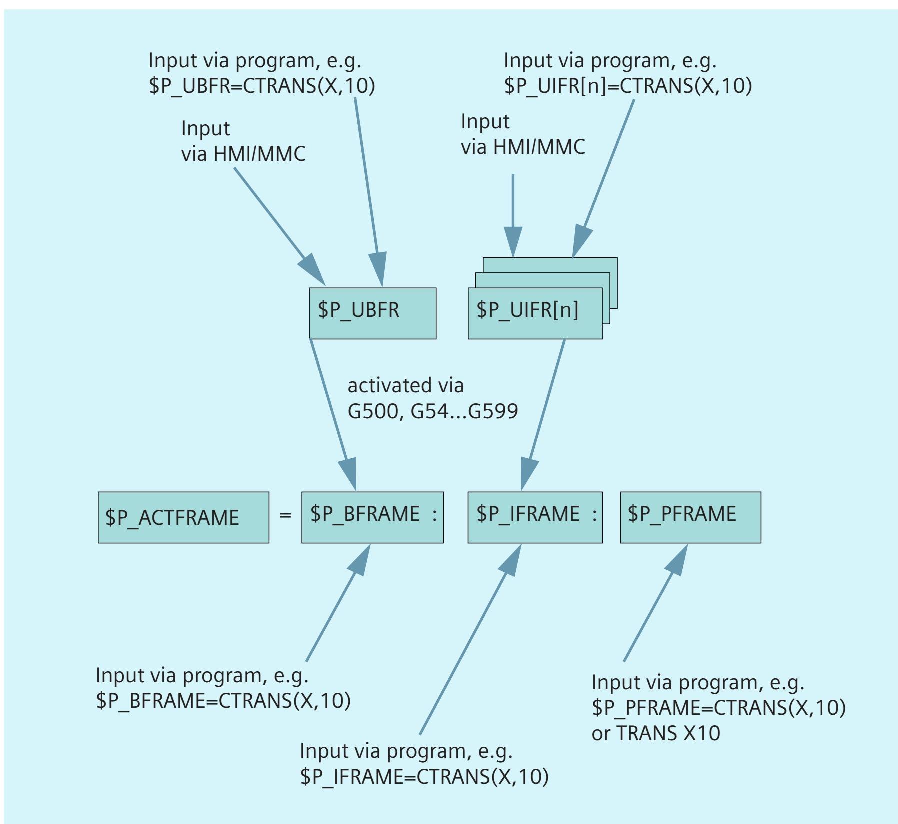


Figure 4-2 Frame variable $P_ACTFRAME

If one of the following frames `$P_BFRAME` / `$P_CHBFRAME[<n>]`, `$P_IFRAME` or `$P_PFRAME` is changed, the current total frame `$P_ACTFRAME` is recalculated.



Basic frame and settable frame are effective after Reset if MD 20110 RESET_MODE_MASK is set as follows:

Bit0=1, bit14=1 --> `$P_UBFR` (basic frame) acts

Bit0=1, bit5=1 --> `$P_UIFR [$P_UIFRNUM]` (settable frame) acts

Data storage: channel-specific base frames $P_CHBFR[<n>]

The frame variables $P_CHBFR[<n>] read/write the base frames in the data storage. The data storage frame is not immediately active in the channel when written. The written frame is activated with:

- Channel reset and MD20110 $MC_RESET_MODE_MASK, Bit0 == 1 and Bit14 == 1
- Command `G500`, `G54` ... `G57`, `G505` ... `G599` (activation/deactivation of base frames with subsequent recalculation of the current total frames)

Data storage: Channel-specific settable frames $P_UIFR[<n>]

The frame variables $P_UIFR[<n>] read/write the settable base frames in the data storage. The frame is not immediately active in the channel when written. The written frame in the channel is calculated with:

- `G500` command (deactivate all settable frames or zero offsets)
- `G54` ... `G57`, `G505` ... `G599` command (activate a settable frame or zero offset)

Active settable frame	Data storage frame	(corresponds to command)
`$P_IFRAME` =	`$P_UIFR[0]`	`G500`
	`$P_UIFR[1]`	`G54`
	`$P_UIFR[2]`	`G55`
	`$P_UIFR[3]`	`G56`
	`$P_UIFR[4]`	`G57`
	`$P_UIFR[5]`	`G505`
	`$P_UIFR[6]`	`G506`
	...	...
	`$P_UIFR[99]`	`G599`

4.8.2 Value assignments to frames

4.8.2.1 Assigning direct values (axis value, angle, scale)

You can directly assign values to frames or frame variables in the NC program.

Syntax

Syntax

```
$P_PFRAME = CTRANS(X, <offset value>, Y, <offset value>, Z, <offset value>, ...)
```

```
$P_PFRAME = ROT(X, <angle>, Y, <angle>, Z, <angle>, ...)
```

```
$P_UIFR[..] = CROT(X, <angle>, Y, <angle>, Z, <angle>,  ...
```

```
$P_PFRAME = CSCALE(X, <scale>, Y, <scale>, Z, <scale>, ...)
$P_PFRAME = CMIRROR(X, Y, Z)
```

The syntax for `$P_CHBFRAME[<n>]` is identical to `$P_PFRAME`.

Meaning

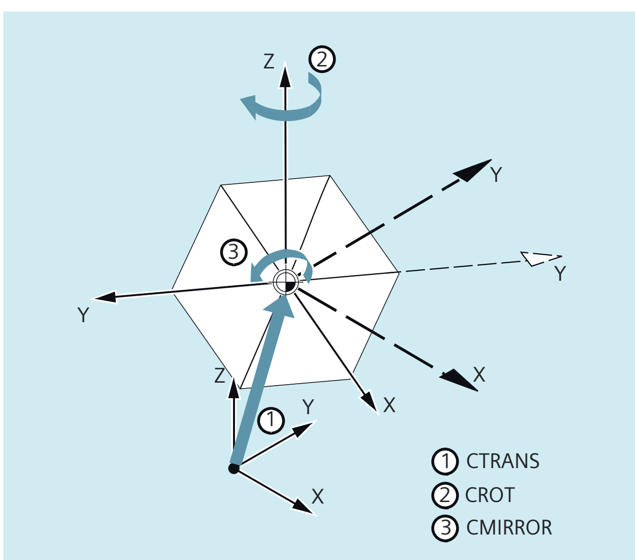
CTRANS:	Translation of specified axes
CROT:	Rotation around specified axes
CSCALE:	Scale change on specified axes
CMIRROR:	Direction reversal on specified axis
X, Y, Z:	Offset value in the direction of the specified geometry axis
<offset value>:	Offset value
<angle>:	The angle with the rotation
<scale>:	Scale value

Examples

Value assignments to frame components of the current programmable frame

Value assignment to the translation, rotation and mirror frame components of the current programmable frame:

```
$P_PFRAME = CTRANS(X,10,Y,20,Z,5) : CROT(Z,45) : CMIRROR(Y)
```



Writing the rotation components of a frame

Assignment of values to all three axes of the rotation component of the settable data storage frame `$P_UIFR` with `CROT` :

```
$P_UIFR[5] = CROT(X, 0, Y, 0, Z, 0)
```

Alternatively, the direct assignment of the individual values to the associated axis of the rotation component of the data storage frame:

```
$P_UIFR[5, Y, RT]=0
$P_UIFR[5, X, RT]=0
$P_UIFR[5, Z, RT]=0
```

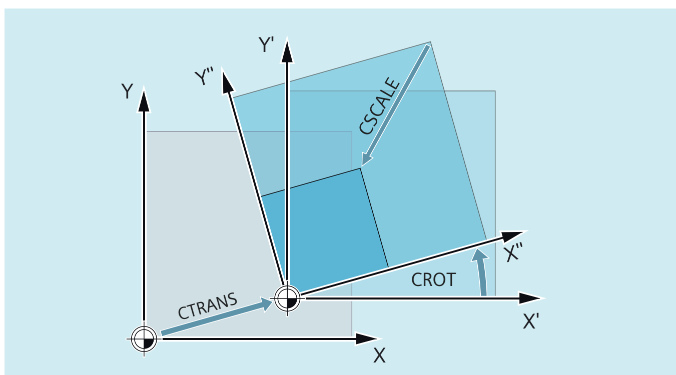
Description

The chaining operator `:` combines several operations on a frame with each other. The operations are processed successively from left to right.

Example

Chained operations on $P_PFRAME with offset, rotation and scaling:

```
$P_PFRAME = CTRANS(...) : CROT(...) : CSCALE...
```



4.8.2.2 Reading and changing frame components (TR, FI, RT, SC, MI)

This feature allows you to access **individual** data of a frame, e.g. a specific offset value or angle of rotation. You can modify these values or assign them to another variable.

Syntax

`R10=$P_UIFR[$P_UIFNUM,X,RT]`	Assign the angle of rotation RT around the X axis from the currently valid settable zero offset $P_UIFR-NUM to the variable R10.
`R12=$P_UIFR[25,Z,TR]`	Assign the offset value TR in Z from the data set of set frame no. 25 to the variable R12.
`R15=$P_PFRAME[Y,TR]`	Assign the offset value TR in Y of the current programmable frame to the variable R15.
`$P_PFRAME[X,TR] = 25`	Modify the offset value TR in X of the current programmable frame. X25 applies immediately.

Meaning

$P_UIFRNUM:	This command automatically establishes the reference to the currently valid settable zero offset.
P_UIFR[n,…,…]:	Specify the frame number n to access the settable frame no. n.
	Specify the component to be read or modified:
TR:	TR Translation
FI:	FI Translation Fine
RT:	RT Rotation
SC:	SC Scale scale modification
MI:	MI Mirroring
X, Y, Z:	The corresponding axis X, Y, Z is also specified (see examples).

Value range for RT rotation

Rotation around 1st geometry axis: -180° to +180°
Rotation around 2nd geometry axis: -90° to +90°
Rotation around 3rd geometry axis: -180° to +180°

Description

Calling frame

By specifying the system variable $P_UIFRNUM you can access the current zero offset set with $P_UIFR or G54, G55, ...
($P_UIFRNUM contains the number of the currently set frame).

All other stored settable $P_UIFR frames are called up by specifying the appropriate number $P_UIFR[n].

For predefined frame variables and user-defined frames, specify the name, e.g. $P_IFRAME.

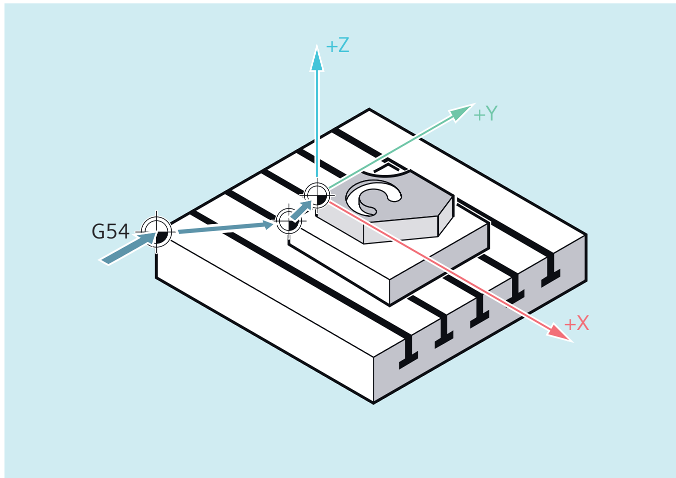
Calling data

The axis name and the frame component of the value you want to access or modify are written in square brackets, e.g. [X, RT] or [Z, MI].

4.8.2.3 Calculating with frames

A frame can be assigned to another frame or frames can be chained to each other in the NC program.

Frame chainings are suitable for the description of several workpieces, arranged on a pallet, which are to be machined in the same process.



The frame components can only contain intermediate values for the description of pallet tasks. These are chained to generate various workpiece zeroes.

Examples

Assignments

Program code	Comment
DEF FRAME SETTING_1	; Definition of a local frame variable
SETTING_1 = CTRANS(X,10)	; Assignment of the function result to the frame variable
$P_PFRAME = SETTING_1	; Assignment of the frame variable to the current frame
DEF FRAME SETTING_4	; Definition of a local frame variable
SETTING_4 = $P_PFRAME	; Buffer the current frame in the frame variable
...	
$P_PFRAME = SETTING_4	; Fetch the current frame from the frame variable

Chainings

The operator `:` chains frames with each other in the programmed sequence. The frame components, such as offsets and rotations, are executed successively additive.

Program code	Comment
$P_IFRAME = $P_UIFR[15] : $P_UIFR[16]	; Assignment of the result frame from the chaining of the
	; two settable data storage frames on the active
	; settable total frame.
	; Application example:
	; $P_UIFR[15]: Offset
	; $P_UIFR[16]: Rotation

Program code	Comment
$P_UIFR[3] = $P_UIFR[4] : $P_UIFR[5]	; Assignment of the result frame from the chaining of the
	; two settable data storage frames on a
	; different settable data storage frame

4.8.2.4 Definition of frame variables (DEF FRAME)

In addition to the predefined frame variables, user frame variables can also be defined. The user-defined frame variables are user variables of type FRAME. The name of the frame can be assigned freely in accordance with the rules for user variables.

The CTRANS, CROT, CSCALE and CMIRROR functions assign values to user-defined frame variables.

Syntax

```
DEF FRAME <name>
```

Meaning

DEF FRAME:	Define user variable of the type FRAME.
<name>:	Name of the frame variable

Example

Definition of a "PALETTE" frame variable and the assignment of offset and rotation values:

Program code	Comment
DEF FRAME PALETTE	; Define PALETTE frame variable
PALETTE = CTRANS(...) : CROT(...)	; Assignment of the result frame of the chaining for
	; offset and rotation on the PALETTE frame variable

4.8.3 Coarse and fine offsets (CTRANS, CFINE)

Fine offset

A fine offset `CFINE(...)` can be applied to the following frames:

- Settable frames: $P_UIFR or $P_IFRAME
- Basic frames: $P_NCBFR[<n>], $P_CHBFR[<n>], $P_CHBFRAMES[<n>] or $P_ACTBFRAME
- Programmable frame: $P_PFRAME

The fine offset of a frame is programmed with the `CFINE(...)` command.

Coarse offset

A coarse offset CTRANS(...) can be applied to all frames.

Total offset

The total offset results from the addition of the coarse and the fine offset.

Machine data

Enable of the fine offset

The fine offset is enabled with the machine data:

MD18600 $MN_MM_FRAME_FINE_TRANS = 1

Syntax

Fine offset

- Complete frame
  - <frame> = CFINE(<K_1>,<value>)
  - <frame> = CFINE(<K_1>,<value>, <K_2>, <value>)
  - <frame> = CFINE(<K_1>,<value>, <K_2>, <value>, <K_3>, <value>)
- Frame component
  - <frame>[<n>, <K_1>, FI] = <value>

Coarse offset

- Complete frame
  - <frame> = CTRANS(<K_1>,<value>)
  - <frame> = CTRANS(<K_1>,<value>, <K_2,<value>)
  - <frame> = CTRANS(<K_1>,<value>, <K_2,<value>, <K_3,<value>)
- Frame component
  - <frame>[<n>,<K_1>,TR] = <value>

In particular for the programmable frame $P_PFRAME:

- TRANS <K_1> <value>
- TRANS <K_1> <value> <K_2> <value>
- TRANS <K_1> <value> <K_2> <value> <K_3> <value>

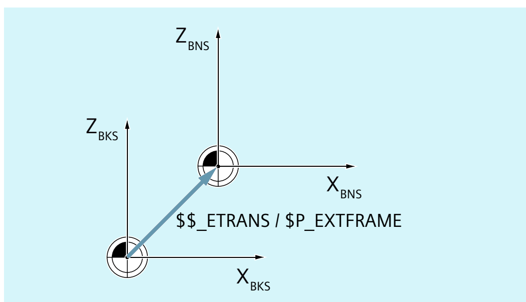
Meaning

<Frame>:	Frame, e.g. settable frame of the data storage $P_UIFR[<n>]
CFINE:	Fine offset, additive offset.
CTRANS:	Coarse offset, absolute offset.

TRANS:	Only programmable frame: Coarse offset, absolute offset.
<K_n>:	Coordinate axes X, Y, Z
<value>:	Offset value

4.8.4 External zero offset ($AA_ETRANS)

The external zero offset is a linear offset between the base coordinate system (BCS) and the basic origin system (BOS).



The external zero offset with $AA_ETRANS acts in two ways depending on the machine data parameterization:

1. After activation by the NC/PLC interface signal, the system variable $AA_ETRANS acts directly as offset value
2. After activation by the NC/PLC interface signal, the value of the system variable $AA_ETRANS is transferred to the active system frames $P:EXTFRAME and the data storage frame $P_EXTFR. The active total frame $P_ACTFRAME is then recalculated.

Machine data

In conjunction with the system variable $AA_ETRANS, a differentiation is made between two procedures selected with the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK,Bit1 = <value>

<value>	Meaning
0	Function: $AA_ETRANS[<axis>] written directly by PLC, HMI or NC program. Enable for retraction of the zero offset for $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 storage frame $P_EXTFR Enable for retraction of the zero offset for $AA_ETRANS[<axis>] by: DB31, ... DBX3.0. The following is performed in the channel: • Stop all traversal movements in the channel (other than command and PLC axes) • Preprocessing stop with subsequent reorganization (STOPRE) • Coarse offset of active frame $P_EXTFRAME[<axis>] = $AA_ETRANS[<axis>] • Coarse offset of data storage frame $P_EXTFR[<axis>] = $AA_ETRANS[<axis>] • Recalculation of the active total frame $P_ACTFRAME • Retraction of the offset in the programmed axes. • Continuation of the interrupted traversing motion or of the NC program

Programming

- Syntax
  `$AA_ETRANS[<axis>] = <value>`
- Meaning

`$AA_ETRANS`:	System variable for buffering the external zero offset
`<axis>`:	Channel axis
`<value>`:	Offset value

NC/PLC interface signal

DB31, ... DBX3.0 = 0 → 1 ⇒ $P_EXTFRAME[<axis>] = $P_EXTFR[<axis>] = $AA_ETRANS[<axis>]

4.8.5 Set actual value with loss of the referencing status (PRESETON)

The PRESETON() procedure sets a new actual value for one or more axes in the machine coordinate system (MCS). This corresponds to a work offset of the MCS of the axis. This does not cause the axis to be traversed.

A preprocessing stop with synchronization is triggered by PRESETON. The actual position is assigned to the axis only at standstill.

If the axis is not assigned to the channel for PRESETON, the next steps depend on the axis-specific configuration of the axis interchange behavior:

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:

DB390x.DBX0.4 / 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 currently valid measuring system (inch/metric) An active diameter programming (DIAMON) is taken into account	
	Type:	REAL

More information

PRESETON in NC programs

A detailed description of PRESETON in NC programs is provided in the Function Manual Basic Functions.

PRESETONS in synchronized actions

A detailed description of PRESETON in synchronized actions is provided in Function Manual Synchronized Actions.

4.8.6 Set actual value without loss of the referencing status (PRESETONS)

The PRESETONS() procedure sets a new actual value for one or more axes in the machine coordinate system (MCS). This corresponds to a work offset of the MCS of the axis. This does not cause the axis to be traversed.

A preprocessing stop with synchronization is triggered by PRESETONS. The actual position is assigned to the axis only 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

Setting a new actual value in the machine coordinate system (MCS) with PRESETONS does **not** change the referencing status of the machine axis.

Preconditions

- **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 mode of the active measuring system:
  - MD34200 $MA_ENC_REFP_MODE[<measuring system>] = 0 (no reference point approach possible)
  - MD34200 $MA_ENC_REFP_MODE[<measuring system>] = 1 (referencing for incremental, rotary or linear measuring systems: Zero pulse on the encoder track)

Programming

Syntax

```
PRESETONS(<axis_1>, <value_1> [, <axis_2>, <value_2>, ... <axis_8>,
<value_8>])
```

Meaning

`PRESETONS:`	Set actual value without 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 input is made in the active measuring system (inch/metric) An active diameter programming (DIAMON) is taken into account	
	Type:	REAL

Further information

PRESETONS in NC programs

A detailed description of PRESETONS in NC programs is provided in the Function Manual Basic Functions.

PRESETONS in synchronized actions

A detailed description of PRESETONS in synchronized actions is provided in Function Manual Synchronized Actions.

4.8.7 Frame calculation from three measuring points in space (MEAFRAME)

The MEAFRAME function is used to support measuring cycles. It calculates the frame from three ideal points and the corresponding measured points.

When a workpiece is positioned for machining, its position relative to the Cartesian machine coordinate system is generally both offset and rotated in relation to its ideal position. For exact machining or measuring either a costly physical adjustment of the part is required or the motions defined in the part program must be changed.

A frame can be defined by sampling three points in space whose ideal positions are known. A touch-trigger probe or optical sensor is used for sampling that touches special holes precisely fixed on the supporting plate or probe balls.

Syntax

```
MEAFRAME(<ideal points>,<measuring points>,<quality>)
```

Meaning

`MEAFRAME`:	Function call
`<ideal points>`:	2-dim. REAL array containing the three coordinates of the ideal points
`<measuring points>`:	2-dim. REAL array containing the three coordinates of the measured points

<quality>:	Variable with which information on the quality of the FRAME calculation is returned		
	Type:	VAR REAL	
	Value:	-1	The ideal points are almost on a straight line: The frame could not be calculated. The returned FRAME variable contains a neutral frame.
		-2	The measuring points are almost on a straight line: The frame could not be calculated. The returned FRAME variable contains a neutral frame.
		-4	The calculation of the rotation matrix failed for a different reason.
		≥ 0.0	Sum of distortions (distances between the points), that are required to transform the measured triangle into a triangle that is congruent to the ideal triangle.

Note

Quality of the measurement

In order to map the measured coordinates onto the ideal coordinates using a rotation and a translation, the triangle formed by the measured points must be congruent to the ideal triangle. This is achieved by means of a compensation algorithm that minimizes the sum of squared deviations needed to reshape the measured triangle into the ideal triangle.

Since the effective distortion can be used to judge the quality of the measurement, MEAFRAME returns it as an additional variable.

Note

The frame created by MEAFRAME can be transformed by the ADDFRAME function into another frame in the frame chain (see example "Chaining with ADDFRAME").

Examples

Example 1:

Part program 1:

Program code
...
DEF FRAME CORR_FRAME

Setting measuring points:

Program code	Comment
DEF REAL IDEAL_POINT[3,3]= SET(10.0,0.0,0.0,0.0,10.0,0.0,0.0,0.0,10.0)	
DEF REAL MEAS_POINT[3,3]= SET(10.1,0.2,-0.2,-0.2,10.2,0.1,-0.2,0.2,9.8)	; For test.

Program code	Comment
DEF REAL FIT_QUALITY=0	
DEF REAL ROT_FRAME_LIMIT=5	; Permits max. five degree rotation of the part position.
DEF REAL FIT_QUALITY_LIMIT=3	; Permits max. 3 mm offset between the ideal and the measured triangle.
DEF REAL SHOW_MCS_POS1[3]	
DEF REAL SHOW_MCS_POS2[3]	
DEF REAL SHOW_MCS_POS3[3]	

Program code	Comment
N100 G01 G90 F5000	
N110 X0 Y0 Z0	
N200 CORR_FRAME=MEAFRAME(IDEAL_POINT,MEAS_POINT,FIT_QUALITY)	
N230 IF FIT_QUALITY < 0	
SETAL(65000)	
GOTOF NO_FRAME	
ENDIF	
N240 IF FIT_QUALITY > FIT_QUALITY_LIMIT	
SETAL(65010)	
GOTOF NO_FRAME	
ENDIF	
N250 IF CORR_FRAME[X,RT] > ROT_FRAME_LIMIT	; Limiting the 1st RPY angle.
SETAL(65020)	
GOTOF NO_FRAME	
ENDIF	
N260 IF CORR_FRAME[Y,RT] > ROT_FRAME_LIMIT	; Limiting the 2nd RPY angle.
SETAL(65021)	
GOTOF NO_FRAME	
ENDIF	
N270 IF CORR_FRAME[Z,RT] > ROT_FRAME_LIMIT	; Limiting the 3rd RPY angle.
SETAL(65022)	
GOTOF NO_FRAME	
ENDIF	
N300 $P_IFRAME=CORR_FRAME	; Activate sample frame with settable frame.
	; Check frame by positioning the geometry axes to the ideal point.
N400 X=IDEAL_POINT[0,0] Y=IDEAL_POINT[0,1] Z=IDEAL_POINT[0,2]	
N410 SHOW_MCS_POS1[0]=$AA_IM[X]	
N420 SHOW_MCS_POS1[1]=$AA_IM[Y]	
N430 SHOW_MCS_POS1[2]=$AA_IM[Z]	

Program code	Comment
N500 X=IDEAL_POINT[1,0] Y=IDEAL_POINT[1,1] Z=IDEAL_POINT[1,2]	
N510 SHOW_MCS_POS2[0]=$AA_IM[X]	
N520 SHOW_MCS_POS2[1]=$AA_IM[Y]	
N530 SHOW_MCS_POS2[2]=$AA_IM[Z]	
N600 X=IDEAL_POINT[2,0] Y=IDEAL_POINT[2,1] Z=IDEAL_POINT[2,2]	
N610 SHOW_MCS_POS3[0]=$AA_IM[X]	
N620 SHOW_MCS_POS3[1]=$AA_IM[Y]	
N630 SHOW_MCS_POS3[2]=$AA_IM[Z]	
N700 G500	; Deactivate settable frame as with zero frame (no value entered, pre-assigned).
No_FRAME	; Deactivate settable frame, as pre-assigned with zero frame (no value entered).
M0	
M30	

Example 2: Chaining of frames

Chaining of MEAFRAME for offsets

The MEAFRAME function returns an offset frame. If this offset frame is chained to the settable frame $P_UIFR[1] that was active during the call of the function (e.g. G54), a settable frame is provided for further conversions for the traversing or machining.

Chaining with ADDFRAME

If you want this offset frame in the frame chain to apply at a different position or if other frames are active before the settable frame, the ADDFRAME function can be used for chaining into one of the channel basic frames or a system frame.

The following must not be active in the frames:

- Mirroring with MIRROR
- Scaling with SCALE

The input parameters for the setpoints and actual values are the workpiece coordinates. These coordinates must always be specified metrically or in inches (G71/G70) and radius-related (DIAMOF) in the basic system of the control.

Additional information on ADDFRAME, see Function Manual Basic Functions.

4.8.8 Global frames

There is only one set of global frames for all channels on each control. Global frames can be read and written from all channels. The global frames are activated in the respective channel.

Channel axes and machine axes with offsets can be scaled and mirrored by means of global frames.

Geometrical relationships and frame chains

With global frames there is no geometrical relationship between the axes. It is therefore not possible to perform rotations or program geometry axis identifiers.

Rotations cannot be used on global frames. The programming of a rotation is denied with alarm 18310 "Channel %1 Block %2 Frame: rotation not allowed".

It is possible to chain global frames and channel-specific frames. The resulting frame contains all frame components including the rotations for all axes. The assignment of a frame with rotation components to a global frame is denied with alarm "Frame: rotation not allowed".

Global frames

Global basic frames $P_NCBFR[n]

Up to eight global basic frames can be configured:

Channel-specific basic frames can also be available.

Global frames can be read and written from all channels of a control. When writing global frames, the user must ensure channel coordination. This can be implemented, for example, through wait markers (WAITMC).

Note

Machine manufacturer

The number of global basic frames is configured via the machine data.

Further information: Function Manual Basic Functions

Settable frames ($P_UIFR[n])

All settable frames G500, G54...G599 can be configured either globally or channel-specifically.

Note

Machine manufacturer

All settable frames can be reconfigured as global frames with the aid of machine data MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES.

Channel axis identifiers and machine axis identifiers can be used as axis identifiers in frame program commands. Programming the geometry identifiers is rejected with an alarm.

4.8.8.1 Channel-specific frames ($P_CHBFR, $P_UBFR)

Settable frames or basic frames can be read and written via the part program and via the OPI by the operator and by the PLC.

The fine offset can also be used for global frames. Global frames are suppressed in the same way as channel-specific frames, via G53, G153, SUPA and G500.

Machine manufacturer

The number of basic frames can be configured in the channel via the machine data MD28081 $MC_MM_NUM_BASE_FRAMES. The standard configuration is designed for at least one basic frame per channel. A maximum of eight basic frames are supported per channel. In addition to the eight basic frames, there can also be eight NCU global basic frames in the channel.

Channel-specific frames

$P_CHBFR[n]

System variable $P_CHBFR[n] can be used to read and write the basic frames. When a basic frame is written, the chained total basic frame is not activated until the execution of a G500, G54 ... G599 statement. The variable is used primarily for storing write operations to the basic frame on HMI or PLC. These frame variables are saved by the data backup.

First basic frame in the channel

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. The variable can also be read and written in the program.

$P_UBFR

$P_UBFR is identical to $P_CHBFR[0]. One basic frame always exists in the channel by default, so that the system variable is compatible with older versions. If there is no channel-specific basic frame, an alarm is issued at read/write: "Frame: statement not permissible".

4.8.8.2 Frames active in the channel

Frames active in the channel are entered from the part program via the relevant system variables of these frames. This also includes system frames. The current system frame can be read and written in the part program via these system variables.

Frames currently active in the channel

Overview

Current system frames	For:
$P_PARTFRAME	TCARR and PAROT
$P_SETFRAME	Preset actual value memory and scratching
$P_EXTFRAME	External work offset
$P_NCBFRAME[n]	Current global basic frames
$P_CHBFRAME[n]	Current channel basic frames
$P_BFRAME	Current 1. Basic frame in the channel
$P_ACTBFRAME	Complete basic frame
$P_CHBFRMASK and $P_NCBFRMASK	Complete basic frame
$P_IFRAME	Current settable frame
Current system frames	For:
$P_TOOLFRAME	TOROT and TOFRAME

$P_WPFRAME	Workpiece reference points
$P_TRAFRAME	Transformations
$P_PFRAME	Current programmable frame
Current system frame	For:
$P_CYCFRAME	Cycles
P_ACTFRAME	Current total frame
FRAME chaining	Current frame is made up of the complete basic frame

$P_NCBFRAME [n] current global basic frames

System variable $P_NCBFRAME[n] can be used to read and write the current global basic frame field elements. The resulting total basic frame is calculated by means of the write process in the channel.

The modified frame is activated only in the channel in which the frame was programmed. If the frame is to be modified for all channels of a control, $P_NCBFR[n] and $P_NCBFRAME[n] must be written simultaneously. The other channels must then activate the frame, e.g. with G54. Whenever a basic frame is written, the complete basic frame is calculated again.

$P_CHBFRAME[n] Current channel basic frames

System variable $P_CHBFRAME[n] can be used to read and write the current channel basic frame field elements. The resulting complete basic frame is calculated by means of the write process in the channel. Whenever a basic frame is written, the complete basic frame is calculated again.

$P_BFRAME current 1st Basic frame in the channel

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.

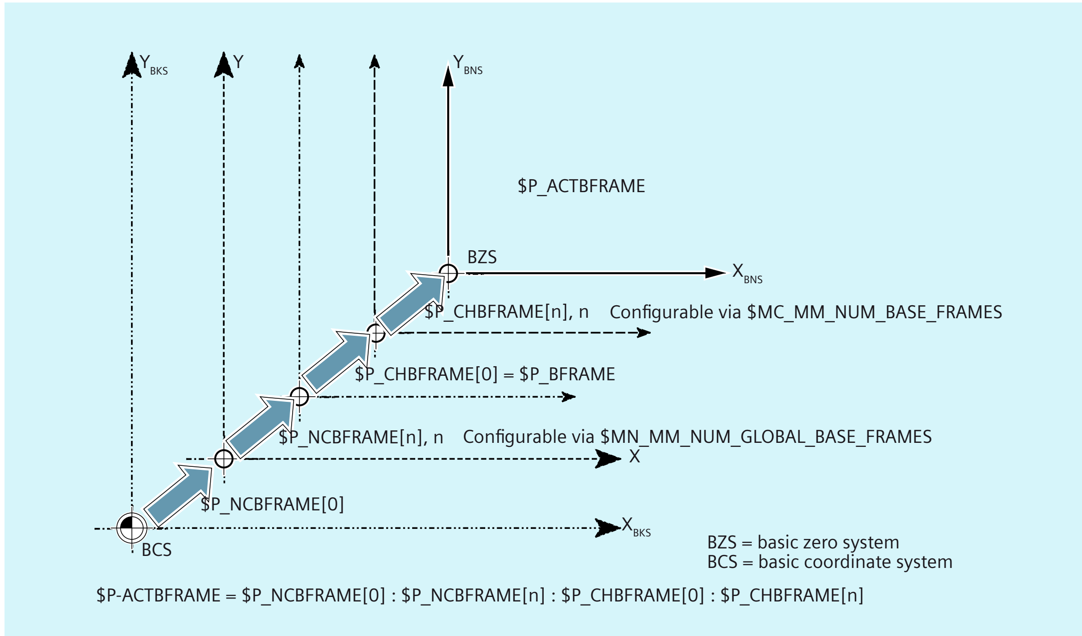
$P_BFRAME is identical to $P_CHBFRAME[0]. The system variable always has a valid default value. If there is no channel-specific basic frame, an alarm is issued at read/write: "Frame: statement not permissible".

$P_ACTBFRAME Complete basic frame

The $P_ACTFRAME variable determines the chained complete basic frame. The variable is read-only.

$P_ACTFRAME corresponds to:

$P_NCBFRAME[0] : ... : $P_NCBFRAME[n] : $P_CHBFRAME[0] : ... : $P_CHBFRAME[n].



$P_CHBFRMASK and $P_NCBFRMASK Complete basic frame

The user can select which basic frames are to be included in the calculation of the "Complete" basic frame via the system variables $P_CHBFRMASK and $P_NCBFRMASK. The variables can only be programmed in the program and read via the OPI. The value of the variable is interpreted as a bit mask and specifies which basic frame field element of $P_ACTFRAME is to be included in the calculation.

$P_CHBFRMASK can be used to specify which channel-specific basic frames and $P_NCBFRMASK can be used to specify which global basic frames are to be included in the calculation.

The complete basic frame and the complete frame are recalculated with the programming of the variables. After a reset and in the basic setting, the values of $P_CHBFRMASK and $P_NCBFRMASK are as follows:

$P_CHBFRMASK = $MC_CHBFRAME_RESET_MASK

$P_NCBFRMASK = $MC_CHBFRAME_RESET_MASK

Example:

`$P_NCBFRMASK = 'H81'` ;$P_NCBFRAME[0] : $P_NCBFRAME[7]

`$P_CHBFRMASK = 'H11'` ;$P_CHBFRAME[0] : $P_CHBFRAME[4]

$P_IFRAME Current settable frame

The predefined frame variable $P_IFRAME can be used to read and write the current settable frame, which is valid in the channel, in the part program. The written settable frame is immediately included in the calculation.

In the case of 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 a control,

$P_UIFR[n] and $P_IFRAME must be written simultaneously. The other channels must then activate the corresponding frame, e.g. with G54.

$P_PFRAME Current programmable frame

$P_PFRAME is the programmable frame that results from the programming of TRANS/ATRANS, G58/G59, ROT/AROT, SCALE/ASCALE, MIRROR/AMIRROR or from the assignment of CTRANS, CROT, CMIRROR, CSCALE to the programmable frame.

Current, programmable frame variable that establishes the reference between the settable zero system (SZS) and the workpiece coordinate system (WCS).

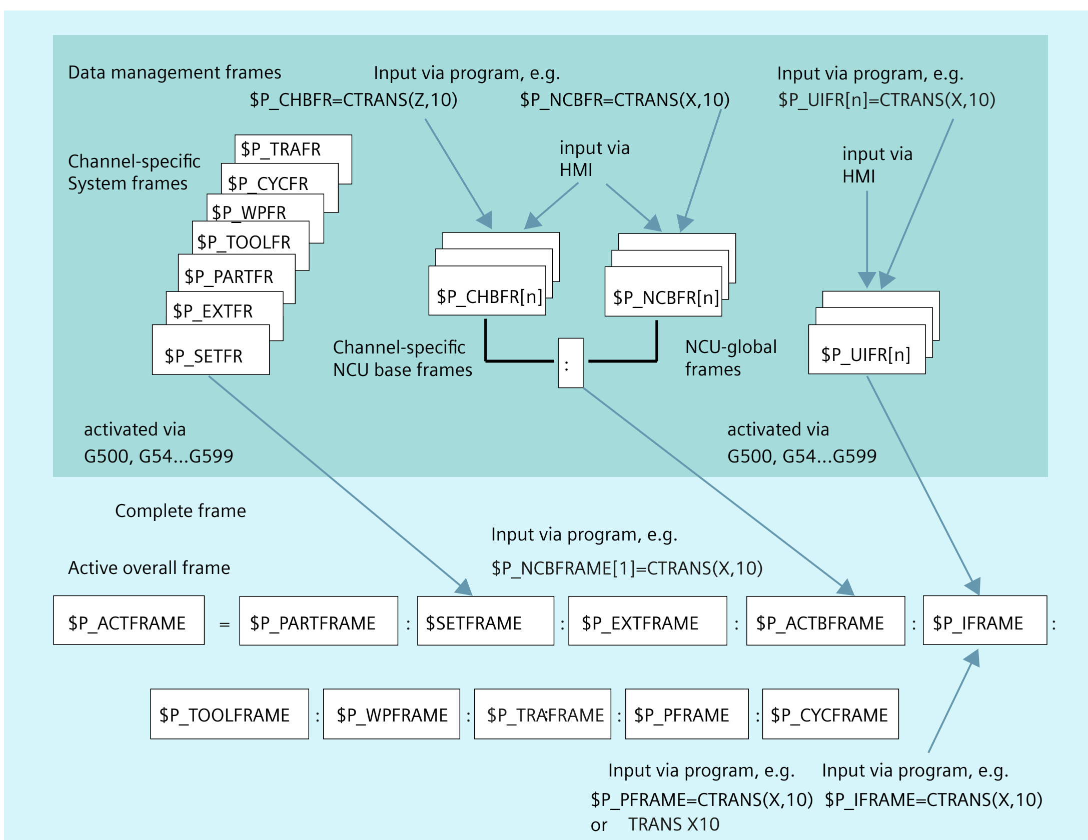
P_ACTFRAME Current complete frame

The resulting current complete frame $P_ACTFRAME is now a chain of all basic frames, the current settable frame and the programmable frame. The current frame is always updated whenever a frame component is changed.

`$P_ACTFRAME` corresponds to:

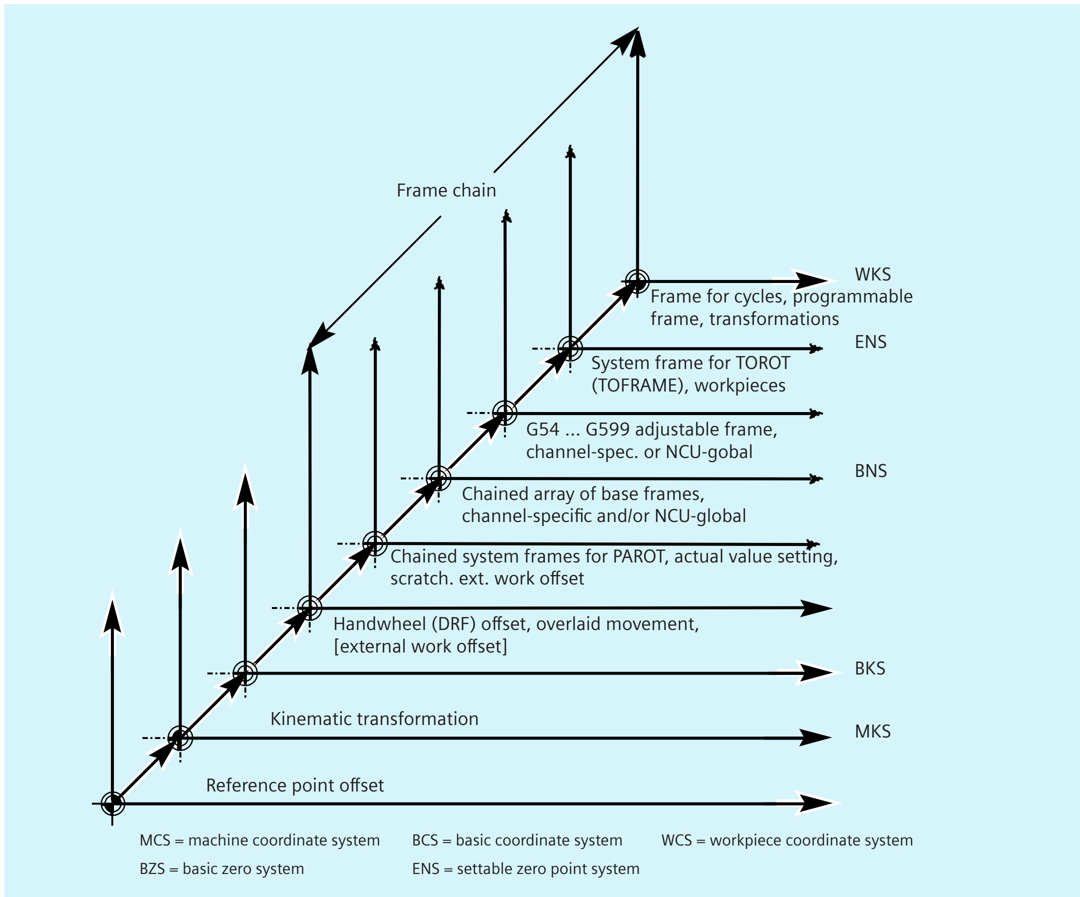
`$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_ACTBFRAME : $P_IFRAME :`

`$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME : $P_PFRAME : $P_CYCFRAME`



Frame chaining

The current frame is composed of the complete basic frame, the settable frame, the system frame and the programmable frame in accordance with the current complete frame specified above.



4.9 Transformations

4.9.1 Introduction and overview

4.9.1.1 Overview of transformation types

Transformations adapt the control to different machine kinematics.

Activating / deactivating

A transformation is activated by a function call in the NC program.

→ Chapter "Constraints when selecting a transformation (Page 640)"

→ Chapter "Deselecting a transformation (TRAFOOF) (Page 641)"

→ Chapter "Reactivating deselected transformations after a block search (SEATRAON) (Page 641)"

Kinematic transformation

In the case of kinematic transformations (TRANSMIT, TRACYL, TRAANG), positions can be programmed in the Cartesian coordinate system. The control system transforms the programmed traversing movements of the Cartesian coordinate system to the traversing movements of the real machine axes.

TRANSMIT and TRACYL

For milling on turning machines, the following machining types can be activated for the agreed transformation:

- Face machining in turning clamp with TRANSMIT
  → Chapter "Activating face end transformation (TRANSMIT) (Page 642)"
- Machining of arbitrarily running grooves on cylindrical bodies with TRACYL
  → Chapter "Activate cylinder surface transformation (TRACYL) (Page 643)"

TRAANG

If the option of setting the infeed axis for inclined infeed is required, e.g. for grinding technology, TRAANG can be used to program a configurable angle for the transformation declared.

→ Chapter "Activating an oblique angle transformation with programmable angle (TRAANG) (Page 646)".

→ Chapter "Oblique plunge-cutting on grinding machines (G5, G7) (Page 647)".

Cartesian PTP travel

Kinematic transformation also includes "Cartesian PTP travel", for which up to 8 different articulated joint positions can be programmed. Although the positions are programmed in a Cartesian coordinate system, the movement of the machine occurs in the machine coordinates.

→ Chapter "Cartesian PTP travel (Page 649)".

Concatenated transformation

Two transformations can be switched one after the other. For the second transformation chained here, the motion parts for the axes are taken from the first transformation.

The first transformation can be:

- Orientation transformation TRAORI
- Polar transformation TRANSMIT
- Cylinder transformation TRACYL
- Inclined axis transformation TRAANG

The second transformation must be Inclined axis TRAANG.

→ Chapter "Activate concatenated transformation (TRACON) (Page 660)".

4.9.1.2 Traversing movements and orientation movements during transformations

Travel movements and orientation movements

Orientation movements of the tool can be programmed using the rotary axis identifiers A..., B..., C... of the virtual axes as appropriate for the application either by entering Euler or RPY angles or directional or surface normal vectors, normalized vectors for the rotary axis of a taper or for intermediate orientation on the peripheral surface of a taper.

In the case of kinematic transformation with TRANSMIT, TRACYL and TRAANG, the controller maps the programmed Cartesian coordinate system traversing movements to the traversing movements of the real machine axes.

Kinematic transformations TRANSMIT, TRACYL and TRAANG

For milling on turning machines or an axis that can be set for inclined infeed during grinding, the following axis arrangements apply by default in accordance with the transformation declared:

TRANSMIT	Activation of polar transformation
Face machining in the turning clamp	A rotary axis An infeed axis vertical to the axis of rotation A longitudinal axis parallel to the axis of rotation

TRACYL	Activation of the cylinder surface transformation
Machining of grooves with any path on cylindrical bodies	A rotary axis An infeed axis vertical to the axis of rotation A longitudinal axis parallel to the axis of rotation

TRAANG	Activation of the inclined axis transformation
Machining with an oblique in-feed axis	A rotary axis An infeed axis with parameterizable angle A longitudinal axis parallel to the axis of rotation

4.9.2 Activation/deactivation

4.9.2.1 Constraints when selecting a transformation

Function

Transformations can be selected via a part program or MDA. Please note:

- No intermediate movement block is inserted (chamfer/radii).
- Spline block sequences must be excluded; if not, a message is displayed.
- Fine tool compensation must be deselected (FTOCOF); if not a message is displayed.
- Tool radius compensation must be deselected (G40); if not a message is displayed.
- An activated tool length offset is included in the transformation by the control.
- The control deselects the current frame active before the transformation.
- The control deselects an active operating range limit for axes affected by the transformation (corresponds to WALIMOF).
- Protection zone monitoring is deselected.
- Continuous path control and rounding are interrupted.
- All the axes specified in the machine data must be synchronized relative to a block.
- Axes that are exchanged are exchanged back; if not, a message is displayed.
- A message is output for dependent axes.

Tool change

Tools may only be changed when the tool radius compensation function is deselected.

A change in tool length offset and tool radius compensation selection/deselection must not be programmed in the same block.

Frame change

All statements, which refer exclusively to the base coordinate system, are permissible (FRAME, tool radius compensation). However, a frame change with G91 (incremental dimension) – unlike with an inactive transformation – is not handled separately. The increment to be traveled is evaluated in the workpiece coordinate system of the new frame – regardless of which frame was effective in the previous block.

Exceptions

Axes affected by the transformation cannot be used

- as a preset axis (alarm),
- for approaching a checkpoint (alarm),
- for referencing (alarm).

4.9.2.2 Deselecting a transformation (TRAFOOF)

The predefined TRAFOOF procedure deactivates all active transformations and frames.

For deselecting the transformation, the same secondary conditions (Page 640) apply as for selecting.

In addition, the following points should be noted:

- Frames that are needed again after switching off with TRAFOOF must be activated by reprogramming.
- A geometry axis configuration changed via GEOAX(...) is reset by TRAFOOF to the values set in the machine data MD20050 $MA_AXCONF_GEOAX_ASSIGN_TAB[<n>].

Syntax

```
...
TRAFOOF
```

Meaning

TRAFOOF	Deactivating all active transformations/frames

4.9.2.3 Reactivating deselected transformations after a block search (SEATRAON)

Using the predefined SEATRAON procedure, users can reactivate the last transformation that was deselected after a block search.

The following transformations are taken into consideration:

- Classic transformations: TRANSMIT, TRACYL and TRAORI
- Transformations based on the kinematic chain: TRANSMIT_K, TRACYL_K, TRAORI_STAT, TRAORI_DYN, TRAINT

The call is only permissible in the automatically activated PROG-EVENT program ($P_PROG_EVENT==5) in the area for the block search.

Application

If the basic setting to reactivate the transformation is active (MD52212 $MCS_FUNCTION_MASK_TECH, Bit 19 = 0), then the last deselected transformation is automatically reactivated at the beginning of the PROG_EVENT program.

However, if spindle programs for a block search are collected and only output at a later point in time, e.g. because the output of help functions is suppressed in the action blocks (MD11450 SEARCH_RUN_MODE, Bit 2 = 1), then the transformation must also be reactivated at a later point in time. Otherwise, a conflict can occur between the spindle axis and the transformation axis

In this or in similar cases, by appropriately programming SEATRAON, users must ensure that the transformation is reactivated at the correct point in time. In the case described, i.e. after handling the spindle.

Programming

SEATRAON must be located alone in the block:

```
...
SEATRAON
...
```

More information

Reactivation

In the PROG_EVENT program, for the block search using $P_PROG_EVENT==5, it must be ensured that the last transformation that was deselected is reactivated.

There are two options of doing this:

- Automatically using the basic setting:
  MD52212 $MCS_FUNCTION_MASK_TECH, bit 19 = 0
  If the basic setting is active, the last transformation that was deselected is automatically reactivated at the beginning of the PROG_EVENT program.
- On a user-specific basis by programming SEATRAON.

Reading transformation data

The data of the transformation deactivated in the block search, can be read using the following system variables before calling SEATRAON:

- $P_SEARCH_TRAFO

- $P_SEARCH_TRAFO_NUM

- $P_SEARCH_TRAFO_PARSET

- $P_SEARCH_TRAFO_PAR

- $P_SEARCH_TRAFO_NAME

Relevant alarms

Programming SEATRAON in an PROG_EVENT program outside an active block search with $P_PROG_EVENT==5 is not permissible, and results in the output of Alarm 14460 "SEATRAON was called outside the search run program events/ASUBs".

A transformation that was deselected and after a block search in the PROG_EVENT program was not reactivated, is displayed using Alarm 14450 "Transformation deleted during search".

4.9.3 Kinematic transformation

4.9.3.1 Activating face end transformation (TRANSMIT)

The front face transformation (TRANSMIT) is activated in the part program or synchronized action using the `TRANSMIT` statement.

Syntax

```
TRANSMIT
TRANSMIT(<n>)
```

Meaning

`TRANSMIT:`	Activate TRANSMIT with the first TRANSMIT data set
`TRANSMIT(n):`	Activate TRANSMIT with the nth TRANSMIT data set

Note

A TRANSMIT transformation active in the channel is activated with:

- Deactivate transformation: `TRAFOOF`
- Activation of another transformation: E.g. `TRACYL`, `TRAANG`, `TRAORI`

4.9.3.2 Activate cylinder surface transformation (TRACYL)

The cylinder surface transformation (TRACYL) is activated in the part program or synchronized action using the `TRACYL` statement.

Syntax

```
TRACYL(<d>)
TRACYL(<d>,<n>)
TRACYL(<d>,<n>,<k>)
```

Meaning

`TRACYL(<d>):`	Activate TRACYL with the first TRACYL data set and working diameter <d>		
`TRACYL (<d>,<n>):`	Activate TRACYL with the <n>th TRACYL data set and working diameter <d>		
`<d>:`	Reference or working diameter The value must be greater than 1.		
`<n>:`	TRACYL data set number (optional)		
	Range of values:	1, 2	
`<k>:`	The parameter <k> is only relevant for transformation type 514		
	k = 0:	without groove side correction	
	k = 1:	with groove side correction	
	If the parameter is not specified, then the parameterized basic position applies: $MC_TRACYL_DEFAULT_MODE_<n> With <n> = TRACYL data set number		

Note

A TRACYL transformation active in the channel is switched-off with:

- Deactivate transformation: `TRAFOOF`
- Activation of another transformation: E.g. `TRAANG`, `TRANSMIT`, `TRAORI`

Example

Program code	Comment
...	
N40 TRACYL(40.)	; Activate TRACYL with the first TRACYL data set and working diameter 40 mm.
...	

Further information

Program structure

A part program for milling a groove with TRACYL transformation 513 (TRACYL with groove side offset) generally comprises the following steps:

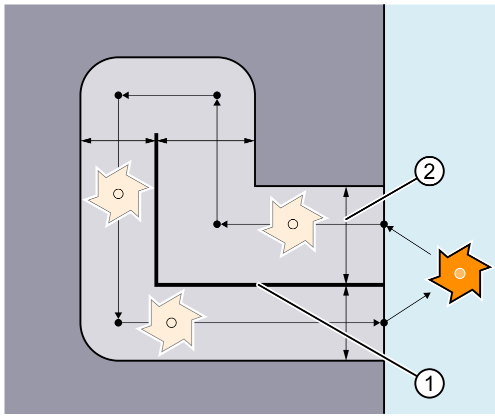
1. Select tool.
2. Select TRACYL.
3. Select suitable coordinate offset (frame).
4. Positioning.
5. Program OFFN.
6. Select TRC.
7. Approach block (position TRC and approach groove side).
8. Groove center line contour.
9. Deselect TRC.
10. Retraction block (retract TRC and move away from groove side).
11. Positioning.
12. TRAFOOF.
13. Reselect original coordinate shift (frame).

Contour offset (OFFN)

In order to mill grooves using TRACYL transformation 513, the center line of the groove and **half of the groove width** via the OFFN address are programmed in the part program.

To avoid damage to the groove side, OFFN acts only when the tool radius compensation is active.

It is possible to change OFFN within a part program. This allows the groove center line to be offset from the center:



① OFFN
② Programmed path

Note

OFFN should be at least as large as the tool radius to avoid damage occurring to the opposite side of the groove wall.

Note

OFFN acts differently with TRACYL than it does without TRACYL. Since, even without TRACYL, OFFN is included when TRC is active, OFFN should be reset to zero after TRAFOOF.

NOTICE
Effect of OFFN depends on the transformation type For TRACYL transformation 513 (TRACYL with groove side offset), half the groove width is programmed for OFFN. For TRACYL transformation 512 (TRACYL with groove side offset), the value of OFFN acts as an allowance for the TRC.

Tool radius compensation (TRC)

For TRACYL transformation 513, the TRC is not taken into account relative to the groove side, but to the programmed center of the groove. In order that the tool travels to the left of the groove side, statement G42 must be programmed instead of G41 or the value of OFFN specified with a negative sign.

Tool diameter

With TRACYL and a tool whose diameter is less than the groove width, the same groove side geometry is not generated as with a tool whose diameter is the same as the groove width. To improve the precision, it is recommended that the tool diameter is selected to be only slightly less than the groove width.

Axis utilization

Note

The following axes cannot be used as a positioning axis or a reciprocating axis:
- The geometry axis in the peripheral direction of the cylinder peripheral surface (Y axis).
- The additional linear axis for groove side compensation (Z axis).

4.9.3.3 Activating an oblique angle transformation with programmable angle (TRAANG)

The oblique angle transformation with programmable angle is activated in the part program or synchronized action using the `TRAANG` statement.

Restriction for kinematic chains

For a parameterization with machine data, the angle α of the inclined axis with respect to the corresponding coordinate axis is defined in a rectangular coordinate system via machine data. The angle α can be changed when called via TRAANG(<α>). A changed angle results in a change to the kinematic model, so other applications may also be affected by this. A specification of the angle is therefore only permissible if the angle is identical with the angle resulting from the kinematic chain.

Note

Angle definition

The angle is only defined if the inclined axis results from the rotation around a coordinate axis. There is therefore only one geometry axis, which is not parallel to one of the coordinate axes and lies in the main plane.

Syntax

```
TRAANG
TRAANG()
TRAANG(, <n>)
TRAANG(<α>)
TRAANG(<α>,<n>)
```

Meaning

`TRAANG`: `TRAANG()`:	Activate TRAANG with the first TRAANG data set and last valid angle <α>
`TRAANG(, <n>)`:	Activate TRAANG with the <n>th TRAANG data set and last valid angle <α>
`TRAANG(<α>)`:	Activate TRAANG with the first TRAANG data set and angle <α>
`TRAANG(<α>,<n>)`:	Activate TRAANG with the <n>th TRAANG data set and angle <α>

<α>:	Angle of the inclined axis (optional)	
	Value range:	-90° < α < + 90°
	The initial state parameterized in the machine data is effective if an angle is not specified: MD2xxxx $MC_TRAANG_ANGLE_<n>	
<n>:	TRAANG data set number (optional)	
	Value range:	1, 2

Note

Oblique angle transformation `TRAANG` active in the channel is deactivated using:

- Deactivate transformation: `TRAFOOF`
- Activation of another transformation: E.g. `TRACYL`, `TRANSMIT`, `TRAORI`

Example

Program code	Comment
`N20 TRAANG(45)`	`; Activate TRAANG with the first TRAANG data set and angle 45°`

4.9.3.4 Oblique plunge-cutting on grinding machines (G5, G7)

The G commands G7 and G5 are used to simplify programming of oblique plunge-cutting on grinding machines with "inclined axis (TRAANG)", so that when plunge cutting, only the inclined axis is traversed.

Only the required end position of the plunge-cutting motion has to be programmed in X and Z. For G7, starting from the actual position of the X axis, the NC calculates and approaches the programmed end position and angle α of the inclined axis.

The starting position is calculated from the point where the two straight lines intersect:

- Straight line parallel to the Z axis, at a distance from the actual position of the X axis
- Straight line parallel to the inclined axis through the programmed end position

With the subsequent G5, the inclined axis is traversed to the programmed end position.

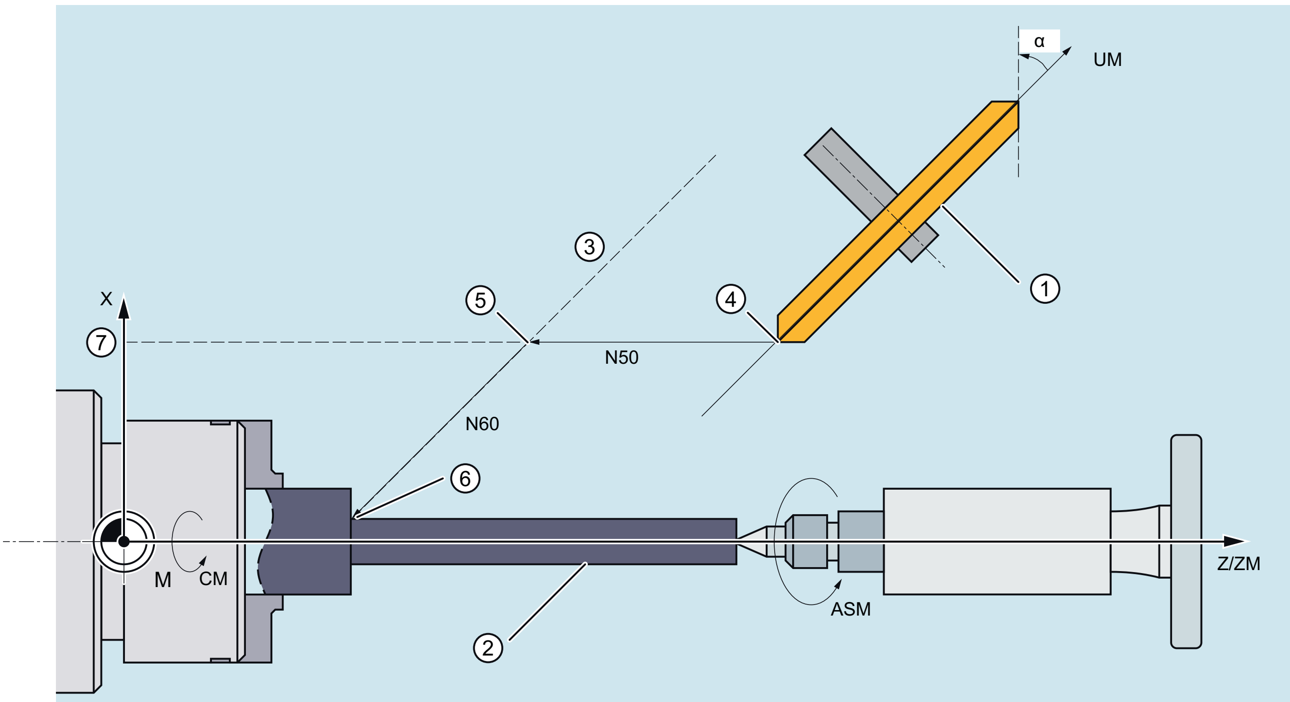
Syntax

```
G7 <Endpos_X> <Endpos_Z>
G5 <Endpos_X>
```

Meaning

`G7`:	Calculate the starting position for the oblique plunge-cutting and approach.
`G5`:	Traverse the inclined axis to the programmed end position
<Endpos_X>:	X axis end position
<Endpos_Z>:	End position of the Z axis

Example



①	Grinding wheel
②	Workpiece
③	Parallel to the inclined axis through the programmed end position
④	Starting position
⑤	Plunge-cutting: Starting position
⑥	Plunge-cutting: End position
⑦	Parallel to the Z axis, at a distance from the actual position of the X axis
X	Geometry axis
Z	Geometry axis
ZM	Machine axis
UM	Machine axis

Figure 4-3 Programming an inclined axis

Program code	Comment
`N... G18`	`; Select XZ plane.`
`N40 TRAANG (45.0)`	`; Activate TRAANG transformation, angle = 45°`
`N50 G7 X40 Z70 F4000`	`; Calculate the starting position and approach`
`N60 G5 X40 F100`	`; Traverse inclined axis to the end position.`
`N70 ...`	

4.9.4 Cartesian PTP travel

4.9.4.1 Activating/deactivating Cartesian PTP travel (PTP, PTPG0, PTPWOC, CP)

The Cartesian point-to-point or PTP travel is activated/deactivated in the NC program using G group 49 commands.

The commands are modal. The default setting is travel with Cartesian path motion (CP).

Contrary to CP, for active PTP travel, only the Cartesian target point is transformed, and the machine axes are traversed in synchronism.

In order that the Cartesian target point can be uniquely converted into machine axis values, in addition to position and angular data, information is also necessary that identifies the axis positions. This data is retrieved from the adjustable addresses STAT (Page 650) and TU (Page 655).

Requirement

Transformation TRAORI_DYN, TRAORI_STAT, TRANSMIT_K, RCTRA or ROBX is active (conventional TRAORI and TRANSMIT).

RCTRA or ROBX are not available for kinematic chains.

Syntax

```
PTP / PTPG0 / PTPWOC
...
CP
```

Meaning

`PTP`:	Activating point-to-point motion PTP The programmed Cartesian position in G0 and G1 blocks is approached with synchronous axis motion.
`PTPG0`:	Activating point-to-point motion PTPG0 Only in G0 blocks is the programmed Cartesian position approached with synchronous axis motion. In G1 blocks, a switchover is made to CP path motion.
`PTPWOC`:	Activate point-to-point movement PTPWOC (only possible if orientation transformation is active) Just the same as PTP, however, without any compensatory motion, which is caused by motion of rotary axes and orientation axes.
`CP`:	Deactivating point-to-point motion and activating path motion CP Cartesian path motion is executed with CP.

Note

PTPWOC

It does not make any sense to use PTPWOC in combination with a RCTRA or ROBX transformation! RCTRA or ROBX are not available for kinematic chains.

Examples

See:

- Example 1: PTP travel of a 6-axis robot with ROBX transformation (Page 657)
- Example 2: PTP travel for generic 5-axis transformation (Page 658)
- Example 3: PTPG0 and TRANSMIT (Page 659)

4.9.4.2 Specify the position of the joints (STAT)

Position data with Cartesian coordinates and specification of the tool orientation are not sufficient to uniquely identify the machine position, as several joint positions are possible for the same tool orientation. Depending on the kinematics involved, there can be as many as 8 different joint positions. These different joint positions are transformation-specific.

In an order to avoid any ambiguity, the joint positions are specified under the STAT address.

Note

The control takes into account programmed STAT values only for PTP motions. They are ignored with CP motions because a change of position is not normally possible while traversing with an active transformation. When traversing with active CP, the position for the target point is taken from the starting point.

Syntax

```
STAT=<Value>
```

Meaning

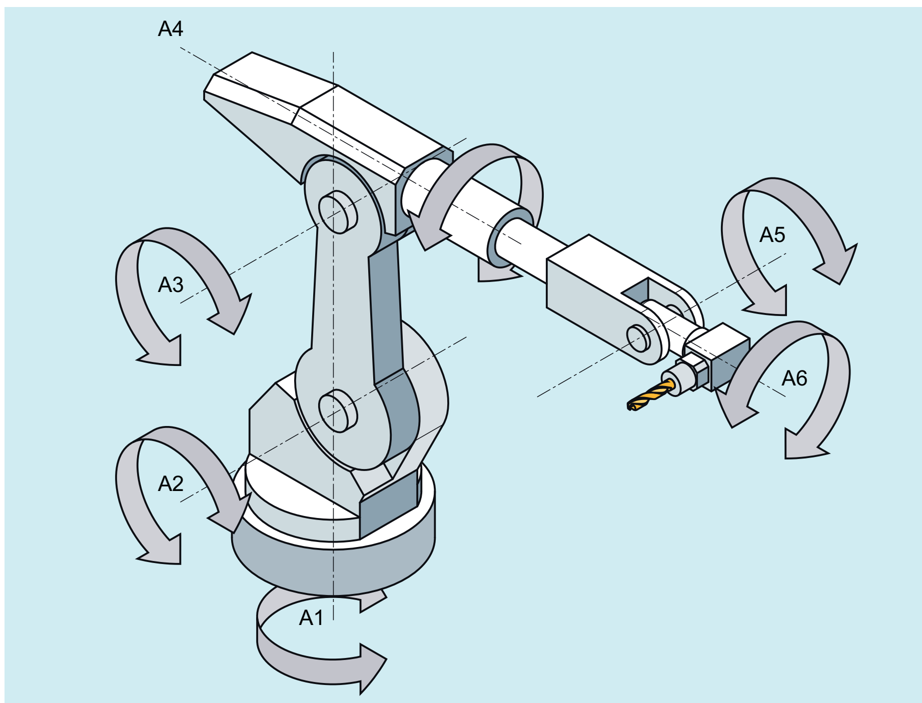
`STAT`:	Adjustable address to specify joint positions
`<value>`:	Binary or decimal value Contains one bit for each possible position. The significance of the bits is defined by the particular transformation.

The use of STAT is to be illustrated by the example of a 6-axis articulated robot with milling spindle. The kinematic transformation is to be realized using the ROBX robot transformation (precondition: Compile cycle "RMCC/ROBX Transformation Extended Robotics" is loaded and active).

Note

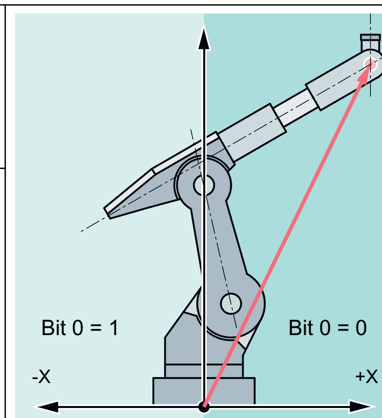
Example with ROBX only with conventional machine data parameterization

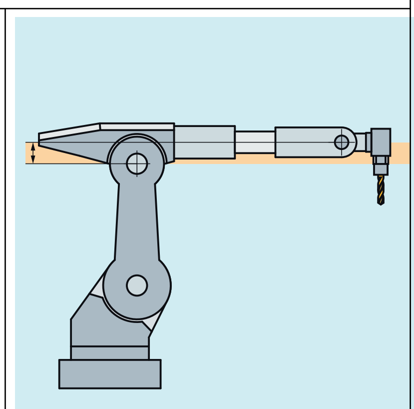
The example is only possible with machine data parameterization, not with kinematic chains.



Axes A1, A2 and A3 are the main axes of the articulated robot. The axes A4, A5 and A6, which are also designated as head or hand/wrist axes, are positioned in the working area with the main axes. The additional motion options of the hand/wrist axes enable the milling spindle to be orientated in space as required for the particular machining task. Various articulated joint positions are possible to achieve the same tool orientation.

The articulated joint positions required for machining are selected by programming bit 0 ... 2 of the adjustable STAT address:

<table>
<tr><td rowspan="3">Bit 0</td><td colspan="3">Position of the intersection points of the hand/wrist axes (A4, A5, A6)</td></tr>
<tr><td>= 0</td><td>Basic range (shoulder right)
The robot is in the basic range if the X value of the intersection point of the hand/wrist axes is positive in relation to to the A1 coordinate system.</td><td rowspan="2">

Example: The intersection point of the hand/wrist axes lies in the basic range</td></tr>
<tr><td>= 1</td><td>Overhead range (shoulder left)
The robot is in the overhead range if the X value of the intersection point of the hand/ wrist axes is negative in relation to the A1 co-ordinate system.</td></tr>
<tr><td rowspan="4">Bit 1</td><td colspan="3">Position of axis 3
The angle at which the value of bit 1 changes depends on the particular robot type. The following applies to robots whose axes 3 and 4 intersect:</td></tr>
<tr><td>= 0</td><td colspan="2">A3 <0° (elbow down)</td></tr>
<tr><td>= 1</td><td colspan="2">A3 ≥0° (elbow up)</td></tr>
<tr><td colspan="2">Note:
For robots with an offset between axes 3 and 4, the angle at which the value of bit 1 changes depends on the magnitude of this offset.</td><td>
Offset between A3 and A4</td></tr>
<tr><td rowspan="3">Bit 2</td><td colspan="3">Position of axis 5</td></tr>
<tr><td>= 0</td><td colspan="2">A5 ≥0° (no handflip)</td></tr>
<tr><td>= 1</td><td colspan="2">A5 <0° (handflip)</td></tr>
</table>

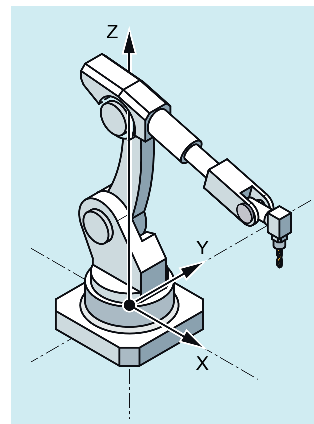
Program example:

Program code	Comment
`...`	
`N14 T="T8MILLD20" D1`	`; $TC_DP3[1,1]=132.95`

Program code	Comment
N16 ORIMKS	
N17 G1 PTP X1665.67 Y0 Z1377.405 A=0 B=0 C=0 **STAT=...** F2000	; The STAT value defines the articulated joint positions (see below)
...	

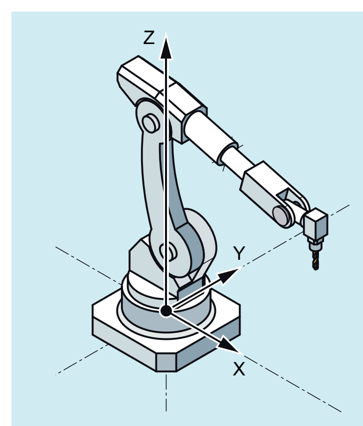
STAT=1 ('B001')

→ Shoulder left
→ Elbow down
→ No handflip



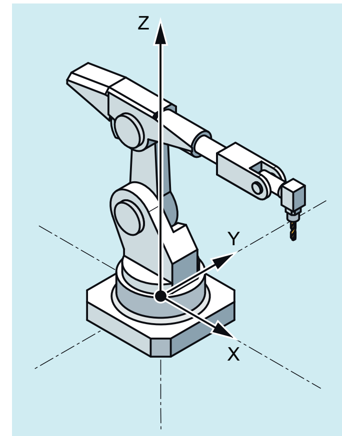
STAT=2 ('B010')

→ Shoulder right
→ Elbow up
→ No handflip



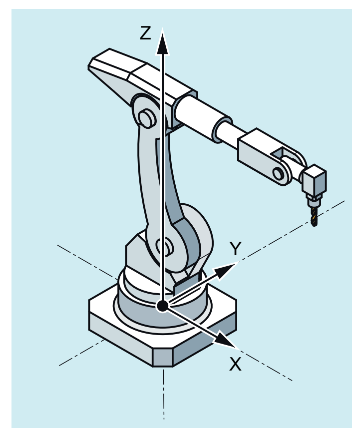
`STAT=5 ('B101')`

→ Shoulder left
→ Elbow down
→ Handflip



`STAT=6 ('B110')`

→ Shoulder right
→ Elbow up
→ Handflip



TRANSMIT_K (conventional TRANSMIT)

For TRANSMIT_K, the address STAT is used to resolve the ambiguity regarding the pole.

The following applies if the rotary axis must rotate through 180° or the contour for CP would go through the pole:

Bit 0	Only relevant for $NT_POLE_SIDE_FIX[n] = 1 or 2 for machine data parameterization the $MC_TRANSMIT_POLE_SIDE_FIX_1/2 = 1 or 2:	
	= 0	Rotary axis traverses through +180° or rotates clockwise.
	= 1	Rotary axis rotates through -180° or rotates counterclockwise.
Bit 1	Only relevant for $NT_POLE_SIDE_FIX[n] = 0 for machine data parameterization the $MC_TRANSMIT_POLE_SIDE_FIX_1/2 = 0:	
	= 0	The axis traverses through the pole. The rotary axis does not rotate.
	= 1	The axis rotates around the pole. Bit 0 of STAT is relevant.

4.9.4.3 Specify the sign of the axis angle (TU)

In order that rotary axes can also approach axis angles exceeding +180° or less than -180° without requiring a special traversing strategy (e.g. intermediate point), the sign of the axis angle must be specified under the adjustable address TU.

Note

The control only takes into account programmed TU values for PTP motion. CP motion is ignored.

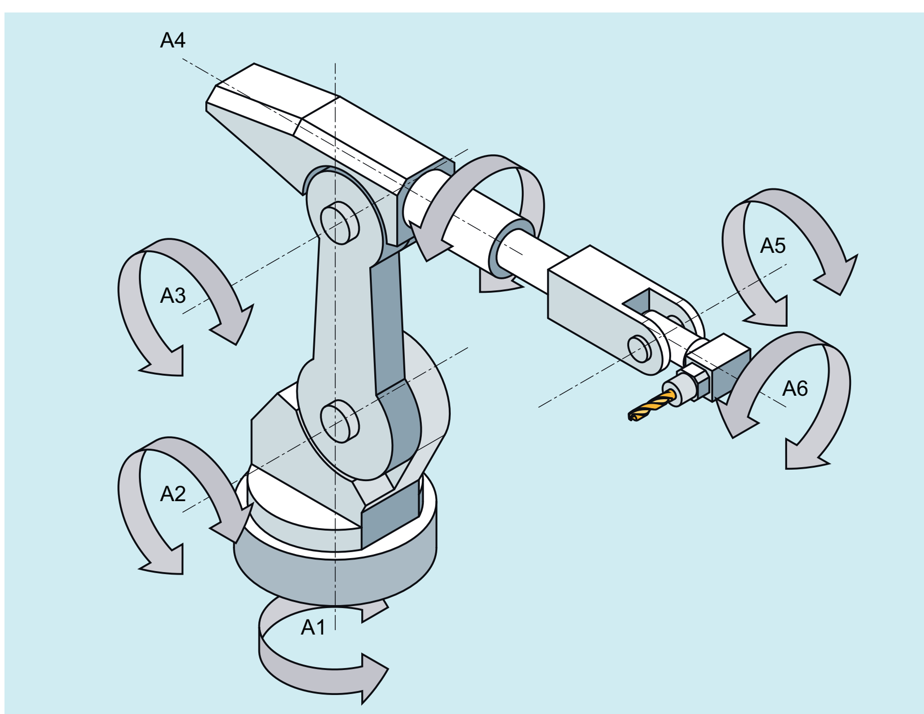
Syntax

```
TU=<Value>
```

Meaning

<table>
<tr><td><code>TU</code>:</td><td colspan="4">Adjustable address to specify axis angle signs</td></tr>
<tr><td rowspan="3"><code><value></code>:</td><td colspan="4">Binary or decimal value
For each axis that is involved in the transformation, there is a bit that indicates the sign of the axis angle (θ), and therefore the traversing direction.</td></tr>
<tr><td rowspan="2">Bit</td><td>= 0</td><td>Axis angle sign: +</td><td>Axis angular range: 0° ≤ θ < 360°</td></tr>
<tr><td>= 1</td><td>Axis angle sign: -</td><td>Axis angular range: - 360° < θ < 0°</td></tr>
</table>

Example: 6-axis articulated robot



Bit	Meaning	Value	Axis angle sign	Axis angle
Bit 0 [1)]	Sign for the axis angle of A1	= 0	+/-	≥ 0°
		= 1	-	< 0°
Bit 1 [1)]	Sign for the axis angle of A2	= 0	+/-	≥ 0°
		= 1	-	< 0°
Bit 2 [1)]	Sign for the axis angle of A3	= 0	+/-	≥ 0°
		= 1	-	< 0°
Bit 3 [1)]	Sign for the axis angle of A4	= 0	+/-	≥ 0°
		= 1	-	< 0°
Bit 4 [1)]	Sign for the axis angle of A5	= 0	+/-	≥ 0°
		= 1	-	< 0°
Bit 5 [1)]	Sign for the axis angle of A6	= 0	+/-	≥ 0°
		= 1	-	< 0°

[1)] The actual TU bit numbers obtained from the channel axis numbers of the robot axes! In the example, robot axes (A1 to A6) are the first six axes in the channel; as a consequence, TU bits 0 ... 5 are used. For another channel axis assignment of the robot axes, the TU bit numbers of the robot axes would correspondingly change (e.g.: robot axes are the 3rd to 8th channel axis, i.e. TU bits 2 ... 7 are used for the robot axes).

TU=19 (corresponds to TU='B010011) would therefore signify:

Bit	**Value**		**Axis angle**
0	= 1	⇒	$\theta_{A1} <0°$
1	= 1	⇒	$\theta_{A2} <0°$
2	= 0	⇒	$\theta_{A3} \geq 0°$
3	= 0	⇒	$\theta_{A4} \geq 0°$
4	= 1	⇒	$\theta_{A5} <0°$
5	= 0	⇒	$\theta_{A6} \geq 0°$

Note

In the case of axes with a traversing range > ±360°, the axis always moves along the shortest path because the axis position cannot be specified uniquely by the TU information.

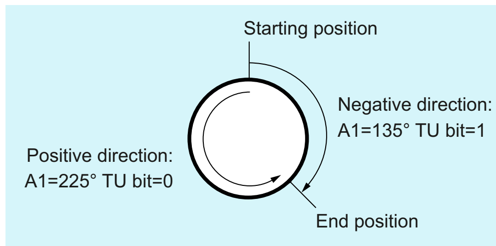
If no TU is programmed for a position, then depending on MD30455 $MA_MISC_FUNCTION_MASK the shorter or longer path is traversed.

TRANSMIT_K (conventional TRANSMIT)

For PTP travel with TRANSMIT active, the address of TU has no meaning!

Example

The rotary axis position shown in the following diagram can be approached in the negative or positive direction. The angular position is programmed under address A1. The traversing direction is only absolutely clear when TU is specified.



4.9.4.4 Example 1: PTP travel of a 6-axis robot with ROBX transformation

In the following application example, Cartesian PTP travel and the associated NC commands are shown in the form of an example.

Note

Only with conventional machine data parameterization

The example does not work with kinematic chains.

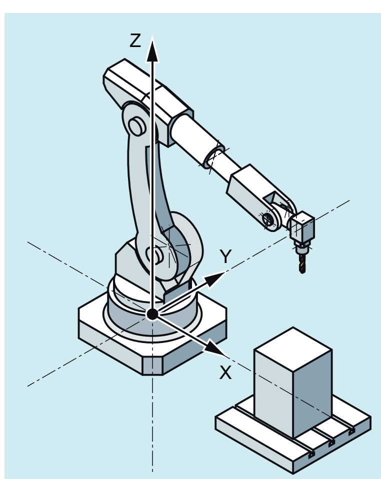


Figure 4-4 6-axis articulated robot with milling spindle

```
N1 G90
N2 T="T8MILLD20" D1 M6
N3 TRAORI
;$P_UIFR[1]=CTRANS(X,1500,Y,0,Z,400):CROT(X,0,Y,0,Z,-90)
N4 G54
N5 M3 S20000
```

```
N6 ORIWKS
N7 ORIVIRT1
N8 CYCLE832(0.01,_FINISH,1)
;HOME
N9 TRAFOOF
N10 G0 RA1=0.0000 RA2=-90.0000 RA3=90.0000 A=0.0000 B=90.0000 C=0.0000
N11 TRAORI
N12 G54
N13 G0 PTP X1369.2426 Y956.7528 Z502.5517 A=135.5761 B=-33.2223
C=161.1435 STAT='B010' TU='B001011'
N14 G0 X1355.1242 Y1014.9394 Z424.9695 A=135.8491 B=-33.1439
C=160.9941 STAT='B010' TU='B001011'
N15 G1 CP X1354.8361 Y1016.1269 Z423.3862 A=136.0635 B=-33.0819 C=160.8770
F1000
N16 G1 X1336.4283 Y1016.1269 Z426.6311 A=136.0484 B=-32.2151 C=160.9643
F2000
N17 G1 X1317.9831 Y1016.1269 Z429.6730 A=136.0175 B=-31.3394 C=161.0655
;HOME
N18 TRAFOOF
N19 G0 RA1=0.0000 RA2=-90.0000 RA3=90.0000 A=0.0000 B=90.0000 C=0.0000
N20 M30
```

4.9.4.5 Example 2: PTP travel for generic 5-axis transformation

Assumption: Right-angled CA kinematics used as basis.

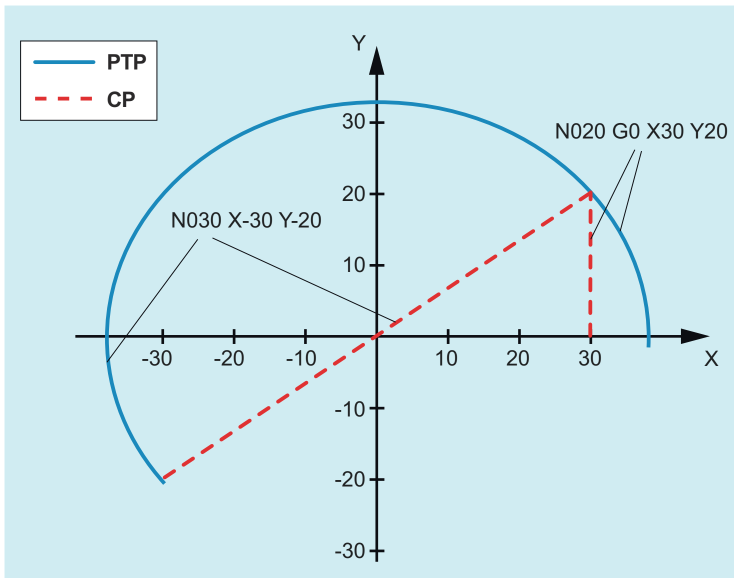
Program code	Comment
TRAORI	;Transformation CA kinematics ON
PTP	; Activate PTP traversal
N10 A3=0 B3=0 C3=1	; rotary axis positions C=0 A=0
N20 A3=1 B3=0 C3=1	; rotary axis positions C=90 A=45
N30 A3=1 B3=0 C3=0	; rotary axis positions C=90 A=90
N40 A3=1 B3=0 C3=1 **STAT=1**	; rotary axis positions C=270 A=–45

Select clear approach position of rotary axis position:

In block N40, the rotary axes – as a result of the programming of **STAT=1** – travel the longer distance from their start point (C=90, A=90) to the end point (C=270, A=–45). On the other hand, with **STAT=0**, the rotary axes would travel along the shortest path to the end point (C=90, A=45).

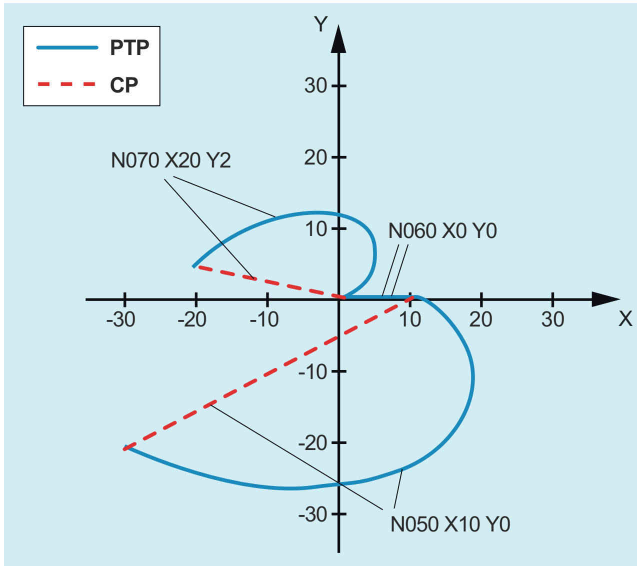
4.9.4.6 Example 3: PTPG0 and TRANSMIT

Traversing around the pole with PTPG0 and TRANSMIT



Program code	Comment
N001 G0 X30 Z0 F10000 T1 D1 G90	;Initial setting absolute dimension
N002 SPOS=0	
N003 **TRANSMIT**	;TRANSMIT transformation
N010 **PTPG0**	; for each G0 block, automatically PTP – and then CP again.
N020 G0 X30 Y20	
N030 X-30 Y-20	
N120 G1 X30 Y20	
N110 X30 Y0	
M30	

Traversing from the pole with PTPG0 and TRANSMIT



Programming	Comment
N001 G0 X90 Z0 F10000 T1 D1 G90	;Initial setting
N002 SPOS=0	
N003 **TRANSMIT**	;TRANSMIT transformation
N010 **PTPG0**	; for each G0 block, automatically PTP – and then CP again.
N020 G0 X90 Y60	
N030 X-90 Y-60	
N040 X-30 Y-20	
N050 X10 Y0	
N060 X0 Y0	
N070 X-20 Y2	
N170 G1 X0 Y0	
N160 X10 Y0	
N150 X-30 Y-20	
M30	

4.9.5 Activate concatenated transformation (TRACON)

A configured concatenated transformation is activated in the part program or synchronized action via the TRAFOON (Page 669) or TRACON operation.

To be able to activate a transformation parameterized with a kinematic chain with TRACON, this must be parameterized in the system variable $NT_TRACON_CHAIN.

Syntax

```
TRACON(<Trafo_No>,<Par_1>,...,<Par_n>,<Par_n+1>)
...
TRAFOOF
```

Meaning

<table>
<tr><td>TRACON:</td><td colspan="3">Activate concatenated transformation
If another transformation was previously activated, it is implicitly disabled by means of TRACON().</td></tr>
<tr><td rowspan="7"><Trafo_No>:</td><td colspan="3">Number of the concatenated transformation</td></tr>
<tr><td>Type:</td><td colspan="2">INT</td></tr>
<tr><td>Value range:</td><td colspan="2">0 ... 2</td></tr>
<tr><td rowspan="4">Value:</td><td>0, 1</td><td>First/only concatenated transformation</td></tr>
<tr><td>2</td><td>Second concatenated transformation</td></tr>
<tr><td>Not speci-fied</td><td>Same meaning as with 0 or 1</td></tr>
<tr><td colspan="2">Note:
Values not equal to 0, 1, 2 generate an error alarm.</td></tr>
<tr><td rowspan="4"><Par_1>, ..., <Par_n>, <Par_n+1>:</td><td colspan="3">Parameters for the concatenated transformations</td></tr>
<tr><td colspan="2"><Par_1>, ..., <Par_n></td><td>Parameters of the first transforma-tion in the chain
The actual number depends on the transformation type:
• TRAORI: 2 parameters
• TRACYL: 1 to 2 parameters</td></tr>
<tr><td colspan="2"><Par_n+1></td><td>Parameters of the second transfor-mation in the chain (TRAANG)
≙ angle of the inclined axis</td></tr>
<tr><td colspan="3">If parameters are not set, the defaults or the parameters last used take effect. Commas must be used to ensure that the specified pa-rameters are evaluated in the sequence in which they are expected, if default settings are to be effective for previous parameters. In par-ticular, a comma is required before at least one parameter, even though it is not necessary to specify <Trafo_No> - for example TRA-CON(, 3.7).</td></tr>
<tr><td>TRAFOOF:</td><td colspan="3">Deactivate the last activated (concatenated) transformation</td></tr>
</table>

Example

Program code	Comment
...	
N230 TRACON(1,45.)	; Activate first concatenated transformation.

Program code	Comment
	; The previously active transformation is automatically de-selected.
	; The angle for the inclined axis is 45°.
...	
N330 TRACON(2,40.)	; Activate second concatenated transformation.
	; The angle for the inclined axis is 40°.
...	
N380 TRAFOOF	; Deactivate second concatenated transformation.
...	

4.10 Kinematic chains

4.10.1 Deletion of components (DELOBJ)

The `DELOBJ()` function "deletes" components by resetting the assigned system variables to their default values:

- Elements from kinematic chains
- Protection areas, protection area elements and collision pairs
- Transformation data

Syntax

```
[<RetVal>=] DELOBJ(<CompType>[,,,<NoAlarm>)])
[<RetVal>=] DELOBJ(<CompType>,<Index1>[,,<NoAlarm>])
[<RetVal>=] DELOBJ(<CompType>[,<Index1>][,<Index2>][,<NoAlarm>])
```

Meaning

`DELOBJ`:	Deletion of elements from kinematic chains, protection areas, protection area elements, collision pairs and transformation data
`<CompType>`:	Component type to be deleted
	Data type: STRING
	Value: "KIN_CHAIN_ELEM" **Meaning**: System variables of all kinematic elements: $NK_...
	Value: "KIN_CHAIN_SWITCH" **Meaning**: System variable $NK_SWITCH[<i>]
	Value: "KIN_CHAIN_ALL" **Meaning**: All kinematic elements and switches. Is the same as the successive call of DELOBJ with "KIN_CHAIN_ELEM" and "KIN_CHAIN_SWITCH"
	Value: "PROT_AREA" **Meaning**: System variables of the protection areas: • $NP_PROT_NAME • $NP_CHAIN_NAME • $NP_CHAIN_ELEM • $NP_1ST_PROT
	Value: "PROT_AREA_ELEM" **Meaning**: System variables of the protection area elements of machine protection areas and/or automatic tool protection areas: • $NP_NAME • $NP_NEXT • $NP_NEXTP • $NP_COLOR • $NP_D_LEVEL • $NP_USAGE • $NP_TYPE • $NP_FILENAME • $NP_PARA • $NP_OFF • $NP_DIR • $NP_ANG
	Value: "PROT_AREA_COLL_PAIRS" **Meaning**: System variables of the collision pairs: • $NP_COLL_PAIR • $NP_SAFETY_DIST
	Value: "PROT_AREA_ALL" **Meaning**: All protection areas, protection area elements and collision pairs (system variable $NP_...) Is the same as the successive call of DELOBJ with "PROT_AREA," "PROT_AREA_ELEM," and "PROT_AREA_COLL_PAIRS"
	Value: "TRAFO_DATA" **Meaning**: System variables of all transformations $NT_...

<Index1>:	Index of the first component to be deleted (**optional**)	
	Data type:	INT
	Default value:	-1
	Value range:	-1 ≤ x ≤ (maximum number of configured components -1)
	Value	**Meaning**
	0, 1, 2, ...	Index of the component to be deleted
	-1	All components of the specified type are deleted. <Index2> is not evaluated.
<Index2>:	Index of the last components to be deleted (**optional**) If <Index2> is not programmed, only the system variables of the component referenced in <Index1> are deleted.	
	Data type:	INT
	Default value:	Only the system variables of the component referenced in <Index1> are deleted.
	Value range:	<Index1> < x ≤ (max. number of configured components -1)
<NoAlarm>:	Alarm suppression (**optional**)	
	Data type:	BOOL
	Default value:	FALSE
	Value	**Meaning**
	FALSE	In the event of an error (<RetVal> < 0), program processing is stopped and an alarm displayed.
	TRUE	In the event of an error, the program processing is not stopped and no alarm displayed. Application: User-specific reaction corresponding to the return value
<RetVal>:	Function return value	
	Data type:	INT
	Value range:	0, -1, -2, ... -7
	Value	**Meaning**
	0	No error occurred
	-1	Call of the function without parameters. At least parameter <CompType> must be specified.
	-2	<CompType> identifies an unknown component
	-3	<Index1> is less than -1
	-4	<Index1> is greater than the configured number of components
	-5	<Index1> has a value not equal to -1 when deleting a component **group**
	-6	<Index2> is less than <Index1>
	-7	<Index2> is greater than the configured number of components
	-8	Insufficient write permission

4.10.2 Index determination by means of names (NAMETOINT)

User-specific names are entered in the system variable arrays of type STRING. Based on the identifier of the system variables and the name, the `NAMETOINT()` function determines the index value belonging to the name under which it is stored in the system variable array.

Syntax

```
<RetVal> = NAMETOINT(<SysVar>,<Name>[,<NoAlarm>])
```

Meaning

`NAMETOINT`:	Determining the system variable index	
`<SysVar>`:	Name of the system variable array of typeSTRING	
	Data type:	STRING
	Range of values:	Name of all NC system variable arrays of type STRING
`<Name>`:	Character string or name for which the system variable index is to be determined.	
	Data type:	STRING
`<NoAlarm>`:	Alarm suppression (**optional**)	
	Data type:	BOOL
	Default value:	FALSE
	Value	**Meaning**
	TRUE	In the event of an error, the program processing is not stopped and no alarm displayed. Application: User-specific reaction corresponding to the return value
	FALSE	In the event of an error (<RetVal> < 0), program processing is stopped and an alarm displayed.
`<RetVal>`:	System variable index or error message	
	Data type:	INT
	Range of values:	-1 ≤ x ≤ (max. number of configured components -1)
	Value	**Meaning**
	≥ 0	The sought name has been found under the specified system variable index.
	-1	The sought name has not been found or an error has occurred.

Example

Program code	Comment
`DEF INT INDEX`	
`$NP_PROT_NAME[27]="Cover"`	
`...`	
`INDEX = NAMETOINT("$NP_PROT_NAME","Cover")`	`; INDEX == 27`

4.11 Collision avoidance with kinematic chains

Note

Protection areas

The protection areas specified in the following chapters refer to the "Geometric machine modeling" function.

Information about this function, see Function Manual Monitoring and Compensation.

4.11.1 Check for collision pair (COLLPAIR)

4.11.2 Request recalculation of the machine model of the collision avoidance (PROTA)

If system variables of the kinematic chain $NK_..., the geometric machine modeling or the collision avoidance $NP_... are written in the part program, the `PROTA` procedure must subsequently be called so that the change becomes effective in the NC-internal machine model of the collision avoidance.

Syntax

```
PROTA[(<Par>)]
```

Meaning

`PROTA`:	Request recalculation of the machine model of the collision avoidance		
	• Triggers a preprocessing stop. • Must be alone in the block.		
`<Par>`:	Parameter (**optional**)		
	Data type:	STRING	
	Value:	---	No parameters. The machine model is recalculated. The states of the protection areas are retained.
		"R"	The machine model is recalculated. The protection areas are set to their initialization status corresponding to $NP_INIT_STAT.

Supplementary conditions

Simulation

The `PROTA` procedure must not be used in part programs in conjunction with the simulation (simNC).

Example: Avoiding the PROTA call while the simulation is active.

```
Program code                    Comment
...
IF $P_SIM == FALSE              ; IF simulation not active
   PROTA                         THEN recalculate collision model
ENDIF                           ; ENDIF
...
```

4.11.3 Setting the protection zone status (PROTS)

The PROTS() procedure sets the state of protection areas to the specified value.

Syntax

```
PROTS(<State>[,<Name_1>,...,<Name_n>])
```

Meaning

PROTS:	Sets the state of protection areas		
	• Must be alone in the block.		
<State>:	Status to which the specified protection areas are to be set		
	Data type:	CHAR	
	Value:	"A"or "a"	Status: Active
		"I"or "i"	Status: Inactive
		"P"or "p"	Status: Preactivated or PLC-controlled [1)]
		"R"or "r"	Status: NC-internal value of the initialization status [2)]
<Name_1> ... <Name_n>:	Name of one or more protection areas that are to be set to the specified status (**optional**) If no name is specified, the specified status is set for all defined protection areas.		
	Data type:	STRING	
	Value range:	Parameterized protection area names	
	Note The maximum number of protection areas that can be specified as parameters depends only on the maximum possible number of characters per program line.		
[1)] The activation/deactivation is performed via: DB2600.DBX4.0..11.7			
[2)] The status is set to the NC-internal value of the initialization status, i.e. to the value that the system variable $NP_INIT_STAT had at the time of the last PROTA(...) (Page 666) call.			

4.11.4 Determining the clearance of two protection zones (PROTD)

The PROTD() function calculates the clearance of two protection areas.

Function properties:

- The clearance calculation is performed independent of the protection area status (activated, deactivated, preactivated).
- To calculate the clearance of two protection areas, only protection area elements are used, which are marked with $NP_USAGE = "C" or "A". Protection area elements of the protection area, which are marked with $NP_USAGE = "V", are not taken into consideration.
- Protection areas, where all protection area elements of the protection area are marked with $NP_USAGE = "V", cannot be used for the clearance calculation.
- The clearance calculation is performed with the positions valid at the end of the previous block.
- Overlays that are included in the main run calculation (e.g. DRF offset or external work offset) are included in the clearance calculation with the values valid at the function **interpretation time**.

Note

Synchronization

When using the `PROTD()` function, it is the sole responsibility of the user to synchronize the main run and preprocessing, if required, with the `STOPRE` preprocessing stop.

Collision

If there is a collision between the specified protection areas, the function returns a clearance of 0.0. There is a collision if both the protection areas touch or intersect each other.

The safety clearance for the collision check (MD10622 $MN_COLLISION_SAFETY_DIST) is not taken into account in the clearance calculation.

Syntax

```
[<RetVal> =] PROTD([<Name_1>],[<Name_2>],VAR <Vector>[,<System>])
```

Meaning

`PROTD:`	Calculates the clearance of the two specified protection areas.	
	• Must be alone in the block.	
`<RetVal>:`	Function return value: Absolute clearance value of the two protection areas or 0.0 with collision (see above: Collision paragraph)	
	Data type:	REAL
	Range of values:	0.0 ≤ x ≤ +max. REAL value
`<Name_1>,` `<Name_2>:`	Names of the two protection areas whose clearance is to be calculated (**optional**)	
	Data type:	STRING
	Range of values:	Parameterized protection area names
	Default value:	"" (empty string) If no protection areas have been specified, the function calculates the current smallest clearance from all the activated and preactivated protection areas in the collision model.

<Vector>:	Return value: 3-dimensional clearance vector from protection area <Name_2> to protection area <Name_1> with: • <Vector>[0]: X coordinate in the world coordinate system • <Vector>[1]: Y coordinate in the world coordinate system • <Vector>[2]: Z coordinate in the world coordinate system For collision: <Vector> == zero vector		
	Data type:	VAR REAL [3]	
	Range of values:	<Vector> [n]: 0.0 ≤ x ≤ ±max. REAL value	
<System>:	System of units (inch/metric) for clearance and clearance vector (**optional**)		
	Data type:	BOOL	
	Value:	FALSE (Default)	System of units corresponding to the currently active G command from G Group 13 (G70, G71, G700, G710).
		TRUE	System of units corresponding to the set basic system: MD52806 $MN_ISO_SCALING_SYSTEM

4.12 Transformation with kinematic chains

4.12.1 Activating transformation (TRAFOON)

A transformation defined with kinematic chains is activated with the predefined TRAFOON procedure. The call must be alone in a block.

Note

Alternatively, a transformation defined with kinematic chains can also be activated via conventional NC commands, such as TRAORI or TRANSMIT. For this purpose, an appropriate value, not equal to zero, must be entered in the $NT_TRAFO_INDEX system variable.

For more information on $NT_TRAFO_INDEX see "System Variables Parameter Manual".

Syntax

```
TRAFOON(<Trafoname>)
TRAFOON(<Trafoname>,<Diameter>,<k>)
TRAFOON(<Trafoname>,<γ>)
```

Meaning

<table>
<tr><td>TRAFOON</td><td colspan="3">Procedure for activating a transformation defined with kinematic chains</td></tr>
<tr><td rowspan="4"><Trafoname></td><td colspan="3">Name of the transformation data set</td></tr>
<tr><td>Data type:</td><td colspan="2">STRING</td></tr>
<tr><td>Value range:</td><td colspan="2">All names of transformation data sets defined via $NK_NAME</td></tr>
<tr><td colspan="3">Note:
The name of the transformation data set must be unique. It must only occur once in $NT_NAME.</td></tr>
<tr><td rowspan="3"><Diameter></td><td colspan="3">Reference or working diameter (TRACYL cylinder surface transformation only)</td></tr>
<tr><td>Data type:</td><td colspan="2">REAL</td></tr>
<tr><td colspan="3">The value must be > 1.</td></tr>
<tr><td rowspan="5"><k></td><td colspan="3">Defines the use of the slot side offset (TRACYL cylinder surface transformation only).</td></tr>
<tr><td>Data type:</td><td colspan="2">BOOL</td></tr>
<tr><td rowspan="2">Value:</td><td>FALSE</td><td>Without groove side offset</td></tr>
<tr><td>TRUE</td><td>With groove side offset</td></tr>
<tr><td colspan="3">Corresponds to the TRACYL transformation type 514 (groove side offset can be programmed). If <k> is not specified, the parameterized setting of bit 10 in $NT_CNTRL[<n>] applies.</td></tr>
<tr><td rowspan="5"><γ></td><td colspan="3">Alignment of the tool cutting edge to the circular tangent (TRAINT rotary interpolation transformation only)
Angle γ is used to compensate cutting edge parameters using function CUTMODK. The actual angle of the tool cutting edge must be realized by appropriately positioning the spindle.</td></tr>
<tr><td>Data type:</td><td colspan="2">INT</td></tr>
<tr><td>Value range:</td><td colspan="2">0, 180</td></tr>
<tr><td rowspan="2">Value:</td><td>0</td><td>The cutting edge is perpendicular to the tangent of the circle and is outside the circle (→ outside machining).</td></tr>
<tr><td>180</td><td>The cutting edge is also perpendicular to the tangent of the circle and is inside the circle (→ inside machining).</td></tr>
</table>

Example

Program code	Comment
`TRAFOON["Trans_1"]`	`Activates the transformation with the name Trans_1.`

4.12.2 Activating/deactivating rotary interpolation transformation TRAINT

Rotary interpolation transformation TRAINT is activated in the NC program using instruction TRAFOON and deactivated using instruction TRAFOOF.

Certain conditions must be fulfilled for activation; otherwise, this is rejected and an alarm is output.

Conditions for activation

- The center of rotation must be defined as zero point of the current frame (WCS zero point).
- The direction of the axis of rotation corresponds to the Z axis in the WCS.
- To activate rotary interpolation transformation TRAINT, it is not permissible that the machine moves, i.e. the axes do not traverse and the spindle does not rotate.
- The direction of the tool spindle is antiparallel to the z direction of the WCS.
- The axes are positioned so that after TRAFOON (and after taking into account the tool length), the y value in the WCS is equal to 0.
- The spindle is turned so that the desired position of the tool cutting edge to the workpiece is achieved.

Note

As the spindle points anti-parallel to the Z axis, the negative angle must be approached.

- Angle γ is used to compensate cutting edge parameters using function CUTMODK. The actual angle of the tool cutting edge must be realized by appropriately positioning the spindle.
- Angle γ must be 0° or 180°.

Activating a transformation

Once all conditions have been satisfied, rotary interpolation transformation TRAINT can be activated using instruction TRAFOON:

```
TRAFOON(<transformation name>,<γ>)
```

The following rules apply:

- The transformation name must be defined using system variable $NT_NAME[<n>].
- The transformation type must be specified as follows using system variable $NT_TRAFO_TYPE[<n>]:
  `$NT_TRAFO_TYPE[<n>]="TRAINT"`
- Angle γ is used for the relative alignment of the tool cutting edge to the tangent of the circle:

γ = 0	The cutting edge is perpendicular to the tangent of the circle and is outside the circle (→ outside machining).
γ = 180°	The cutting edge is also perpendicular to the tangent of the circle and is inside the circle (→ inside machining).

Both system variables are part of the transformation based on the kinematic chain.

Deactivating a transformation

It is only possible to deactivate the transformation with the spindle at standstill and is carried out using the TRAFOOF instruction.

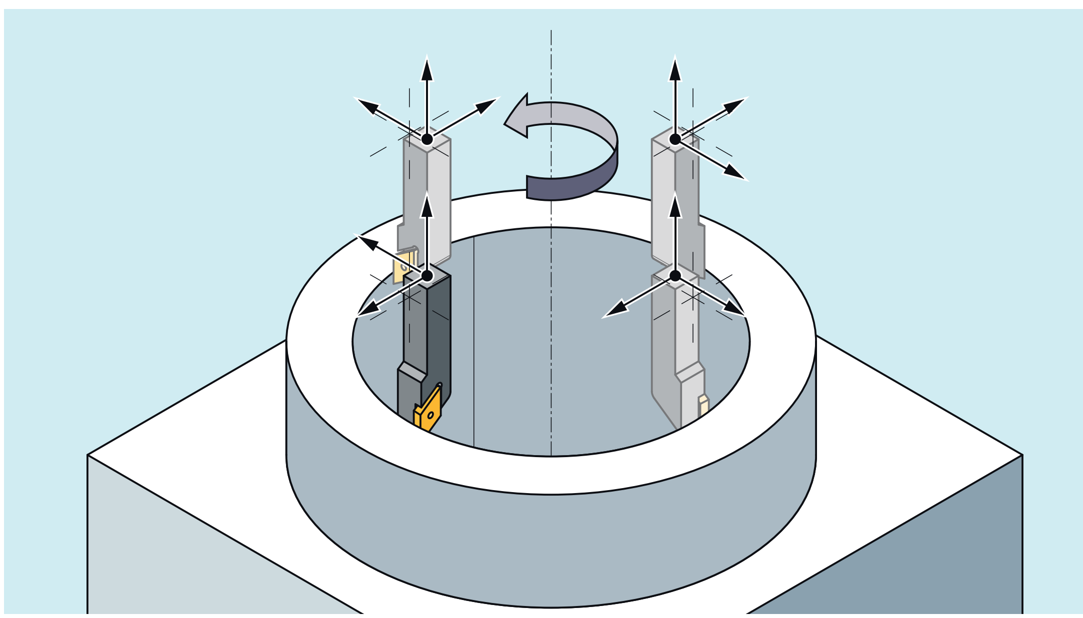
More information

Function

The rotary interpolation transformation TRAINT – also called interpolation turning – is used to provide users on a suitable machine tool (e.g. milling machine, turning machine with three linear axes) with the environment of a simple turning machine so that the NC commands and cycles function as on a turning machine.

Using the transformation, rotary motion is translated into circular movements of the linear axes. Correspondingly, the tool cutting edge is always aligned with the center of rotation during the motion. The rotary motion is not realized using an axis of rotation of the machine, but through traversing motion of the linear axes (x, y and z), which results in circular motion. This distinguishes interpolation turning from the "Turning on a milling machine" functionality.

The axis of rotation can be selected as the center of the turning operations and of the spindle at any mechanically accessible workpiece position and can be freely oriented in space for 4 or 5-axis machines.



NOTICE
Linear axes coupled to a tool spindle
The linear axes are coupled to the tool spindle during interpolation turning. A rotation of the spindle automatically results in motions of the linear axes. This is especially true if there is no path motion, the path motion is stopped due to G4, or the path override is set to 0.
In contrast to a turning machine, an NC stop also stops the spindle motion.

Machine kinematics as a kinematic chain

Interpolation turning is only implemented as transformation based on a kinematic chain.

The following conditions apply:

- The axis of rotation is always parallel to the direction of the tool spindle (z axis when turning).
- The axis of rotation can be freely oriented in space, for example with CYCLE800 or frames, but must remain fixed during the rotation.
- Spindle direction and Z axis of the current WCS must be in parallel.
- The center of the rotary interpolation transformation must lie in the current zero point of the WCS.

The rotation is translated into movements of the linear axes through the transformation. Correspondingly, the tool cutting edge is always aligned with the center of rotation during the motion.

CYCLE806 (this option requires a license)

CYCLE806 (Page 973) can also be used to program interpolation turning.

Tool length compensation

Directly after activating the transformation, the WCS position y must be equal to 0. Here it should be noted that without transformation the tool lengths in the BCS are used without taking the spindle position (i.e. the actual orientation of the tool) into account.

Without transformation, a difference between the tool tip and the WCS position therefore occurs when the tool is rotated (SPOS unequal to 0 or unequal to clamping angle). This must be corrected accordingly before TRAFOON (generally, y is then not equal to 0 in the WCS, so that after TRAFOON and taking into account the rotation of the tool y is equal to 0).

Variables $P_TRAINT_ROT_ANGLE and $P_TRAINT_SPOS_ANGLE support users in establishing the WCS position y equal to 0.

- $P_TRAINT_ROT_ANGLE:
  The $P_TRAINT_ROT_ANGLE variable reads the correct frame rotation for selection of the transformation TRAINT.
- $P_TRAINT_SPOS_ANGLE:
  Variable $P_TRAINT_SPOS_ANGLE reads the correct position of the spindle for selection of transformation TRAINT.

Velocity control

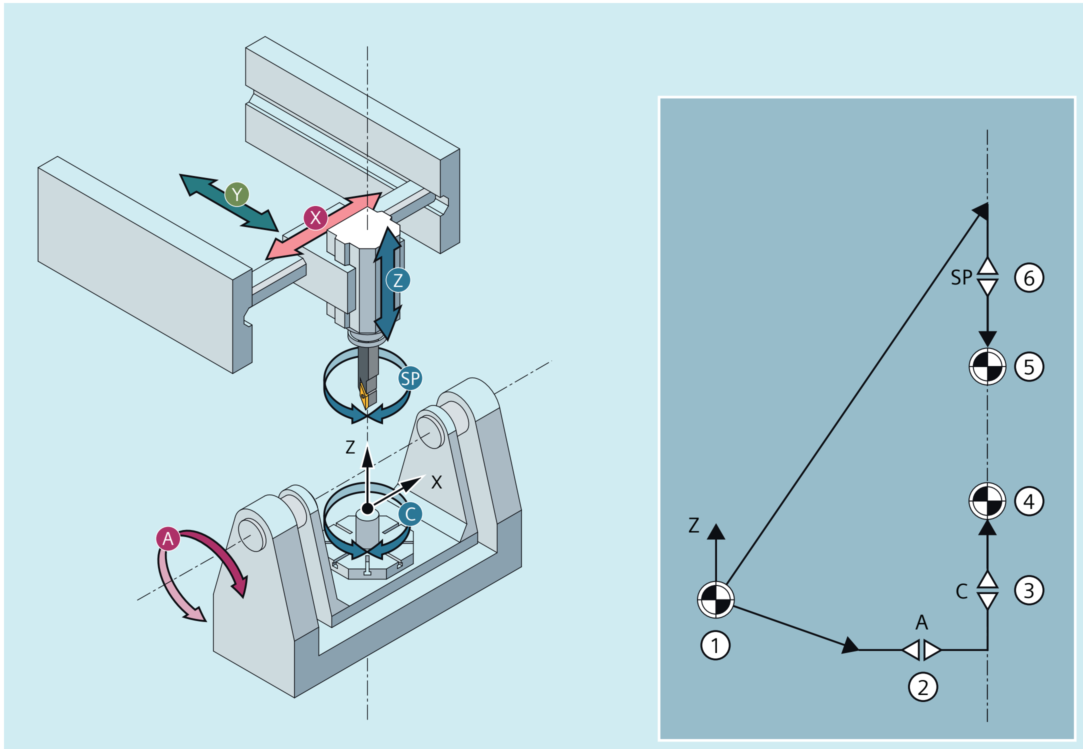
On a turning machine, the total motion is the result of the rotary motion by the turning spindle and the path motion by the geometry axes X and Z. An axis of the path motion has no influence on the rotary motion and vice versa. This independence does not exist with interpolation turning. The machine axes X, Y and Z provide both rotational and path motion.

You can set the weighting of the shares via machine data (value range 0.001 to 0.999).

More information: Function Manual Transformations

Example of kinematics

A 5-axis milling machine in AC table kinematics is shown in the following example.



① World coordinate system
② Rotary axis A
③ Rotary axis C
④ Workpiece reference point (end of the part chain)
⑤ Tool reference point (end of the tool chain)
⑥ Spindle

4.12.3 Calculate angle for aligning the tool for TRAINT (CALCTRAVAR)

For the TRAINT rotary interpolation transformations, the spindle and axes must be positioned so that the tool tip is oriented toward the center of rotation and is at y = 0. Complex calculations are required to determine the appropriate angles and positions. Support for this task is provided by the CALCTRAVAR function.

Precondition

Before calling CALCTRAVAR, the kinematic chain must be defined.

Note

CALCTRAVAR calculates the angles based on the defined kinematic chain. Changes in the kinematic chain **after** calling CALCTRAVAR can therefore lead to the values calculated by CALCTRAVAR no longer being correct.

Syntax

```
<Status> = CALCTRAVAR(<Result>, "<Transformer name>", <γ>)
```

Meaning

<table>
<tr><td><code>CALCTRAVAR(...)</code></td><td colspan="3">Predefined function: Calculate angle for aligning the tool for TRAINT</td></tr>
<tr><td rowspan="3"><code><Result></code></td><td colspan="3">2-dimensional result variable
After calling CALCTRAVAR, the values required for pre-positioning are stored here:
• <Result>[0]: Correct position of the spindle for selection of transformation TRAINT
• <Result>[1]: Correct frame rotation for selecting the transformation TRAINT</td></tr>
<tr><td>Data type:</td><td colspan="2">Real</td></tr>
<tr><td colspan="3">Note:
In case of error (<status> ≠ 0), the returned angles have the value 0.</td></tr>
<tr><td rowspan="3"><code><Transformer name></code></td><td colspan="3">Name of the TRAINT transformation data set to be activated later with TRAFOON</td></tr>
<tr><td>Data type:</td><td colspan="2">STRING</td></tr>
<tr><td>Value range:</td><td colspan="2">All names of TRAINT transformation data sets defined via $NK_NAME</td></tr>
<tr><td rowspan="6"><γ></td><td colspan="3">Alignment of the tool cutting edge to the circular tangent
Angle γ is used to compensate cutting edge parameters using function CUT-MODK.</td></tr>
<tr><td>Data type:</td><td colspan="2">INT</td></tr>
<tr><td>Value range:</td><td colspan="2">0, 180</td></tr>
<tr><td rowspan="2">Value:</td><td>0</td><td>The cutting edge is perpendicular to the tangent of the circle and is outside the circle (→ outside machining).</td></tr>
<tr><td>180</td><td>The cutting edge is also perpendicular to the tangent of the circle and is inside the circle (→ inside machining).</td></tr>
<tr><td colspan="3">Note:
If the parameter is not programmed, the value 0 is assumed by default.</td></tr>
</table>

`<Status>`	Return value for the function status		
	Data type:	INT	
	Value range:	0 ... 8	
	Value:	0	No error, call was successful
		1	Parameter <Name> is missing or empty
		2	Parameter <γ> has an invalid value (may only be 0 or 180)
		3	Error in the preparation of the kinematic chain
		4	Spindle parameterized for TRAINT not found in the kinematic chain
		5	No spindle has been parameterized for TRAINT
		6	Parameter <Name> does not denote a defined transformation
		7	General fault
		8	No tool programmed

4.13 Tool offsets

4.13.1 Offset memory

Structure of the offset memory

Every data field can be called with a T and D number, and contains not only the geometric specifications for the tool but also further entries, such as the tool type.

User cutting edge data

User cutting edge data can be configured via machine data. Please refer to the machine manufacturer's instructions.

Tool parameters

Note

Individual values in the offset memory

The individual values of the offset memory P1 to P25 can be read and written by the program via system variables. All other parameters are reserved.

The tool parameters $TC_DP6 to $TC_DP8, $TC_DP10 and $TC_DP11 as well as $TC_DP15 to $TC_DP17, $TC_DP19 and $TC_DP20 have another meaning depending on tool type.

Tool parameter number (DP)	Meaning of system variables	Remark
$TC_DP1	Tool type	For overview see list
$TC_DP2	Cutting edge position	Only for turning tools
Geometry	**Length compensation**	
$TC_DP3	Length 1	Allocation to
$TC_DP4	Length 2	Type and level
$TC_DP5	Length 3	
Geometry	**Radius**	
$TC_DP6 [1)] $TC_DP6 [2)]	Radius 1 / length 1 diameter d	Milling/turning/grinding tool Slotting saw
$TC_DP7 [1)] $TC_DP7 [2)]	Length 2 / corner radius, tapered milling tool Slot width b corner radius	Milling tools Slotting saw
$TC_DP8 [1)] $TC_DP8 [2)]	Rounding radius 1 for milling tools projecting length k	Milling tools Slotting saw
$TC_DP9 [1) 3)]	Rounding radius 2	Reserved
$TC_DP10 [1)]	Angle 1 face end of tool	Tapered milling tools
$TC_DP11 [1)]	Angle 2 tool longitudinal axis	Tapered milling tools
Wear	**Length and radius compensation**	
$TC_DP12	Length 1	
$TC_DP13	Length 2	
$TC_DP14	Length 3	
$TC_DP15 [1)] $TC_DP15 [2)]	Radius 1 / length 1 diameter d	Milling/turning/grinding tool Slotting saw
$TC_DP16 [1)] $TC_DP16 [3)]	Length 2 / corner radius, tapered milling tool, slot width b corner radius	Milling tools Slotting saw
$TC_DP17 [1)] $TC_DP17 [2)]	Rounding radius 1 for milling tools projecting length k	Milling / 3D face milling Slotting saw
$TC_DP18 [1) 3)]	Rounding radius 2	Reserved
$TC_DP19[1)]	Angle 1 face end of tool	Tapered milling tools
$TC_DP20[1)]	Angle 2 tool longitudinal axis	Tapered milling tools
Tool base dimension/ adapter	**Length offsets**	
$TC_DP21	Length 1	
$TC_DP22	Length 2	
$TC_DP23	Length 3	
Technology		
$TC_DP24	Clearance angle	Only for turning tools
$TC_DP25		Reserved

1) Also applies with milling tools for 3D face milling

2) For slotting saw tool type

3) reserved

Remarks

Several entry components are available for geometric variables (e.g. length 1 or radius). These are added together to produce a value (e.g. total length 1, total radius), which is then used for the calculations.

Offset values not required must be assigned the value zero.

Tool parameters $TC-DP1 to $TC-DP23 with contour tools

Note

The tool parameters not listed in the table, such as $TC_DP7, are not evaluated, i.e. their content is meaningless.

Tool parameter number (DP)	Meaning	Cutting Dn		Remark
$TC_DP1	Tool type			400 to 599
$TC_DP2	Cutting edge position			
Geometry	**Length compensation**			
$TC_DP3	Length 1			
$TC_DP4	Length 2			
$TC_DP5	Length 3			
Geometry	**Radius**			
$TC_DP6	Radius			
Geometry	**Limit angle**			
$TC_DP10	Minimum limit angle			
$TC_DP11	Maximum limit angle			
Wear	**Length and radius compensation**			
$TC_DP12	Wear length 1			
$TC_DP13	Wear length 2			
$TC_DP14	Wear length 3			
$TC_DP15	Wear radius			
Wear	**Limit angle**			
$TC_DP19	Wear min. limit angle			
$TC_DP20	Wear max. limit angle			
Tool base dimension/ adapter	**Length offsets**			
$TC_DP21	Length 1			
$TC_DP22	Length 2			
$TC_DP23	Length 3			

Basic value and wear value

The resultant values are each a total of the basic value and wear value (e.g. $TC_DP6 + $TC_DP15 for the radius). The basic measurement ($TC_DP21 – $TC_DP23) is also added to the tool length of the first cutting edge. All the other parameters, which may also impact on effective tool length for a standard tool, also affect this tool length (adapter, orientatable toolholder, setting data).

Limit angles 1 and 2

Limit angles 1 and 2 each refer to the vector of the cutting edge center point to the cutting edge reference point and are counted counterclockwise.

4.13.2 Additive offsets

4.13.2.1 Selecting additive offsets (DL)

Additive offsets can be considered as process offsets that can be programmed in the machining. They refer to the geometrical data of a cutting edge and are therefore a component of tool cutting data.

Data of an additive offset is addressed using a DL number (DL: Locationdependent; offsets regarding the location of use) and entered via the user interface.

Application

Dimension errors caused be the location of use can be compensated using additive offsets.

Syntax

```
DL=<number>
```

Meaning

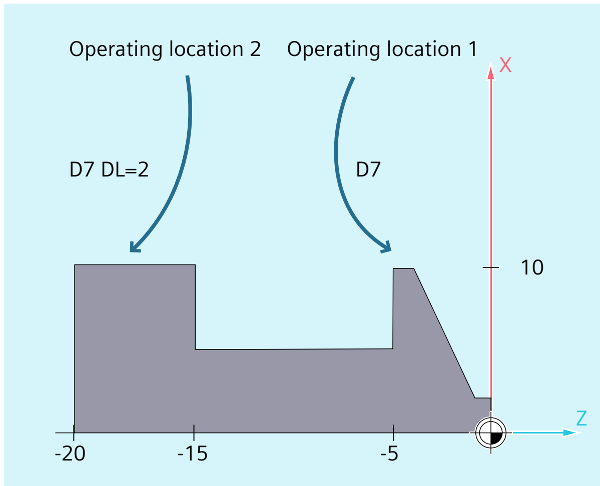
`DL`:	Command to activate an additive offset
`<number>`:	The additive tool offset data to be activated is specified using the `<number>` parameter

Note

The machine data is used to define the number of additive offsets and also activate them (→ carefully observe the machine OEM's data!).

Example

The same cutting edge is used for two bearing seats:



Program code	Comment
N110 T7 D7	; The revolver is positioned to location 7. D7 and DL=1 are activated and moved through in the next block.
N120 G0 X10 Z1	
N130 G1 Z-6	
N140 G0 DL=2 Z-14	; DL=2 is activated in addition to D7 and is moved through in the next block.
N150 G1 Z-21	
N160 G0 X200 Z200	; Approach tool change point.
...	

4.13.2.2 Specify wear and setup values ($TC_SCPxy[t,d], $TC_ECPxy[t,d])

4.13.2.3 Delete additive offsets (DELDL)

The DELDL command deletes the additive offsets for the cutting edge of a tool (to release memory space). Both the defined wear values and the setup values are deleted.

Syntax

```
DELDL[<t>,<d>]
DELDL[<t>]
DELDL
<Status>=DELDL[<t>,<d>]
```

Meaning

`DELDL`:	Command to delete additive offsets	
`<t>`:	T number of the tool	
`<d>`:	D number of the tool cutting edge	
`DELDL[<t>,<d>]`:	All additive offsets of the cutting edges `<d>` of the tool `<t>` are deleted.	
`DELDL[<t>]`:	All additive offsets of all cutting edges of tool `<t>` are deleted.	
`DELDL`:	All additive offsets of all cutting edges of all tools of the TO unit are deleted (for the channel in which the command is programmed).	
`<Status>`:	Delete status	
	Value:	Meaning:
	0	Deletion was successfully completed.
	-	Offsets have not been deleted (if the parameter settings specify exactly one tool edge), or not deleted completely (if the parameter settings specify several cutting edges).

Note

Wear and setting-up values of active tools cannot be deleted (essentially the same as the delete behavior of `D` or tool data).

4.13.3 Special handling of tool offsets

The evaluation of the sign for tool length and wear can be controlled using setting data SD42900 to SD42960.

The same applies to the behavior of the wear components when mirroring geometry axes or changing the machining plane, and also to temperature compensation in tool direction.

Wear values:

If reference is made to wear values in the following, then this should be understood as the sum of the actual wear values ($TC_DP12 to $TC_DP20) and the sum offsets with the wear values ($SCPX3 to $SCPX11) and setting-up values ($ECPX3 to $ECPX11).

Information about summed offsets, see Function Manual Tool Management.

Setting data

SD42900 $SC_MIRROR_TOOL_LENGTH	Mirroring of tool-length components and components of the tool base dimension.
SD42910 $SC_MIRROR_TOOL_WEAR	Mirroring of wear values of the tool-length components.
SD42920 $SC_WEAR_SIGN_CUTPOS	Evaluating the sign of the wear components as a function of the cutting edge position.
SD42930 $SC_WEAR_SIGN	Inverts the sign of wear dimensions.

SD42935 $SC_WEAR_TRANSFORM	Transformation of wear values.
SD42940 $SC_TOOL_LENGTH_CONST	Assignment of tool length components to geometry axes.
SD42950 $SC_TOOL_LENGTH_TYPE	Assignment of the tool length components independent of tool type.
SD42960 $SC_TOOL_TEMP_COMP	Temperature compensation value in tool direction. Also operative when tool orientation is programmed.

Further information

Activation of modified setting data

When the setting data described above is 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 is to be reevaluated, the tool must be selected again.

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.

Orientable toolholders and new setting data

Setting data SD42900 to SD42940 has no effect on the components of an active toolholder with orientation capability. However, the calculation with an orientable toolholder always allows for a tool with its total resulting length (tool length + wear + tool base dimension). All modifications initiated by the setting data are included in the calculation of the resulting total length, i.e. vectors of the orientable toolholder are independent of the machining plane.

Note

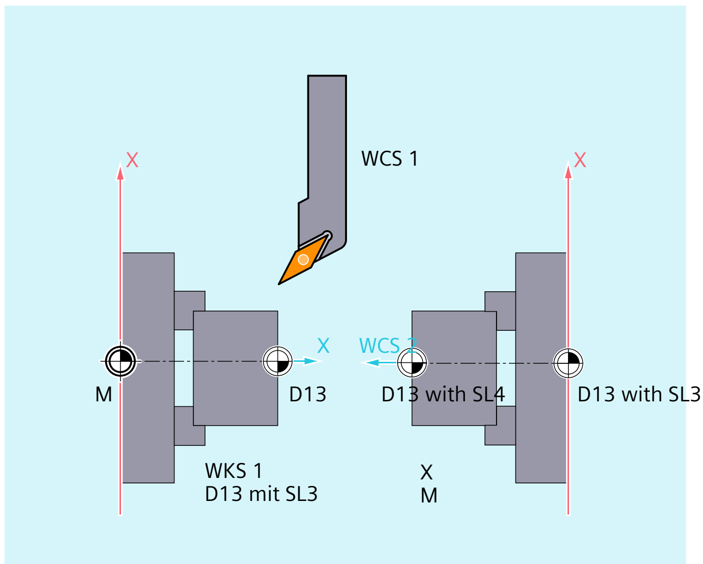
When orientable toolholders are used, it is frequently practical to define all tools for a non-mirrored basic system, even those which are only used for mirrored machining. When machining with mirrored axes, the toolholder is then rotated such that the actual position of the tool is described correctly. All tool-length components then automatically act in the correct direction, dispensing with the need for control of individual component evaluation via setting data, depending on the mirroring status of individual axes.

Further application options

The use of orientable toolholder functionality can also be useful if there is no physical option of turning tools on the machine, even though tools with different orientations are permanently installed. Tool dimensioning can then be performed uniformly in a basic orientation, where the dimensions relevant for machining are calculated according to the rotations of a virtual toolholder.

4.13.3.1 Mirroring of tool lengths

When setting data SD42900 $SC_MIRROR_TOOL_LENGTH and SD42910 $SC_MIRROR_TOOL_WEAR are not set to zero, then you can mirror the tool length components and components of the basis dimensions with wear values and their associated axes.



SD42900 $SC_MIRROR_TOOL_LENGTH

Setting data **not equal to** zero:

The tool length components ($TC_DP3, $TC_DP4 and $TC_DP5) and the components of the basis dimensions ($TC_DP21, $TC_DP22 and $TC_DP23) are mirrored against their associated axes, also mirrored – by inverting the sign.

The wear values are **not** mirrored. If these are also be be mirrored, then setting data SD42910 $SC_MIRROR_TOOL_WEAR must be set.

SD42910 $SC_MIRROR_TOOL_WEAR

Setting data **not equal to** zero:

The wear values of the tool length components - whose associated axes are mirrored - are also mirrored by inverting the sign.

4.13.3.2 Wear sign evaluation

When setting data SD42920 $SC_WEAR_SIGN_CUTPOS and SD42930 $SC_WEAR_SIGN are set not equal to zero, then you can invert the sign evaluation of the wear components.

SD42920 $SC_WEAR_SIGN_CUTPOS

Setting data **not equal to** zero:

For tools with the relevant cutting edge position (turning and grinding tools, tool types 400), then the sign evaluation of the wear components in the machining plane depends on the cutting edge position. This setting data is of no significance for tool types without relevant cutting edge position.

In the following table, the dimensions, whose sign is inverted using SD42920 (not equal to zero), are designed using an X:

Cutting edge position	Length 1	Length 2
1		
2		X
3	X	X
4	X	
5		
6		
7		X
8	X	
9		

Note

The sign evaluation using SD42920 and SD42910 are independent of one another. If, for example, the sign of a dimension is changed using both setting data, then the resulting sign remains unchanged.

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.

If a positive wear dimension is entered, the tool becomes "shorter" and "thinner", refer to Chapter "tool offset, special handling", activating changed setting data".

4.13.3.3 Coordinate system of the active machining operation (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS)

Depending on the kinematics of the machine or the availability of an orientable tool carrier, the wear values measured in one of these coordinate systems are converted or transformed to a suitable coordinate system.

Coordinate systems of active machining operation

The following coordinate systems produce tool length offsets which the tool length wear component incorporates in an active tool via the corresponding G command of Group 56:

- Machine coordinate system (MCS)
- Basic coordinate system (BCS)
- Workpiece coordinate system (WCS)
- Tool coordinate system (TCS)
- Tool coordinate system of kinematic transformation (KCS)

Syntax

```
TOWSTD
TOWMCS
TOWWCS
TOWBCS
TOWTCS
TOWKCS
```

Meaning

`TOWSTD`:	Initial setting value for offsets in tool length wear value
`TOWMCS`:	Offsets in tool length in MCS
`TOWWCS`:	Offsets in tool length in WCS
`TOWBCS`:	Offsets in tool length in BCS
`TOWTCS`:	Offsets in tool length at tool carrier reference point (orientable tool carrier)
`TOWKCS`:	Compensations of tool length for tool head (kinematic transformation)

Further information

Distinguishing features

The most important distinguishing features are shown in the following table:

G command	Wear value	Active orientable tool carrier
TOWSTD	Initial value, tool length	Wear values are subject to rotation.
TOWMCS	Wear value in MCS. TOWMCS is identical to TOWSTD if a tool carrier that can be orientated is not active.	It only rotates the vector of the resultant tool length without taking into account the wear.
TOWWCS	The wear value is converted to the MCS in the WCS.	The tool vector is calculated as for TOWMCS without taking into account the wear.
TOWBCS	The wear value is converted to the MCS in the BCS.	The tool vector is calculated as for TOWMCS without taking into account the wear.
TOWTCS	The wear value is converted to the MCS in the workpiece coordinate system.	The tool vector is calculated as for TOWMCS without taking into account the wear.

TOWWCS, TOWBCS, TOWTCS: The wear vector is added to the tool vector.

Linear transformation

The tool length can be defined meaningfully in the MCS only if the MCS is generated by linear transformation from the BCS.

Non-linear transformation

For example, if with TRANSMIT a non-linear transformation is active, then when specifying the MCS as requested coordinate system, BCS is automatically used.

No kinematic transformation and no orientable tool carrier

If neither a kinematic transformation nor an orientable tool carrier is active, then all the other four coordinate systems (except for the WCS) are combined. It is then only the WCS, which is different to the other systems. Since only tool lengths need to be evaluated, translations between the coordinate systems are irrelevant.

Inclusion of wear values in calculation

The setting data **SD42935 $SC_WEAR_TRANSFORM** defines which of the three wear components:

- Wear
- Total offsets fine
- Total offsets coarse

should be subject to a rotation using adapter transformation or a tool carrier that can be orientated if one of the following G commands is active:

- TOWSTD
  Basic position. For corrections in the 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 tool coordinate system at the tool carrier fixture (T tool carrier reference).
- TOWKCS
  Wear values in the coordinate system of the tool head for kinematic transformation.

Note

Evaluation of individual wear components (assignment to geometry axes, sign evaluation) is influenced by the following factors:

- Active plane
- Adapter transformation
- Setting data:
  - SD42910 $SC_MIRROR_TOOL_WEAR
  - SD42920 $SC_WEAR_SIGN_CUTPOS
  - SD42930 $SC_WEAR_SIGN
  - SD42940 $SC_TOOL_LENGTH_CONST
  - SD42950 $SC_TOOL_LENGTH_TYPE

4.13.3.4 Tool length and plane change

When setting data SD42940 $SC_TOOL_LENGTH_CONST is set not equal to zero, then you can assign the tool length components – such as lengths, wear and basic dimension – to the geometry axes for turning and grinding tools when changing the plane.

SD42940 $SC_TOOL_LENGTH_CONST

Setting data **not equal to** zero:

The assignment of tool length components (length, wear and tool base dimension) to geometry axes does not change when the machining plane is changed (`G17` - `G19`).

The following table shows the assignment of tool length components to geometry axes for turning and grinding tools (tool types 400 to 599):

Content	Length 1	Length 2	Length 3
17	Y	X	Z
*)	X	Z	Y
19	Z	Y	X
-17	X	Y	Z
-18	Z	X	Y
-19	Y	Z	X

*) Each value not equal to 0, which is not equal to one of the six listed values, is evaluated as value 18.

The following table shows the assignment of tool length components to geometry axes for all other tools (tool types < 400 or > 599):

Operating plane	Length 1	Length 2	Length 3
*)	Z	Y	X
18	Y	X	Z
19	X	Z	Y
-17	Z	X	Y
-18	Y	Z	X
-19	X	Y	Z

*) Each value not equal to 0, which is not equal to one of the six listed values, is evaluated as value 17.

Note

For representation in tables, it is assumed that geometry axes up to 3 are designated with X, Y, Z. The axis order and not the axis identifier determines the assignment between a compensation and an axis.

4.13.4 Online tool offset

4.13.4.1 Defining a polynomial function (FCTDEF)

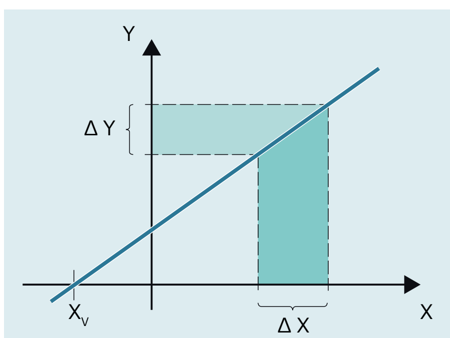
Certain dressing strategies (e.g. dressing roller) are characterized by the fact that the grinding wheel radius is continuously (linearly) reduced as the dressing roller is fed in. This strategy requires a linear function between infeed of the dressing roller and writing the wear value of each length. The linear function is defined using the predefined procedure FCTDEF(...) for up to third order polynomial functions.

Straight line equation

$y = f(x) = a_0 + a_1 * x_1$

a_1: Gradient of the straight line, with $a_1 = \Delta x / \Delta y$

a_0: Shift of the straight line along the X axis with $a_0 = -a1 * X_V$



Syntax

```
FCTDEF(<Func>,<LLimit>,<ULimit>,<a0>,<a1>,<a2>,<a3>)
```

Meaning

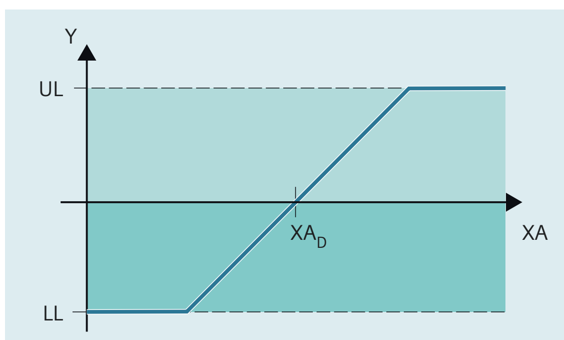
`FCTDEF(...):`	Defining a polynomial function for PUTFTOCF(...): $y = f(x) = a_0 + a_1*x + a_2*x^2 + a_3*x^3$	
`<Func>:`	Function number	
	Data type:	INT
	Range of values:	1, 2, 3
`<LLimit>:`	Lower limit value	
	Data type:	REAL
`<ULimit>:`	Upper limit value	
	Data type:	REAL
`<a0>,<a1>,<a2>,<a3>:`	Coefficients of polynomial function	
	Data type:	REAL

Example

Definitions

- Function number: 1
- Lower and upper limit value: -100, 100
- Gradient of the characteristic: $a_1 = 1$
- The operating point should be located at the center of the characteristic. Based on the setpoint position of axis XA in the WCS at the instant that the function is defined in the NC program, the characteristic must be shifted in the negative Y direction: $a_0 = -a_1 * XA_D = -1 *$ $AA_IW
- $a_2 = a_3 = 0$

Characteristic



UL	Upper limit value
LL	Lower limit value
XA_D	Setpoint of axis XA at the time that the function is defined in the NC program

Programming

Program code	Comment
`FCTDEF(1,-100,100,-$AA_IW[XA],1)`	`; Function definition`

4.13.4.2 Write online tool offset continuously (PUTFTOCF)

Using the predefined procedure PUTFTOCF(...), an online tool offset is executed based on a polynomial function previously defined with FCTDEF(...) (Page 688).

Note

The online tool offset can also be realized using a synchronized action.

For further information, see Function Manual Synchronized Actions.

Syntax

```
PUTFTOCF(<Func>,<RefVal>,<ToolPar>,<Chan>,<Sp>)
```

Meaning

PUTFTOCF(...):	Write online tool offset, continuously block-by-block using the polynomial function defined with FCTDEF(...)	
<Func>:	Function number, defined in the function definition with FCTDEF(...)	
	Data type:	INT
	Range of values:	1, 2, 3
<RefVal>:	Reference value, from which the offset is to be derived (e.g. setpoint of an axis).	
	Data type:	VAR REAL
<ToolPar>:	Number of the wear parameter (length 1, 2 or 3) in which the offset value is to be included.	
	Data type:	INT
<Chan>:	Number of the channel in which the online tool offset is to take effect. **Note:** Only required if the offset is not to take effect in the active channel.	
	Data type:	INT
<Sp>:	Number of the spindle for which the online tool offset is to take effect. **Note:** Only required if the offset is to be applied to a non-active grinding wheel rather than the active tool that is currently in use.	
	Data type:	INT

4.13.4.3 Write online tool offset, discrete (PUTFTOC)

Function

Using the predefined procedurePUTFTOC(...), an online tool offset is executed based on a fixed offset value.

Syntax

```
PUTFTOC(<CorrVal>,<ToolPar>,<Chan>,<Sp>)
```

Meaning

PUTFTOC(...):	Write online tool offset	
<CorrVal>:	Offset value, which is added to the wear parameter.	
	Data type:	REAL
<ToolPar>:	Number of the wear parameter (length 1, 2 or 3) in which the offset value is to be included.	
	Data type:	INT
<Chan>:	Number of the channel in which the online tool offset is to take effect. **Note:** Only required if the offset is not to take effect in the active channel.	
	Data type:	INT

`<Sp>`:	Number of the spindle for which the online tool offset is to take effect. **Note:** Only required if the offset is to be applied to a non-active grinding wheel rather than the active tool that is currently in use.	
	Data type:	INT

4.13.4.4 Activate/deactivate online tool offset (FTOCON/FTOCOF)

The online tool offset is activated or deactivated using the G commands FTOCON and FTOCOF.

Syntax

```
FTOCON
...
FTOCOF
```

Meaning

`FTOCON`:	Activate online tool offset The command must be programmed in the channel in which the online tool offset is to be activated.
`FTOCOF`:	Deactivate online tool offset The command must be programmed in the channel in which the online tool offset is to be deactivated. **Note:** On FTOCOF, the axis does not move further out for the tool offset. However, the value calculated with PUTFTOC/PUTFTOCF remains in the cutting-specific offset data. To finally deactivate the online tool offset, the tool (T...) must again be selected/deselected after FTOCOF.

4.13.5 Free assignment of D numbers, cutting edge numbers

4.13.5.1 Free assignment of D numbers, cutting edge numbers (CE address)

D number

The D numbers can be used as offset numbers. The number of the cutting edge can also be addressed via the CE address. The cutting edge number can be written by the system variable $TC_DPCE.

Default setting: Compensation no. == cutting edge no.

Machine data are used to define the maximum number of D numbers (cutting edge numbers) and the maximum number of cutting edges per tool (→ machine manufacturer).

The following commands are only practical if the maximum cutting edge number (MD18105) was specified to be greater than the number of cutting edges per tool (MD18106). Observe the machine manufacturer's specifications.

Further information

Function Manual Tools

4.13.5.2 Free assignment of D numbers: Checking D numbers (CHKDNO)

Using the `CKKDNO` command, you can check whether the existing D numbers were uniquely assigned. The D numbers of all tools defined within a TO unit may not occur more than once. No allowance is made for replacement tools.

Syntax

```
state=CHKDNO(Tno1,Tno2,Dno)
```

Meaning

`state`:	=TRUE:	The D numbers are assigned uniquely to the checked areas.
	= FALSE:	There was a D number collision or the parameters are invalid. Tno1, Tno2 and Dno return the parameters that caused the collision. These data can now be evaluated in the part program.
`CHKDNO(Tno1,Tno2)`:	All D numbers of the part specified are checked.	
`CHKDNO(Tno1)`:	All D numbers of Tno1 are checked against all other tools.	
`CHKDNO`:	All D numbers of all tools are checked against all other tools.	

4.13.5.3 Free assignment of D numbers: Rename D numbers (GETDNO, SETDNO)

You must assign unique D numbers. Two different cutting edges of a tool must not have the same D number.

GETDNO

This command returns the D number of a particular cutting edge (ce) of a tool with tool number t. If no D number exists for the entered parameters, d=0 will be set. If the D number is invalid, a value greater than 32000 is returned.

SETDNO

This command assigns the value d of the D number to a cutting edge (ce) of tool t. The result of this statement is returned via state (TRUE or FALSE). 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.

Syntax

```
d = GETDNO (t,ce)
state = SETDNO (t,ce,d)
```

Meaning

d:	D number of the tool edge
t:	T number of the tool
ce:	Cutting edge number (CE number) of the tool
state:	Indicates whether the command could be executed (TRUE or FALSE).

Example for renaming a D number

Programming	Comment
`$TC_DP2[1.2]=120`	
`$TC_DP3[1,2]  = 5.5`	
`$TC_DPCE[1,2] = 3`	`; Cutting edge number CE`
`...`	
`N10 def int DNoOld, DNoNew = 17`	
`N20 DNoOld = GETDNO(1,3)`	
`N30 SETDNO(1,3,DNoNew)`	

The new D value 17 is then assigned to cutting edge CE=3. Now the data for the cutting edge is addressed via D number 17; both via the system variables and in the programming with the NC address.

4.13.5.4 Free assignment of D numbers: Determine T number to the specified D number (GETACTTD)

The pre-defined function GETACTTD determines the T number associated with an absolute D number. There is no 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.

Syntax

```
<Status>=GETACTTD(<TNo>,<DNo>)
```

Meaning

GETACTTD():	Function call	
<DNo>:	D number for which the T number shall be searched.	
	Data type:	INT
<TNo>:	T number found	
	Data type:	VAR INT

`<status>`:	Result		
	Data type:	INT	
	Value:	0	The T number was found. <Tno> contains the value of the T number.
		-1	No T number exists for the specified D number; <Tno>=0.
		-2	The D number is not absolute. <TNo> receives the value of the first tool found that contains the D number with the value <Dno>.
		-5	The function was not able to be executed for another reason.

4.13.5.5 Free assignment of D numbers: Invalidate D numbers (DZERO)

The `DZERO` command is used for support during retooling. Compensation data sets tagged with this command are no longer verified by the `CHKDNO` command. These data sets can be accessed again by setting the D number once more with `SETDNO`.

Syntax

```
DZERO
```

Meaning

`DZERO`:	Marks all D numbers of the TO unit as invalid.

4.13.6 Toolholder kinematics

Requirements

A toolholder can only orientate a tool in all possible directions in space if

- Two rotary axes V_1 and V_2 are present.
- The rotary axes are mutually orthogonal.
- The tool longitudinal axis is perpendicular to the second rotary axis V_2.

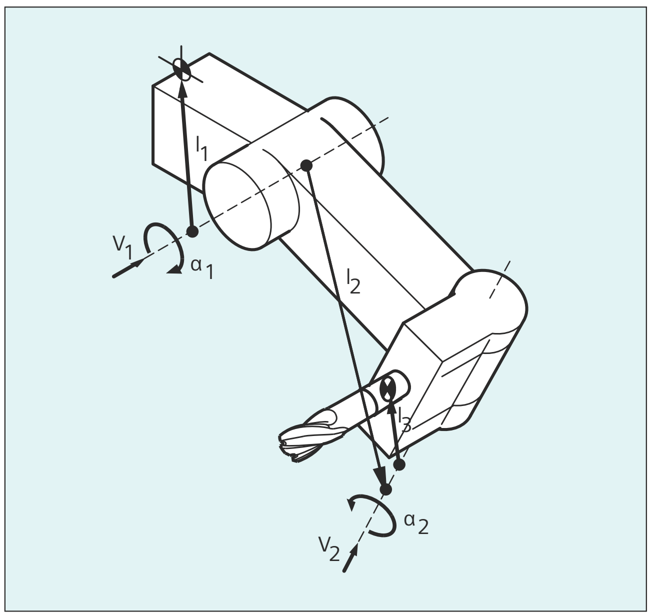
In addition, the following requirement is applicable to machines for which all possible orientations have to be settable:

- The tool longitudinal axis must be perpendicular to the first rotary axis V_1.

Function

The toolholder kinematics with a maximum of two rotary axes v_1 or v_2 are defined using the 17 system variables $TC_CARR1[m] to $TC_CARR17[m]. The description of the toolholder consists of:

- The vectoral distance from the first rotary axis of the toolholder I_1, the vectoral distance from the first rotary axis to the second rotary axis I_2, the vectoral distance from the second rotary axis to the reference point of the tool I_3.
- The direction vectors of both rotary axes V_1, V_2.
- The angles of rotation α1, α2 around the two axes. The rotation angles are counted in viewing direction of the rotary axis vectors, positive, in clockwise direction of rotation.



For machines with **resolved kinematics** (both the tool and the part can rotate), the system variables have been extended with the entries $TC_CARR18[m] to $TC_CARR23[m].

Parameters

Function of the system variables for orientable toolholders			
Designation	**x component**	**y component**	**z component**
l_1 offset vector	$TC_CARR1[m]	$TC_CARR2[m]	$TC_CARR3[m]
l_2 offset vector	$TC_CARR4[m]	$TC_CARR5[m]	$TC_CARR6[m]
v_1 rotary axis	$TC_CARR7[m]	$TC_CARR8[m]	$TC_CARR9[m]
v_2 rotary axis	$TC_CARR10[m]	$TC_CARR11[m]	$TC_CARR12[m]
α_1 angle of rotation α_2 angle of rotation	$TC_CARR13[m] $TC_CARR14[m]		
l_3 offset vector	$TC_CARR15[m]	$TC_CARR16[m]	$TC_CARR17[m]

Extensions of the system variables for orientable toolholders			
Designation	**x component**	**y component**	**z component**
l_4 offset vector	$TC_CARR18[m]	$TC_CARR19[m]	$TC_CARR20[m]
Axis identifier Rotary axis v_1 Rotary axis v_2	Axis identifier of the rotary axes v_1 and v_2 (initialized with zero) $TC_CARR21[m] $TC_CARR22[m]		
Kinematic type	$TC_CARR23[m]		
Tool **P**art **M**ixed mode	Kinematics type T ->	Kinematics type P ->	Kinematics type M
	Only the tool can rotate (default).	Only the part can rotate	Part and tool can rotate
Offset of the Rotary axis v_1 Rotary axis v_2	Angle in degrees of the rotary axes v_1 and v_2 on assuming the initial setting $TC_CARR24[m] $TC_CARR25[m]		
Angle offset of the rotary axis v_1 Rotary axis v_2	Offset of the Hirth tooth system in degrees for rotary axes v_1 and v_2 $TC_CARR26[m] $TC_CARR27[m]		
Angle increment v_1 rotary axis v_2 rotary axis	Offset of the Hirth tooth system in degrees for rotary axes v_1 and v_2 $TC_CARR28[m] $TC_CARR29[m]		
Min. position Rotary axis v_1 Rotary axis v_2	Software limit for the minimum position of the rotary axes v_1 and v_2 $TC_CARR30[m] $TC_CARR31[m]		
Max. position Rotary axis v_1 Rotary axis v_2	Software limits for the maximum position of the rotary axes v_1 and v_2 $TC_CARR32[m] $TC_CARR33[m]		
Toolholder name	A toolholder can be given a name instead of a number. $TC_CARR34[m]		
User: Axis name 1 Axis name 2 Identifier	Intended use in user measuring cycles $TC_CARR35[m] $TC_CARR36[m] $TC_CARR37[m]		
Position	$TC_CARR38[m]	$TC_CARR39[m]	$TC_CARR40[m]
Fine offset	Parameters that can be added **to the values in the basic parameters**.		
l_1 offset vector	$TC_CARR41[m]	$TC_CARR42[m]	$TC_CARR43[m]
l_2 offset vector	$TC_CARR44[m]	$TC_CARR45[m]	$TC_CARR46[m]
l_3 offset vector	$TC_CARR55[m]	$TC_CARR56[m]	$TC_CARR57[m]
l_4 offset vector	$TC_CARR58[m]	$TC_CARR59[m]	$TC_CARR60[m]
v_1 rotary axis	$TC_CARR64[m]		
v_2 rotary axis	$TC_CARR65[m]		

Note

Explanations of parameters

"m" specifies the number of the toolholder to be programmed.

$TC_CARR47 to $TC_CARR54 and $TC_CARR61 to $TC_CARR63 are not defined and produce an alarm if read or write access is attempted.

The start/end points of the distance vectors on the axes can be freely selected. The rotation angles α_1, α_2 around the two axes are defined in the initial state of the toolholder by 0°. In this way, the kinematics of a toolholder can be programmed for any number of possibilities.

Toolholders with only one or no rotary axis at all can be described by setting the direction vectors of one or both rotary axes to zero.
With a toolholder without rotary axis the distance vectors act as additional tool offsets whose components cannot be affected by a change of machining plane (G17 to G19).

Parameter extensions

Parameters of the rotary axes

The system variables have been extended by the entries $TC_CARR24[m] to $TC_CARR33[m] and described as follows:

Offset of rotary axes v_1, v_2	Changing the position of the rotary axis v_1 or v_2 for the initial setting of the oriented toolholder.
The **angle offset/ angle increment** of the rotary axes v_1, v_2	The offset or the angle increment of the Hirth tooth system of the rotary axes v_1 and v_2. Programmed or calculated angle is rounded up to the next value that results from phi = s + n * d when n is an integer.
The minimum and maximum position of the rotary axes v_1, v_2	The minimum and maximum position of the rotary axis limit angle (software limit) of the rotary axes v1 and v2.

Parameters for the user

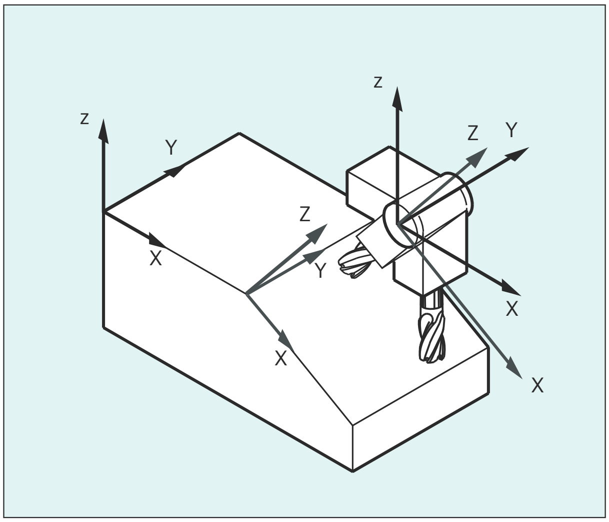
$TC_CARR34 to $TC_CARR40 contain parameters that are freely available to users and up to SW 6.4 were as standard, not further evaluated within the NCK or had no significance.

Fine offset parameters

$TC_CARR41 to $TC_CARR65 include fine offset parameters that can be added to the values in the basis parameters. The fine offset value assigned to a basic parameter is obtained when the value 40 is added to the parameter number.

Example

The toolholder used in the following example can be fully described by a rotation around the Y axis.



Program code	Comment
N10 $TC_CARR8[1]=1	; Definition of the Y component of the first rotary axis of toolholder 1.
N20 $TC_DP1[1,1] = 120	; Definition of a shaft miller.
N30 $TC_DP3[1,1]=20	; Definition of a shaft miller, 20 mm long.
N40 $TC_DP6[1,1]=5	; Definition of a shaft miller with 5 mm radius.
N50 ROT Y37	; Frame definition with 37° rotation around the Y axis.
N60 X0 Y0 Z0 F10000	; Approach start position.
N70 G42 CUT2DF TCOFR TCARR=1 T1 D1 X10	Set radius compensation, tool length compensation in rotated frame, select toolholder 1, tool 1.
N80 X40	; Perform machining under a rotation of 37°.
N90 Y40	
N100 X0	
N110 Y0	
N120 M30	

Further information

Resolved kinematics

For machines with resolved kinematics (both the tool as well as the workpiece can be rotated), the system variables have been expanded by the entries `$TC_CARR18[m]` up to `$TC_CARR23[m]` and are described as follows:

The rotatable tool table consisting of:

- The vectorial clearance of the second rotary axis V_2 to the reference point of a tool table that can be rotated I_4 of the third rotary axis.

The rotary axes consisting of:

- The two channel identifiers for the reference of the rotary axes $V_1$and V_2, whose position is, when required, accessed to determine the orientation of the toolholder that can be orientated.

The type of kinematics with one of the values T, P or M:

- Kinematics type T: Only tool can rotate.
- Kinematics type P: Only part can rotate.
- Kinematics type M: Tool and part can rotate.

Clearing the toolholder data

Data of all toolholder data sets can be deleted using `$TC_CARR1[0]=0`.

The kinematic type `$TC_CARR23[T]=T` must be assigned with one of the three permissible uppercase or lowercase letters (T,P,M) and for this reason, should not be deleted.

Changing the toolholder data

Each of the described values can be modified by assigning a new value in the part program. Any character other than T, P or M results in an alarm when an attempt is made to activate the toolholder that can be orientated.

Reading the toolholder data

Each of the described values can be read by assigning it to a variable in the part program.

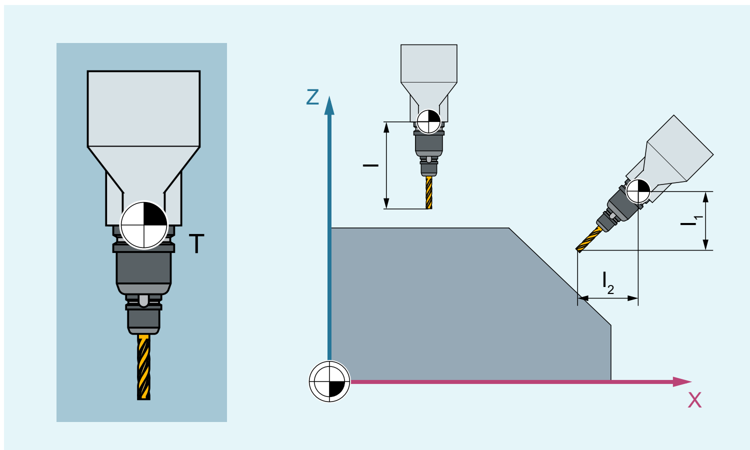
Fine offsets

An illegal fine offset value is only detected if a toolholder that can be orientated is activated, which contains such a value and at the same time setting data SD42974 $SC_TOCARR_FINE_CORRECTION = TRUE.

The maximum permissible fine offset is limited to a permissible value in the machine data.

4.13.7 Tool length compensation for orientable toolholders (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ)

As the spatial orientation of the tool carrier and thus of the tool changes, its tool length components also change:



T Tool carrier reference point

l_1, l_2 Tool length components

After a reset, e.g. through manual setting or change of the tool carrier with a fixed spatial orientation, the tool length components also have to be determined again. The commands of G group 42 "tool carrier" are used for this purpose.

Syntax

```
TCARR=<m>
TCOABS
TCOFR
TCOFRZ/TCOFRY/TCOFRX
```

Meaning

Element	Type	Meaning	
TCARR=<m>	Address	Request tool carrier	
		<m>	Number of the tool carrier
TCOABS	G command	Determine tool length components from the orientation of the current tool carrier	
TCOFR	G command	Determine tool length components from the orientation of the active frame	

Element	Type	Meaning	
TCOFRZ/TCOFRY/TCOFRX	G command	The TCOFRX/TCOFRY/TCOFRZ command assumes a tool oriented in the corresponding direction (X/Y/Z) and calculates the setting angles of the orientable tool carrier so that the tool in the active frame is oriented in the same direction.	
		TCOFRZ	The tool oriented in the Z direction is aligned so that it is also oriented in the Z direction in the active frame.
		TCOFRY	The tool oriented in the Y direction is aligned so that it is also oriented in the Y direction in the active frame.
		TCOFRX	The tool oriented in the X direction is aligned so that it is also oriented in the X direction in the active frame.

Further information

Tool carrier selection (TCARR)

A tool carrier data block is connected to the tool carrier which describes its geometry. If a tool is active at the time of tool carrier selection, it is assigned to the newly selected tool carrier and the geometry data of the tool carrier take effect immediately. It is not necessary to change or re-program the active tool. When a new tool is activated, it is treated as if it was mounted on the active tool carrier.

With TCARR=0, the active tool carrier is deselected.

Note

The current geometry data of the tool carrier can also be defined in the NC program via the corresponding system variables (see "Access to tool carrier data blocks").

Note

From the point of view of the control, tool carrier numbers <m> and tool numbers <n> can be freely combined. 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.

Access to tool carrier data blocks

The current geometry data of a tool carrier can be accessed from the NC program as follows:

- Write:
  $TC_CARR<n>[<m>]=<Value>
- Read:
  <Value>=$TC_CARR<n>[<m>]
  (<Value> must be a variable of data type REAL)

If the referenced tool carrier is not defined, an alarm is displayed.

Setting tool carrier data blocks to zero

All values of all tool carrier data blocks can be deleted from within the part program using one command:

```
$TC_CARR1[0] = 0
```

Individual tool carrier data blocks can be selectively deleted using the predefined procedure DELTC.

Effect of tool carrier selection

A tool carrier becomes effective when both a tool carrier and a tool have been activated. The selection of the tool carrier alone has no effect.

The effect of a tool carrier selection depends on the command from the G group 42 (tool carrier), which was programmed together with TCARR.

Changing the G command from this group causes the tool length components to be recalculated when a tool carrier is active.

Calculating the tool length compensation from the tool carrier orientation (TCOABS)

TCOABS calculates the tool length compensation from the current orientation angles of the tool carrier, stored in the system variables $TC_CARR13 and $TC_CARR14. The considered orientation is independent of the orientation of the active frame.

For the definition of the tool carrier kinematics with system variables, see Chapter "Tool carrier kinematics (Page 694)" in the NC Programming Manual.

Determining the tool length components from the orientation of the active frame (TCOFR)

With TCOFR, the orientation angles of the tool carrier are determined from the orientation of the active frame. The values stored in the tool carrier data are not changed, however. These are also used to resolve the ambiguity that can arise when calculating the rotation angles from a frame (see following paragraph).

Ambiguities

With two axes, a particular tool orientation defined by the frame can generally be set with **two** different rotation angle pairs. Of these two possible positions, the control selects the one in which the rotation angles are as close as possible to the programmed rotation angles.

Note

In cases where ambiguity may occur, it is usually necessary to store the angles to be expected from the frame in the tool carrier data.

Tool orientation in the active frame (TCOFRZ/TCOFRY/TCOFRX)

The TCOFRX/TCOFRY/TCOFRZ command assumes an orientation in the X/Y/Z direction for calculating the tool carrier's setting angles, regardless of the active orientation. The selection of the machining plane (G17/G18/G19) does not affect the orientation in the active frame. However, it has an influence on the calculation of the tool length compensation.

Frame change

The user can change the frame after selecting the tool. This does not have any effect on the tool length compensation components.

The rotation angles stored in the tool carrier data are not affected by the rotation angles defined with frames. When changing from TCOFR to TCOABS, the original (programmed) angles of rotation in the tool carrier data is reactivated.

Recalculation of the tool length compensation

To recalculate the tool length compensation after a frame change, the tool must be selected again. When the tool length compensation is calculated, the angle of rotation of the tool carrier is calculated in an intermediate step. Since tool carriers with two axes of rotation generally have two rotation angle pairs with which the tool orientation can be adapted to the active frame, the rotation angle values stored in the tool carrier data must at least approximately correspond to the mechanically set rotation angles.

Note

In virtually any case where ambiguities may arise, it is necessary to store the approximate angle expected from the frame in the tool carrier data.

Note

The tool orientation must be manually adapted to the active frame.

Note

The control system cannot check the angles of rotation calculated via the frame orientation for adjustability on the machine. If the machine is designed in such a way that the axes of rotation of the tool carrier cannot reach the tool orientation calculated by the frame orientation, an alarm is output.

Note

The combination of tool precision compensation and the functions for tool length offset on movable tool carriers is not permissible. If both functions are called simultaneously, an error message is issued.

Note

With TOFRAME (Page 329) it is possible to define a frame based on the orientation direction of the selected tool carrier.

Note

When orientation transformation is active (3, 4 or 5-axis transformation), it is possible to select a tool carrier with an orientation deviating from the zero position without causing output of an alarm.

Note

With an active frame without rotation, orientation with TCOFRX/TCOFRY/TCOFRZ leads to a trivial solution, since the tool orientation already points in the X/Y/Z direction.

Orientation in the Z direction (TOFRAME)

With the TOFRAME (Page 329) command of the G group 53 (tool-related frame rotation), it is possible to define a frame whose Z direction is aligned in parallel to the orientation direction of the selected tool carrier.

If no tool carrier or a tool carrier without orientation change is active, then the Z direction in the new frame is as follows:

- 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.

Transfer parameter from standard and measuring cycles

For the transfer parameter of standard and measuring cycles, the following defined value ranges apply.

For angle values, the following value ranges are defined:

Rotation around ...	Value range
1st geometry axis	-180° ≤ angle ≤ +180°
2nd geometry axis	-90° ≤ angle ≤ +90°
3rd geometry axis	-180° ≤ angle ≤ +180°

Note

When transferring angular values to a standard or measuring cycle, the following should be carefully observed:

Values smaller than the computational accuracy of the NC must be rounded to zero!

The calculation resolution of the NC for angular positions is defined in the machine data:

MD10210 $MN_INT_INCR_PER_DEG

4.13.8 Modifying the orientable tool carrier according to the machine measurement (CORRTC)

Measured kinematic chain elements of a tool carrier can be written to special correction elements with the `CORRTC` function.

Note

The correction values written with the CORRTC function are not immediately effective in the tool carrier. The correction values only become active after deselecting the tool carrier, NEWCONF and selecting the tool carrier.

Syntax

```
<_Corr_Status> = CORRTC(<_Corr_Vect>, <_Corr_Index>, <_Corr_Mode>,
[ <_No_Alarm>])
```

Meaning

`CORRTC`:	Function call		
`<_Corr_Status>`:	Function return value		
	Data type:	INT	
	Values:	0	The function was executed without an error.
		1	No tool carrier is active.
		2	The active tool carrier was not defined with kinematic chains.
		10	The `<_Corr_Index>` call parameter is negative.
		11	The `<_Corr_Mode>` call parameter is negative.
		12	Invalid reference to a section of a subchain (_CORR_INDEX).
		13	No correction element has been defined in the section referred to by the _CORR_INDEX parameter ($TC_CARR_CORR_ELEM).
		20	The hundreds position of <_CORR_MODE> is invalid. Only the values 0 and 1 are permissible.
		21	The thousands position of <_CORR_MODE> is invalid. Only the values 0 and 1 are permissible.
		30	For the correction of an offset vector, the deviation from the current value in at least one coordinate is greater than the maximum value specified by the setting data SD41612 $SN_CORR_TRAFO_LIN_MAX.
		31	The attempt to write a system variable was rejected because of missing write rights.
`<_Corr_Vect>`:	Correction vector The content of the correction vector is defined by the following parameters <_Corr_Index> and `<_Corr_Mode>`. If `<_Corr_Status> = 30`, the content of the vector is overwritten (see above).		
	Data type:	REAL	
`<_Corr_Index>`:	Designates the section for which the direction vector of the correction element is to be corrected.		
	Data type:	INT	

`<_Corr_Mode>:`	Correction mode		
	Data type:	INT	
	The `<Corr_Index>` parameter is decimal coded (unit to thousands position):		
	Unit position:	Reserved	
	Tens position:	Specifies how the correction element to which the content of `<_Corr_Index>` refers, is to be modified.	
		xx**0**x	The correction vector is written immediately to the correction element. This variant can be used to immediately write the correction element without the index <n> of the relevant system data ($NK_OFF_DIR[<n>, ...]) having to be known.
		xx**1**x	As 0, but with the difference that the transferred correction value is interpreted in world coordinates. A difference between variants 0 and 1 can always occur when the kinematic chain in the initial state (positions of all rotary axes equal to 0) contains other rotations.
		xx**2**x	As 1, but with the difference that the correction value refers to the entire section, i.e. a value is entered in the correction element so that the entire section reaches the length defined by the correction value.
	Hundreds position:	Specifies how the content of the `<_Corr_Vect>` parameter is to be interpreted.	
		x**0**xx	The transferred correction vector `<_Corr_Vect>` contains the entire new length of the correction element or the section to which the `<_Corr_Index>` in conjunction with the tens position of `<_Corr_Mode>` refers (absolute correction).
		x**1**xx	The transferred correction vector `<_Corr_Vect>` only contains the difference compared to the current length of the correction element or the section to which the `<_Corr_Index>` in conjunction with the tens position of `<_Corr_Mode>` refers (incremental correction).
	Thousands position:	Determines whether or not the maximum permissible correction is to be limited by the setting data $SN_ CORR_TOCARR_LIN_MAX.	
		0xxx	Threshold value monitoring is active.
		1xxx	The threshold value monitoring is suppressed.
`<_No_Alarm>:`	Response in the event of an error (return value > 0) (**optional**)		
	Data type:	BOOL	
	Value:	FALSE (default)	In the event of an error, the program execution is stopped and an alarm displayed.
		TRUE	In the event of an error, the program processing is not stopped and no alarm is displayed. Application: User-specific response corresponding to the return value

Further information about CORRTC

The kinematic structure of a tool carrier is described by one (type T and type P) or two (type M) kinematic chains (subchains), which start from the associated reference point, machine zero or tool carrier reference point). One of the two chains, the tool chain, ends at the reference point of the tool, the other chain, the workpiece chain ends in the zero point of the basic coordinate system.

The CORRTC function writes axis directions for machines with tool carriers in special correction elements. A kinematic chain is described, for example, with elements of the type OFFSET, which are defined via $ NK_TYPE.

CORRTC works with sections

The two subchains can each be divided into a maximum of four sections:

- Section 1 begins at the starting point of the chain and ends at the first rotary axis.
- Section 2 is the section between rotary axis 1 and rotary axis 2.
- Section 3 is the section between rotary axis 2 and the end of the chain.

The following figure shows an orientable tool carrier with 2 rotary axes.

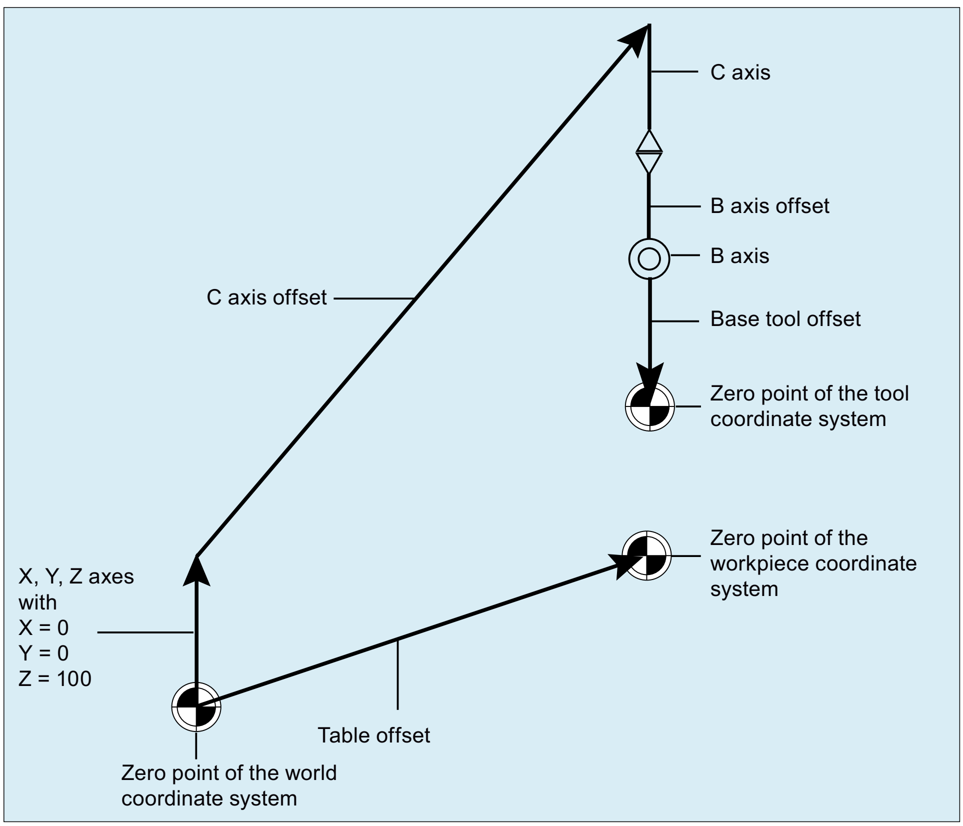


Figure 4-5 CORRTC example

The sections are clearly defined: If the kinematic subchain is executed from its starting point to its end point, then the first section has the index 1, next has index 2, and so on.

Correction elements

A reference can be made to a constant kinematic chain element (chain element of the type $NK_TYPE[<n>] = "OFFSET") in each of these sections with the $TC_CARR_CORR_ELEM [, 0 ... 3] system variables. The correction values determined during the machine measurement are written to the so designated elements with the CORRTC function.

The sequence of references in $TC_CARR_CORR_ELEM[m, 0 ... 3] must correspond to the sections described above, that is there can only be one chain element in $TC_CARR_CORR_ELEM[m, 0] which belongs to the offset vector **l**1, etc.

The reference value is always the corresponding value effective in the tool carrier active when CORRTC is called. After selection of the tool carrier, changed contents of the stored kinematic data have no effect on the method of operation of the CORRTC function.

4.13.9 Online tool length compensation (TOFFON, TOFFOF)

Use the system variable $AA_TOFF[<n>] to overlay the effective tool lengths in accordance with the three tool directions three-dimensionally in real time.

The three geometry axis identifiers are used as index <n>. Thus, the number of active direction offsets is determined by the geometry axes that are active at the same time.

All offsets can be active at the same time.

The online tool length compensation function can be used in combination with orientable tool carriers (TCARR).

Note

Online tool length compensation is an **option,** which must be enabled in advance. It is only useful in conjunction with an active orientable tool carrier.

Syntax

```
...
TOFFON(<CorrDir>[,<Offset>])
WHEN TRUE DO $AA_TOFF[<CorrDir>]                          ; In synchronized actions.
...
TOFFOF(<CorrDir>)
```

Meaning

TOFFON	**Activate** online tool length compensation	
	<CorrDir>	Tool direction (X, Y, Z), in which the online tool length compensation should be active.
	<Offset>	When activating, an offset value can be specified for the relevant direction of compensation and this is immediately recovered.
TOFFOF	**Reset** online tool length compensation The compensation values in the specified compensation direction are reset and a preprocessing stop is initiated.	

Example

Program code	Comment
...	
DEF REAL XOFFSET	
$MC_TOFF_MODE=1	; MD21190
$MC_TOFF_VELO[0]=10000	; MD21194
$MC_TOFF_VELO[1]=10000	
$MC_TOFF_VELO[2]=10000	
$MC_TOFF_ACCEL[0]=1	; MD21196
$MC_TOFF_ACCEL[1]=1	
$MC_TOFF_ACCEL[2]=1	
NEWCONF	
	; Deactivate tool carrier
	;CYCLE800()
TCARR=0	
$P_WPFR=CTRANS()	
$P_WPFRAME=$P_WPFR	
G0 X0 Y0 Z0 A0 C0	
M0	
	; Activate ToolCarrier
	;CY-CLE800(0,"HEAD",100000,57,0,0,0,30,0,0,0,0,0,-1,100,101)
G17	
$P_WPFR=CTRANS()	
$P_WPFR=CROT(X,30)	
$P_WPFRAME=$P_WPFR	
CUT2DF TCARR=2	
TCOFR	
PAROT	
G0 C=$P_TCANG[3] A=$P_TCANG[4]	
M0	
	; Activate tool length compensation in the Z direction
TOFFON(Z)	
	; Tool length compensation in the Z direction: 10 mm
WHEN TRUE DO $AA_TOFF[Z]=10	
M0	
G4 F5	
M0	
	; Deactivate tool length compensation in the Z direction
TOFFOF(Z)	

Program code	Comment
`M0`	
	`; Deactivate tool carrier`
	`;CYCLE800()`
`TCARR=0`	
`$P_WPFR=CTRANS()`	
`$P_WPFRAME=$P_WPFR`	
`M30`	

```
DEF REAL XOFFSET
$MC_TOFF_MODE = 1           ; MD21190
$MC_TOFF_VELO[0] = 10000    ; MD21194
$MC_TOFF_VELO[1] = 10000
$MC_TOFF_VELO[2] = 10000
$MC_TOFF_ACCEL[0] = 1       ; MD21196
$MC_TOFF_ACCEL[1] = 1
$MC_TOFF_ACCEL[2] = 1
NEWCONF
; Deactivate ToolCarrier
;CYCLE800()
TCARR=0
$P_WPFR = CTRANS()
$P_WPFRAME = $P_WPFR
G0 X0 Y0 Z0 A0 C0
M0
; Activate ToolCarrier
;CYCLE800(0,"HEAD",100000,57,0,0,0,30,0,0,0,0,0,-1,100,101)
G17
$P_WPFR = CTRANS()
$P_WPFR = CROT(X,30)
$P_WPFRAME = $P_WPFR
CUT2DF TCARR=2
TCOFR
PAROT
G0 C=$P_TCANG[3]  A=$P_TCANG[4]
M0
```

```
; Activate tool length compensation in the Z direction
TOFFON(Z)
; Tool length compensation in the Z direction: 10 mm
WHEN TRUE DO $AA_TOFF[Z] = 10
M0
G4 F5
M0
; Tool length compensation in the Z direction: 0 mm
WHEN TRUE DO $AA_TOFF[Z] = 0
G4 F5
M0
; Deactivate tool length compensation in the Z direction
TOFFOF(Z)
M0
; Deactivate ToolCarrier
;CYCLE800()
TCARR=0
$P_WPFR = CTRANS()
$P_WPFRAME = $P_WPFR
M30
```

PS: If it goes to the doc. The MDs have a wrong identifier. It should be 21194 $MC_TOFF_VEL**O** or 21196 $MC_TOFF_ACC**EL**.

h

More information

Block preparation

In the case of block preparation in forward motion, the tool length offset currently active in the main run is considered. In order to utilize the maximum permissible axis velocities as far as possible, it is necessary to halt the block preparation with a preprocessing stop (STOPRE) while a tool length offset is being generated.

The tool length offset is always known at the prerun time even if the tool length offsets are not changed after the program start. Or if, after a change in tool length offsets, more blocks have been processed than the IPO buffer can accommodate between the prerun and the main run. In this way, correct axis velocities are applied quickly for short blocks.

System variables

Information on online tool length compensation can be queried via the following system variables:

System variable	Meaning for online tool length compensation	
$AA_TOFF_VAL[<n>]	Integrated value of the overlaid compensation movement in the corresponding tool direction	
$AA_TOFF_LIMIT[<n>]	Status of the limitation by: • SD42970 $SC_TOFF_LIMIT (upper limit of online tool length compensation) • SD42972 $SC_TOFF_LIMIT_MINUS (lower limit of online tool length compensation)	
	= 0	Limit value not reached
	= 1	Limit value reached in positive axis direction
	= -1	Limit value reached in negative axis direction
$AA_TOFF_PREP_DIFF[<n>]	The difference between the currently active compensation in the interpolator and the compensation that was active at the time of block preparation.	

Tool change / orientation changes

When changing the tool or changing the tool direction, the active online tool length compensations $AA_TOFF[<n>] are retained.

Note

Changing the effective tool length using online tool length compensation produces changes in the compensatory movements of the axes in the event of changes in orientation. The resulting velocities can be higher or lower depending on machine kinematics and the current axis position.

TOFFOF

After deselecting online tool length compensation in the specified compensation direction, the accumulated tool length compensations are removed and transferred to the base coordinate system. The prerun is synchronized with the current position in main run. Since no axes are traversed here, the values of the system variable $AA_IM[<Ax>] (current MCS setpoint of an axis) do not change. Only the values of the system variables $AA_IW[<Ax>] (actual WCS setpoint of an axis) or $AA_IB[<Ax>] (actual BCS setpoint of an axis) are changed. These variables now contain the deselected share of tool length compensation.

4.13.10 Modification of the offset data for rotatable tools

4.13.10.1 Activating the modification of the offset data for rotatable tools (CUTMOD)

The modification of the offset data for rotatable tools is activated in the NC-program by the CUTMOD language command.

Syntax

```
CUTMOD = <Value>
```

Meaning

`CUTMOD`:	Activating the modification of the offset data for rotatable tools		
`<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.
		-3	The function is only activated if no orientable tool carrier is active. The values supplied from system variables $P_ADT... may be modified with respect to the corresponding tool parameters depending on the active rotation. The activation of the designated orientable tool carrier deactivates the function. The function is not influenced by a reset.
		< -3	Values less than 3 are ignored. I.e. this case is treated as if CUTMOD was not programmed. **Note:** This value range should not be used as it is reserved for possible subsequent expansions.

Note

SD42984 $SC_CUTDIRMOD

The CUTMOD 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.

More 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
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 has been activated with the `CUTMOD` command, and the tool was rotated by an orientable tool carrier.

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 is based on the modified cutting edge data determined with the CUTMOD function.
$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.

<table>
<tr><th>System variable</th><th colspan="2">Meaning</th></tr>
<tr><td>$P_CUT_INV /
$AC_CUT_INV</td><td colspan="2">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 $SC_TOOL_LENGTH_TYPE = 2).
2. The modification of the offset data was activated with the <code>CUTMOD</code> command.
3. An orientable tool carrier is active, which was selected with the CUTMOD command.
4. The tool is rotated by the orientable tool carrier so that the resulting normal of the tool cutting edge is rotated with respect to the initial position by more than 90° (typically 180°).
If at least one of the specified four conditions is not fulfilled, the variable returns the value FALSE. For tools whose cutting edge position is not defined, the value of the variable is always FALSE.</td></tr>
<tr><td rowspan="11">$P_CUTMOD_ERR</td><td colspan="2">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:</td></tr>
<tr><td>Bit 0:</td><td>No valid cut direction is defined for the active tool.</td></tr>
<tr><td>Bit 1:</td><td>The cutting edge angle (clearance angle and holder angle) of the active tool are both zero.</td></tr>
<tr><td>Bit 2:</td><td>The clearance angle of the active tool has an impermissible value (< 0° or > 180°).</td></tr>
<tr><td>Bit 3:</td><td>The holder angle of the active tool has an impermissible value (< 0° or > 90°).</td></tr>
<tr><td>Bit 4:</td><td>The plate angle of the active tool has an impermissible value (< 0° or > 90°).</td></tr>
<tr><td>Bit 5:</td><td>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°).</td></tr>
<tr><td>Bit 6:</td><td>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.</td></tr>
<tr><td>Bit 7:</td><td>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.</td></tr>
<tr><td>Bit 8:</td><td>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).</td></tr>
<tr><td>Bit 9:</td><td>Orientable tool carrier is active when CUTMOD = -3.</td></tr>
</table>

$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.

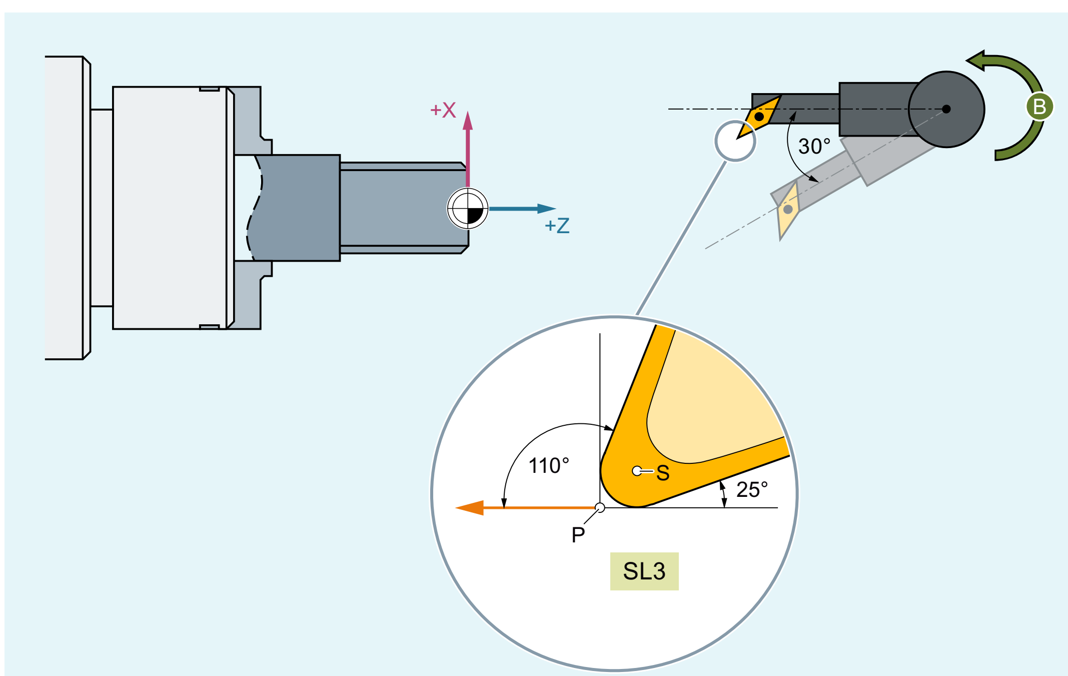
This priority rule of the setting data over the G code can be deactivated by setting bit 18 of the machine data $MC_TOOL_PARAMETER_DEF_MASK. This means that when this bit is set, the plane defined with the G command of group 6 is still valid.

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.

Example

For a tool with cutting edge position 3 and an orientable tool carrier that can rotate the tool around the B axis, the cutting edge position shall be modified after a tool rotation with the aid of the CUTMOD command.



S: Cutting edge center point

P: Cutting edge reference point

SL: Cutting edge position

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
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`.

4.13.11 Working with tool environments

Overview of functions

- Save tool environment (TOOLENV) (Page 719)
- Delete tool environment (DELTOOLENV) (Page 722)
- Read T, D and DL number (GETTENV) (Page 723)
- Read tool lengths and/or tool length components (GETTCOR) (Page 724)
- Change tool components (SETTCOR) (Page 730)

System variables overview

- Read information about the saved tool environments ($P_TOOLENVN, ($P_TOOLENV) (Page 724)

4.13.11.1 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

Further 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.

4.13.11.2 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

<table>
<tr><td colspan="2" rowspan="2"><code>DELTOOLENV(...):</code></td><td colspan="3">Predefined function to delete a tool environment</td></tr>
<tr><td colspan="2">Alone in the block:</td><td>Yes</td></tr>
<tr><td colspan="2" rowspan="5"><code><Status>:</code></td><td colspan="3">Function return value. Negative values indicate error states.</td></tr>
<tr><td colspan="2">Data type:</td><td>INT</td></tr>
<tr><td rowspan="3">Value:</td><td>0</td><td>Function OK</td></tr>
<tr><td>-1</td><td>No memory reserved for tool environments:
MD18116 $MN_MM_NUM_TOOL_ENV = 0
This means that the "tool environments" functionality is not available.</td></tr>
<tr><td>-2</td><td>A tool environment with the specified name does not exist.</td></tr>
<tr><td colspan="5">Parameters</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2"><code><name>:</code></td><td colspan="3">Name of data set to be deleted</td></tr>
<tr><td colspan="2">Data type:</td><td>STRING</td></tr>
<tr><td colspan="5"></td></tr>
<tr><td colspan="2"><code>DELTOOLENV():</code></td><td colspan="3"><code>DELTOOLENV()</code> deletes data sets describing tool environments without specifying a name</td></tr>
</table>

4.13.11.3 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.		

4.13.11.4 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:

<table>
<tr><td rowspan="6">$P_TOOLENVN:</td><td colspan="4">Supplies the number of data sets (which have still not been deleted) – defined using TOOLENV – to describe tool environments</td></tr>
<tr><td>Syntax:</td><td colspan="3"><n> = $P_TOOLENVN</td></tr>
<tr><td rowspan="3">Meaning:</td><td rowspan="3"><n>:</td><td colspan="2">Number of defined data sets</td></tr>
<tr><td>Data type:</td><td>INT</td></tr>
<tr><td>Value range:</td><td>0 ... MD18116 $MN_MM_NUM_TOOL_ENV</td></tr>
<tr><td colspan="4">This system variable can be accessed even if no tool environments are possible (MD18116 = 0). In this case, the return value is "0".</td></tr>
<tr><td rowspan="8">$P_TOOLENV:</td><td colspan="4">Supplies the name of the <i>th data set to describe a tool environment</td></tr>
<tr><td>Syntax:</td><td colspan="3"><Name> = $P_TOOLENV[<i>]</td></tr>
<tr><td rowspan="5">Meaning:</td><td rowspan="2"><name>:</td><td colspan="2">Name of the data set with number <i></td></tr>
<tr><td>Data type:</td><td>STRING</td></tr>
<tr><td rowspan="3"><i>:</td><td colspan="2">Number of the data set</td></tr>
<tr><td>Data type:</td><td>INT</td></tr>
<tr><td>Value range:</td><td>1 ... $P_TOOLENVN</td></tr>
<tr><td colspan="4">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_TOOLENV), then the following alarm is output:
Alarm 17020 "inadmissible array index 1"</td></tr>
</table>

4.13.11.5 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

<table>
<tr><td rowspan="2">GETTCOR(...):</td><td colspan="2">Predefined function to read tool lengths or to read tool length components</td></tr>
<tr><td>Alone in the block:</td><td>Yes</td></tr>
</table>

`<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:

Transformation 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.

4.13.11.6 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>.		
			Characters:	`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".		
			Characters:	`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.			

<table>
<tr><td rowspan="9">3</td><td rowspan="9"><code><CorComp></code>:</td><td colspan="3">Specifies the component(s) of the tool data sets that are to be described (optional).</td></tr>
<tr><td colspan="2">Data type:</td><td>INT</td></tr>
<tr><td rowspan="6">Value:</td><td><code>0</code></td><td>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.</td></tr>
<tr><td><code>1</code></td><td>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.</td></tr>
<tr><td><code>2</code></td><td>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.</td></tr>
<tr><td><code>3 - 5</code></td><td>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.</td></tr>
<tr><td><code>6</code></td><td>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.</td></tr>
<tr><td><code>7 – 11</code></td><td>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.</td></tr>
<tr><td colspan="3">If this parameter is not specified then its value is "0".</td></tr>
</table>

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:

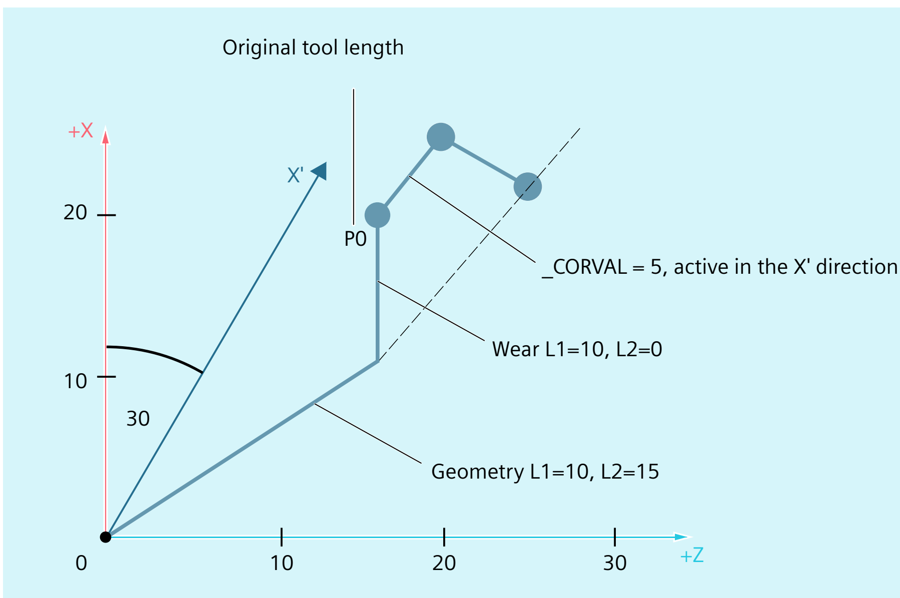
$TC_DP3[1,1]: 21.830 ; geometry L1

$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

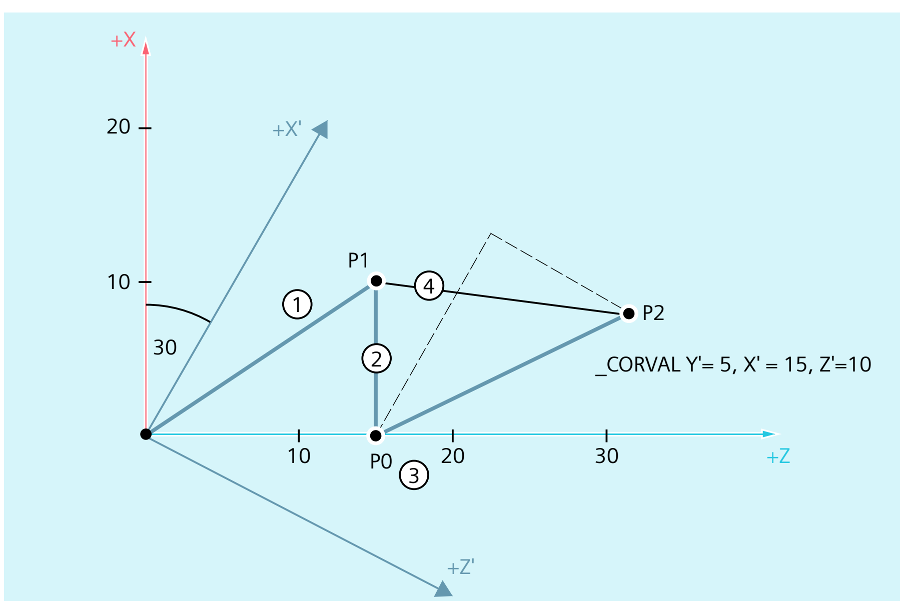
Program code	Comment
N10 DEF REAL _CORVAL[3]	
N20 $TC_DP1[1,1]=500	; Turning tool
N30 $TC_DP3[1,1]=10.0	; Geometry L1
N40 $TC_DP4[1,1]=15.0	; Geometry L2
N50 $TC_DP12[1,1]=10.0	; Wear L1
N60 $TC_DP13[1,1]=0.0	; Wear L2
N70 _CORVAL[0]=5.0	
N80 ROT Y-30	
N90 T1 D1 G18 G0	
N100 R1=**SETTCOR(_CORVAL,"GW",3,3)**	
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:



① Geometry L1=10, L2=15
② Wear L1=10, L2=0 (negative evaluation)
③ Original tool length
④ Resulting wear component

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
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.

4.13.12 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

<table>
<tr><td rowspan="2"><code>LENTOAX(...):</code></td><td colspan="2">Predefined function to read the assignment of tool lengths L1, L2 and L3 of the active tool to the coordinate axes</td></tr>
<tr><td>Alone in the block:</td><td>Yes</td></tr>
</table>

<table>
<tr><td colspan="2" rowspan="6"><Status>:</td><td colspan="3">Function return value. Negative values indicate error states.</td></tr>
<tr><td colspan="2">Data type:</td><td>INT</td></tr>
<tr><td rowspan="4">Value:</td><td>0</td><td>Function OK
Information provided in <AxInd> is sufficient for the description (all tool length components are in parallel to the geometry axes).</td></tr>
<tr><td>1</td><td>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).</td></tr>
<tr><td>-1</td><td>Invalid string in parameter <Coord>.</td></tr>
<tr><td>-2</td><td>No tool active.</td></tr>
<tr><td colspan="5">Parameters</td></tr>
<tr><td rowspan="5">1</td><td rowspan="5"><AxInd>:</td><td colspan="3">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</td></tr>
<tr><td colspan="2">Data type:</td><td>INT[3]</td></tr>
<tr><td rowspan="2">Value:</td><td>0</td><td>No assignment exists (axis does not exist)</td></tr>
<tr><td>1 ... 3
or
-1 ... -3</td><td>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.</td></tr>
<tr><td colspan="3">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>.</td></tr>
<tr><td rowspan="3">2</td><td rowspan="3"><Matrix>:</td><td colspan="3">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.</td></tr>
<tr><td colspan="2">Data type:</td><td>REAL</td></tr>
<tr><td colspan="3">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.</td></tr>
</table>

<table>
<tr><td rowspan="8">3</td><td rowspan="8"><Coord>:</td><td colspan="3">coordinate system applicable for the assignment (optional)</td></tr>
<tr><td colspan="2">Data type:</td><td>STRING</td></tr>
<tr><td rowspan="5">Charac-
ters:</td><td>MCS
M</td><td>The tool length is represented in the machine coordinate sys-
tem.</td></tr>
<tr><td>BCS
B</td><td>The tool length is represented in the basic coordinate system.</td></tr>
<tr><td>WCS
W</td><td>The tool length is represented in the workpiece coordinate system (default setting).</td></tr>
<tr><td>KCS
K</td><td>The tool length is represented in the tool coordinate system of the kinematic transformation.</td></tr>
<tr><td>TCS
T</td><td>The tool length is represented in the tool coordinate system.</td></tr>
<tr><td colspan="3">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).</td></tr>
</table>

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:

$$\text{<Matrix>} = \begin{pmatrix} 0 & \sin 60° & \cos 60° \\ 0 & \cos 60° & -\sin 60° \\ 1 & 0 & 0 \end{pmatrix}$$

4.14 Path traversing behavior

4.14.1 Feedrate characteristic (FNORM, FLIN, FCUB, FPO)

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 characteristics can be programmed either directly or as interpolating splines. This makes it possible to program continuously smooth speed characteristics, depending on the curvature of the workpiece to be machined.

These speed characteristics enable jerk-free acceleration changes and thus the production of uniform workpiece surfaces.

Syntax

```
F… FNORM
F… FLIN
F… FCUB
F=FPO(…,…,…)
```

Meaning

`FNORM`	Feedrate **normal** according to DIN 66025 (basic setting) The feedrate value is specified as a function of the traverse path of the block and is then valid as a modal value.
`FLIN`	Path velocity profile **linear** The feedrate value is approached linearly via the traverse path from the current value at the block beginning to the block end and is then valid as a modal value. The feedrate characteristic FLIN is active with G93, G94 and G95, **not** with G96/G961 and G97/G971. With G95 and changing spindle speed due to override or synchronized action, it is possible that the synchronization is slightly delayed.

`FCUB`	Path velocity profile **cubic** The F values programmed block-by-block are connected by a spline relative to the end of the block. The spline starts and ends at a tangent to the previous or subsequent feedrate function. If the F address is missing from a block, the last F value to be programmed is used. The feedrate characteristic FCUB is active with G93 and G94, **not** with G95, G96/G961 and G97/G971.
`F=FPO(...,...,...)`	Path velocity profile **via polynomial** The F address defines the feedrate characteristic via a polynomial from the current value to the block end. The end value is valid thereafter as a modal value.

The functions for programming the path traversing characteristics apply regardless of the programmed feedrate characteristic.

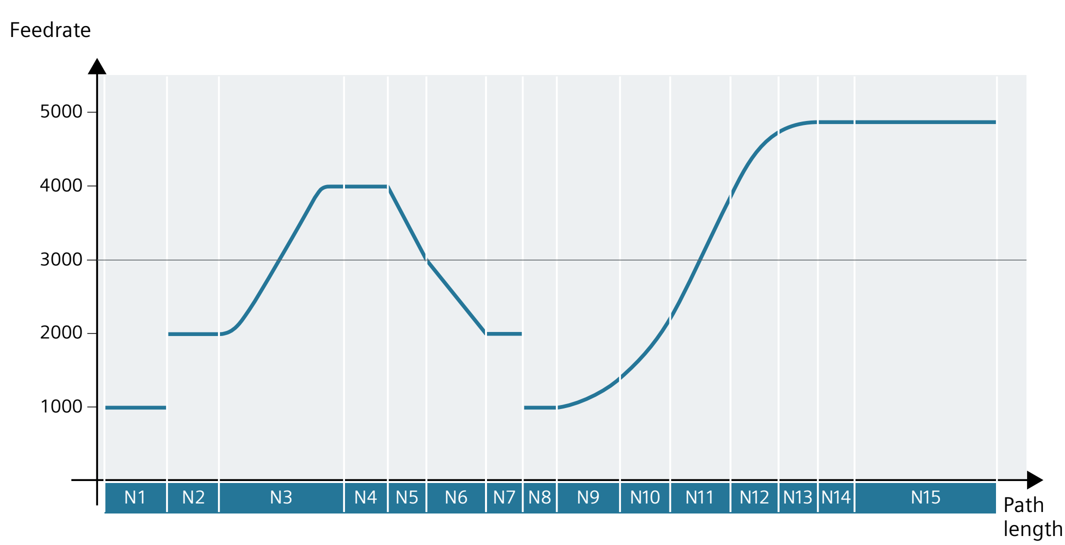
The programmable feedrate characteristic is always absolute, regardless of G90 or G91.

Note

Feedrate optimization on curved path sections

Feedrate polynomial F=FPO() and feedrate spline FCUB should always be traversed at constant feedrate on contour CFC. This enables the creation of a continuous-acceleration target feedrate profile.

Example: Various feed profiles



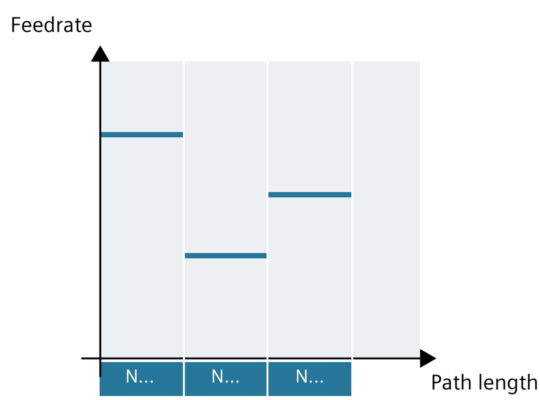
Program code	Comment
`N1 F1000 FNORM G1 X8 G91 G64`	`; Constant feedrate profile, incremental dimension data`
`N2 F2000 X7`	`; Setpoint velocity step change`
`N3 F=FPO(4000, 6000, -4000)`	`; Feed profile via polynomial with feedrate 4000 at the end of the block.`

Program code	Comment
N4 X6	; Polynomial feedrate 4000 is valid as modal value.
N5 F3000 FLIN X5	; Linear feed profile
N6 F2000 X8	; Linear feed profile
N7 X5	; Linear feedrate is valid as modal value
N8 F1000 FNORM X5	; Constant feedrate profile with acceleration step change.
N9 F1400 FCUB X8	; All of the following F values programmed in blocks are connected with splines.
N10 F2200 X6	
N11 F3900 X7	
N12 F4600 X7	
N13 F4900 X5	; Switch off spline profile.
N14 FNORM X5	
N15 X20	

More information

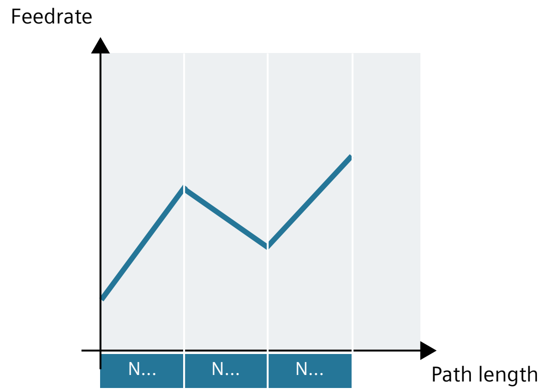
FNORM

The feedrate address F defines the path feedrate as a constant value according to DIN 66025.



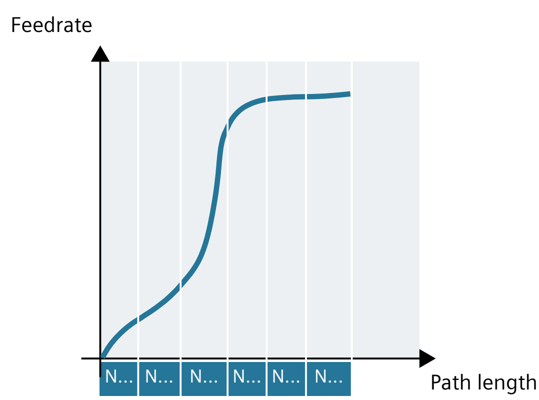
FLIN

The feedrate characteristic is approached linearly from the current feedrate value to the programmed F value until the end of the block.



FCUB

The feedrate is approached according to a cubic characteristic from the current feedrate value to the programmed F value until the end of the block. The control uses splines to connect all the feedrate values programmed non-modally that have an active FCUB. The feedrate values act here as interpolation points for calculation of the spline interpolation.



F=FPO(... , ... , ...)

The feedrate characteristic is programmed directly via a polynomial. The polynomial coefficients are specified according to the same method used for polynomial interpolation.

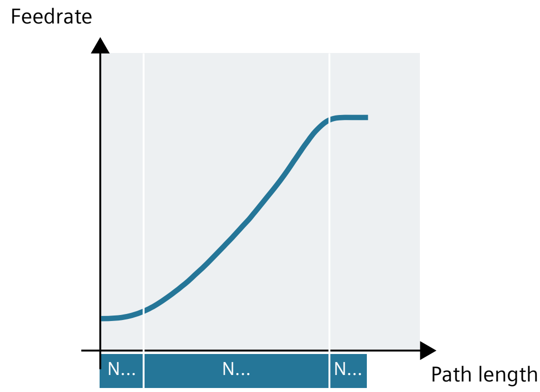
Example:

```
F=FPO(endfeed,quadf,cubf)
```

endfeed, quadf and cubf are previously defined variables.

endfeed	Feedrate at block end
quadf	Quadratic polynomial coefficient
cubf	Cubic polynomial coefficient

With an active FCUB, the spline is linked tangentially to the characteristic defined via FPO at the block beginning and block end.



Behavior with compressor function active

For active compressor function (COMP...) and combining several blocks to create a spline segment, the following applies:

FNORM:	The F word of the last block in the group applies to the spline segment.
FLIN:	The F word of the last block in the group applies to the spline segment. The programmed F value applies until the end of the segment and is then approached linearly.
FCUB:	The generated feedrate spline deviates from the programmed end points by a maximum of the value defined in the machine data MD20172 $MC_COMPRESS_VELO_TOL.
F=FPO(..., ..., ...):	These blocks are not compressed.

4.14.2 Acceleration behavior

4.14.2.1 Acceleration mode (BRISK, BRISKA, SOFT, SOFTA, DRIVE, DRIVEA)

The following part program commands are available for programming the current acceleration mode:

- "BRISK, BRISKA"
  The single axes or the path axes traverse with maximum acceleration until the programmed feedrate is reached (**acceleration without jerk limitation**).
- "SOFT, SOFTA"
  The single axes or the path axes traverse with constant acceleration until the programmed feedrate is reached (**acceleration with jerk limitation**).
- "DRIVE, DRIVEA"
  The single axes or the path axes traverse with maximum acceleration up to a programmed velocity limit (MD setting!). The acceleration rate is then reduced (MD setting) until the programmed feedrate is reached.

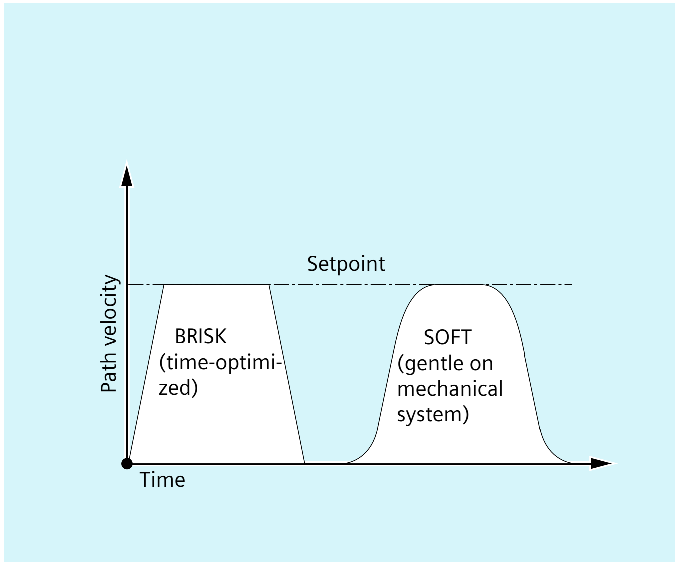


Figure 4-6 Path velocity curve with `BRISK` and `SOFT`

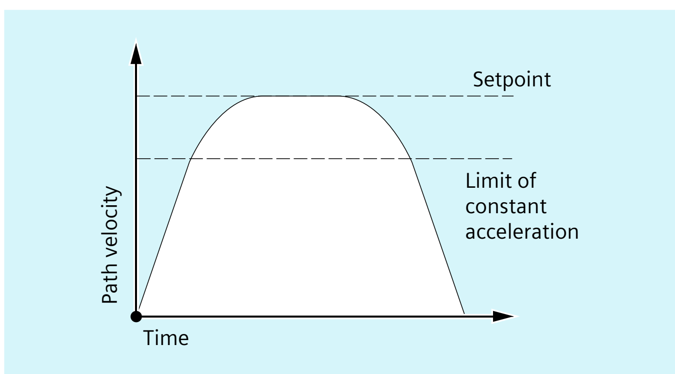


Figure 4-7 Path velocity curve with `DRIVE`

Syntax

```
BRISK
BRISKA(<axis1>,<axis2>,…)
SOFT
SOFTA(<axis1>,<axis2>,…)
DRIVE
DRIVEA(<axis1>,<axis2>,…)
```

Meaning

`BRISK:`	Command for activating the "acceleration without jerk limitation" for the path axes.
`BRISKA:`	Command for activating the "acceleration without jerk limitation" for single axis movements (JOG, JOG/INC, positioning axis, oscillating axis, etc.).

`SOFT:`	Command for activating the "acceleration with jerk limitation" for the path axes.
`SOFTA:`	Command for activating the "acceleration with jerk limitation" for single axis movements (JOG, JOG/INC, positioning axis, oscillating axis, etc.).
`DRIVE:`	Command for activating the reduced acceleration above a configured velocity limit (MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT) for the path axes.
`DRIVEA:`	Command for activating the reduced acceleration above a configured velocity limit (MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT) for single axis movements (JOG, JOG/INC, positioning axis, oscillating axis, etc.).
`(<axis1>,<axis2>, etc.):`	Single axes for which the called acceleration mode is to apply.

Supplementary conditions

Changing acceleration mode during machining

If the acceleration mode is changed in a part program during machining (`BRISK` ↔ `SOFT`), then there is a block change with exact stop at the end of the block during the transition even with continuous-path mode.

Examples

Example 1: SOFT and BRISKA

```
Program code
N10 G1 X… Y… F900 SOFT
N20 BRISKA(AX5,AX6)
...
```

Example 2: DRIVE and DRIVEA

```
Program code
N05 DRIVE
N10 G1 X… Y… F1000
N20 DRIVEA (AX4, AX6)
...
```

4.14.2.2 Influence of acceleration on following axes (VELOLIMA, ACCLIMA, JERKLIMA)

The dynamics limits of the following axes/spindles during coupled motion (Page 807) can be manipulated using the VELOLIMA, ACCLIMA, and JERKLIMA functions from the part program or from synchronized actions, even if the axis coupling is already active.

Note

The JERKLIMA function is not available for all types of coupling.

- ACCLIMA and VELOLIMA act on the total motion of the spindle during spindle operation, i.e. also in the uncoupled case. In the variables $AC_SMAXACC_INFO or $AC_SMAXVELO_INFO, a limitation is indicated by the identifier 22.
- In spindle operation, VELOLIMA is limited to 100 percent.
- The following applies for spindle couplings regarding the permissible speed:
  If the coupling motion already uses up the maximum speed and therefore the programmed basic motion stops, alarm 22015 "Following spindle has no dynamics for additional motion" is displayed.

Syntax

```
VELOLIMA(<axis>)=<value>
ACCLIMA(<axis>)=<value>
JERKLIMA(<axis>)=<value>
```

Meaning

`VELOLIMA`:	Command to correct the parameterized maximum **velocity**
`ACCLIMA`:	Command to correct the parameterized maximum **acceleration**
`JERKLIMA`:	Command to correct the parameterized maximum **jerk**
`<axis>`:	Following axis whose dynamics limits need to be corrected
`<Value>`:	Percentage correction value

Example

Correction of the dynamics limits for a following axis (AX4)

Program code	Comment
`...`	
`VELOLIMA[AX4]=75`	`; Limit correction to 75% of the maximum axial velocity stored in the machine data.`
`ACCLIMA[AX4]=50`	`; Limit correction to 50% of the maximum axial acceleration stored in the machine data.`
`JERKLIMA[AX4]=50`	`; Limit correction to 50% of the maximum axial jerk stored in the machine data.`
`...`	

4.14.2.3 Activation of technology-specific dynamic values (DYNNORM, DYNPOS, DYNROUGH, DYNSEMIFIN, DYNFINISH, DYNPREC)

The appropriate dynamic response for differing technological machining steps can be activated with the commands of G group 59 "Dynamic response mode for path interpolation".

Dynamic values and G commands can be configured and are, therefore, dependent on machine data settings.

Further information: Function Manual Basic Functions

Syntax

Activate dynamic values:

```
DYNNORM/DYNPOS/DYNROUGH/DYNSEMIFIN/DYNFINISH/DYNPREC
```

Note

The dynamic values are already active in the block in which the associated G command is programmed. Machining is not stopped.

Read or write a specific field element:

```
R<m>=$MA...[n,X]
$MA...[n,X]=<value>
```

Meaning

DYNNORM:	Activate **normal dynamic response**	
DYNPOS:	Activate dynamic response for **positioning mode, tapping**	
DYNROUGH:	Activate dynamic response for **roughing**	
DYNSEMIFIN:	Activate dynamic response for **semi-finishing**	
DYNFINISH:	Activate dynamic response for **finishing**	
DYNPREC:	Activate dynamic response for **smooth finishing**	
R<m>:	R-parameter with number <m>	
$MA...[n,X]:	Machine data with field element affecting dynamic response	
<n>:	Array index	
	Value range:	0 ... 5
	0	Normal dynamic response (DYNNORM)
	1	Dynamic response for positioning mode (DYNPOS)
	2	Dynamic response for roughing (DYNROUGH)
	3	Dynamic response for semi-finishing (DYNSEMIFIN)
	4	Dynamic response for finishing (DYNFINISH)
	5	Dynamic response for smooth finishing (DYNPREC)
<X>:	Axis address	
<Value>:	Dynamic value	

Examples

Example 1: Activate dynamic values

Program code	Comment
`DYNNORM G1 X10`	`; Initial setting`
`DYNPOS G1 X10 Y20 Z30 F…`	`; Positioning mode, tapping`
`DYNROUGH G1 X10 Y20 Z30 F10000`	`;Roughing`
`DYNSEMIFIN G1 X10 Y20 Z30 F2000`	`; Semi-finishing`

Program code	Comment
DYNFINISH G1 X10 Y20 Z30 F1000	;Finishing
DYNPREC G1 X10 Y20 Z30 F600	; Smooth finishing

Example 2: Read or write a specific field element

Maximum acceleration for roughing, axis X

Program code	Comment
R1=$MA_MAX_AX_ACCEL[2,X]	; reading
$MA_MAX_AX_ACCEL[2,X]=5	; writing

4.14.3 Traversing with feedforward control (FFWON, FFWOF)

The feedforward control reduces the velocity-dependent overtravel when contouring towards zero. Traversing with feedforward control permits higher path accuracy and thus improved machining results.

Syntax

```
FFWON
FFWOF
```

Meaning

FFWON:	Command to **activate** the feedforward control
FFWOF:	Command to **deactivate** the feedforward control

Note

The type of feedforward control and which path axes are to be traversed with feedforward control is specified via machine data.

Default: Velocity-dependent feedforward control

Option: Acceleration-dependent feedforward control

Example

Program code
N10 FFWON
N20 G1 X… Y… F900 SOFT

4.14.4 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.

<table>
<tr><td rowspan="11"><code>ATOL</code>:</td><td colspan="5">Address for programming an axis-specific tolerance</td></tr>
<tr><td>Applications:</td><td colspan="4">• All compressor functions
• ORISON orientation smoothing
• All smoothing types except G641, G644 and OSD</td></tr>
<tr><td>Preprocessing stop:</td><td colspan="4">No</td></tr>
<tr><td>Effective:</td><td colspan="4">Modal</td></tr>
<tr><td><code><Axis></code>:</td><td colspan="4">Name of the channel axis to which the programmed tolerance will apply</td></tr>
<tr><td rowspan="6"><code><Value></code>:</td><td colspan="4">The value for the axis tolerance will be specified as a length or an angle dependent on the axis type (linear or rotary axis).</td></tr>
<tr><td>Type:</td><td colspan="3">REAL</td></tr>
<tr><td rowspan="2">Unit:</td><td colspan="2">For linear axes:</td><td>inch/mm (dependent on the current dimensions setting)</td></tr>
<tr><td colspan="2">For rotary axes:</td><td>degrees</td></tr>
<tr><td rowspan="2">Value range:</td><td>≥ 0:</td><td colspan="2">Tolerance value</td></tr>
<tr><td>< 0:</td><td colspan="2">The programmed tolerance value is deleted
⇒ The tolerance value parameterized in the machine or setting data becomes effective again.</td></tr>
</table>

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.

Program code	Comment
`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...`	

More information

System variables

The currently effective tolerances can be read via the following system variables:

- Reading with preprocessing stop (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 (only 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.

Constraints

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.

4.14.5 Switch programmable contour accuracy on/off (CPRECON, CPRECOF)

The "Programmable contour accuracy" function reduces the path error on curved contours through automatic adaptation of the velocity.

It is switched on or off in the NC program with the modally effective commands of G group 39 (programmable contour accuracy).

Syntax

```
CPRECON
...
CPRECOF
```

Meaning

`CPRECON`:	Switch "Programmable contour accuracy" function **on**
`CPRECOF`:	Switch "Programmable contour accuracy" function **off**

Example

Program code	Comment
N10 G0 X0 Y0	
N20 CPRECON	; Activate the "programmable contour accuracy".
N30 G1 G64 X100 F10000	; Machining with 10 m/min in the continuous-path mode.
N40 G3 Y20 J10	; Automatic feed limitation in circular block.
N50 G1 X0	; Feedrate again without limitation (10 m/min).
...	
N100 CPRECOF	; Deactivate the "programmable contour accuracy".
N110 G0 ...	

See also

Programming contour/orientation tolerance (CTOL, OTOL, ATOL) (Page 755)

More information

Contour accuracy

The contour accuracy to be maintained is specified depending on the configuration of the machine (MD20470 $MC_MC_CPREC_WITH_FFW; see the machine manufacturer's information) either via the setting date SD42450$SC_CONTPREC or via the programmed contour tolerance CTOL. The smaller the value and the smaller the K_V factor of the geometry axes, the greater the path feedrate is reduced on curved contours.

Contour accuracy for rapid traverse movements

When machining workpieces with curved contours and the "Programmable contour accuracy" function is active, the velocity is reduced to maintain the specified contour accuracy even during tool movements at rapid traverse, e.g. in corners and corner rounding blocks when bypassing the workpiece. In order to minimize the reduction of the path velocity during rapid traverse movements, a rapid traverse contour accuracy deviating from the workpiece machining can be set for the "Programmable contour accuracy" function:

SD42451 $SC_CONTPREC_G00_ABS (contour accuracy with rapid traverse)

If SD42451 = 0, the contour accuracy set in $SC_CONTPREC[DYNNORM] applies to rapid traverse movements.

Minimum path feedrate

The user can use the following setting data to specify a minimum path feedrate for the "Programmable contour accuracy" function:

SD42460 $SC_MINFEED (minimum path feedrate for CPRECON)

The feedrate will not limited below this value, unless a lower F value has been programmed or the dynamic limitations of the axes force a lower path velocity.

No influence on positioning axes

The "Programmable contour accuracy" function only considers the geometry axes of the path. It does not have any effect of the velocities for the positioning axes.

Behavior at part program start and after channel or program end reset

At part-program start and after channel or program end reset, the initial control setting defined for the G function group 39 becomes effective (see the machine manufacturer's information).

4.14.6 Activating/deactivating automatic filter switching (AFISON, AFISOF)

With the "Automatic filter switchover" function, the user has the option of marking areas within the NC program in which all axes enabled for this function are automatically switched over to the 2nd filter chain for G0 motions. If the 2nd filter chain is parameterized so that oscillations are strongly damped, a greater jerk can be set for G0 motions. As a result, the path velocity is slowed down less during G0 motions, e.g. at corners, and the program runtime is shortened.

Syntax

The voice commands for switching the function on and off must each be alone in a block.

```
AFISON
...
AFISOF
```

Meaning

AFISON	Switch **on** the "Automatic filter switchover" function
AFISOF	Switch **off** the "Automatic filter switchover" function

Example

```
Program code              Comment
...
N390 G1 X1100             ; Filter chain 1 effective
                          ; Stop at X=1100
N400 AFISON               ; Automatic switchover to filter chain 2
N410 G0 X1150             ; Stop at X=1150
N420 G1 X1200             ; Automatic switchover to filter chain 1
N430 AFISOF
N440 G0 X1300             ; Filter chain 1 also effective in the G0 block
N450 G1 X1400
N460 AFISON
N470 G1 X1450
N480 G1 X1500
```

Program code	Comment
N490 **AFISOF**	
N500 G1 X1600	
N510 G0 X1700	; Filter chain 1 also effective in the G0 block
N520 **AFISON**	
N530 G1 X1750	; Stop at X=1750
	; Automatic switchover to filter chain 2
N540 G0 X1800	; Stop at X=1800
	; Automatic switchover to filter chain 1
N550 **AFISOF**	
N560 G1 X1900	
N570 G0 X2000	; Filter chain 1 also effective in the G0 block
	; Stop at X=2100
N580 **AFISON**	; Automatic switchover to filter chain 2
N590 G0 X2050	
N600 G0 X2100	; Stop at X=2100
	; Automatic switchover to filter chain 1
N610 **AFISOF**	
N620 G0 X2200	; Filter chain 1 also effective in the G0 block
...	

More information

Prerequisites

To be able to use the "Automatic filter switching" function, the following requirements must be met:

- The "Jerk adaptation" option, which requires a license, must be set.
  MD19321 $ON_TECHNO_FUNCTION_MASK_1, bit 22 = 1
- The function must be enabled for the channel:
  MD20630 $MC_AFIS_MODE = 1
- The function must be enabled for each axis intended for automatic filter switching:
  MD32332 $MD_AFIS_ENABLE = 1
- In all axes enabled for the function:
  - The jerk limitation must be active:
    MD32400 $MA_AX_JERK_ENABLE = 1
  - Two jerk filter types must be selected (→ MD32402 $MA_AX_JERK_MODE) and set.

If the requirements are not met, alarm 14782 or 26380 is output.

Stopping when switching on/off

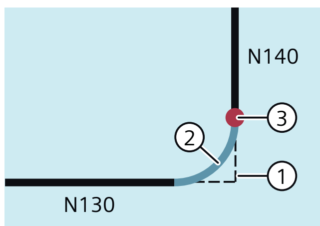
When the function is switched on and off, the path motion only stops if the position setpoint filter chain has to be switched.

Smoothing

If a stop and an automatic switchover of the filter chain takes place during active smoothing (G64x), this has no influence on the smoothing contour.

Example:

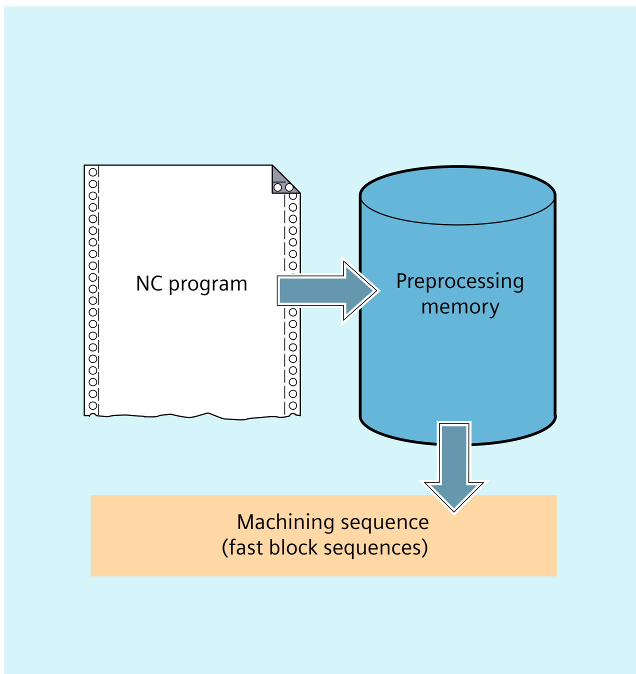
```
N100 G642 F1000 CTOL=10 CTOLG0=10
N110 G0 X0 Y0
N120 AFISON
N130 G1 X100 Y0
N140 G0 X100 Y100
```



① Programmed path
② Smoothing contour
③ Stop and switch to the 2nd filter chain

4.14.7 Program sequence with preprocessing memory (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE)

Depending on its expansion level, the control system has a certain quantity of so-called preprocessing memory in which prepared blocks are stored prior to program execution and then output as high-speed block sequences while machining is in progress. These sequences allow short paths to be traversed at a high velocity. Provided that there is sufficient residual control time available, the preprocessing memory is always filled.



Designate machining step

The beginning and end of the machining step to be buffered in the preprocessing memory are identified in the part program with "STOPFIFO" and "STARTFIFO" respectively. The processing of the preprocessed and buffered blocks starts only after the "STARTFIFO" command or if the preprocessing memory is full.

Automatic preprocessing memory control

Automatic preprocessing memory control is called with the "FIFOCTRL" command. "FIFOCTRL" acts initially just like "STOPFIFO". Whatever the programming, processing will not start until the preprocessing memory is full. However, the response to the emptying of the preprocessing memory does differ: With "FIFOCTRL", the path velocity is reduced increasingly once the fill level reaches 2/3 in order to prevent complete emptying and deceleration to standstill.

Preprocessing stop

Programming the "STOPRE" command in a block will stop block preprocessing and buffering. The following block is not executed until all preprocessed and saved blocks have been executed in full. The preceding block is halted in exact stop (as with G9).

NOTICE
Program abort
If tool offset or spline interpolations are active, a "STOPRE" command should not be programmed, as this will lead to contiguous block sequences being interrupted.

Syntax

Table 4-2 Identify machining step:

```
STOPFIFO
...
STARTFIFO
```

Table 4-3 Automatic preprocessing memory control:

```
...
FIFOCTRL
...
```

Table 4-4 Preprocessing stop:

```
...
STOPRE
...
```

Note

The "STOPFIFO", "STARTFIFO", "FIFOCTRL" and "STOPRE" commands must be programmed in their own block.

Meaning

`STOPFIFO`:	"STOPFIFO" identifies the start of a machining step to be buffered in the preprocessing memory. "STOPFIFO" stops processing and fills the preprocessing memory until: • "STARTFIFO" or "STOPRE" is recognized or • The preprocessing memory is full or • The end of the program is reached
`STARTFIFO`:	"STARTFIFO" starts rapid processing of the machining step; the preprocessing memory is filled in parallel to this.
`FIFOCTRL`:	Activation of automatic preprocessing memory control
`STOPRE`:	Stop preprocessing

Note

The preprocessing memory is not filled or filling is interrupted if the machining step contains commands that require unbuffered operation (search for reference, measuring functions, etc.).

Note

The control generates an internal preprocessing stop in the event of access to status data ($SA...).

Example: Stop preprocessing

Program code	Comment
...	
N30 MEAW=1 G1 F1000 X100 Y100 Z50	; Measurement block with probe at first measuring input and linear interpolation.
N40 STOPRE	; Preprocessing stop.
...	

4.14.8 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

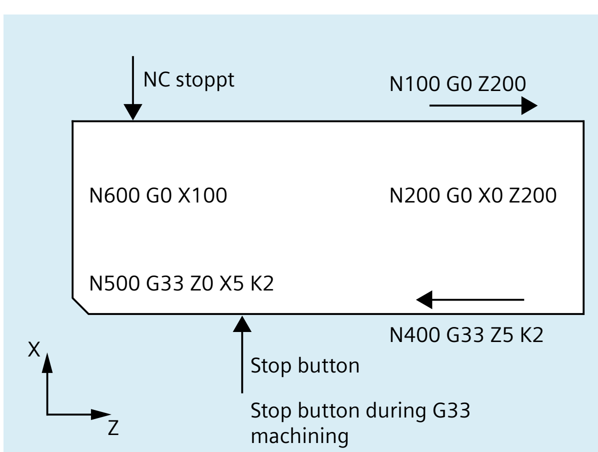
DELAYFSTON:	Defining the start of a stop delay range	
	Alone in the block:	Yes
DELAYFSTOF:	Define the end of the stop delay area	
	Alone in the block:	Yes

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

4.14.9 Prevent program position for SERUPRO (IPTRLOCK, IPTRUNLOCK)

For some complicated mechanical situations on the machine it is necessary to the stop block search SERUPRO.

By using a programmable interruption pointer it is possible to intervene before an untraceable point with "Search at point of interruption".

It is also possible to define untraceable sections in part program sections that the NC cannot yet re-enter. When the program is interrupted, the NC notes the last block that was processed that can then be searched for via the HMI user interface.

Syntax

```
IPTRLOCK
IPTRUNLOCK
```

The commands are located in a part program line and allow a programmable interruption pointer

Meaning

`IPTRLOCK`:	Start of untraceable program section
`IPTRUNLOCK`:	End of untraceable program section

Both commands are only permitted in part programs, but **not** in synchronous actions.

Example

Nesting of untraceable program sections in two program levels with implicit "IPTRUNLOCK". Implicit "IPTRUNLOCK" in subprogram 1 ends the untraceable section.

Program code	Comment
`N10010  IPTRLOCK()`	
`N10020  R1 = R1 + 1`	
`N10030  G4 F1`	`; Hold block of the search-suppressed program section starts.`
`...`	
`N10040 subprogram2`	
`...`	`; Interpretation of subprogram 2.`
`N20010  IPTRLOCK ()`	`; Ineffective, repeated start.`
`...`	
`N20020  IPTRUNLOCK ()`	`; Ineffective, end at another level.`
`N20030  RET`	
`...`	
`N10060  R2 = R2 + 2`	
`N10070  RET`	`; End of search-suppressed program section.`
`N100 G4 F2`	`; Main program is continued.`

The interruption pointer then produces an interruption at 100 again.

Further information

Acquiring and finding untraceable sections

Untraceable program sections are identified with language commands "IPTRLOCK" and "IPTRUNLOCK".

Command "IPTRLOCK" freezes the interruption pointer at a single block executable in the main run (SB1). This block will be referred to as the hold block below. If the program is aborted after "IPTRLOCK", this hold block can be searched for from the HMI user interface.

Continuing from the current block

The interruption pointer is placed on the current block with "IPTRUNLOCK" as the interruption point for the following program section.

Once the search target is found a new search target can be repeated with the hold block.

An interrupt pointer edited by the user must be removed again via the HMI.

Rules for nesting

The following points govern the interaction between language commands "IPTRLOCK" and "IPTRUNLOCK" with nesting and the subprogram end:

1. "IPTRLOCK" is activated implicitly at the end of the subprogram in which "IPTRUNLOCK" is called.
2. "IPTRLOCK" in an untraceable section has no effect.
3. If subprogram 1 calls subprogram 2 in an untraceable section, the whole of subprogram 2 remains untraceable. "IPTRUNLOCK" in particular has no effect in subprogram 2.

Further information: Function Manual Basic Functions

System variable

An untraceable section can be detected in the part program with "$P_IPTRLOCK".

Automatic interrupt pointer

The automatic interrupt pointer automatically defines a previously defined coupling type as untraceable. Using machine data, for the

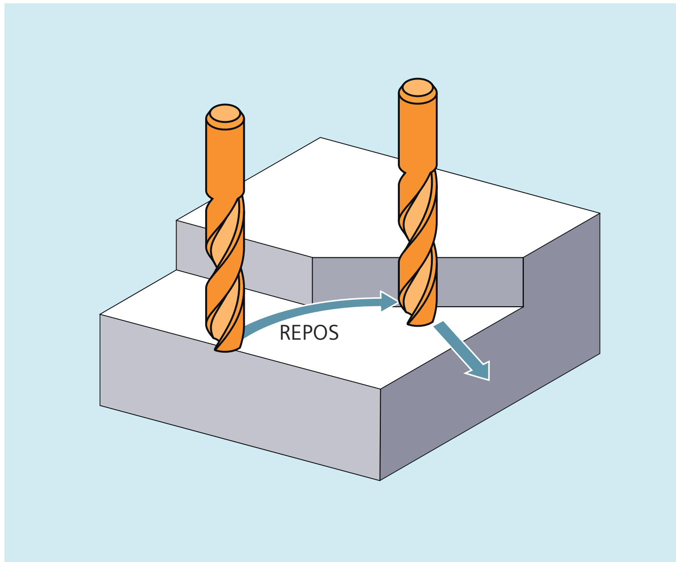
- Electronic gear for "EGON"
- Axial master value coupling for "LEADON"

the automatic interrupt pointer is activated. If the programmed interrupt pointer and the automatic interrupt pointer that can be activated via machine data overlap, then the largest possible untraceable section will be generated.

4.14.10 Repositioning to the contour (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMIBL, RMBBL, RMEBL, RMNBL)

If you interrupt the program run and retract the tool during the machining operation – because, for example, the tool has broken or you wish to measure the workpiece – you can reposition at any selected point on the contour under control by the program.

The command `REPOS` acts in an ASUB as a subprogram return (e.g. M17). The following blocks are not executed. For information on interrupting program runs, see also "Interrupt routine (ASUB) (Page 529)."



Syntax

```
REPOSA RMIBL DISPR=…
REPOSA RMBBL
REPOSA RMEBL
REPOSA RMNBL
REPOSL RMIBL DISPR=…
REPOSL RMBBL
REPOSL RMEBL
REPOSL RMNBL
REPOSQ RMIBL DISPR=… DISR=…
REPOSQ RMBBL DISR=…
REPOSQ RMEBL DISR=…
REPOSQA DISR=…
REPOSH RMIBL DISPR=… DISR=…
REPOSH RMBBL DISR=…
REPOSH RMEBL DISR=…
REPOSHA DISR=…
```

Meaning

Selecting the approach path

`REPOSA:`	Repositioning to the contour with the geometry axes along a straight line. All other channel axes are also repositioned.
`REPOSL:`	Repositioning to the contour with the geometry axes along a straight line. Other axes have to be programmed explicitly.
`REPOSQ DISR=… :`	Repositioning to the contour with the geometry axes along a quadrant of radius DISR. Other axes have to be programmed explicitly.

REPOSQA DISR=… :	Repositioning to the contour with the geometry axes along a quadrant of radius DISR. All other channel axes are also repositioned.
REPOSH DISR=… :	Repositioning to the contour with the geometry axes along a semicircle of diameter DISR. Other axes have to be programmed explicitly.
REPOSHA DISR=… :	Repositioning to the contour with the geometry axes along a semi-circle of radius DISR. All other channel axes are also repositioned.

Selecting the repositioning point

RMIBL:	Approach interruption point
RMIBL DISPR=…:	Entry point at distance DISPR in mm/inch in front of interruption point
RMBBL:	Approach block start point
RMEBL:	Approach end of block
RMEBL DISPR=… :	Approach block end point at distance DISPR in front of end point
RMNBL:	Approach at nearest path point
A0 B0 C0 :	Axes in which approach is to be made

Note

Compatibility

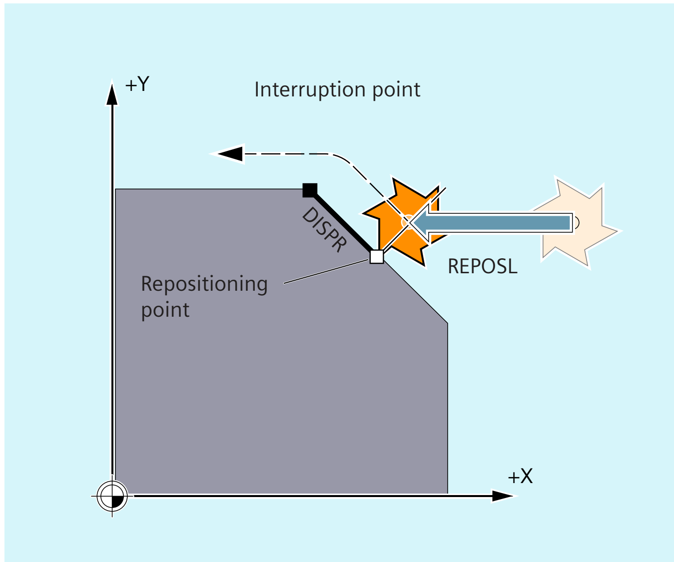
To remain compatible with older software versions, you can still program the `REPOS` approach mode via the modal commands `RMI`, `RMB`, `RME` and `RMN`. When used within an ASUB, this should be allocated the attribute `SAVE` in the `PROC` statement. Otherwise the modal `REPOS` approach mode used in the ASUB will take effect in subsequent `REPOS` processes, too, if it deviates from the preset `RMI`.

Repositioning to the contour along a straight line, REPOSA, REPOSL

The tool approaches the repositioning point along a straight line.

Example

```
REPOSL RMIBL DISPR=6 F400
```

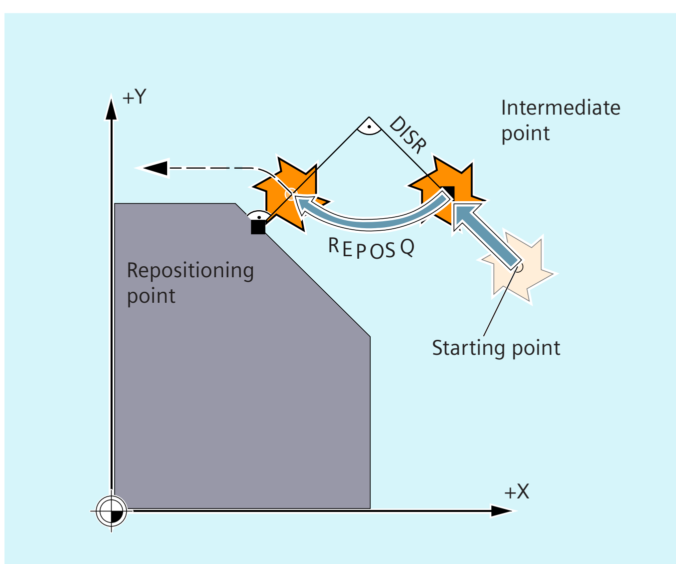


Repositioning to the contour along a quadrant, REPOSQ, REPOSQA

The tool approaches the repositioning point along a quadrant with a radius of `DISR=...`. The control automatically calculates the necessary intermediate point between the start and repositioning point.

Example

```
REPOSQ RMIBL DISR=10 F400
```

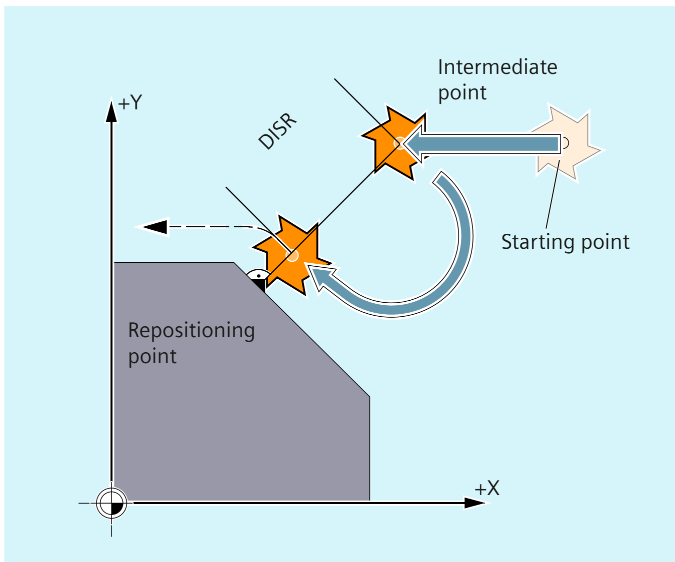


Repositioning to the contour along a semicircle, REPOSH, REPOSHA

The tool approaches the repositioning point along a semi-circle with a diameter of `DISR=...`. The control automatically calculates the necessary intermediate point between the start and repositioning point.

Example

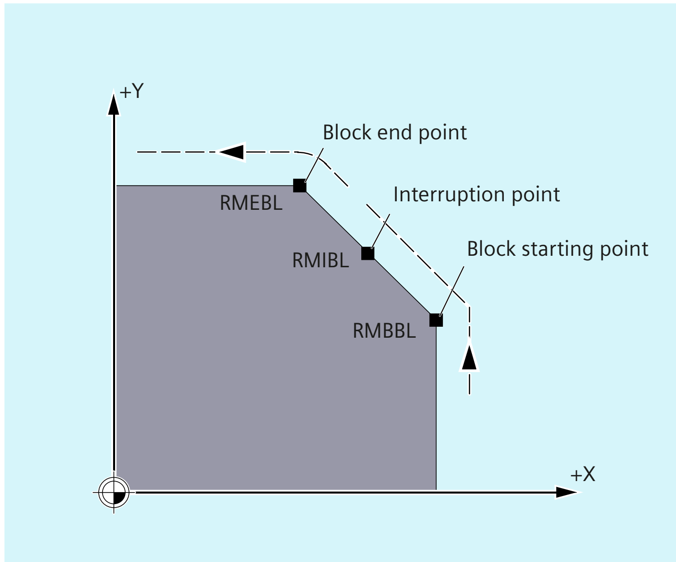
```
REPOSH RMIBL DISR=20 F400
```



Specifying the repositioning point (not for SERUPRO approaching with RMNBL)

With reference to the NC block in which the program run has been interrupted, it is possible to select one of three different repositioning points:

- RMIBL, interruption point
- RMBBL, block start point or last end point
- RMEBL, block end point



RMIBL DISPR=... or RME DISPR=... allows you to select a repositioning point which lies before the interruption point or the block end point.

DISPR=... allows you to describe the contour distance in mm/inch between the repositioning point and the interruption before the end point. Even for high values, this point cannot be further away than the block start point.

If no DISPR=... command is programmed, then DISPR=0 applies and with it the interruption point (with RMIBL) or the block end point (with RMEBL).

DISPR sign

The sign of DISPR is evaluated. In the case of a plus sign, the behavior is as previously.

In the case of a minus sign, approach is behind the interruption point or, with RMBBL, behind the block start point.

The distance between interruption point and approach point depends on the value of DISPR. Even for higher values, this point can lie in the block end point at the maximum.

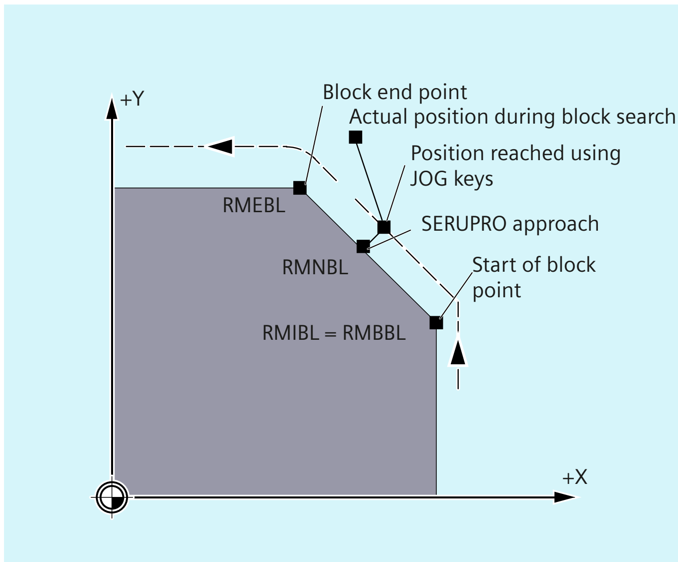
Application example:

A sensor will recognize the approach to a clamp. An ASUB is initiated to bypass the clamp.

Afterwards, a negative DISPR is repositioned on one point behind the clamp and the program is continued.

SERUPRO approach with RMNBL

If an abort is forced during machining at any position, the shortest path from the abort point is approached with SERUPRO approach and RMNBL so that afterward only the distance-to-go is processed. The user starts a SERUPRO process at the interruption block and uses the JOG keys to move in front of the problem component of the target block.



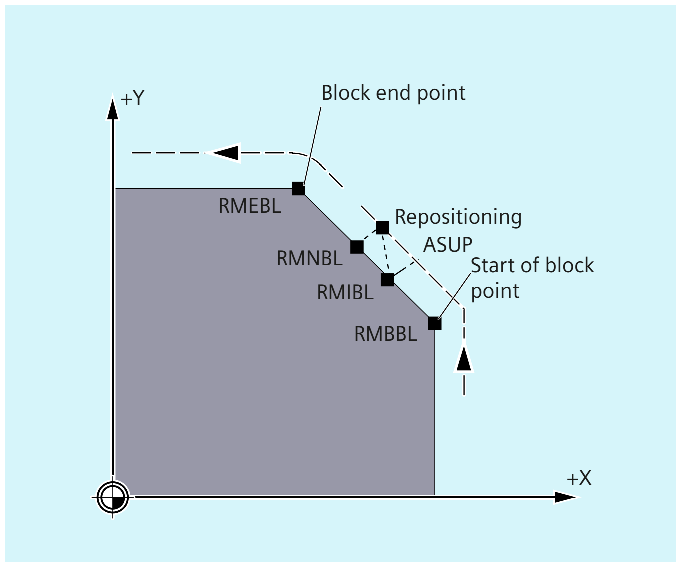
Note

SERUPRO

For SERUPRO, RMIBL and RMBBL are identical. RMNBL is not only limited to SERUPRO, but is generally valid.

Approach from the nearest path point RMNBL

When REPOSA is interpreted, the repositioning block with RMNBL is not started again in full after an interruption, but only the distance-to-go processed. The nearest path point of the interrupted block is approached.



Status for the valid REPOS mode

The valid REPOS mode of the interrupted block can be read with synchronized actions and variable $AC_ REPOS_PATH_MODE:

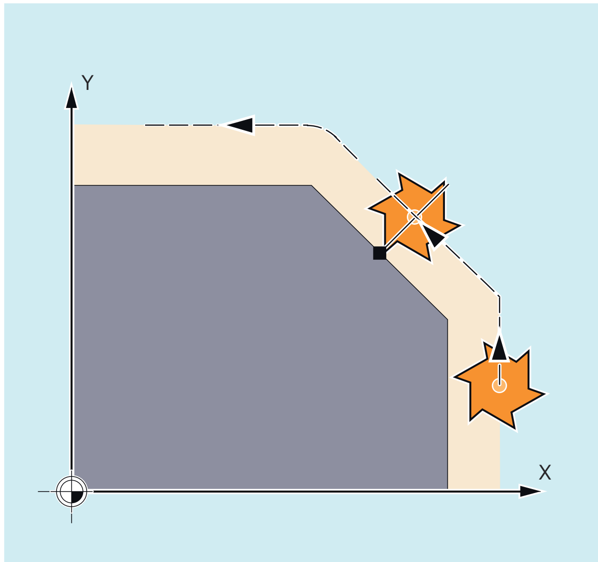
0 Approach not defined
1 RMBBL: Approach to beginning
2 RMIBL: Approach to point of interruption
3 RMEBL: Approach to end of block
4 RMNBL: Approach to next path point of the interrupted block

Approaching with a new tool

The following applies if you have stopped the program run due to tool breakage:

When the new D number is programmed, the machining program is continued with modified tool offset values at the repositioning point.

Where tool offset values have been modified, it may not be possible to reapproach the interruption point. In such cases, the point closest to the interruption point on the new contour is approached (possibly modified by DISPR).



Approach contour

The motion with which the tool is repositioned on the contour can be programmed. Enter zero for the addresses of the axes to be traversed.

The REPOSA, REPOSQA and REPOSHA commands automatically reposition all axes. Individual axis names need not be specified.

When the commands REPOSL, REPOSQ and REPOSH are programmed, all geometry axes are traversed automatically, i.e. they do not have to be specified in the command. All other axes must be specified in the commands.

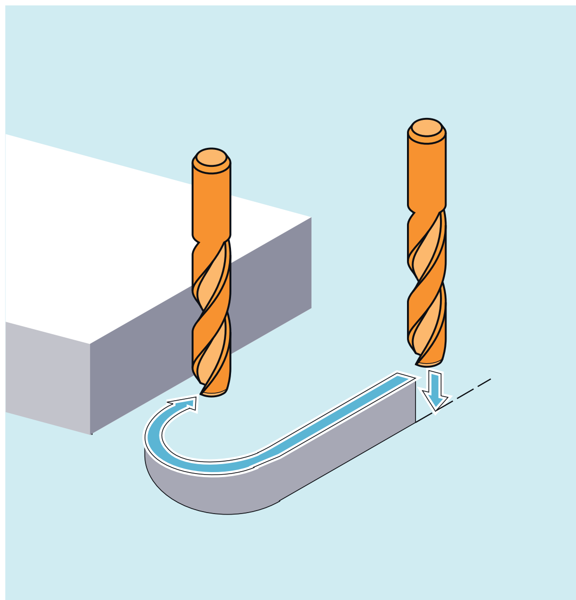
The following applies to the REPOSH and REPOSQ circular motions:

The circle is traversed in the specified working planes G17 to G19.

If you specify the third geometry axis (infeed direction) in the approach block, the repositioning point is approached along a helix in case the tool position and programmed position in the infeed direction do not coincide.

In the following cases, there is an automatic switchover to linear approach REPOSL:

- You have not specified a value for DISR.
- No defined approach direction is available (program interruption in a block without travel information).
- With an approach direction that is perpendicular to the current working plane.



4.14.11 Influencing the motion control

4.14.11.1 Adapting the maximum axis velocity or spindle speed (VELOLIM)

In the part program, the maximum possible velocity of an axis or the maximum possible gear-stage-dependent speed of a spindle set via machine data can be reduced using the VELOLIM command.

Effectiveness

VELOLIM acts:

- In the AUTOMATIC operating modes
- On path and positioning axes.
- On spindles in spindle/axis operations

Syntax

```
VELOLIM[<Ax>]=<Value>
```

Meaning

<table>
<tr><td>VELOLIM:</td><td colspan="2">Adapting the velocity or speed limit value</td></tr>
<tr><td><Ax>:</td><td colspan="2">Axis or spindle whose velocity or speed limit value should be adapted.
Using MD30455 $MA_MISC_FUNCTION_MASK, bit 6, it can be set as to whether VELOLIM is effective independent of whether used as spindle or axis (bit 6 = 1) - or is able to be programmed separately for each operating mode (bit 6 = 0). If they are to be separately effective, then the selection is made using the identifier when programming:
• Spindle identifier for spindle operating modes: S<n>
• Axis identifier for the axis mode, e.g. index: C</td></tr>
<tr><td rowspan="3"><Value>:</td><td colspan="2">Percentage correction value
For axes or spindles in the axis mode with setting MD30455 bit 6 = 0 the correction value refers to the configured maximum axis velocity (MD32000 $MA_MAX_AX_VELO).
For spindles in the spindle or axis mode and setting MD30455 bit 6 = 1, the correction value refers to the maximum speed of the active gear stage (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]).</td></tr>
<tr><td>Value range:</td><td>1 ... 100</td></tr>
<tr><td colspan="2">The value 100 does not influence the velocity or speed.</td></tr>
</table>

Note

Response at the end of the part program and for a channel reset

The response of VELOLIM at the end of the part program and for a channel reset depends on the setting of bit 0 in machine data MD32320 $MA_DYN_LIMIT_RESET_MASK:

Note

Speed limiting in spindle operation

Speed limiting using "VELOLIM" (less than 100 %) can be detected in spindle operation via the following system variables:

- $AC_SMAXVELO (maximum possible spindle speed)
- $AC_SMAXVELO_INFO (identifier for the speed-limiting cause)

Examples

Example 1: Velocity limitation, machine axis

Program code	Comment
...	
N70 VELOLIM[X]=80	; The axis slide in the X direction should only be traversed with a maximum of 80% of the velocity permissible for the axis.
...	

Example 2: Speed limiting, spindle 1 (AX5)

Configuring:

- MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[1, AX5] = 1000
  (Maximum speed of gear unit stage 1 = 1000 rpm)
- MD30455 $MA_MISC_FUNCTION_MASK[AX5], bit 6 = 1
  (Programming VELOLIM acts together for spindle and axis operation independent of the programmed identifier)

Programming:

Program code	Comment
`N05 VELOLIM[S1]=90`	`; Limiting the maximum speed of spindle 1 to 90% of 1000 rpm.`
`...`	
`N50 VELOLIM[C]=45`	`; Limiting the maximum speed of spindle 1 to 45% of 1000 rpm, C is the axis identifier of S1.`
`...`	

4.14.11.2 Adapting maximum axis jerk (JERKLIM)

In the part program, using command JERKLIM, the maximum jerk of an axis for path motion - set using machine data - can be reduced or increased in critical program sections.

Requirement

The acceleration mode SOFT must be active.

Effectiveness

JERKLIM acts:

- In the AUTOMATIC operating modes
- Only on path axes

Syntax

```
JERKLIM[<Ax>]=<Value>
```

Meaning

`JERKLIM`:	Adapting the jerk limit value	
`<Ax>`:	Machine axis whose jerk limit value is to be adapted	
`<Value>`:	Percentage correction value The correction value refers to the configured maximum axis jerk for path motion (MD32431 $MA_MAX_AX_JERK).	
	Value range:	1 ... 200
	Value 100 does not influence the jerk.	

Note

Response at the end of the part program and for a channel reset

The response of JERKLIM at the end of the part program and for a channel reset depends on the setting of bit 0 in machine data MD32320 $MA_DYN_LIMIT_RESET_MASK:

Example

Program code	Comment
N1000 G0 X0 Y0 F10000 SOFT G64	
N1100 G1 X20 RNDM=5 ACC[X]=20 ACC[Y]=30	
N1200 G1 Y20 JERKLIM[Y]=200	; The axis slide in the Y direction can be accelerated/decelerated with max. 200% of the jerk permissible for the axis.
N1300 G1 X0 JERKLIM[X]=2	; The axis slide in the X direction should only be accelerated/decelerated with max. 2% of the jerk permissible for the axis.
N1400 G1 Y0	
M30	

4.14.11.3 Adapting the maximum path velocity (FLIM)

With the function "Adapt maximum path velocity", the velocity of the path motion resulting from the axial limitation values can be limited or reduced in critical program sections in the part program. The velocity value to which the maximum path velocity is to be limited is programmed via the address FLIM. The programmed value is always effective only until the next NC reset or the end of the part program.

The override also affects the feedrate limited by FLIM. FLIM can therefore be moderately exceeded using the override switch.

Effectiveness

The function is effective:

- In the AUTOMATIC operating modes
- Only on path axes
- Only for G94
- Not with rapid traverse

Note

The function "Adjust maximum path velocity" is only effective in combination with linear feedrate G94. The function cannot be used in combination with other feed types (G93, G931, G95, G96, G97, G971, G972).

Syntax

```
...
FLIM=<Value>
...
FLIM=-1
...
```

Meaning

`FLIM`	Address for adjusting the maximum path velocity	
`<Value>`	Speed value to which the maximum path velocity should be limited	
	Data type:	REAL
	Value range:	$1.0 * 10^{-6}$ ... $1.0 * 10^{38}$
	Unit:	mm/min or inch/min (depending on the active system of units)
`FLIM=-1`	Cancels the limit programmed by `FLIM=<Value>`.	

Example

Program code	**Comment**
`...`	
`N1000 G0 X0 Y0 F10000 G64 G710`	
`N1100 G1 X20 RNDM=5`	
`N1200 G1 Y20` **`FLIM=5000`**	`; The axis slide in Y direction is moved at max. 5 m/min.`
`N1300 G1 X0 Y40`	`; The path movement of the axis slides in X direction and Y direction are moved at max. 5 m/min.`
`N1400 G1 Y0` **`FLIM=-1`**	`; The axis slide in Y direction is moved at 10 m/min.`
`M30`	

4.14.11.4 Adapting maximum path acceleration (PACCLIM)

With the function "Adapt maximum path acceleration", the acceleration of the path motion resulting from the axial limitation values can be reduced in critical program sections in the part program. The acceleration value to which the maximum path acceleration is to be reduced is programmed via the address PACCLIM. The programmed value is always effective only until the next NC reset or the end of the part program.

Requirements

Licensing

A license is required for this optional function ("Path acceleration limitation", article number: 6FC5800-0xP26-0YB0) and must be assigned to the hardware via the license management.

Effectiveness

The function is effective:

- In the AUTOMATIC operating modes
- Only on path axes

Syntax

```
...
PACCLIM=<Value>
...
PACCLIM=-1
...
```

Meaning

<table>
<tr><td>PACCLIM</td><td colspan="2">Address for adjusting the maximum path acceleration</td></tr>
<tr><td rowspan="4"><Value></td><td colspan="2">Acceleration value to which the maximum path acceleration is to be reduced</td></tr>
<tr><td>Data type:</td><td>REAL</td></tr>
<tr><td>Value range:</td><td>$1.0 * 10^{-6}$ ... $1.0 * 10^{38}$</td></tr>
<tr><td>Unit:</td><td>m/s^2 or $inch/s^2$ (depending on the active system of units)</td></tr>
<tr><td>PACCLIM=-1</td><td colspan="2">Cancels the limit programmed by PACCLIM=<Value>.</td></tr>
</table>

Note

The effect of PACCLIM is similar to the effect of the setting data SD42500 $SC_SD_MAX_PATH_ACCEL (maximum path acceleration). In contrast to SD42500, however, PACCLIM works block-synchronously.

Note

When calculating the limitation value, the value of SD42500 $SC_SD_MAX_PATH_ACCEL is only taken into account if SD42502 $SC_IS_SD_MAX_PATH_ACCEL is set to "TRUE".

If both PACCLIM and SD42500 $SC_SD_MAX_PATH_ACCEL are active, the smaller of the two limitation values becomes effective.

Example

Program code	Comment
...	
N1000 G0 X0 Y0 F10000 G64 G710	
N1100 G1 X20 RNDM=5	
N1200 G1 Y20 **PACCLIM=0.5**	; The axis slide in Y direction is accelerated/decelerated at max. 0.5 m/ss.

Program code	Comment
N1300 G1 X0 Y40	; The path movement of the axis slides in X direction and Y direction is accelerated/decelerated at max. 0.5 m/ss.
N1400 G1 Y0	
M30	

4.14.12 Block change behavior with active coupling (CPBC)

The CPBC command specifies the block change criterion that must be satisfied so that a block change can be executed in the part program with active coupling (Page 807).

Syntax

```
CPBC[<following axis>] = <criterion>
```

Meaning

`CPBC`:	Block change criterion with active coupling	
`<following axis>`:	Axis identifier of the following axis	
`<criterion>`:	Block change criterion	
	Type:	STRING
	Value	**Meaning:** Block change is performed
	"NOC"	Irrespective of the coupling status
	"IPOSTOP"	For setpoint synchronism
	"COARSE"	For actual value synchronism "coarse"
	"FINE"	For actual value synchronism "fine"

Example

```
; Block change takes place with:
;  - Coupling to following axis X2 == active
;  - Setpoint synchronism == active
CPBC[X2]="IPOSTOP"
```

4.15 Axis functions

4.15.1 Axis replacement, spindle replacement (RELEASE, GET, GETD)

One or more axes or spindles can only ever be interpolated in one channel. If an axis has to alternate between two different channels (e.g. pallet changer) it must first be enabled in the current channel and then transferred to the other channel. Axis replacement is effective between channels.

Axis replacement extensions

An axis/spindle can be replaced either with a preprocessing stop and synchronization between preprocessing and main run, or without a preprocessing stop. An axis interchange is also possible via:

- Frame with rotation if this process links the axis with other axes.
- Synchronized actions, see Motion-synchronous actions, "Axis replacement `RELEASE`, `GET`".

Machine manufacturer

Please refer to the machine manufacturer's instructions. For the purpose of axis replacement, one axis must be defined uniquely in all channels in the configurable machine data and the axis replacement characteristics can also be set using machine data.

Syntax

`RELEASE (axis name, axis name, ...)` or `RELEASE (S1)`

`GET (axis name, axis name, ...)` or `GET (S2)`

`GETD (axis name, axis name, etc.)` or `GETD(S3)`

With GETD (GET Directly), an axis is fetched directly from another channel. This means that no suitable RELEASE must be programmed for this GETD in another channel. It also means that other channel communication has to be established (e.g. wait markers).

Meaning

`RELEASE (axis name, axis name, etc.):`	Release the axis (axes)
`GET (axis name, axis name, etc.):`	Accept the axis (axes)
`GETD (axis name, axis name, etc.):`	Directly accept the axis (axes)
`Axis name:`	Axis assignment in the system: AX1, AX2, ... or specify machine axis name
`RELEASE(S1):`	Release spindles S1, S2, ...
`GET(S2):`	Accept spindles S1, S2, ...
`GETD(S3):`	Direct acceptance of spindles S1, S2, ...

GET request without preprocessing stop

If, following a GET request **without** preprocessing stop, the axis is enabled again with `RELEASE(axis)` or `WAITP(axis)`, a subsequent `GET` will induce a `GET` **with** preprocessing stop.

> ⚠ **CAUTION**
>
> **Axis assignment changed**
>
> An axis or spindle accepted with GET remains assigned to this channel even after a key or program RESET.
>
> When a program is restarted the replaced axes or spindles must be reassigned in the program if the axis is required in its original channel.
>
> It is assigned to the channel defined in the machine data on POWER ON.

Examples

Example 1: Axis exchange between two channels

Of the six axes, the following are used for machining in channel 1: 1st, 2nd, 3rd and 4th axis. 5th and 6th axis is used in channel 2 for the workpiece change.

Axis 2 should be exchanged between two channels and after POWER ON can be assigned to channel 1.

Program "MAIN" in channel 1:

Program code	Comment
`INIT (2,"TRANSFER2")`	`; Select program TRANSFER2 in channel 2.`
`N… START (2)`	`; Start the program in channel 2.`
`N… GET (AX2)`	`; Accept axis AX2.`
`...`	
`N… RELEASE (AX2)`	`; Release axis AX2.`
`N… WAITM (1,1,2)`	`; Wait for WAIT marker in channel 1 and 2 for synchronizing in both channels.`
`...`	`; Rest of program after axis replacement.`
`N… M30`	

Program "TRANSFER2" in channel 2:

Programming	Comment
`N… RELEASE (AX2)`	
`N160 WAITM(1,1,2)`	`; Wait for WAIT marker in channel 1 and 2 for synchronizing in both channels.`
`N150 GET(AX2)`	`; Accept axis AX2.`
`...`	`; Rest of program after axis replacement.`
`N… M30`	

Example 2: Axis exchange without synchronization

If the axis does not have to be synchronized no preprocessing stop is generated by GET.

Programming	Comment
N01 G0 X0	
N02 RELEASE(AX5)	
N03 G64 X10	
N04 X20	
N05 GET(AX5)	; If synchronization is not required, then this is not a block that can be executed.
N06 G01 F5000	; Block that cannot be executed.
N07 X20	; Block that cannot be executed, because X position as in N04.
N08 X30	; First block that can be executed after N05.
...	

Example 3: Activating an axis exchange without a preprocessing stop

Requirement: Axis replacement without a preprocessing stop must be configured via machine data.

Programming	Comment
N010 M4 S100	
N011 G4 F2	
N020 M5	
N021 SPOS=0	
N022 POS[B]=1	
N023 WAITP(B)	; Axis B becomes the neutral axis.
N030 X1 F10	
N031 X100 F500	
N032 X200	
N040 M3 S500	; Axis does not trigger a preprocessing stop / REORG
N041 G4 F2	
N050 M5	
N099 M30	

If the spindle or axis B is traversed, e.g. to 180 degrees and then back to 1 degree immediately after block N023 as the **PLC axis**, this axis will revert to its neutral status and will not trigger a preprocessing stop in block `N40`.

Further information

Requirements for axis replacement

- The axis must be defined in all channels that use the axis in the machine data.
- It is necessary to define to which channel the axis will be assigned after POWER ON in the **axis**-specific machine data.

Description

Release axis: RELEASE

When enabling the axis please note:

1. The axis must not be involved in a transformation.
2. All the axes involved in an axis link (tangential control) must be enabled.
3. A concurrent positioning axis cannot be replaced in this situation.
4. All the following axes of a gantry master axis are transferred with the master.
5. With coupled axes (coupled motion, master value coupling, electronic gear) only the leading axis of the group can be enabled.

Accept axis: GET

The actual axis replacement is performed with this command. The channel for which the command is programmed takes full responsibility for the axis.

Effects of GET:

Axis replacement with synchronization:

An axis always has to be synchronized if it has been assigned to another channel or the PLC in the meantime and has not been synchronized with "WAITP", G74 or delete distance-to-go before GET.

- A preprocess stop occurs (as for STOPRE).
- Execution is interrupted until the replacement has been completed.

Automatic "GET"

If an axis is in principle available in a channel but is not currently defined as a "channel axis", GET is executed automatically. If the axis/axes is/are already synchronized no preprocess stop is generated.

Setting the axis replacement behavior variably

The transfer point of axes can be set as follows using machine data:

- Automatic axis replacement between two channels then also takes place when the axis has been brought to a neutral state by WAITP (response as before)
- When requesting an axis container rotation, all axes of the axis container which can be assigned to the executing channel are brought into the channel using implicit GET or GETD. A subsequent axle replacement is only permitted again once the axis container rotation has been completed.

- When an intermediate block is inserted in the main run, a check will be made to determine whether or not reorganization is required. Reorganization is only necessary if the axis states of this block do **not** match the current axis states.
- Instead of a GET block with preprocess stop and synchronization between preprocessing and main run, axes can be replaced without a preprocess stop. In this case, an intermediate block is simply generated with the GET request. In the main run, when this block is executed, the system checks whether the states of the axes in the block match the actual axis states.

4.15.2 Transfer axis to another channel (AXTOCHAN)

The `AXTOCHAN` language command can be used to request an axis in order to move it to a different channel. The axis can be moved to the corresponding channel both from the NC part program and from a synchronized action.

Syntax

```
AXTOCHAN(axis name,channel number[,axis name,channel number[,...]])
```

Meaning

Element	Description
`AXTOCHAN:`	Request axis for a specific channel
`Axis name:`	Axis assignment in the system: X, Y, ... or entry of machine axis names concerned. The executing channel does not have to be the same channel or even the channel currently in possession of the interpolation right for the axis.
`Channel number:`	Name of the channel to which the axis is to be assigned

Note

Competing positioning axis and PLC controlled axis exclusively

A PLC axis cannot replace the channel as a competing positioning axis. An axis controlled exclusively by the PLC cannot be assigned to the NC program.

Further information: Function Manual Axes and Spindles

Example

AXTOCHAN in the NC program

Axes X and Y have been declared in the first and second channels. Currently, channel 1 has the interpolation right and the following program is started in that channel.

Program code	Comment
`N110 AXTOCHAN(Y,2)`	`;Move Y axis to the second channel`
`N111 M0`	
`N120 AXTOCHAN(Y,1)`	`; Retrieve Y axis (neutral).`
`N121 M0`	

Program code	Comment
N130 AXTOCHAN(Y,2,X,2)	;Move Y axis and X axis to the second channel (axes are neutral).
N131 M0	
N140 AXTOCHAN(Y,2)	; Move Y axis to the second channel (NC program).
N141 M0	

Further information

AXTOCHAN in the NC program

A `GET` is only executed in the event of the axis being requested for the NC program in the same channel (this means that the system waits for the state to actually change). If the axis is requested for another channel or is to become the neutral axis in the same channel, the request is sent accordingly.

AXTOCHAN from a synchronized action

In the event of an axis being requested for the same channel, `AXTOCHAN` from a synchronized action is mapped to a `GET` from a synchronized action. In this case, the axis becomes the neutral axis on the first request for the same channel. On the second request, the axis is assigned to the NC program in the same way as the GET request in the NC program. For more information about GET requests from a synchronized action, see "Motion-synchronous actions".

4.15.3 Axis functions (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL)

"AXNAME" is used, e.g. to generate cycles that are generally valid, if the names of the axes are not known.

"AX" is used to indirectly program geometry and synchronous axes. The axis identifier is saved in a type AXIS variable or is supplied from a command such as "AXNAME" or "SPI".

"SPI" is used if axis functions are programmed for a spindle, e.g. a synchronous spindle.

"AXTOSPI" is used to convert an axis identifier into a spindle index (inverse function to "SPI").

"AXSTRING" is used to convert an axis identifier (data type AXIS) into a string (inverse function to "AXNAME").

"ISAXIS" is used in universal cycles in order to ensure that a specific geometry axis exists and thus that any following $P_AXNX call is not aborted with an error message.

"MODAXVAL" is used in order to determine the modulo position for modulo rotary axes.

Syntax

```
AXNAME("string")
AX[AXNAME("string")]
SPI(n)
```

`AXTOSPI(A)` or `AXTOSPI(B)` or `AXTOSPI(C)`

```
AXSTRING( SPI(n) )
ISAXIS(<geometry axis number>)
<Modulo position>=MODAXVAL(<axis>,<axis position>)
```

Meaning

AXNAME:	Converts an input string into axis identifiers; the input string must contain a valid axis name.
AX:	Variable axis identifier
SPI:	Converts the spindle number into an axis identifier; the transfer parameter must contain a valid spindle number.
n:	Spindle number
AXTOSPI:	Converts an axis identifier into an integer spindle index. "AXTOSPI" corresponds to the inverse function to "SPI".
X, Y, Z:	Axis identifier of AXIS type as variable or constant
AXSTRING:	The string is output with the associated spindle number.
ISAXIS:	Checks whether the specified geometry axis exists.
MODAXVAL:	For modulo rotary axes, determines the modulo position; this corresponds to the modulo rest referred to the parameterized modulo range (in the default setting, this is 0 to 360 degrees; the start and size of the modulo range can be changed using MD30340 MODULO_RANGE_START and MD30330 $MA_MODULO_RANGE).

Note

SPI extensions

The axis function SPI(n) can also be used to read and write frame components. This means that frames can be written, e.g. with the syntax `$P_PFRAME[SPI(1),TR]=2.22`.

An axis can be traversed by additionally programming axis positions using the address `AX[SPI(1)]=<axis position>`. The prerequisite is that the spindle is either in the positioning or axis mode.

Examples

Example 1: AXNAME, AX, ISAXIS

Program code	Comment
`OVRA[AXNAME("Transverse axis")]=10`	; Override for transverse axis
`AX[AXNAME("Transverse axis")]=50.2`	; End position for transverse axis
`OVRA[SPI(1)]=70`	; Override for spindle 1
`AX[SPI(1)]=180`	; End position for spindle 1
`IF ISAXIS(1) == FALSE GOTOF CONTINUE`	; Abscissa available?
`AX[$P_AXN1]=100`	; Move abscissa
`CONTINUE:`	

Example 2: AXSTRING

When programming with AXSTRING[SPI(n)], the axis index of the axis, which is assigned to the spindle, is no longer output as spindle number, but instead the string "Sn" is output.

Program code	Comment
AXSTRING[SPI(2)]	; String "S2" is output.

Example 3: MODAXVAL

The modulo position of modulo rotary axis A is to be determined.

Axis position 372.55 is the starting value for the calculation.

The parameterized modulo range is 0 to 360 degrees:

MD30340 MODULO_RANGE_START = 0

MD30330 $MA_MODULO_RANGE = 360

Program code	Comment
R10=MODAXVAL(A,372.55)	; Calculated modulo position R10 = 12.55.

Example 4: MODAXVAL

If the programmed axis identifier does not refer to a modulo rotary axis, then the value to be converted (<axis position>) is returned unchanged.

Program code	Comment
R11=MODAXVAL(X,372.55)	; X is a linear axis; R11 = 372.55.

4.15.4 Replaceable geometry axes (GEOAX)

The "Switchable geometry axes" function allows the geometry axes configured via machine data to be replaced by other channel axes.

Syntax

```
GEOAX(<n>,<channel axis>,<n>,<channel axis>,<n>,<channel axis>)
GEOAX()
```

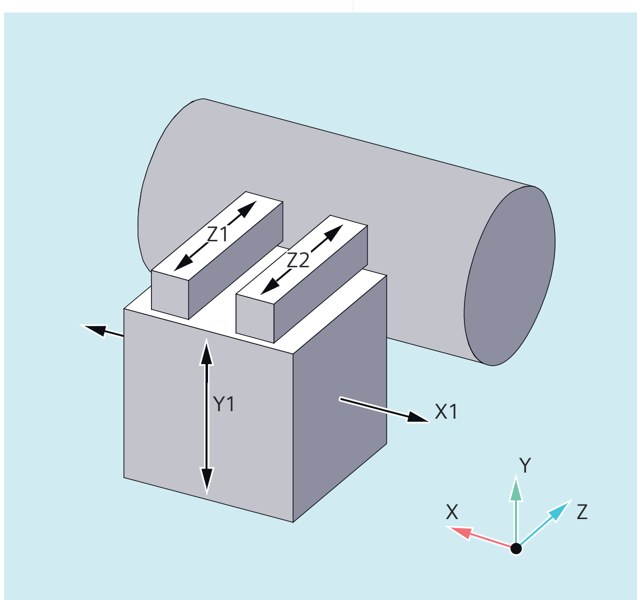
Meaning

`GEOAX(...)`	Function for switching geometry axes. **Note:** `GEOAX()` without parameter specification activates the basic configuration of the geometry axes parameterized in the machine data again.
`<n>`	Number of the geometry axis that is to be replaced by the specified channel axis. Range of values: 0, 1, 2, 3 **Note:** 0: The specified channel axis is removed from the geometry axis group without being replaced 1: 1. geometry axis ≙ coordinate axis X (abscissa) of the WCS 2: 2. geometry axis ≙ coordinate axis Y (ordinate) of the WCS 3: 3. geometry axis ≙ coordinate axis Z (applicate) of the WCS
`<channel axis>`	Name of the channel axis which is to added to the geometry axis group

Examples

Example 1: Switching two axes alternating as geometry axis

A tool slide can be traversed using channel axes X1, Y1, Z1, Z2:



The geometry axes are configured so that after powering-up, initially Z1 is effective as 3rd geometry axis under the geometry axis name "Z" and together with X1 and Y1 forms the geometry axis group.

Axes Z1 and Z2 should now be used, alternating, as geometry axis Z in the part program:

Program code	Comment
...	
N100 GEOAX(3,Z2)	; Channel axis Z2 acts as 3rd geometry axis (Z).
N110 G1 ...	
N120 GEOAX(3,Z1)	; Channel axis Z1 acts as 3rd geometry axis (Z).
...	

Example 2: Changing over the geometry axes for six channel axes

A machine has six channel axes with the names XX, YY, ZZ, U, V, W.

The basic setting of the geometry axis configuration via machine data is:

Channel axis XX = 1st geometry axis (X axis)

Channel axis YY = 2nd geometry axis (Y axis)

Channel axis ZZ = 3rd geometry axis (Z axis)

Program code	Comment
N10 GEOAX()	; The basic configuration of the geometry axes is effective.
N20 G0 X0 Y0 Z0 U0 V0 W0	; All axes in rapid traverse to position 0.
N30 GEOAX(1,U,2,V,3,W)	; Channel axis U becomes the first (X), V the second (Y)
	; and W the third geometry axis (Z).
N40 GEOAX(1,XX,3,ZZ)	; Channel axis XX becomes the first (X), ZZ the third
	; geometry axis (Z). Channel axis V remains the second
	; geometry axis (Y).
N50 G17 G2 X20 I10 F1000	; Full circle in the X/Y plane. Channel axes
	; XX and V traverse.
N60 GEOAX(2,W)	; Channel axis W becomes the second geometry (Y).
N80 G17 G2 X20 I10 F1000	; Full circle in the X/Y plane. Channel axes
	; XX and W traverse.
N90 GEOAX()	; Reset to the initial state.
N100 GEOAX(1,U,2,V,3,W)	; Channel axis U becomes the first (X), V the second
	; (Y) and W the third geometry axis (Z).
N110 G1 X10 Y10 Z10 XX=25	; Channel axes U, V, W each traverse to
	; position 10. XX as special axis traverses to position 25.
N120 GEOAX(0,V)	; V is removed from the geometry axis group.
	; U and W remain the first (X) and third
	; geometry axis (Z).
	; The second geometry (Y) axis remains unassigned.
N130 GEOAX(1,U,2,V,3,W)	; Channel axis U remains the first (X), V becomes
	; the second (Y), W remains the third geometry axis (Z).

Program code	Comment
N140 GEOAX(3,V)	; V becomes the third geometry axis (Z), whereby W
	; is overwritten and therefore removed from the geom-etry
	; axis group. The second geometry axis (Y)
	; still remains unassigned.

Machine data

Axis configuration

Assignment of geometry, special and machine axes to channel axes:

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB
- MD20050 $MC_AXCONF_GEOAX_ASIGN_TAB
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB
- MD20070 $MC_AXCONF_MACHAX_USED
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX

Reset behavior

Reset behavior of changed geometry axis assignments:

- MD20110 $MC_RESET_MODE_MASK, bit 12
- MD20118 $MC_GEOAX_CHANGE_RESET

NC start behavior

- MD20112 $MC_START_MODE_MASK, bit 12

Notification to the PLC user program

Parameterization option of the M command which is output on the NC/PLC interface when the geometry axes are changed.

- MD22532 $MC_GEOAX_CHANGE_M_CODE

Supplementary conditions

No geometry axis changeover

- If one of the following functions is active, a geometry axis changeover is not possible:
  - Transformation
  - Spline interpolation
  - Tool radius compensation
  - Tool fine offset
- The geometry axis and another channel axis have the same name.
- One of the axes participating in the geometry axis changeover is involved in an action that goes beyond block limits, e.g. block-wide positioning axis or following axis of an axis coupling.

Rotary axes

Rotary axes cannot be programmed as geometry axes.

Axis state after replacing

An axis replaced by the changeover in the geometry axis group can be programmed as supplementary axis after the changeover operation via its channel axis names.

Frames, protection areas, working area limits

All frames, protection areas and working area limits are deleted after changing over the geometry axes.

Polar coordinates

Replacing the geometry axes with `GEOAX` sets analog to a level change with `G17`-`G19`, the modal polar coordinates to a value of 0.

DRF, WO

A possible handwheel offset (DRF) or an external work offset (WO) remains effective after the changeover.

Basic configuration of the geometry axes

The `GEOAX()` command calls the basic configuration of the geometry axis group.

The system automatically changes back to the basic configuration after POWER ON and when changing over into the "reference point approach" mode.

Tool length compensation

An active tool length compensation is also effective after the changeover operation. However, for geometry axes that have been newly added or those where the position has been replaced, it is still considered not to have been moved through. For the first motion command for these geometry axes, the resulting traversing distance correspondingly comprises the sum of the tool length compensation and the programmed traversing distance.

Geometry axes, which retain their position in the axis group after a replacement operation, also retain their status with respect to tool length compensation.

Geometry axis configuration for active transformation

- The geometry axis configuration parameterized for an active transformation via transformation machine data cannot be changed using the "Switchable geometry axes" function.
- Different data sets must be parameterized in the transformation machine data for a different geometry axis configuration for a transformation.
- A geometry axis configuration changed using `GEOAX` is deleted by activating a transformation.
- With regard to the geometry axes, the transformation-specific geometry axis parameterizations of active transformations have priority over the parameterizations relevant for the changeover of geometry axes.
  Example: A transformation is active. According to the machine data, the transformation should be retained at a channel reset. At the same time, the basic configuration of the geometry axes should be restored at a channel reset. The geometry axis configuration that has been specified for the transformation is retained.
- If a transformation is switched off, the parameterized basic setting of the geometry axis configuration takes effect again.

JOG mode, REF machine function

When switching over to the JOG mode, REF machine function (reference point approach), the geometry axis configuration parameterized in the machine data takes effect

4.15.5 Wait for valid axis position (WAITENC)

Using the language command "WAITENC", the NC program waits until the synchronized or restored axis positions are available for the axes configured with
MD34800 $MA_WAIT_ENC_VALID = 1.

An interruption can take place in the wait state, e.g. by starting an ASUB or by changing the operating mode to JOG. When the program is continued, where relevant, the wait state is resumed.

Note

In the user interface, the wait state is displayed using the hold state "Wait for measuring system".

Syntax

"WAITENC" can be programmed in the program section of any NC program.

Programming must be realized in a dedicated block:

```
...
WAITENC
...
```

Example

"WAITENC" is for example used in an event-controlled user program, .../_N_CMA_DIR/_N_PROG_EVENT_SPF, as shown in the following application example.

Application example: Tool withdrawal after POWER OFF with orientation transformation

Machining with tool orientation was interrupted due to a power failure.
When powering up again, the event-controlled user program .../_N_CMA_DIR/_N_PROG_EVENT_SPF is called.

In the event-controlled user program, the system waits for synchronized or restored axis positions using "WAITENC"; in order to then be able to calculate a frame, which aligns the Work in the tool direction.

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`	
`...`	

The tool can then be retracted in JOG mode by means of a retraction movement towards the tool axis.

4.15.6 Programmable parameter set changeover (SCPARA)

The changeover to a specific parameter set can be requested for an axis using the SCPARA command.

Note

No parameter set changeover during thread cutting

During thread cutting G33 and tapping G331/G332, the parameter set is selected by the control and cannot be changed.

Disabled parameter set changeover

A parameter set changeover can also be requested via the NC/PLC interface. In order to avoid changeover conflicts, the parameter set changeover of the NC (SCPARA) can be blocked via the NC/PLC interface:

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

Note

If a parameter set changeover is requested by SCPARA while the parameter set changeover is locked via the NC/PLC interface, the changeover is rejected without an error message.

Syntax

```
SCPARA[<Axis>]=<Value>
```

Meaning

`SCPARA`:	Command: Change parameter set	
`<axis>`:	Axis identifier (channel axis)	
	Type:	AXIS
`<Value>`:	Parameter set number: 1, 2, 3, ... max. parameter block number	

Example

```
Program code            Comment
...
N110 SCPARA[X]= 3       ; Select: Axis X, 3. Parameter set
...
```

Further information

Enable of the parameter set changeover

The parameter set changeover of the axis must be explicitly enabled:

MD35590 $MA_PARAMSET_CHANGE_ENABLE[<axis>]

Read parameter set number

The number of the selected parameter set (specified parameter set) can be read via the system variable $AA_SCPAR.

4.15.7 Activate/deactivate adaptation (CADAPTON, CADAPTOF)

Using the predefined CADAPTON() and CADAPTOF() procedures, predefined adaptations for adjusting the dynamic response or control parameters can be activated, updated and deactivated from the part program. They are also used in CYCLE782 (Page 961).

Syntax

```
...
CADAPTON(<Result>,<Axis>,<InVar>[,<InVal>])
...
CADAPTOF(<Result>,<Axis>,<InVar>)
...
```

Meaning

CADAPTON():	Activates adaptation relationship		
CADAPTOF():	Deactivates adaptation relationship **Note:** Ignore active adaptation relationship (MD16501 = 1) CADAPTOF(), a previously programmed input value (<InVal>) remains active.		
<Result>:	Result variable: Return value for the status (called by reference parameter)		
	Data type:	INT	
	Value:	0	No error
		1	No valid adaptation table parameterized
		2	<Axis> parameter invalid
		3	<InVar> parameter invalid
		4	Reserved
		5	<InVal> parameter invalid
<Axis>:	Machine axis name of the input axis of the adaptation relationship		
	Data type:	AXIS	
	Range of values:	Machine axis names defined in the channel	
	Note: Using this parameter, those adaptations are addressed, which, in MD16504 $MN_CADAPT_INPUT_AX, entered a value corresponding to parameter <Axis>. The other <InVar> and <InVal> parameters are assigned to this axis.		
<InVar>:	Input variable of the adaptation relationship		
	Data type:	INT	
	Value:	1	Inertia of the axis
		2	Axis position
		3	Axis speed

<InVal>:	Input value of the adaptation relationship Optional parameter for intelligent load adjustment (<InVar>=1).		
	Data type:	REAL	
	Value:	> 0	Actual moment of inertia
		= 0	Load/loading is not defined or is not known. In this particular case, the assigned adaptations supply substitute value 1.0 as output variable.
		< 0	The call is exited with <Status> = 5 (parameter <InVal> invalid). The last input value activated remains active.
	Note: If an input value is not programmed, the last programmed value or the default value (= 0) after the control system has run up is active.		

Example

Program code	Comment
`DEF INT RESULT`	`; Definition of result variables`
`...`	
`CADAPTON(RESULT,MX1,2)`	`; Activate adaptation`
`IF RESULT <> 0`	`; Evaluate the result variables (this is required after each CADAPTON/CADAPTOF instruction)`
`   MSG("CADAPT-RESULT=" << RESULT)`	
`   STOPRE`	
`   SETAL(61000)`	
`ENDIF`	
`...`	
`CADAPTOF(RESULT,MX1,2)`	`; Deactivate adaptation`
`...`	`; Evaluate result variables`
`...`	

Further information

Block search

- **Block search without calculation**
  CADAPTON/CADAPTOF instructions that are programmed between the beginning of the program and target block are ignored. Users must select the search target so that the adaptations relevant for the machining section are selected.
- **Block search with calculation at the contour or at block end point**
  The activation and/or deactivation commands - as well as possible programmed input values - are collected, and the actual status at the end of the search is output with the search action blocks.
- **Block search with calculation in Program test (SERUPRO) mode**
  When SERUPRO is deactivated, the status of the adaptations achieved using the CADAPTON/CADAPTOF instructions is transitioned into real operation.

4.15.8 Adapting the FIR jerk filter to the dynamic mode (CALCFIR)

After changing the dynamic response mode, in order to achieve an identical damping effect and contour accuracy for all axes in the channel for which an FIR lowpass jerk filter is active, the dynamic-response dependent FIR filter settings must be calculated and activated. This is realized by calling the NC language command CALCFIR.

CALCFIR is either called automatically or manually depending on the setting in machine data MD20570 $MC_CALCFIR_BY_DYN_MODE_CHANGE:

- MD20570 = 1: Automatic call (recommended variant)
  CALCFIR does not have to be explicitly programmed, but can be automatically called after each change to the dynamic response mode.
- MD20570 = 0: Manual call (default setting)
  CYCLE832 is programmed at the start of an NC program for free-form surface machining. When the program is being executed, CYCLE832 calls the manufacturer cycle CUST_832 and the NC language command CALCFIR inserted by the machine manufacturer in CUST_832 is executed.

Effectiveness

The filter settings overwritten by CALCFIR remain active until they are again overwritten by the next automatic or manual CALCFIR call. This is always required, if, using a command of G group 59 (dynamic response mode for path interpolation), a change to the actual dynamic response mode is programmed.

Response when the control system powers up / channel reset / end of program reset

When the control system powers up, and for a channel/end of program reset, the FIR filter settings are activated to match the initial setting of G group 59.

Requirements

- "Top Speed Plus" option is set.

  Note

  To achieve optimum results with "Top Speed Plus", it is recommended that option "Top Surface" is also used.

- The following conditions must be satisfied for axes in the interpolation group:
  - FIR low-pass jerk filter is active.
  - FIR filter settings dependent on the dynamic response are configured.
  - Overwriting filter settings by CALCFIR is enabled.
- Jerk limiting SOFT/SOFTA is active.

Syntax

CALCFIR is programmed in a separate block. The call in the manufacturer cycle CUST_832 is realized directly after programming the dynamic response mode:

```
...
```

```
DYN...
CALCFIR
...
```

Meaning

`DYN...:`	Command from G group 59 to select the dynamic response mode	
	`DYNNORM:`	Activate **normal dynamic response**
	`DYNPOS:`	Activate dynamic response for **positioning mode, tapping**
	`DYNROUGH:`	Activate dynamic response for **roughing**
	`DYNSEMIFIN:`	Activate dynamic response for **semi-finishing**
	`DYNFINISH:`	Activate dynamic response for **finishing**
	`DYNPREC:`	Activate dynamic response for **smooth-finishing**
`CALCFIR:`	Predefined procedure to dynamically adapt the FIR low-pass jerk filters	

Note

The automatic or manual call of CALCFIR results in an implicit NEWCONFIG to activate the result of the FIR filter calculation - both in the axis-specific machine data to parameterize the FIR low-pass jerk filter as well as for CPRECON (Page 759). The implicit NEWCONFIG means that additional NEWCONFIG-relevant machine data become active.

Constraints

Use in synchronized actions

CALCFIR **cannot** be programmed in synchronized actions.

4.15.9 Read/write drive parameters in the part program (DRVPRD, DRVPWR)

The DRVPRD and DRVPWR voice commands enable the machine manufacturer to read and write drive parameters axis-specifically at the part program level.

Since DRVPRD and DRVPWR are predefined procedures that must be in an NC block alone, exactly one drive parameter per channel can be read **or** written at any given time.

DRVPRD and DRVPWR are used in a cycle created by the machine manufacturer.

Prerequisites

The following requirements apply:

- Commissioning of the NC-controlled drive is completed.
- Cycles in the CST directory (Siemens) or CMA directory (manufacturer) always have the required read and write permissions.

Syntax

```
DRVPRD(<Result>,<Axis>,<DrvParNo>,<DrvParIdx>,<Value>)
DRVPWR(<Result>,<Axis>,<DrvParNo>,<DrvParIdx>,<Value>)
```

Meaning

DRVPRD():	Predefined procedure for **reading** drive parameters		
DRVPWR():	Predefined procedure for **writing** drive parameters		
<Result>:	Result variable: Return value for the status		
	Data type:	INT	
	Value:	= 0	Function executed without error
		> 0	Function terminated with error For details, see "More information" > "Diagnostics via return values".
<Axis>:	Machine axis name		
	Data type:	AXIS	
	Value range:	Machine axis names defined in the channel	
<DrvParNo>:	Number of the drive parameter		
	Data type:	INT	
<DrvParIdx>:	Index of the drive parameter		
	Data type:	INT	
<Value>:	Value of the drive parameter DRVPRD: read value DRVPWR: value to be written		
	Data type:	REAL	

Example

Read parameter p1460[0] "Speed controller P gain adaptation speed" of the X axis:

Program code	Comment
N100 DEF INT _Status	; Variable definition for return value of the voice command
N110 DEF REAL _Result	; Variable definition for read parameter value
N120 DRVPRD(_Status,AX1,1460,0,_Result)	; Read parameter p1460

More information

Diagnostics via return values

Access to a drive parameter is invalid if the return value of the voice command returns a non-zero value.

Two number ranges are reserved for the return value:

Return values when the voice command is called:

0 No access error.
1 Impermissible value.
2 Axis not available.
3 Drive not available.
4 Missing access rights.
5 Not possible in block search/program test.
6 Access to the drive is not possible.

Return values when writing/reading the drive parameter:

1000 Impermissible parameter number.
1001 Parameter value cannot be changed.
1002 Lower or upper limit violated.
1003 Sub-index incorrect.
1004 No array, no sub-index.
...
1107 Write access not allowed when controller is enabled.
1110 Write access only permitted in the commissioning state: Motor (p0010 = 3).
...
1204 No write access.

To distinguish the return values of the voice command from the return values of the drive, the value 1000 is added.

More information: SINUMERIK Diagnostics Manual, alarm 201042 "Parameter error during project download"

Behavior during program simulation

The available drive parameters depend on the type and scope of drive simulation used. If the drive simulation is missing, the voice command returns the return value 3.

With the variable $P_SIM, the machine manufacturer has the possibility to provide a special treatment for the program simulation.

Interaction with other functions

- Parallel write/read accesses
  Parallel read/write accesses from the HMI or other clients can have an effect on the execution time of the voice command.
- REPOS
  If the execution of the voice command is interrupted by a REORG event, the command is repeated.
- Block search / program test
  In the block search and program test (also SERUPRO), the voice command for reading and writing drive parameters is not supported. The result variable returns access error 5.
  A query with the system variables $P_SEARCH and $P_ISTEST in the part program as to whether block search or program test is active makes it possible, for example, to skip DRVPRD and DRVPWR and implement a replacement strategy.

Constraints

The following conditions must be observed during programming:

- Use in synchronized actions
  The voice command cannot be programmed in synchronized actions because a synchronized action does not wait for the non-cyclic access to be executed.
  When using the voice command in a synchronized action, alarm 12571 is output.
- Readable/writable drive parameters
  Only axis-specific drive parameters of a SERVO DO or HLA DO and parameters of PROFIdrive standard drives that are assigned to NC axes can be read or written.
  It is not possible to read/write parameters of the following DOs: CU_I, CU_NX, CU3x, TM, HUB, INFEED.

4.16 Axis couplings

4.16.1 Coupled motion (TRAILON, TRAILOF)

When a defined leading axis is moved, the coupled motion axes (= following axes) assigned to it traverse through the distances described by the leading axis, allowing for a coupling factor.

Together, the leading axis and following axis represent coupled axes.

Application

Typical applications are:

- Traversing of an axis by means of a simulated axis
  The leading axis is a simulated axis and the coupled axis a real axis. In this way, the real axis can be traversed as a function of the coupling factor.
- Two-sided machining with 2 coupled motion groups

Syntax

```
TRAILON(<following axis>,<leading axis>,<coupling factor>)
```

```
TRAILOF(<following axis>,<leading axis>,<leading axis 2>)
TRAILOF(<following axis>)
```

Meaning

`TRAILON`:	Command for activating and defining a coupled axis grouping Effective: Modal
`<following axis>`:	Parameter 1: Axis name of trailing axis **Note:** A coupled-motion axis can also act as the leading axis for other coupled-motion axes. In this way, it is possible to create a range of different coupled axis groupings.
`<leasing axis>`:	Parameter 2: Axis name of trailing axis
`<coupling factor>`:	Parameter 3: Coupling factor The coupling factor specifies the desired relationship between the paths of the coupled-motion axis and the leading axis: <coupling factor> = path of coupled-motion axis/path of leading axis Type: REAL Default: 1 The input of a negative value causes the master and coupled axes to traverse in opposition. If a coupling factor is not programmed, then coupling factor 1 automatically applies.

`TRAILOF`:	Command for deactivating a coupled axis grouping
	Effective: Modal
	`TRAILOF` with 2 parameters deactivates only the coupling to the specified leading axis:
	`TRAILOF(<following axis>,<leading axis>)`
	If a coupled-motion axis has two leading axes, `TRAILOF` can be called with three parameters to deactivate both couplings.
	`TRAILOF(<following axis>,<leading axis>,<leading axis 2>)`
	Programming `TRAILOF` without specifying a leading axis produces the same result:
	`TRAILOF(<following axis>)`

Note

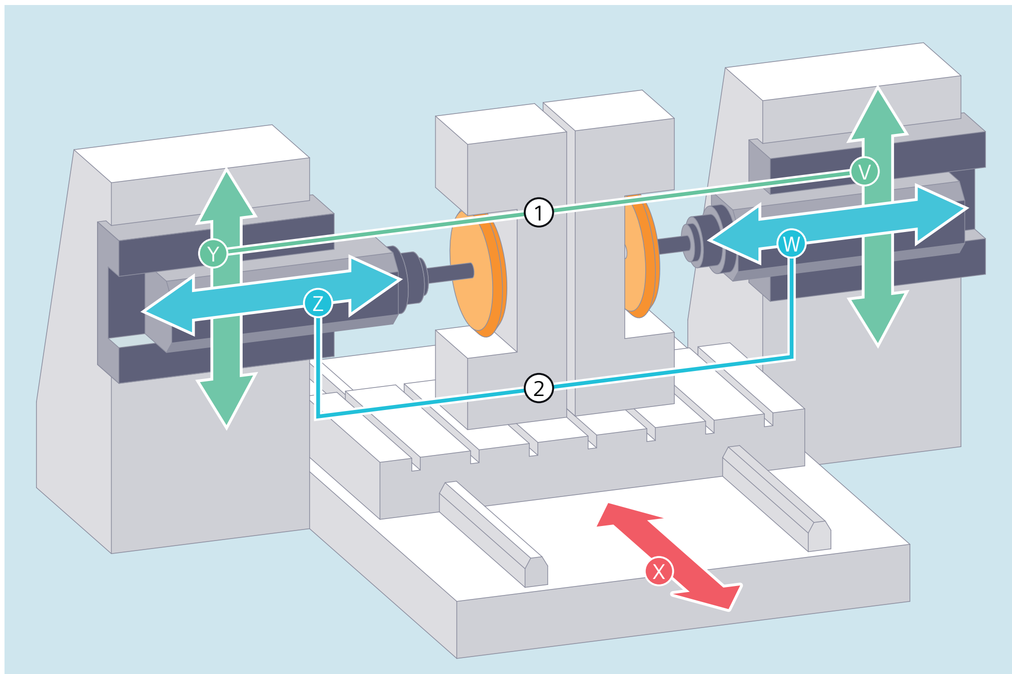
Coupled axis motion is always executed in the base coordinate system (BCS).

The number of coupled axis groupings which may be simultaneously activated is limited only by the maximum possible number of combinations of axes on the machine.

Example

Two-sided machining with two coupled motion groups:

- 1st leading axis Y, coupled motion axis V
- 2nd leading axis Z, coupled motion axis W



Program code	Comment
…	
N100 TRAILON(V,Y)	; Activation of 1st coupled axis group.
N110 TRAILON(W,Z,-1)	; Activation of 2nd coupled axis grouping, Negative coupling factor: Coupled-motion axis traverses in the opposite direction from leading axis.
N120 G0 Z10	; Infeed of Z and W axes in opposite axial directions.
N130 G0 Y20	; Infeed of Y and V axes in same axis direction.
…	
N200 G1 Y22 V25 F200	; Overlaying of a dependent and independent movement of coupled motion axis V.
…	
TRAILOF(V,Y)	; Deactivation of 1st coupled axis grouping.
TRAILOF(W,Z)	; Deactivation of 2nd coupled axis grouping.

Further information

Axis types

A coupled axis grouping can consist of any desired combinations of linear and rotary axes. A simulated axis can also be defined as a leading axis.

Coupled-motion axes

Up to two leading axes can be assigned simultaneously to a trailing axis. The assignment is made in different combinations of coupled axes.

A coupled-motion axis can be programmed with the full range of available motion commands (`G0`, `G1`, `G2`, `G3`, etc.). The coupled axis not only traverses the independently defined paths, but also those derived from its leading axes on the basis of coupling factors.

Dynamics limit

The dynamics limit is dependent on the type of activation of the coupled axis grouping:

- Activation in part program
  If activation is performed in the part program and all leading axes are active as program axes in the activated channel, the dynamic response of all coupled-motion axes is taken into account during traversing of the leading axis to avoid overloading the coupled-motion axes.
  If activation is performed in the part program with leading axes that are not active as program axes in the activating channel ($AA_TYP ≠ 1), then the dynamic response of the coupled-motion axes is not taken into account during traversing of the leading axis. This can cause the overloading of coupled-motion axes with a dynamic response which is less than that required for the coupling.
- Activation in synchronized action
  If activation is performed in a synchronized action, the dynamic response of the coupled-motion axes is not taken into account during traversing of the leading axis. This can cause the overloading of coupled-motion axes with a dynamic response which is less than that required for the coupling.

> ⚠ **CAUTION**
>
> **Axis overload**
>
> If a coupled axis grouping is activated:
> - In synchronized actions
> - In the part program with leading axes that are not program axes in the channel of the coupled-motion axes
>
> It is the specific responsibility of the user / machine manufacturer to take suitable action to ensure that the traversing of the leading axis will not cause the overloading of the coupled-motion axes.

Coupling status

The coupling status of an axis can be checked in the part program with the system variable:

$AA_COUP_ACT[<axis>]

Value	Meaning
0	No coupling active
8	Coupled motion active

Display of distance-to-go of the coupled-motion axis for modulo rotary axes

If the leading and coupled-motion axes are modulo rotary axes, traversing movements in the leading axis from n * 360° with n = 1, 2, 3 ... , add up in the distance-to-go display of the coupled-motion axis until the coupling is switched off.

Example: Program section with TRAILON and leading axis B and following axis C

Program code	Comment
TRAILON(C,B,1)	; Activate coupling
G0 B0	; Starting position
	; Distance-to-go display at block start:
G91 B360	; B=360, C=360
G91 B720	; B=720, C=1080
G91 B360	; B=360, C=1440

4.16.2 Electronic gear (EG)

The "Electronic gear" function allows you to control the movement of a **following axis** according to linear traversing block as a function of up to five **leading axes**. The relationship between each leading axis and the following axis is defined by the coupling factor.

The following axis motion part is calculated by an addition of the individual leading axis motion parts multiplied by their respective coupling factors. When an EG axis grouping is activated, it is possible to synchronize the following axes in relation to a defined position. A gear group can be:

- Defined
- Activated
- Deactivated
- Deleted

.

The following axis movement can be optionally derived from

- Setpoints of the leading axes, as well as
- Actual values of leading axes.

Non-linear relationships between each leading axis and the following axis can also be realized as extension using **curve tables** (see "Path traversing behavior" section). Electronic gears can be cascaded, i.e., the following axis of an electronic gear can be the leading axis for a further electronic gear.

4.16.2.1 Defining an electronic gear (EGDEF)

An EG axis group is defined by specifying the following axis and at least one, however not more than five, leading axis, each with the relevant coupling type.

Requirement

Requirements for defining an EG axis group:

It is not permissible to define an axis coupling for the following axis (or an existing one must first be deleted with EGDEL).

Syntax

```
EGDEF(following axis,leading axis1,coupling type1,leading
axis2,coupling type2,...)
```

Meaning

`EGDEF:`	Definition of an electronic gear	
`Following axis:`	Axis that is influenced by the leading axes	

`Leading axis1`
`,...,`
`Leading axis5`
	Axes that influence the following axis	

`Coupling type1`
`,...,`
`Coupling type5`
	Coupling type
	The coupling type does not need to be the same for all leading axes and must be programmed separately for each individual master.	

	Value:	**Meaning:**
	0	The following axis is influenced by the **actual value** of the corresponding leading axis.
	1	The following axis is influenced by the **setpoint** of the corresponding leading axis.

Note

The coupling factors are preset to zero when the EG axis grouping is defined.

Note

EGDEF triggers preprocessing stop. The gearbox definition with EGDEF should also be used unaltered if, for systems, one or more leading axes affect the following axis via a **curve table**.

Example

Program code	Comment
`EGDEF(C,B,1,Z,1,Y,1)`	`; Definition of an EG axis group. Leading axes B, Z, Y influence the following axis C via the setpoint.`

4.16.2.2 Switch-in the electronic gearbox (EGON, EGONSYN, EGONSYNE)

There are 3 ways to switch-in an EG axis group.

Syntax

Variant 1:

The EG axis group is selectively switched-in without synchronization with:

```
EGON(FA,"block change mode",LA1,Z1,N1,LA2,Z2,N2,...,LA5,Z5,N5)
```

Variant 2:

The EG axis group is selectively activated with synchronization with:
`EGONSYN(FA,"block change mode",SynPosFA,[,LAi,SynPosLAi,Zi,Ni])`

Variant 3:

The EG axis group is selectively switched-in with synchronization and the approach mode specified with:
`EGONSYNE(FA,"block change mode",SynPosFA,approach mode[,LAi,SynPosLAi,Zi,Ni])`

Meaning

Variant 1:

FA	Following axis	
`Block change mode:`	The following modes can be used:	
	`"NOC"`	Block change takes place immediately
	`"FINE"`	Block change is performed in "Fine synchronism"
	`"COARSE"`	Block change is performed in "Coarse synchronism"
	`"IPOSTOP"`	Block change is performed for setpoint-based synchronism
LA1, ... LA5	Leading axes	
Z1, ... Z5	Numerator for coupling factor i	
N1, ... N5	Denominator for coupling factor i Coupling factor i = numerator i/denominator i	

Only the leading axes previously specified with the EGDEF command may be programmed in the activation line. At least one leading axis must be programmed.

Variant 2:

`FA`	Following axis	
`Block change mode:`	The following modes can be used:	
	`"NOC"`	Block change takes place immediately
	`"FINE"`	Block change is performed in "Fine synchronism"
	`"COARSE"`	Block change is performed in "Coarse synchronism"
	`"IPOSTOP"`	Block change is performed for setpoint-based synchronism
`[,LAi,SynPosLAi,Zi,Ni]`	(do not write the square brackets) Min. 1, max. 5 sequences of:	
`LA1, ... LA5`	Leading axes	
`SynPosLAi`	Synchronized position for i-th leading axis	
`Z1, ... Z5`	Numerator for coupling factor i	
`N1, ... N5`	Denominator for coupling factor i Coupling factor i = numerator i/denominator i	

Only leading axes previously specified with the EGDEF command may be programmed in the activation line. Through the programmed "Synchronized positions" for the following axis (SynPosFA) and for the leading axes (SynPosLA), positions are defined for which the axis grouping is interpreted as *synchronous*. If the electronic gear is not in the synchronized state when the grouping is switched on, the following axis traverses to its defined synchronized position.

Variant 3:

The parameters correspond to those of variant 2 plus:

`Approach mode:`	The following modes can be used:	
	`"NTGT"`	Approach next tooth gap time-optimized
	`"NTGP"`	Approach next tooth gap path-optimized
	`"ACN"`	Traverse rotary axis in negative direction absolute
	`"ACP"`	Traverse rotary axis in positive direction absolute
	`"DCT"`	Time-optimized for programmed synchronous position
	`"DCP"`	Distance-optimized to the programmed synchronous position

Variant 3 only affects modulo following axes that are coupled to modulo leading axes. Time optimization takes account of velocity limits of the following axis.

More information

Description of the switch-in versions

Variant 1:

The positions of the leading axes and following axis at the instant the grouping is switched on are stored as "Synchronized positions". The "Synchronized positions" can be read with the system variable `$AA_EG_SYN`.

Variant 2:

If modulo axes are contained in the coupling group, their position values are modulo-reduced. This ensures that the next possible synchronized position is approached (so-called *relative synchronization*: e.g. the next tooth gap).

If following axis superimposition is not enabled for the following axis, it is not traversed to the synchronous position. Instead, the program stops at the EGONSYN block and self-clearing alarm 16771 is output until the release signal is set.

Variant 3:

The tooth distance (deg.) is calculated like this: 360 * Zi/Ni. If the following axis is stopped at the time of calling, path optimization returns responds identically to time optimization.

If the following axis is already in motion, NTGP will synchronize at the next tooth gap irrespective of the current velocity of the following axis. If the following axis is already in motion, NTGT will synchronize at the next tooth gap depending on the current velocity of the following axis. The axis is also decelerated, if necessary.

Curve tables

The following should be noted if a **curve table** is used for one of the leading axes:

Ni	The denominator of the coupling factor of the linear couplings must be set to 0.
Zi	For a denominator of 0, the numerator is interpreted as the number of the curve table to be used. **Note** The curve table must already have been defined at the instant of switch on.
LAi	The leading axis specified corresponds to the one specified for coupling via coupling factor (linear coupling).

Additional information about using curve tables and cascading electronic gears and their synchronization is provided in:

Reference

Function Manual Axes and Spindles

Response of the electronic gear for power on, RESET, operating mode change, block search

- **No** coupling is active after POWER ON.
- The status of active couplings is not affected by RESET or operating mode switchover.
- During block searches, commands for switching, deleting and defining the electronic gear are not executed or collected, but skipped.

System variables of the electronic gear

By means of the electronic gear's system variables, the part program can determine the current states of an EG axis grouping and react to them if required.

The system variables of the electronic gearbox are designated as follows:

$AA_EG_ ...

or

$VA_EG_ ...

4.16.2.3 Switching-in the electronic gearbox (EGOFS, EGOFC)

There are 3 different ways to switch-out an active EG axis group.

Programming

Variant 1:

Syntax	Meaning
`EGOFS(following axis)`	The electronic gear is deactivated. The following axis is braked to a standstill. This call triggers a preprocessing stop.

Variant 2:

Syntax	Meaning
`EGOFS(following axis,leading axis1, …,leading axis5)`	This command parameter setting made it possible to **selectively** remove the influence of the individual leading axes on the following axis' motion.

At least one leading axis must be specified. The influence of the specified leading axes on the slave is selectively inhibited. This call triggers a preprocessing stop. If the call still includes active leading axes, then the slave continues to operate under their influence. If the influence of all leading axes is excluded by this method, then the following axis is braked to a standstill.

Variant 3:

Syntax	Meaning
`EGOFC(following spindle1)`	The electronic gear is deactivated. The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation. This call triggers a preprocessing stop.

Note

This variant is only permitted for spindles.

4.16.2.4 Deleting the definition of an electronic gear (EGDEL)

An EG axis group must be switched-out before its definition can be deleted.

Programming

Syntax	Meaning
`EGDEL(following axis)`	The coupling definition of the axis group is deleted. Additional axis groups can be defined by means of `EGDEF` until the maximum number of simultaneously activated axis groups is reached. This call triggers a preprocessing stop.

4.16.2.5 Rotational feedrate (G95) / electronic gear (FPR)

The FPR command can be used to specify the following axis of an electronic gear as the axis, which determines the revolutional feedrate. Please note the following with respect to this command:

- The feedrate is determined by the setpoint velocity of the following axis of the electronic gear.
- The setpoint velocity is calculated from the speeds of the leading spindles and modulo axes (which are not path axes) and from their associated coupling factors.
- Speed parts of linear or non-modulo leading axes and overlaid movement of the following axis are not taken into account.

4.16.3 Synchronous spindle coupling (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC)

Synchronous spindle coupling enables speed-synchronous traversing of the following spindle (FS) and leading spindle (LS) of a synchronous spindle pair. With active coupling (synchronous mode), the following spindle follows the movements of the leading spindle according to the specified ratio.

The synchronism mode is adjustable. The following variants are available:

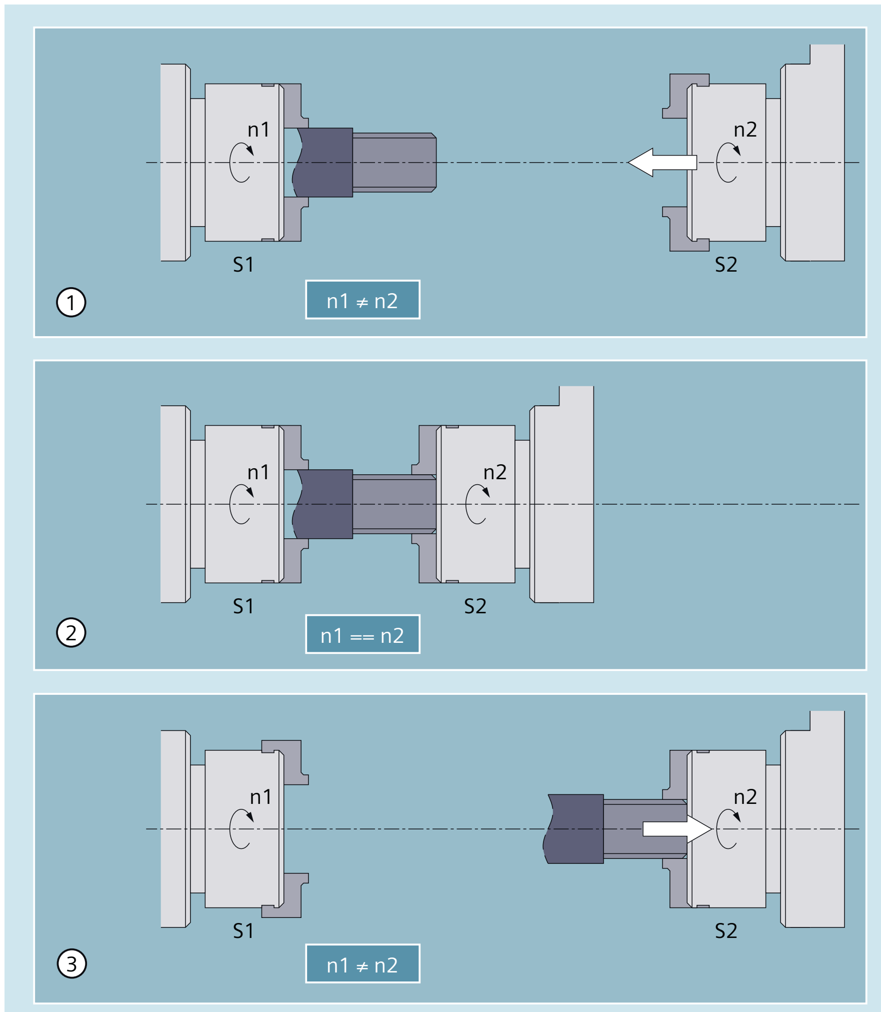
- Speed synchronism ($n_{FS} = n_{LS}$)
- Position synchronism ($\phi_{FS} = \phi_{LS}$)
- Position synchronism with angular offset ($\phi_{FS} = \phi_{LS} + \Delta\phi$)

The synchronous spindle pairs for each machine can be assigned a fixed configuration by means of channel-specific machine data or defined for specific applications via the part program. Up to 2 synchronous spindle pairs can be operated simultaneously on each NC channel.

Application

Typical applications are:

- Workpiece transfer on the fly, e.g. for machining on the rear side (transmission ratio: 1:1)

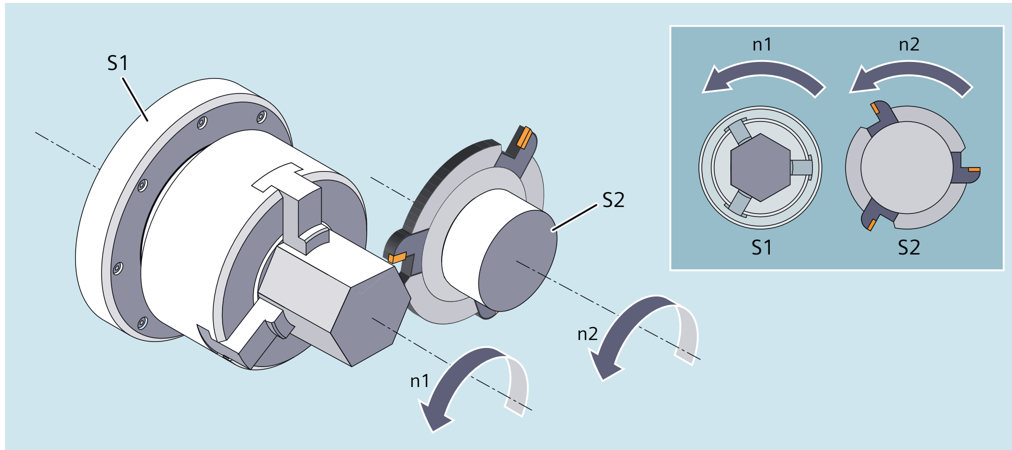


① The speeds n1 of spindle S1 and n2 of spindle S2 are different (n1 ≠ n2).
Synchronization is therefore required before workpiece transfer (transmission ratio: 1:1).

② After synchronization of the spindle speeds (n1 == n2), the workpiece is transferred.

③ After workpiece transfer, machining of the rear side can take place.

- Multi-edge machining (polygon turning), speed synchronism (transformation ratio: n_1:n_2)



Syntax

```
COUPDEF(<FS>,<LS>,<TRNom>,[<TRDenom>],<BlockChange>,<CouplingType>)
COUPON(<FS>[,<LS>,<POSFS>])
COUPONC(<FS>[,<LS>])
COUPOF(<FS>[,<LS>])
COUPOFS(<FS>[,<LS>])
COUPRES(<FS>[,<LS>])
COUPDEL(<FS>[,<LS>])
WAITC(<FS>,<BlockChange>,<LS>,<BlockChange>)
```

Meaning

COUPDEF	Define/change coupling on user-specific basis
COUPON	Activate coupling The following spindle synchronizes to the leading spindle based on the actual speed.
COUPONC	Activate the coupling and accept the spindle programming M3 S... or M4 S... A difference in speed for the following spindle is transferred immediately.
COUPOF	Deactivate coupling
COUPOFS	Deactivate coupling with stop of the following spindle
COUPRES	Reset coupling parameters to the configured values The values set in the machine and setting data are activated.
COUPDEL	Delete user-defined coupling
WAITC	Wait for synchronized run condition (NOC are increased to IPO during block changes)
<FS>	Identifier (spindle number) of the following spindle
<LS>	Identifier (spindle number) of the leading spindle **Note:** Specifying the leading spindle is only mandatory for COUPDEL. It is optional for the other commands.
Optional parameters:	

<table>
<tr><td><TRNom></td><td colspan="2">Numerator of the gear ratio</td></tr>
<tr><td rowspan="2"><TRDenom></td><td colspan="2">Denominator of the gear ratio</td></tr>
<tr><td colspan="2">The transmission ratio is the ratio between the following and leading spindle speeds:
n_{FS} / n_{LS}
Programming is performed by specifying the numerator and denominator of the transmission ratio:
<code>COUPDEF(...,...,<TRNom>,[<TRDenom>],...,..)</code>
Specifying the denominator is optional. The default value of 1.0 is set if nothing is specified.
Example:
Following spindle S2 and leading spindle S1, transformation ratio = 1/1
<code>COUPDEF(S2, S1, 1.0)</code>
Note:
The transformation ratio can also be changed on-the-fly (when the coupling is active and the spindles are rotating).</td></tr>
<tr><td rowspan="6"><BlockChange></td><td colspan="2">Block change behavior</td></tr>
<tr><td>"NOC"</td><td>Immediately (default setting)</td></tr>
<tr><td>"FINE"</td><td>On reaching "Synchronism fine"</td></tr>
<tr><td>"COARSE"</td><td>On reaching "Synchronism coarse"</td></tr>
<tr><td>"IPOSTOP"</td><td>When reaching IPOSTOP; i.e. after setpoint-based synchronism</td></tr>
<tr><td colspan="2">The block change behavior is effective modally.
Note:
The following abbreviated notation is permissible when programming the block change behavior: "NO" / "FI" / "CO" / "IP"</td></tr>
<tr><td rowspan="5"><CouplingType></td><td colspan="2">Coupling type: Coupling between FS and LS</td></tr>
<tr><td>"DV"</td><td>Setpoint linkage (default)</td></tr>
<tr><td>"AV"</td><td>Actual value coupling</td></tr>
<tr><td>"VV"</td><td>Speed coupling</td></tr>
<tr><td colspan="2">The coupling type is modal.
Note:
The coupling type may only be changed when the coupling is deactivated.</td></tr>
<tr><td rowspan="3"><POSFS></td><td colspan="2">Angular offset between the leading and following spindles referred to 0° position of the leading spindle in the positive direction of rotation.</td></tr>
<tr><td>Value range:</td><td>0°... 359.999°</td></tr>
<tr><td colspan="2">Note:
The angular offset can also be changed when the coupling is active.</td></tr>
</table>

Examples

Working with leading and following spindles

Program code	Comment
	Leading spindle = master spindle = spindle 1
	Following spindle = spindle 2

Program code	Comment
N05 M3 S3000 M2=4 S2=500	Leading spindle rotates at 3000 rpm, following spindle at 500 rpm.
N10 **COUPDEF(S2,S1,1,1,"NOC","Dv")**	Definition of the coupling (can also be configured).
...	
N70 SPCON	Bring leading spindle into closed-loop position control (setpoint coupling).
N75 SPCON(2)	Bring following spindle into closed-loop position control.
N80 **COUPON(S2,S1,45)**	On-the-fly coupling to offset position = 45 degrees.
...	
N200 FA[S2]=100	Positioning speed = 100 degrees/min
N205 SPOS[2]=IC(-90)	Traverse with 90 degrees overlay in negative direction.
N210 WAITC(S2,"Fine")	Wait for "fine" synchronism.
N212 G1 X... Y... F...	Machining
...	
N215 SPOS[2]=IC(180)	Traverse with 180 degrees overlay in the positive direction.
N220 G4 S50	Dwell time = 50 revolutions of the master spindle
N225 FA[S2]=0	Activate configured velocity (MD).
N230 SPOS[2]=IC(-7200)	20 revolutions. Move with configured velocity in the negative direction.
...	
N350 **COUPOF(S2,S1)**	Couple-out on-the-fly, S=S2=3000
N355 SPOSA[2]=0	Stop FS at zero degrees.
N360 G0 X0 Y0	
N365 WAITS(2)	Wait for spindle 2.
N370 M5	Stop FS.
N375 M30	

Programming a difference in speed

Program code	Comment
	Leading spindle = master spindle = spindle 1
	Following spindle = spindle 2
N01 M3 S500	Leading spindle rotates at 500 rpm.
N02 M2=3 S2=300	Following spindle rotates at 300 rpm.
...	
N10 G4 F1	Dwell time of master spindle.
N15 **COUPDEF (S2,S1,-1)**	Coupling factor with ratio -1:1
N20 **COUPON (S2,S1)**	Activate coupling. The speed of the following spindle results from the speed of the leading spindle and coupling factor.
...	
N26 M2=3 S2=100	Programming a difference in speed.

Examples of transfer of a movement for difference in speed

1. Activate coupling during previous programming of following spindle with COUPON

Program code	Comment
	Leading spindle = master spindle = spindle 1
	Following spindle = spindle 2
N05 M3 S100 M2=3 S2=200	Leading spindle rotates at 100 rpm, following spindle at 200 rpm.
N10 G4 F5	Dwell time = 5 seconds of master spindle
N15 **COUPDEF(S2,S1,1)**	Transformation ratio of FS to LS is 1.0 (default).
N20 **COUPON(S2,S1)**	On-the-fly coupling to the leading spindle.
N10 G4 F5	Following spindle rotates at 100 rpm.

2. Activate coupling during previous programming of following spindle with COUPONC

Program code	Comment
	Leading spindle = master spindle = spindle 1
	Following spindle = spindle 2
N05 M3 S100 M2=3 S2=200	Leading spindle rotates at 100 rpm, following spindle at 200 rpm.
N10 G4 F5	Dwell time = 5 seconds of master spindle
N15 **COUPDEF(S2,S1,1)**	Transformation ratio of FS to LS is 1.0 (default).
N20 **COUPONC(S2,S1)**	On-the-fly coupling to leading spindle and transfer previous speed to S2.
N10 G4 F5	S2 rotates at 100 rpm + 200 rpm = 300 rpm

3. Activate coupling with following spindle stationary with COUPON

Program code	Comment
	Leading spindle = master spindle = spindle 1
	Following spindle = spindle 2
N05 SPOS=10 SPOS[2]=20	Following spindle S2 in positioning mode.
N15 **COUPDEF(S2,S1,1)**	Transformation ratio of FS to LS is 1.0 (default).
N20 **COUPON(S2,S1)**	On-the-fly coupling to the leading spindle.
N10 G4 F1	Coupling is closed, S2 stops at 20 degrees.

4. Activate the coupling with following spindle stationary using COUPONC

Note

The following spindle is in the positioning or axis mode

If the following spindle is in positioning or axis mode before coupling, then the following spindle behaves the same for COUPON(<FS>,<LS>) and COUPONC(<FS>,<LS>).

Note

Leading spindle in the axis mode

If, prior to the coupling being defined, the leading spindle is in axis operation, the velocity limit value from machine data

MD32000 $MA_MAX_AX_VELO (maximum axis velocity) will still apply even after the coupling is activated.

To avoid this behavior, the axis must be switched to spindle mode (M3 S... or M4 S...) prior to the coupling being defined.

More information

Configured coupling

For the configured coupling, the LS and FS are defined via machine data. The configured spindles cannot be changed in the part program. The coupling can be parameterized in the part program using COUPDEF (on condition that no write protection is valid).

User-defined coupling

COUPDEF can be used to redefine or change a coupling in the part program. If a coupling is already active, it has to be deleted first with COUPDEL before a new coupling is defined.

Following spindle (FS) and leading spindle (LS)

The coupling is uniquely defined using the identifiers for the FS and LS. The identifiers must be programmed with every COUPDEF instruction. The other coupling parameters are modal and only have to be programmed if they change.

Position the following spindle

Even with activated synchronous spindle coupling, the FS can be positioned in the range ±180° independently of the LS.

- Spindle positioning of the FS with SPOS
  Example: `SPOS[2]=IC(-90)`
  For **more information** on SPOS, see Chapter "Positioning spindles (SPOS, SPOSA, M19, M70, WAITS) (Page 127)".

Differential speed

A speed difference results in speed control mode and active synchronous spindle coupling through signed overlay of an FS speed because of LS movement and an FS speed because of spindle programming:

- Synchronous spindle coupling with COUPONC
- `S<FS>=<speed>` [`M<FS>=<direction of rotation>`]

Note

The following conditions must be observed:

- Speed S... must also be reprogrammed with direction of rotation M3/M4.
- The overlay of a spindle speed (M<direction of rotation> S<FS>) through the LS movement with synchronous spindle coupling COUPONC only becomes effective if the overlay has been enabled.
- The dynamic responses of the leading spindle have to be restricted to such an extent that when overlaying is applied to the following spindle, its dynamics limit values are not exceeded.

Velocity, acceleration: FA, ACC, OVRA, VELOLIMA

Axial velocity and acceleration of a following spindle can be programmed with:

- `FA[SPI(S<n>)]` or `FA[S<n>]` (axial velocity)
- `ACC[SPI(S<n>)]` or `ACC[S<n>]` (axial acceleration)
- `OVRA[SPI(S<n>)]` and `OVRA[S<n>]` (axial override)
- `VELOLIMA[SPI(S<n>)]` and `VELOLIMA[S<n>]` (increase and reduction of axial velocity respectively)

With <n> = 1, 2, 3, ... (spindle number of the following spindles)

More information: Chapter "Feed control (Page 115)"

Note

A reduction or increase of the maximum axial jerk has no effect with spindles.

Programmable block change behavior WAITC

WAITC can be used to define block change behavior, for example after a change to coupling parameters or positioning actions, with a variety of synchronism conditions (coarse, fine, IPOSTOP). If no synchronism conditions are specified, the block change behavior specified in the COUPDEF definition will apply.

Examples:

- Wait for synchronism condition FINE to be fulfilled for following spindle S2 and COARSE to be fulfilled for following spindle S4:
  `WAITC(S2,"FINE",S4,"COARSE")`
- Wait for synchronism condition according to COUPDEF to be fulfilled: `WAITC( )`

System variables

- Current coupling status of following spindle
  The current coupling status of a following spindle can be read bit-coded via:
  `<Value> = $AA_COUP_ACT[<FS>]`

Bit	<value>	Meaning
-	0	No coupling active
2	4	Synchronous spindle coupling active

  All other values refer to the axis mode.
  If the spindle is a following spindle or several couplings, then the value of the coupling state of all couplings is returned as a total state.

- Current angular offset
  The current angular offset of the following spindle to the leading spindle can be read via:
  – $AA_COUP_OFFS[<FS>] (angular offset on the setpoint side)
  – $VA_COUP_OFFS[<FS>] (angular offset on the actual value side)

  Application example:
  Correction of the angular offset difference in the NC program after cancelling the follow-up mode:
  Angular offset difference = programmed angular offset - system variable

4.16.4 Generic coupling (CP...)

"Generic Coupling" is a general coupling function, combining all coupling characteristics of existing coupling types (coupled motion, master value coupling, electronic gearbox and synchronous spindle).

The function allows flexible programming:

- Users can select the coupling properties required for their applications (building block principle).
- Each coupling property can be programmed individually.
- The coupling properties of a defined coupling (e.g. coupling factor) can be changed.
- Later use of additional coupling properties is possible.
- The coordinate reference system of the following axis (basic coordinate system or machine coordinate system) is programmable.
- Certain coupling properties can also be programmed with synchronized actions.
  More information: Function Manual Synchronized Actions

Note

Previous coupling calls for coupled motion (TRAIL*), Master value coupling (LEAD*), Electronic Gearbox (EG*) and Synchronous spindle (COUP*) are supported via adaptive cycles.

Overview of all keywords and coupling characteristics

The following table gives an overview of all keywords of the generic coupling and the programmable coupling characteristics:

Keyword	Coupling characteristics / meaning	Syntax
`CPDEF`	Creation of a coupling module	`CPDEF=(<FAx>)`
`CPDEL`	Deletion of a coupling module	`CPDEL=(<FAx>)`
`CPLA`	Definition of a leading axis	`CPLA[<FAx>]=(<LAx>)`

`CPLDEF`	Definition of a leading axis and creation of a coupling module
	(also possible with `CPDEF` + `CPLA`)
		`CPLDEF[<FAx>]=(<LAx>)`
		or
		`CPDEF=(<FAx>) CPLA[<FAx>]=(<LAx>)`

`CPLDEL`	Deletion of a leading axis of a coupling module
	(also possible with `CPDEF` + `CPLA`)
		`CPLDEL[<FAx>]=(<LAx>)`
		or
		`CPDEL=(<FAx>) CPLA[<FAx>]=(<LAx>)`

`CPON`	Switching on a coupling module	`CPON=(<FAx>)`
`CPOF`	Switching off a coupling module	`CPOF=(<FAx>)`
`CPLON`	Switching on a leading axis of a coupling module	`CPLON[<FAx>]=<LAx>`
`CPLOF`	Switching off a leading axis of a coupling module	`CPLOF[<FAx>]=<LAx>`
`CPLNUM`	Numerator of the coupling factor	`CPLNUM[FAx,LAx]=<value>`
`CPLDEN`	Denominator of the coupling factor	`CPLDEN[FAx,LAx]=<value>`
`CPLCTID`	Number of the curve table	`CPLCTID[FAx,LAx]=<value>`

`CPLSETVAL`	Coupling reference	`CPLSETVAL[FAx,LAx]="<coupling reference>"`
		`"<coupling reference>":`
		`"CMDPOS"` Setpoint value coupling
		`"CMDVEL"` Speed coupling
		`"ACTPOS"` Actual value coupling

`CPFRS`	Coordinate reference system	`CPFRS[FAx]="<coordinate reference>"`
		`"<coordinate reference>":`
		`"BCS"` Basic Coordinate System
		`"MCS"` Machine Coordinate System

<table>
<tr><th>Keyword</th><th>Coupling characteristics / meaning</th><th colspan="3">Syntax</th></tr>
<tr><td rowspan="5">CPBC</td><td rowspan="5">Block change criterion</td><td colspan="3">CPBC[FAx]="<block change criterion>"</td></tr>
<tr><td rowspan="4">"<block change criterion>":</td><td>"NOC"</td><td>Block change is performed irrespective of the coupling status.</td></tr>
<tr><td>"IPOSTOP"</td><td>Block change is performed with setpoint synchronism.</td></tr>
<tr><td>"COARSE"</td><td>Block change is performed with actual value synchronism “coarse”.</td></tr>
<tr><td>"FINE"</td><td>Block change is performed with actual value synchronism "fine".</td></tr>
<tr><td colspan="5"></td></tr>
<tr><td>CPFPOS + CPON</td><td>Synchronized position of the following axis when switching on</td><td colspan="3">CPON=FAx CPFPOS[FAx]=<value></td></tr>
<tr><td>CPLPOS + CPON</td><td>Synchronized position of the leading axis when switching on</td><td colspan="3">CPLPOS[FAx,LAx]=<value></td></tr>
</table>

Keyword	Coupling characteristics / meaning	Syntax		
CPFMSON	Synchronization mode	CPFMSON[FAx]="<synchronization mode>"		
		"<synchronization mode>":	"CFAST"	The coupling is closed time-optimized.
			"CCOARSE"	The coupling is only closed when the following axis position, required according to the coupling rule, is in the range of the current following axis position.
			"NTGT"	The next tooth gap is approached time-optimized.
			"NTGP"	The next tooth gap is approached path-optimized.
			"NRGT"	The next segment is approached in a time-optimized manner, in accordance with the ratio of the number of gears to the number of teeth.
			"NRGP"	The next segment is approached in a path-optimized manner, in accordance with the ratio of the number of gears to the number of teeth.
			"ACN"	For rotary axes only! The rotary axis traverses to the synchronized position in the negative axis direction. Synchronization is realized immediately.
			"ACP"	For rotary axes only! The rotary axis traverses to the synchronized position in the positive axis direction. Synchronization is realized immediately.
			"DCT"	For rotary axes only! The rotary axis traverses to the programmed synchronized position time-optimized. Synchronization is realized immediately.
			"DCP"	For rotary axes only! The rotary axis traverses to the programmed synchronized position path-optimized. Synchronization is realized immediately.

<table>
<tr><th>Keyword</th><th>Coupling characteristics / meaning</th><th colspan="3">Syntax</th></tr>
<tr><td rowspan="4"><code>CPFMON</code></td><td rowspan="4">Behavior of the following axis when switching on</td><td colspan="3"><code>CPFMON[FAx]= "<switch-on behavior>"</code></td></tr>
<tr><td rowspan="3"><code>"<switch-on behavior>":</code></td><td><code>"STOP"</code></td><td>For spindles only!
An active motion of the following spindle is stopped before switch-on.</td></tr>
<tr><td><code>"CONT"</code></td><td>For spindles and main traverse axes only!
The current motion of the following axis/spindle is taken over into the coupling as start motion.</td></tr>
<tr><td><code>"ADD"</code></td><td>For spindles only!
The motion components of the coupling operate in addition to the currently overlaid motion, i.e. the current motion of the following axis/spindle is retained as overlaid motion.</td></tr>
<tr><td rowspan="3"><code>CPFMOF</code></td><td rowspan="3">Behavior of the following axis at complete switch-off</td><td colspan="3"><code>CPFMOF[FAx]="<switch-off behavior>"</code></td></tr>
<tr><td rowspan="2"><code>"<switch-off behavior>":</code></td><td><code>"STOP"</code></td><td>Stop of a following axis/ spindle.
An active overlaid motion is also braked to standstill. The coupling is then opened</td></tr>
<tr><td><code>"CONT"</code></td><td>For spindles and main traverse axes only!
The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation.</td></tr>
<tr><td><code>CPFPOS + CPOF</code></td><td>Switch-off position of the following axis when switching off</td><td colspan="3"><code>CPOF=(FAx) CPFPOS[FAx]=<value></code></td></tr>
<tr><td colspan="5"></td></tr>
</table>

Keyword	Coupling characteristics / meaning	Syntax		
`CPMRESET`	Coupling behavior for RESET	`CPMRESET[FAx]="<Reset behavior>"`		
		`"<reset behavior>":`	`"NONE"`	The current state of the coupling is retained.
			`"ON"`	When the appropriate coupling module is created, the coupling is switched on. All defined leading axis relationships are activated. This is also performed when all or parts of these leading axis relationships are active, i.e. resynchronization is performed even with a completely activated coupling.
			`"OF"`	An active overlaid movement is also braked to standstill. The coupling is then deactivated. When the relevant coupling module was created without an explicit definition (CPDEF), the coupling module is deleted. Otherwise it is retained, i.e. it can still be used.
			`"OFC"`	Possible only in spindles! The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation. The coupling is switched off. When the relevant coupling module was created without an explicit definition (CPDEF), the coupling module is deleted. Otherwise it is retained, i.e. it can still be used.
			`"DEL"`	An active overlaid motion is also braked to standstill. The coupling is then deactivated and then deleted.
			`"DELC"`	Possible only in spindles! The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation. The coupling is deactivated and then deleted.

<table>
<tr><th>Keyword</th><th>Coupling characteristics / meaning</th><th colspan="3">Syntax</th></tr>
<tr><td rowspan="6">CPMSTART</td><td rowspan="6">Coupling behavior at part program start</td><td colspan="3">CPMSTART[FAx]="<start behavior>"</td></tr>
<tr><td rowspan="5">"<start behavior>":</td><td>"NONE"</td><td>The current state of the coupling is retained.</td></tr>
<tr><td>"ON"</td><td>Coupling switched-on. All defined leading axis relationships are activated. This is also performed when all or parts of these leading axis relationships are active, i.e. resynchronization is performed even with a completely activated coupling.</td></tr>
<tr><td>"OF"</td><td>The coupling is switched off. When the relevant coupling module was created without an explicit definition (CPDEF), the coupling module is deleted. Otherwise it is retained, i.e. it can still be used.</td></tr>
<tr><td>"DEL"</td><td>The coupling is deactivated and then deleted.</td></tr>
<tr></tr>
<tr><td rowspan="2">CPMPRT</td><td rowspan="2">Coupling response at part program start under block search run via program test</td><td colspan="3">CPMPRT[FAx]="<start behavior>"</td></tr>
<tr><td>"<start behavior>":</td><td colspan="2">see CPMSTART</td></tr>
<tr><td colspan="5"></td></tr>
<tr><td>CPLINTR</td><td>Offset value of the input value of a leading axis</td><td colspan="3">CPLINTR[FAx,LAx]=<value></td></tr>
<tr><td>CPLINSC</td><td>Scaling factor of the input value of a leading axis</td><td colspan="3">CPLINSC[FAx,LAx]=<value></td></tr>
<tr><td>CPLOUTTR</td><td>Offset value for the output value of a coupling</td><td colspan="3">CPLOUTTR[FAx,LAx]=<value></td></tr>
<tr><td>CPLOUTSC</td><td>Scaling factor for the output value of a coupling</td><td colspan="3">CPLOUTSC[FAx,LAx]=<value></td></tr>
<tr><td colspan="5"></td></tr>
<tr><td>CPSYNCOP</td><td>Threshold value of position synchronism "Coarse"</td><td colspan="3">CPSYNCOP[FAx]=<value></td></tr>
<tr><td>CPSYNFIP</td><td>Threshold value of position synchronism "Fine"</td><td colspan="3">CPSYNFIP[FAx]=<value></td></tr>
<tr><td>CPSYNCOP2</td><td>Second threshold value for the "Coarse" position synchronism</td><td colspan="3">CPSYNCOP2[FAx]=<value></td></tr>
<tr><td>CPSYNFIP2</td><td>Second threshold value for the "Fine" position synchronism</td><td colspan="3">CPSYNFIP2[FAx]=<value></td></tr>
<tr><td>CPSYNCOV</td><td>Threshold value of velocity synchronism "Coarse"</td><td colspan="3">CPSYNCOV[FAx]=<value></td></tr>
<tr><td>CPSYNFIV</td><td>Threshold value of velocity synchronism "Fine"</td><td colspan="3">CPSYNFIV[FAx]=<value></td></tr>
<tr><td colspan="5"></td></tr>
</table>

Keyword	Coupling characteristics / meaning	Syntax		
`CPMBRAKE`	Response of the following axis to certain stop signals and stop commands	`CPMBRAKE[FAx]=<bit-coded value>`		
`CPMVDI`	Response of the following axis to certain NC/PLC interface signals	`CPMVDI[FAx]=<bit-coded value>`		
`CPMALARM`	Suppression of special coupling-related alarm outputs	`CPMALARM[FAx]=<bit-coded value>`		
`CPSETTYPE`	Coupling type	`CPSETTYPE[FAx]="<coupling type>"`		
		`"<coupling type>":`	`"CP"`	Freely programmable
			`"TRAIL"`	Coupling type "Coupled motion"
			`"LEAD"`	Coupling type "Master Value Coupling"
			`"EG"`	Coupling type "Electronic gearbox"
			`"COUP"`	Coupling type "Synchronous spindle"

FAx: Following axis/spindle
LAx: Leading axis/spindle

Note

Coupling characteristics, which are not explicitly programmed (in part program of synchronized actions), become effective with their default settings.

Depending on the settings of the keyword `CPSETTYPE` instead of the default settings (`CPSETTYPE="CP"`) preset coupling characteristics can become effective.

More information

For detailed information on generic couplings, see:

- Function Manual Axes and Spindles

4.16.5 Tangential control

4.16.5.1 Introduction and overview

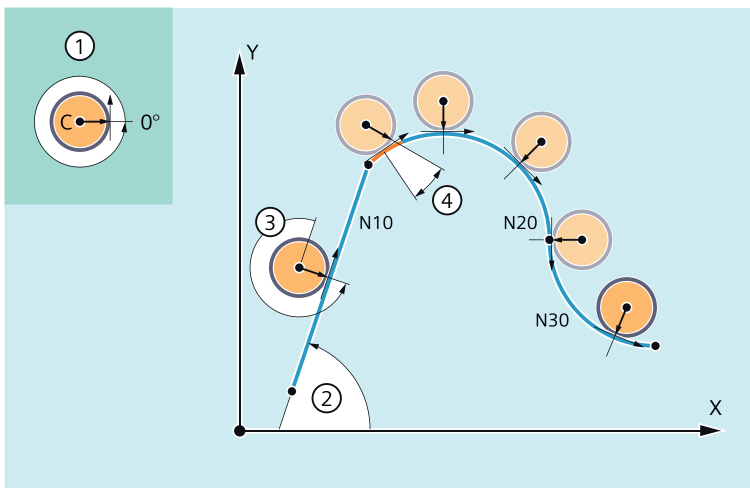
Function

Using the "Tangential control" coupling function, a rotary axis is coupled as following axis to two geometry axes as leading axes, so that the alignment of the following axis is a function of the path tangent of the leading axes.

If the contour described by the leading axes has a discontinuous block transition or corner, then one of the following corner behaviors can be selected:

- The dynamic response of the rotary axis has no effect on the leading axes.
- The dynamic response of the rotary axis is considered in the path planning of the leading axes together with programmable parameters "rounding clearance" and "angular tolerance".
- The leading axes are stopped before the corner, and in an automatically generated intermediate block, the following axis is realigned.
  The corner angle is detected depending on the value of machine data MD37400 $MA_EPS_TLIFT_TANG_STEP (tangential angle for corner detection).

The following figure illustrates the mode of operation of the tangential control:



①	Initial position and positive direction of rotation of the following/rotary axes C
②	Angle of the path tangent in machining plane X/Y
③	Offset angle = 270° or -90° The rotary axis is tracked with a programmable offset angle of 270° with respect to the path tangent.
④	Rounding clearance and present angular deviation At the block transition from N10 to N20, the contour has a discontinuous transition or corner. As a result of a dynamic response that is too low, the following axis cannot follow the path tangent over part of the path (orange). However, the dynamic response of the following axis is sufficient so that it can precisely follow circular blocks N20 and N30 further along the contour.

Application

Typical applications of tangential control are:

- Tangential positioning a rotatable tool during nibbling
- Tracking the workpiece alignment for a belt saw
- Positioning of a dressing tool on a grinding wheel
- Positioning of a cutting wheel for glass or paper working
- Tangential feed of a wire for 5-axis welding

Programming

The tangential coupling can be defined, activated, deactivated and deleted in the NC program:

- Defining coupling (TANG) (Page 834)
- Activating intermediate block generation (TLIFT) (Page 835)
- Activating the coupling (TANGON) (Page 836)
- Deactivating the coupling (TANGOF) (Page 838)
- Deleting a coupling (TANGDEL) (Page 838)

4.16.5.2 Defining coupling (TANG)

Via the predefined procedure TANG(...), a tangential coupling between a rotary axis is defined as the following axis and two geometry axes as the leading axes. The following axis is continuously aligned with the path tangent of the leading axes.

Note

Coupling factor

A coupling factor of 1 does not have to be programmed explicitly.
The direction of the tangential axis is rotated using the coupling factor of -1.

Syntax

```
TANG(<FAx>, <LAx1>, <LAx2>, <CoupFac>, <CoordSys>, <OptMode>)
```

Meaning

`TANG(...)`	Define tangential coupling		
`<FAx>`	Axis name of the following axis (rotary axis)		
	Data type:	AXIS	
	Value range:	Channel axis names	
`<LAx1>` `<LAx2>`	Axis names of the leading axes (geometry axes) [1)]		
	Data type:	AXIS	
	Value range:	Geometry axis names of the channel	
`<CoupFac>`	Factor n of the angle change of the following axis for changing the path tangent of the leading axes: $\text{Angle change}_{\text{following axis}} = \text{angle change}_{\text{path tangent}} * n$		
	Data type:	REAL	
	Default value:	1.0	
`<CoordSys>`	Active coordinate system [2)]		
	Data type:	CHAR	
	Value:	`"B"`	Basic coordinate system (default value)
		`"W"`	Workpiece coordinate system (not available)

<table>
<tr><td rowspan="4"><code><OptMode></code></td><td colspan="3">Optimization mode</td></tr>
<tr><td>Data type:</td><td colspan="2">CHAR</td></tr>
<tr><td rowspan="2">Value:</td><td><code>"S"</code></td><td>Standard (default value)
The dynamic response of the rotary axis has no effect on the leading axes. If the dynamic response of the rotary axis is greater than required for tracking, this method is sufficiently precise. If the dynamic response of the rotary axis is not great enough to follow the change in the path tangent, the orientation of the rotary axis will deviate from the target orientation along an undefined rounding clearance.</td></tr>
<tr><td><code>"P"</code></td><td>The dynamic response of the rotary axis is considered in the path planning of the leading axes.
For this purpose, on activation of the tangential coupling with TANGON(), two additional parameters must be specified:
• Rounding clearance
• Angular tolerance
See Section "Activating the coupling (TANGON) (Page 836)"
Note
With kinematic transformations, we recommend using optimization method "P."</td></tr>
<tr><td colspan="4">Note
Default values do not have to be programmed explicitly.</td></tr>
<tr><td colspan="4">1) Note
As the leading axes for tangential coupling, the geometry axes must be used that travel along the programmed path in the machine coordinate system (MCS) with reference to the initial position of the machine. For example, if swivel cycle CYCLE800 is used on a milling machine with a swivel head, depending on how the cycle is configured, interpolation will be performed in the WCS, e.g. with the geometry axes X and Y. The tangential coupling, however, must be defined with the geometry axes as the leading axes, which travel along the programmed path in the MCS. For this purpose, the geometry axes in the non-swiveled condition of the machine must be used as the leading axes.</td></tr>
<tr><td colspan="4">2) Note
The basic coordinate system (BCS) must not be rotated with respect to the MCS. For example, if the BCS is rotated with the ROT command or with the swivel cycle CYCLE800, the tangential control is no longer correct.</td></tr>
</table>

4.16.5.3 Activating intermediate block generation (TLIFT)

If the tangent change of the following axis at any position along the programmed path of the leading axes exceeds the limit parameterized in machine data MD37400 $MA_EPS_TLIFT_TANG_STEP, further path planning will depend on the set behavior at corners. Without use of the predefined procedure TLIFT(...), the path is traversed in accordance with the rounding behavior programmed in connection with TANG(...) (Page 834) and TANGON(...) (Page 836).

Activating intermediate block generation

If TLIFT(...) is programmed after TANG(...), an intermediate block automatically generated by the control is inserted at this point when a corner is detected during preprocessing.

When the program is executed, the leading axes are stopped when the intermediate block is reached. In the intermediate block, the following axis is rotated with maximum axis dynamics toward the path tangent of the following block. The leading axes are then traversed further on the programmed path.

Deactivating intermediate block generation

To deactivate intermediate block generation, the tangential coupling must be defined again using TANG(...), but without subsequent activation of intermediate block generation by means of TLIFT(...).

Syntax

```
TLIFT(<FAx>)
```

Meaning

`TLIFT(...)`	Activate corner detection with intermediate block calculation	
`<FAx>`	Axis name of the following axis (rotary axis)	
	Data type:	AXIS
	Value range:	Channel axis names

Speed of rotation of the following axis

Path axis

If the following axis had already been traversed as a path axis before tangential coupling was activated, the rotational movement is performed in the intermediate block as a path axis.

If you specify the reference radius with `FGREF[<Ax>]=0.001`, the rotational movement will be performed with the parameterized maximum axis velocity:

MD32000 $MA_MAX_AX_VELO[<following axis>]

Positioning axis

If the following axis had not yet been traversed as a path axis before tangential coupling was activated, the rotation is performed in the intermediate block as a positioning axis.

The rotational movement is performed with the parameterized positioning axis velocity:

MD32060 $MA_POS_AX_VELO[<following axis>]

4.16.5.4 Activating the coupling (TANGON)

Via the predefined procedure TANGON(...), a tangential coupling previously defined with TANG(...) (Page 834) is activated. The following axis is then continuously aligned with the path tangent during subsequent travel.

Angle of the following axis

The angle of the following axis with respect to the path tangent depends on the transformation ratio specified in TANG(...), the offset angle parameterized in the machine data MD37402 $MA_TANG_OFFSET, and the offset angle specified for TANGON(...), which is applied additively.

Optimization "P"

If the value "P" was specified as the optimization parameter in the definition of the tangential coupling (TANG(...)), the parameter "rounding clearance" and optionally the parameter "angular tolerance" must be set when coupling is activated.

If the value 0 is specified as the angular tolerance, only the parameter "rounding clearance" will be active.

If a value greater than 0 is specified as the angular tolerance, the active rounding clearance results from the minimum of the parameterized rounding clearance and the rounding clearance based on the parameterized angular tolerance.

If the dynamic response of the following axis is not sufficient to follow the parameterized conditions, the path velocity of the leading axes will be reduced accordingly.

Syntax

```
TANGON(<FAx>, <OffsetAngle>, <MaxRoundingPath>, <MaxAngTol>)
```

Meaning

`TANGON(...)`	Activate tangential coupling	
`<FAx>`	Axis name of the following axis (rotary axis)	
	Data type:	AXIS
	Value range:	Channel axis names
`<OffsetAngle>`	Offset angle of following axis with respect to the path tangent The reference point is the zero point of the rotary axis.	
	Data type:	REAL
`<MaxRoundingPath>`	Maximum permissible rounding clearance If the rounding clearance is increased due to the dynamic conditions, the path velocity of the leading axes is reduced.	
	Data type:	REAL
`<MaxAngTol>`	Maximum permissible tolerance with respect to the specified angle between the following axis zero setting and the path tangent	
	Data type:	REAL

4.16.5.5 Deactivating the coupling (TANGOF)

Via the predefined procedure TANGOF(...), a tangential coupling defined with TANG(...) (Page 834) and activated with TANGON(...) (Page 836) is deactivated. The following axis is then no longer aligned with the path tangent of the leading axis. However, the coupling of the following axis to the leading axes is retained even after deactivation, which prevents the following functions, for example:

- Plane change
- Geometry axis switchover
- Definition of a new tangential coupling for the following axis

Final cancellation of the connection of the coupling of the following axis to the leading axes is not completed until the coupling has been deleted with TANGDEL(...) (Page 838).

Programming

```
TANGOF(<FAx>)
```

Meaning

`TANGOF(...)`	Deactivate a tangential coupling	
`<FAx>`	Axis name of the following axis (rotary axis)	
	Data type:	AXIS
	Value range:	Channel axis names

4.16.5.6 Deleting a coupling (TANGDEL)

A tangential coupling defined with TANG(...) (Page 834) will be retained even after deactivation of the tangential coupling with TANGOF(...) (Page 838). The existing tangential coupling then continues to prevent, for example, the following functions:

- Plane change
- Geometry axis switchover
- Definition of a new tangential coupling for the following axis

With the predefined procedure TANGDEL(...), the existing tangential coupling is deleted after the tangential coupling has been deactivated with TANGOF(...).

Syntax

```
TANGDEL(<FAx>)
```

Meaning

`TANGDEL(...)`	Delete a tangential coupling defined with TANG()	
	Effective:	Non-modal

<FAx>	Axis name of the following axis whose tangential coupling is to be deleted	
	Data type:	AXIS
	Value range:	Channel axis names

Examples

Leading axis change

Before a new tangential coupling can be defined with another leading axis for the following axis, the existing tangential coupling must first be deleted.

Program code	Comment
N10 **TANG**(A, X, Y, 1)	; Define tangential coupling for following axis A: A to X and Y
N20 TANGON(A)	; Activate tangential coupling for following axis A
N30 X10 Y20	
...	
N80 TANGOF(A)	; Deactivate tangential coupling for following axis A
N90 **TANGDEL**(A)	; Delete tangential coupling for following axis A
...	
N120 **TANG**(A, X, Z)	; Define new tangential coupling for following axis A
N130 TANGON(A)	; Activate new tangential coupling for following axis A
...	

Geometry axis switchover

Before geometry axis switchover can be performed for an existing coupling, the coupling must first be deleted.

Program code	Comment
N10 GEOAX(2,Y1)	; 2nd geometry axis = machine axis Y1
N20 TANG(A, X, Y)	; Define tangential coupling for following axis A
N30 TANGON(A, 90)	; Activate tangential coupling for following axis A
N40 G2 F8000 X0 Y0 I0 J50	; Motion block
N50 TANGOF(A)	; Deactivate tangential coupling for following axis A
N60 **TANGDEL**(A)	; Delete tangential coupling for following axis A
N70 **GEOAX**(2, Y2)	; 2nd geometry axis = machine axis Y2
N80 TANG(A, X, Y)	; Define new tangential coupling for following axis A
N90 TANGON(A, 90)	; Activate new tangential coupling for following axis A
...	

4.16.6 Main/sub-coupling (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS)

A main/sub-coupling is a speed setpoint coupling between a main axis and any number of sub-axes - performed at the position controller level - with and without torque equalization control.

The main/sub-coupling is used, for example:

- For power amplification in mechanically coupled drives
- For compensation of gear and tooth backlash by applying a pre-tensioning torque

A main/sub-coupling can be static or temporary, depending on the setting in the machine data MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[<Sub-axis>]:

- MD37262 = 1
  The coupling is static, i.e. it is permanently switched on and cannot be influenced by NC commands in the part program.
- MD37262 = 0
  The coupling is temporary and can be dynamically switched on/off and reconfigured by means of NC commands in the part program.
  A dynamic change of the assignment of main axes and sub-axes is retained even after a change of operating mode, reset and part program end. The assignment configured in the machine data only becomes effective again after a new control run-up (NC reset).

By writing the machine data MD37262 in the part program or synchronous action with subsequent NEWCONF warm restart, a static coupling can also be switched off/on at a later time. This is relevant, for example, if the actual value for the sub-axis is to be synchronized by PRESETON to the same value of the main axis.

Syntax

Dynamic switching on/off:

```
MASLON(<SubAx_1>,<SubAx_2>,...)
MASLOF(<SubAx_1>,<SubAx_2>,...)
MASLOFS(<SubAx_1>,<SubAx_2>,...)
```

Dynamic configuring:

```
MASLDEF(<SubAx_1>,<SubAx_2>,...,<MainAx>)
MASLDEL(<SubAx_1>,<SubAx_2>,...)
```

Meaning

MASLON	Switch on temporary main/sub coupling
MASLOF	Disconnect active main/sub-coupling
MASLOFS	Disconnect main/sub-coupling and brake sub-spindles automatically
MASLDEF	Creating/changing a main/sub-coupling group from the part program
MASLDEL	Disconnect main/sub-coupling and delete definition of coupling group
<SubAx_1>, ...	Sub-axis 1 ... n
<MainAx>	Main axis

Example

With a static main/sub-coupling, the actual value of the sub-axis is set to the value of the main axis with PRESETON.

Program code	Comment
`$MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=0`	; Switch off static coupling of the sub-axis.
`NEWCONF`	; Activate machine data change.
`STOPRE`	
`MASLOF(Y1)`	; Deactivate temporary coupling.
`PRESETON(AX2,$VA_IM(M_AX))`	; Actual value of the sub-axis = actual value of the main axis
`$MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=1`	; Switch on static coupling of the sub-axis.
`NEWCONF`	; Activate machine data change.

More information

Coupling state

The actual coupling state of a sub-axis can be read in the part program and synchronized action using the following system variable:

$AA_MASL_STAT[<sub-axis>]

Value	Meaning
0	Coupling of the sub-axis is not active. - OR - The specified axis is not a sub-axis
> 0	Coupling is active. <Value> == machine axis number of the master axis

Positioning mode

For axes and spindles in the positioning mode, the coupling is only closed and opened at standstill.

Coupling behavior for spindles in speed control mode

For spindles in the speed control mode, the coupling behavior of MASLON, MASLOF, MASLOFS and MASLDEL are specified explicitly via the following machine data:

MD37263 $MA_MS_SPIND_COUPLING_MODE

For the default setting MD37263 = 0, the sub-axes are coupled-in and coupled-out only when the axes involved are at standstill. MASLOFS corresponds to MASLOF.

For MD37263 = 1, the coupling instruction is immediately executed and therefore also the motion. For MASLON the coupling is immediately closed and for MASLOFS or MASLOF immediately opened. With MASLOF, the sub-spindles rotating at this instant keep their speeds until a new speed is programmed. However, with MASLOFS, they are braked automatically.

Preprocessing stop

For MASLOF/MASLOFS, the implicit preprocessing stop is not required. Because of the missing preprocessing stop, the $P system variables for the sub-axes do not provide updated values until next programming.

Preset actual value

For the sub-axis, the actual value can be synchronized to the same value of the main axis using PRESETON. For this purpose, permanent main/sub-coupling must be briefly switched off to set the actual value of the non-referenced sub-axis to the value of the main axis by means of a control run-up. Then the coupling is permanently re-established.

Permanent main/sub-coupling is activated with the following MD setting:

MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE = 1

It has no effect on the NC commands of temporary coupling.

4.17 Synchronized actions

4.17.1 Definition of a synchronized action

A synchronous action is defined in a block of a part program. Any further commands that are not part of the synchronous action, may not be programmed within this block.

A synchronous action consists of the following components:

Validity, ID no. (optional)	Condition part (optional)			Action part with condition fulfilled			Action part with condition unfulfilled (optional)		
	Frequency	G command (optional)	Condition	Keyword	G command (optional)	Actions	Keyword	G command (optional)	Actions
--- 1) ID=<no.> IDS=<no.>	--- 1) WHENEVER FROM WHEN EVERY	G...	Logical Expression	DO	G...	Action 1 ... Action n	ELSE	G...	Action 1 ... Action n

1) Not programmed

Syntax

```
DO <action 1> ... <action n>
<frequency> [<G function>] <condition> DO <action 1> ... <action n>
ID=<No> <frequency> [<G function>] <condition> DO <action 1> ...
<action n>
IDS=<No> <frequency> [<G function>] <condition> DO <action 1> ...
<action n>
```

```
IDS=<no.> <frequency> [<G function>] <condition> DO <action 1...n>
ELSE <action 1...n>
```

Further information

Function Manual Synchronous Actions

4.18 Grinding

4.18.1 Activate/deactivate grinding-specific tool monitoring (TMON, TMOF)

With the predefined procedures TMON(...) and TMOF(...), the grinding-specific tool monitoring is activated or deactivated (geometry and speed monitoring).

Requirement

The tool-specific parameters $TC_TPG1 to $TC_TPG9 must be set.

Syntax

```
TMON(<TNo>)
...
TMOF(<TNo>)
```

Meaning

TMON(...):	Activate grinding-specific tool monitoring The command must be programmed in the channel in which the grinding-specific tool monitoring is to be activated.
TMOF(...):	Deactivate grinding-specific tool monitoring The command must be programmed in the channel in which the grinding-specific tool monitoring is to be deactivated.
<TNo>:	T number **Note:** Only required if the monitoring is to be switched on or off for an inactive grinding wheel rather than the active grinding wheel that is currently in use.
TMOF(0):	Deactivate monitoring for all tools

4.19 Extended stop and retract (ESR)

The extended stop and retract function - subsequently called ESR - offers the possibility of flexibly responding when a fault situation occurs as a function of the process:

- **Extended stop**
  Assuming that the specific fault situation permits it, all of the axes, enabled for extended stopping, are stopped in an orderly way.
- **Retraction**
  The tool currently in use is retracted from the workpiece as quickly as possible.
- **Generator operation (SINAMICS drive function "Vdc control")**
  If a parameterizable value of the DC-link voltage is fallen below, e.g. because the line voltage fails, the electrical energy required for retraction is generated by recovering the braking energy of the drive intended for this purpose (generator operation).

Trigger sources

General sources (NC-external/global or mode group-/channel-specific):

- Digital inputs (e.g. on NCU module) or the control-internal digital output image that can be read back ($A_IN, $A_OUT)
- Channel state ($AC_STAT)
- VDI signals ($A_DBB)
- Group messages of a number of alarms ($AC_ALARM_STAT)

Axial sources

- Emergency retraction threshold of the following axis (synchronism of electronic coupling, $VA_EG_SYNCDIFF[<following axis>])
- Drive: DC-link warning threshold (imminent undervoltage), $AA_ESR_STAT[<axis>]
- Drive: Generator minimum speed threshold (no further regenerative rotation energy available), $AA_ESR_STAT[<axis>].

Gating logic of the static synchronized actions: Source/response link

The static synchronized actions' flexible gating possibilities are used to trigger specific reactions relatively quickly according to the sources.

Linking all relevant sources using static synchronized actions is the responsibility of the user. They can selectively evaluate the source system variables as a whole or by means of bit masks, and then make a logic operation with their desired reactions. The static synchronized actions are effective in all operating modes.

Activation

Function enable

The functions generator operation, shutdown, retraction are released by setting the corresponding control signal $AA_ESR_ENABLE. This control signal can be changed by synchronized actions.

Function triggering

ESR is triggered jointly for all enabled axes by setting the system variable $AC_ESR_TRIGGER.

Generator operation is "automatically" activated in the drive when an imminent DC-link undervoltage is detected.

Drive-independent stopping and/or retraction become active when a communication failure (between the NC and drive) is detected and when a DC-link undervoltage is detected in the drive (configuration and enable required).

Drive-independent stopping and/or retraction can also be triggered by the NC by setting the appropriate control signal $AN_ESR_TRIGGER (broadcast command to all drives).

Further information

Function Manual Axes and Spindles

4.19.1 NC-controlled ESR

4.19.1.1 NC-controlled retraction (POLF, POLFA, POLFMASK, POLFMLIN)

Certain initial conditions are required for NC-controlled retraction. When these requirements have been satisfied, then the rapid lift (LIFTFAST) configured for retraction axis(axes) in the channel is activated by setting the system variable $AC_ESR_TRIGGER (or $AA_ESR_TRIGGER for single axes).

Syntax

```
POLF(<axis>)=<position>
POLFA(<axis>,<type>,<position>)
POLFMASK(<axis_1>,<axis_2>,...)
POLFMLIN(<axis_1>,<axis_2>,...)
```

The following abbreviated forms are permitted for `POLFA`:

```
POLFA(<axis>,<type>) ; Abbreviated form for single axis retraction
POLFA(axis,0/1/2) ; Quick deactivation or activation
POLFA(axis,0,$AA_POLFA[axis]) ; Causes a preprocessing stop
POLFA(axis,0) ; Does not cause a preprocessing stop
```

Meaning

POLF:	Address for specifying the target position of the retraction axis POLF is modal.			
	<axis>:	Name of the geometry or channel/machine axis that retracts		
	<position>:	Retraction position		
		Type:	REAL	
		WCS is valid for geometry axes, otherwise MCS. With the same identifiers for geometry and channel/machine axes, retraction is in the WCS.		
POLFA:	Predefined subprogram call for the specification of the retraction position of single axes			
	<axis>:	Channel axis identifier		
	<type>:	Position specification mode		
		Type:	INT	
		Value:	0:	Mark position value as invalid
			1:	Position value is absolute
			2:	Position value is incremental (distance)
		Note: If an axis is not a single axis or if the type is missing or type=0, then a corresponding alarm is output.		
	<position>:	Retraction position (see above)		
		Note: The position value is also accepted with type=0. Only this value is marked as invalid and has to be reprogrammed for retraction.		
POLFMASK:	Predefined subprogram call for selection of the axes that are to be retracted after tripping of rapid lift **independently of one another**.			
	<axis_1>,…:	Names of the axes that are to be traversed to their positions defined with POLF during rapid lift. All the axes specified must be in the same coordinate system.		
	POLFMASK() without specification of an axis deactivates the rapid lift for all axes that have been retracted independently of one another.			
POLFMLIN:	Predefined subprogram call for selection of the axes that are to be retracted after tripping of rapid lift **in linear relation**.			
	<axis_1>,…:	See above.		
	POLFMLIN() without specification of an axis deactivates the rapid lift for all axes that have been retracted in linear relation.			

Note

Before rapid retraction to a fixed position can be enabled via POLFMASK or POLFMLIN, a position must have been programmed with POLF for the selected axes.

Note

If axes are enabled one after the other with POLFMASK, POLFMLIN or POLFMLIN, POLFMASK, then the last definition always applies for the particular axis.

Note

The positions programmed with `POLF` and the activation by `POLFMASK` or `POLFMLIN` are deleted when the part program is started. This means that the user must reprogram the values for `POLF` and the selected axes in `POLFMASK` or `POLFMLIN` in each part program.

Note

If, when using the abbreviated form `POLFA` only the type is changed, then the user must ensure that either the retraction position or the retraction path contains a practical and sensible value. In particular, the retraction position and the retraction path have to be set again after Power On.

Example

Retracting an individual axis:

Program code	Comment
MD37500 $MA_ESR_REACTION[AX1]=21	; NC-controlled retraction.
...	
$AA_ESR_ENABLE[AX1]=1	
POLFA(AX1,1,20.0)	; AX1 is assigned the axial retraction position 20.0 (absolute).
$AA_ESR_TRIGGER[AX1]=1	; Retraction starts from here.

Further information

Requirements for NC-controlled retraction

- A retraction axis is configured for the NC-controlled retraction in the channel:
  MD37500 $MA_ESR_REACTION = 21
- ESR must be must be enabled for this axis:
  $AA_ESR_ENABLE = 1
- Delay times are defined:
  MD21380 $MC_ESR_DELAY_TIME1
  MD21381 $MC_ESR_DELAY_TIME2
- The axis-specific retraction positions have been configured with `POLF` in the part program.
- The axes are selected with `POLFMASK`/`POLFMLIN` for the NC-controlled retraction.
- The activate signals must be set for the retraction movement and remain set.

Enable and start NC-controlled reactions

If system variable $AC_ESR_TRIGGER = 1 is set and if a retraction axis is configured in this channel (i.e. MD37500 $MA_ESR_REACTION = 21) and $AA_ESR_ENABLE = 1 is set for this axis, then rapid lift (LIFTFAST) is activated in this channel.

The lift movement configured with `POLF` (or `POLF`) for the axes selected with `POLFMASK` or `POLFMLIN` replaces the path motion defined for these axes in the part program.

The sum of the MD21380 $MC_ESR_DELAY_TIME1 and MD21381 $MC_ESR_DELAY_TIME2 times is the maximum available for the retraction. When this time has expired, rapid deceleration with follow-up is also initiated for the retraction axis.

Note

The extended retraction (i.e. LIFTFAST/LFPOS triggered by $AC_ESR_TRIGGER) **cannot be interrupted** and can only be terminated prematurely via an emergency stop.

Note

Retraction initiated via $AC_ESR_TRIGGER is locked, in order to prevent multiple retractions.

Single axis retraction

With single axis retraction, the retraction position of the single axis must have been programmed with `POLFA` and the following conditions must be satisfied:

- $AA_ESR_ENABLE = 1
- <Axis> must be a single axis at the time of triggering ($AAAA_ESR_TRIGGER = 1).
- <Type> must be either 1 or 2.

Retraction direction during rapid lift

The frame valid at the time when the lift fast is activated is taken into consideration.

Note

Frames with rotation also affect the direction of lift via `POLF`.

Axis replacement

Retraction axes must always be assigned to exactly one NC channel and may not be switched among the channels. When an attempt is made to exchange a retraction axis in another channel, an alarm is output. Only once this axis has been deactivated again using $AA_ESR_ENABLE[AX] = 0 can it be exchanged in a new channel. Once the axis has been exchanged, axes can be acted upon again with $AA_ESR_ENABLE[AX] = 1.

Neutral axes

Neutral axes cannot undertake NC-controlled ESR.

4.19.1.2 NC-controlled stopping

The NC-controlled stopping is activated for the stopping axes configured in the channel by setting system variable $AC_ESR_TRIGGER (or $AA_ESR_TRIGGER for single axes).

Requirements

- A stopping axis is configured for the NC-controlled stopping in the channel:
  MD37500 $MA_ESR_REACTION = 22
- ESR must be enabled for this axis:
  $AA_ESR_ENABLE = 1
- Delay times are defined:
  MD21380 $MC_ESR_DELAY_TIME1 (delay time, ESR axes)
  MD21381 $MC_ESR_DELAY_TIME2 (ESR time for interpolatory braking)

Execution

This axis continues interpolating as programmed for the time period set in MD21380: After the time delay specified in MD21380 has expired, controlled braking (ramp stop) is initiated: The time period in MD21381 is the maximum available for the interpolatory controlled braking. After this period expires, fast braking with subsequent tracking is initiated.

Example

Stopping a single axis:

Program code	Comment
MD37500 $MC_ESR_REACTION[AX1] = 22	; NC-controlled stopping.
MD21380 $MC_ESR_DELAY_TIME1[AX1] = 0.3	
MD21381 $MC_ESR_DELAY_TIME2[AX1] = 0.06	
...	
$AA_ESR_ENABLE[AX1]=1	
$AA_ESR_TRIGGER[AX1]=1	; Stopping starts from here.

4.19.2 Drive-integrated ESR

4.19.2.1 Configuring drive-integrated stopping (ESRS)

The drive parameters for "stopping" of the drive-integrated ESR function are configured using the `ESRS(...)` function.

Syntax

```
ESRS(<access_1>,<stopping time_1>[,...,<axis_n>,<stopping time_n>])
```

Meaning

<table>
<tr><td rowspan="1"><code>ESRS(...):</code></td><td colspan="2">Function to write to the drive parameters for the ESR function "stopping"
The function:
• Must be alone in the block.
• Triggers a preprocessing stop.
• Cannot be used in synchronized actions.</td></tr>
<tr><td rowspan="3"><code><axis_1>,
...,
<axis_n>:</code></td><td colspan="2">Axis for which drive-integrated stopping should be configured
For this axis, drive parameter p0888 (configuration) is written to in the drive:
p0888 = 1</td></tr>
<tr><td>Type:</td><td>AXIS</td></tr>
<tr><td>Range of values:</td><td>Channel axis identifier</td></tr>
<tr><td rowspan="4"><code><stopping time_1>,
...,
<stopping time_n>:</code></td><td colspan="2">Time during which the drive continues to travel with the actual speed setpoint after a fault has occurred
For the specified axis, drive parameter p0892 (timer) is written to in the drive:
p0892 = <stopping time></td></tr>
<tr><td>Unit:</td><td>s</td></tr>
<tr><td>Type:</td><td>REAL</td></tr>
<tr><td>Range of values:</td><td>0.00 - 20.00</td></tr>
<tr><td colspan="3">A maximum of 5 axes can be programmed in a function call; n = 5</td></tr>
</table>

4.19.2.2 Configuring drive-integrated retraction (ESRS)

The drive parameters for "retraction" of the drive-integrated ESR function are configured using the `ESRR(...)` function.

Syntax

```
ESRR(<axis_1>,<retraction distance_1>,<retraction
velocity_1>[,...,<axis_n>,<retraction distance_n>,<retraction
velocity_n>])
```

Meaning

<table>
<tr><td rowspan="1"><code>ESRR(...):</code></td><td colspan="2">Function to write to the drive parameters for the ESR function "retract"
The function:
• Must be alone in the block.
• Triggers a preprocessing stop.
• Cannot be used in synchronized actions.</td></tr>
<tr><td rowspan="3"><code><axis_1>,
...,
<axis_n>:</code></td><td colspan="2">Axis for which drive-integrated retraction should be configured
For this axis, drive parameter p0888 (configuration) is written to in the drive:
p0888 = 2</td></tr>
<tr><td>Type:</td><td>AXIS</td></tr>
<tr><td>Range of values:</td><td>Channel axis identifier</td></tr>
<tr><td rowspan="4"><code><retraction distance_1>,
...,
<retraction distance_n>:</code></td><td colspan="2">For the drive, the retraction distance is converted into a retraction speed. For the specified axis, the value is written to drive parameter p0893 (speed):
p0893 = (<retraction distance _n> converted into retraction speed)</td></tr>
<tr><td>Unit:</td><td>mm/min, inch/min, degrees/min (depending on the unit of the axis)</td></tr>
<tr><td>Type:</td><td>REAL</td></tr>
<tr><td>Range of values:</td><td>MIN - MAX</td></tr>
<tr><td rowspan="4"><code><retraction velocity_1>,
...,
<retraction velocity_n>:</code></td><td colspan="2">For the drive, the retraction velocity is converted into a time. For the specified axis, the value is written to drive parameter p0892 (timer) [s]:
p0892 = <retraction distance_n> / <retraction velocity _n></td></tr>
<tr><td>Unit:</td><td>mm/min, inch/min, degrees/min (depending on the unit of the axis)</td></tr>
<tr><td>Type:</td><td>REAL</td></tr>
<tr><td>Range of values:</td><td>0.00 - MAX</td></tr>
<tr><td colspan="3">A maximum of 5 axes can be programmed in a function call; n = 5</td></tr>
</table>

4.20 Program runtime/part counter

Information on the program runtime and workpiece counter is 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.

4.20.1 Program runtime

The "program runtime" function provides internal NC timers to monitor technological processes, which can be read into the part program and into synchronized actions via the NC and channel-specific system variables.

The trigger for the runtime measurement ($AC_PROG_NET_TIME_TRIGGER) is the only system variable of the function that can be written to – and is used to selectively measure program sections. This means, by writing $AC_PROG_NET_TIME_TRIGGER in the NC program, the time measurement can be enabled and disabled again:

System variable	Meaning	Activity
NC-specific		
$AN_SETUP_TIME	Time since the last control power up with default values ("cold restart") in minutes. Is automatically reset to "0" every time the control powers up with default values.	• Always active
$AN_POWERON_TIME	Time since the last normal control power up ("warm restart") in minutes. Is automatically reset to "0" every time the control powers up normally.	
Channel-specific		
$AC_OPERATING_TIME	Total runtime of NC programs in automatic mode in seconds. The value is automatically reset to "0" every time the control powers up.	• Activated via MD27860 • Only AUTOMATIC mode
$AC_CYCLE_TIME	Runtime of the selected NC program in seconds. The value is automatically reset to "0" every time a new NC program starts up.	
$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. 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.	
$AC_ACT_PROG_NET_TIME	Actual net runtime of the current NC program in seconds. Is automatically reset to "0" when a new NC program starts.	• Always active • Only AUTOMATIC mode
$AC_OLD_PROG_NET_TIME	Net runtime in seconds of the program that has just be correctly ended with M30	
$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.	

System variable	Meaning		Activity
$AC_PROG_NET_TIME_TRIGGER	Trigger for the runtime measurement:		• Only AUTOMATIC mode
	0	Neutral state The trigger is not active.	
	1	Exit Ends the measurement and copies the value from $AC_ACT_PROG_NET_TIME into $AC_OLD_PROG_NET_TIME.$AC_ACT_PROG_NET_TIME is set to "0" and then continues to run.	
	2	Start Starts the measurement and in so doing sets $AC_ACT_PROG_NET_TIME to "0". $AC_OLD_PROG_NET_TIME is not changed.	
	3	Stop Stops the measurement. Does not change $AC_OLD_PROG_NET_TIME and keeps $AC_ACT_PROG_NET_TIME constant until it resumes	
	4	Resume The measurement is resumed, i.e. a measurement that was previously stopped is continued. $AC_ACT_PROG_NET_TIME continues. $AC_OLD_PROG_NET_TIME is not changed.	
All system variables are reset to 0 as a result of POWER ON!			

Note

Machine manufacturer

Machine data MD27860 $MC_PROCESSTIMER_MODE is used to switch-in the timer that can be activated.

The behavior of active time measurements for certain functions (e.g. GOTOS, override = 0%, active test run feed, program test, ASUB, PROG_EVENT, ...) is configured using machine data MD27850 $MC_PROG_NET_TIMER_MODE and MD27860 $MC_PROCESSTIMER_MODE.

Further information: Function Manual Basic Functions

Note

Residual time for a workpiece

If the same workpieces are machined one after the other, 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.

Supplementary conditions

- **Block search**
  No program runtimes are determined through block searches.
- **REPOS**
  The duration of a REPOS process is added to the current processing time ($AC_ACT_PROG_NET_TIME).

Examples

Example 1: 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 2: 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

Program code
N50 DO $AC_PROG_NET_TIME_TRIGGER=4
N60 mySubProgrammC
N70 DO $AC_PROG_NET_TIME_TRIGGER=1
N80 mySubProgrammD
N90 M30

4.20.2 Workpiece counter

The "Workpiece counter" function makes available various counters which can be used in particular internally in the control to count workpieces.

The counters exist as channel-specific system variables with read and write access in a range of values from 0 to 999 999 999.

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.
$AC_TOTAL_PARTS	Total number of completed workpieces (actual workpiece total) This counter specifies the total number of all workpieces produced since the start time. The value is only automatically reset to "0" when the control powers up with default values.
$AC_ACTUAL_PARTS	Number of completed workpieces (actual workpiece total) This counter registers the total number of all 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 powers up with default values and can be read and written independent of their activation.

Note

Channel-specific machine data can be used to control counter activation, counter reset timing and the counting algorithm.

Note

Workpiece counting with user-defined M command

Machine data can be set so that the count pulses for the various workpiece counters are triggered using user-defined M commands rather than the end of the program (`M2`/`M30`).

4.21 Program simulation

4.21.1 Function

The current part program is calculated completely in the program 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.

4.21.2 Program simulation in conjunction with handling channels

In program simulation in conjunction with a handling channel, the following special considerations must be observed:

- The program for the handling channel must be selected via the predefined procedure `INIT`.
  Example:
  ```
  INIT(3,"/_N_SPF_DIR/_N_LADER1_BELADEN_SPF")
  ; Selection of the program "LADER1_BELADEN" for channel 3
  (handling channel)
  ```
- The program selected in the handling channel must be started via the predefined procedure `START`.
  Example:
  ```
  START(3)
  ; Start selected program in channel 3 (handling channel)
  ```
- In the program simulation, no external PLC acknowledgments, e.g. for limit switches (gripper open/close), have to be considered. This means that, during program simulation, the command lines with the required external acknowledgments must be skipped.

Note

If the handling channel is not relevant for program simulation, the `START` command should be avoided in the simulation.

Note

Wait markers in the handling channel that cannot be acknowledged in the program simulation must be skipped with $P_SIM.

Note

In simulation of a multichannel data program (Prog. Sync), e.g. balance cutting, the program start must be avoided in a further channel.

Program example:

Program code	Comment
...	
IF $P_SIM==TRUE	; If simulation is active:
GOTOF simulation	; Program jumps to jump mark "Simulation".
ENDIF	
START(3)	; Start selected program in channel 3 (handling channel)
WAITM(4,1,3)	; Wait for wait marker 4 in channel 3 (handling channel).
Simulation:	
; *** Execution or simulation of the main program ***	
...	
...	
M30	

4.22 Additional functions

4.22.1 Activate machine data (NEWCONF)

The `NEWCONF` command activates all machine data. The function can also be activated in the HMI user interface by pressing the "MD data effective" softkey.

When the "NEWCONF" function is executed there is an implicit preprocessing stop; in other words, path movement is interrupted.

Syntax

```
NEWCONF
```

Meaning

`NEWCONF`:	Command for setting all machine data of the "NEW_CONFIG" effectiveness level active

Cross-channel execution of NEWCONF from the part program

If changes are made to axial machine data from the part program and then activated with NEWCONF, NEWCONF will only activate the machine data containing changes affecting the part program channel.

Note

In order to ensure that all changes are applied, the NEWCONF command must be executed in every channel in which the axes or functions affected by the changes to the machine data is being calculated.

No axial machine data is effective for NEWCONF.

An axial RESET must be performed for axes controlled by the PLC.

Example

Milling: Machine drill position with different technologies

Program code	Comment
N10 $MA_CONTOUR_TOL[AX]=1.0	; Change machine data.
N20 NEWCONF	; Activate machine data.
...	

4.22.2 Check scope of NC language present (STRINGIS)

Using the function "STRINGIS(...)" it can be checked as to whether the specified string is available as element of the NC programming language in the actual language scope.

Definition

```
INT STRINGIS(STRING <Name>)
```

Syntax

```
STRINGIS(<Name>)
```

Meaning

STRINGIS:	Function with return value
<name>:	Name of the NC programming language element to be checked
Return value:	The return value format is yxx (decimal).

Elements of the NC programming language

The following elements of the NC programming language can be checked:

- G commands of all existing G groups, e.g. "G0", "INVCW", "POLY", "ROT", "KONT", "SOFT", "CUT2D", "CDON", "RMBBL", "SPATH"
- DIN or NC addresses, such as "ADIS", "RNDM", "SPN", "SR", "MEAS"
- Functions, e.g. "TANG(...)" or "GETMDACT"
- Procedures, e.g. "SBLOF".
- Keywords, e.g. "ACN", "DEFINE" or "SETMS"
- System data, e.g. machine data $M... , setting data $S... or option data $O...
- System variables $A... , $V... , $P...
- Arithmetic parameter R...
- Cycle names of activated cycles
- GUD and LUD variables
- Macro names
- Label names

Return value

Only the first three decimal positions of the return value are relevant. The return value format is yxx, with y = basis information and xx = detailed information.

Return value	Meaning	
000	The string <Name> is not known in this system [1)]	
100	The string <Name> is an element of the NC programming language, but currently cannot be programmed (option/function is inactive)	
2xx	The string <Name> is a programmable element of the NC programming language (option/function is active). The detailed information xx contains additional information about the element type:	
	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 ($O...)
	07	System parameters, e.g. system variable ($...) or arithmetic parameter (R...)
	08	Cycle (The cycle must be loaded into the NC and the cycle programs must be active. [3)])
	09	GUD variable (The GUD variable must be defined in a GUD definition file and available in the NC)
	10	Macro name (The macro must be defined in a macro definition file and available in the NC) [4)]
	11	LUD variable of the current part program
	12	ISO G command (ISO language mode must be active)

Return value	Meaning
400	The string <Name> is an NC address that was not identified as xx == 01 or 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. For unknown strings that are actually Siemens NC language commands, the value 0 is returned. This behavior can be changed using MD10711 $MN_NC_LANGUAGE_CONFIGURATION: MD10711 = 1 ⇒ For Siemens NC language commands, the value 100 is always returned.

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 an address extension when being checked with STRINGIS.
Example: 000 == STRINGIS("M02"), but 400 == STRINGIS("M").

3) Names for cycle parameters cannot be checked with STRINGIS.

4) Addresses defined as a macro (e.g. G, H, M, L) are identified as a macro.

Examples

In the following examples it is assumed that the NC language elements specified as string - as long as nothing else is noted - 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 an NC address with extension:
   ```
   201 == STRINGIS("A")
   201 == STRINGIS("A2")
   ```
4. String "INVCW" is defined as a 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 "$TC_MPP4" is a system parameter (magazine location state):
    - Tool magazine management is active: `207 == STRINGIS("$TC_MPP4") ;`
    - Tool magazine management is not active: `000 == STRINGIS("$TC_MPP4")`

    Also refer to 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")`

Tool magazine management

If the tool magazine management function is not active, supplies STRINGIS for 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:

- MD18800 $MN_MM_EXTERN_LANGUAGE (activation, external NC languages)
- MD10880 $MN_ MM_EXTERN_CNC_SYSTEM (control system to be adapted)

STRINGIS checks the specified string initially as SINUMERIK G command. If the string is not a SINUMERIK G command, then it is subsequently checked as ISO G command.

Programmed switchovers (`G290` (SINUMERIK mode), `G291` (ISO Mode)) have no effect on STRINGIS.

Example

The machine data, relevant for the function STRINGIS(...), has the following values:

- MD10711 $MN_NC_LANGUAGE_CONFIGURATION = 2 (only the NC language commands whose options are set are considered to be known)
- MD19410 $ON_TRAFO_TYPE_MASK = 'H0' (option: transformations)
- MD10700 $MN_PREPROCESSING_LEVEL='H43' (preprocessing for cycles is active)

The following program example is executed without error message:

Program code	Comment
`N1 R1=STRINGIS("TRACYL")`	; R1 == 0, because TRACYL is identified as "not known" because of the missing transformation option
`N2 IF STRINGIS("TRACYL") == 204`	
`N3  TRACYL(1,2,3)`	; N3 is skipped
`N4 ELSE`	
`N5  G00`	; and instead, N5 is executed
`N6 ENDIF`	
`N7 M30`	

4.22.3 Interactively call the window from the part program (MMC)

User-specific dialogs from an NC program can be displayed on the user interface via the predefined subprogram MMC(...).

The configuration of the dialogs can be done for the following types of dialogs:

- Run MyScreens
- Easy XML
- User XML

Further information:

- Programming Manual Run MyScreens
- Programming Manual Easy XML

Syntax

```
MMC("<ADDRESS>,<COMMAND>,<FILE>,<DIALOG>","<QUIT>")
```

Meaning

MMC(...):	Subprogram identifier The parameters are specified position-coded and separated by a comma within two strings, the command string and the acknowledgement string.	
Parameters within the command string:		
<ADDRESS>:	Operating area in which the configured user dialog boxes are implemented	
	Function	**Operating areas**
	"Run MyScreens" user dialog	CYCLES
	"Easy XML" user dialog	CYCLES
	User XML	XML
	Pop-up window "Run MyScreens"	POPUPDLG
	Popup window "Easy XML"	POPUPDLG
<COMMAND>:	Command to be executed	
	Function	**Commands**
	"Run MyScreens" user dialog	PICTURE_ON, PICTURE_OFF
	"Easy XML" user dialog	PICTURE_ON, PICTURE_OFF
	User XML	XML_ON, XML_OFF
	Pop-up window "Run MyScreens"	PICTURE_ON, PICTURE_OFF
	Popup window "Easy XML"	PICTURE_ON, PICTURE_OFF

<table>
<tr><td rowspan="8"><FILE>:</td><td colspan="3">Name of the file in which the dialog to be displayed is programmed</td></tr>
<tr><td colspan="2">Function</td><td>Files</td></tr>
<tr><td colspan="2">"Run MyScreens" user dialog</td><td><name>.com</td></tr>
<tr><td colspan="2">"Easy XML" user dialog</td><td><name>.xml</td></tr>
<tr><td colspan="2">User XML</td><td><name>.xml</td></tr>
<tr><td colspan="2">Pop-up window "Run MyScreens"</td><td><name>.com</td></tr>
<tr><td colspan="2">Popup window "Easy XML"</td><td><name>.xml</td></tr>
<tr><td colspan="2">Popup window "Easy XML" with configuration direct in the NC program (see example 2)</td><td>xmldial_emb.xml</td></tr>
<tr><td rowspan="4"><DIALOG>:</td><td colspan="3">Name of the dialog to be displayed</td></tr>
<tr><td colspan="2">Function</td><td>Dialog name</td></tr>
<tr><td colspan="2">All functions except popup window "Easy XML" with configuration direct in the NC program</td><td>Name of the dialog configured in the <FILE> file</td></tr>
<tr><td colspan="2">Popup window "Easy XML" with configuration direct in the NC program (see example 3)</td><td>main</td></tr>
<tr><td colspan="4">Parameters within the acknowledgment string:</td></tr>
<tr><td rowspan="3"><QUIT>:</td><td colspan="3">Acknowledgment type</td></tr>
<tr><td>N:</td><td colspan="2">No acknowledgment.
Program execution is continued when the command has been transmitted. There is no feedback if the command could not be successfully executed.
Note
Acknowledgement type "N" must be used if a display time (dwell time) is programmed in the NC program (see Example 2 below)</td></tr>
<tr><td>A:</td><td colspan="2">Asynchronous acknowledgment
The program execution is continued after the command is issued. The return value is saved in a user-specific acknowledgement variable (GUD variable), which is defined within the scope of the dialog configuration, and can be read in the NC program.</td></tr>
</table>

Supplementary conditions

- The definition files *.com of the dialogs must be saved in the "proj" folder.
- The Easy XML definition files *.xml of the dialogs must be saved in the "appl" folder.
  If the definition files are saved in a different directory, the path must be specified indirectly, starting from the "appl" directory.
- User-defined dialogs from different channels cannot be simultaneously displayed.
- The MMC functionality is not supported in the simulation.

Examples

Example 1

Display of a dialog and response to the user operation in an NC program.

Program code	Comment
; The acknowledgement variable QUIT has already been created as a global user variable (GUD)	
; Of the type STRING when the dialog was configured:	
; DEF NCK STRING[20] QUIT	
QUIT = "XXX"	; Initialize acknowledgment variable
G4 F5	
MMC("CYCLES,PICTURE_ON,test.com,test1","A")	; Display dialog
	; - Operating area: CYCLES
	; - Picture status: PICTURE_ON (display)
	; - Dialog screen file: test.com
	; - Dialog screen: test1
INPUT:	; Wait for user input
STOPRE	; Preprocessing stop
IF MATCH (QUIT,"RUN") >= 0 GOTOF WORK	; Softkey "RUN"
IF MATCH (QUIT,"CHK") >= 0 GOTOF CHECK	; Softkey "CHK"
GOTOB INPUT	; => Wait
WORK:	; Softkey "RUN" pressed
MSG("Continue with processing -> NC start")	; Output message
MMC("CYCLES,PICTURE_OFF","N")	; Close dialog
M0	; Wait for NC start
GOTOF END	; => Program end
CHECK:	; Softkey "CHK" pressed
MSG("Approach position -> NC start")	; Output message
MMC("CYCLES,PICTURE_OFF","N")	; Close dialog
M0	; Wait for NC start
GOTOF END	; => Program end
END:	
...	

Example 2

The display time of a dialog is defined in the NC program via a dwell time, for example.

Program code	Comment
F1000 G94	
...	
MMC("POPUPDLG,PICTURE_**ON**,xmldial_emb.xml,main","N")	; Display dialog

Program code	Comment
X200	
Z40	
MMC("POPUPDLG,PICTURE_**OFF**","N")	; Close dialog

Example 3

Embedding a popup script in an NC program and its use.

Program code

```
PROC POPUP_TEST
; ----------------------------- Script -----------------------------
; <main_dialog entry="rpara_main">
;    <let name="xpos" />
;    <let name="ypos" />
;    <let name="field_name" type="string" />
;    <let name="num" />
;    <menu name="rpara_main">
;       <open_form name="rpara_form"/>
;       <softkey_back>
;          <close_form />
;       </softkey_back>
;    </menu>
;    <form name="rpara_form">
;       <init>
;          <caption>mask from NC part program</caption>
;          <let name="count" >0</let>
;          <op>
;             xpos =  120;
;             ypos = 34;
;             "nck/Channel/Parameter/R[10]" = 10;
;          </op>
;          <!-- load the number of controls -->
;          <op>
;             num = "nck/Channel/Parameter/R[10]";
;          </op>
;          <while>
;             <condition> count < num</condition>
;             <print name="field_name" text="edit%d">count</print>
;             <op>
;                ypos = ypos + 24;
;                count = count + 1;
;             </op>
;          </while>
;       </init>
;       <paint>
;          <op>
```

Program code

```
;             xpos =  8;
;             ypos = 36;
;             count = 0;
;          </op>
;          <while>
;             <condition>count < num</condition>
;             <print name="field_name" text="R-Parameter%d">count</print>
;             <text xpos = "$xpos" ypos = "$ypos" >$$$field_name</text>
;             <op>
;                ypos = ypos + 24;
;                count = count + 1;
;             </op>
;          </while>
;       </paint>
;    </form>
; </main_dialog>
; ========================= Program section ===========================
...
G94 F100
MMC("POPUPDLG,PICTURE_ON,xmldial_emb.xml,main","N")
G4 F4
X200
MMC("POPUPDLG,PICTURE_OFF","N")
G4 F2
X0
...
```

4.22.4 Process DataShare - output to an external device/file (EXTOPEN, WRITE, EXTCLOSE)

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	Reserved
		12	Write mode: Data contradicts extdev.ini
		16	Invalid external path has been programmed
		22	External device not mounted

<table>
<tr><td rowspan="9"><ExtDev>:</td><td colspan="3">Parameter 2: Symbolic identifier for the external device/file to be opened</td></tr>
<tr><td>Type:</td><td colspan="2">STRING</td></tr>
<tr><td colspan="3">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:</td></tr>
<tr><td colspan="2">"LOCAL_DRIVE":</td><td>Local CF card (pre-defined)</td></tr>
<tr><td colspan="2">"CYC_DRIVE":</td><td>Reserved drive name for use in SIEMENS cycles (pre-defined)</td></tr>
<tr><td colspan="2">"/dev/ext/1", ...
"/dev/ext/9":</td><td>Available network drives
Note:
It is necessary to configure in the extdev.ini file!</td></tr>
<tr><td colspan="2">"/dev/cyc/1",
"/dev/cyc/2":</td><td>Reserved drive names for use in SIEMENS cycles
Note:
It is necessary to configure in the extdev.ini file!</td></tr>
<tr><td colspan="3">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"</td></tr>
<tr><td colspan="3">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".</td></tr>
<tr><td rowspan="4"><SyncMode>:</td><td colspan="3">Parameter 3: Processing mode for the WRITE commands to this device/file</td></tr>
<tr><td>Type:</td><td colspan="2">STRING</td></tr>
<tr><td rowspan="2">Values:</td><td>"SYN":</td><td>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.</td></tr>
<tr><td>"ASYN":</td><td>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.</td></tr>
</table>

<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
```

Program code

```
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
…
```

4.22.5 Setting an alarm (SETAL)

Alarms can be set from within an NC program.

The alarm text to be configured via the user interface is output in the status display of the user interface.

Alarm texts can contain predefined parameters or parameters with variable user texts.

An alarm always goes hand in hand with a response from the control according to the alarm category.

Syntax

```
SETAL(<No>[,<String>])
```

Meaning

`SETAL(...)`	Predefined procedure for setting an alarm SETAL must be programmed in a separate NC block.		
`<No>`	Alarm number		
	Data type:	INT	
	Value range:	60000 ... 64999 (reserved)	Alarms for SIEMENS cycles
		65000 ... 69999	Alarms for user cycles
`<String>`	String (optional)		
	Data type:	STRING	

Alarm texts and alarm parameters

Alarm texts

Alarm texts are configured in the user interface.

Alarm parameters

User cycle alarms can contain the parameter values %1 ... %4:

%1	Predefined parameter: Channel number
%2	Predefined parameter: Block number, label
%3	Predefined parameter: Offset of the alarm number User cycle alarms are assigned to the following number ranges: • 65000 ... 65499 • 65500 ... 65999 • 66000 ... 66999 • 67000 ... 67999 • 68000 ... 68999 • 69000 ... 69999 The displayed offset value refers to the start number of the associated number range and is determined as follows: Alarm number - start number in the number range = offset value
%4	Additional alarm parameter Replaced by the string of data type STRING specified in the SETAL call for user cycle alarms.

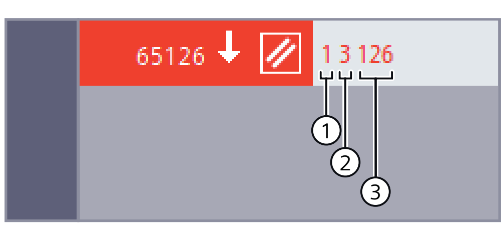
Examples

The following examples are used to demonstrate the output of the alarm parameter values. For reasons of clarity, no alarm text has been stored in the alarms used, so that only the values of the transmitted parameters are output in sequence.

Example 1: SETAL call without specification of a string

Program code	Comment
N10 ...	
N20 ...	
N30 **SETAL(65126)**	; Set alarm no. 65126
...	

After the SETAL call, the following information appears in the status display of the user interface:

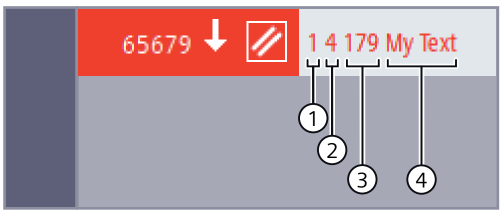


① Channel number = 1
② Block number = 3 (3rd program line)
③ Offset value = 126 (65126 - 65000 = 126)

Example 2: SETAL call with specification of a string

```
Program code                          Comment
N10 ...
N20 ...
N30 ...
N40 SETAL(65679, "My Text")           ; Set alarm no. 65679
...
```

After the SETAL call, the following information appears in the status display of the user interface:

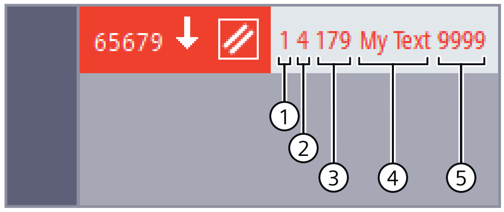


① Channel number = 1
② Block number = 4 (4th program line)
③ Offset value = 179 (65679 - 65500 = 179)
④ String specified in the SETAL call

Example 3: SETAL call with specification of a chained string

```
Program code                          Comment
N10 ...
N20 ...
N30 ...
N40 SETAL(65679, "My Text " <<9999)   ; Set alarm no. 65679
...
```

After the SETAL call, the following information appears in the status display of the user interface:



①	Channel number = 1
②	Block number = 4 (4th program line)
③	Offset value = 179 (65679 - 65500 = 179)
④⑤	Chained string specified in the SETAL call ④ The string of data type STRING specified in the SETAL call in quotes "" forms the first section of the chained string. ⑤ The following value of data type INT specified with the chaining operator << forms the second section of the chained string. It is converted to the STRING data type and appended to the first section.

Note

Chaining to form a common string is only possible if the value to be appended is preceded by the chaining operator <<. Otherwise no conversion into the data type STRING takes place and the alarm 12330 "Type of parameter ... wrong" is output.

More information

Alarm response and acknowledgment

User cycle alarms are assigned to number ranges that differ with regard to alarm response and acknowledgment:

Number range	Alarm response	Alarm acknowledgment
65000 - 65499	Display, NC Start disable	Reset
65500 - 65999	Display, NC Start disable (not for ASUBs 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

Query current language of the user interface

If an alarm is to be output in the language active at the user interface, then the user requires information about the language that is currently set at the HMI. This information can be queried in the part program and in synchronous actions via the system variable $AN_LANGUAGE_ON_HMI (Page 1227).

4.22.6 Define blank (WORKPIECE)

The controller must know the shape and size of a blank to be able to display it in the graphical simulation. The user therefore has the capability of defining blanks via the user interface or directly in the NC program. The definitions of blanks are retained beyond a (program end/ channel/BAG) reset. They are automatically deleted the next time that the control system powers up.

Syntax

```
WORKPIECE("<WP>", "<RefP>", "<ZeroOffset>", "<Type>", <Par5>,
<Par6>, ..., <Par12>)
```

Meaning

<table>
<tr><td colspan="2" rowspan="3"><code>WORKPIECE(...):</code></td><td colspan="3">Predefined procedure for defining a blank</td></tr>
<tr><td>Preprocessing stop:</td><td colspan="2">Yes</td></tr>
<tr><td>Alone in the block:</td><td colspan="2">Yes</td></tr>
<tr><td colspan="5">Parameters:</td></tr>
<tr><td rowspan="3">1</td><td rowspan="3"><code>"<WP>":</code></td><td colspan="3">Name of the workpiece (optional)</td></tr>
<tr><td>Data type:</td><td colspan="2">STRING</td></tr>
<tr><td colspan="3">A specification is only necessary if there can be several workpieces in one channel. Without specifying, "WORKP<n>" is automatically accepted, with <n> being the number of the declaring channel.</td></tr>
<tr><td rowspan="7">2</td><td rowspan="7"><code>"<RefP>":</code></td><td colspan="3">Clamping (optional, only for milling machines)</td></tr>
<tr><td>Data type:</td><td colspan="2">STRING</td></tr>
<tr><td rowspan="4">Range of values:</td><td>"Table"</td><td>Clamping of the fixed table</td></tr>
<tr><td>"A"</td><td>Clamping on rotary axis A</td></tr>
<tr><td>"B"</td><td>Clamping on rotary axis B</td></tr>
<tr><td>"C"</td><td>Clamping on rotary axis C</td></tr>
<tr><td colspan="3">Precondition:
The table or the rotary axis must be enabled via the corresponding machine data for the clamping of the blank (see SINUMERIK Operate Commissioning Manual).</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2"><code>"<ZeroOffset>":</code></td><td colspan="3">Settable work offset for positioning the blank (not programmable)</td></tr>
<tr><td colspan="3">The selection of a settable work offset for positioning the blank is only offered for the blank entry via the user interface. For the direct definition of the blank in the part program, the blank always relates to the currently valid work offset.</td></tr>
</table>

4	`"<Type>":`	Blank shape		
		Data type:	STRING	
		Range of values:	`"CYLINDER":`	Cylinder
			`"PIPE":`	Pipe
			`"RECTANGLE":`	Centered cuboid
			`"BOX":`	Cuboid
			`"N_CORNER":`	Polygon with n edges
5 ... 12	`<Par5>` ... `<Par12>`:	Parameters for description of the blank shape		
		Data type:	REAL	
		The number of parameters required and their meaning depend on the respective blank shape and the value of the bit parameter. See: • "Parameters for description of the blank shape" table • "Bit parameters" table		
`WORPIECE():`		A WORKPIECE call without parameters deletes all blank definitions.		
`WORPIECE(<WP>):`		A WORKPIECE call with workpiece name only deletes this blank definition.		

Table 4-5 Parameters for description of the blank shape

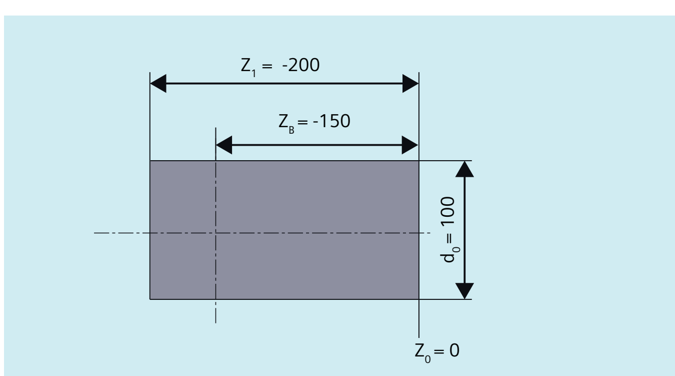
Blank shape	**Parameter**							
	<Par5>	**<Par6>**	**<Par7>**	**<Par8>**	**<Par9>**	**<Par10>**	**<Par11>**	**<Par12>**
Cylinder	Bit parameter Real value that is interpreted as bit-coded integer value. The bits define the meaning of the following parameters (see "Bit parameters" table).	Reference point Z_0	Length Z_1	Machining dimension Z_B	Outer diameter d_0	-	Rotation about rotary axis	-
Pipe		Reference point Z_0	Length Z_1	Machining dimension Z_B	Outer diameter d_0	Wall thickness (inc) / inner diameter d_1 (abs)	Rotation about rotary axis	-
Centered cuboid		Reference point Z_0	Length Z_1	Machining dimension Z_B	Width W	Length L	Rotation about rotary axis	-
Cuboid		Reference point Z_0	Length Z_1	Machining dimension Z_B	X_0	Y_0	X_1	Y_1
Polygon with n edges		Reference point Z_0	Length Z_1	Machining dimension Z_B	Number of corners	Width across flats	Rotation about rotary axis	-

Table 4-6 Bit parameter

Bit	Meaning	
4 (0x0010)	Cuboid: X_1	
	= 0	inc
	= 1	abs
5 (0x0020)	Cuboid: Y_1	
	= 0	inc
	= 1	abs
6 (0x0040)	Length Z_1 (final dimension)	
	= 0	inc
	= 1	abs
Bit 7 (0x0080)	Machining dimension Z_B	
	= 0	inc
	= 1	abs
Bit 8 (0x0100)	Pipe: Wall thickness / inner diameter	
	= 0	inc
	= 1	abs
9 (0x0200)	Polygon with n edges	
	= 0	Width across flats
	= 1	Edge length
12 (0x1000)	Clamping for turning machines	
	= 0	Main spindle
	= 1	Counterspindle
13 (0x2000)	Counterspindle	
	= 0	with mirroring
	= 1	without mirroring

Examples

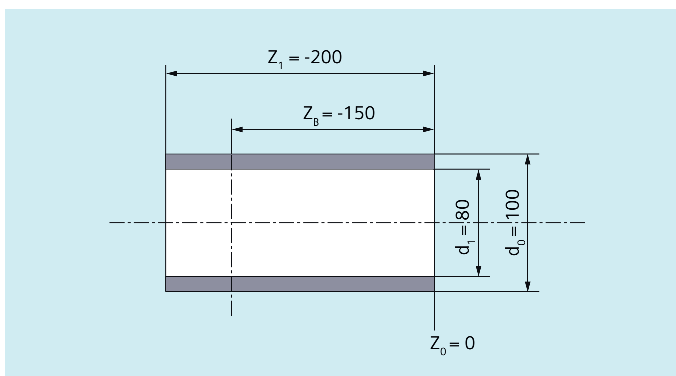
Example 1: Cylinder-shaped blank on a turning machine



Program code	Comment
...	

Program code	Comment
`WORKPIECE(,,,"CYLINDER",0,0,-200,-150,100)`	**; Blank definition:**
	; Blank shape: Cylinder
	; Bit parameter=0(no bit set) → Values for length and machining dimension are incremental, blank on main spindle
	; Reference point(Z0)=0
	; Length(Z1)=-200
	; Machining dimension(ZB)=-150
	; Outer diameter(d0)=100
...	

Example 2: Pipe-shaped blank on a turning machine



Program code	Comment
...	
`WORKPIECE(,,,"PIPE",256,0,-200,-150,100,80)`	**; Blank definition:**
	; Blank shape: Pipe
	; Bit parameter=256(Bit8=1) → Inner diameter is absolute; length and machining dimension are incremental, blank on main spindle
	; Reference point(Z0)=0
	; Length(Z1)=-200
	; Machining dimension(ZB)=-150
	; Outer diameter(d0)=100
	; Inner diameter(d1)=80
...	

4.22.7 Switch language mode (G290, G291)

The controller gives you the capability of reading in part programs from external CNC systems and processing them. The prerequisite is that the corresponding NC language mode (ISO dialect) has been defined during commissioning.

The ISO dialect mode can be activated separately for each channel. For example, channel 1 can run in ISO dialect mode while channel 2 is active in SINUMERIK mode.

The switchover between SINUMERIK mode and ISO dialect mode is done in the NC program via the commands of the G-group 47. The active tool, tool compensation and work offsets are not influenced by the switchover.

Syntax

```
G291
...
G290
```

Meaning

G290:	Activate SINUMERIK language mode	
	Alone in the block:	yes
	Effective:	Modal
G291:	Activate ISO language mode	
	Alone in the block:	yes
	Effective:	Modal

Conditions

SINUMERIK mode

- The default of the G commands can be defined for each channel via machine data.
- No language commands from the ISO dialects can be programmed in SINUMERIK mode.

ISO dialect mode

- The ISO dialect mode can be set with machine data as the basic setting of the control system. In ISO dialect mode, the control system then reboots by default.
- Only G commands from the ISO dialect can be programmed. The programming of SINUMERIK G functions is not possible in ISO dialect mode.
- ISO dialect and SINUMERIK language cannot be mixed in the same NC block.
- G commands cannot be used to switch between ISO dialect M (milling) and ISO dialect T (turning).
- Subprograms that are programmed in SINUMERIK mode can be called.
- If SINUMERIK functions are to be used, a switchover to SINUMERIK mode must first be made (see example).

Example

Compression of linear blocks in the ISO dialect mode

Program code	Comment
N5 G290	; Activate SINUMERIK language mode.
N10 COMPCAD	; COMPCAD is a command in the Siemens language and activates a compressor function that replaces the successive linear blocks with polynomial blocks with path lengths that are as long as possible.
N15 G291	; Activate ISO language mode.
N20 G01 X100 Y100 F1000	; Since COMPCAD has been activated in the SINUMERIK mode, even linear blocks in the ISO dialect mode can be compressed with this function.
...	

Further information

Function Manual ISO Dialects

4.23 User stock removal programs

4.23.1 Supporting functions for stock removal

Preprogrammed stock removal programs are provided for stock removal. Beyond this, you have the possibility of generating your own stock removal programs using the following listed functions:

- Generate contour table (CONTPRON) (Page 880)
- Generate coded contour table (CONTDCON) (Page 885)
- Determine point of intersection between two contour elements (INTERSEC) (Page 889) (Only for tables that were generated using CONTPRON)
- Execute the contour elements of a table block-by-block (EXECTAB) (Page 890) (Only for tables that were generated using CONTPRON)
- Calculate circle data (CALCDAT) (Page 891)
- Deactivate contour preparation (EXECUTE) (Page 893)

Note

You can use these functions universally, not just for stock removal.

Preconditions

The following conditions must be satisfied before calling the CONTPRON or CONTDCON functions:

- A starting point was approached that permits collision-free machining.
- The cutting edge radius compensation is deactivated using G40.

4.23.2 Generate contour table (CONTPRON)

`CONTPRON` switches on the contour preparation. The NC blocks that are subsequently called are not executed, but are split-up into individual movements and stored in the contour table. Each contour element corresponds to one row in the two-dimensional array of the contour table. The number of relief cuts is returned.

Syntax

Activate contour preparation:

```
CONTPRON(<contour table>,<machining type>,<relief cuts>,
<machining direction>)
```

Deactivate contour preparation and return to the normal execution mode:

```
EXECUTE(<ERROR>)
```

See "Deactivate contour preparation (EXECUTE) (Page 893)"

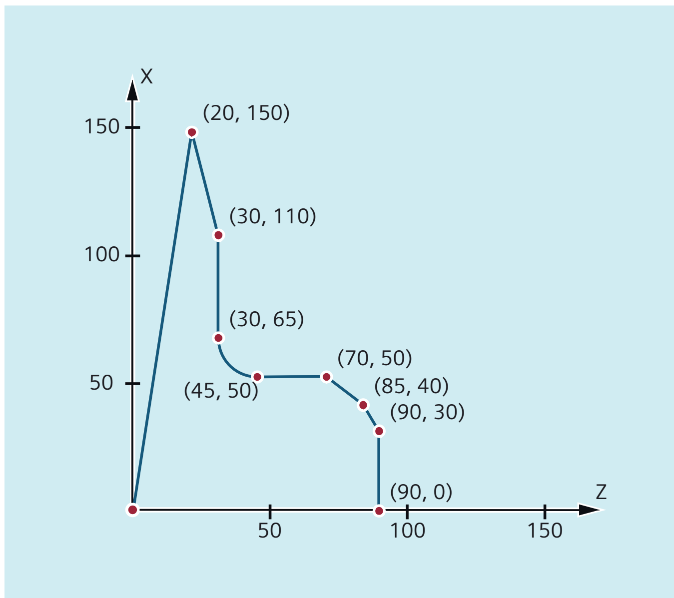
Meaning

`CONTPRON:`	Predefined procedure to activate the contour preparation to generate a contour table		
`<contour table>:`	Name of contour table		
`<machining type>:`	Parameter for the machining type		
	Type:	CHAR	
	Value:	`"G":`	Longitudinal turning: Internal machining
		`"L":`	Longitudinal turning: External machining
		`"N":`	Face turning: Internal machining
		`"P":`	Face turning: External machining
`<relief cuts>:`	Result variable for the number of relief cut elements that occur		
	Type:	INT	
`<machining direction>:`	Parameters for the machining direction		
	Type:	INT	
	Value:	0	Contour preparation, forward (default value)
		1	Contour preparation in both directions

Example 1

Generating a contour table with:

- Name "KTAB"
- Max. 30 contour elements (circles, straight lines)
- One variable for the number of relief cut elements that occur
- One variable for error messages



NC program:

Program code	Comment
`N10 DEF REAL KTAB[30,11]`	; Contour table with the name KTAB and max. 30 contour elements, parameter value 11 (number of table columns) is a fixed quantity.
`N20 DEF INT ANZHINT`	; Variable for the number of relief cut elements with the name ANZHINT.
`N30 DEF INT ERROR`	; Variable for error feedback signal (0=no error, 1=error).
`N40 G18`	
`N50 CONTPRON(KTAB,"G",ANZHINT)`	; Activate contour preparation.
`N60 G1 X150 Z20`	; N60 to N120: Contour description
`N70 X110 Z30`	
`N80 X50 RND=15`	
`N90 Z70`	
`N100 X40 Z85`	
`N110 X30 Z90`	
`N120 X0`	

Program code	Comment
N130 EXECUTE(ERROR)	; End filling the contour table, switch-over to normal program mode.
N140 …	; Continue to process the table.

Contour table KTAB:

Index Line	Column									
(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
7	7	11	0	0	20	150	0	82.40535663	0	0
0	2	11	20	150	30	110	-1111	104.0362435	0	0
1	3	11	30	110	30	65	0	90	0	0
2	4	13	30	65	45	50	0	180	45	65
3	5	11	45	50	70	50	0	0	0	0
4	6	11	70	50	85	40	0	146.3099325	0	0
5	7	11	85	40	90	30	0	116.5650512	0	0
6	0	11	90	30	90	0	0	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

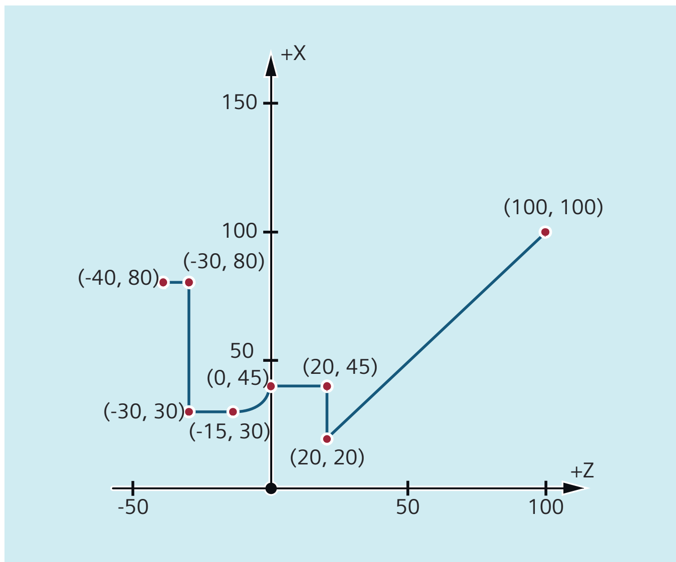
Explanation of the column contents:

(0)	Pointer to next contour element (to the row number of that column)
(1)	Pointer to previous contour element
(2)	Coding the contour mode for motion Possible values for X = abc a = 10^2 G90 = 0 G91 = 1 b = 10^1 G70 = 0 G71 = 1 c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3
(3), (4)	Starting point of contour elements (3) = abscissa, (4) = ordinate of the current plane
(5), (6)	Starting point of the contour elements (5) = abscissa, (6) = ordinate of the current plane
(7)	Max/min indicator: Identifies local maximum and minimum values on the contour
(8)	Maximum value between contour element and abscissa (for longitudinal machining) or ordinate (for face cutting). The angle depends on the type of machining programmed.
(9), (10)	Center point coordinates of contour element, if it is a circle block. (9) = abscissa, (10) = ordinate

Example 2

Generating a contour table with

- Name KTAB
- Max. 92 contour elements (circles, straight lines)
- Operating mode: Longitudinal turning, external machining
- Preparation, forward and backward



NC program:

Program code	Comment
`N10 DEF REAL KTAB[92,11]`	`; Contour table with name KTAB and max. 92 contour elements, parameter value 11 is a fixed quantity.`
`N20 DEF CHAR BT="L"`	`; Mode for CONTPRON: Longitudinal turning, external machining`
`N30 DEF INT HE=0`	`;Number of relief cut elements=0`
`N40 DEF INT MODE=1`	`; Preparation, forward and backward`
`N50 DEF INT ERR=0`	`; Error feedback signal`
`...`	
`N100 G18 X100 Z100 F1000`	
`N105 CONTPRON(KTAB,BT,HE,MODE)`	`; Activate contour preparation.`
`N110 G1 G90 Z20 X20`	
`N120 X45`	
`N130 Z0`	
`N140 G2 Z-15 X30 K=AC(-15) I=AC(45)`	
`N150 G1 Z-30`	
`N160 X80`	
`N170 Z-40`	

Program code	Comment
`N180 EXECUTE(ERR)`	`; End filling the contour table, switch-over to normal program mode.`
`...`	

Contour table KTAB:

After contour preparation is finished, the contour is available in both directions.

Index	Column										
Line	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
0	6 1)	7 2)	11	100	100	20	20	0	45	0	0
1	0 3)	2	11	20	20	20	45	-3	90	0	0
2	1	3	11	20	45	0	45	0	0	0	0
3	2	4	12	0	45	-15	30	5	90	-15	45
4	3	5	11	-15	30	-30	30	0	0	0	0
5	4	7	11	-30	30	-30	45	-1111	90	0	0
6	7	0 4)	11	-30	80	-40	80	0	0	0	0
7	5	6	11	-30	45	-30	80	0	90	0	0
8	1 5)	2 6)	0	0	0	0	0	0	0	0	0
	...										
83	84	0 7)	11	20	45	20	80	0	90	0	0
84	90	83	11	20	20	20	45	-1111	90	0	0
85	0 8)	86	11	-40	80	-30	80	0	0	0	0
86	85	87	11	-30	80	-30	30	88	90	0	0
87	86	88	11	-30	30	-15	30	0	0	0	0
88	87	89	13	-15	30	0	45	-90	90	-15	45
89	88	90	11	0	45	20	45	0	0	0	0
90	89	84	11	20	45	20	20	84	90	0	0
91	83 9)	85 10)	11	20	20	100	100	0	45	0	0

Explanation of column contents and comments for lines 0, 1, 6, 8, 83, 85 and 91

The explanations of the column contents given in example 1 apply.

Always in table line 0:

1) Predecessor: Line n contains the contour end (forward)

2) Successor: Line n is the contour table end (forward)

Once each within the contour elements forward:

3) Predecessor: Contour start (forward)

4) Successor: Contour end (forward)

Always in line contour table end (forward) +1:

5) Predecessor: Number of relief cuts (forward)

6) Successor: Number of relief cuts (backward)

Once each within the contour elements backward:

7) Successor: Contour end (backward)

8) Predecessor: Contour start (backward)

Always in last line of table:

9) Predecessor: Line n is the contour table start (backward)

10) Successor: Line n contains the contour start (backward)

Further information

Permitted traversing commands, coordinate system

The following G commands can be used for the contour programming:

- G group 1: `G0`, `G1`, `G2`, `G3`

In addition, the following are possible:

- Rounding and chamfer
- Circle programming using `CIP` and `CT`

The spline, polynomial and thread functions result in errors.

Changes to the coordinate system by activating a frame are not permissible between `CONTPRON` and `EXECUTE`. The same applies for a change between `G70` and `G71` or `G700` and `G710`.

Replacing the geometry axes with `GEOAX` while preparing the contour table produces an alarm.

Relief cut elements

The contour description for the individual relief cut elements can be performed either in a subprogram or in individual blocks.

Stock removal independent of the programmed contour direction

The contour preparation with `CONTPRON` was expanded so that after it has been called, the contour table is available independent of the programmed direction.

4.23.3 Generate coded contour table (CONTDCON)

With the contour preparation activated with `CONTDCON`, the following NC blocks that are called are saved in a coded form in a 6-column contour table to optimize memory use. Each contour element corresponds to one row in the contour table. When familiar with the coding rules specified below, e.g. you can combine DIN code programs for cycles from the table lines. The data of the output point is saved in the table line with the number 0.

Syntax

Activate contour preparation:
`CONTDCON(<contour table>,<machining direction>)`

Deactivate contour preparation and return to the normal execution mode:

`EXECUTE(<ERROR>)`

See "Deactivate contour preparation (EXECUTE) (Page 893)"

Meaning

`CONTDCON:`	Predefined procedure to activate the contour preparation to generate a coded contour table		
`<contour table>:`	Name of the contour table		
`<machining direction>:`	Parameter for machining direction		
	Type:	INT	
	Value:	0	Contour preparation according to the sequence of contour blocks (default value)
		1	Not permissible

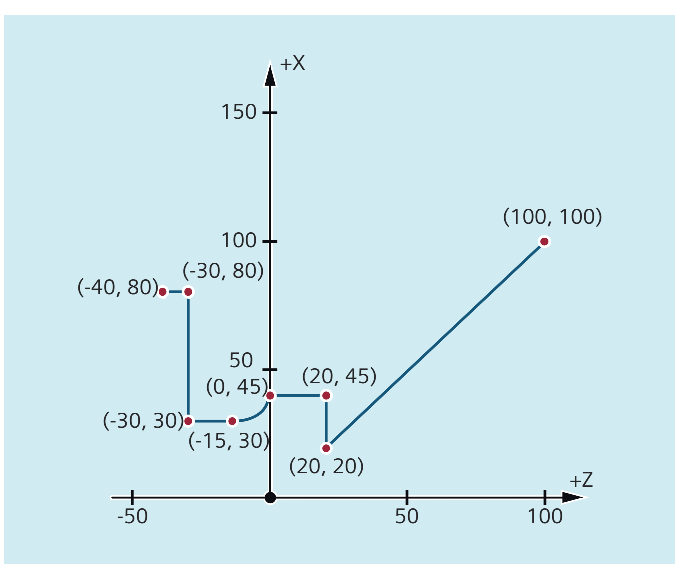
Note

The G commands permitted for `CONTDCON` in the program section to be included in the table are more comprehensive than for `CONTPRON`. Further, feedrates and feedrate type are saved for each contour section.

Example

Generating a contour table with:

- Name "KTAB"
- Contour elements (circles, straight lines)
- Operating mode: Turning
- Machining direction: Forward



NC program:

Program code	Comment
`N10 DEF REAL KTAB[9,6]`	`;Contour table with name KTAB and 9 table cells. These allow 8 contour sets. The parameter value 6 (column number in table) is a fixed size.`
`N20 DEF INT MODE = 0`	`; Variable for the machining direction. Standard value 0: Only in the programmed direction of the contour.`
`N30 DEF INT ERROR = 0`	`; Variable for the error feedback signal.`
`...`	
`N100 G18 G64 G90 G94 G710`	
`N101 G1 Z100 X100 F1000`	
`N105 CONTDCON (KTAB, MODE)`	`; Contour preparation call (MODE can be omitted).`
`N110 G1 Z20 X20 F200`	`; Contour description.`
`N120 G9 X45 F300`	
`N130 Z0 F400`	
`N140 G2 Z-15 X30 K=AC(-15) I=AC(45)F100`	
`N150 G64 Z-30 F600`	
`N160 X80 F700`	
`N170 Z-40 F800`	
`N180 EXECUTE(ERROR)`	`; End filling the contour table, switchover to normal program mode.`
`...`	

Contour table KTAB:

	Column index					
	0	1	2	3	4	5
Line index	Contour mode	End point abscissa	End point ordinate	Center point abscissa	Center point ordinate	Feedrate
0	30	100	100	0	0	7
1	11031	20	20	0	0	200
2	111031	20	45	0	0	300
3	11031	0	45	0	0	400
4	11032	-15	30	-15	45	100
5	11031	-30	30	0	0	600
6	11031	-30	80	0	0	700
7	11031	-40	80	0	0	800
8	0	0	0	0	0	0

Explanation of the column contents:

Line 0 Coding for the **starting point:**

Column 0:	10^0 (ones digit): G0 = 0
	10^1 (tens digit): G70 = 0, G71 = 1, G700 = 2, G710 = 3
Column 1:	Starting point abscissa
Column 2:	Starting point ordinate
Column 3-4:	0
Column 5:	Line index of last contour piece in the table

Lines 1-n: Entries for **contour pieces**

Column 0:	10^0 (ones digit): G0 = 0, G1 = 1, G2 = 2, G3 = 3
	10^1 (tens digit): G70 = 0, G71 = 1, G700 = 2, G710 = 3
	10^2 (hundreds digit): G90 = 0, G91 = 1
	10^3 (thousands digit): G93 = 0, G94 = 1, G95 = 2, G96 = 3
	10^4 (ten thousands digit): G60 = 0, G44 = 1, G641 = 2, G642 = 3
	10^5 (hundred thousands digit): G9 = 1
Column 1:	End point abscissa
Column 2:	End point ordinate
Column 3:	Center point abscissa for circular interpolation
Column 4:	Center point ordinate for circular interpolation
Column 5:	Feedrate

Further information

Permitted traversing commands, coordinate system

The following G groups and G commands can be used for the contour programming:

G group 1:	`G0`, `G1`, `G2`, `G3`
G group 10:	`G60`, `G64`, `G641`, `G642`
G group 11:	`G9`
G group 13:	`G70`, `G71`, `G700`, `G710`
G group 14:	`G90`, `G91`
G group 15:	`G93`, `G94`, `G95`, `G96`, `G961`

In addition, the following are possible:

- Rounding and chamfer
- Circle programming using `CIP` and `CT`

The spline, polynomial and thread functions result in errors.

Changes to the coordinate system by activating a frame are not permissible between `CONTDCON` and `EXECUTE`. The same applies for a change between `G70` and `G71` or `G700` and `G710`.

Replacing the geometry axes with `GEOAX` while preparing the contour table produces an alarm.

Machining direction

The contour table generated using `CONTDCON` is used for stock removal in the programmed direction of the contour.

4.23.4 Determine point of intersection between two contour elements (INTERSEC)

`INTERSEC` determines the point of intersection of two normalized contour elements from the contour tables generated using `CONTPRON`.

Syntax

```
<Status>=INTERSEC(<contour table_1>[<contour element_1>],
<contour table_2>[<contour element_2>],<intersection
point>,<machining type>)
```

Meaning

<table>
<tr><td>INTERSEC:</td><td colspan="3">Predefined function to determine the point of intersection between two contour elements from the contour tables generated with CONTPRON</td></tr>
<tr><td rowspan="4"><Status>:</td><td colspan="3">Variable for the point of intersection status</td></tr>
<tr><td>Type:</td><td colspan="2">BOOL</td></tr>
<tr><td rowspan="2">Value:</td><td>TRUE</td><td>Point of intersection found</td></tr>
<tr><td>FALSE</td><td>No intersection found</td></tr>
<tr><td><contour table_1>:</td><td colspan="3">Name of the first contour table</td></tr>
<tr><td><contour element_1>:</td><td colspan="3">Number of the contour element of the first contour table</td></tr>
<tr><td><contour table_2>:</td><td colspan="3">Names of the second contour table</td></tr>
<tr><td><contour element_2>:</td><td colspan="3">Number of the contour element of the second contour table</td></tr>
<tr><td rowspan="2"><point of intersection>:</td><td colspan="3">Intersection coordinates in the active plane (G17 / G18 / G19)</td></tr>
<tr><td>Type:</td><td colspan="2">REAL</td></tr>
<tr><td rowspan="4"><machining type>:</td><td colspan="3">Parameter for the machining type</td></tr>
<tr><td>Type:</td><td colspan="2">INT</td></tr>
<tr><td rowspan="2">Value:</td><td>0</td><td>Point of intersection calculation in the active plane with parameter 2
(standard value)</td></tr>
<tr><td>1</td><td>Point of intersection calculation independent of the transferred plane</td></tr>
</table>

Note

Please note that the variables must be defined before they are used.

The values defined with CONTPRON must be observed when transferring the contours:

Parameter	Meaning
2	Coding of contour mode for the movement
3	Contour start point abscissa
4	Contour start point ordinate
5	Contour end point abscissa
6	Contour end point ordinate
9	Center point coordinates for abscissa (only for circle contour)
10	Center point coordinates for ordinate (only for circle contour)

Example

Calculate the intersection of contour element 3 in table TABNAME1 and contour element 7 in table TABNAME2. The intersection coordinates in the active plane are stored in the variables ISCOORD (1st element = abscissa, 2nd element = ordinate). If no intersection exists, the program jumps to NOCUT (no intersection found).

Program code	Comment
`DEF REAL TABNAME1[12,11]`	`; Contour table 1`
`DEF REAL TABNAME2[10,11]`	`; Contour table 2`
`DEF REAL ISCOORD [2]`	`; Variable for the intersection coordinates.`
`DEF BOOL ISPOINT`	`; Variable for the intersection status.`
`DEF INT MODE`	`; Variable for the machining type.`
`…`	
`MODE=1`	`; Calculation independent of the active plane.`
`N10 ISPOINT=INTERSEC(TABNAME1[3],TABNAME2[7], ISCOORD,MODE)`	`; Intersection of the contour elements call.`
`N20 IF ISPOINT==FALSE GOTOF NOCUT`	`; Jump to NOCUT.`
`…`	

4.23.5 Execute the contour elements of a table block-by-block (EXECTAB)

Using EXECTAB, you can execute the contour elements of a table – that were generated, e.g. with CONTPRON – block-by-block.

Syntax

```
EXECTAB(<contour table>[<contour element>])
```

Meaning

EXECTAB:	Predefined procedure to execute a contour element
<contour table>:	Name of the contour table
<contour element>:	Number of the contour element

Example

Contour elements 0 to 2 in table KTAB should be executed block-by-block.

Program code	Comment
N10 EXECTAB(KTAB[0])	; Traverse element 0 of table KTAB.
N20 EXECTAB(KTAB[1])	; Traverse element 1 of table KTAB.
N30 EXECTAB(KTAB[2])	; Traverse element 2 of table KTAB.

4.23.6 Calculate circle data (CALCDAT)

With CALCDAT, you can calculate the radius and the circle center point coordinates from the three or four points known along the circle. The specified points must be different.

Where four points do not lie directly on the circle an average value is formed for the circle center point and the radius.

Note

Calculation regulation for the averaging

The arc calculation is performed four times:

1. With circle points 1, 2, 3
2. With circle points 1, 2, 4
3. With circle points 1, 3, 4
4. With circle points 2, 3, 4

The values of the circle center point coordinates abscissa and ordinate are calculated by adding the abscissa and ordinate values of the four arc calculations and dividing by four.

The radius is calculated by forming the root from the sum of the four radii from the arc calculations and multiplying the result with 0.5.

Syntax

```
<Status>=CALCDAT(<circle points>[<number>,<type>],<number>,<result>)
```

Meaning

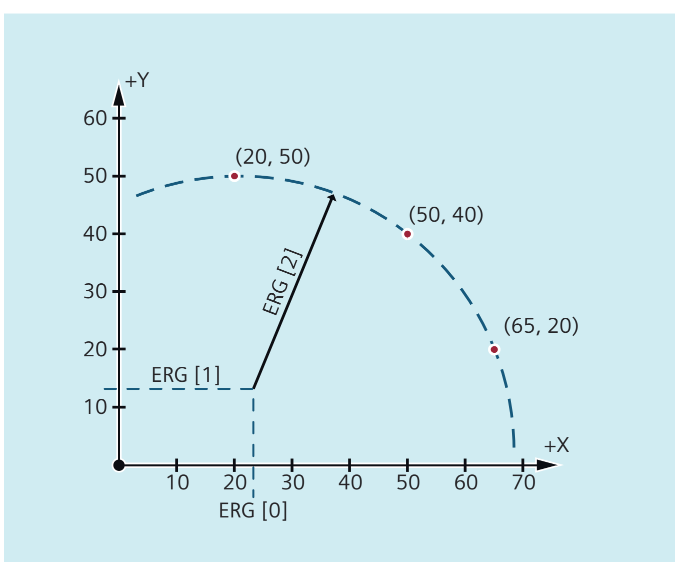
`CALCDAT:`	Predefined function to calculate the radius and center point coordinates of a circle from three or four points		
`<Status>:`	Variable for the circle calculation status		
	Type:	BOOL	
	Value:	TRUE	The specified points lie on a circle.
		FALSE	The specified points **do not** lie on a circle.
`<circle points>[]:`	Variable to specify the circle points using parameters		
	`<number>:`	Number of circle points (3 or 4)	
	`<type>:`	Type of coordinate data, e.g. 2 for 2-point coordinates	
`<number>:`	Parameter for the number of the points used for the calculation (3 or 4)		
`<result>[3]:`	Variable for result: Circle center point coordinates and radius		
	0	Circle center point coordinate: Abscissa value	
	1	Circle center point coordinate: Ordinate value	
	2	Radius	

Note

Please note that the variables must be defined before they are used.

Example

Using three points it should be determined as to whether they are located on a circle segment.



Program code	Comment
N10 DEF REAL PT[3,2]=(20,50,50,40,65,20)	; Variable to specify the points of a circle.
N20 DEF REAL RES[3]	; Variable for result.
N30 DEF BOOL STATUS	; Variable for status.
N40 STATUS=CALCDAT(PKT,3,ERG)	; Call of the determined circle data.
N50 IF STATUS == FALSE GOTOF ERROR	; Jump to error.

4.23.7 Deactivate contour preparation (EXECUTE)

`EXECUTE` deactivates the contour preparation and at the same time the system returns to the normal execution mode.

Syntax

```
EXECUTE(<ERROR>)
```

Meaning

<table>
<tr><td>EXECUTE:</td><td colspan="2">Predefined procedure to terminate contour preparation</td></tr>
<tr><td rowspan="5"><ERROR>:</td><td colspan="2">Variable for the error feedback signal</td></tr>
<tr><td>Type:</td><td>INT</td></tr>
<tr><td colspan="2">The value of the variable indicates whether the contour was able to be prepared error-free:</td></tr>
<tr><td>0</td><td>Error</td></tr>
<tr><td>1</td><td>No error</td></tr>
</table>

Example

```
Program code
...
N30 CONTPRON(...)
N40 G1 X... Z...
...
N100 EXECUTE(...)
...
```

4.24 Programming cycles externally

4.24.1 Technology cycles

4.24.1.1 Fundamentals

Technology cycles

A technology cycle is a predefined NC program in which a specific, generally valid, machining operation, such as tapping of a thread or milling a pocket, is programmed. The adaptation to a specific machining situation is realized using parameters that are transferred to the cycle during the call.

Cycle description

This documentation of the technology cycles only refers to external programming, and is therefore restricted to describing the syntax and parameters Refer to the corresponding commissioning and operating manuals for information about commissioning technology cycles and for handling the specific cycle user interfaces.

Syntax

The program line specified under "syntax" indicates how the cycle call should be programmed.

Special care must be given regarding the following points:

- Correct cycle name
- Call sequence of the transfer parameters

Parameters

All cycle parameters are described with the following data in the table under "Parameters":

- Data type
- Meaning
- Value range
- Dependency on other parameters

The column for reference to the parameter in the screen form is provided to more easily locate values programmed on the control when externally generated cycle calls are recompiled.

Parameters marked with "only for the user interface" are of no significance for the cycle function. They are only needed in order to be able to recompile cycle calls completely. The cycle can be recompiled even if they are not programmed. The fields are then appropriately color-coded, and must be filled out in the screen form.

Parameters marked with "reserved" must be programmed with the value 0 or a comma so that the assignment of the following call parameters matches the internal cycle parameters. Exception: string parameters with the value "" or a comma.

Note

If certain transfer parameters (e.g. <_VARI>, <_GMODE>, <_DMODE>, <_AMODE>) have been indirectly programmed as parameters, the input screen form is opened when recompiling but it cannot be stored as there is no unique assignment to defined selection fields.

Repeating cycles on a position pattern

Drilling and milling cycles can be repeated on the position pattern (modal calls). In cases such as these, MCALL should be written in the same line before the cycle.

Example:

```
...
MCALL CYCLE83(...)
...
```

4.24.1.2 Technology-specific overview

The following overview table lists all available externally programmable technology cycles and the technology assigned to each of them:

Technology	Technology cycle
Drilling	• CYCLE81 - drilling, centering (Page 934) • CYCLE82 - drilling, counterboring (Page 935) • CYCLE85 - reaming (Page 944) • CYCLE86 - boring (Page 945) • CYCLE83 – deep-hole drilling 1 (Page 938) • CYCLE830 - deep-hole drilling 2 (Page 974) • CYCLE84 - tapping without compensating chuck (Page 941) • CYCLE840 - tapping with compensating chuck (Page 982) • CYCLE78 - Drill thread milling (Page 930) • CYCLE802 - arbitrary positions (Page 969) • HOLES1 – row position pattern (Page 897) • CYCLE801 – grid or frame position pattern (Page 967) • HOLES2 – circle or pitch circle position pattern (Page 898)
Turning	• CYCLE951 - stock removal (Page 993) • CYCLE930 - groove (Page 988) • CYCLE940 – undercut form E and F / undercut thread (Page 991) • CYCLE99 - thread turning (Page 954) • CYCLE98 - thread chain (Page 950) • CYCLE92 - cut-off (Page 946)
Contour turning	• CYCLE62 - contour call (Page 916) • CYCLE952 – stock removal / residual stock removal / plunge cutting / residual plunge cutting / plunge turning / residual plunge turning (Page 996) • CYCLE953 - Surface turning (Page 1002)
Milling	• CYCLE61 - Face milling (Page 914) • POCKET3 – rectangular pocket (Page 900) • POCKET4 – circular pocket (Page 902) • CYCLE76 – rectangular spigot (Page 925) • CYCLE77 – circular spigot (Page 928) • CYCLE79 - multi-edge (Page 932) • SLOT1 - longitudinal slot (Page 905) • SLOT2 - circumferential slot (Page 907) • CYCLE899 – open slot (Page 985) • LONGHOLE - elongated hole (Page 910) • CYCLE70 - thread milling (Page 921) • CYCLE60 – engraving (Page 912)

Technology	Technology cycle
Contour milling	• CYCLE62 - contour call (Page 916) • CYCLE72 - Path milling (Page 922) • CYCLE63 – contour pocket milling / contour pocket residual material / contour spigot milling / contour spigot residual material (Page 916) • CYCLE64 - Predrilling contour pocket (Page 919)
Grinding	• CYCLE495 - form-truing (Page 959) • CYCLE435 - Set dresser coordinate system (Page 959) • CYCLE4071 - longitudinal grinding with infeed at the reversal point (Page 1003) • CYCLE4072 - longitudinal grinding with infeed at the reversal point and cancel signal (Page 1005) • CYCLE4073 - longitudinal grinding with continuous infeed (Page 1009) • CYCLE4074 - longitudinal grinding with continuous infeed and cancel signal (Page 1011) • CYCLE4075 - surface grinding with infeed at the reversal point (Page 1014) • CYCLE4077 - surface grinding with infeed at the reversal point and cancel signal (Page 1017) • CYCLE4078 - surface grinding with continuous infeed (Page 1021) • CYCLE4079 - surface grinding with intermittent infeed (Page 1023)
Other	• CYCLE782 - adjust to load (Page 961) • CYCLE800 – swivel plane / swivel tool / align tool (Page 964) • CYCLE805 - Y turning (Page 971) • CYCLE806 - Interpolation turning (Page 973) • CYCLE832 - High-Speed Settings (Page 979)
All	• GROUP_BEGIN - beginning of program block (Page 1026) • GROUP_END - end of program block (Page 1026) • GROUP_ADDEND - End of trial cut addition (Page 1027)

4.24.1.3 HOLES1 – row position pattern

Syntax

```
HOLES1(<SPCA>, <SPCO>, <STA1>, <FDIS>, <DBH>, <NUM>, <_VARI>,
<_UMODE>, <_HIDE>, <_NSP>, <_DMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	X0	<SPCA>	REAL	Reference point for row of holes along the 1st axis (abs)		
2	Y0	<SPCO>	REAL	Reference point for row of holes along the 2nd axis (abs)		
3	α0	<STA1>	REAL	Basic angle of rotation (angle to 1st axis)		
4	L0	<FDIS>	REAL	Distance from 1st hole to reference point		
5	L	<DBH>	REAL	Spacing between the holes		
6	N	<NUM>	INT	Number of holes		
7		<_VARI>	INT	Reserved		
8		<_UMODE>	INT	Reserved		
9		<_HIDE>	STRING [200]	Hidden positions • Max. 198 characters • Specification of consecutive position numbers, e.g. "1,3" (positions 1 and 3 are not executed)		
10		<_NSP>	INT	Reserved		
11		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)

4.24.1.4 HOLES2 – circle or pitch circle position pattern

Syntax

```
HOLES2(<CPA>, <CPO>, <RAD>, <STA1>, <INDA>, <NUM>, <_VARI>,
<_UMODE>, <_HIDE>, <_NSP>, <_DMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning	
1	X0	<CPA>	REAL	Center point for circle of holes along the 1st axis (abs) Reference point in the 1st axis	(for XY) (for XA, YB, ZC)
2	Y0	<CPO>	REAL	Center point for circle of holes along the 2nd axis (abs) Reference point in the 2nd axis	(for XY) (for XA, YB, ZC)
3	R	<RAD>	REAL	Radius of the circle of holes	(for XY)

No.	Parameter mask	Parameter internal	Data type	Meaning			
4	α0	`<STA1>`	REAL	Starting angle or 1st rotary axis position			(for XY) (for XA, YB, ZC)
5	α1	`<INDA>`	REAL	Advance angle (for pitch circle only)			(for XY, XA, YB, ZC)
					< 0 =	Clockwise	
					> 0 =	Counter-clockwise	
6	N	`<NUM>`	INT	Number of positions			
7		`<_VARI>`	INT	Machining type			
				UNITS:	Reserved		
				TENS:	Positioning type		
					0 =	Approach position - linear	
					1 =	Approach position - circular path	
				HUNDREDS:	Reserved		
				THOUSANDS:	Circular pattern		
					0 =	Compatibility mode, if INDA = 0 then full circle, INDA <> 0 then pitch circle	
					1 =	Full circle	
					2 =	Pitch circle	
				TEN THOUSANDS:	Position pattern with rotary axis		
					0 =	XY (without rotary axis)	(for XY)
					1 =	XA (X axis and rotary axis around X)	(only for XA)
					2 =	YB (Y axis and rotary axis around Y)	(only for YB)
					3 =	ZC (Z axis and rotary axis around C)	(only for ZC)
				ONE MILLION + HUNDRED THOUSANDS:	Offset (for several rotary axes around the same axis; if index too large, then 1st axis)		
					00 =	1st A, B or C axis	
					01 =	2nd A, B or C axis	
					...		
					10 =	20th A, B or C axis	
8		`<_UMODE>`	INT	Reserved			
9		`<_HIDE>`	STRING [200]	Reserved			
10		`<_NSP>`	INT	Reserved			
11		`<_DMODE>`	INT	Display mode			
				UNITS:	Machining plane G17/18/19		
					0 =	Compatibility, the plane effective before the cycle call remains active	
					1 =	G17 (only active in the cycle)	
					2 =	G18 (only active in the cycle)	
					3 =	G19 (only active in the cycle)	

4.24.1.5 POCKET3 – rectangular pocket

Syntax

```
POCKET3(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_LENG>, <_WID>, <_CRAD>,
<_PA>, <_PO>, <_STA>, <_MID>, <_FAL>, <_FALD>, <_FFP1>, <_FFD>,
<_CDIR>, <_VARI>, <_MIDA>, <_AP1>, <_AP2>, <_AD>, <_RAD1>, <_DP1>,
<_UMODE>, <_FS>, <_ZFS>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<_RTP>	REAL	Retraction plane (abs)		
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Pocket depth (abs/inc), see <_AMODE>		
5	L	<_LENG>	REAL	Pocket length (inc, to be entered with sign)		
6	W	<_WID>	REAL	Pocket width (inc, to be entered with sign)		
7	R	<_CRAD>	REAL	Corner radius of pocket		
8	X0	<_PA>	REAL	Reference point 1st axis (abs)		
9	YO	<_PO>	REAL	Reference point 2nd axis (abs)		
10	α0	<_STA>	REAL	Angle of rotation, angle between longitudinal axis (L) and 1st axis		
11	DZ	<_MID>	REAL	Maximum depth infeed		
12	UXY	<_FAL>	REAL	Finishing allowance, plane		
13	UZ	<_FALD>	REAL	Finishing allowance, depth		
14	F	<_FFP1>	REAL	Feedrate in the plane		
15	FZ	<_FFD>	REAL	Depth infeed rate		
16		<_CDIR>	INT	Milling direction:	0 =	Down-cut
					1 =	Up-cut
17		<_VARI>	INT	Machining type		
				UNITS:		
					1 =	Roughing
					2 =	Finishing
					4 =	Edge finishing
					5 =	Chamfering
				TENS:		
					0 =	Predrilled, infeed with G0
					1 =	Vertically, infeed with G1
					2 =	Helical
					3 =	Oscillation on pocket longitudinal axis
				HUNDREDS:	Reserved	
18	DXY	<_MIDA>	REAL	Maximum plane infeed, for unit, see <_AMODE>		
19	L1	<_AP1>	REAL	Length of premachining (inc)		

No.	Parameter mask	Parameter internal	Data type	Meaning		
20	W1	<_AP2>	REAL	Width of premachining (inc)		
21	AZ	<_AD>	REAL	Depth of premachining (inc)		
22	ER	<_RAD1>	REAL	Radius of helical path on helical insertion		
	EW			Maximum insertion angle for oscillation		
23	EP	<_DP1>	REAL	Helical pitch on helical insertion		
24		<_UMODE>	INT	Reserved		
25	FS	<_FS>	REAL	Chamfer width (inc)		
26	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE>		
27		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining/only calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining
				THOUSANDS:	Dimensioning via center/corner	
					0 =	Compatibility mode
					1 =	Dimensioning via center
					2 =	Dimensioning of corner point, pocket position +LENG/+WID
					3 =	Dimensioning of corner point, pocket position -LENG/+WID
					4 =	Dimensioning of corner point, pocket position +LENG/-WID
					5 =	Dimensioning of corner point, pocket position -LENG/-WID
				TEN THOUSANDS:	Complete machining/remachining	
					0 =	Compatibility mode (process <_AP1>, <_AP2> and <_AD> as before)
					1 =	Complete machining
					2 =	Post machining

No.	Parameter mask	Parameter internal	Data type	Meaning		
28		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Type of feedrate: G group (G94/G95) for surface and depth feedrate	
					0 =	Compatibility mode
					1 =	G command as before cycle call. G94/G95 same for surface and depth feedrate
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple
29		<_AMODE>	INT	Alternative mode		
				UNITS:	Pocket depth (Z1)	
					0 =	Absolute (compatibility mode)
					1 =	Incremental
				TENS:	Unit for plane infeed (DXY)	
					0 =	mm
					1 =	% of tool diameter
				HUNDREDS:	Insertion depth for chamfering (ZFS)	
					0 =	Absolute
					1 =	Incremental

4.24.1.6 POCKET4 – circular pocket

Syntax

```
POCKET4(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_CDIAM>, <_PA>, <_PO>,
<_MID>, <_FAL>, <_FALD>, <_FFP1>, <_FFD>, <_CDIR>, <_VARI>,
<_MIDA>, <_AP1>, <_AD>, <_RAD1>, <_DP1>, <_UMODE>, <_FS>, <_ZFS>,
<_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<_RTP>	REAL	Retraction plane (abs)		
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Pocket depth (abs/inc), see <_AMODE>		
5	Ø	<_CDIAM>	REAL	Pocket diameter or radius, see <_DMODE>		
6	X0	<_PA>	REAL	Reference point 1st axis (abs)		
7	Y0	<_PO>	REAL	Reference point 2nd axis (abs)		
8	DZ	<_MID>	REAL	Maximum depth setting, see <_VARI> = by planes Maximum helical setting, see <_VARI> = helically		
9	UXY	<_FAL>	REAL	Finishing allowance, plane		
10	UZ	<_FALD>	REAL	Finishing allowance, depth		
11	F	<_FFP1>	REAL	Feedrate for surface machining		
12	FZ	<_FFD>	REAL	Depth infeed rate		
13		<_CDIR>	INT	Milling direction	0 =	Down-cut
					1 =	Up-cut
14		<_VARI>	INT	Machining type		
				UNITS:	Machining	
					1 =	Roughing
					2 =	Finishing
					4 =	Edge finishing
					5 =	Chamfering
				TENS:	Infeed type (roughing and finishing)	
					0 =	Predrilled, infeed with G0 (pocket is premachined)
					1 =	Vertically, infeed with G1
					2 =	Helical
				HUNDREDS:	Reserved	
				THOUSANDS:		
					0 =	Plane-by-plane
					1 =	Helical
15	DXY	<_MIDA>	REAL	Maximum plane infeed, see <_AMODE>, 0 = 0.8 x tool diameter		
16	Ø	<_AP1>	REAL	Diameter/radius of premachining (inc)		
17	AZ	<_AD>	REAL	Depth of premachining (inc)		
18	ER	<_RAD1>	REAL	Radius of helical path on helical insertion		
19	EP	<_DP1>	REAL	Helical pitch on insertion on helical path		
20		<_UMODE>	INT	Reserved		
21	FS	<_FS>	REAL	Chamfer width (inc)		
22	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE>		

No.	Parameter mask	Parameter internal	Data type	Meaning		
23		`<_GMODE>`	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Machining/calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Complete machining/remachining	
					0 =	Compatibility mode (process `<_AP1>` and `<_AD>` as before)
					1 =	Complete machining
					2 =	Post machining
24		`<_DMODE>`	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Type of feedrate: G group (G94/G95) for surface and depth feedrate	
					0 =	Compatibility mode
					1 =	G command as before cycle call. G94/G95 same for surface and depth feedrate
				HUNDREDS:		
					0 =	Compatibility mode (enter `<_CDIAM>/<_AP1>` as radius)
					1 =	Enter `<_CDIAM>/<_AP1>` as diameter
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple

No.	Parameter mask	Parameter internal	Data type	Meaning		
25		<_AMODE>	INT	Alternative mode		
				UNITS:	Pocket depth (Z1)	
					0 =	Absolute (compatibility mode)
					1 =	Incremental
				TENS:	Unit for infeed width (DXY)	
					0 =	mm
					1 =	% of tool diameter
				HUNDREDS:	Insertion depth for chamfering (ZFS)	
					0 =	Absolute
					1 =	Incremental

4.24.1.7 SLOT1 - longitudinal slot

Syntax

```
SLOT1 (<RTP>, <RFP>, <SDIS>, <_DP>, <_DPR>, <NUM>, <LENG>, <WID>,
<_CPA>, <_CPO>, <RAD>, <STA1>, <INDA>, <FFD>, <FFP1>, <_MID>,
<CDIR>, <_FAL>, <VARI>, <_MIDF>, <FFP2>, <SSF>, <_FALD>, <_STA2>,
<_DP1>, <_UMODE>, <_FS>, <_ZFS>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<RTP>	REAL	Retraction plane (abs)		
2	Z0	<RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Slot depth (abs)		
5		<_DPR>	REAL	Slot depth (inc) with respect to Z0 (enter without sign)		
6		<NUM>	INT	Number of slots = 1		
7	L	<LENG>	REAL	Slot length		
8	W	<WID>	REAL	Slot width		
9	X0	<_CPA>	REAL	Reference point in the 1st axis of the plane		
10	Y0	<_CPO>	REAL	Reference point in the 2nd axis of the plane		
11		<RAD>	REAL	Reserved		
12	α	<STA1>	REAL	Angle of rotation		
13		<INDA>	REAL	Reserved		
14	FZ	<FFD>	REAL	Depth infeed rate		
15	F	<FFP1>	REAL	Feedrate		
16	DZ	<_MID>	REAL	Maximum depth infeed		
17		<CDIR>	INT	Milling direction	0 =	Down-cut
					1 =	Up-cut

No.	Parameter mask	Parameter internal	Data type	Meaning		
18	UXY	<_FAL>	REAL	Finishing allowance on plane or slot edge		
19		<VARI>	INT	Machining type		
				UNITS:		
					0 =	Reserved
					1 =	Roughing
					2 =	Finishing
					4 =	Edge finishing (only machine the edge)
					5 =	Chamfering
				TENS:	Approach	
					0 =	Predrilled, infeed with G0 (slot is premachined)
					1 =	Vertically, infeed with G1
					2 =	Helical
					3 =	Oscillation
				HUNDREDS:	Reserved	
20	DZF	<_MIDF>	REAL	Reserved		
21	FF	<FFP2>	REAL	Reserved		
22	SF	<SSF>	REAL	Reserved		
23	UZ	<_FALD>	REAL	Finishing allowance, depth		
24	ER	<_STA2>	REAL	Radius of helical path on helical insertion		
	EW			Maximum insertion angle for oscillation		
25	EP	<_DP1>	REAL	Insertion depth per rev for helix		
26		<_UMODE>	INT	Reserved		
27	FS	<_FS>	REAL	Chamfer width (inc) for chamfering		
28	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE>		
29		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining or just calculation of start point	
					1 =	Normal machining
				THOUSANDS:	Dimensioning of reference point, slot length	
					0 =	Center
					1 =	Inner left-hand +L
					2 =	Inner right-hand -L
					3 =	Left-hand edge +L
					4 =	Right-hand edge -L

No.	Parameter mask	Parameter internal	Data type	Meaning		
30		`<_DMODE>`	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Software version identification	
					1 =	Function extension SLOT1
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple
31		`<_AMODE>`	INT	Alternative mode		
				UNITS:	Final depth Z1 (abs/inc)	
					0 =	Compatibility
					1 =	Z1 (inc)
					2 =	Z1 (abs)
				TENS:	Reserved	
				HUNDREDS:	Insertion depth for chamfering ZFS	
					0 =	ZFS (abs)
					1 =	ZFS (inc)

Note

The cycle is provided with new functions that are not on earlier software versions. Consequently certain parameters in the screen form (`<NUM>`, `<RAD>`, `<INDA>`) are no longer displayed. Multiple slots on one position pattern can be programmed using "MCALL" and calling the desired position pattern, e.g. HOLES2.

4.24.1.8 SLOT2 - circumferential slot

Syntax

```
SLOT2(<RTP>, <RFP>, <SDIS>, <_DP>, <_DPR>, <NUM>, <AFSL>, <WID>,
<_CPA>, <_CPO>, <RAD>, <STA1>, <INDA>, <FFD>, <FFP1>, <_MID>,
<CDIR>, <_FAL>, <VARI>, <_MIDF>, <FFP2>, <SSF>, <_FFCP>, <_UMODE>,
<_FS>, <_ZFS>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<RTP>	REAL	Retraction plane (abs)		
2	Z0	<RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Slot depth (abs)		
5		<_DPR>	REAL	Slot depth (inc) with respect to Z0 (enter without sign)		
6	N	<NUM>	INT	Number of slots		
7	α1	<AFSL>	REAL	Opening angle of the slot		
8	W	<WID>	REAL	Slot width		
9	X0	<_CPA>	REAL	Reference point = Center point of circle, 1st axis of the plane		
10	Y0	<_CPO>	REAL	Reference point = Center point of circle, 2nd axis of the plane		
11	R	<RAD>	REAL	Radius of the circle		
12	α0	<STA1>	REAL	Starting angle		
13	α2	<INDA>	REAL	Incrementing angle		
14	FZ	<FFD>	REAL	Depth infeed rate		
15	F	<FFP1>	REAL	Feedrate		
16	DZ	<_MID>	REAL	Maximum depth infeed		
17		<CDIR>	INT	Milling direction	0 =	Down-cut
					1 =	Up-cut
18	UXY	<_FAL>	REAL	Finishing allowance on plane or slot edge		
19		<VARI>	INT	Machining type		
				UNITS:		
					0 =	Complete machining
					1 =	Roughing
					2 =	Finishing
					3 =	Edge finishing
					5 =	Chamfering
				TENS:		
					0 =	Intermediate positioning with G0 line
					1 =	Intermediate positioning on circular path
				HUNDREDS:	Reserved	
				THOUSANDS:		
					0 =	Compatibility mode, if <INDA> = 0 then full circle, <INDA> <> 0 then pitch circle
					1 =	Full circle
					2 =	Pitch circle
20	DZF	<_MIDF>	REAL	Reserved		
21		<FFP2>	REAL	Reserved		
22		<SSF>	REAL	Reserved		

No.	Parameter mask	Parameter internal	Data type	Meaning		
23	FF	<_FFCP>	REAL	Reserved		
24		<_UMODE>	INT	Reserved		
25	FS	<_FS>	REAL	Chamfer width (inc)		
26	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE>		
27		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining or just calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining
28		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Software version identification	
					1 =	SLOT2 functions as of software version 2.5
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple
29		<_AMODE>	INT	Alternative mode		
				UNITS:	Final depth Z1 (abs/inc)	
					0 =	Compatibility
					1 =	Z1 (inc)
					2 =	Z1 (abs)
				TENS:	Reserved	
				HUNDREDS:	Insertion depth for chamfering ZFS	
					0 =	ZFS (abs)
					1 =	ZFS (inc)

4.24.1.9 LONGHOLE - elongated hole

Syntax

```
LONGHOLE(<RTP>, <RFP>, <SDIS>, <_DP>, <_DPR>, <NUM>, <LENG>,
<_CPA>, <_CPO>, <RAD>, <STA1>, <INDA>, <FFD>, <FFP1>, <MID>,
<_VARI>, <_UMODE>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<RTP>	REAL	Retraction plane (abs)		
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Long hole depth (abs)		
5		<_DPR>	REAL	Long hole depth (inc) with respect to Z0 (enter without sign)		
6		<NUM>	INT	Number of long holes = 1		
7	L	<LENG>	REAL	Length of long hole		
8	X0	<_CPA>	REAL	Reference point 1st axis of the plane		
9	Y0	<_CPO>	REAL	Reference point 2nd axis of the plane		
10		<RAD>	REAL	Reserved		
11	α0	<STA1>	REAL	Angle of rotation		
12		<INDA>	REAL	Reserved		
13	FZ	<FFD>	REAL	Depth infeed rate		
14	F	<FFP1>	REAL	Feedrate		
15	DZ	<MID>	REAL	Maximum depth infeed		
16		<_VARI>	INT	Machining type		
				UNITS:	Infeed type	
					1 =	Vertically with G1
					3 =	Oscillation
				HUNDREDS:	Reserved	
17		<_UMODE>	INT	Reserved		

No.	Parameter mask	Parameter internal	Data type	Meaning		
18		`<_GMODE>`	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining or just calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining
				THOUSANDS:	Dimensioning of reference point, slot length	
					0 =	Center
					1 =	Inner left-hand +L
					2 =	Inner right-hand -L
					3 =	Left-hand edge +L
					4 =	Right-hand edge -L
19		`<_DMODE>`	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Type of feedrate: G group (G94/G95) for surface and depth feedrate	
					0 =	Compatibility mode
					1 =	G command as before cycle call. G94/G95 same for surface and depth feedrate
				HUNDREDS:	Reserved	
				THOUSANDS:	Software version identification	
					1 =	Function extension LONGHOLE (dimensioning of reference point)
20		`<_AMODE>`	INT	Alternative mode		
				UNITS:	Final depth Z1 (abs/inc)	
					0 =	Compatibility
					1 =	Z1 (inc)
					2 =	Z1 (abs)

Note

The cycle is provided with new functions that are not on earlier software versions. Consequently certain parameters in the screen form (`<NUM>, <RAD>, <INDA>`) are no longer displayed. Multiple slots on one position pattern can be programmed using "MCALL" and calling the desired position pattern, e.g. HOLES2.

4.24.1.10 CYCLE60 – engraving

Syntax

```
CYCLE60(<_TEXT>, <_RTP>, <_RFP>, <_SDIS>, <_DP>, <_DPR>, <_PA>,
<_PO>, <_STA>, <_CP1>, <_CP2>, <_WID>, <_DF>, <_FFD>, <_FFP1>,
<_VARI>, <_CODEP>, <_UMODE>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1		<_TEXT>	STRING [200]	Text to be engraved (up to 100 characters)
2	RP	<_RTP>	REAL	Retraction plane (abs)
3	Z0	<_RFP>	REAL	Reference point of tool axis (abs)
4	SC	<_SDIS>	REAL	Safety clearance (to be added to the reference plane, enter without sign)
5	Z1	<_DP>	REAL	Depth (abs), see <_AMODE>
6	Z1	<_DPR>	REAL	Depth (inc), see <_AMODE>
7	X0	<_PA>	REAL	Reference point 1st axis of plane (abs) - right-angled, see <_VARI>
	R			Reference point, length (radius) - polar, see <_VARI>
8	Y0	<_PO>	REAL	Reference point 2nd axis of plane (abs) - right-angled, see <_VARI>
	α0			Reference point, angle with respect to 1st axis - polar, see <_VARI>
9	α1	<_STA>	REAL	Text direction, angle of line of text with respect to 1st axis), see <_VARI>
10	XM	<_CP1>	REAL	Center of the text circle, 1st axis of plane (abs) - right-angled, see <_VARI>
	LM			Center of circle of text, length (radius) with respect to WNP - polar, see <_VARI>
11	YM	<_CP2>	REAL	Center of the text circle, 2nd axis of plane (abs) - right-angled, see <_VARI>
	αM			Center of text circle, angle with respect to 1st axis axis - polar, see <_VARI>
12	W	<_WID>	REAL	Height of characters (enter without sign)
13	DX1 DX2	<_DF>	REAL	Distance between characters / overall width, see <_VARI>
	α2			Opening angle, see <_VARI>
14	FZ	<_FFD>	REAL	Depth infeed rate, see <_DMODE>
15	F	<_FFP1>	REAL	Feedrate for surface machining

No.	Parameter mask	Parameter internal	Data type	Meaning		
16		`<_VARI>`	INT	Machining (alignment and reference point for engraved text)		
				UNITS:	Reference point	
					0 =	Right-angled
					1 =	Polar
				TENS:	Text alignment	
					0 =	Text on one line
					1 =	Text in an upward pointing arc
					2 =	Text in a downward curving arc
				HUNDREDS:	Reserved	
				THOUSANDS:	Reference point of the text, horizontal	
					0 =	Left
					1 =	Center
					2 =	Right
				TEN THOUSANDS:	Reference point of the text, vertical	
					0 =	Bottom
					1 =	Center
					2 =	Top
				HUNDRED THOUSANDS:	Text length	
					0 =	Character spacing
					1 =	Overall text width (linear text only)
					2 =	Opening angle (only for circular text)
				ONE MILLION:	Circle center	
					0 =	Right-angled (Cartesian)
					1 =	Polar
				TEN MILLIONS:	Mirror writing	
					0 =	Compatibility
					1 =	Mirror writing ON
					2 =	Mirror writing OFF
17		`<_CODEP>`	INT	Code page number for writing (currently only 1252)		
18		`<_UMODE>`	INT	Reserved		
19		`_GMODE>`	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining/only calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining

<table>
<tr><th>No.</th><th>Parameter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td rowspan="9">20</td><td rowspan="9"></td><td rowspan="9"><_DMODE></td><td rowspan="9">INT</td><td colspan="3">Display mode</td></tr>
<tr><td rowspan="5">UNITS:</td><td colspan="2">Machining plane G17/18/19</td></tr>
<tr><td>0 =</td><td>Compatibility, the plane effective before the cycle call remains active</td></tr>
<tr><td>1 =</td><td>G17 (only active in the cycle)</td></tr>
<tr><td>2 =</td><td>G18 (only active in the cycle)</td></tr>
<tr><td>3 =</td><td>G19 (only active in the cycle)</td></tr>
<tr><td rowspan="3">TENS:</td><td colspan="2">Type of feedrate: G group (G94/G95) for surface and depth feedrate</td></tr>
<tr><td>0 =</td><td>Compatibility mode</td></tr>
<tr><td>1 =</td><td>G command as before cycle call. G94/G95 same for surface and depth feedrate</td></tr>
<tr><td rowspan="5">21</td><td rowspan="5"></td><td rowspan="5"><_AMODE></td><td rowspan="5">INT</td><td colspan="3">Alternative mode</td></tr>
<tr><td rowspan="4">UNITS:</td><td colspan="2">Final depth (<_DP>, <_DPR>)</td></tr>
<tr><td>0 =</td><td>Compatibility</td></tr>
<tr><td>1 =</td><td>Incremental (<_DPR>)</td></tr>
<tr><td>2 =</td><td>Absolute (<_DP>)</td></tr>
</table>

4.24.1.11 CYCLE61 - Face milling

Syntax

```
CYCLE61(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_PA>, <_PO>, <_LENG>,
<_WID>, <_MID>, <_MIDA>, <_FALD>, <_FFP1>, <_VARI>, <_LIM>,
<_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<_RTP>	REAL	Retraction plane (abs)
2	Z0	<_RFP>	REAL	Reference point of tool axis, height of blank (abs)
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)
4	Z1	<_DP>	REAL	Height of finished part (abs/inc), see <_AMODE>
5	X0	<_PA>	REAL	Corner point 1 in 1st axis (abs)
6	Y0	<_PO>	REAL	Corner point 1 in 2nd axis (abs)
7	X1	<_LENG>	REAL	Corner point 2 in 1st axis (abs/inc), see <_AMODE>
8	Y1	<_WID>	REAL	Corner point 2 in 2nd axis (abs/inc), see <_AMODE>
9	DZ	<_MID>	REAL	Maximum depth infeed
10	DXY	<_MIDA>	REAL	Maximum plane infeed (for unit, see <_AMODE>)
11	UZ	<_FALD>	REAL	Finishing allowance, depth

No.	Parameter mask	Parameter internal	Data type	Meaning		
12	F	<_FFP1>	REAL	Machining feedrate		
13		<_VARI>	INT	Machining type		
				UNITS:	Machining	
					1 =	Roughing
					2 =	Finishing
				TENS:	Machining direction	
					1 =	Parallel to the 1st axis, in one direction
					2 =	Parallel to the 2nd axis, in one direction
					3 =	Parallel to the 1st axis, varying direction
					4 =	Parallel to the 2nd axis, varying direction
14		<_LIM>	INT	Limits		
				UNITS:	Limit 1st axis negative	
					0 =	No
					1 =	Yes
				TENS:	Limit 1st axis positive	
					0 =	No
					1 =	Yes
				HUNDREDS:	Limit 2nd axis negative	
					0 =	No
					1 =	Yes
				THOUSANDS:	Limit 2nd axis positive	
					0 =	No
					1 =	Yes
15		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)

No.	Parameter mask	Parameter internal	Data type	Meaning		
16		<_AMODE>	INT	Alternative mode		
				UNITS:	Final depth (<_DP>)	
					0 =	Absolute
					1 =	Incremental
				TENS:	Units for plane infeed (<_MIDA>)	
					0 =	mm
					1 =	% of tool diameter
				HUNDREDS:	Reserved	
				THOUSANDS:	Length of surface	
					0 =	Incremental
					1 =	Absolute
				TEN THOUSANDS:	Width of surface	
					0 =	Incremental
					1 =	Absolute

4.24.1.12 CYCLE62 - contour call

Syntax

```
CYCLE62(<_KNAME>,<_TYPE>, <_LAB1>, <_LAB2>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	PRG/CON	<_KNAME>	STRING [140]	Contour name or subprogram name does not have to be programmed in _TYPE = 2		
2		<_TYPE>	INT	Determination of contour input		
					0 =	Subprogram
					1 =	Contour name
					2 =	Labels
					3 =	Labels in the subprogram
3	LAB1	<_LAB1>	STRING[32]	Label 1, start of contour		
4	LAB2	<_LAB2>	STRING[32]	Label 2, end of contour		

4.24.1.13 CYCLE63 – contour pocket milling / contour pocket residual material / contour spigot milling / contour spigot residual material

Syntax

```
CYCLE63(<_PRG>, <_VARI>, <_RP>, <_Z0>, <_SC>, <_Z1>, <_F>, <_FZ>,
<_DXY>, <_DZ>, <_UXY>, <_UZ>, <_CDIR>, <_XS>, <_YS>, <_ER>, <_EP>,
```

```
<_EW>, <_FS>, <_ZFS>, <_TR>, <_DR>, <_UMODE>, <_GMODE>, <_DMODE>,
<_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	PRG	<_PRG>	STRING [100]	Name of removal program		
2		<_VARI>	INT	Machining type		
				UNITS:	Machining process	
					1 =	Roughing
					3 =	Base finishing
					4 =	Edge finishing
					5 =	Chamfering
				TENS:	Infeed type	
					0 =	Central insertion
					1 =	Helical insertion
					2 =	Oscillating insertion
				HUNDREDS:	Reserved	
				THOUSANDS:	Lift mode	
					0 =	Lift off to retraction plane
					1 =	Lift off to reference point + safety clearance
				TEN THOUSANDS:	Start point for roughing and finishing base	
					0 =	Auto
					1 =	Manual
3	RP	<_RP>	REAL	Retraction plane (abs)		
4	Z0	<_Z0>	REAL	Reference point of tool axis (abs)		
5	SC	<_SC>	REAL	Safety clearance (to be added to reference point, enter without sign)		
6	Z1	<_Z1>	REAL	Final depth (see <_AMODE> UNITS)		
7	F	<_F>	REAL	Feedrate in the plane during roughing/finishing		
8	FZ	<_FZ>	REAL	Depth infeed rate		
9	DXY	<_DXY>	REAL	Infeed plane - unit (see <_AMODE> TENS)		
10	DZ	<_DZ>	REAL	Depth infeed		
11	UXY	<_UXY>	REAL	Finishing allowance, plane		
12	UZ	<_UZ>	REAL	Finishing allowance, depth		
13		<_CDIR>	INT	Milling direction	0 =	Down-cut
					1 =	Up-cut
14	XS	<_XS>	REAL	Starting point X, absolute		
15	YS	<_YS>	REAL	Starting point Y, absolute		
16	ER	<_ER>	REAL	Helical insertion: Radius		
17	EP	<_EP>	REAL	Helical insertion: Pitch		
18	EW	<_EW>	REAL	Oscillating insertion: Maximum insertion angle		

No.	Parameter mask	Parameter internal	Data type	Meaning		
19	FS	<_FS>	REAL	Chamfer width (inc) for chamfering		
20	ZFS	<_ZFS>	REAL	Insertion depth of tool tip when chamfering (see <_AMODE> HUNDREDS)		
21	TR	<_TR>	STRING[32]	Reference tool name when machining residual material		
22	DR	<_DR>	INT	Reference tool D number when machining residual material		
23		<_UMODE>	INT	Reserved		
24		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining/only calculation of start point	
					0 =	Normal machining (no compatibility mode needed)
					1 =	Normal machining
					2 =	Reserved
25		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Reserved	
				HUNDREDS:	Technology mode	
					1 =	Pocket
					2 =	Spigot
				THOUSANDS:	Machine residual material	
					0 =	No
					1 =	Yes
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple
				HUNDRED THOUSANDS:	Automatic program name	
					0 =	No
					1 =	Yes

No.	Parameter mask	Parameter internal	Data type	Meaning		
26		<_AMODE>	INT	Alternative mode		
				UNITS:	Final depth (Z1)	
					0 =	Absolute (compatibility mode)
					1 =	Incremental
				TENS:	Unit for plane infeed (DXY)	
					0 =	mm
					1 =	% of tool diameter
				HUNDREDS:	Insertion depth for chamfering (ZFS)	
					0 =	Absolute
					1 =	Incremental
				THOUSANDS:	---	Reserved

4.24.1.14 CYCLE64 - Predrilling contour pocket

Syntax

```
CYCLE64(<_PRG>, <_VARI>, <_RP>, <_Z0>, <_SC>, <_Z1>, <_F>, <_DXY>,
<_UXY>, <_UZ>, <_CDIR>, <_TR>, <_DR>, <_UMODE>, <_GMODE>, <_DMODE>,
<_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	PRG	<_PRG>	STRING [100]	Name of drilling/centering program		
2		<_VARI>	INT	Machining type		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Lift mode	
					0 =	Lift off to retraction plane
					1 =	Lift off to reference point + safety clearance
3	RP	<_RP>	REAL	Retraction plane (abs)		
4	Z0	<_Z0>	REAL	Reference point (abs)		
5	SC	<_SC>	REAL	Safety clearance (to be added to reference point, enter without sign)		
6	Z1	<_Z1>	REAL	Drilling/centering depth (see <_AMODE> UNITS)		
7	F	<_F>	REAL	Drilling/centering feedrate		
8	DXY	<_DXY>	REAL	Infeed plane - unit (see <_AMODE> TENS)		
9	UXY	<_UXY>	REAL	Finishing allowance, plane		
10	UZ	<_UZ>	REAL	Finishing allowance, depth		

No.	Parameter mask	Parameter internal	Data type	Meaning		
11		<_CDIR>	INT	Milling direction	0 =	Down-cut
					1 =	Up-cut
12	TR	<_TR>	STRING[20]	Reference tool name		
13	DR	<_DR>	INT	Reference tool D number		
14		<_UMODE>	INT	Reserved		
15		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining/only calculation of start point	
					0 =	Normal machining (no compatibility mode needed)
					1 =	Normal machining
					2 =	Reserved
25		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Technology mode	
					1 =	Predrilling
					2 =	Centering
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	---	Reserved
				HUNDRED THOUSANDS:	Automatic program name	
					0 =	No
					1 =	Yes
26		<_AMODE>	INT	Alternative mode		
				UNITS:	Drilling/centering depth Z1	
					0 =	Absolute (compatibility mode)
					1 =	Incremental
				TENS:	Unit for plane infeed (DXY)	
					0 =	mm
					1 =	% of tool diameter

4.24.1.15 CYCLE70 - thread milling

Syntax

```
CYCLE70(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_DIATH>, <_H1>, <_FAL>,
<_PIT>, <_NT>, <_MID>, <_FFR>, <_TYPTH>, <_PA>, <_PO>, <_NSP>,
<_VARI>, <_PITA>, <_PITM>, <_PTAB>, <_PTABA>, <_GMODE>, <_DMODE>,
<_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<_RTP>	REAL	Retraction plane (abs)		
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Thread length (abs, inc), see <_AMODE> Take account of runout at base of hole (at least half pitch)		
5	∅	<_DIATH>	REAL	Nominal diameter of the thread		
6	H1	<_H1>	REAL	Thread depth		
7	U	<_FAL>	REAL	Finishing allowance		
8	P	<_PIT>	REAL	Pitch (select <_PITA>: mm, inch, MODULE, threads/inch)		
9	NT	<_NT>	INT	Number of teeth on the tool tip Tool length is always with respect to bottom tooth.		
10	DXY	<_MID>	REAL	Maximum infeed per cut <_MID> > <_H1>: All in one cut		
11	F	<_FFR>	REAL	Milling feed		
12		<_TYPTH>	INT	Thread type	0 =	Internal thread
					1 =	External thread
13	X0	<_PA>	REAL	Circle center 1st axis (abs)		
14	Y0	<_PO>	REAL	Circle center 2nd axis (abs)		
15	αS	<_NSP>	REAL	Start angle (multi-start thread)		
16		<_VARI>	INT	Machining type		
				UNITS:		
					1 =	Roughing
					2 =	Finishing
				TENS:		
					1 =	From top to bottom
					2 =	From bottom to top
				HUNDREDS:		
					0 =	Right-hand thread
					1 =	Left-hand thread

No.	Parameter mask	Parameter internal	Data type	Meaning		
17		<_PITA>	INT	Evaluation of thread pitch		
					0 =	Compatibility mode
					1 =	Pitch in mm
					2 =	Pitch in threads per inch (TPI)
					3 =	Pitch in inches
					4 =	Pitch as MODULE
18		<_PITM>	STRING[15]	String as marker for pitch input (for the interface only)		
19		<_PTAB>	STRING[20]	String for thread table ("", "ISO", "BSW", "BSP", "UNC") (for the interface only)		
20		<_PTABA>	STRING[20]	String for selection from thread table (e.g. "M 10", "M 12", ...) (for the interface only)		
21		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Machining/calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining
22		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
23		<_AMODE>	INT	Alternative mode		
				UNITS:	Thread length (<_DP>)	
					0 =	Absolute
					1 =	Incremental

4.24.1.16 CYCLE72 - Path milling

Syntax

```
CYCLE72(<_KNAME>, <_RTP>, <_RFP>, <_SDIS>, <_DP>, <_MID>, <_FAL>,
<_FALD>, <_FFP1>, <_FFD>, <_VARI>, <_RL>, <_AS1>, <_LP1>, <_FF3>,
<_AS2>, <_LP2>,<_UMODE>, <_FS>, <_ZFS>, <_GMODE>, <_DMODE>,
<_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1		`<_KNAME>`	STRING [141]	Name of the contour subprogram		
2	RP	`<_RTP>`	REAL	Retraction plane (abs)		
3	Z0	`<_RFP>`	REAL	Reference point of tool axis (abs)		
4	SC	`<_SDIS>`	REAL	Safety clearance (to be added to reference point, enter without sign)		
5	Z1	`<_DP>`	REAL	End point, final depth (abs/inc), see `<_AMODE>`		
6	DZ	`<_MID>`	REAL	Maximum depth infeed (inc; enter without sign)		
7	UXY	`<_FAL>`	REAL	Finishing allowance, plane (inc), allowance at edge contour		
8	UZ	`<_FALD>`	REAL	Finishing allowance depth (inc), allowance at base (enter without sign)		
9	FX	`<_FFP1>`	REAL	Feedrate on contour		
10	FZ	`<_FFD>`	REAL	Feedrate for depth infeed (or spatial infeed)		
11		`<_VARI>`	INT	Machining type		
				UNITS:	Machining	
					1 =	Roughing
					2 =	Finishing
					5 =	Chamfering
				TENS:		
					0 =	Intermediate paths with G0
					1 =	Intermediate paths with G1
				HUNDREDS:	Retraction at the end of contour	
					0 =	Retraction at the end of contour to reference point
					1 =	Retraction at the end of contour to reference point + `<_SDIS>`
					2 =	Retraction at the end of contour by `<_SDIS>`
					3 =	No retraction at the end of contour, approach next start point with contour feed
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Machine contour	
					0 =	Machine contour forward
					1 =	Machine contour backward Restrictions with backward machining: • Max 170 contour elements (including chamfers or rounding) • Only values in the (X/Y) and F planes are evaluated

No.	Parameter mask	Parameter internal	Data type	Meaning		
12		<_RL>	INT	Machining direction		
					40 =	Center of contour (G40, approach and retract: straight line or vertical)
					41 =	Left of contour (G41, approach and retract: straight line or circle)
					42 =	Right of contour (G42, approach and retract: straight line or circle)
13		<_AS1>	INT	Contour approach movement		
				UNITS:		
					1 =	Straight line
					2 =	Quadrant
					3 =	Semi-circle
					4 =	Approach and retraction vertically
				TENS:		
					0 =	Last movement, in the plane
					1 =	Last movement, spatial
14	L1	<_LP1>	REAL	Approach path or approach radius (inc; enter without sign)		
15	FZ	<_FF3>	REAL	Feedrate for intermediate paths (G94/G95 as to contour)		
16		<_AS2>	INT	Contour approach movement (not vertical approach/retract)		
				UNITS:		
					1 =	Straight line
					2 =	Quadrant
					3 =	Semi-circle
				TENS:		
					0 =	Last movement, in the plane
					1 =	Last movement, spatial
17	L2	<_LP2>	REAL	Retract path or retract radius (inc, to be entered without sign)		
18		<_UMODE>	INT	Reserved		
19	FS	<_FS>	REAL	Chamfer width (inc)		
20	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE>		
21		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining/only calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining

No.	Parameter mask	Parameter internal	Data type	Meaning		
22		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Type of feedrate: G group (G94/G95) for surface and depth feedrate	
					0 =	Compatibility mode
					1 =	G command as before cycle call. G94/G95 same for surface and depth feedrate
				THOUSANDS:		
					0 =	Compatibility mode: Contour name is in <_KNAME>
					1 =	Contour name is programmed in CYCLE62 and transferred to _SC_CONT_NAME
23		<_AMODE>	INT	Alternative mode		
				UNITS:	End point Z1 (<_DP>)	
					0 =	Absolute (compatibility mode)
					1 =	Incremental
				TENS:	Units for plane infeed	
					0 =	mm, inch
					1 =	Reserved
				HUNDREDS:	Insertion depth for chamfering (<_ZFS>)	
					0 =	Absolute
					1 =	Incremental

Note

If the following transfer parameters are programmed indirectly (as parameters), the screen form is not reset:

```
<_VARI>, <_RL>, <_AS1>, <_AS2>, <_UMODE>, <_GMODE>, <_DMODE>,
<_AMODE>
```

4.24.1.17 CYCLE76 – rectangular spigot

Syntax

```
CYCLE76(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_DPR>, <_LENG>, <_WID>,
<_CRAD>, <_PA>, <_PO>, <_STA>, <_MID>, <_FAL>, <_FALD>, <_FFP1>,
```

```
<_FFD>, <_CDIR>, <_VARI>, <_AP1>, <_AP2>, <_FS>, <_ZFS>, <_GMODE>,
<_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<_RTP>	REAL	Retraction plane (abs)
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)
4	Z1	<_DP>	REAL	Spigot depth (abs)
5		<_DPR>	REAL	Spigot depth (inc) with respect to Z0 (enter without sign)
6	L	<_LENG>	REAL	Spigot length, see <_GMODE> (enter without sign)
7	W	<_WID>	REAL	Spigot width, see <_GMODE> (enter without sign)
8	R	<_CRAD>	REAL	Spigot corner radius (enter without sign)
9	X0	<_PA>	REAL	Reference point for spigot in 1st axis of plane (abs)
10	Y0	<_PO>	REAL	Reference point for spigot in 2nd axis of plane (abs)
11	α0	<_STA>	REAL	Angle of rotation, angle between longitudinal axis (L) and 1st axis of plane
12	DZ	<_MID>	REAL	Maximum depth infeed (inc; enter without sign)
13	UXY	<_FAL>	REAL	Finishing allowance, plane (inc), allowance at edge contour
14	UZ	<_FALD>	REAL	Finishing allowance depth (inc), allowance at base (enter without sign)
15	FX	<_FFP1>	REAL	Feedrate on contour
16	FZ	<_FFD>	REAL	Depth infeed rate
17		<_CDIR>	INT	Milling direction (enter without sign) UNITS: 0 = Down-cut 1 = Up-cut
18		<_VARI>	INT	Machining UNITS: 1 = Roughing 2 = Finishing 5 = Chamfering
19	L1	<_AP1>	REAL	Length of blank spigot
20	W1	<_AP2>	REAL	Width of blank spigot
21	FS	<_FS>	REAL	Chamfer width (inc)
22	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs, inc), see <_AMODE>

No.	Parameter mask	Parameter internal	Data type	Meaning		
23		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining or just calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining
				THOUSANDS:	Dimensioning of spigot acc. to center or corner	
					0 =	Compatibility mode
					1 =	Dimensioning via center
					2 =	Dimensioning of corner point, spigot +L +W
					3 =	Dimensioning of corner point, spigot -L +W
					4 =	Dimensioning of corner point, spigot +L -W
					5 =	Dimensioning of corner point, spigot -L -W
				TEN THOUSANDS:	Complete machining or remachining	
					0 =	Compatibility mode
					1 =	Complete machining
					2 =	Post machining
24		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	---	Reserved
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple

No.	Parameter mask	Parameter internal	Data type	Meaning		
25		<_AMODE>	INT	Alternative mode		
				UNITS:	Final depth Z1 (DP)	
					0 =	Compatibility
					1 =	Incremental
					2 =	Absolute
				TENS:	Reserved	
				HUNDREDS:	Insertion depth for chamfering (ZFS)	
					0 =	Absolute
					1 =	Incremental

4.24.1.18 CYCLE77 – circular spigot

Syntax

```
CYCLE77(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_DPR>, <_CDIAM>, <_PA>,
<_PO>, <_MID>, <_FAL>, <_FALD>, <_FFP1>, <_FFD>, <_CDIR>, <_VARI>,
<_AP1>, <_FS>, <_ZFS>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<_RTP>	REAL	Retraction plane (abs)		
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Spigot depth (abs)		
5		<_DPR>	REAL	Spigot depth (inc) with respect to Z0 (enter without sign)		
6	Ø	<_CDIAM>	REAL	Spigot diameter (enter without sign)		
7	X0	<_PA>	REAL	Reference point for spigot in 1st axis of plane (abs)		
8	Y0	<_PO>	REAL	Reference point for spigot in 2nd axis of plane (abs)		
9	DZ	<_MID>	REAL	Maximum depth infeed (inc; enter without sign)		
10	UXY	<_FAL>	REAL	Finishing allowance, plane (inc), allowance at edge contour		
11	UZ	<_FALD>	REAL	Finishing allowance depth (inc), allowance at base (enter without sign)		
12	FX	<_FFP1>	REAL	Feedrate on contour		
13	FZ	<_FFD>	REAL	Depth infeed rate		
14		<_CDIR>	INT	Milling direction (enter without sign)		
				UNITS:		
					0 =	Down-cut
					1 =	Up-cut

No.	Parameter mask	Parameter internal	Data type	Meaning		
15		<_VARI>	INT	Machining type		
				UNITS:	Machining	
					1 =	Roughing to final machining allowance
					2 =	Finishing (allowance X/Y/Z=0)
					5 =	Chamfering
16	Ø1	<_AP1>	REAL	Diameter of blank spigot		
17	FS	<_FS>	REAL	Chamfer width (inc)		
18	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE>		
19		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining/only calculation of start point	
					0 =	Compatibility mode
					1 =	Normal machining
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Complete machining/remachining	
					0 =	Compatibility mode (process <_AP1> as before)
					1 =	Complete machining
					2 =	Post machining
20		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	---	Reserved
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple

No.	Parameter mask	Parameter internal	Data type	Meaning		
21		<_AMODE>	INT	Alternative mode		
				UNITS:	Final depth Z1 (DP)	
					0 =	Absolute (compatibility mode)
					1 =	Incremental
					2 =	Absolute
				TENS:	Reserved	
				HUNDREDS:	Insertion depth for chamfering (ZFS)	
					0 =	Absolute
					1 =	Incremental

4.24.1.19 CYCLE78 - Drill thread milling

Syntax

```
CYCLE78(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_ADPR>, <_FDPR>, <_LDPR>,
<_DIAM>, <_PIT>, <_PITA>, <_DAM>, <_MDEP>, <_VARI>, <_CDIR>,
<_GE>, <_FFD>, <_FRDP>, <_FFR>, <_FFP2>, <_FFA>, <_PITM>, <_PTAB>,
<_PTABA>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<_RTP>	REAL	Retraction plane (abs)		
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Final drilling depth (abs/inc), see <_AMODE>		
5		<_ADPR>	REAL	Predrilling depth with reduced drilling feedrate (inc) effective with <_VARI> TEN THOUSANDS		
6	D	<_FDPR>	REAL	Maximum depth infeed (inc) D ≥ Z1 ⇒ One infeed to the final drilling depth D < Z1 ⇒ Deep drilling cycle with multiple infeeds and chip removal		
7	ZR	<_LDPR>	REAL	Remaining drilling depth when through-drilling (inc) with FR feed		
8	∅	<_DIAM>	REAL	Nominal diameter of the thread		
9	P	<_PIT>	REAL	Pitch as a numerical value		
10		<_PITA>	INT	Evaluation of thread pitch P		
					1 =	Pitch in mm/rev
					2 =	Pitch in threads/inch
					3 =	Pitch in inch/rev
					4 =	Pitch as MODULE
11	DF	<_DAM>	REAL	Absolute value / percentage for each additional infeed (degression), see<_AMODE>		
12	V1	<_MDEP>	REAL	Minimum infeed (inc), only active for degression		

No.	Parameter mask	Parameter internal	Data type	Meaning		
13		<_VARI>	INT	Machining type		
				UNITS:	Reserved	
				TENS:	Swarf removal before thread milling	
					0 =	No chip removal before thread milling (only active at final drilling depth)
					1 =	Chip removal before thread milling (only active at final drilling depth)
				HUNDREDS:	Right-hand/left-hand threads	
					0 =	Right-hand thread
					1 =	Left-hand thread
				THOUSANDS:	Remaining drilling depth with drilling feedrate	
					0 =	No remaining drilling depth with drilling feedrate FR
					1 =	Remaining drilling depth with drilling feedrate FR
				TEN THOUSANDS:	Predrilling with reduced feedrate	
					0 =	No predrilling with reduced feedrate
					1 =	Predrilling with reduced feedrate Predrilling feedrate = 0.3 F1, if F1 < 0.15 mm/rev Predrilling feedrate = 0.1 mm/rev, if F1 ≥ 0.15 mm/rev
14		<_CDIR>	INT	Milling direction	0 =	Down-cut
					1 =	Up-cut
					4 =	Up-cut + down-cut (combined roughing + finishing)
15	Z2	<_GE>	REAL	Retraction distance before thread milling (inc)		
16	F1	<_FFD>	REAL	Drilling feedrate (mm/min or in/min or mm/rev)		
17	FR	<_FRDP>	REAL	Drilling feedrate for remaining drilling depth (mm/min or mm/rev)		
18	F2	<_FFR>	REAL	Feedrate for thread milling (mm/min or mm/tooth)		
19	FS	<_FFP2>	REAL	Finishing feedrate for <_CDIR> =4 (mm/min or mm/tooth)		
20		<_FFA>	INT	Evaluation of feedrates		
				UNITS:	Drilling feed F1	
				TENS:	Drilling feedrate for remaining drilling depth FR	
				HUNDREDS:	Feedrate for thread milling F2	
				THOUSANDS:	Finishing feedrate FS	
21		<_PITM>	STRING[15]	String as marker for pitch input (for the interface only)1)		
22		<_PTAB>	STRING[20]	String for thread table ("", "ISO", "BSW", "BSP", "UNC") (for the interface only)1)		
23		<_PTABA>	STRING[20]	String for selection from thread table (e.g. "M 10", "M 12", ...) (for the interface only)1)		

No.	Parameter mask	Parameter internal	Data type	Meaning		
24		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data), reserved		
25		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
26		<_AMODE>	INT	Alternative mode		
				UNITS:	Drilling depth = Final drilling depth Z1 abs/inc	
					0 =	Absolute
					1 =	Incremental
				TENS:	Absolute value / percentage DF for each additional infeed (degression)	
					0 =	Absolute value
					1 =	Percentage (0.001 to 100%)

Note

1) Parameters 21, 22 and 23 are only used for thread selection in the screen form thread tables. The thread tables cannot be accessed via cycle definition in the cycle run time.

4.24.1.20 CYCLE79 - multi-edge

Syntax

```
CYCLE79(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_NUM>, <_SWL>, <_PA>,
<_PO>, <_STA>, <_RC>, <_AP1>, <_MIDA>, <_MID>, <_FAL>, <_FALD>,
<_FFP1>, <_CDIR>, <_VARI>, <_FS>, <_ZFS>, <_GMODE>, <_DMODE>,
<_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<_RTP>	REAL	Retraction plane (abs)
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)
4	Z1	<_DP>	REAL	Multiple-edge depth (abs/inc), see <_AMODE>
5	N	<_NUM>	INT	Number of edges (1...n)

<table>
<tr><th>No.</th><th>Parameter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td>6</td><td>SW/L</td><td><_SWL></td><td>REAL</td><td colspan="3">Width across flats or edge length (depending on <_VARI>)
("SW" for width across flats, "L" for edge length)
Width across flats only if even number of edges, and single edge</td></tr>
<tr><td>7</td><td>X0</td><td><_PA></td><td>REAL</td><td colspan="3">Spigot reference point, 1st axis (abs)</td></tr>
<tr><td>8</td><td>Y0</td><td><_PO></td><td>REAL</td><td colspan="3">Spigot reference point, 2nd axis (abs)</td></tr>
<tr><td>9</td><td>α0</td><td><_STA></td><td>REAL</td><td colspan="3">Angle of rotation, center of edge against 1st axis (X axis)</td></tr>
<tr><td>10</td><td>R1/FS1</td><td><_RC></td><td>REAL</td><td colspan="3">Corner rounding with <_NUM> > 2 (radius/chamfer, see <_AMODE>) (inc, to be entered without sign)
("R1" for radius, "FS1" for chamfer)</td></tr>
<tr><td>11</td><td>∅</td><td><_AP1></td><td>REAL</td><td colspan="3">Unmachined diameter of spigot</td></tr>
<tr><td>12</td><td>DXY</td><td><_MIDA></td><td>REAL</td><td colspan="3">Maximum infeed width (for unit, see <_AMODE>)</td></tr>
<tr><td>13</td><td>DZ</td><td><_MID></td><td>REAL</td><td colspan="3">Maximum depth infeed</td></tr>
<tr><td>14</td><td>UXY</td><td><_FAL></td><td>REAL</td><td colspan="3">Finishing allowance, plane</td></tr>
<tr><td>15</td><td>UZ</td><td><_FALD></td><td>REAL</td><td colspan="3">Finishing allowance, depth</td></tr>
<tr><td>16</td><td>F</td><td><_FFP1></td><td>REAL</td><td colspan="3">Machining feedrate</td></tr>
<tr><td rowspan="2">17</td><td rowspan="2"></td><td rowspan="2"><_CDIR></td><td rowspan="2">INT</td><td rowspan="2">Milling direction</td><td>0 =</td><td>Down-cut</td></tr>
<tr><td>1 =</td><td>Up-cut</td></tr>
<tr><td rowspan="9">18</td><td rowspan="9"></td><td rowspan="9"><_VARI></td><td rowspan="9">INT</td><td colspan="3">Machining type</td></tr>
<tr><td rowspan="5">UNITS:</td><td colspan="2">Machining</td></tr>
<tr><td>1 =</td><td>Roughing</td></tr>
<tr><td>2 =</td><td>Finishing</td></tr>
<tr><td>3 =</td><td>Edge finishing</td></tr>
<tr><td>5 =</td><td>Chamfering</td></tr>
<tr><td rowspan="3">TENS:</td><td colspan="2">Width across flats or edge length</td></tr>
<tr><td>0 =</td><td>Width across flats</td></tr>
<tr><td>1 =</td><td>Edge length</td></tr>
<tr><td>19</td><td>FS</td><td><_FS></td><td>REAL</td><td colspan="3">Chamfer width (inc)</td></tr>
<tr><td>20</td><td>ZFS</td><td><_ZFS></td><td>REAL</td><td colspan="3">Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE></td></tr>
<tr><td rowspan="5">21</td><td rowspan="5"></td><td rowspan="5"><_GMODE></td><td rowspan="5">INT</td><td colspan="3">Geometrical mode (evaluation of programmed geometrical data)</td></tr>
<tr><td>UNITS:</td><td colspan="2">Reserved</td></tr>
<tr><td>TENS:</td><td colspan="2">Reserved</td></tr>
<tr><td rowspan="2">HUNDREDS:</td><td colspan="2">Select machining or just calculation of start point</td></tr>
<tr><td>1 =</td><td>Normal machining</td></tr>
</table>

No.	Parameter mask	Parameter internal	Data type	Meaning		
22		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/18/19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	---	Reserved
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple
23		<_AMODE>	INT	Alternative mode		
				UNITS:	Final depth (<_DP>)	
					0 =	Absolute
					1 =	Incremental
				TENS:	Units for plane infeed (<_MIDA>)	
					0 =	mm
					1 =	% of tool diameter
				HUNDREDS:	Insertion depth for chamfering (<_ZFS>)	
					0 =	Absolute
					1 =	Incremental
				THOUSANDS:	Corner rounding (<_RC>)	
					0 =	Radius
					1 =	Chamfer

4.24.1.21 CYCLE81 - drilling, centering

Syntax

```
CYCLE81(<RTP>, <RFP>, <SDIS>, <DP>, <DPR>, <DTB>, <_GMODE>,
<_DMODE>, <_AMODE>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<RTP>	REAL	Retraction plane (abs)
2	Z0	<RFP>	REAL	Reference point (abs)
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)

No.	Parameter mask	Parameter internal	Data type	Meaning		
4	Z1/∅	<DP>	REAL	Drilling depth (abs) / centering diameter (abs), see <_GMODE>		
5	Z1	<DPR>	REAL	Drilling depth (inc)		
6	DT	<DTB>	REAL	Dwell time at final drilling depth, see <_AMODE>		
7		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Centering with respect to depth/diameter	
					0 =	Compatibility, depth
					1 =	Diameter
8		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
9		<_AMODE>	INT	Alternative mode		
				UNITS:	Drilling depth Z1 (abs/inc)	
					0 =	Compatibility, from DP/DPR programming
					1 =	Incremental
					2 =	Absolute
				TENS:	Dwell time at final drilling depth DT in seconds/revolutions	
					0 =	Compatibility, from DTB sign (> 0 seconds or < 0 revolutions)
					1 =	In seconds
					2 =	In revolutions

4.24.1.22 CYCLE82 - drilling, counterboring

Syntax

```
CYCLE82(<RTP>, <RFP>, <SDIS>, <DP>, <DPR>, <DTB>, <_GMODE>,
<_DMODE>, <_AMODE>, <_VARI>, <S_ZA>, <S_FA>, <S_ZD>, <S_FD>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<RTP>	REAL	Retraction plane (abs)
2	Z0	<RFP>	REAL	Reference point (abs)
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)

No.	Parameter mask	Parameter internal	Data type	Meaning		
4	Z1	<DP>	REAL	Drilling depth (abs), see <_AMODE>		
5	Z1	<DPR>	REAL	Drilling depth (inc), see <_AMODE>		
6	DT	<DTB>	REAL	Dwell time at final drilling depth, see <_AMODE>		
7		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Drilling depth with respect to tip/shank	
					0 =	Compatibility, tip
					1 =	Shank
8		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Basic

No.	Parameter mask	Parameter internal	Data type	Meaning		
9		<_AMODE>	INT	Alternative mode		
				UNITS:	Drilling depth Z1 (abs/inc)	
					0 =	Compatibility, from DP/DPR programming
					1 =	Incremental
					2 =	Absolute
				TENS:	Dwell time DT at final drilling depth in seconds/revolutions	
					0 =	Compatibility, from DT sign (> 0 seconds / < 0 revolutions)
					1 =	In seconds
					2 =	In revolutions
				HUNDREDS:	Drilling depth ZA abs/inc	
					0 =	Incremental
					1 =	Absolute
				THOUSANDS:	Evaluation of predrilling feedrate	
					0 =	As % of drilling feedrate
					1 =	F/min
					2 =	F/rev
				TEN THOUSANDS:	Remaining drilling depth ZD abs/inc	
					0 =	Incremental
					1 =	Absolute
				HUNDRED THOUSANDS:	Evaluation of remaining drilling feedrate	
					0 =	As % of drilling feedrate
					1 =	F/min
					2 =	F/rev
10		<_VARI>	INT	Predrilling/through-drilling machining type		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Through drilling	
					0 =	Through drilling "No"
					1 =	Through drilling "Yes"
				TEN THOUSANDS:	Predrilling	
					0 =	Predrilling "No"
					1 =	Predrilling "Yes"
11	ZA	<S_ZA>	REAL	Incremental predrilling depth in relation to reference point or absolute (see <_AMODE> HUNDREDS)		
12	FA	<S_FA>	REAL	Predrilling feedrate as value or in % (in conjunction with <_AMODE> THOUSANDS)		

No.	Parameter mask	Parameter internal	Data type	Meaning
13	ZD	<S_ZD>	REAL	Incremental remaining drilling depth in relation to final drilling depth or absolute (see <_AMODE> TEN THOUSANDS)
14	FD	<S_FD>	REAL	Remaining drilling feedrate as value or in % (in conjunction with <_AMODE> HUNDRED THOUSANDS)

4.24.1.23 CYCLE83 – deep-hole drilling 1

Syntax

```
CYCLE83(<RTP>, <RFP>, <SDIS>, <DP>, <DPR>, <FDEP>, <FDPR>, <_DAM>,
<DTB>, <DTS>, <FRF>, <VARI>, <_AXN>, <_MDEP>, <_VRT>, <_DTD>,
<_DIS1>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<RTP>	REAL	Retraction plane (abs)		
2	Z0	<RFP>	REAL	Reference point (abs)		
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<DP>	REAL	Final drilling depth (abs), see <_AMODE>		
5	Z1	<DPR>	REAL	Final drilling depth (inc), see <_AMODE>		
6	D	<FDEP>	REAL	1st drilling depth (abs), see <_AMODE>		
7	D	<FDPR>	REAL	1st drilling depth (inc), see <_AMODE>		
8	DF	<_DAM>	REAL	Degression value / percentage for each additional infeed, see <_AMODE>		
9	DTB	<DTB>	REAL	Dwell time at drilling depth, see <_AMODE>		
10	DTS	<DTS>	REAL	Dwell time at start point (for chip removal only), see <_AMODE>		
11	FD1	<FRF>	REAL	Percentage for the feedrate for the first infeed, see <_AMODE>		
12		<VARI>	INT	Machining type		
				UNITS:	Chip breaking/removal	
					0 =	Chip breaking
					1 =	Swarf removal
13		<_AXN>	INT	Tool axis		
					0 =	3rd geometry axis
					1 =	1st geometry axis
					2 =	2nd geometry axis
					> 2	3rd geometry axis
14	V1	<_MDEP>	REAL	Minimum infeed (only for degression percentage)		
15	V2	<_VRT>	REAL	Retraction distance after each machining step (for chip breaking only)		
					> 0	Variable retraction distance
					0 =	Default value 1 mm
16	DT	<_DTD>	REAL	Dwell time at final drilling depth, see <_AMODE>		

No.	Parameter mask	Parameter internal	Data type	Meaning		
17	V3	<_DIS1>	REAL	Limit distance (for chip removal only), see <_AMODE>		
18		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Drilling depth with respect to tip/shank	
					0 =	Tip
					1 =	Shank
19		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	---	Reserved
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple

No.	Parameter mask	Parameter internal	Data type	Meaning		
20		<_AMODE>	INT	Alternative mode		
				UNITS:	Drilling depth = Final drilling depth Z1 (abs/inc)	
					0 =	Compatibility, from programming <DP>/<DPR>
					1 =	Incremental
					2 =	Absolute
				TENS:	Dwell time at drilling depth DTB in seconds/revolutions	
					0 =	Compatibility, from DTB sign (> 0 seconds or < 0 revolutions)
					1 =	In seconds
					2 =	In revolutions
				HUNDREDS:	Dwell time at start point of DTS in seconds/revolutions	
					0 =	Compatibility, from DTS sign (> 0 seconds or < 0 revolutions)
					1 =	In seconds
					2 =	In revolutions
				THOUSANDS:	Dwell time at final drilling depth DTD in seconds/revolutions	
					0 =	Compatibility, from DTD sign (> 0 seconds or < 0 revolutions)
					1 =	In seconds
					2 =	In revolutions
				TEN THOUSANDS:	1st drilling depth D (abs/inc)	
					0 =	Compatibility, from programming <FDEPF>/<DPR>
					1 =	Incremental
					2 =	Absolute
				HUNDRED THOUSANDS:	Degression value / percentage <_DAM> for each additional infeed	
					0 =	Compatibility, from <_DAM> sign (> 0 degression value or < 0 factor 0.001 to 1.0)
					1 =	Degression value
					2 =	Percentage (0.001 to 100%)
				ONE MILLION:	Limit distance V3 automatic/manual	
					0 =	Compatibility from <_DIS1> sign (= 0 automatic or > 0 manual)
					1 =	Automatic (calculated in the cycle)
					2 =	Manual (programmed value)
				TEN MILLIONS:	Feedrate factor for first infeed <FRF> as factor/percentage	

No.	Parameter mask	Parameter internal	Data type	Meaning		
					0 =	Compatibility, as a factor (0.001 to 1.0, FRF = 0 means 100%)
					1 =	Percentage (0.001 to 999.999%)

4.24.1.24 CYCLE84 - tapping without compensating chuck

Syntax

```
CYCLE84(<RTP>, <RFP>, <SDIS>, <DP>, <DPR>, <DTB>, <SDAC>, <MPIT>,
<PIT>, <POSS>, <SST>, <SST1>, <_AXN>, <_PITA>, <_TECHNO>, <_VARI>,
<_DAM>, <_VRT>, <_PITM>, <_PTAB>, <_PTABA>, <_GMODE>, <_DMODE>,
<_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning			
1	RP	<RTP>	REAL	Retraction plane (abs)			
2	Z0	<RFP>	REAL	Reference point (abs)			
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)			
4	Z1	<DP>	REAL	Drilling depth = final drilling depth (abs), see <_AMODE>			
5	Z1	<DPR>	REAL	Drilling depth = final drilling depth (inc), see <_AMODE>			
6	DT	<DTB>	REAL	Dwell time at drilling depth in seconds			
7	SDE	<SDAC>	INT	Direction of rotation after end of cycle			
8		<MPIT>	REAL	Thread size for "ISO metric" only (pitch is calculated internally during run time)			
9	P	<PIT>	REAL	Pitch as a value, for unit see <_PITA>			
10	αS[1)]	<POSS>	REAL	Spindle position for oriented spindle stop			
11	S	<SST>	REAL	Spindle speed for tapping			
12	SR	<SST1>	REAL	Spindle speed for retraction			
13		<_AXN>	INT	Drilling axis	0 =	3rd geometry axis	
					1 =	1st geometry axis	
					2 =	2nd geometry axis	
					≥ 3 =	3rd geometry axis	
14		<_PITA>	INT	Pitch unit (evaluation of <PIT> and <MPIT>)			
					0 =	Pitch in mm	- evaluation<MPIT>/<PIT>
					1 =	Pitch in mm	- evaluation<PIT>
					2 =	Pitch in TPI	- evaluation of <PIT> (threads per inch)
					3 =	Pitch in inches	- evaluation<PIT>
					4 =	MODULUS	- evaluation<PIT>

No.	Parameter mask	Parameter internal	Data type	Meaning		
15		<_TECHNO>	INT	Technology[1]		
				UNITS:	Exact stop response	
					0 =	Exact stop response active as before cycle call
					1 =	Exact stop G601
					2 =	Exact stop G602
					3 =	Exact stop G603
				TENS:	Feedforward control	
					0 =	With/without feedforward control active as before cycle call
					1 =	With feedforward control FFWON
					2 =	Without feedforward control FFWOF
				HUNDREDS:	Acceleration	
					0 =	SOFT/BRISK/DRIVE active as before cycle call
					1 =	With jerk limitation SOFT
					2 =	Without jerk limitation BRISK
					3 =	Reduced acceleration DRIVE
				THOUSANDS:	MCALL spindle mode	
					0 =	Reactivate spindle operation for MCALL
					1 =	For MCALL remain in position control
16		<_VARI>	INT	Machining type		
				UNITS:	0 =	1 cut
					1 =	Chip breaking (deep hole tapping)
					2 =	Chip removal (deep hole tapping)
				THOUSANDS:	ISO/SIEMENS mode not relevant for screen form	
					0 =	Call from ISO compatibility
					1 =	Call from SIEMENS context
17	D	<_DAM>	REAL	Maximum depth infeed (for chip removal/breaking only)		
18	V2	<_VRT>	REAL	Retraction distance after each machining step (for chip breaking only), see <_AMODE>		
19		<_PITM>	STRING[15]	String as marker for pitch input[2]		
20		<_PTAB>	STRING[5]	String for thread table ("", "ISO", "BSW", "BSP", "UNC")[2]		
21		<_PTABA>	STRING[20]	String for selection from thread table (e.g. "M 10", "M 12", ...)[2]		
22		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	

No.	Parameter mask	Parameter internal	Data type	Meaning		
23		`<_DMODE>`	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Compatibility mode (for recompilation screen form only), if MD 52216 bit0 = 1[1)]	
					0 =	Technology parameters are displayed (compatibility): TECHNO parameters effective
					1 =	Technology parameters are not displayed: Technology active "as before cycle call"
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple

No.	Parameter mask	Parameter internal	Data type	Meaning		
24		<_AMODE>	INT	Alternative mode		
				UNITS:	Drilling depth = Final drilling depth Z1 (abs/inc)	
					0 =	Compatibility, from programming <DP>/<DPR>
					1 =	Incremental
					2 =	Absolute
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Thread direction of rotation right/left	
					0 =	Compatibility, from PIT/MPTI sign
					1 =	Right
					2 =	Left
				TEN THOUSANDS:	Reserved	
				HUNDRED THOUSANDS:	Reserved	
				ONE MILLION:	Retraction distance after each machining step V2 manual/automatic	
					0 =	Compatibility, from <_VRT> programming (> 0 variable value or ≤ 0 standard value 1 mm / 0.0394 inch)
					1 =	Automatic (standard value 1 mm / 0.0394 inch)
					2 =	Manual (programmed as under V2)

[1)] Technology fields may be hidden, depending on the setting date SD52216 $MCS_FUNCTION_MASK_DRILL

[2)] Parameters 19, 20 and 21 are only used for thread selection in the screen form thread tables. The thread tables cannot be accessed via cycle definition in the cycle run time.

4.24.1.25 CYCLE85 - reaming

Syntax

```
CYCLE85(<RTP>, <RFP>, <SDIS>, <DP>, <DPR>, <DTB>, <FFR>, <RFF>,
<_GMODE>, <_DMODE>, <_AMODE>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<RTP>	REAL	Retraction plane (abs)
2	Z0	<RFP>	REAL	Reference point (abs)
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)
4	Z1	<DP>	REAL	Drilling depth (abs), see <_AMODE>
5	Z1	<DPR>	REAL	Drilling depth (inc), see <_AMODE>
6	DT	<DTB>	REAL	Dwell time at final drilling depth, see <_AMODE>

No.	Parameter mask	Parameter internal	Data type	Meaning		
7	F	<FFR>	REAL	Feedrate		
8	FR	<RFF>	REAL	Feedrate during retraction		
9		<_GMODE>	INT	Reserved		
10		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
11		<_AMODE>	INT	Alternative mode (drilling)		
				UNITS:	Drilling depth Z1 (abs/inc)	
					0 =	Compatibility, from DP/DPR programming
					1 =	Incremental
					2 =	Absolute
				TENS:	Dwell time DT at final drilling depth in seconds/revolutions	
					0 =	Compatibility, from DT sign (> 0 seconds or < 0 revolutions)
					1 =	In seconds
					2 =	In revolutions

4.24.1.26 CYCLE86 - boring

Syntax

```
CYCLE86(<RTP>, <RFP>, <SDIS>, <DP>, <DPR>, <DTB>, <SDIR>, <RPA>,
<RPO>, <RPAP>, <POSS>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<RTP>	REAL	Retraction plane (abs)		
2	Z0	<RFP>	REAL	Reference point (abs)		
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<DP>	REAL	Drilling depth (abs), see <_AMODE>		
5	Z1	<DPR>	REAL	Drilling depth (inc), see <_AMODE>		
6	DT	<DTB>	REAL	Dwell time at final drilling depth, see <_AMODE>		
7	DIR	<SDIR>	INT	Direction of spindle rotation	3 =	M3
					4 =	M4

<table>
<tr><th>No.</th><th>Parameter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td>8</td><td>DX</td><td><RPA></td><td>REAL</td><td colspan="3">Lift-off distance in X direction</td></tr>
<tr><td>9</td><td>DY</td><td><RPO></td><td>REAL</td><td colspan="3">Lift-off distance in the Y direction</td></tr>
<tr><td>10</td><td>DZ</td><td><RPAP></td><td>REAL</td><td colspan="3">Lift-off distance in the Z direction</td></tr>
<tr><td>11</td><td>SPOS</td><td><POSS></td><td>REAL</td><td colspan="3">Spindle position for lift-off (for oriented spindle stop, in degrees)</td></tr>
<tr><td rowspan="4">12</td><td rowspan="4"></td><td rowspan="4"><_GMODE></td><td rowspan="4">INT</td><td colspan="3">Geometrical mode (evaluation of programmed geometrical data)</td></tr>
<tr><td rowspan="3">UNITS:</td><td colspan="2">Lift mode</td></tr>
<tr><td>0 =</td><td>Lift off, compatibility</td></tr>
<tr><td>1 =</td><td>Do not lift off contour</td></tr>
<tr><td rowspan="6">13</td><td rowspan="6"></td><td rowspan="6"><_DMODE></td><td rowspan="6">INT</td><td colspan="3">Display mode</td></tr>
<tr><td rowspan="5">UNITS:</td><td colspan="2">Machining plane G17/G18/G19</td></tr>
<tr><td>0 =</td><td>Compatibility, the plane effective before the cycle call remains active</td></tr>
<tr><td>1 =</td><td>G17 (only active in the cycle)</td></tr>
<tr><td>2 =</td><td>G18 (only active in the cycle)</td></tr>
<tr><td>3 =</td><td>G19 (only active in the cycle)</td></tr>
<tr><td rowspan="9">14</td><td rowspan="9"></td><td rowspan="9"><_AMODE></td><td rowspan="9">INT</td><td colspan="3">Alternative mode</td></tr>
<tr><td rowspan="4">UNITS:</td><td colspan="2">Drilling depth Z1 (abs/inc)</td></tr>
<tr><td>0 =</td><td>Compatibility, from programming<DP>/<DPR></td></tr>
<tr><td>1 =</td><td>Incremental</td></tr>
<tr><td>2 =</td><td>Absolute</td></tr>
<tr><td rowspan="4">TENS:</td><td colspan="2">Dwell time at final drilling depth DT in seconds/revolutions</td></tr>
<tr><td>0 =</td><td>Compatibility, from DT sign (> 0 seconds or < 0 revolutions)</td></tr>
<tr><td>1 =</td><td>In seconds</td></tr>
<tr><td>2 =</td><td>In revolutions</td></tr>
</table>

4.24.1.27 CYCLE92 - cut-off

Syntax

```
CYCLE92(<_SPD>, <_SPL>, <_DIAG1>, <_DIAG2>, <_RC>, <_SDIS>, <_SV1>,
<_SV2>, <_SDAC>, <_FF1>, <_FF2>, <_SS2>, <_DIAGM>, <_VARI>, <_DN>,
<_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	X0	<_SPD>	REAL	Reference point (abs, always diameter)
2	Y0	<_SPL>	REAL	Reference point (abs)

No.	Parameter mask	Parameter internal	Data type	Meaning		
3	X1	<_DIAG1>	REAL	Depth for speed reduction, see <_AMODE> (UNITS)		
4	X2	<_DIAG2>	REAL	Final depth, see <_AMODE> (TENS)		
5	R/FS	<_RC>	REAL	Rounding status or chamfer width, see <_AMODE> (THOUSANDS)		
6	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
7	S	<_SV1>	REAL	Constant spindle speed, see <_AMODE> (TEN THOUSANDS)		
	V			Constant cutting rate		
8	SV	<_SV2>	REAL	Maximum speed at constant cutting speed		
9	DIR	<_SDAC>	INT	Direction of spindle rotation	3 =	For M3
					4 =	For M4
10	F	<_FF1>	REAL	Infeed as far as depth for speed reduction		
11	FR	<_FF2>	REAL	Reduced infeed as far as final depth		
12	SR	<_SS2>	REAL	Reduced speed as far as final depth		
13	XM	<_DIAGM>	REAL	Depth to withdraw parts gripper (abs, always diameter)		
14		<_VARI>	INT	Machining type		
				UNITS:	Retraction	
					0 =	Retraction to <_SPD> + <_SDIS>
					1 =	No retraction at the end
				TENS:	Parts gripper	
					0 =	No, do not execute M command
					1 =	Yes, call from CUST_TECH-CYC(101)- open drawer, CUST_TECHCYC(102)- close drawer
15		<_DN>	INT	D number for 2nd edge of tool; if not programmed ⇒ D+1		
20		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)

No.	Parameter mask	Parameter internal	Data type	Meaning			
21		<_AMODE>	INT	Alternative mode			
				UNITS:	Depth for speed reduction (<_DIAG1>)		
					0 =	Absolute, value of transverse axis in the diameter	
					1 =	Incremental, value of transverse axis in the radius	
				TENS:	Final depth (<_DIAG2>)		
					0 =	Absolute, value of transverse axis in the diameter	
					1 =	Incremental, value of transverse axis in the radius	
				HUNDREDS:	Reserved		
				THOUSANDS:	Radius/chamfer (<_RC>)		
					0 =	Radius	
					1 =	Chamfer	
				TEN THOUSANDS:	Spindle speed / cutting rate (<_SV1>)		
					0 =	Constant spindle speed	
					1 =	Constant cutting rate	

4.24.1.28 CYCLE95 - contour cutting

Syntax

```
CYCLE95(<NPP>, <MID>, <FALZ>, <FALX>, <FAL>, <FF1>, <FF2>, <FF3>,
<_VARI>, <DT>, <DAM>, <_VRT>, <_GMODE>, <_DMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	CON	<NPP>	STRING [140]	Contour name
2	D	<MID>	REAL	Maximum depth infeed during roughing, see <_GMODE>
3	UZ	<FALZ>	REAL	Finishing allowance in Z
4	UX	<FALX>	REAL	Finishing allowance in X
5	U	<FAL>	REAL	Finishing allowance parallel to contour (effective in both axes)
6	F	<FF1>	REAL	Feedrate for roughing
7	FY	<FF2>	REAL	Insertion feedrate, relief cuts
8	FS	<FF3>	REAL	Finishing feedrate

No.	Parameter mask	Parameter internal	Data type	Meaning		
9		`<_VARI>`	INT	Machining type		
				UNITS and TENS:		
					1 =	Roughing, longitudinal, external
					2 =	Roughing, transverse, external
					3 =	Roughing, longitudinal, internal
					4 =	Roughing, transverse, internal
					5 =	Finishing, longitudinal, external
					6 =	Finishing, transverse, external
					7 =	Finishing, longitudinal, internal
					8 =	Finishing, transverse, internal
					9 =	Complete machining, longitudinal, external
					10 =	Complete machining, transverse, external
					11 =	Complete machining, longitudinal, internal
					12 =	Complete machining, transverse, internal
				HUNDREDS:		
					0 =	With rounding at the contour, without residual corners
					1 =	Without rounding at the contour
					2 =	Rounding only to previous intersection, residual corners can result
10	DT	`<DT>`	REAL	Dwell time at feed interruption		
11	DI	`<DAM>`	REAL	Distance for feed interruptions		
12	VRT	`<_VRT>`	REAL	Lift-off distance from the contour		
					0 =	A lift-off distance of 1 mm is used internally regardless of the active system (inch or metric)
					> 0 =	Lift-off distance
13		`<_GMODE>`	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Evaluation of the infeed depth	
					0 =	Infeed depth is calculated corresponding to the G group DIAMON/DIAMOF
					1 =	Infeed depth acts as radius value (independent of DIAMON/ DIAMOF)

<table>
<tr><th>No.</th><th>Parameter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td rowspan="10">14</td><td rowspan="10"></td><td rowspan="10"><_DMODE></td><td rowspan="10">INT</td><td colspan="3">Display mode</td></tr>
<tr><td rowspan="5">UNITS:</td><td colspan="2">Machining plane G17/G18/G19</td></tr>
<tr><td>0 =</td><td>Compatibility, the plane effective before the cycle call remains active</td></tr>
<tr><td>1 =</td><td>G17 (only active in the cycle)</td></tr>
<tr><td>2 =</td><td>G18 (only active in the cycle)</td></tr>
<tr><td>3 =</td><td>G19 (only active in the cycle)</td></tr>
<tr><td rowspan="3">THOUSANDS:</td><td colspan="2"></td></tr>
<tr><td>0 =</td><td>Compatibility mode: Contour name is in NPP</td></tr>
<tr><td>1 =</td><td>Contour name is programmed in CYCLE62 and transferred to _SC_CONT_NAME</td></tr>
</table>

4.24.1.29 CYCLE98 - thread chain

Syntax

```
CYCLE98(<_PO1>, <_DM1>, <_PO2>, <_DM2>, <_PO3>, <_DM3>, <_PO4>,
<_DM4>, <APP>, <ROP>, <TDEP>, <FAL>, <_IANG>, <NSP>, <NRC>, <NID>,
<_PP1>, <_PP2>, <_PP3>, <_VARI>, <_NUMTH>, <_VRT>, <_MID>, <_GDEP>,
<_IFLANK>, <_PITA>, <_PITM1>, <_PITM2>, <_PITM3>, <_DMODE>,
<_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	Z0	<_PO1>	REAL	Reference point in Z (abs)
2	X0	<_DM1>	REAL	Reference point in X (abs), in diameter
3	Z1	<_PO2>	REAL	Intermediate point 1 in Z (abs/inc), see <_AMODE> (UNITS)
4	X1	<_DM2>	REAL	Intermediate point 1 in X (abs/inc), see <_AMODE> (TENS) or
	X1α			Thread inclination 1 (-90° to 90°) abs is always diameter, inc is always radius
5	Z2	<_PO3>	REAL	Intermediate point 2 in Z, (abs/inc), see <_AMODE> (HUNDREDS)
6	X2	<_DM3>	REAL	Intermediate point 2 in X (abs/inc), see <_AMODE> (THOUSANDS) or
	X2α			Thread inclination 2 (-90° to 90°) abs is always diameter, inc is always radius
7	Z3	<_PO4>	REAL	End point in Z, (abs/inc), see <_AMODE> (TEN THOUSANDS)
8	X3	<_DM4>	REAL	End point in X, (abs/inc), see <_AMODE> (HUNDRED THOUSANDS) or
	X3α			Thread inclination 3 (-90° to 90°) abs is always diameter, inc is always radius
9	LW	<APP>	REAL	Thread run-in (inc, to be entered without sign)

No.	Parameter mask	Parameter internal	Data type	Meaning
10	LR	<ROP>	REAL	Thread run-out (inc, to be entered without sign)
11	H1	<TDEP>	REAL	Thread depth (inc, to be entered without sign)
12	U	<FAL>	REAL	Finishing allowance in X and Z
13	DP	<_IANG>	REAL	Infeed slope as a distance or an angle, see <_AMODE> (ONE MILLION)
	αP			The infeed slope is applied according to the setting of parameter <_VARI> (HUNDREDS). Definition for <_VARI_HUNDREDS = 0 - Compatibility mode: > 0 = Side infeed on one side 0 = Infeed vertical in the thread < 0 = Side infeed with alternating sides Definition for _VARI_HUNDREDS<>0: > 0 = Infeed on the positive side 0 = Center infeed < 0 = Infeed on the negative side
14	α0	<NSP>	REAL	Starting angle offset for the 1st thread
15		<NRC>	INT	Number of roughing cuts, see <_VARI> (TEN THOUSANDS)
16	NN	<NID>	INT	Number of non-cuts
17	P0	<_PP1>	REAL	Pitch for 1st section of thread, see <_PITA>
18	P1	<_PP2>	REAL	Pitch for 2nd section of thread, see <_PITA>
19	P2	<_PP3>	REAL	Pitch for 3rd section of thread, see <_PITA>

No.	Parameter mask	Parameter internal	Data type	Meaning		
20		`<_VARI>`	INT	Machining		
				UNITS:	Technology	
					1 =	External thread with linear infeed
					2 =	Internal thread with linear infeed
					3 =	External thread with degressive infeed, cross-section of cut remains constant
					4 =	Internal thread with degressive infeed, cross-section of cut remains constant
				TENS:	Reserved	
				HUNDREDS:	Infeed type	
					0 =	Compatibility mode for `<_IANG>`
					1 =	Infeed on one side
					2 =	Infeed alternate sides
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Alternative depth infeed	
					0 =	Compatibility, preset number of roughing cuts (`<_NRC>`)
					1 =	Preset value for 1st infeed (`<_MID>`)
				HUNDRED THOUSANDS:	Machining type	
					0 =	Compatibility (roughing and finishing)
					1 =	Roughing
					2 =	Finishing
					3 =	Roughing and finishing
				ONE MILLION:	Machining sequence for multistart thread	
					0 =	In ascending order of threads
					1 =	In descending order of threads
21	N	`<_NUMTH>`	INT	Number of thread turns		
22		`<_VRT>`	REAL	Return distance (inc)		
					0 =	A lift-off distance of 1 mm is used internally regardless of the active system (inch or metric)
					> 0 =	Lift-off distance
23	D1	`<_MID>`	REAL	First infeed, see `<_VARI>` (TEN THOUSANDS)		
24	DA	`<_GDEP>`	REAL	Thread changeover depth (only effective with "multiple start")		
					0 =	Do not observe any thread changeover depth
					> 0 =	Observe thread changeover depth
25		`<_IFLANK>`	REAL	Infeed slope as width (for interface only)		

No.	Parameter mask	Parameter internal	Data type	Meaning		
26		<_PITA>	INT	Evaluation of thread pitch		
					0 =	Compatibility mode for pitch, evaluation <_PP1> to <_PP3> as previously, according to active system (metric/inch)
					1 =	Pitch in mm
					2 =	Pitch in TPI (threads per inch)
					3 =	Pitch in inches
					4 =	MODULUS
27		<_PITM1>	STRING[15]	String as marker for pitch input (for the interface only)		
28		<_PITM2>	STRING[15]	String as marker for pitch input (for the interface only)		
29		<_PITM3>	STRING[15]	String as marker for pitch input (for the interface only)		
30		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	---	Reserved
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple

No.	Parameter mask	Parameter internal	Data type	Meaning		
31		<_AMODE>	INT	Alternative mode		
				UNITS:	1st intermediate point in Z (Z1)	
					0 =	Absolute
					1 =	Incremental
				TENS:	1st intermediate point in X (X1)	
					0 =	Absolute
					1 =	Incremental
					2 =	α
				HUNDREDS:	2nd intermediate point in Z (Z2)	
					0 =	Absolute
					1 =	Incremental
				THOUSANDS:	2nd intermediate point in X (X2)	
					0 =	Absolute
					1 =	Incremental
					2 =	α
				TEN THOUSANDS:	End point in Z (Z3)	
					0 =	Absolute
					1 =	Incremental
				HUNDRED THOUSANDS:	End point in X (X3)	
					0 =	Absolute
					1 =	Incremental
					2 =	α
				ONE MILLION:	Select infeed slope as angle or width	
					0 =	Infeed angle <_IANG>
					1 =	Infeed slope <_IFLANK>
				TEN MILLIONS:	Single/multiple thread	
					0 =	Compatibility mode (starting angle <_NSP> is evaluated)
					1 =	Single thread (with starting angle offset <_NSP>)
					2 =	Multiple

4.24.1.30 CYCLE99 - thread turning

Syntax

```
CYCLE99(<_SPL>, <_SPD>, <_FPL>, <_FPD>, <_APP>, <_ROP>, <_TDEP>,
<_FAL>, <_IANG>, <_NSP>, <_NRC>, <_NID>, <_PIT>, <_VARI>, <_NUMTH>,
<_SDIS>, <_MID>, <_GDEP>, <_PIT1>, <_FDEP>, <_GST>, <_GUD>,
<_IFLANK>, <_PITA>, <_PITM>, <_PTAB>, <_PTABA>, <_DMODE>, <_AMODE>,
<_S_XRS>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	Z0	<_SPL>	REAL	Reference point (abs)
2	X0	<_SPD>	REAL	Reference point (abs, always diameter)
3	Z1	<_FPL>	REAL	End point in conjunction with <_AMODE> (UNITS)
4	X1	<_FPD>	REAL	End point in conjunction with <_AMODE> (TENS)
5	LW/LW2	<_APP>	REAL	Thread run-in in conjunction with <_AMODE> (HUNDREDS) or Thread run-in = thread run-out in conjunction with <_AMODE> (HUNDREDS)
6	LR	<_ROP>	REAL	Thread run-out
7	H1	<_TDEP>	REAL	Thread depth
8	U	<_FAL>	REAL	Finishing allowance in X and Z
9	DP αP	<_IANG>	REAL	Infeed slope as a distance or an angle, in conjunction with <_AMODE> (THOUSANDS) > 0 = Infeed on the positive side < 0 = Infeed on the negative side 0 = Center infeed
10	α0	<_NSP>	REAL	Starting angle offset (only effective with "single start")
11	ND	<_NRC>	INT	Number of roughing cuts, in combination with <_VARI> (TEN THOUSANDS)
12	NN	<_NID>	INT	Number of non-cuts
13	P	<_PIT>	REAL	Pitch as a value, in conjunction with <_PITA>

<table>
<tr><th>No.</th><th>Parameter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td rowspan="21">14</td><td rowspan="21"></td><td rowspan="21"><_VARI></td><td rowspan="21">INT</td><td colspan="3">Machining type</td></tr>
<tr><td rowspan="5">UNITS:</td><td colspan="2">Technology</td></tr>
<tr><td>1 =</td><td>External thread with linear infeed</td></tr>
<tr><td>2 =</td><td>Internal thread with linear infeed</td></tr>
<tr><td>3 =</td><td>External thread with degressive infeed, cross-section of cut remains constant</td></tr>
<tr><td>4 =</td><td>Internal thread with degressive infeed, cross-section of cut remains constant</td></tr>
<tr><td>TENS:</td><td colspan="2">Reserved</td></tr>
<tr><td rowspan="3">HUNDREDS:</td><td colspan="2">Infeed type</td></tr>
<tr><td>1 =</td><td>Infeed on one side</td></tr>
<tr><td>2 =</td><td>Infeed alternate sides</td></tr>
<tr><td>THOUSANDS:</td><td colspan="2">Reserved</td></tr>
<tr><td rowspan="3">TEN THOUSANDS:</td><td colspan="2">Alternative depth infeed</td></tr>
<tr><td>0 =</td><td>Preset number of roughing cuts (<_NRC>)</td></tr>
<tr><td>1 =</td><td>Preset value for 1st infeed (<_MID>)</td></tr>
<tr><td rowspan="4">HUNDRED THOUSANDS:</td><td colspan="2">Machining type</td></tr>
<tr><td>1 =</td><td>Roughing</td></tr>
<tr><td>2 =</td><td>Finishing</td></tr>
<tr><td>3 =</td><td>Roughing and finishing</td></tr>
<tr><td rowspan="3">ONE MILLION:</td><td colspan="2">Machining sequence for multistart thread</td></tr>
<tr><td>0 =</td><td>In ascending order of threads</td></tr>
<tr><td>1 =</td><td>In descending order of threads</td></tr>
<tr><td>15</td><td>N</td><td><_NUMTH></td><td>INT</td><td colspan="3">Number of thread turns</td></tr>
<tr><td>16</td><td>VR</td><td><_SDIS></td><td>REAL</td><td colspan="3">Return distance, inc</td></tr>
<tr><td>17</td><td>D1</td><td><_MID></td><td>REAL</td><td colspan="3">First infeed depth, in conjunction with <_VARI> (TEN THOUSANDS)</td></tr>
<tr><td rowspan="3">18</td><td rowspan="3">DA</td><td rowspan="3"><_GDEP></td><td rowspan="3">REAL</td><td colspan="3">Thread changeover depth (only effective with "multiple start")</td></tr>
<tr><td rowspan="2"></td><td>0 =</td><td>Do not observe any thread changeover depth</td></tr>
<tr><td>> 0 =</td><td>Observe thread changeover depth</td></tr>
<tr><td rowspan="4">19</td><td rowspan="4">G</td><td rowspan="4"><_PIT1></td><td rowspan="4">REAL</td><td colspan="3">Change of pitch per revolution</td></tr>
<tr><td rowspan="3"></td><td>0 =</td><td>Pitch is constant (G33)</td></tr>
<tr><td>> 0 =</td><td>Pitch increases (G34)</td></tr>
<tr><td>< 0 =</td><td>Pitch decreases (G35)</td></tr>
<tr><td>20</td><td></td><td><_FDEP></td><td>REAL</td><td colspan="3">Insertion depth (enter without sign)</td></tr>
<tr><td>21</td><td>N1</td><td><_GST></td><td>INT</td><td colspan="3">Starting thread N1 = 1...N, in conjunction with <_AMODE> (HUNDRED THOUSANDS)</td></tr>
<tr><td>22</td><td></td><td><_GUD></td><td>INT</td><td colspan="3">Reserved</td></tr>
<tr><td>23</td><td></td><td><_IFLANK></td><td>REAL</td><td colspan="3">Infeed slope as width (for interface only)</td></tr>
</table>

No.	Parameter mask	Parameter internal	Data type	Meaning		
24		`<_PITA>`	INT	Pitch unit (evaluation of PIT and/or MPIT)		
					0 =	Pitch in mm - MPIT/PIT evaluation
					1 =	Pitch in mm - PIT evaluation
					2 =	Pitch in TPI - PIT evaluation (threads per inch)
					3 =	Pitch in inches - PIT evaluation
					4 =	MODULE - PIT evaluation
25		`<_PITM>`	STRING[15]	String as marker for pitch input (for the interface only)[1)]		
26		`<_PTAB>`	STRING[20]	String for thread table (for the interface only)[1)]		
27		`<_PTABA>`	STRING[20]	String for selection in the thread table (for the interface only)[1)]		
28		`<_DMODE>`	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Type of thread	
					0 =	Longitudinal thread
					1 =	Face thread
					2 =	Tapered thread
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Basic

No.	Parameter mask	Parameter internal	Data type	Meaning		
29		<_AMODE>	INT	Alternative mode		
				UNITS:	Thread length in Z	
					0 =	Absolute
					1 =	Incremental
				TENS:	Thread length in X	
					0 =	Absolute, value of transverse axis in the diameter
					1 =	Incremental, value of transverse axis in the radius
					2 =	α
				HUNDREDS:	Calculation of approach/run-in path <_APP>	
					0 =	Thread run-in <_APP>
					1 =	Thread run-in = thread run-out <_APP> = -<_ROP>
					2 =	Specify thread run-in path <_APP> = -<_APP>
				THOUSANDS:	Select infeed slope as angle or width	
					0 =	Infeed angle <_IANG>
					1 =	Infeed slope <_IFLANK>
				TEN THOUSANDS:	Single/multiple thread	
					0 =	Single thread (with starting angle offset <_NSP>)
					1 =	Multiple
				HUNDRED THOUSANDS:	Starting thread <_GST>	
					0 =	Full machining
					1 =	Start machining from this thread
					2 =	Only machine this thread
				ONE MILLION:	Sag compensation for longitudinal thread	
					0 =	Segment height, crowned thread XS
					1 =	Radius, crowned thread RS
30	XS/RS	<_S_XRS>	REAL	Sag compensation for longitudinal thread in conjunction with <_AMODE>: ONE MILLION		

Note

[1)] Parameters <_PITM>, <_PTAB> and <_PTABA> are only used for thread selection in the screen form thread tables.
The thread tables cannot be accessed via cycle definition in the cycle run time.

4.24.1.31 CYCLE435 - Set dresser coordinate system

Syntax

```
CYCLE435(<_T>, <_DD>, <S_TA>, <S_DA>, <S_AD>, <S_AL>, <S_PVD>,
<S_PVL>, <S_PD>, <S_PL>, <_AMODE>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning
1		<_T>	STRING[32]	Tool name of the grinding wheel
2		<_DD>	INT	Cutting edge number of the grinding wheel
3		<S_TA>	STRING[32]	Dressing tool reference point - dressing tool name
4		<S_DA>	INT	Cutting edge number of the dressing tool
5		<S_AD>	REAL	Dressing value, diameter
6		<S_AL>	REAL	Dressing value, face
7		<S_PVD>	REAL	Form-truing offset, diameter
8		<S_PVL>	REAL	Form-truing offset, face
9		<S_PD>	REAL	Form-truing allowance, diameter
10		<S_PL>	REAL	Form-truing allowance, face
11		<_AMODE>	INT	Alternative mode
				UNITS: active tool at the end of the cycle
				0 = dressing tool active
				1 = wheel active

4.24.1.32 CYCLE495 - form-truing

Syntax

```
CYCLE495(<_T>, <_DD>, <_SC>, <_F>, <_VARI>, <_D>, <_DX>, <_DZ>,
<S_PA>, <S_N>, <_DMODE>, <_AMODE>, <S_FW>, <S_HW>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1		<_T>	STRING[20]	Tool name of the grinding wheel
2		<_DD>	INT	Cutting edge number of the grinding wheel
3		<_SC>	REAL	Lift-off distance for avoiding obstacles, incremental
4		<_F>	REAL	Form-truing feedrate

No.	Parameter mask	Parameter internal	Data type	Meaning		
5		<_VARI>	INT	Machining type		
				UNITS:	Form-truing type	
					1 =	Parallel to the axis
					2 =	Parallel to the contour
				TENS:	Machining direction	
					0 =	Pulling Possible with cutting edge positions 1 to 4
					1 =	Pushing Possible with cutting edge positions 1 to 4
					2 =	Alternating Possible with cutting edge positions 1 to 8
					3 =	Start → end Possible with cutting edge positions 1 to 8
					4 =	End → start Possible with cutting edge positions 1 to 8
				HUNDREDS:	Infeed direction	
					1 =	Infeed X for G18 or Y- for G19
					2 =	Infeed X+ for G18 or Y+ for G19
					3 =	Infeed Z- for G18 and for G19
					4 =	Infeed Z+ for G18 and for G19
6		<_D>	REAL	Dressing value for form-truing type parallel to the axis		
7		<_DX>	REAL	Dressing value X for G18 or Y for G19 for form-truing type parallel to the contour		
8		<_DZ>	REAL	Dressing value Z for G18 and G19 for form-truing type parallel to the contour		
9		<S_PA>	REAL	Form-truing allowance		
10		<S_N>	INT	Number of strokes in the form-truing program		
11		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)

<table>
<tr><th>No.</th><th>Parameter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td rowspan="7">12</td><td rowspan="7"></td><td rowspan="7"><_AMODE></td><td rowspan="7">INT</td><td colspan="3">Alternative mode</td></tr>
<tr><td rowspan="3">UNITS:</td><td colspan="2">Form-truing selection, new/continue</td></tr>
<tr><td>1 =</td><td>New</td></tr>
<tr><td>2 =</td><td>Continue</td></tr>
<tr><td rowspan="3">TENS:</td><td colspan="2">Select form-truing allowance</td></tr>
<tr><td>0 =</td><td>From the rough contour to the lowest point of the contour</td></tr>
<tr><td>1 =</td><td>From the rough contour to the highest point of the contour</td></tr>
<tr><td>13</td><td></td><td><S_FW></td><td>REAL</td><td colspan="3">Clear angle of the dressing tool</td></tr>
<tr><td>14</td><td></td><td><S_HW></td><td>REAL</td><td colspan="3">Holder angle of the dressing tool</td></tr>
</table>

Note

The contour interpretation when profiling with CYCLE495 is carried out using the predefined procedure CONTDCON. Since CONTDCON is **not** permitted when tool radius compensation (G41/G42) is active, tool radius compensation must be deactivated with G40 before calling CYCLE495.

4.24.1.33 CYCLE782 - adjust to load

Note

To use CYCLE782, a license is required for the option "Intelligent load adjustment":

Syntax

```
CYCLE782(<S_MODE>, <S_TESTAXIS>, <S_VALUE>)
```

Parameters

No.	Parame-ter mask	Parameter internal	Data type	Meaning		
1		<S_MODE>	INT	Processing mode		
				UNITS:	Select/deselect	
					0 =	Deactivate adaptation
					1 =	Activate adaptation
					2 =	Determine and activate the basic inertia of the axis (commissioning function)
				TENS:	Measuring version (only for <S_MODE> HUNDREDS POSITION = 0)	
					0 =	Standard mode
					1 =	Precise mode (e.g. where friction is determined, etc.)
				HUNDREDS	A moment of inertia is measured or specified	
					0 =	measurement In this mode, traversing motion is executed to determine the load.
					1 =	A moment of inertia is specified In this mode, traversing motion is not executed to determine the load. Instead, a moment of inertia, for example determined by the automatic servo tuning function (AST), is transferred.
				THOUSANDS	Measurement result display	
					0 =	Measurement result screen OFF
					1 =	Measurement result screen ON
				TEN THOUSANDS	Duration of measurement result display	
					1 =	The display disappears automatically after 8 s
					3 =	Acknowledgment with NC start
				HUNDRED THOUSAND	Loading selection [1)]	
					0 =	Determining the complete loading process: Axis empty + workpiece
					1 =	Determining the loading separately: Workpiece
2		<S_TESTAXIS>	AXIS	Channel axis: • that should be activated for the adaptations • for which the moment of inertia should be determined (only for <S_MODE> HUNDREDS POSITION = 0)		
3		<S_VALUE>	REAL	Moment of inertia value (only for <S_MODE> HUNDREDS POSITION = 1)		

1) value is automatically taken from machine data 52212 $MCS_FUNCTION_MASK_TECH bit 18.

Note

The commissioning function "Determine and activate basic inertia of the axis" (<S_MODE> UNIT PLACE = 2) is not supported by the user interface. The cycle call must be programmed for this. The measurement result is entered into the machine data 53350 $MAS_ILC_BASE_VALUE .

The value of the parameter "Selection of loading" (<S_MODE> ONE HUNDRED THOUSANDS DIGIT) is read by the user interface from the machine data 52212 $MCS_FUNCTION_MASK_TECH Bit 18 and entered in the cycle call.

Loading can be determined or specified for linear and for rotary axes. For linear axes, a mass, and for rotary axes, a moment of inertia is always displayed in the measurement result screen or specified in the user interface.

Independent of the axis type, a moment of inertia is always written to machine data 53350 $MAS_ILC_BASE_VALUE and to the adaptation.

Examples

Example 1:

At the start of an NC program, the load level of axis MX1 should be determined – and this value used as basis to update the adaptations. The result should be displayed and the display acknowledged with an NC start.

Program code

```
...
CYCLE782(31011,MX1)
...
```

Example 2:

During the course of the program, the moment of inertia value, for example determined using AST, should be transferred to variable _MY_VALUE, the load cycle called with this value and therefore adaptations activated for MX2. A result should not be displayed.

Program code

```
DEF REAL _MY_VALUE
_MY_VALUE=...
...
CYCLE782(101,MX2,_MY_VALUE)
...
```

Example 3:

Adaptations for axis MX2 should be deactivated.

Program code

```
...
CYCLE782(0,MX2)
...
```

4.24.1.34 CYCLE800 – swivel plane / swivel tool / align tool

Syntax

```
CYCLE800(<_FR>, <_TC>, <_ST>, <_MODE>, <_X0>, <_Y0>, <_Z0>, <_A>,
<_B>, <_C>, <_X1>, <_Y1>, <_Z1>, <_DIR>, <_FR_I>, <_DMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1		`<_FR>`	INT	Retraction mode:	0 =	No retraction
					1 =	Retraction machine axis Z
					2 =	Retraction machine axis Z and then XY
					3 =	Reserved
					4 =	Maximum retraction in tool direction
					5 =	Incremental retraction in tool direction
2		`<_TC>`	STRING[32]	Name of swivel data block:	""	"" (no name) if only one swivel data block exists
					"0"	Deselect swivel data block (delete the swivel frames)

No.	Parameter mask	Parameter internal	Data type	Meaning		
3		<_ST>	INT	Status transformations		
				UNITS:		
					0 =	New, swivel level is deleted and recalculated using the current parameters
					1 =	Additive, swivel level is added to active swivel level
				TENS:	Track tool tip yes/no (only active when the SWIVEL function is created in the commissionig)	
					0 =	Do not track tool tip
					1 =	Track tool tip (TRAORI)
				HUNDREDS:	Approach/align tool (function is shown in tool swivel screen form)	
					0 =	Do not approach tool
					1 =	Approach tool (preferably radial mill)
					2 =	Align **turning** tool (when B axis kinematic is set up for milling in commissioning swiveling)
					3 =	Align **milling** tool (when B axis kinematic is set up for milling in commissioning swiveling)
				THOUSANDS:	Internal "Swiveling in JOG" parameter	
				TEN THOUSANDS:	See direction parameter <_DIR>	
					0 =	Swivel "Yes"
					1 =	Swivel "No", "Minus" direction[3)]
					2 =	Swivel "No", "Plus" direction[3)]
				HUNDRED THOUSANDS:	See direction parameter <_DIR>	
					0 =	Compatibility
					1 =	Direction selection "Minus" optimized (only for user interface) [4)]
					2 =	Direction selection "Plus" optimized (only for user interface) [4)]

No.	Parameter mask	Parameter internal	Data type	Meaning		
4		<_MODE> 5)	INT	Swivel mode: Evaluation of swivel angle and swivel sequence (bit-coded)		
				Bit: 7 6	0 0:	Swivel angle axis-by-axis -> see parameters <_A>, <_B>, <_C>
					0 1:	Solid angle -> see parameters <_A>, <_B> 1)
					1 0:	Projection angle -> see parameters <_A>, <_B>, <_C> 1)
					1 1:	Direct rotary axis swivel mode -> see parameters <_A>, <_B> 1)
				Bit: 5 4 3 2 1 0 (these do not apply to solid angles)	x x x x 0 1	1st rotation _A around X
					x x x x 1 0	1st rotation _A around Y
					x x x x 1 1	1st rotation _A around Z
					x x 0 1 x x	2nd rotation _B around X
					x x 1 0 x x	2nd rotation _B around Y
					x x 1 1 x x	2nd rotation _B around Z
					0 1 x x x x	3rd rotation _C around X
					1 0 x x x x	3rd rotation _C around Y
					1 1 x x x x	3rd rotation _C around Z
5	X0	<_X0>	REAL	Reference point X prior to rotation		
6	Y0	<_Y0>	REAL	Reference point Y prior to rotation		
7	Z0	<_Z0>	REAL	Reference point Z prior to rotation		
8	X(A)	<_A>	REAL	1st rotation acc. to setting in parameter <_MODE>		
9	Y(B)	<_B>	REAL	2nd rotation acc. to setting in parameter <_MODE>		
10	Z(C)	<_C>	REAL	3rd rotation acc. to setting in parameter <_MODE>		
11	X1	<_X1>	REAL	Reference point X after rotation		
12	Y1	<_Y1>	REAL	Reference point Y after rotation		
13	Z1	<_Z1>	REAL	Reference point Z after rotation		
14	- or +	<_DIR>	INT	Initiate travel of rotary axes (default = -1!)		
					-1 =	Position at smaller value of rotary axis 1 or 2 2)
					+1 =	Position at larger value of rotary axis 1 or 2 2)
					0 =	Do not swivel (merely calculate swivel frame) 1) 3)
15	FR	<_FR_I>	REAL	Value (inc) of retraction in tool direction incremental		

No.	Parameter mask	Parameter internal	Data type	Meaning		
16		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Representation of the beta value during align tool	
					0 =	Value
					1 =	Arrow

Note

If the following transfer parameters are programmed indirectly (as parameters), the screen form is not reset: `<_FR>, <_ST>, <_TC>, <_MODE>, <_DIR>`

1) Can be selected if the SWIVEL function is created in the commissioning

2) Can be selected if direction reference to rotary axis 1 or 2 is set in IBN SWIVEL

If direction reference is "No" there is no selection field

3) Swivel selection "No" can be grayed out SD 55221 Bit 0

Swivel "No", "Minus" direction corresponds to `<_DIR>` = 0 and _ST TEN THOUSANDS = 1

Swivel "No", "Plus" direction corresponds to `<_DIR>` = 0 and _ST TEN THOUSANDS = 2

4) The direction selection for rotary axis 1 or 2 also occurs if the rotary axis with the direction reference is in the pole position (position value equals zero).

5) Coding example: Axis-by-axis rotation, rotary sequence ZYX

Binary: 00011011 Decimal: 27

The axis identifiers XYZ correspond to the geometry axes of the NC channel. Individual rotations around the XYZ axes are permissible. For example, rotary sequence around ZXZ is not permitted in one call of CYCLE800

4.24.1.35 CYCLE801 – grid or frame position pattern

Syntax

```
CYCLE801(<_SPCA>, <_SPCO>, <_STA>, <_DIS1>, <_DIS2>, <_NUM1>,
<_NUM2>, <_VARI>, <_UMODE>, <_ANG1>, <_ANG2>, <_HIDE>, <_NSP>,
<_DMODE>)
```

Parameters

No.	Parameter mask	Parameters internal	Data type	Meaning		
1	X0	<_SPCA>	REAL	Reference point for position pattern (grid/frame) along the 1st axis (abs)		
2	Y0	<_SPCO>	REAL	Reference point for position pattern (grid/frame) along the 2nd axis (abs)		
3	α0	<_STA>	REAL	Basic angle of rotation (angle to 1st axis)	< 0 =	Clockwise rotation
					> 0 =	Counterclockwise rotation
4	L1	<_DIS1>	REAL	Distance between columns (position distance from the 1st axis, enter without sign)		
5	L2	<_DIS2>	REAL	Distance between rows (distance from the 2nd axis, enter without sign)		
6	N1	<_NUM1>	INT	Number of columns		
7	N2	<_NUM2>	INT	Number of rows		
8		<_VARI>	INT	Machining type		
				UNITS:	Position pattern	
					0 =	Grid
					1 =	Frame
				TENS:	Reserved	
				HUNDREDS:	Reserved	
9		<_UMODE>	INT	Reserved		
10	αX	<_ANG1>	REAL	Shear angle to 1st axis (lines inclined in relation to the 1st axis)		
					< 0 =	Clockwise measurement (0 to -90 degrees)
					> 0 =	Counter-clockwise measurement (0 to 90 degrees)
11	αY	<_ANG2>	REAL	Shear angle to 2nd axis (columns inclined in relation to the 2nd axis)		
					< 0 =	Clockwise measurement (0 to -90 degrees)
					> 0 =	Counter-clockwise measurement (0 to 90 degrees)
12		<_HIDE>	STRING [200]	Hidden positions • Max. 198 characters • Specification of consecutive position numbers, e.g. "1,3" (positions 1 and 3 are not executed)		
13		<_NSP>	INT	Reserved		
14		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)

4.24.1.36 CYCLE802 - arbitrary positions

Syntax

```
CYCLE802(<_XA>, <_YA>, <_X0>, <_Y0>, <_X1>, <_Y1>, <_X2>, <_Y2>,
<_X3>, <_Y3>, <_X4>, <_Y4>, <_X5>, <_Y5>, <_X6>, <_Y6>, <_X7>,
<_Y7>, <_X8>, <_Y8>, <_VARI>, <_UMODE>, <_DMODE>, <S_ABA>, <S_AB0>,
<S_AB1>, <S_AB2>, <S_AB3>, <S_AB4>, <S_AB5>, <S_AB6>, <S_AB7>,
<S_AB8>)
```

Parameters

No.	Parameter mask	Parameters internal	Data type	Meaning
1		<_XA>	INT	Alternatives for all X positions (9-digit decimal value) Number of digits: 876543210 (digit position corresponds to drilling position Xn) Position value: 1 = Absolute (1st programmed position is always absolute); 2 = Incremental
2		<_YA>	INT	Alternatives for all Y positions (9-digit decimal value) Number of digits: 876543210 (digit position corresponds to drilling position Yn) Position value: 1 = Absolute (1st programmed position is always absolute); 2 = Incremental
3	X0	<_X0>	REAL	1. Position X
4	Y0	<_Y0>	REAL	1. Position Y
5	X1	<_X1>	REAL	2. Position X
6	Y1	<_Y1>	REAL	2. Position Y
7	X2	<_X2>	REAL	3. Position X
8	Y2	<_Y2>	REAL	3. Position Y
9	X3	<_X3>	REAL	4. Position X
10	Y3	<_Y3>	REAL	4. Position Y
11	X4	<_X4>	REAL	5. Position X
12	Y4	<_Y4>	REAL	5. Position Y
13	X5	<_X5>	REAL	6. Position X
14	Y5	<_Y5>	REAL	6. Position Y
15	X6	<_X6>	REAL	7. Position X
16	Y6	<_Y6>	REAL	7. Position Y
17	X7	<_X7>	REAL	8. Position X
18	Y7	<_Y7>	REAL	8. Position Y
19	X8	<_X8>	REAL	9. Position X
20	Y8	<_Y8>	REAL	9. Position Y

No.	Parameter mask	Parameters internal	Data type	Meaning		
21		<_VARI>	INT	Machining		
				HUNDREDS:	(Only for call from Jobshop) (At present only 0 and 2 evaluated)	
					0 =	Do not clamp spindle
					1 =	Only clamp spindle for vertical insertion with G00 or G01
					2 =	Clamp spindle during the entire machining operation
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Position pattern with/without rotary axis – axis combination (with <_VARI> HUNDRED THOUSANDS)	
					0 =	XY (only XY without rotary axis, compatibility)
					1 =	X,Y or Z and rotary axis: XA, YB, ZC (1 rotary axis with geometry axis around which the rotary axis rotates)
					2 =	XY and rotary axis: XYA, XYB, XYC (1 rotary axis with 1st and 2nd geometry axis, without TRACYL)
				HUNDRED THOUSANDS:	Rotary axis	
					0 =	Without rotary axis (only XY, compatibility)
					1 =	A axis (rotary axis around X)
					2 =	B axis (rotary axis around Y)
					3 =	C axis (rotary axis around Z)
				TEN MILLIONS + ONE MILLION:	Position pattern with rotary axis – offset (for several rotary axes around the same axis; if index too large, then 1st axis)	
					00 =	1st A, B or C axis or for compatibility
					01 =	2nd A, B or C axis
					...	
					19 =	20th A, B or C axis
22		<_UMODE>	INT	Selection of the spindle to be clamped: (Only for call from Jobshop) (Call of user cycle CUST_TECHCYC)		
					3 =	Clamp/release main spindle
					23 =	Clamp/release counterspindle

No.	Parameter mask	Parameters internal	Data type	Meaning		
23		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
24		<S_ABA>	INT	Alternatives for all AB positions (9-digit decimal value) Number of digits: 876543210 (digit position corresponds to position ABn)		
				Position value:	1 =	Absolute (1st programmed position is always absolute)
					2 =	Incremental
25	A0	<S_AB0>	REAL	1st rotary axis position for position pattern with rotary axis (in conjunction with <_VARI>))		
26	A1	<S_AB1>	REAL	2nd rotary axis position for position pattern with rotary axis		
27	A2	<S_AB2>	REAL	3rd rotary axis position for position pattern with rotary axis		
28	A3	<S_AB3>	REAL	4th rotary axis position for position pattern with rotary axis		
29	A4	<S_AB4>	REAL	5th rotary axis position for position pattern with rotary axis		
30	A5	<S_AB5>	REAL	6th rotary axis position for position pattern with rotary axis		
31	A6	<S_AB6>	REAL	7th rotary axis position for position pattern with rotary axis		
32	A7	<S_AB7>	REAL	8th rotary axis position for position pattern with rotary axis		
33	A8	<S_AB8>	REAL	9th rotary axis position for position pattern with rotary axis		

Note

Positions that are not required for parameters X1/Y1/A1 to X8/Y8/A8 can be ignored. The alternative values for <_XA>, <_YA> and <S_ABA>, however, must be provided in full for all 9 positions.

For position pattern XA, YB or ZC (a geometry axis and rotary axis), the axis of the machining plane that is not traversed via the position pattern (Y for G17 and XA) must be positioned before the cycle call.

4.24.1.37 CYCLE805 - Y turning

Note

For Y turning with CYCLE805, option "Turning with the Y axis" is required, for which a license is required!

Syntax

```
CYCLE805(<_MODE>, < _TC>, < _GAMA>, <_FR_I>, <S_ZS>, <S_XS>,
<_AMODE>)
```

Parameters

<table>
<tr><th>No.</th><th>Parameter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td rowspan="8">1</td><td></td><td rowspan="8"><_MODE></td><td rowspan="8">INT</td><td colspan="3">Mode</td></tr>
<tr><td rowspan="3">Selecting/ deselecting</td><td rowspan="3">UNITS:</td><td colspan="2">Selecting/deselecting</td></tr>
<tr><td>0 =</td><td>Deselect Y turning</td></tr>
<tr><td>1 =</td><td>Select Y turning</td></tr>
<tr><td></td><td>TENS:</td><td colspan="2">Reserved</td></tr>
<tr><td rowspan="3">Follow-up</td><td rowspan="3">HUNDREDS:</td><td colspan="2">Track tool tip yes/no (only active when the "Swivel with CYCLE800" is set up on the machine)</td></tr>
<tr><td>0 =</td><td>Do not track tool tip</td></tr>
<tr><td>1 =</td><td>Track tool tip (TRAORI)</td></tr>
<tr><td rowspan="4">2</td><td rowspan="4">TC</td><td rowspan="4"><_TC></td><td rowspan="4">STRING[32]</td><td colspan="3">Name of the swivel data set</td></tr>
<tr><td rowspan="3"></td><td>"<Name>"</td><td>If several swivel data sets are available</td></tr>
<tr><td>""
(no name, empty string)</td><td>If only one swivel data set is available</td></tr>
<tr><td>"0"</td><td>Deselect Y turning</td></tr>
<tr><td>4</td><td>Gamma</td><td><_GAMA></td><td>REAL</td><td colspan="3">Angle gamma (rotation around the tool axis) / alignment angle of the cutting plate</td></tr>
<tr><td>5</td><td>FR</td><td><_FR_I></td><td>REAL</td><td colspan="3">Retract in the tool direction (incremental)</td></tr>
<tr><td>6</td><td>ZS</td><td><S_ZS></td><td>REAL</td><td colspan="3">Start/retract point Z
(see parameter <_AMODE>, "UNITS" and "HUNDREDS")</td></tr>
<tr><td>7</td><td>XS/YR</td><td><S_XS></td><td>REAL</td><td colspan="3">Start point X / retract point Y
(see parameter <_AMODE>, "TENS" and "HUNDREDS")</td></tr>
</table>

No.	Parame-ter mask	Parameter in-ternal	Data type	Meaning		
8		<_AMODE>	INT	Alternative mode		
				UNITS:	Start/retract point Z: Reference to the position specification	
					0 =	Absolute
					1 =	Incremental
				TENS:	Start point X / retract point Y: Reference to the position specification	
					0 =	absolute (diameter)
					1 =	Incremental (radius)
				HUNDREDS:	Start/retract point yes/no	
					0 =	No
					1 =	yes
				THOUSANDS:	Working range (only active if MD52218 bit 17 = 1)	
					0 =	Working range 1 (WCS rotation around Z: -90°)
					1 =	Working range 2 (WCS rotation around Z: +90°)

4.24.1.38 CYCLE806 - Interpolation turning

Note

Interpolation turning with CYCLE806 is an option that requires a license.

Syntax

```
CYCLE806(<_MODE>, <S_XC>, <S_YC>, <_STA>, <_SV1>, <_SV2>, <_SDIR>,
<_AMODE>, <_DMODE>, <S_TRA>, <S_XS>, <S_ZS>)
```

Parameters

No.	Parame-ter mask	Parameter in-ternal	Data type	Meaning		
1		<_MODE>	INT	Mode		
				UNITS:	Selecting/deselecting	
					0 =	Deselect interpolation turning
					1 =	Select interpolation turning
2	XC	<S_XC>	REAL	Center of rotation X		
3	YC	<S_YC>	REAL	Center of rotation Y		
4	Gamma	<_STA>	REAL	Angle gamma (rotation around the tool axis); reserved		
5	S	<_SV1>	REAL	Constant spindle speed or constant cutting speed (see <_AMODE> UNITS)		
6	PS	<_SV2>	REAL	Maximum speed at constant cutting speed (reserved)		

No.	Parameter mask	Parameter internal	Data type	Meaning		
7		<_SDIR>	INT	Reserved		
					0 =	Direction of rotation from tool
					3 =	Direction of rotation M3
					4 =	Direction of rotation M4
8		<_AMODE>	INT	Alternative mode		
				UNITS:	Spindle movement (<_SV1>)	
					0 =	Constant spindle speed
					1 =	Constant cutting speed
				HUNDREDS:	Starting point	
					0 =	No
					1 =	Yes
9		<_DMODE>	INT	Display mode		
				UNITS:	Working plane G17/18/19	
					1 =	G17 (only active in the cycle)
10	TC	<S_TRA>	STRING	Name of transformation Empty: Use 1st kinematics transformation TRAINT		
11	XS	<S_XS>	REAL	Starting point X (diameter) (see <_AMODE> HUNDREDS)		
12	ZS	<S_ZS>	REAL	Starting point Z (see <_AMODE> HUNDREDS)		

4.24.1.39 CYCLE830 - deep-hole drilling 2

Syntax

```
CYCLE830(<RTP>, <RFP>, <SDIS>, <_DP>, <FDEP>, <_DAM>, <DTB>,
<DTS>, <FRF>, <VARI>, <_MDEP>, <_VRT>, <_DTD>, <_DIS1>, <S_FP>,
<S_SDAC2>, <S_SV2>, <S_FB>, <_SDAC>, <_SV1>, <S_SPOS>, <S_ZA>,
<S_FA>, <S_ZP>, <S_FS>, <S_ZS1>, <S_ZS2>, <S_N>, <S_ZD>, <S_FD>,
<S_FR>, <S_SDAC3>, <S_SV3>, <S_CON>, <S_COFF>, <_GMODE>, <_DMODE>,
<_AMODE>, <S_AMODE2>, <S_AMODE3>, <S_ZPV>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<RTP>	REAL	Retraction plane (abs)
2	Z0	<RFP>	REAL	Reference point (abs)
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, without sign)
4	Z1	<_DP>	REAL	Final drilling depth abs/inc (see <_AMODE>UNITS)
5	D	<FDEP>	REAL	1st drilling depth for the absolute or incremental chip breaking/removal in relation to the reference point with/without predrilling or in relation to pilot hole depth (see <_AMODE> TEN THOUSANDS)

No.	Parameter mask	Parameter internal	Data type	Meaning		
6	DF	<_DAM>	REAL	Absolute value / percentage for each additional infeed, degression absolute value / percentage (see <_AMODE> HUNDRED THOUSANDS)		
7	DTB	<DTB>	REAL	Dwell time at each drilling depth (see <_AMODE> TENS)		
8	DTS	<DTS>	REAL	Dwell time during chip removal at starting point (see <_AMODE> HUNDREDS)		
9	FD1	<FRF>	REAL	Percentage for the feedrate for the first infeed (see <_AMODE> TEN MILLION)		
10		<VARI>	INT	Machining		
				UNITS:	Chip breaking / swarf removal	
					0 =	In one cut
					1 =	Chip breaking
					2 =	Swarf removal
					3 =	Chip breaking and swarf removal
				TENS:	Retraction during swarf removal	
					0 =	To pilot hole depth
					1 =	To safety clearance
				HUNDREDS:	Soft first cut	
					0 =	No
					1 =	Yes
				THOUSANDS:	Through drilling	
					0 =	No
					1 =	Yes
				TEN THOUSANDS:	Predrilling / pilot hole	
					0 =	Without predrilling
					1 =	With predrilling
					2 =	With pilot hole
				HUNDRED THOUSANDS:	Retraction	
					0 =	To pilot hole depth
					1 =	To retraction plane
11	V1	<_MDEP>	REAL	Minimum incremental infeed (only for degression percentage)		
12	V2	<_VRT>	REAL	Retraction distance after each incremental machining step (for chip breaking only)		
					0 =	Default value 1 mm
					> 0 =	Variable retraction distance
13	DT	<_DTD>	REAL	Dwell time at final drilling depth (see <_AMODE> THOUSANDS)		
14	V3	<_DIS1>	REAL	Incremental limit distance for chip removal only (see <_AMODE> ONE-MILLION)		
15	FP	<S_FP>	REAL	Feedrate for travel into the pilot hole as value or in % (in conjunction with <S_AMODE2> HUNDREDS)		
16		<S_SDAC2>	INT	Direction of spindle rotation during approach		
					3 =	M3
					4 =	M4
					5 =	M5 (default)

No.	Parameter mask	Parameter internal	Data type	Meaning		
17	SP	<S_SV2>	REAL	Approach with	constant spindle speed (see <S_AMODE2> TEN MILLION)	
	V4				constant cutting rate	
				Spindle speed in % of the drilling speed		
18	F	<S_FB>	REAL	Drilling feedrate (see <S_AMODE2> UNITS)		
19		<_SDAC>	REAL	Direction of spindle rotation during drilling		
					3 =	M3
					4 =	M4
20	S	<_SV1>	REAL	Drilling with	constant spindle speed (see <S_AMODE2> ONE MILLION)	
	V5				constant cutting rate	
21	SPOS	<S_SPOS>	REAL	Spindle position, only if approach with M5		
22	ZA	<S_ZA>	REAL	Incremental predrilling depth in relation to reference point or absolute (see <S_AMODE3> UNITS)		
23	FA	<S_FA>	REAL	Predrilling feedrate as value or in % (in conjunction with <S_AMODE2> TENS)		
24	ZP	<S_ZP>	REAL	Incremental pilot hole in relation to reference point or absolute or factor of the hole diameter (see <S_AMODE3> TENS)		
25	FS	<S_FS>	REAL	First cut feedrate as value or in % (in conjunction with <S_AMODE2> THOUSANDS)		
26	ZS1	<S_ZS1>	REAL	Depth of each first cut with constant feedrate (inc)		
27	ZS2	<S_ZS2>	REAL	Depth of each first cut for feedrate increase (inc)		
28	N	<S_N>	INT	Number of chip breaking strokes before each chip removal		
29	ZD	<S_ZD>	REAL	Incremental remaining drilling depth in relation to final drilling depth or absolute (see <S_AMODE3> HUNDREDS)		
30	FD	<S_FD>	REAL	Remaining drilling feedrate as value or in % (in conjunction with <S_AMODE2> TEN THOUSANDS)		
31	FR	<S_FR>	REAL	Retraction feedrate (in conjunction with <S_AMODE2> HUNDRED THOUSANDS)		
32		<S_SDAC3>	INT	Direction of spindle rotation during retraction		
					3 =	M3
					4 =	M4
					5 =	M5
33	SR	<S_SV3>	REAL	Retraction with	constant spindle speed (see <S_AMODE2> HUNDRED MILLION)	
	V6				constant cutting rate	
				Spindle speed in % of the drilling speed		
34	Coolant on	<S_CON>	STRING[10]	Coolant on, M command or subprogram call		
35	Coolant off	<S_COFF>	STRING[10]	Coolant off, M command or subprogram call		
36		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Drilling depth with respect to tip/shank	
					0 =	Tip
					1 =	Shank

No.	Parameter mask	Parameter internal	Data type	Meaning		
37		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Basic
38		<_AMODE>	INT	Alternative mode 1		
				UNITS:	Drilling depth = Final drilling depth Z1 abs/inc	
					0 =	Incremental
					1 =	Absolute
				TENS:	Dwell time at each drilling depth DTB in seconds/revolutions	
					0 =	In seconds
					1 =	In revolutions
				HUNDREDS:	Dwell time for chip removal DTS in seconds/revolutions	
					0 =	In seconds
					1 =	In revolutions
				THOUSANDS:	Dwell time at final drilling depth DT in seconds/revolutions	
					0 =	In seconds
					1 =	In revolutions
				TEN THOUSANDS:	1st drilling depth D abs/inc	
					0 =	Incremental
					1 =	Absolute
				HUNDRED THOUSANDS:	Absolute value / percentage DF for each additional infeed (degression)	
					0 =	Absolute value
					1 =	Percentage (0.001 to 100%)
				ONE MILLION:	Limit distance V3 automatic/manual	
					0 =	Automatic (calculated in the cycle)
					1 =	Manual (programmed value)

No.	Parameter mask	Parameter internal	Data type	Meaning		
39		<S_AMODE2>	INT	Alternative mode 2		
				UNITS:	UNITS: Drilling feedrate F	
					0 =	F/min
					1 =	F/rev
				TENS:	Evaluation of predrilling feedrate FA	
					0 =	As % of drilling feedrate
					1 =	F/min
					2 =	F/rev
				HUNDREDS:	Evaluation of feedrate for travel into pilot hole FP	
					0 =	As % of drilling feedrate
					1 =	F/min
					2 =	F/rev
				THOUSANDS:	Evaluation of first cut feedrate FS	
					0 =	As % of drilling feedrate
					1 =	F/min
					2 =	F/rev
				TEN THOUSANDS:	Evaluation of through-drilling feedrate FD	
					0 =	As % of drilling feedrate
					1 =	F/min
					2 =	F/rev
				HUNDRED THOUSANDS:	Retraction feedrate FR	
					0 =	F/min
					1 =	Rapid traverse
				ONE MILLION:	Drilling - spindle speed / cutting rate (S/V5)	
					0 =	Constant spindle speed
					1 =	Constant cutting rate
				TEN MILLIONS:	Approach with spindle speed / cutting rate (SP/V4)	
					0 =	Constant spindle speed
					1 =	Constant cutting rate
					2 =	Spindle speed in % of the drilling speed
				HUNDRED MILLIONS:	Retraction - spindle speed / cutting rate (SR/V6)	
					0 =	Constant spindle speed
					1 =	Constant cutting rate
					2 =	Spindle speed in % of the drilling speed

No.	Parameter mask	Parameter internal	Data type	Meaning		
40		<S_AMODE3>	INT	Alternative mode 3		
				UNITS:	Drilling depth ZA abs/inc	
					0 =	Incremental
					1 =	Absolute
				TENS:	Depth of the pilot hole ZP	
					0 =	Incremental
					1 =	Absolute
					2 =	Factor of the hole diameter
				HUNDREDS:	Remaining drilling depth ZD abs/inc	
					0 =	Incremental
					1 =	Absolute
41	ZPV	<S_ZPV>	REAL	Incremental limit distance from pilot hole depth		

4.24.1.40 CYCLE832 - High-Speed Settings

Syntax

```
CYCLE832(<S_TOL>, <S_TOLM>, <S_OTOL>)
```

Note

CYCLE832 does not relieve the machine manufacturer from optimization tasks that are necessary when commissioning the machine. This involves the optimization of the axes involved in the machining process and NCU settings (precontrol, jerk limiting, etc.).

Parameters

<table>
<tr><th>No.</th><th>Parame-ter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="3">Meaning</th></tr>
<tr><td>1</td><td>Tolerance</td><td><S_TOL></td><td>REAL</td><td colspan="3">Contour tolerance
The contour tolerance corresponds to the axis tolerance of the geometry axes.</td></tr>
<tr><td rowspan="15">2</td><td rowspan="15"></td><td rowspan="15"><S_TOLM></td><td rowspan="15">INT</td><td colspan="3">Machining type (technology)</td></tr>
<tr><td rowspan="6">UNITS:</td><td colspan="2"></td></tr>
<tr><td>0 =</td><td>Deselection</td></tr>
<tr><td>1 =</td><td>Finishing</td></tr>
<tr><td>2 =</td><td>Rough finishing (semi-finishing)</td></tr>
<tr><td>3 =</td><td>Roughing</td></tr>
<tr><td>4 =</td><td>Smooth finishing (precision)</td></tr>
<tr><td rowspan="3">TENS:</td><td colspan="2"></td></tr>
<tr><td>0 =</td><td>Compatibility[1] or no orientation tolerance</td></tr>
<tr><td>1 =</td><td>Orientation tolerance in parameter <S_OTOL></td></tr>
<tr><td>HUNDREDS
...
HUNDRED THOUSANDS</td><td colspan="2">Assigned
for reasons of
compatibility</td></tr>
<tr><td rowspan="4">ONE MILLION:</td><td colspan="2"></td></tr>
<tr><td>0 =</td><td>Compatibility. The best available mold making function is automatically used:
• Option Top Surface not active:
□ Advanced Surface
• Option Top Surface active:
⇒ Top Surface with smoothing</td></tr>
<tr><td>1 =</td><td>Top Surface without smoothing</td></tr>
<tr><td>2 =</td><td>Top Surface with smoothing</td></tr>
<tr><td>3</td><td>ORI tolerance</td><td><S_OTOL></td><td>REAL</td><td colspan="3">Orientation tolerance or version identifier CYCLE832
Tolerance parameter for the orientation of the workpiece.
Is required when executing a high-speed machining program on machines with dynamic orientation transformation (e.g. 5-axis machining).
Parameter <S_OTOL> must be programmed. This also applies for applications on 3-axis machines for programs without orientation of the tool (<S_OTOL> = 1).</td></tr>
</table>

1) Orientation tolerance derived from the cycle setting data SD55451 ... SD55454 (orientation tolerance for dynamic response mode...) or SD55445 ... SD55449 (contour tolerance for dynamic response mode...) multiplied by the factor from SD55441 ... SD55444.
Further information: SINUMERIK Operate Commissioning Manual

Plain text entry

To improve the readability of the cycle call, parameter <S_TOLM> (machining type) can also be entered in the plain text. Plain texts are independent of any language. The following entries are permitted:

`_OFF`	for	0	Deselection
`_FINISH`	for	1	Finishing
`_SEMIFIN`	for	2	Rough finishing
`_ROUGH`	for	3	Roughing
`_PRECISION`	for	4	Smooth finishing
`_ORI_FINISH`	for	11	Finishing with input of an orientation tolerance
`_ORI_SEMIFIN`	for	12	Semi-finishing with input of an orientation tolerance
`_ORI_ROUGH`	for	13	Roughing with input of an orientation tolerance
`_ORI_PRECISION`	for	14	Smooth finishing with input of an orientation tolerance
`_TOP_SURFACE_SMOOTH_OFF`	for	1000000	Top Surface without smoothing
`_TOP_SURFACE_SMOOTH_ON`	for	2000000	Top Surface with smoothing

For plain text input for Top Surface, plain texts are combined as shown in the following example:

```
CYCLE832(0.1, _TOP_SURFACE_SMOOTH_OFF+_ORI_FINISH, 1)
```

Note

The plain texts are based on the function names of the G group 59 (dynamic mode for path interpolation). With these plain texts, 3-axis machines and machines with multi-axis orientation transformation (TRAORI) are clearly separated in the application.

Deselecting CYCLE832

When CYCLE832 is deselected, parameter <S_TOL> must be transferred with zero.

Example: `CYCLE832(0,0,1)`

The syntax `CYCLE832()` is also permitted for deselecting CYCLE832.

Examples

Example 1: CYCLE832 on 3-axis machine without orientation transformation

a) Cycle call with plain text input

Program code	Comment
G710	; Dimension system is metric.

Program code	Comment
CYCLE832(0.004,_FINISH,1)	; CYCLE832 call with: Contour tolerance = 0.004 mm, machining type: Finishing
...	; Execution of a high-speed machining program

b) Cycle call without plain text input

Program code	Comment
G710	; See above
CYCLE832(0.004,1,1)	; See above
...	; See above

Example 2: CYCLE832 on 5-axis machine with orientation transformation

a) Cycle call and deselection with plain text input

Program code	Comment
G710	; Dimension system is metric.
TRAORI	; Activate orientation transformation.
CYCLE832(0.3,_ORI_ROUGH,0.8)	; CYCLE832 call with: Contour tolerance = 0.3 mm, machining type: Roughing with input of an orientation tolerance; orientation tolerance = 0.8 degrees
...	; Execution of a high-speed machining program
CYCLE832(0,_OFF,1)	; Contour tolerance = 0, machining type: Deselection of CYCLE832, orientation tolerance = 0 degrees

b) Cycle call and deselection without plain text input

Program code	Comment
G710	; See above
TRAORI	; See above
CYCLE832(0.3,13,0.8)	; See above
...	; See above
CYCLE832(0,0,1)	; See above

4.24.1.41 CYCLE840 - tapping with compensating chuck

Syntax

```
CYCLE840(<RTP>, <RFP>, <SDIS>, <DP>, <DPR>, <DTB>, <SDR>, <SDAC>,
<ENC>, <MPIT>, <PIT>, <_AXN>, <_PITA>, <_TECHNO>, <_PITM>, <_PTAB>,
<_PTABA>, <_GMODE>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	RP	<RTP>	REAL	Retraction plane (abs)
2	Z0	<RFP>	REAL	Reference point (abs)
3	SC	<SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)
4	Z1	<DP>	REAL	Drilling depth (abs), see <_AMODE>
5	Z1	<DPR>	REAL	Drilling depth (inc), see <_AMODE>
6	DT	<DTB>	REAL	Dwell time in seconds at drilling depth / safety clearance after retraction, see <ENC>
7		<SDR>	INT	Direction of rotation for retraction
8	SDE	<SDAC>	INT	Direction of rotation after end of cycle
9		<ENC>	INT	Tapping with spindle mounted encoder (G33)/tapping without spindle mounted encoder (G63) 0 = With spindle mounted encoder - Pitch from <MPIT>/<PIT> - without DT 20 = With spindle mounted encoder - Pitch from <MPIT>/<PIT> - with DT after retraction to safety clearance 11 = Without spindle mounted encoder - Pitch from <MPIT>/<PIT> - with DT at drilling depth 1 = Without spindle mounted encoder - Pitch from programmed feedrate - with DT at drilling depth (feedrate = speed · pitch)
10		<MPIT>	REAL	Thread size for "ISO metric" only (pitch is calculated internally during run time) Range of values: 3 to 48 (for M3 to M48), alternative to <PIT>
11		<PIT>	REAL	Pitch as a value, for unit see <_PITA> Range of values: > 0, alternative to MPIT
12		<_AXN>	INT	Drilling axis 0 = 3rd geometry axis 1 = 1st geometry axis 2 = 2nd geometry axis ≥ 3 = 3rd geometry axis

No.	Parameter mask	Parameter internal	Data type	Meaning			
13		<_PITA>	INT	Pitch unit (evaluation of <PIT> and <MPIT>)			
					0 =	Pitch in mm	- evaluation<MPIT>/<PIT>
					1 =	Pitch in mm	- evaluation<PIT>
					2 =	Pitch in TPI	- evaluation of <PIT> (threads per inch)
					3 =	Pitch in inches	- evaluation<PIT>
					4 =	MODULUS	- evaluation<PIT>
14		<_TECHNO>	INT	Technology[1)]			
				UNITS:	Exact stop response		
					0 =	Exact stop response active as before cycle call	
					1 =	Exact stop G601	
					2 =	Exact stop G602	
					3 =	Exact stop G603	
				TENS:	Feedforward control		
					0 =	With/without feedforward control active as before cycle call	
					1 =	With feedforward control FFWON	
					2 =	Without feedforward control FFWOF	
15		<_PITM>	STRING[15]	String as marker for pitch input[2)]			
16		<_PTAB>	STRING[5]	String for thread table ("", "ISO", "BSW", "BSP", "UNC")[2)]			
17		<_PTABA>	STRING[20]	String for selection from thread table (e.g. "M 10", "M 12", ...)[2)]			
18		<_GMODE>	INT	Reserved			

No.	Parameter mask	Parameter internal	Data type	Meaning		
19		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:	Compatibility mode (for recompilation screen form only), if MD 52216 bit0 = 1[1)]	
					0 =	Technology parameters are displayed (compatibility): TECHNO parameters effective
					1 =	Technology parameters are not displayed: Technology active "as before cycle call"
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple
20		<_AMODE>	INT	Alternative mode		
				UNITS:	Drilling depth Z1 (abs/inc)	
					0 =	Compatibility, from programming <DP>/<DPR>
					1 =	Incremental
					2 =	Absolute

1) Technology fields may be hidden, depending on the setting date SD52216 MCS_FUNCTION_MASK_DRILL

2) Parameters 15, 16 and 17 are only used for thread selection in the screen form thread tables. The thread tables cannot be accessed via cycle definition in cycle run time.

4.24.1.42 CYCLE899 – open slot

Syntax

```
CYCLE899(<_RTP>, <_RFP>, <_SDIS>, <_DP>, <_LENG>, <_WID>, <_PA>,
<_PO>, <_STA>, <_MID>, <_MIDA>, <_FAL>, <_FALD>, <_FFP1>, <_CDIR>,
<_VARI>, <_GMODE>, <_DMODE>, <_AMODE>, <_UMODE>, <_FS>, <_ZFS>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning		
1	RP	<_RTP>	REAL	Retraction plane (abs)		
2	Z0	<_RFP>	REAL	Reference point of tool axis (abs)		
3	SC	<_SDIS>	REAL	Safety clearance (to be added to reference point, enter without sign)		
4	Z1	<_DP>	REAL	Slot depth (abs/inc), see <_AMODE>		
5	L	<_LENG>	REAL	Length of slot (inc)		
6	W	<_WID>	REAL	Width of slot (inc)		
7	X0	<_PA>	REAL	Reference point/starting position 1st axis (abs)		
8	Y0	<_PO>	REAL	Reference point/starting position 2nd axis (abs)		
9	α0	<_STA>	REAL	Angle of rotation with respect to 1st axis		
10	DZ	<_MID>	REAL	Maximum infeed depth (inc), for vortex milling only		
11	DXY	<_MIDA>	REAL	Maximum plane infeed, see <_AMODE>		
12	UXY	<_FAL>	REAL	Finishing allowance, plane		
13	UZ	<_FALD>	REAL	Finishing allowance, depth		
14	F	<_FFP1>	REAL	Feedrate		
15		<_CDIR>	INT	Milling direction		
				UNITS:		
					0 =	Down-cut
					1 =	Up-cut
					4 =	Alternating
16		<_VARI>	INT	Machining		
				UNITS:		
					1 =	Roughing
					2 =	Finishing
					3 =	Base finishing
					4 =	Edge finishing
					5 =	Rough-finishing
					6 =	Chamfering
				TENS:	Reserved	
				HUNDREDS:	Reserved	
				THOUSANDS:		
					1 =	Vortex milling
					2 =	Plunge cutting

No.	Parameter mask	Parameter internal	Data type	Meaning		
17		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Reserved	
				TENS:	Reserved	
				HUNDREDS:	Select machining/only calculation of start point	
					1 =	Normal machining
				THOUSANDS:	Dimensioning via center/edge	
					0 =	Dimensioning via center
					1 =	"Left-hand" dimensioning using edge ("-" direction of 1st axis)
					2 =	"Right-hand" dimensioning using edge ("+" direction of 1st axis)
18		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	---	Reserved
				HUNDREDS:	---	Reserved
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: Simple
19		<_AMODE>	INT	Alternative mode		
				UNITS:	Slot depth Z1	
					0 =	Absolute
					1 =	Incremental
				TENS:	Unit for plane infeed (<_MIDA>) DXY	
					0 =	mm
					1 =	% of tool diameter
				HUNDREDS:	Insertion depth for chamfering ZFS	
					0 =	Absolute
					1 =	Incremental
20		<_UMODE>	INT	Reserved		
21	FS	<_FS>	REAL	Chamfer width (inc)		
22	ZFS	<_ZFS>	REAL	Insertion depth (tool tip) on chamfering (abs/inc), see <_AMODE>		

4.24.1.43 CYCLE930 - groove

Syntax

```
CYCLE930(<_SPD>, <_SPL>, <_WIDG>, <_WIDG2>, <_DIAG>, <_DIAG2>,
<_STA>, <_ANG1>, <_ANG2>, <_RCO1>, <_RCI1>, <_RCI2>, <_RCO2>,
<_FAL>, <_IDEP1>, <_SDIS>, <_VARI>, <_DN>, <_NUM>, <_DBH>, <_FF1>,
<_NR>, <_FALX>, <_FALZ>, <_DMODE>, <_AMODE>, <_GMODE>, <S_FB>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	X0	<_SPD>	REAL	Reference point in the plane axis (always diameter)
2	Z0	<_SPL>	REAL	Reference point along the longitudinal axis
3	B1	<_WIDG>	REAL	Width at bottom of groove
4	B2	<_WIDG2>	REAL	Width at top of groove (for interface only)
5	T1	<_DIAG>	REAL	Depth of groove at the reference point for abs and longitudinal machining = diameter, otherwise inc
6	T2	<_DIAG2>	REAL	Groove depth opposite the reference point (for interface only), for abs and longitudinal machining = diameter, otherwise inc
7	α0	<_STA>	REAL	Angle of inclination (-180 ≤ <_STA> ≤ 180)
8	α1	<_ANG1>	REAL	Side angle 1 (0 ≤ <_ANG1> < 90) at the side of the groove determined by the reference point
9	α2	<_ANG2>	REAL	Side angle 2 (0 ≤ <_ANG2> < 90) opposite the reference point
10	R1/FS1	<_RCO1>	REAL	Rounding radius or chamfer width 1, external at the reference point
11	R2/FS2	<_RCI1>	REAL	Rounding radius or chamfer width 2, internal at the reference point
12	R3/FS3	<_RCI2>	REAL	Rounding radius or chamfer width 3, internal opposite the reference point
13	R4/FS4	<_RCO2>	REAL	Rounding radius or chamfer width 4, external opposite the reference point
14	U	<_FAL>	REAL	Finishing allowance in X and Z, see <_VARI> (TEN THOUSANDS) (to be entered without sign)
15	D	<_IDEP1>	REAL	Maximum depth infeed on insertion (enter without sign) 0 = 1st cut directly to full depth > 0 = 1st cut <_IDEP1>, 2nd cut 2 <_IDEP1> etc.
16	SC	<_SDIS>	REAL	Safety clearance (enter without sign)

No.	Parameter mask	Parameter internal	Data type	Meaning		
17		<_VARI>	INT	Machining type		
				UNITS:	Reserved	
				TENS:	Machining process	
					1 =	Roughing
					2 =	Finishing
					3 =	Roughing and finishing
				HUNDREDS:	Position longitudinal/transverse external/ internal +Z/+Z and +X/-X	
					1 =	Longitudinal/external +Z
					2 =	Transverse/internal -X
					3 =	Longitudinal/internal +Z
					4 =	Transverse/internal +X
					5 =	Longitudinal/external -Z
					6 =	Transverse/external -X
					7 =	Longitudinal/internal -Z
					8 =	Transverse/external +X
				THOUSANDS:	Position of reference point	
					0 =	Upper reference point
					1 =	Lower reference point
				TEN THOUSANDS:	Define effect of finishing allowances	
					0 =	Finishing allowance U parallel to the contour
					1 =	Separate UX and UZ finishing al-lowances
18		<_DN>	INT	D number for 2nd cutting edge of the tool		
					> 0 =	D number for tool offset of 2nd edge of grooving tool
					0 =	No 2nd cutting edge programmed
19	N	<_NUM>	INT	Number of grooves (0 = 1 groove)		
20	DP	<_DBH>	REAL	Distance between grooves (only needed when <_NUM> > 1)		
21	F	<_FF1>	REAL	Feedrate		
22		<_NR>	INT	Identification for form of groove corresponds to vertical softkey for form selection		
					0 =	90° sides without chamfers/ rounding
					1 =	Inclined sides with chamfers/ rounding (without α0)
					2 =	As 1, but on taper (with α0)
23	UX	<_FALX>	REAL	Finishing allowance in X axis, see <_VARI> (TEN THOUSANDS) (to be entered without sign)		
24	UZ	<_FALZ>	REAL	Finishing allowance in Z axis, see <_VARI> (TEN THOUSANDS) (to be entered without sign)		

No.	Parameter mask	Parameter internal	Data type	Meaning			
25		<_DMODE>	INT	Display mode			
				UNITS:	Machining plane G17/G18/G19		
					0 =	Compatibility, the plane effective before the cycle call remains active	
					1 =	G17 (only active in the cycle)	
					2 =	G18 (only active in the cycle)	
					3 =	G19 (only active in the cycle)	
				TENS:	Technology mode comb grooving		
					0 =	No	
					1 =	Yes	
26		<_AMODE>	INT	Alternative mode			
				UNITS:	Dimensioning for top of groove (for interface only)		
					0 =	At the reference point	
					1 =	Opposite the reference point	
				TENS:	Depth		
					0 =	Absolute	
					1 =	Incremental	
				HUNDREDS:	Dimensioning for width (for interface only)		
					0 =	At outer diameter (top)	
					1 =	At inner diameter (bottom)	
				THOUSANDS:	Radius/chamfer 1 (<_RCO1>)		
					0 =	Radius	
					1 =	Chamfer	
				TEN THOUSANDS:	Radius/chamfer 2 (<_RCI1>)		
					0 =	Radius	
					1 =	Chamfer	
				HUNDRED THOUSANDS:	Radius/chamfer 3 (<_RCI2>)		
					0 =	Radius	
					1 =	Chamfer	
				ONE MILLION:	Radius/chamfer 4 (<_RCO2>)		
					0 =	Radius	
					1 =	Chamfer	
				TEN MILLIONS:	Type of feed FB		
					0 =	Feedrate value	
					1 =	Percentage value related to F	
27		<_GMODE>	INT	Geometry mode (reserved)			
28	FB	<S_FB>	REAL	Feedrate for grooving the webs Specification as feedrate value or as percentage value related to F, see <_AMODE> (TEN MILLIONS).			

4.24.1.44 CYCLE940 – undercut form E and F / undercut thread

Various undercuts can be programmed using the CYCLE940 cycle. In some cases, these differ significantly regarding the parameterization.

The additional columns in the table indicate which parameters are required for which undercut type. They correspond to the vertical selection softkeys in the cycle screen form:

- E: Undercut form E
- F: Undercut form F
- A-D: DIN thread undercut (forms A-D)
- T: Thread undercut (free definition of form)

Syntax

```
CYCLE940(<_SPD>, <_SPL>, <_FORM>, <_LAGE>, <_SDIS>, <_FFP>,
<_VARI>, <_EPD>, <_EPL>, <_R1>, <_R2>, <_STA>, <_VRT>, <_MID>,
<_FAL>, <_FALX>, <_FALZ>, <_PITI>, <_PTAB>, <_PTABA>, <_DMODE>,
<_AMODE>)
```

Parameters

<table>
<tr><th>No.</th><th>Param-eter mask</th><th>Parameter internal</th><th>Data type</th><th colspan="4">Prog. for form</th><th colspan="3">Meaning</th></tr>
<tr><td></td><td></td><td></td><td></td><td>E</td><td>F</td><td>A-D</td><td>T</td><td colspan="3"></td></tr>
<tr><td>1</td><td>X0</td><td><_SPD></td><td>REAL</td><td>x</td><td>x</td><td>x</td><td>x</td><td colspan="3">Reference point in the plane axis (always diameter)</td></tr>
<tr><td>2</td><td>Z0</td><td><_SPL></td><td>REAL</td><td>x</td><td>x</td><td>x</td><td>x</td><td colspan="3">Reference point on longitudinal axis (abs)</td></tr>
<tr><td rowspan="8">3</td><td rowspan="8">FORM</td><td rowspan="8"><_FORM></td><td rowspan="8">CHAR</td><td>x</td><td>x</td><td>x</td><td>x</td><td colspan="3">Form of undercut (capital letters, e.g. "T")
Selection, table from which the undercut values should be taken</td></tr>
<tr><td colspan="4" rowspan="7"></td><td rowspan="7"></td><td>A =</td><td>External, reference DIN76,
A = normal</td></tr>
<tr><td>B =</td><td>External, reference DIN76,
B = short</td></tr>
<tr><td>C =</td><td>Internal, reference DIN76,
C = normal</td></tr>
<tr><td>D =</td><td>Internal, reference DIN76,
D = short</td></tr>
<tr><td>E =</td><td>Reference DIN509</td></tr>
<tr><td>F =</td><td>Reference DIN509</td></tr>
<tr><td>T=</td><td>Free-form</td></tr>
<tr><td rowspan="4">4</td><td rowspan="4">POSI-TION</td><td rowspan="4"><_LAGE></td><td rowspan="4">INT</td><td>x</td><td>x</td><td>x</td><td>x</td><td rowspan="4">Position of under-cut (parallel Z)</td><td>0 =</td><td>External +Z: ____|</td></tr>
<tr><td colspan="4" rowspan="3"></td><td>1 =</td><td>External -Z: |____/</td></tr>
<tr><td>2 =</td><td>Internal +Z: /-----|</td></tr>
<tr><td>3 =</td><td>Internal -Z: |-----\</td></tr>
<tr><td>5</td><td>SC</td><td><_SDIS></td><td></td><td>x</td><td>x</td><td>x</td><td>x</td><td colspan="3">Safety clearance (inc)</td></tr>
<tr><td>6</td><td>F</td><td><_FFP></td><td></td><td>x</td><td>x</td><td>x</td><td>x</td><td colspan="3">Machining feedrate (mm/rev)</td></tr>
</table>

No.	Param-eter mask	Parameter internal	Data type	Prog. for form				Meaning
7		<_VARI>	INT	-	-	x	x	Machining type
								UNITS: Machining 1 = Roughing 2 = Finishing 3 = Roughing + finishing
								TENS: Machining strategy 0 = Parallel to the contour 1 = Longitudinal
								Undercut forms E and F are always machined in a single pass like finishing.
8	X1	<_EPD>		x	x	-	-	Allowance X (abs/inc), see <_AMODE>
				-	-	-	x	Undercut depth (abs/inc), see <_AMODE>
9	Z1	<_EPL>		-	x	-	-	Allowance Z
				-	-	-	x	Undercut width (abs/inc), see <_AMODE>
10	R1	<_R1>		-	-	-	x	Rounding radius on slopes
11	R2	<_R2>		-	-	-	x	Rounding radius in the corner
12	α	<_STA>		-	-	x	x	Insertion angle
13	VX	<_VRT>		x	x	-	-	Cross-feed X (abs/inc), see <_AMODE>
				-	-	x	x	Cross-feed X when finishing, (abs/inc), see <_AMODE>
14	D	<_MID>		-	-	x	x	Depth infeed
15	U	<_FAL>		-	-	x	x	Finishing allowance parallel to contour, see <_AMODE>
16	UX	<_FALX>		-	-	x	x	Finishing allowance X
17	UZ	<_FALZ>		-	-	x	x	Finishing allowance Z
18	P	<_PITI>	INT	-	-	x	-	Select pitch, form A-D, corresponds to M1 ... M68
								0 = 0.20 1 = 0.25 2 = 0.30 3 = 0.35 4 = 0.40 5 = 0.45 6 = 0.50 7 = 0.60 8 = 0.70 9 = 0.75 10 = 0.80 11 = 1.00 12 = 1.25 13 = 1.50 14 = 1.75 15 = 2.00 16 = 2.50 17 = 3.00 18 = 3.50 19 = 4.00 20 = 4.50 21 = 5.00 22 = 5.50 23 = 6.00
				x	x	-	-	Select radius/depth, form E, F
								0 = 0.6 x 0.3 1 = 1.0 x 0.4 2 = 1.0 x 0.2 3 = 1.6 x 0.3 4 = 2.5 x 0.4 5 = 4.0 x 0.5 6 = 0.4 x 0.2 7 = 0.6 x 0.2 8 = 0.1 x 0.1 9 = 0.2 x 0.1
19		<_PTAB>	STRING [5]					String for thread table ("", "ISO", "BSW", "BSP", "UNC") (for the surface only)
20		<_PTABA>	STRING [20]					String for selection from thread table (e.g. "M 10", "M 12", ...) (for the surface only)

No.	Param-eter mask	Parameter internal	Data type	Prog. for form				Meaning		
21		<_DMODE>	INT					Display mode		
				x	x	x	x	UNITS:	Machining plane G17/G18/G19	
									0 =	Compatibility, the plane effective before the cycle call remains active
									1 =	G17 (only active in the cycle)
									2 =	G18 (only active in the cycle)
									3 =	G19 (only active in the cycle)
22		<_AMODE>	INT					Alternative mode		
				x	x	-	x	UNITS:	Parameter <_EPD> allowance X or undercut depth	
									0 =	Absolute (always diameter)
									1 =	Incremental
				x	x	-	x	TENS:	Parameter <_EPL> allowance Z or undercut width	
									0 =	Absolute
									1 =	Incremental
				x	x	x	x	HUNDREDS:	Parameter <_VRT> cross-feed X	
									0 =	Absolute (always diameter)
									1 =	Incremental
				-	-	x	x	THOUSANDS:	Finishing allowance	
									0 =	Finishing allowance parallel to the contour (<_FAL>)
									1 =	Separate machining allowance (<_FALX>/<_FALZ>)

4.24.1.45 CYCLE951 - stock removal

Syntax

```
CYCLE951(<_SPD>, <_SPL>, <_EPD>, <_EPL>, <_ZPD>, <_ZPL>, <_LAGE>,
<_MID>, <_FALX>, <_FALZ>, <_VARI>, <_RF1>, <_RF2>, <_RF3>, <_SDIS>,
<_FF1>, <_NR>, <_DMODE>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	X0	<_SPD>	REAL	Reference point (abs, always diameter)
2	Z0	<_SPL>	REAL	Reference point (abs)
3	X1	<_EPD>	REAL	End point
4	Z1	<_EPL>	REAL	End point

No.	Parameter mask	Parameter internal	Data type	Meaning		
5	XM α1 α2	<_ZPD>	REAL	Intermediate point, see <_DMODE> (TENS)		
6	ZM α1 α2	<_ZPL>	REAL	Intermediate point, see <_DMODE> (TENS)		
7	Position	<_LAGE>	INT	Position of stock removal corner	0 =	External/rear
					1 =	External/front
					2 =	Internal/rear
					3 =	Internal/front
8	D	<_MID>	REAL	Maximum depth infeed on insertion		
9	UX	<_FALX>	REAL	Finishing allowance in X		
10	UZ	<_FALZ>	REAL	Finishing allowance in Z		
11		<_VARI>	INT	Machining type		
				UNITS:	Stock removal direction (longitudinal or transverse) in the coordinate system	
					1 =	Longitudinal
					2 =	Face
				TENS:		
					1 =	Roughing to final machining allowance
					2 =	Finishing
				HUNDREDS:	Reserved	
				THOUSANDS:	Reserved	
				TEN THOUSANDS:	Reserved	
12	R1/FS1	<_RF1>	REAL	Rounding radius or chamfer width 1, see <_AMODE> (TEN THOUSANDS)		
13	R2/FS2	<_RF2>	REAL	Rounding radius or chamfer width 2, see <_AMODE> (HUNDRED THOUSANDS)		
14	R3/FS3	<_RF3>	REAL	Rounding radius or chamfer width 3, see <_AMODE> (ONE MILLION)		
15	SC	<_SDIS>	REAL	Safety clearance		
16	F	<_FF1>	REAL	Feedrate for roughing/finishing		
17		<_NR>	INT	Identification of stock removal type (corresponds to vertical softkey for selecting form):		
					0 =	Stock removal 1, 90 degree corner without chamfers/rounding
					1 =	Stock removal 2, 90 degree corner with chamfers/rounding
					2 =	Stock removal 3, any corner with chamfers/rounding

No.	Parameter mask	Parameter internal	Data type	Meaning		
18		<_DMODE>	INT	Display mode		
				UNITS:	Machining plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Form of input <_ZPD>/<_ZPL>	
					0 =	Xm/Zm
					1 =	Xm/α1
					2 =	Xm/α2
					3 =	α1/Zm
					4 =	α2/Zm
					5 =	α1/α2
21		<_AMODE>	INT	Alternative mode		
				UNITS:	Intermediate point in X	
					0 =	Absolute, value of transverse axis in the diameter
					1 =	Incremental, value of transverse axis in the radius
				TENS:	Intermediate point in Z	
					0 =	Absolute
					1 =	Incremental
				HUNDREDS:	End point in X	
					0 =	Absolute, value of transverse axis in the diameter
					1 =	Incremental, value of transverse axis in the radius
				THOUSANDS:	End point in Z.	
					0 =	Absolute
					1 =	Incremental
				TEN THOUSANDS:	Radius/chamfer 1	
					0 =	Radius
					1 =	Chamfer
				HUNDRED THOUSANDS:	Radius/chamfer 2	
					0 =	Radius
					1 =	Chamfer
				ONE MILLION:	Radius/chamfer 3	
					0 =	Radius
					1 =	Chamfer

4.24.1.46 CYCLE952 – stock removal / residual stock removal / plunge cutting / residual plunge cutting / plunge turning / residual plunge turning

Syntax

```
CYCLE952(<_PRG>, <_CON>, <_CONR>, <_VARI>, <_F>, <_FR>, <_RP>,
<_D>, <_DX>, <_DZ>, <_UX>, <_UZ>, <_U>, <_U1>, <_BL>, <_XD>, <_ZD>,
<_XA>, <_ZA>, <_XB>, <_ZB>, <_XDA>, <_XDB>, <_N>, <_DP>, <_DI>,
<_SC>, <_DN>, <_GMODE>, <_DMODE>, <_AMODE>, <_PK>, <_DCH>, <_FS>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	PRG	`<_PRG>`	STRING[100]	Name of the stock removal program
2	CON	`<_CON>`	STRING[100]	Name of the program from which the updated contour of the blank is read (for residual machining)
3	CONR	`<_CONR>`	STRING[100]	Name of the program into which the updated contour for the blank (see `<_AMODE>` TEN THOUSANDS) will be written

No.	Parameter mask	Parameter internal	Data type	Meaning		
4		<_VARI>	INT	Machining type		
				UNITS:	Type of stock removal	
					1 =	Longitudinal
					2 =	Face
					3 =	Parallel to the contour
				TENS:	Machining process (see <_GMODE> HUNDREDS)	
					1 =	Roughing
					2 =	Finishing
					3 =	Complete machining
					4 =	Roughing, two-channel
					5 =	Finishing, two-channel
					6 =	Roughing PrimeTurning™ (PrimeTurning™ option required!)
					7 =	Finishing PrimeTurning™ (PrimeTurning™ option required!)
				HUNDREDS:	Machining direction	
					1 =	Machining direction X -
					2 =	Machining direction X +
					3 =	Machining direction Z -
					4 =	Machining direction Z +
				THOUSANDS:	Infeed direction	
					1 =	External X -
					2 =	Internal X +
					3 =	Front face Z -
					4 =	Rear face Z +
				TEN THOUSANDS:	Define effect of finishing allowances	
					0 =	Separate UX and UZ finishing allowances
					1 =	Finishing allowance U parallel to the contour
				HUNDRED THOUSANDS:	Rounding	
					0 =	Compatibility, automatic rounding
					1 =	With rounding at the contour
					2 =	Without rounding
					3 =	Automatic rounding
				ONE MILLION:	Relief cuts	
					0 =	Position is not evaluated during grooving, - residual and groove turning, - remainder
					1 =	Machine relief cuts
					2 =	No machining of relief cuts
				TEN MILLIONS:	Behind/in front of turning center	

No.	Parameter mask	Parameter internal	Data type	Meaning		
					0 =	Machining in front of the turning center
					1 =	Reserved
5	F	<_F>	REAL	Feedrate for roughing/finishing		
	FZ			Feedrate abscissa groove turning		
6	FR	<_FR>	REAL	Feedrate for insertion into relief cuts, roughing		
	FX			Feedrate ordinate groove turning		
7	RP	<_RP>	REAL	Retraction plane for internal machining (abs., always diameter)		
8	D	<_D>	REAL	Roughing infeed (see <_AMODE> UNITS)		
9	DX	<_DX>	REAL	X infeed (see <_AMODE> UNITS)		
10	DZ	<_DZ>	REAL	Z infeed (see <_AMODE> UNITS)		
11	UX	<_UX>	REAL	Finishing allowance X, (see <_VARI> TEN THOUSANDS)		
12	UZ	<_UZ>	REAL	Finishing allowance Z, (see <_VARI> TEN THOUSANDS)		
13	U	<_U>	REAL	Finishing allowance parallel to contour, (see <_VARI> TEN THOUSANDS)		
14	U1	<_U1>	REAL	Additional finishing allowance while finishing (see <_AMODE> THOUSANDS)		
15	BL	<_BL>	INT	Definition of blank	1 =	Cylinder with allowance
					2 =	Allowance at finished-part contour
					3 =	Contour of blank is specified
16	XD	<_XD>	REAL	Definition of blank X (see <_AMODE> HUNDRED THOUSANDS)		
17	ZD	<_ZD>	REAL	Definition of blank Z (see <_AMODE> ONE MILLION)		
18	XA	<_XA>	REAL	Limit 1 X (abs., always diameter)		
19	ZA	<_ZA>	REAL	Limit 1 Z (abs.)		
20	XB	<_XB>	REAL	Limit 2 X (see <_AMODE> TEN MILLIONS)		
21	ZB	<_ZB>	REAL	Limit 2 Z (see <_AMODE> HUNDRED MILLIONS)		
22	XDA	<_XDA>	REAL	Grooving limit 1 for the 1st groove position on the end face (abs., always diameter)		
23	XDB	<_XDB>	REAL	Grooving limit 2 for the 1st groove position on the end face (abs., always diameter)		
24	N	<_N>	INT	Number of grooves		
25	DP	<_DP>	REAL	Distance between grooves	Longitudinal groove: Parallel to Z axis	
					Transverse groove: Parallel to X axis	
26	DI	<_DI>	REAL	Distance for interruption of infeed	0 =	No interruption
					> 0 =	With interruption
27	SC	<_SC>	REAL	Safety clearance for avoiding obstacles, incremental		
28	D2	<_DN>	INT	D number for 2nd cutting edge if not programmed ⇒ D+1		

No.	Parameter mask	Parameter internal	Data type	Meaning		
29		<_GMODE>	INT	Geometrical mode (evaluation of programmed geometrical data)		
				UNITS:	Re-serve d	
				TENS:	Re-serve d	
				HUNDREDS:	Select machining/only calculation of start point	
					0 =	Normal machining (no compati-bility mode needed)
					1 =	Normal machining
					2 =	Calculate starting position - no machining (only for call from ShopMill/ShopTurn)
				THOUSANDS:	Limit	
					0 =	No
					1 =	Yes
				TEN THOUSANDS:	Enter limit 1 X	
					0 =	No
					1 =	Yes
				HUNDRED THOUSANDS:	Enter limit 2 X	
					0 =	No
					1 =	Yes
				ONE MILLION:	Enter limit 1 Z	
					0 =	No
					1 =	Yes
				TEN MILLIONS:	Enter limit 2 Z	
					0 =	No
					1 =	Yes

No.	Parameter mask	Parameter internal	Data type	Meaning		
30		`<_DMODE>`	INT	Display mode		
				UNITS:	Working plane G17/G18/G19	
					0 =	Compatibility, the plane effective before the cycle call remains active
					1 =	G17 (only active in the cycle)
					2 =	G18 (only active in the cycle)
					3 =	G19 (only active in the cycle)
				TENS:	Technology mode	
					1 =	Stock removal along the contour
					2 =	Contour grooving
					3 =	Groove turning
				HUNDREDS:	Machine residual material	
					0 =	No
					1 =	Yes
				THOUSANDS:	---	Reserved
				TEN THOUSANDS:	Technology scaling in cycle screen forms (Page 1027)	
					0 =	Input: Complete
					1 =	Input: normal
				HUNDRED THOUSANDS:	Automatic program name	
					0 =	No
					1 =	Yes

No.	Parameter mask	Parameter internal	Data type	Meaning		
31		<_AMODE>	INT	Alternative mode		
				UNITS:	Select infeed	
					0 =	DX and DZ infeed for stock removal parallel to contour
					1 =	D infeed
				TENS:	Infeed strategy	
					0 =	Variable cutting depth (90 ... 100%)
					1 =	Constant cutting depth
				HUNDREDS:	Cut segmentation	
					0 =	Uniform
					1 =	Align to edges
				THOUSANDS:	Select contour allowance U1, double finishing	
					0 =	No
					1 =	Yes
				TEN THOUSANDS:	Update selection of blank	
					0 =	No
					1 =	Yes
				HUNDRED THOUSANDS:	Select allowance on blank XD	
					0 =	Absolute, value of transverse axis in the diameter
					1 =	Incremental, value of transverse axis in the radius
				ONE MILLION:	Select allowance on blank ZD	
					0 =	Absolute
					1 =	Incremental
				TEN MILLIONS:	Select limit 2 XB	
					0 =	Absolute, value of transverse axis in the diameter
					1 =	Incremental, value of transverse axis in the radius
				HUNDRED MILLION:	Select limit 2 ZB	
					0 =	Absolute
					1 =	Incremental
				ONE BILLION:		
					0 =	Leading channel
					1 =	Following channel
32		<_PK>	INT	Number of the partner channel if there are more than 2 channels available at the machine		
33	DCH	<_DCH>	REAL	Channel offset		
34	FS	<_FS>	REAL	Finishing feedrate during complete machining		

4.24.1.47 CYCLE953 - Surface turning

Note

To use CYCLE953, a license is required for the following option:

- Surface turning (article number: 6FC5800-0AR51-0YB0)

Syntax

```
CYCLE953(<_PRG>, <S_SRC>, <_LAB1>, <_LAB2>, <S_AX1>, <S_AX2>,
<S_AX3>, <S_ROT_AX>, <_SV1>, <_SV2>, <S_TOL>, <_AMODE>)
```

Parameters

No.	Parameter mask	Parameter internal	Data type	Meaning
1	PRG	`<_PRG>`	STRING[160]	Path and file name of the optimized program
2	SRC	`<S_SRC>`	STRING[160]	Path and file name of the source program
3	LAB1	`<_LAB1>`	STRING[31]	Jump marker 1: Start of the optimization range (only with <_AMODE> TENS DIGIT = 1) Default value: "Cutting"
4	LAB2	`<_LAB2>`	STRING[31]	Jump marker 2: End of the optimization range (only with <_AMODE> TENS DIGIT = 1) Default value: "Retract Move"
5	AX1	`<S_AX1>`	STRING[16]	Identifier 1st Geometry axis in source program
6	AX2	`<S_AX2>`	STRING[16]	Identifier 2nd Geometry axis and polar axis in the source program
7	AX3	`<S_AX3>`	STRING[16]	Identifier 3rd Geometry axis in source program (infeed axis)
8	ROT	`<S_ROT_AX>`	STRING[16]	Channel axis name of the rotary axis for the blank chucking
9	V	`<S_SV1>`	REAL	Constant cutting speed
10	PS	`<S_SV2>`	REAL	Maximum speed at constant cutting speed
11	TOL	`<S_TOL>`	REAL	Maximum path tolerance
12		`<_AMODE>`	INT	Machining mode UNITS: Type of position specification in the source program 0 = Cartesian 1 = Polar TENS: Scaling range 0 = Complete program 1 = Program section

4.24.1.48 CYCLE4071 - longitudinal grinding with infeed at the reversal point

Syntax

```
CYCLE4071(<S_A>, <S_B>, <S_W>, <S_U>, <S_I>, <S_K>, <S_H>, <S_A1>,
<S_A2>)
```

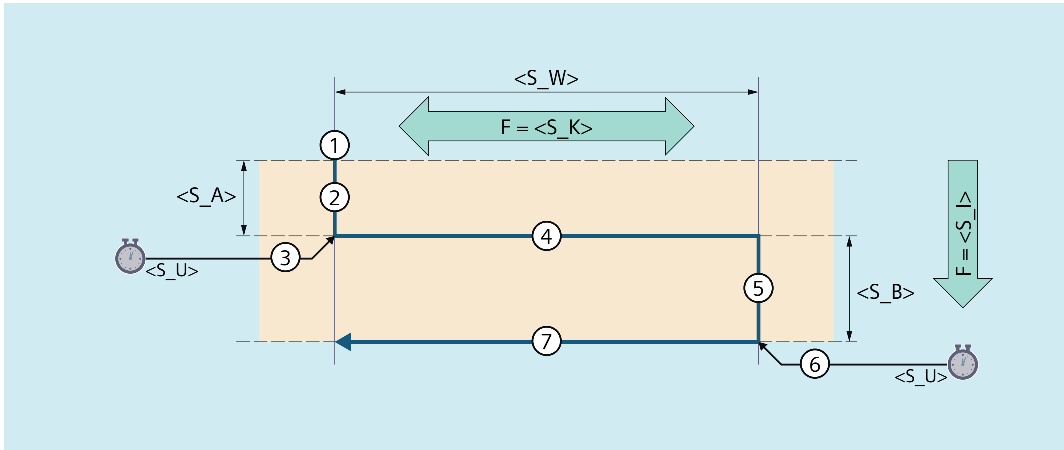
Parameters

No.	Parameter	Data type	Meaning
1	`<S_A>`	REAL	Infeed depth at the start
2	`<S_B>`	REAL	Infeed depth at the end
3	`<S_W>`	REAL	Grinding width
4	`<S_U>`	REAL	Sparking-out time
5	`<S_I>`	REAL	Feedrate for infeed
6	`<S_K>`	REAL	Feedrate for transverse infeed
7	`<S_H>`	INT	Number of repetitions
8	`<S_A1>`	AXIS	Infeed axis (optional) or 1st geometry axis
9	`<S_A2>`	AXIS	Oscillating axis (optional) or 2nd geometry axis

Function

The cycle is used for the execution of repeating infeeds. The infeed depth at the start and at the end can be different. There is a tangential motion between the infeeds.

Sequence



①	Start of the cycle at the current position of the oscillating axis.
②	Traversing of the infeed axis to the infeed depth at the start `<S_A>` with the feedrate for infeed `<S_I>`.
③	Sparking out with the sparking-out time `<S_U>`.
④	Traversing of the oscillating axis with the grinding width `<S_W>` as travel path and the feedrate for transverse infeed `<S_K>`.
⑤	Traversing of the infeed axis to the infeed depth at the end `<S_B>` with the feedrate for infeed `<S_I>`.
⑥	Sparking out with the sparking-out time `<S_U>`.
⑦	Traversing of the oscillating axis with the grinding width `<S_W>` as travel path to the starting point and the feedrate for transverse infeed `<S_K>`.
	Indicates reiterating sequential steps. The sequence is repeated until the programmed number of repetitions `<S_H>` has been reached.

Note

The sequence cannot be interrupted with a single block.

Example

Executing two oscillating motions with the following cycle parameters:

- Infeed depth at the start: 0.02 mm
- Infeed depth at the end: 0.01 mm
- Stroke: 100 mm
- Sparking-out time: 1 s
- Infeed feedrate: 1 mm/min
- Transverse feedrate: 1000 mm/min

- Repetitions: 2
- Oscillating and infeed axes: Standard geometry axes

Program code

```
N10 T1 D1
N20 CYCLE4071(0.02,0.01,100,1,1,1000,2)
N30 M30
```

4.24.1.49 CYCLE4072 - longitudinal grinding with infeed at the reversal point and cancel signal

Syntax

```
CYCLE4072(<S_GAUGE>, <S_A>, <S_B>, <S_W>, <S_U>, <S_I>, <S_K>,
<S_H>, <S_A1>, <S_A2>)
```

Parameters

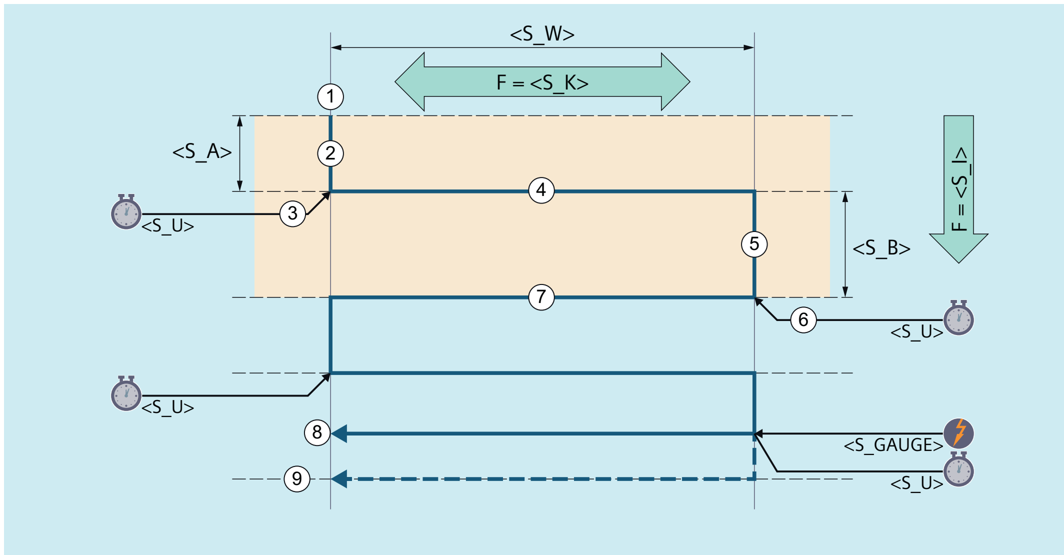
No.	Parameter	Data type	Meaning
1	<S_GAUGE>	STRING	Cancel conditions for infeed: 1. Number of a rapid input 2. Logical expression
2	<S_A>	REAL	Infeed depth at the start
3	<S_B>	REAL	Infeed depth at the end
4	<S_W>	REAL	Grinding width
5	<S_U>	REAL	Sparking-out time
6	<S_I>	REAL	Feedrate for infeed
7	<S_K>	REAL	Feedrate for transverse infeed
8	<S_H>	INT	Number of repetitions
9	<S_A1>	AXIS	Infeed axis (optional) or 1st geometry axis
10	<S_A2>	AXIS	Oscillating axis (optional) or 2nd geometry axis

Function

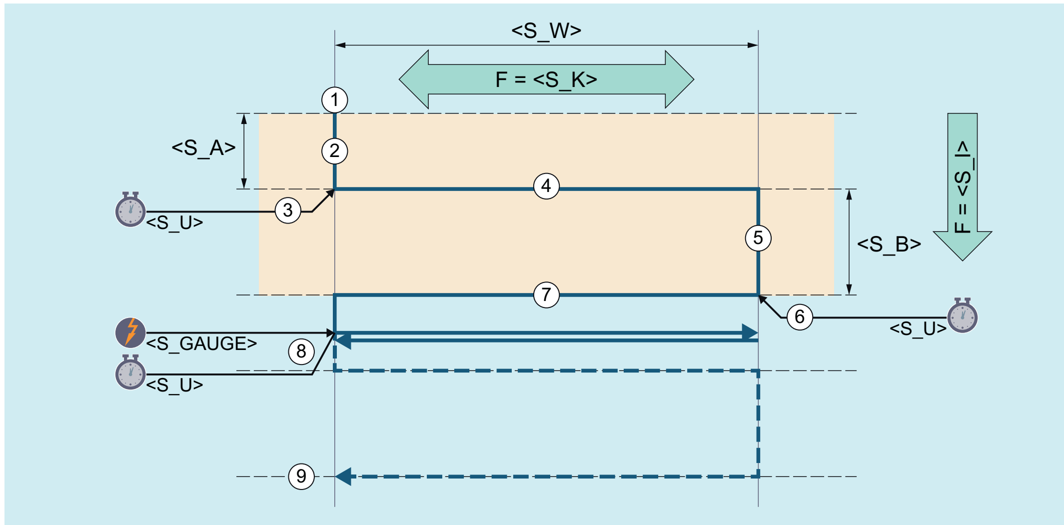
The cycle is used for the execution of repeating infeeds taking into account an external cancel signal. The infeed depth can be different at the start and at the end. There is a tangential motion between the infeeds. The depth infeed is cancelled when the cancel condition is satisfied. A complete stroke is always performed after the cancellation of the depth infeed.

Sequence

Cancellation of the infeed at the end



Cancellation of the infeed at the start



① Start of the cycle at the current position of the oscillating axis.

② Traversing of the infeed axis to the infeed depth at the start <S_A> with the feedrate for infeed <S_I>.

③ Sparking out with the sparking-out time <S_U>.

④ Traversing of the oscillating axis with the grinding width <S_W> as travel path and the feedrate for transverse infeed <S_K>.

⑤ Traversing of the infeed axis to the infeed depth at the end <S_B> with the feedrate for infeed <S_I>.

⑥ Sparking out with the sparking-out time <S_U>.

⑦ Traversing of the oscillating axis with the grinding width <S_W> as travel path to the starting point and the feedrate for transverse infeed <S_K>.

⑧ Cancel signal: The machining stops when the next start point is reached.

⑨ Without Cancel signal: The sequence is repeated until the programmed number of repetitions <S_H> has been reached.

Indicates reiterating sequential steps.

Note

The sequence cannot be interrupted with a single block.

Resources

As resources, the cycle uses a block-wide synchronized action and a synchronized action variable. The synchronized action is determined dynamically from the free area of the synchronized action range (CUS.DIR - 1 ..., CMA.DIR - 1000 ..., CST.DIR – 1199 ...). SYG_IS[1] is used as the synchronized action variable.

Examples

Example 1: Oscillation with two strokes:

Cycle parameters

- Infeed depth at the start: 0.02 mm
- Infeed depth at the end: 0.01 mm
- Stroke: 100 mm
- Sparking-out time: 1 s
- Infeed feedrate: 1 mm/min
- Transverse feedrate: 1000 mm/min
- Repetitions: 2
- Oscillating and infeed axes: Standard geometry axes

Cancel signal: Rapid input 1 ($A_IN[1])

Program code

```
N10 T1 D1
N20 CYCLE4072("1",0.02,0.01,100,1,1,1000,2)
N30 M30
```

Example 2: Oscillation with two strokes:

Cycle parameters

- Infeed depth at the start: 0.02 mm
- Infeed depth at the end: 0.01 mm
- Stroke: 100 mm
- Sparking-out time: 1 s
- Infeed feedrate: 1 mm/min
- Transverse feedrate: 1000 mm/min
- Repetitions: 2
- Oscillating and infeed axes: Standard geometry axes

Cancel signal: Variable $A_DBR[20] < 0.01

Program code

```
N10 T1 D1
N20 CYCLE4072("($A_DBR[20]<0.01)",0.02,0.01,100,1,1,1000,2)
N30 M30
```

4.24.1.50 CYCLE4073 - longitudinal grinding with continuous infeed

Syntax

```
CYCLE4073(<S_A>, <S_B>, <S_W>, <S_U>, <S_K>, <S_H>, <S_A1>, <S_A2>)
```

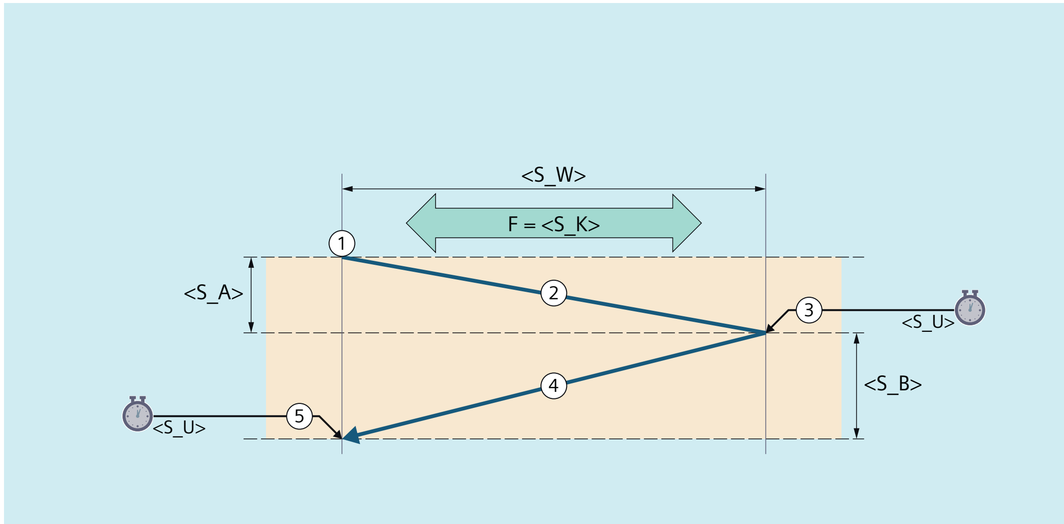
Parameters

No.	Parameter	Data type	Meaning
1	`<S_A>`	REAL	Infeed depth at the start
2	`<S_B>`	REAL	Infeed depth at the end
3	`<S_W>`	REAL	Grinding width
4	`<S_U>`	REAL	Sparking-out time
5	`<S_K>`	REAL	Feedrate for transverse infeed
6	`<S_H>`	INT	Number of repetitions
7	`<S_A1>`	AXIS	Infeed axis (optional) or 1st geometry axis
8	`<S_A2>`	AXIS	Oscillating axis (optional) or 2nd geometry axis

Function

The cycle is used for the execution of repeating infeeds. The infeed from the start to the end and from the end to the start can be different.

Sequence



①	Start of the cycle at the current position of the oscillating axis with infeed depth 0.
②	Traversing of the oscillating axis with the grinding width `<S_W>` as travel path and feedrate for transverse infeed `<S_K>` with continuous increase in the infeed depth up to the infeed depth at the start `<S_A>`.
③	Sparking out with the sparking-out time `<S_U>`.
④	Traversing of the oscillating axis with the grinding width `<S_W>` as travel path to the starting point and feedrate for transverse infeed `<S_K>` with continuous increase in the infeed depth up to the infeed depth at the end `<S_B>`.
⑤	Sparking out with the sparking-out time `<S_U>`.
	Indicates reiterating sequential steps. The sequence is repeated until the programmed number of repetitions `<S_H>` has been reached.

Note

The sequence cannot be interrupted with a single block.

Example

Oscillation with two strokes:

Cycle parameters

- Infeed depth at the start: 0.02 mm
- Infeed depth at the end: 0.01 mm
- Stroke: 100 mm
- Sparking-out time: 1 s
- Transverse feedrate: 1000 mm/min

- Repetitions: 2
- Oscillating and infeed axes: Standard geometry axes

Program code

```
N10 T1 D1
N20 CYCLE4073(0.02,0.01,100,1,1000,2)
N30 M30
```

4.24.1.51 CYCLE4074 - longitudinal grinding with continuous infeed and cancel signal

Syntax

```
CYCLE4074(<S_GAUGE>, <S_A>, <S_B>, <S_W>, <S_U>, <S_K>, <S_H>,
<S_A1>, <S_A2>)
```

Parameters

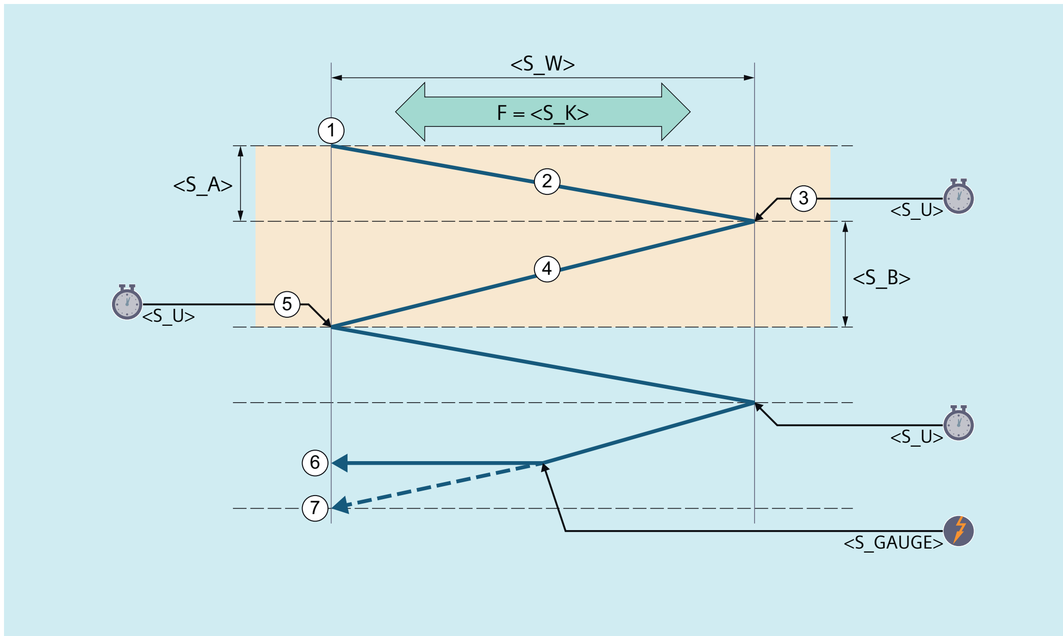
No.	Parameter	Data type	Meaning
1	<S_GAUGE>	STRING	Cancel conditions for infeed: 1. Number of a rapid input 2. Logical expression
2	<S_A>	REAL	Infeed depth at the start
3	<S_B>	REAL	Infeed depth at the end
4	<S_W>	REAL	Grinding width
5	<S_U>	REAL	Sparking-out time
6	<S_K>	REAL	Feedrate for transverse infeed
7	<S_H>	INT	Number of repetitions
8	<S_A1>	AXIS	Infeed axis (optional) or 1st geometry axis
9	<S_A2>	AXIS	Oscillating axis (optional) or 2nd geometry axis

Function

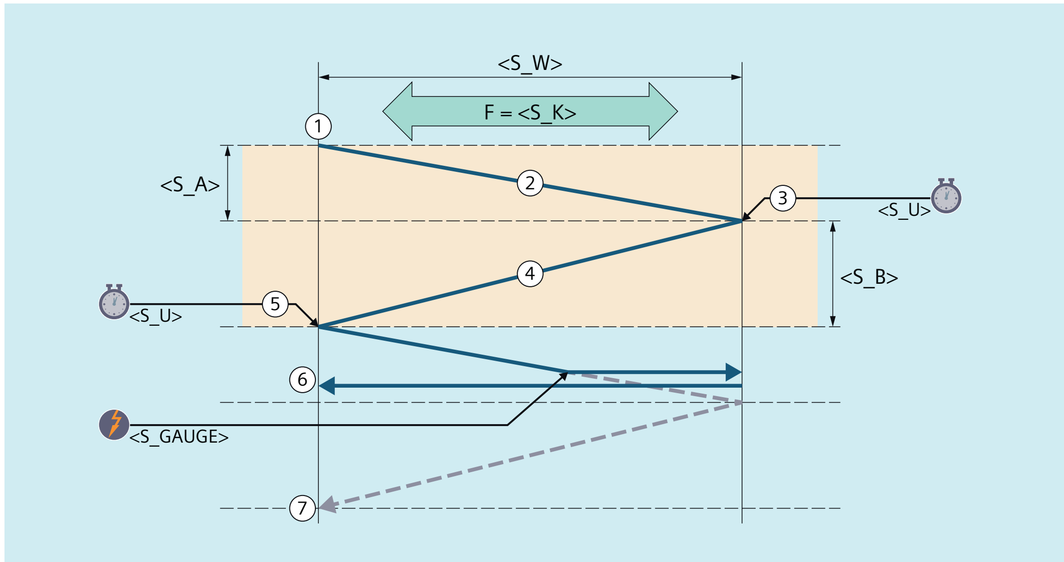
The cycle is used for the execution of repeating infeeds taking into account e.g. an external cancel signal. The infeed depth can be different at the start and at the end. The depth infeed is cancelled when the cancel condition is satisfied. A complete stroke is always performed after the cancellation of the depth infeed.

Sequence

Cancellation of the infeed from the end to the start



Cancellation of the infeed from the start to the end



①	Start of the cycle at the current position of the oscillating axis with infeed depth 0.
②	Traversing of the oscillating axis with the grinding width `<S_W>` as travel path and feedrate for transverse infeed `<S_K>` with continuous increase in the infeed depth up to the infeed depth at the start `<S_A>`.
③	Sparking out with the sparking-out time `<S_U>`.
④	Traversing of the oscillating axis with the grinding width `<S_W>` as travel path to the starting point and feedrate for transverse infeed `<S_K>` with continuous increase in the infeed depth up to the infeed depth at the end `<S_B>`.
⑤	Sparking out with the sparking-out time `<S_U>`.
⑥	Cancel signal: The depth infeed is canceled. The machining stops when the next start point is reached.
⑦	Without Cancel signal: The sequence is repeated until the programmed number of repetitions `<S_H>` has been reached.
(shaded box)	Indicates reiterating sequential steps.

Note

The sequence cannot be interrupted with a single block.

Resources

As resources, the cycle uses a block-wide synchronized action and a synchronized action variable. The synchronized action is determined dynamically from the free area of the synchronized action range (CUS.DIR - 1 ..., CMA.DIR - 1000 ..., CST.DIR – 1199 ...). SYG_IS[1] is used as the synchronized action variable.

Examples

Example 1: Oscillation with two strokes:

Cycle parameters

- Infeed depth at the start: 0.02 mm
- Infeed depth at the end: 0.01 mm
- Stroke: 100 mm
- Sparking-out time: 1 s
- Transverse feedrate: 1000 mm/min
- Repetitions: 2
- Oscillating and infeed axes: Standard geometry axes

Cancel signal: Rapid input 1 ($A_IN[1])

Program code

```
N10 T1 D1
N20 CYCLE4074("1",0.02,0.01,100,1,1000,2)
N30 M30
```

Example 2: Oscillation with two strokes:

Cycle parameters

- Infeed depth at the start: 0.02 mm
- Infeed depth at the end: 0.01 mm
- Stroke: 100 mm
- Sparking-out time: 1 s
- Transverse feedrate: 1000 mm/min
- Repetitions: 2
- Oscillating and infeed axes: Standard geometry axes

Cancel signal: Variable $A_DBR[20] < 0.01

Program code

```
N10 T1 D1
N20 CYCLE4074("($A_DBR[20]<0.01)",0.02,0.01,100,1,1000,2)
N30 M30
```

4.24.1.52 CYCLE4075 - surface grinding with infeed at the reversal point

Syntax

```
CYCLE4075(<S_I>, <S_J>, <S_K>, <S_A>, <S_R>, <S_F>, <S_P>, <S_A1>,
<S_A2>)
```

Parameters

No.	Parameter	Data type	Meaning
1	<S_I>	REAL	Infeed depth at the start
2	<S_J>	REAL	Infeed depth at the end
3	<S_K>	REAL	Total infeed depth
4	<S_A>	REAL	Grinding width
5	<S_R>	REAL	Feedrate for infeed
6	<S_F>	REAL	Feedrate for transverse infeed
7	<S_P>	REAL	Sparking-out time
8	<S_A1>	AXIS	Infeed axis (optional)
9	<S_A2>	AXIS	Oscillating axis (optional)

Function

The cycle is used for machining with a total infeed depth in infeed steps. The infeed depths at the start and at the end can be different. There is a tangential motion between the infeeds.

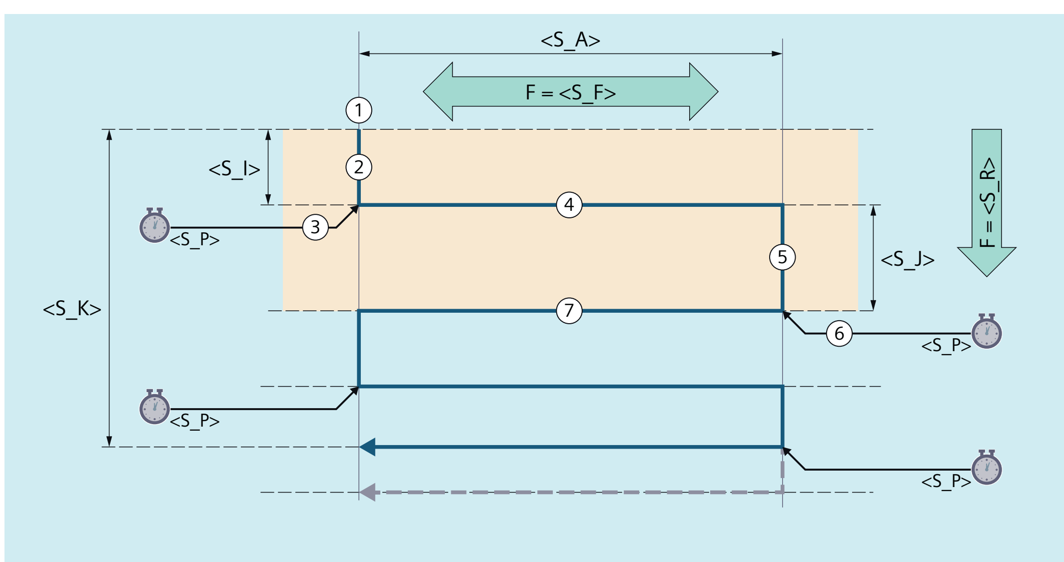
The positional data P1 to P4 can be negative or positive.

The specification of the infeed axis and/or oscillating axis is optional. If one or both parameters are not specified, the cycle uses the first two geometry axes of the channel.

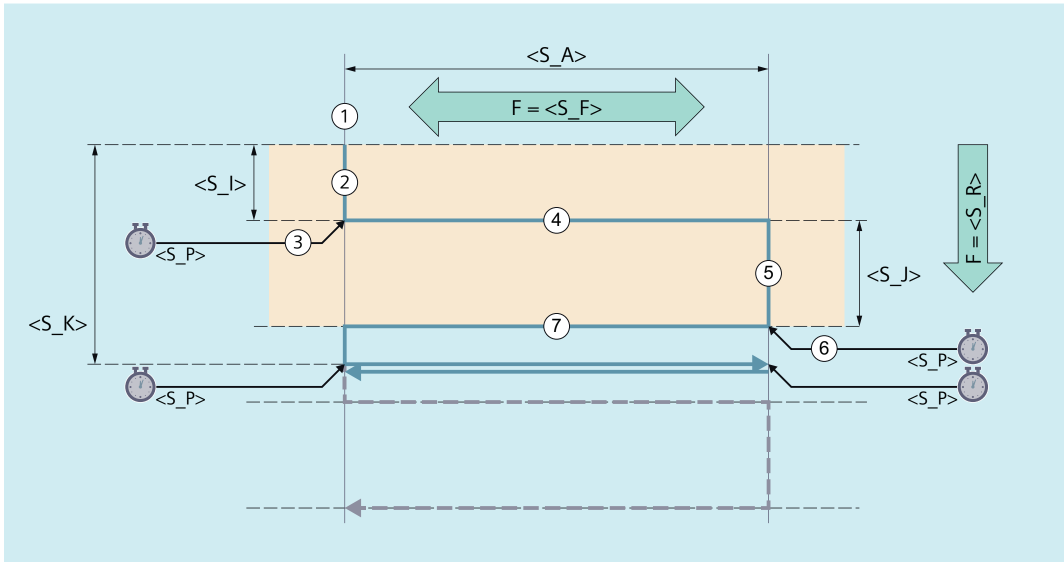
If the sum of the infeed depth at the start and end is 0 or the total infeed depth is 0, only one sparking-out stroke is performed.

Sequence

Total infeed depth reached with infeed at the second reversal point



Total infeed depth reached with infeed at the first reversal point



① Start of the cycle at the current position of the oscillating axis.

② Traversing of the infeed axis to the infeed depth at the start `<S_I>` with the feedrate for infeed `<S_R>`.

③ Sparking out with the sparking-out time `<S_P>`.

④ Traversing of the oscillating axis with the grinding width `<S_A>` as travel path and the feedrate for transverse infeed `<S_F>`.

⑤ Traversing of the infeed axis to the infeed depth at the end `<S_J>` with the feedrate for infeed `<S_R>`.

⑥ Sparking out with the sparking-out time `<S_P>`.

⑦ Traversing of the oscillating axis with the grinding width `<S_A>` as travel path to the starting point and the feedrate for transverse infeed `<S_F>`.

Indicates reiterating sequential steps.
The sequence is repeated until the total infeed depth `<S_K>` has been reached. The last stroke is then distributed unevenly.

Note

The sequence cannot be interrupted with a single block.

Example

Oscillation with:

- 0.02 mm infeed depth at the start
- 0.01 mm infeed depth at the end
- Total infeed depth 1 mm
- 100 mm stroke

- Infeed feedrate 1 mm/min
- Transverse feedrate 1000 mm/min
- 1 second sparking-out time
- Standard geometry axes

Program code
N10 T1 D1
N20 CYCLE4075(0.02,0.01,1,100,1,1000,1)
N30 M30

4.24.1.53 CYCLE4077 - surface grinding with infeed at the reversal point and cancel signal

Syntax

```
CYCLE4077(<S_GAUGE>, <S_I>, <S_J>, <S_K>, <S_A>, <S_R>, <S_F>,
<S_P>, <S_A1>, <S_A2>)
```

Parameters

No.	Parameter	Data type	Meaning
1	<S_GAUGE>	STRING	Cancel condition for infeed: • Number of a rapid input • Logical expression
2	<S_I>	REAL	Infeed depth at the start
3	<S_J>	REAL	Infeed depth at the end
4	<S_K>	REAL	Total infeed depth
5	<S_A>	REAL	Grinding width
6	<S_R>	REAL	Feedrate for infeed
7	<S_F>	REAL	Feedrate for transverse infeed
8	<S_P>	REAL	Sparking-out time
9	<S_A1>	AXIS	Infeed axis (optional)
10	<S_A2>	AXIS	Oscillating axis (optional)

Function

The cycle is used for machining with a total infeed depth in infeed steps. The infeed depths at the start and at the end can be different. There is a tangential motion between the infeeds. The depth infeed is cancelled when the cancel signal of the rapid input is 1 or the cancel condition is satisfied. A complete stroke is performed after the cancellation.

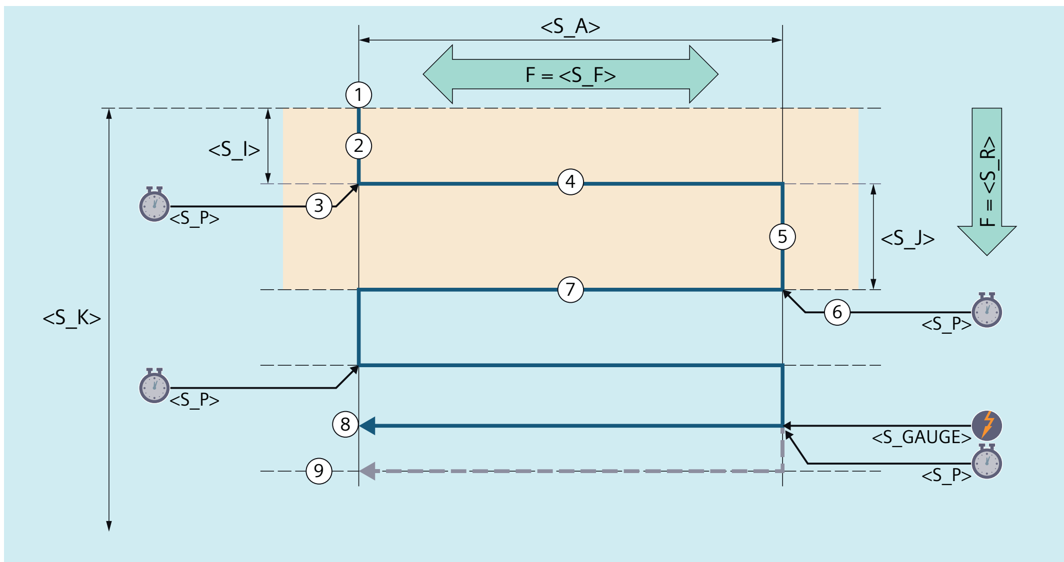
The positional data P2 to P5 can be negative or positive.

The specification of the infeed axis and/or oscillating axis is optional. If one or both parameters are not specified, the cycle uses the first two geometry axes of the channel.

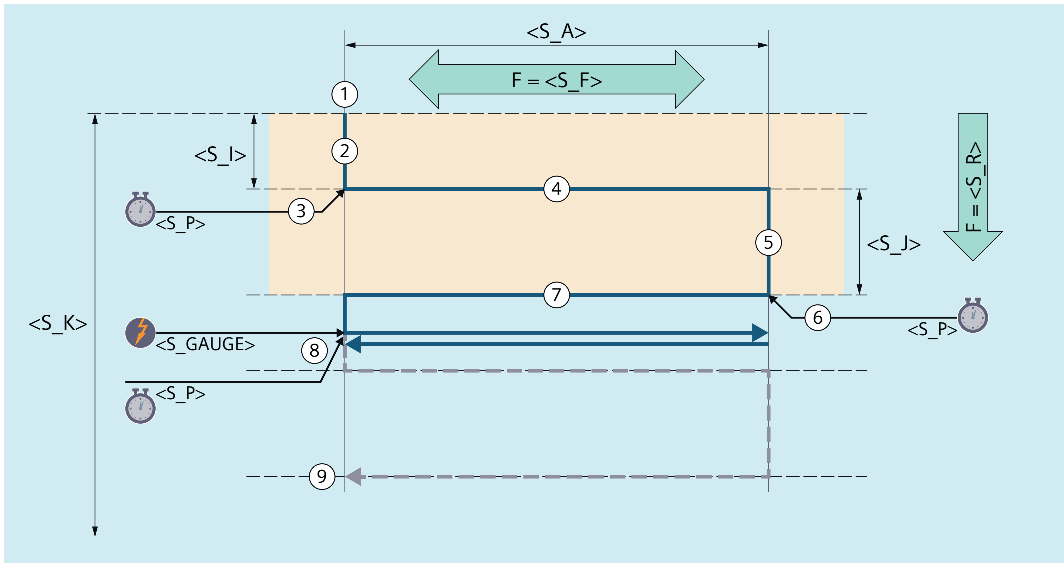
If the sum of the infeed depth at the start and end is 0 or the total infeed depth is 0, only one sparking-out stroke is performed.

Sequence

Cancellation of the infeed at the end



Cancellation of the infeed at the start



① Start of the cycle at the current position of the oscillating axis.

② Traversing of the infeed axis to the infeed depth at the start `<S_I>` with the feedrate for infeed `<S_R>`.

③ Sparking out with the sparking-out time `<S_P>`.

④ Traversing of the oscillating axis with the grinding width `<S_A>` as travel path and the feedrate for transverse infeed `<S_F>`.

⑤ Traversing of the infeed axis to the infeed depth at the end `<S_J>` with the feedrate for infeed `<S_R>`.

⑥ Sparking out with the sparking-out time `<S_P>`.

⑦ Traversing of the oscillating axis with the grinding width `<S_A>` as travel path to the starting point and the feedrate for transverse infeed `<S_F>`.

⑧ Cancel signal: The machining stops when the next start point is reached.

⑨ Without Cancel signal: The sequence is repeated until the total infeed depth `<S_K>` has been reached. The last stroke is then distributed unevenly.

Indicates reiterating sequential steps.

Note

The sequence cannot be interrupted with a single block.

Resources

As resources, the cycle uses a block-wide synchronized action and a synchronized action variable. The synchronized action is determined dynamically from the free area of the synchronized action range (CUS.DIR - 1 ..., CMA.DIR - 1000 ..., CST.DIR – 1199 ...). SYG_IS[1] is used as the synchronized action variable.

Examples

Example 1

Oscillation with:

- 0.02 mm infeed depth at the start
- 0.01 mm infeed depth at the end
- Total infeed depth 1 mm
- 100 mm stroke
- Infeed feedrate 1 mm/min
- Transverse feedrate 1000 mm/min
- 1 second sparking-out time
- Standard geometry axes

Cancel signal: Rapid input 1 ($A_IN[1])

Program code
`N10 T1 D1`
`N20 CYCLE4077("1",0.02,0.01,1,100,1,1000,1)`
`N30 M30`

Example 2

Oscillation with:

- 0.02 mm infeed depth at the start
- 0.01 mm infeed depth at the end
- Total infeed depth 1 mm
- 100 mm stroke
- Infeed feedrate 1 mm/min
- Transverse feedrate 1000 mm/min
- 1 second sparking-out time
- Standard geometry axes

Cancel signal: Dual-port RAM variable 20 less than 0.01 ($A_DBR[20] < 0.01)

Program code
`N10 T1 D1`
`N20 CYCLE4077("($A_DBR[20]<0.01)",0.02,0.01,1,100,1,1000,1)`
`N30 M30`

4.24.1.54 CYCLE4078 - surface grinding with continuous infeed

Syntax

```
CYCLE4078(<S_I>, <S_J>, <S_K>, <S_A>, <S_F>, <S_P>, <S_A1>, <S_A2>)
```

Parameters

No.	Parameter	Data type	Meaning
1	`<S_I>`	REAL	Infeed depth from the start to the end
2	`<S_J>`	REAL	Infeed depth from the end to the start
3	`<S_K>`	REAL	Total infeed depth
4	`<S_A>`	REAL	Grinding width
5	`<S_F>`	REAL	Feedrate
6	`<S_P>`	REAL	Sparking-out time
7	`<S_A1>`	AXIS	Infeed axis (optional)
8	`<S_A2>`	AXIS	Oscillating axis (optional)

Function

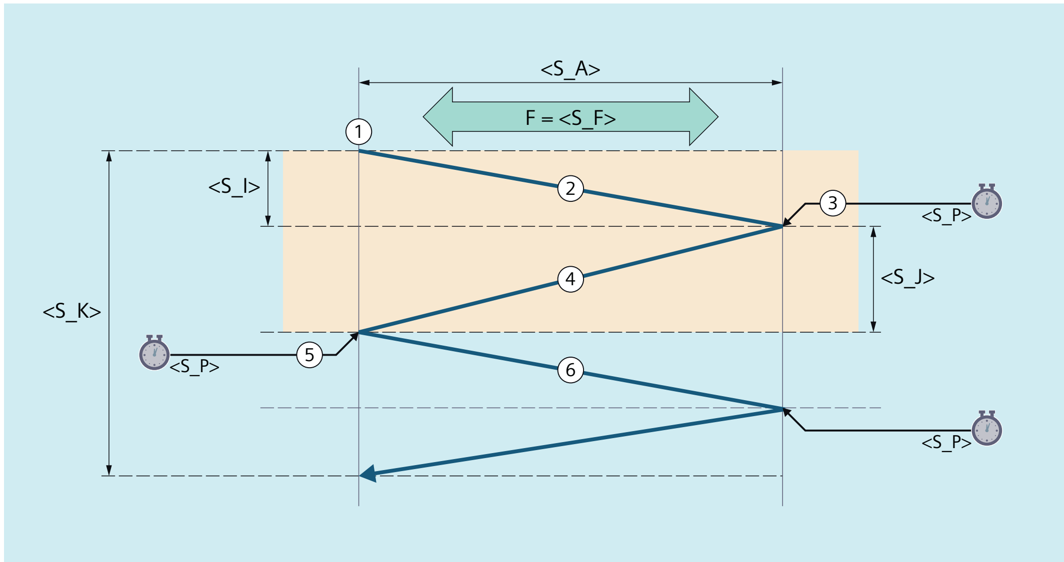
The cycle is used for machining with a total infeed depth by means of continuous infeed. The infeed depths from the start to the end and from the end to the start can be different.

The positional data P1 to P4 can be negative or positive.

The specification of the infeed axis and/or oscillating axis is optional. If one or both parameters are not specified, the cycle uses the first two geometry axes of the channel.

If the sum of the infeed depths P1 and P2 is 0 or the total infeed depth is 0, only one sparking-out stroke is performed.

Sequence



①	Start of the cycle at the current position of the oscillating axis with infeed depth 0.
②	Traversing of the oscillating axis with the grinding width `<S_A>` as travel path and feedrate `<S_F>` with continuous increase in the infeed depth up to the infeed depth at the start <S_I>.
③	Sparking out with the sparking-out time `<S_P>`.
④	Traversing of the oscillating axis with the grinding width `<S_A>` as travel path to the starting point and feedrate `<S_F>` with continuous increase in the infeed depth up to the infeed depth at the end `<S_J>`.
⑤	Sparking out with the sparking-out time `<S_P>`.
⑥	Traversing of the oscillating axis with the grinding width `<S_A>` as travel path to the starting point and feedrate `<S_F>`.
(shaded box)	Indicates reiterating sequential steps. The sequence is repeated until the total infeed depth `<S_K>` has been reached. The last stroke is then distributed unevenly.

Note

The sequence cannot be interrupted with a single block.

Example

Oscillation with:

- 20 mm infeed depth at the start
- 10 mm infeed depth at the end
- Total infeed depth 100 mm
- 100 mm stroke
- Feedrate 1000 mm/min

- 1 second sparking-out time
- Standard geometry axes

Program code
N10 T1 D1
N20 CYCLE4078(20,10,100,100,1000,1)
N30 M30

4.24.1.55 CYCLE4079 - surface grinding with intermittent infeed

Syntax

```
CYCLE4079(<S_I>, <S_J>, <S_K>, <S_A>, <S_R>, <S_F>, <S_P>, <S_A1>,
<S_A2>)
```

Parameters

No.	Parameter	Data type	Meaning
1	<S_I>	REAL	Infeed depth at the start
2	<S_J>	REAL	Infeed depth at the end
3	<S_K>	REAL	Total infeed depth
4	<S_A>	REAL	Grinding width
5	<S_R>	REAL	Feedrate for infeed
6	<S_F>	REAL	Feedrate for transverse infeed
7	<S_P>	REAL	Sparking-out time
8	<S_A1>	AXIS	Infeed axis (optional)
9	<S_A2>	AXIS	Oscillating axis (optional)

Function

The cycle is used for machining with a total infeed depth in infeed steps. The infeed depths at the start and at the end can be different. There is a tangential motion between the infeeds.

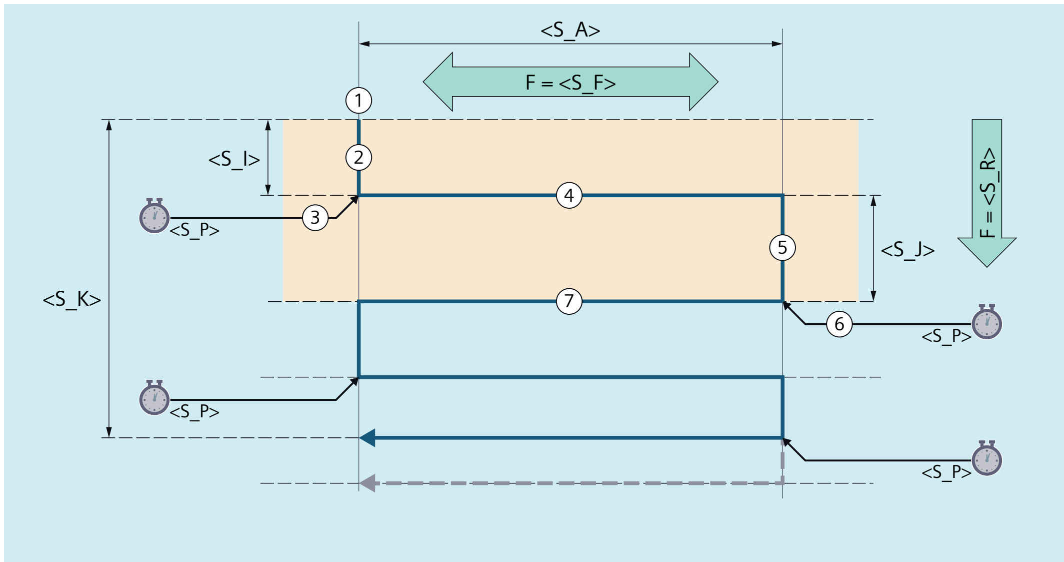
The positional data P1 to P4 can be negative or positive.

The specification of the infeed axis and/or oscillating axis is optional. If one or both parameters are not specified, the cycle uses the first two geometry axes of the channel.

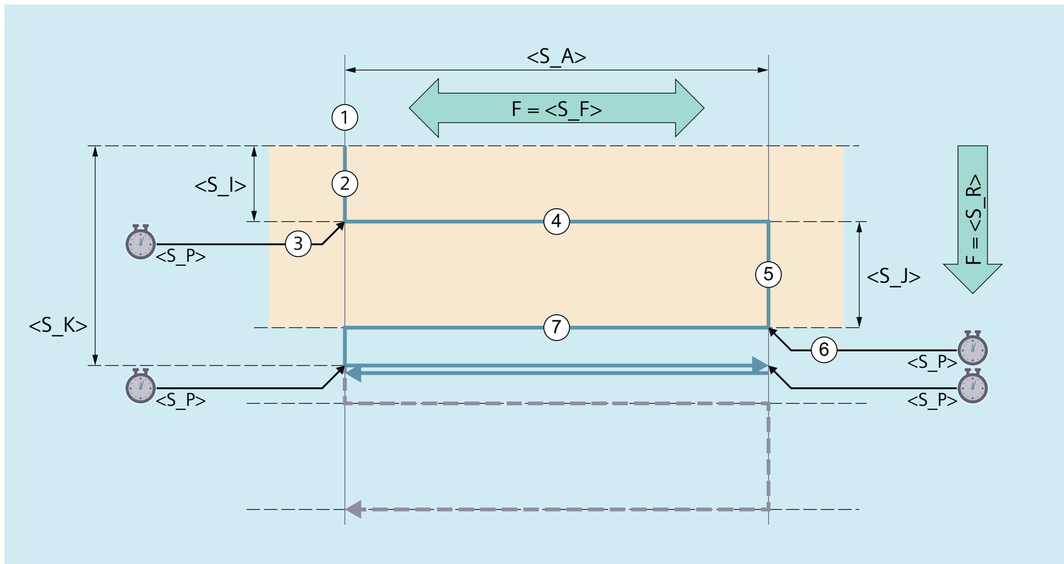
If the sum of the infeed depth at the start and end is 0 or the total infeed depth is 0, only one sparking-out stroke is performed.

Sequence

Total infeed depth reached with infeed at the second reversal point



Total infeed depth reached with infeed at the first reversal point



①	Start of the cycle at the current position of the oscillating axis.
②	Traversing of the infeed axis to the infeed depth at the start `<S_I>` with the feedrate for infeed `<S_R>`.
③	Sparking out with the sparking-out time `<S_P>`.
④	Traversing of the oscillating axis with the grinding width `<S_A>` as travel path and the feedrate for transverse infeed `<S_F>`.
⑤	Traversing of the infeed axis to the infeed depth at the end `<S_J>` with the feedrate for infeed `<S_R>`.
⑥	Sparking out with the sparking-out time `<S_P>`.
⑦	Traversing of the oscillating axis with the grinding width `<S_A>` as travel path to the starting point and the feedrate for transverse infeed `<S_F>`.
▭	Indicates reiterating sequential steps. The sequence is repeated until the total infeed depth `<S_K>` has been reached. The last stroke is then distributed unevenly.

Note

The sequence cannot be interrupted with a single block.

Example

Oscillation with:

- 0.02 mm infeed depth at the start
- 0.01 mm infeed depth at the end
- Total infeed depth 1 mm
- 100 mm stroke

- Infeed feedrate 1 mm/min
- Transverse feedrate 1000 mm/min
- 1 second sparking-out time
- Standard geometry axes

```
Program code
N10 T1 D1
N20 CYCLE4079(0.02,0.01,1,100,1,1000,1)
N30 M30
```

4.24.1.56 GROUP_BEGIN - beginning of program block

Syntax

```
GROUP_BEGIN(<_LEVEL>, <_NAME>, <_SP>, <_MODE>, <S_ICON>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning		
1		<_LEVEL>	INT	Level		
				0 =	Main level	
				1 =	1st sublevel	
2		<_NAME>	STRING[128]	Block name		
3		<_SP>	INT	Spindle		
				0 =	No spindle	
				1 =	Main spindle	
				2 =	Counterspindle	
4		<_MODE>	INT	Mode		
				Bit 0	= 1	GROUP_ADDEND exists
				Bit 1	= 1	ShopTurn: Automatic retraction (traverse to tool change point)
				Bit 12	Reserved	
				Bit 13	Reserved	
5		<S_ICON>	STRING[32]	Name of the icon (only for operator interface)		

4.24.1.57 GROUP_END - end of program block

Syntax

```
GROUP_END(<_LEVEL>, <_SP>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning	
1		`<_LEVEL>`	INT	Level	
				0 =	Main level
				1 =	1st sublevel
2		`<_SP>`	INT	Spindle	
				0 =	No spindle
				1 =	Main spindle
				2 =	Counterspindle

4.24.1.58 GROUP_ADDEND - End of trial cut addition

Syntax

```
GROUP_ADDEND(<_LEVEL>, <_SP>)
```

Parameter

No.	Parameter mask	Parameter internal	Data type	Meaning	
1		`<_LEVEL>`	INT	Level	
				0 =	Main level
				1 =	1st sublevel
2		`<_SP>`	INT	Spindle	
				0 =	No spindle
				1 =	Main spindle
				2 =	Counterspindle

4.24.1.59 Supplementary conditions

Technology scaling in cycle screen forms

When the technology scaling is active, the simplified input can be selected for various cycle screen forms, in which only the most important cycle parameters are displayed

For example, the simplified input can be selected for the following cycle screen forms:

Technology	Cycle screen form
Drilling	Deep-hole drilling
	Tapping
Milling	Rectangular pocket
	Contour milling: Pocket

Technology	Cycle screen form
Turning	Thread turning: Longitudinal
	Contour turning: Stock removal
	Contour turning: Grooving
	Contour turning: Groove turning

In the user interface of the relevant cycle screen forms, the options "Input: **Simple**" and "Input: **Complete**" are available.

Cycle parameters that are not displayed

The cycle parameters that are not displayed in the simplified input are pre-assigned fixed, technologically useful, but not variable values. Or the cycle parameters are assigned parameterizable values via the channel-specific cycle setting data. See the paragraph below "Commissioning" > "Channel-specific cycle setting data"

Switchover from "Input: Complete" > "Input: Simple"

If a cycle screen form is filled in with the setting "Input complete" and then switched to "Input simple", the default or setting data values are used for the parameters no longer displayed when generating the cycle call.

Commissioning

Channel-specific configuration machine data

The technology scaling in cycle screen forms can be activated with the machine data:

MD52210 $MCS_FUNCTION_MASK_DISP, bit 9 = 1 (select display "Input simple")

Channel-specific cycle setting data

If the simplified input in cycle screen forms is active, the values for certain cycle parameters can be specified via the following setting data:

Number	Identifier	Meaning
SD55300	$SCS_EASY_SAFETY_CLEARANCE	Safety clearance
SD55301	$SCS_EASY_DWELL_TIME	Dwell time
SD55305	$SCS_EASY_DRILL_DEEP_FD1	Deep-hole drilling: Percentage: 1st feedrate
SD55306	$SCS_EASY_DRILL_DEEP_DF	Deep-hole drilling: Percentage: Infeed
SD55307	$SCS_EASY_DRILL_DEEP_V1	Deep-hole drilling: Minimum depth infeed
SD55308	$SCS_EASY_DRILL_DEEP_V2	Deep-hole drilling: Retraction distance
SD55309	$SCS_EASY_THREAD_RETURN_DIST	Thread turning: Return distance

4.24.2 Measuring cycles

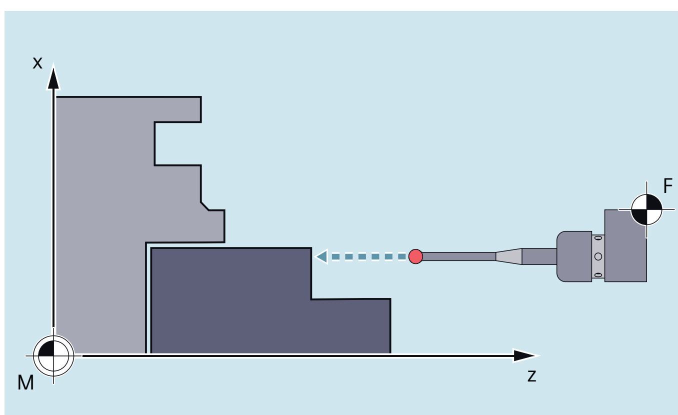
Measuring cycles

A measuring cycle is a predefined NC program in which a specific, generally valid, measuring operation, such as determining the inner diameter of a cylindrical workpiece, is programmed. Parameters are used to adapt to the specific measurement situation; these parameters are transferred to the cycle at the call.

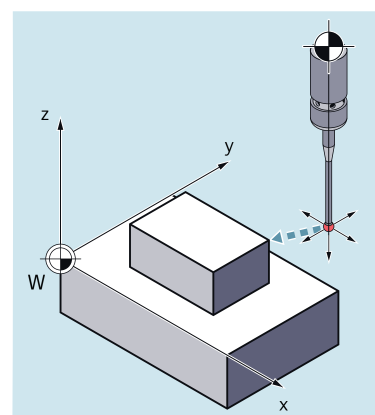
Measuring cycles are available for workpiece and tool measurements for turning and milling technologies.

Workpiece measurement

In workpiece measurement, a probe is moved up to the clamped workpiece in the same way as a tool, and the measured values are acquired. The flexible structure of the measuring cycles makes it possible to perform nearly all measuring tasks required on milling or turning machines.



Example: Measuring a workpiece at the turning machine



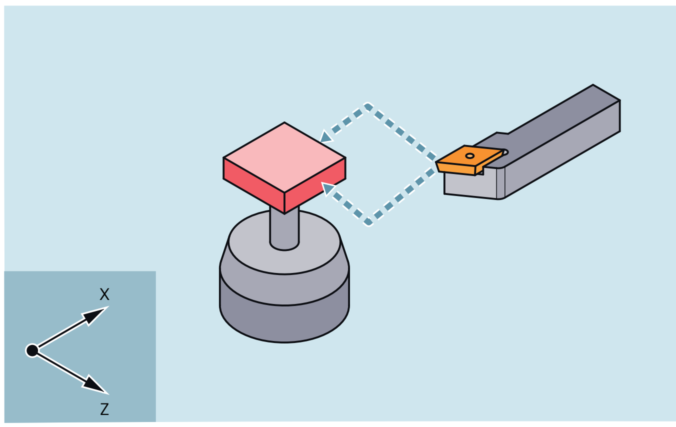
Example: Measuring a workpiece at a milling machine

The result of the workpiece measurement can be optionally used as follows:

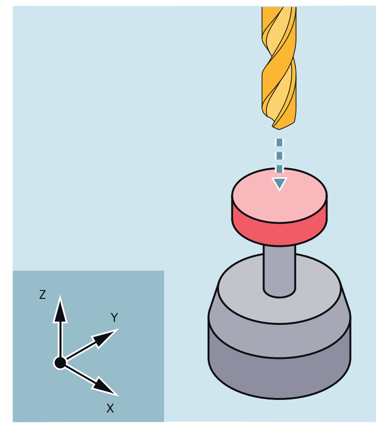
- Compensation in the work offset
- Automatic tool offset
- Measurement without offset

Tool measurement

In tool measurement, the loaded tool is moved up to the probe and the measured values are acquired. The probe is either in a fixed position or is swung into the machining area using an appropriate mechanism.



Example: Measuring a turning tool



Example: Measuring a drill

The tool geometry measured is entered in the appropriate tool offset data set.

Cycle description

This documentation of the measuring cycles only refers to external programming, and is therefore restricted to describing the syntax and parameters

See the Programming Manual Measuring Cycles for a detailed description of the measuring cycles.

Syntax

The program line specified under "syntax" indicates how the cycle call should be programmed.

Special care must be given regarding the following points:

- Correct cycle name
- Call sequence of the transfer parameters

Parameter

All cycle parameters are described with the following data in the table under "Parameters":

- Meaning
- Value range
- Dependency on other parameters

Parameters marked with "reserved" must be programmed with the value 0 or a comma so that the assignment of the following call parameters matches the internal cycle parameters. Exception: string parameters with the value "" or a comma.

4.24.2.1 Overview of the measuring cycles

All predefined measuring cycles for turning and milling are listed and their function briefly described in the following overview table. A detailed description of externally programming the individual measuring cycles is provided in Chapter "Overview of measuring cycle parameters (Page 1032)".

Measuring cycle	Description	Measuring versions
CYCLE973 [2)]	This measuring cycle can be used to calibrate a workpiece probe on a surface on the workpiece or in a groove.	• Calibrate probe - length • Calibrate probe - radius on surface • Calibrate probe - probe in groove
CYCLE974 [2)]	This measuring cycle can be used to determine the workpiece zero in the selected measuring axis or a tool offset with 1-point measurement.	• Turning measurement - front edge • Turning measurement - inside diameter • Turning measurement - outside diameter
CYCLE994 [2)]	This measuring cycle can be used to determine the workpiece zero in the selected measuring axis with 2-point measurement. To do this, two opposite measuring points on the diameter are approached automatically in succession	• Turning measurement - inside diameter • Turning measurement - outside diameter
CYCLE976	Using this measuring cycle, a workpiece probe can be calibrated in a calibration ring or on a calibration ball completely in the working plane or at an edge for a particular axis and direction.	• Calibrate probe - length on surface • Calibrate probe - radius in ring • Calibrate probe - radius on edge • Calibrate probe - calibration on sphere
CYCLE961	This measuring cycle can be used to determine the position of a workpiece corner (inner or outer) and use this as work offset.	• Corner - right-angled corner • Corner - any corner
CYCLE977	This measuring cycle can be used to determine the center point in the plane as well as the width or the diameter.	• Edge distance - groove • Edge distance - rib • Hole - rectangular pocket • Hole - 1 hole • Spigot - rectangular spigot • Spigot - 1 circular spigot
CYCLE978	This measuring cycle can be used to measure the position of an edge in the workpiece coordinate system.	• Edge distance - set edge
CYCLE979	This measuring cycle can be used to measure the center point in the plane and the radius of circle segments.	• Hole - inner circle segment • Spigot - outer circle segment
CYCLE995	With this measuring cycle the angularity of the spindle on a machine tool can be measured.	• 3D - angular deviation spindle
CYCLE996	This measuring cycle can be used to determine transformation-relevant data for kinematic transformations with contained rotary axes.	• 3D - kinematics
CYCLE9960	Transformation-relevant data for kinematic transformations that contain rotary axes can be determined with this measuring cycle.	• 3D - kinematics

Measuring cycle	Description	Measuring versions
CYCLE997	This measuring cycle can be used to determine the center point and diameter of a ball. Furthermore, the center points of three distributed balls can be measured. The plane formed through the three ball center points, regarding its angular position, is determined referred to the working plane in the workpiece coordinate system.	• 3D - sphere • 3D - 3 spheres
CYCLE998	This measuring cycle can be used to determine the angular position of a surface (plane) referred to the working plane and the angle of edges in the workpiece coordinate system.	• Edge distance - align edge • 3D - align plane
CYCLE971 [1)]	This measuring cycle can be used to calibrate a tool probe and measure the tool length and/or tool radius for milling tools.	• Calibrate probe • Measure tool
CYCLE982 [2)]	This measuring cycle can be used to calibrate a tool probe and measure turning, drilling and milling tools on turning machines.	• Calibrate probe • Turning tool • Milling tool • Drill

[1)] Only for milling technology

[2)] Only for turning technology

4.24.2.2 Overview of measuring cycle parameters

CYCLE973 measuring cycle parameters

```
PROC CYCLE973(INT S_MVAR,INT S_PRNUM,INT S_CALNUM,REAL S_SETV,INT S_MA,INT S_MD,REAL
S_FA,REAL S_TSA,REAL S_VMS,INT S_NMSP,INT S_MCBIT,INT _DMODE,INT _AMODE)
```

Table 4-7 CYCLE973 call parameters [1)]

No.	Screen form parameter	Cycle parameter	Meaning	
1		S_MVAR	Measuring variant (default=0012103)	
			Values:	UNITS: Calibration on a surface, edge or in a groove 0 = Length on surface/edge (in the WCS) with known setpoint 1 = Radius on surface (in the WCS) with known setpoint 2 = Length in groove (in the WCS), see S_CALNUM 3 = Radius in groove (in the WCS), see S_CALNUM
				TENS: Reserved 0 = 0
				HUNDREDS: Reserved 0 = 0
				THOUSANDS: Selection of measuring axis and measuring direction for calibration [2)] 0 = No specification (for surface calibration on the groove base, no selection of the measuring axis and measuring direction) [4)] 1 = Specify selection of measuring axis and measuring direction, see S_MA, S_MD (one measuring direction in a measuring axis) 2 = Specify selection of measuring axis, see S_MA (two measuring directions in a measuring axis)
				TEN THOUSANDS: Determination of the positional deviation (probe skew) [2), 3)] 0 = Determine positional deviation 1 = Do not determine positional deviation
				HUNDRED THOUSANDS: Reserved 0 = 0
				ONE MILLION:adapt tool length [7)] 0 = Do not adapt tool length (only trigger points) 1 = Adapt tool length
2	Icon+ number	S_PRNUM	Number of the field of the probe parameters (not probe number) (default=1)	
3		S_CALNUM	Number of the calibration groove for calibration on a groove (default=1) [5)]	
4		S_SETV	Setpoint for calibration on a surface	
5	X0	S_MA	Measuring axis (number of the axis) [6)] (default=1)	
			Values:	1 = 1st axis of the plane (for G18 Z) 2 = 2nd axis of the plane (for G18 X) 3 = 3rd axis of the plane (for G18 Y) [6)]
6	+-	S_MD	Measuring direction (default=1)	
			Values:	0 = Positive measuring direction 1 = Negative measuring direction
7	DFA	S_FA	Measurement path	
8	TSA	S_TSA	Safe area	
9	VMS	S_VMS	Variable measuring velocity for calibration [2)]	
10	Measure- ments	S_NMSP	Number of measurements at the same location [2)] (default=1)	

No.	Screen form parameter	Cycle parameter	Meaning	
11		S_MCBIT	Reserved	
12		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
13		_AMODE	Alternative mode	

1) All default values = 0 or marked as default=x

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) Only relevant for calibration in two axis directions

4) Only measuring axis and measuring direction are determined automatically from the cutting edge position (SL) of the probe.
SL=8 → -X , SL=7 → -Z

5) The number of the calibration groove (n) refers to the following general setting data (all positions in MCS):
For cutting edge position SL=7:
SD54615 $SNS_MEA_CAL_EDGE_BASE_AX1[n] Position of the bottom of the groove in the 1st axis of the plane (for G18 Z)
SD54621 $SNS_MEA_CAL_EDGE_PLUS_DIR_AX2[n] Position of the groove side in the plus direction of the 2nd axis of the plane (for G18 X)
SD54622 $SNS_MEA_CAL_EDGE_MINUS_DIR_AX2[n] Position of the groove side in the minus direction of the 2nd axis of the plane
For cutting edge position SL=8:
SD54619 $SNS_MEA_CAL_EDGE_BASE_AX2[n] Position of the bottom of the groove in the 2nd axis of the plane
SD54620 $SNS_MEA_CAL_EDGE_UPPER_AX2[n] Position of the upper edge of the groove in the 2nd axis of the plane (only for prepositioning of the probe)
SD54617 $SNS_MEA_CAL_EDGE_PLUS_DIR_AX1[n] Position of the groove side in the plus direction of the 1st axis of the plane
SD54618 $SNS_MEA_CAL_EDGE_MINUS_DIR_AX1[n] Position of the groove side in the minus direction of the 1st axis of the plane
Note:
The position values for the groove wall +- can be determined roughly.
The groove width from the difference of the position values of the groove wall must be determined precisely (precision dial gauge).
For calibration in the groove, it is assumed that the tool length of the probe of the calibrated axis = 0.
The positions values for the groove base must also be determined precisely on the machine (no drawing dimensions).

6) Measuring axis S_MA=3 for calibration on a surface and on a turning machine with real 3rd axis of the plane (for G18 Y).

7) Adapt tool length when calibrating length in the groove, or for lengths at the surface.
Workpiece probe in lathes can be defined using 2 lengths (X Z).
Turning probe, type 580
cutting-edge position 7: For length calibration, optionally, the Z length is corrected.
Turning probe, type 580
cutting edge position 8: For length calibration, optionally, the X length is corrected
The tool length is not adapted for the measurement version, radius at groove or radius at the surface.
Only the corresponding trigger points are saved.

CYCLE974 measuring cycle parameters

```
PROC CYCLE974(INT S_MVAR,INT S_KNUM,INT S_KNUM1,INT S_PRNUM,REAL S_SETV,INT S_MA,REAL
S_FA,REAL S_TSA,REAL S_STA1,INT S_NMSP,STRING[32] S_TNAME,INT S_DLNUM,REAL S_TZL,REAL
S_TDIF,REAL S_TUL,REAL S_TLL,REAL S_TMV,INT S_K,INT S_EVNUM,INT S_MCBIT,INT _DMODE,INT
_AMODE,INT _DP)
```

Table 4-8 CYCLE974 call parameters [1)]

No.	Screen form parameter	Cycle parameter	Meaning	
1		S_MVAR	Measuring variant	
			Values:	UNITS: 0 = Measure front face 1 = Inside measurement 2 = Outside measurement
				TENS: Reserved
				HUNDREDS: Correction target 0 = Only measurement (no correction of the WO or no tool offset) 1 = Measurement, determination and correction of the WO (see S_KNUM) [3)] 2 = Measurement and tool offset (see S_KNUM1)
				THOUSANDS: Reserved
				TEN THOUSANDS: Measurement with or without reversal of the main spindle (turning spindle) 0 = Measurement without reversal 1 = Measurement with reversal
2	Selection	S_KNUM	Correction in work offset (WO) or basic WO or basic reference [2)]	
			Values:	UNITS:
				TENS: 0 = No correction 1 to max. 99 numbers of the work offset or 1 to max. 16 numbers of the basic offset
				HUNDREDS: Reserved
				THOUSANDS: Correction in WO or basic WO or basic reference 0 = Correction of the adjustable WO 1 = Correction of the channel-specific basic WO 2 = Correction of the basic reference 3 = Correction of the global basic WO 9 = Correction of the active WO or for G500, last active channel-specific basic WO
				TEN THOUSANDS: Coarse or fine correction in the WO, basic WO or basic reference 0 = Fine correction [6)] 1 = Coarse correction

No.	Screen form parameter	Cycle parameter	Meaning	
3	Selection	`S_KNUM1`	Correction in tool offset [2), 4)]	
			Values:	UNITS:
				TENS:
				HUNDREDS: 0 = No correction 1 to max. 999 D numbers (cutting edge numbers) for tool offset; for additive and setup offset, see also `S_DLNUM`
				THOUSANDS: 0 or unique D number
				TEN THOUSANDS: 0 or unique D number 1 to max. 32000 if unique D numbers in MD have been set up
				HUNDRED THOUSANDS: Tool offset [2)] 0 = No specification (offset in tool geometry) 1 = Offset of length L1 2 = Offset of length L2 3 = Offset of length L3 4 = Radius offset
				ONE MILLION: Tool offset [2)] 0 = No specification (offset of the tool length wear) 1 = Tool offset, additive offset (AO) [5)] Tool offset value is added to the existing AO 2 = Tool offset, setup offset (SO) [5)] SO (new) = SO (old) + AO (old) offset value, AO (new) = 0 3 = Tool offset, setup offset (SO) [5)] Tool offset value is added to the existing SO 4 = Tool offset, geometry
				TEN MILLION: Tool offset [2)] 0 = No specification (offset in tool geometry normal (not inverted)) 1 = Offset inverted
				HUNDRED MILLIONS: Tool offset 0 = Tool offset without replacement tools 1 = Tool offset in replacement tool (_DP)
4	Icon+ number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (default=1)	
5	X0	`S_SETV`	Setpoint	
6	X	`S_MA`	Measuring axis (number of the axis) (default=1)	
			Values:	1 = 1st axis of the plane (for G18 Z) 2 = 2nd axis of the plane (for G18 X) 3 = 3rd axis of the plane (for G18 Y) [5)]
7	DFA	`S_FA`	Measurement path	
8	TSA	`S_TSA`	Safe area	
9	α	`S_STA1`	Starting angle for measurement with reversal	
10	Measure-ments	`S_NMSP`	Number of measurements at the same location [2)] (default=1)	
11	T	`S_TNAME`	Tool name [2)]	
12	DL	`S_DLNUM`	Setup additive offset DL number [5)]	

No.	Screen form parameter	Cycle parameter	Meaning	
13	ST	_DP	Number of the replacement tool (duplo number) to be corrected	
14	TZL	S_TZL	Work offset [2), 4)]	
15	DIF	S_TDIF	Dimensional difference check [2), 4)]	
16	TUL	S_TUL	Upper tolerance limit (incremental to the setpoint) [4)]	
17	TLL	S_TLL	Lower tolerance limit (incremental to the setpoint) [4)]	
18	TMV	S_TMV	Offset range for averaging [2)]	
19	FW	S_K	Weighting factor for averaging [2)]	
20	EVN	S_EVNUM	Number of the empirical mean value memory [2), 7)]	
21		S_MCBIT	Reserved	
22		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
23		_AMODE	Alternative mode	
			Values:	UNITS: Dimensional tolerance yes/no 0 = No 1 = Yes

1) All default values = 0 or marked as default=x

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) Correction in WO only possible for measurement without reversal

4) For tool offset in the channel-specific MD 20360 $TOOL_PARAMETER_DEF_MASK , observe bit0 and bit1

5) Only if the "Setup additive offset" function has been set-up in the general MD 18108 $MN_MM_NUM_SUMCORR . In addition, in the general MD 18080 $MN_MM_TOOL_MANAGEMENT_MASK , bit8 must be set to 1.

6) If WO "fine" has not been set up in MDs, correction is according to WO "coarse"

7) Empirical averaging only possible for tool offset
Value range for empirical mean value memory:
1 to 20 numbers (n) of the empirical value memory, see channel-specific SD55623 $SCS_MEA_EMPIRIC_VALUE[n-1]
10000 to 200000 numbers (n) of the mean value memory, see channel-specific SD55625 $SCS_MEA_AVERAGE_VALUE[n-1]

CYCLE994 measuring cycle parameters

```
PROC CYCLE994(INT S_MVAR,INT S_KNUM,INT S_KNUM1,INT S_PRNUM,REAL S_SETV,INT S_MA,REAL
S_SZA,REAL S_SZO,REAL S_FA,REAL S_TSA,INT S_NMSP,STRING[32] S_TNAME,INT S_DLNUM,REAL
S_TZL,REAL S_TDIF,REAL S_TUL,REAL S_TLL,REAL S_TMV,INT S_K,INT S_EVNUM,INT S_MCBIT,INT
_DMODE,INT _AMODE,INT _DP)
```

Table 4-9 CYCLE994 call parameters [1)]

<table>
<tr><th>No.</th><th>Screen form parameter</th><th>Cycle parameter</th><th colspan="2">Meaning</th></tr>
<tr><td rowspan="5">1</td><td rowspan="5"></td><td rowspan="5">S_MVAR</td><td colspan="2">Measuring variant</td></tr>
<tr><td rowspan="4">Values:</td><td>UNITS: Inside or outside measurement (default = 1)
1 = Inside measurement
2 = Outside measurement</td></tr>
<tr><td>TENS: Reserved</td></tr>
<tr><td>HUNDREDS: Correction target
0 = Only measurement (no correction of the WO or no tool offset)
1 = Measurement and determination and correction of the WO (see S_KNUM) [3)]
2 = Measurement and tool offset (see S_KNUM1)</td></tr>
<tr><td>THOUSANDS: Bypass area
0 = No bypass area
1 = Bypass axis 1st axis of the plane (for G18 Z). Measuring axis, see S_MA.
2 = Bypass axis 2nd axis of the plane (for G18 X). Measuring axis, see S_MA.
3 = Bypass axis 3rd axis of the plane (for G18 Y). Measuring axis, see S_MA. [8)]</td></tr>
<tr><td rowspan="6">2</td><td rowspan="6">Selection</td><td rowspan="6">S_KNUM</td><td colspan="2">Correction of work offset (WO) or basic WO or basic reference [2)]</td></tr>
<tr><td rowspan="5">Values:</td><td>UNITS:</td></tr>
<tr><td>TENS:
0 = No correction
1 to max. 99 numbers of the work offset or
1 to max. 16 numbers of the basic offset</td></tr>
<tr><td>HUNDREDS: Reserved</td></tr>
<tr><td>THOUSANDS: Correction of WO or basic or basic reference
0 = Correction of the adjustable WO
1 = Correction of the channel-specific basic WO
2 = Correction of the basic reference
3 = Correction of the global basic WO
9 = Correction of the active WO or for G500 in last active channel-specific basic WO</td></tr>
<tr><td>TEN THOUSANDS: Coarse or fine correction in the WO, basic WO or basic reference
0 = Fine correction [6)]
1 = Coarse correction</td></tr>
</table>

No.	Screen form parameter	Cycle parameter	Meaning	
3	Selection	`S_KNUM1`	Correction in tool offset [2), 4)]	
			Values:	UNITS:
				TENS:
				HUNDREDS: 0 = No correction 1 to max. 999 D numbers (cutting edge numbers) for tool offset; for additive and setup offset, see also `S_DLNUM`
				THOUSANDS: 0 or unique D numbers
				TEN THOUSANDS: 0 or unique D numbers 1 to max. 32000, if unique D numbers in MD have been set up
				HUNDRED THOUSANDS: Tool offset [2)] 0 = No specification (offset tool geometry) 1 = Offset of length L1 2 = Offset of length L2 3 = Offset of length L3 4 = Radius offset
				ONE MILLION: Tool offset [2)] 0 = No specification (offset of the tool length wear) 1 = Tool offset, additive offset (AO) [5)] Tool offset value is added to the existing AO 2 = Tool offset, setup offset (SO) [5)] SO (new) = SO (old) + AO (old) offset value, AO (new) = 0 3 = Tool offset, setup offset (SO) [5)] Tool offset value is added to the existing SO 4 = Tool offset, geometry
				TEN MILLION: Tool offset [2)] 0 = No specification (offset in tool geometry normal, not inverted) 1 = Offset inverted
				HUNDRED MILLIONS: Tool offset 0 = Tool offset without replacement tools 1 = Tool offset in replacement tool (_DP)
4	Icon+ number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (default=1)	
5	X0	`S_SETV`	Setpoint	
6	X	`S_MA`	Number of the measuring axis (default=1)	
			Values:	1 = 1st axis of the plane (for G18 Z) 2 = 2nd axis of the plane (for G18 X) 3 = 3rd axis of the plane (for G18 Y) [8)]
7	X1	`S_SZA`	Bypass distance in the measured axis	
8	Y1	`S_SZO`	Bypass distance in the bypass axis	
9	DFA	`S_FA`	Measurement path	
10	TSA	`S_TSA`	Safe area	
11	Measure- ments	`S_NMSP`	Number of measurements at the same location [2)] (default=1)	
12	T	`S_TNAME`	Tool name [2)]	

No.	Screen form parameter	Cycle parameter	Meaning	
13	DL	`S_DLNUM`	Setup additive offset DL number [5)]	
14	ST	`_DP`	Number of the replacement tool (duplo number) to be corrected	
15	TZL	`S_TZL`	Work offset [2), 4)]	
16	DIF	`S_TDIF`	Dimensional difference check [2), 4)]	
17	TUL	`S_TUL`	Upper tolerance limit (incremental to the setpoint) [4)]	
18	TLL	`S_TLL`	Lower tolerance limit (incremental to the setpoint) [4)]	
19	TMV	`S_TMV`	Offset range for averaging [2)]	
20	FW	`S_K`	Weighting factor for averaging [2)]	
21	EVN	`S_EVNUM`	Number of the empirical value memory [2), 7)]	
22		`S_MCBIT`	Reserved	
23		`_DMODE`	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
24		`_AMODE`	Alternative mode	
			Values:	UNITS: Dimensional tolerance yes/no 0 = No 1 = Yes

1) All default values = 0 or marked as default=x

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) Correction in WO only possible for measurement without reversal

4) For tool offset, observe the channel MD 20360 $TOOL_PARAMETER_DEF_MASK

5) Only if the "Setup additive offset" function has been set-up in the general MD 18108 $MN_MM_NUM_SUMCORR . In addition, the general MD 18080 $MN_MM_TOOL_MANAGEMENT_MASK , bit8 must be set to 1.

6) If WO "fine" has not been set up in MDs, correction is according to WO "coarse"

7) Empirical averaging only possible for tool offset
Value range for empirical mean value memory:
1 to 20 numbers (n) of the empirical value memory, see channel-specific SD55623 $SCS_MEA_EMPIRIC_VALUE[n-1]
10000 to 200000 numbers (n) of the mean value memory, see channel-specific SD55625 $SCS_MEA_AVERAGE_VALUE[n-1]

8) If Y axis is available on the machine

CYCLE976 measuring cycle parameters

```
PROC CYCLE976(INT S_MVAR,INT S_PRNUM,REAL S_SETV,REAL S_SETV0,INT S_MA,INT S_MD,REAL
S_FA,REAL S_TSA,REAL S_VMS,REAL S_STA1,INT S_NMSP,INT S_SETV1,INT _DMODE,INT _AMODE)
```

Table 4-10 CYCLE976 call parameters [1)]

No.	Screen form parameter	Cycle parameter	Meaning	
1		`S_MVAR`	Measuring version (default=1000)	
			Values:	UNITS: Calibration on surface, calibration ball or in calibration ring [2)] 0 = Length on surface with known setpoint 1 = Radius in calibration ring with known diameter (setpoint) and known center point. 2 = Radius in calibration ring with known diameter (setpoint) and an unknown center point 3 = Radius and length at the calibration ball 4 = Radius at the edge with known setpoint. Note selection of measuring axis and measuring direction. [3)] 5 = Radius between two edges with known setpoint and edge clearance. Measuring axis should be selected.
				TENS: Reserved 0 = 0
				HUNDREDS: Reserved 0 = 0
				THOUSANDS: Selection of measuring axis and measuring direction during calibration 0 = No specification (no selection of the measuring axis and measuring direction required) [8)] 1 = Specify selection of measuring axis and measuring direction, see `S_MA`, `S_MD` (one measuring direction in a measuring axis) 2 = Specify selection of measuring axis, see `S_MA` (two measuring directions in a measuring axis)
				TEN THOUSANDS: Determination of the positional deviation (probe skew) [2)] 0 = Determine positional deviation of the probe [6)] 1 = Do not determine positional deviation
				HUNDRED THOUSANDS: Paraxial calibration or at an angle 0 = Paraxial calibration in the active WCS 1 = Calibration at an angle [7)]
				ONE MILLION: Determination of tool length during calibration on surface or on ball 0 = Tool length is not determined 1 = Tool length is determined [4)] 2 = Infeed axis is calibrated at the ball, the tool length is determined, the tool length measured difference is entered in the calibration data
2	Icon+ number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (default=1)	
3		`S_SETV`	Setpoint	
4	Z0	`S_SETV0`	Setpoint of the longitudinal reference for ball calibration	
5	X / Y / Z	`S_MA`	Measuring axis (number of the axis) [2), 6)] (default=1)	
			Values:	1 = 1st axis of the plane (for G17 X) 2 = 2nd axis of the plane (for G17 Y) 3 = 3rd axis of the plane (for G17 Z)

No.	Screen form parameter	Cycle parameter	Meaning
6	+-	S_MD	Measuring direction [2), 6)] Values: 0 = Positive 1 = Negative
7	DFA	S_FA	Measurement path
8	TSA	S_TSA	Safe area
9	VMS	S_VMS	Variable measuring velocity for calibration [2)]
10	α	S_STA1	Starting angle [2), 5)]
11	Measurements	S_NMSP	Number of measurements at the same location [2)] (default=1)
12	X0	S_SETV1	Edge reference point when calibrating between 2 edges [3)]
13		_DMODE	Display mode Values: UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
14		_AMODE	Alternative mode

1) All default values = 0 or marked as default=x

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) For "Radius in the calibration ring" calibration, the diameter and the center point of the ring must be known (four measuring directions).
For "Radius on two edges" calibration, the distance to the edges in the direction of the measuring axis must be known (two measuring directions).
For "Radius on one edge" calibration, the setpoint of the surface must be known.

4) Measuring variant only calibration on a surface (length on surface), corrected tool length results from `S_MD` and `S_MA`.

5) Only for measuring variant "Calibration ring, ... and known center point" (`S_MVAR=1xxx02`).

6) Measuring axis only for measuring variant `S_MVAR=0` or `=xx1x01` or `=xx2x01` or `=20000`
Measuring variant: "Calibration on a surface" → selection of measuring axis and measuring direction
or on the "Calibration ring, ... and known center point" → selection of an axis direction and selection of measuring axis and measuring direction
or on the "Calibration ring, ... and known center point" → selection of two axis directions and selection of measuring axis
or "Determination of the probe length" → `S_MA=3` → 3rd axis of the plane (for G17 Z)

7) Measuring version, only calibration in calibration ring or on calibration ball
For "Calibration on calibration ball", for measuring at an angle, the axis circles around the ball at the equator.

8) For "Radius in calibration ring" calibration with unknown center point, four measuring directions in the plane (for G17 +-X +-Y).
For "Length on surface" calibration in minus direction of the tool axis (for G17 -Z).

CYCLE978 measuring cycle parameters

```
PROC CYCLE978(INT S_MVAR,INT S_KNUM,INT S_KNUM1,INT S_PRNUM,REAL S_SETV,REAL S_FA,REAL
S_TSA,INT S_MA,INT S_MD,INT S_NMSP,STRING[32] S_TNAME,INT S_DLNUM,REAL S_TZL,REAL
S_TDIF,REAL S_TUL,REAL S_TLL,REAL S_TMV,INT S_K,INT S_EVNUM,INT S_MCBIT,INT _DMODE,INT
_AMODE,INT _DP)
```

Table 4-11 CYCLE978 call parameters [1)]

<table>
<tr><th>No.</th><th>Screen form parameter</th><th>Cycle parameter</th><th colspan="2">Meaning</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6"></td><td rowspan="6">S_MVAR</td><td colspan="2">Measuring variant</td></tr>
<tr><td rowspan="5">Values:</td><td>UNITS: Contour element
0 = Measure surface</td></tr>
<tr><td>TENS: Reserved</td></tr>
<tr><td>HUNDREDS: Correction target
0 = Only measurement (no correction of the WO or no tool offset)
1 = Measurement, determination and correction of the WO (see S_KNUM)
2 = Measurement and tool offset (see S_KNUM1)</td></tr>
<tr><td>THOUSANDS: Reserved</td></tr>
<tr><td>TEN THOUSANDS: Measurement with/without spindle reversal or align probe in the switching direction [9)]
0 = Measurement without spindle reversal, without probe alignment
1 = Measurement with spindle reversal
2 = Align probe in switching direction</td></tr>
<tr><td rowspan="6">2</td><td rowspan="6">Selection</td><td rowspan="6">S_KNUM</td><td colspan="2">Correction of work offset (WO) or basic WO or basic reference [2)]</td></tr>
<tr><td rowspan="5">Values:</td><td>UNITS:</td></tr>
<tr><td>TENS:
0 = No correction
1 to max. 99 numbers of the work offset or
1 to max. 16 numbers of the basic offset</td></tr>
<tr><td>HUNDREDS: Reserved</td></tr>
<tr><td>THOUSANDS: Correction of WO or basic or basic reference
0 = Correction of the adjustable WO
1 = Correction of the channel-specific basic WO
2 = Correction of the basic reference
3 = Correction of the global basic WO
9 = Correction of the active WO or for G500 in last active channel-specific basic WO</td></tr>
<tr><td>TEN THOUSANDS: Coarse or fine correction in the WO, basic WO or basic reference
0 = Fine correction [6)]
1 = Coarse correction</td></tr>
</table>

No.	Screen form parameter	Cycle parameter	Meaning	
3	Selection	`S_KNUM1`	Correction in tool offset [2]	
			Values:	UNITS:
				TENS:
				HUNDREDS: 0 = No correction 1 to max. 999 D numbers (cutting edge numbers) for tool offset, for additive and setup offset, see also `S_DLNUM`
				THOUSANDS: 0 or unique D numbers
				TEN THOUSANDS: 0 or unique D numbers 1 to max. 32000 if unique D numbers in MDs have been set up
				HUNDRED THOUSANDS: Tool offset [2] 0 = No specification (offset in tool geometry) 1 = Offset of length L1 2 = Offset of length L2 3 = Offset of length L3 4 = Radius offset
				ONE MILLION: Tool offset [2] 0 = No specification (offset of the tool radius wear) 1 = Tool offset, additive offset (AO) [5] Tool offset value is added to the existing AO 2 = Tool offset, setup offset (SO) [5] SO (new) = SO (old) + AO (old) offset value, AO (new) = 0 3 = Tool offset, setup offset (SO) [5] Tool offset value is added to the existing SO 4 = Tool offset, geometry
				TEN MILLION: Tool offset [2] 0 = No specification (offset in tool geometry normal, not inverted) 1 = Offset inverted
				HUNDRED MILLIONS: Tool offset 0 = Tool offset without replacement tools 1 = Tool offset in replacement tool (_DP)
4	Icon+number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (value range 1 to 40)	
5	X0	`S_SETV`	Setpoint	
6	DFA	`S_FA`	Measurement path	
7	TSA	`S_TSA`	Safe area	
8	X	`S_MA`	Number of the measuring axis [7] (value range 1 to 3)	
			Values:	1 = 1st axis of the plane (for G17 X) 2 = 2nd axis of the plane (for G17 Y) 3 = 3rd axis of the plane (for G17 Z) measurement in tool direction
9		`S_MD`	Measuring direction of the measuring axis	
			Values:	1 = Positive measuring direction 2 = Negative measuring direction
10	Measurements	`S_NMSP`	Number of measurements at the same location [2] (value range 1 to 9)	
11	TR	`S_TNAME`	Tool name [3]	

No.	Screen form parameter	Cycle parameter	Meaning	
12	DL	S_DLNUM	Setup additive offset DL number 5)	
13	ST	_DP	Number of the replacement tool (duplo number) to be corrected	
14	TZL	S_TZL	Work offset 2), 3)	
15	DIF	S_TDIF	Dimensional difference check 2), 3)	
16	TUL	S_TUL	Upper tolerance limit (incremental to the setpoint) 3)	
17	TLL	S_TLL	Lower tolerance limit (incremental to the setpoint) 3)	
18	TMV	S_TMV	Offset range for averaging 2)	
19	FW	S_K	Weighting factor for averaging 2)	
20	EVN	S_EVNUM	Date set, empirical value memory 2), 8)	
21		S_MCBIT	Reserved	
22		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
23		_AMODE	Alternative mode	
			Values:	UNITS: Dimensional tolerance yes/no 0 = No 1 = Yes

1) All default values = 0 or marked as the range of values a to b

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) Only for offset in tool, otherwise parameter = ""

4) Only for offset in tool and dimensional tolerance "Yes", otherwise parameter = 0

5) Only if the "Setup additive offset" function has been set-up in the general MD 18108 $MN_MM_NUM_SUMCORR . In addition, in the general MD 18080 $MN_MM_TOOL_MANAGEMENT_MASK , bit8 must be set to 1.

6) If WO "fine" has not been set up in MDs, correction is according to WO "coarse"

7) Offset in tool geometry:
For measurement in the plane (`S_MA=1` or `S_MA=2`) Offset in tool radius
For measurement in tool direction (`S_MA=3`) Offset in tool length L1

8) Empirical averaging for tool offset and correction in WO possible
Value range for empirical mean value memory:
1 to 20 numbers (n) of the empirical value memory, see channel-specific SD55623 $SCS_MEA_EMPIRIC_VALUE[n-1]
10000 to 200000 numbers (n) of the mean value memory, see channel-specific SD55625 $SCS_MEA_AVERAGE_VALUE[n-1]

9) When measuring with spindle reversal, the radius/diameter of the probe must be precisely determined. This should be realized with a calibration variant of the CYCLE976 radius at the ring or at the edge or at the ball. Otherwise, the measurement result will be falsified.

CYCLE998 measuring cycle parameters

```
PROC CYCLE998(INT S_MVAR,INT S_KNUM,INT S_RA,INT S_PRNUM,REAL S_SETV,REAL S_STA1,REAL
S_INCA,REAL S_FA,REAL S_TSA,INT S_MA,INT S_MD,REAL S_ID,REAL S_SETV0,REAL S_SETV1,REAL
S_SETV2,REAL S_SETV3,INT S_NMSP,INT S_EVNUM,INT _DMODE,INT _AMODE)
```

Table 4-12 CYCLE998 call parameters [1)]

No.	Screen form parameter	Cycle parameter	Meaning	
1		S_MVAR	Measuring variant (default=5)	
			Values:	UNITS: Contour element 5 = Measure edge (one angle) 6 = Measure plane (two angles)
				TENS: Reserved
				HUNDREDS: Correction target 0 = Only measurement and no correction of WO 1 = Measurement and determination and correction of the WO (see S_KNUM)
				THOUSANDS: Protection zone 0 = No consideration of a protection zone 1 = Consideration of a protection zone
				TEN THOUSANDS: Measurement with spindle reversal (difference measurement) 0 = Measurement without spindle reversal 1 = Measurement with spindle reversal
				HUNDRED THOUSANDS: Measurement at an angle or paraxial 0 = Measurement at an angle 1 = Measurement paraxial
2	Selection	S_KNUM	Correction of work offset (WO) or basic WO or basic reference [2)]	
			Values:	UNITS:
				TENS: 0 = No correction 1 to max. 99 numbers of the work offset or 1 to max. 16 numbers of the basic offset
				HUNDREDS: Reserved
				THOUSANDS: Correction of WO or basic or basic reference 0 = Correction of the adjustable WO 1 = Correction of the channel-specific basic WO 2 = Correction of the basic reference 9 = Correction of the active WO or for G500 in last active channel-specific basic WO
				TEN THOUSANDS: Coarse or fine correction in the WO or basic WO or basic reference [3)] 0 = Fine correction 1 = Coarse correction
3		S_RA	Offset target coordinate rotation or rotary axis [14)]	
	A, B, C		Values:	0 = Correction target coordinate rotation around the axis that results from parameter S_MA [4)] >0 = Correction target rotary axis. Number of the channel axis number of the rotary axis (preferably rotary table). The angular offset is made in the translatory part of the WO of the rotary axis.
4	Icon+ number	S_PRNUM	Number of the field of the probe parameter (default=1)	
5	DX / DY / DZ	S_SETV	Distance (incremental) from the starting position to measuring point P1 of the measuring axis (S_MA) [5)]	

No.	Screen form parameter	Cycle parameter	Meaning
6	α	`S_STA1`	Angle setpoint for "Align edge" or for "Align plane" around the 1st axis of the plane (for G17 X) [9]
7	β	`S_INCA`	Angle setpoint for "Align plane" around the 2nd axis of the plane (for G17 Y) [9]
8	DFA	`S_FA`	Measurement path
9	TSA	`S_TSA`	Safe area Monitoring of the angle difference to the angle setpoint [degrees] [6]
10	X / Y / Z	`S_MA`	Measuring axis, offset axis [7] (default=201) Values: UNITS: Number of the measuring axis 1 = 1st axis of the plane (for G17 X) 2 = 2nd axis of the plane (for G17 Y) 3 = 3rd axis of the plane (for G17 Z) TENS: Reserved HUNDREDS: Number of the offset axis 1 = 1st axis of the plane (for G17 X) 2 = 2nd axis of the plane (for G17 Y) 3 = 3rd axis of the plane (for G17 Z)
11	+-	`S_MD`	Measuring direction of the measuring axis [8] Values: 0 = Measuring direction is determined from the setpoint and the actual position of the measuring axis (compatibility) 1 = Positive measuring direction 2 = Negative measuring direction
12	L2	`S_ID`	For measuring variant "Align edge": Distance (incremental) between the measuring points P1 and P2 in the offset axis (value >0) For measuring variant "Align plane", the parameters listed below apply.
13	L2	`S_SETV0`	Distance between the measuring points P1 and P2 in the 1st axis of the plane [10]
14		`S_SETV1`	Distance between the measuring points P1 and P2 in the 2nd axis of the plane [11), 12]
15	L3x	`S_SETV2`	Distance between the measuring points P1 and P3 in the 1st axis of the plane [11]
16	L3y	`S_SETV3`	Distance between the measuring points P1 and P3 in the 2nd axis of the plane [10]
17	Measurements	`S_NMSP`	Number of measurements at the same location [2] (default=1)
18		`S_EVNUM`	Date set, empirical value memory [2), 13]
19		`_DMODE`	Display mode Values: UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
20		`_AMODE`	Reserved (alternative mode)

1) All default values = 0 or marked as default=x

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) WO "fine" only if correction target is rotary axis and MD 52207 $MCS_AXIS_USAGE_ATTRIB[n] Bit6=1.
If WO has not been set up in MDs, correction is according to WO "coarse".

4) Example for offset in coordinate rotation: `S_MA=102` Measuring axis Y, offset axis X results in coordinate rotation around Z (for G17)

5) Value only relevant for protection zone "Yes" (`S_MVAR` THOUSANDS position = 1)

6) When positioning from measuring point P1 to measuring point P2 in the offset axis, the angles in parameters `S_STA1` and `S_TSA` are added.

7) Number of the measuring axis must not be the same as the number of the offset axis (e.g. 101 not permitted)

8) Measuring direction only for "Align edge" and "Measurement paraxial" (`S_MVAR=10x105`)

9) Angular range `S_STA1` ±45 degrees for "Align edge"
Angular range `S_STA1` 0 to +60 degrees and `S_INCA` ±30 degrees for "Align plane"

10) For measuring variants "Align plane" and "Align edge"

11) For measuring variants "Measure plane" and "Measurement paraxial"

12) Not for measuring cycle version SW04.04.

13) Empirical value generation for correction in WO; value range of the empirical mean value memory:
1 to 20 numbers (n) of the empirical value memory, see channel-specific SD55623 $SCS_MEA_EMPIRIC_VALUE[n-1]

14) The parameter S_RA is only relevant for the measuring variant "Align edge" (S_MVAR xxxxx5).

CYCLE977 measuring cycle parameters

```
PROC CYCLE977(INT S_MVAR,INT S_KNUM,INT S_KNUM1,INT S_PRNUM,REAL S_SETV,REAL S_SETV0,REAL
S_SETV1,REAL S_FA,REAL S_TSA,REAL S_STA1,REAL S_ID,REAL S_SZA,REAL S_SZO,INT S_MA,INT
S_NMSP,STRING[32] S_TNAME,INT S_DLNUM,REAL S_TZL,REAL S_TDIF,REAL S_TUL,REAL S_TLL,REAL
S_TMV,INT S_K,INT S_EVNUM,INT S_MCBIT,INT _DMODE,INT _AMODE,REAL S_XM,REAL_S_YM,INT _DP)
```

Table 4-13 CYCLE977 call parameters [1)]

<table>
<tr><th>No.</th><th>Screen form parameter</th><th>Cycle parameter</th><th colspan="2">Meaning</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6"></td><td rowspan="6"><code>S_MVAR</code></td><td colspan="2">Measuring variant</td></tr>
<tr><td rowspan="5">Values:</td><td>UNITS: Contour element (value range 1 to 6)
1 = Measure hole
2 = Measure spigot (shaft)
3 = Measure groove
4 = Measure rib
5 = Measure rectangle, inside
6 = Measure rectangle, outside</td></tr>
<tr><td>TENS: Reserved</td></tr>
<tr><td>HUNDREDS: Correction target
0 = Only measurement (no correction of the WO or no tool offset)
1 = Measurement and determination and correction of the WO (see <code>S_KNUM</code>)
2 = Measurement and tool offset (see <code>S_KNUM1</code>)</td></tr>
<tr><td>THOUSANDS: Protection zone
0 = No consideration of a protection zone
1 = Consideration of a protection zone</td></tr>
<tr><td>TEN THOUSANDS: Measurement with/without spindle reversal (differential measurement) or align probe in the switching direction
0 = Measurement without spindle reversal, do not align probe
1 = Measurement with spindle reversal
2 = Align probe in switching direction</td></tr>
</table>

No.	Screen form parameter	Cycle parameter	Meaning	
2	Selection	`S_KNUM`	Correction of work offset (WO) or basic WO or basic reference 2)	
			Values:	UNITS:
				TENS: 0 = No correction 1 to max. 99 numbers of the work offset or 1 to max. 16 numbers of the basic offset
				HUNDREDS: Reserved
				THOUSANDS: Correction of WO or basic or basic reference 0 = Correction of the adjustable WO 1 = Correction of the channel-specific basic WO 2 = Correction of the basic reference 3 = Correction of the global basic WO 9 = Correction of the active WO or for G500 in last active channel-specific basic WO
				TEN THOUSANDS: Coarse or fine correction in the WO, basic WO or basic reference 0 = Fine correction 6) 1 = Coarse correction

No.	Screen form parameter	Cycle parameter	Meaning	
3	Selection	`S_KNUM1`	Correction in tool offset [2]	
			Values:	UNITS:
				TENS:
				HUNDREDS: 0 = No correction 1 to max. 999 D numbers (cutting edge numbers) for tool offset; for additive and setup offset, see also `S_DLNUM`
				THOUSANDS: 0 or unique D numbers
				TEN THOUSANDS: 0 or unique D numbers 1 to max. 32000 if unique D numbers in MDs have been set up
				HUNDRED THOUSANDS: Tool offset [2] 0 = No specification (offset tool radius) 1 = Offset of length L1 2 = Offset of length L2 3 = Offset of length L3 4 = Radius offset
				ONE MILLION: Tool offset [2] 0 = No specification (offset of the tool radius wear) 1 = Tool offset, additive offset (AO) [5] Tool offset value is added to the existing AO 2 = Tool offset, setup offset (SO) [5] SO (new) = SO (old) + AO (old) offset value, AO (new) = 0 3 = Tool offset, setup offset (SO) [5] Tool offset value is added to the existing SO 4 = Tool offset, geometry
				TEN MILLION: Tool offset [2] 0 = No specification (offset in tool geometry normal, not inverted) 1 = Offset inverted
				HUNDRED MILLIONS: Tool offset 0 = Tool offset without replacement tools 1 = Tool offset in replacement tool (_DP)
4	Icon+ number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (value range 1 to 40)	
5	X0	`S_SETV`	Setpoint	
6	X0	`S_SETV0`	Setpoint for rectangle in 1st axis of the plane (X for G17)	
7	Y0	`S_SETV1`	Setpoint for rectangle in 2nd axis of the plane (Y for G17)	
8	XM	`S_XM`	Setpoint center point input geometry axis X	
9	YM	`S_YM`	Setpoint center point input geometry axis Y	
10	DFA	`S_FA`	Measurement path	
11	TSA	`S_TSA`	Safe area	
12	α 0	`S_STA1`	Starting angle	

No.	Screen form parameter	Cycle parameter	Meaning	
13	DZ	S_ID	Absolute incremental value 1. Incremental infeed of the 3rd axis of the plane (Z for G17) Infeed direction via sign of S_ID. For measurement of spigot, rib and rectangle outside, S_ID is used to define the lowering to measuring height. 2. Consideration of a protection zone For measurement of hole, groove and rectangle inside and a protection zone, S_ID is used to define the overtravel height.	
14	X1	S_SZA	Diameter or length (width) of the protection zone 7)	
15	Y1	S_SZO	For "Measure rectangle": Width of the protection zone of the 2nd axis of the plane	
16	X	S_MA	Number of the measuring axis 7) (only for measurement of groove or rib)	
			Values:	1 = 1st axis of the plane (for G17 X) 2 = 2nd axis of the plane (for G17 Y)
17	ST	_DP	Number of the replacement tool (duplo number) to be corrected	
18	Measurements	S_NMSP	Number of measurements at the same location 2) (value range 1 to 9)	
19	TR	S_TNAME	Tool name 2)	
20	DL	S_DLNUM	Setup additive offset DL number 5)	
21	TZL	S_TZL	Work offset 2), 4)	
22	DIF	S_TDIF	Dimensional difference check 2), 4)	
23	TUL	S_TUL	Upper tolerance limit (incremental to the setpoint) 4)	
24	TLL	S_TLL	Lower tolerance limit (incremental to the setpoint) 4)	
25	TMV	S_TMV	Offset range for averaging 2)	
26	FW	S_K	Weighting factor for averaging 2)	
27		S_EVNUM	Data set, empirical mean value memory 2), 8)	
28		S_MCBIT	Reserved	
29		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
30		_AMODE	Alternative mode	
			Values:	UNITS: Dimensional tolerance yes/no 0 = No 1 = Yes

1) All default values = 0 or marked as the range of values a to b

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) Only for offset in tool, otherwise parameter = ""

4) Only for offset in tool and dimensional tolerance "Yes", otherwise parameter = 0

5) Only if the "Setup additive offset" function has been set-up in the general MD 18108 $MN_MM_NUM_SUMCORR . In addition, in the general MD 18080 $MN_MM_TOOL_MANAGEMENT_MASK , bit8 must be set to 1.

6) If WO "fine" has not been set up in MDs, correction is according to WO "coarse"

7) Diameter or width of the protection zone within a hole or groove
Diameter or width of the protection zone outside of a spigot or rib

8) Empirical averaging possible for tool offset
Value range for empirical mean value memory:
1 to 20 numbers (n) of the empirical value memory, see channel-specific SD55623 $SCS_MEA_EMPIRIC_VALUE[n-1]
10000 to 200000 numbers (n) of the mean value memory, see channel-specific SD55625 $SCS_MEA_AVERAGE_VALUE[n-1]

CYCLE961 measuring cycle parameters

```
PROC CYCLE961(INT S_MVAR,INT S_KNUM,INT S_PRNUM,REAL S_SETV0,REAL S_SETV1,REAL
S_SETV2,REAL S_SETV3,REAL S_SETV4,REAL S_SETV5,REAL S_SETV6,REAL S_SETV7,REAL S_SETV8,REAL
S_SETV9,REAL S_STA1,REAL S_INCA,REAL S_ID,REAL S_FA,REAL S_TSA,INT S_NMSP,INT S_MCBIT,INT
_DMODE,INT _AMODE)
```

Table 4-14 CYCLE961 call parameters 1)

No.	Screen form parameter	Cycle parameter	Meaning	
1		`S_MVAR`	Measuring variant (default ≥ 6)	
			Values:	UNITS: Contour element 5 = Setup of right-angled inside corner, setpoint specification of angle and distances A1 to A3 6 = Setup of right-angled outside corner, setpoint specification of angle and distances A1 to A3 7 = Setup of inside corner, specification of angle and distances A1 to A4 8 = Setup of outside corner, specification of angle and distances A1 to A4
				TENS: Setpoint specification as distance or via four points 0 = Setpoint specification as distance (polar) 1 = Setpoint specification using four points (measuring points P1 to P4)
				HUNDREDS: Correction target 0 = Only measurement (no correction of the WO or no tool offset) 1 = Measurement and determination and correction of the WO, see `S_KNUM`
				THOUSANDS: Protection zone 0 = No consideration of a protection zone (obstacle) 1 = Consideration of a protection zone (obstacle), see `S_ID`
				TEN THOUSANDS: Position of the corner in the WCS 0 = Position of the corner is determined via parameter `S_STA1` (compatibility) 1 = Position 1 of the corner in the positioned starting point of the measurement 6) 2 = Position 2 of the corner, distances in the 1st axis of the plane (for G17 X) are negative (see `S_SETV0`, `S_SETV1`) 3 = Position 3 of the corner, distances in the 1st and 2nd axis of the plane (for G17 XY) are negative (see `S_SETV0` to `S_SETV3`) 4 = Position 4 of the corner, distances in the 2nd axis of the plane (for G17 Y) are negative (see `S_SETV2`, `S_SETV3`)

No.	Screen form parameter	Cycle parameter	Meaning
2	Selection	`S_KNUM`	Correction of work offset (WO) or basic WO or basic reference [2]
			Values: UNITS:
			TENS: 0 = No correction 1 to max. 99 numbers of the work offset or 1 to max. 16 numbers of the basic offset
			HUNDREDS: Reserved
			THOUSANDS: Correction of WO or basic or basic reference 0 = Correction of the adjustable WO 1 = Correction of the channel-specific basic WO 2 = Correction of the basic reference 9 = Correction of the active WO or for G500 in last active channel-specific basic WO
			TEN THOUSANDS: Coarse or fine correction in the WO, basic WO or basic reference 0 = Fine correction [5] 1 = Coarse correction
3	Icon+ number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (value range 1 to 40)
4	L1/X1	`S_SETV0`	Distance L1 between the pole and measuring point P1 in the direction of the 1st axis of the plane (for G17 X) [3] (if the actual distance L1=0, then L1 = M_SETV1 / 2 is automatically calculated) or starting point P1x of the 1st axis of the plane (for G17 X) [4]
5	L2/Y1	`S_SETV1`	Distance L2 between the pole and measuring point P2 in the direction of the 1st axis of the plane [3] or starting point P1y of the 2nd axis of the plane (for G17 Y) [4]
6	L3/X2	`S_SETV2`	Distance L3 between the pole and measuring point P3 in the direction of the 2nd axis of the plane [3] (if the distance L3=0, then for a corner that is not right angled, L3 = M_SETV3 / 2 is automatically calculated) or starting point P2x of the 1st axis of the plane [4]
7	L4/Y2	`S_SETV3`	Distance L4 between the pole and measuring point P3 in the direction of the 2nd axis of the plane with a corner that is not right angled [3] or starting point P2y of the 2nd axis of the plane [4]
8	XP/X3	`S_SETV4`	Position of the pole in the 1st axis of the plane [3] or starting point P3x of the 1st axis of the plane [4]
9	XP/Y3	`S_SETV5`	Position of the pole in the 2nd axis of the plane [3] or starting point P3y of the 2nd axis of the plane [4]
10	X4	`S_SETV6`	Starting point P4x of the 1st axis of the plane [4]
11	Y4	`S_SETV7`	Starting point P4y of the 2nd axis of the plane [4]
12	X0	`S_SETV8`	Setpoint of the measured corner in the 1st axis of the plane for correcting in WO
13	Y0	`S_SETV9`	Setpoint of the measured corner in the 2nd axis of the plane for correcting in WO
14	α0	`S_STA1`	Starting angle from the positive direction of the 1st axis of the plane to the reference edge of the workpiece in the MCS (+-270 degrees)
15	α1	`S_INCA`	Angle between workpiece reference edges when measuring a non-right-angled corner [7]
16	DZ	`S_ID`	Infeed amount at the measuring height for each measuring point for active protection zone (see `S_MVAR`).
17	DFA	`S_FA`	Measurement path

No.	Screen form parameter	Cycle parameter	Meaning
18	TSA	`S_TSA`	Safe area Monitoring the angular difference to the angle setpoint [degrees]
19	Measurements	`S_NMSP`	Number of measurements at the same location [2] (value range 1 to 9) [2]
20		`S_MCBIT`	Reserved
21		`_DMODE`	Display mode Values: UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
22		`_AMODE`	Alternative mode

1) All default values = 0 or marked as the range of values a to b

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) Input of the measuring points in polar coordinates, taking into account the starting angle `S_STA1` for measuring point 3 or 4 of the incremental angle `S_INCA`.

4) Input of the measuring points in the right-angled coordinate system (input using 4 points)

5) If WO "fine" has not been set up in MDs, correction is according to WO "coarse".

6) Value range of angle `S_INCA`: -180 to +180 degrees

7) Starting angle S_STA1, value range: right-angled corner: +- 90 degrees, any corner: +- 45 degrees

CYCLE979 measuring cycle parameters

```
PROC CYCLE979(INT S_MVAR,INT S_KNUM,INT S_KNUM1,INT S_PRNUM,REAL S_SETV,REAL S_FA,REAL
S_TSA,REAL S_CPA,REAL S_CPO,REAL S_STA1,REAL S_INCA,INT S_NMSP,STRING[32] S_TNAME,REAL
S_DLNUM,REAL S_TZL,REAL S_TDIF,REAL S_TUL,REAL S_TLL,REAL S_TMV,INT S_K,INT S_EVNUM,INT
S_MCBIT,INT _DMODE,INT _AMODE,REAL S_XM,REAL S_YM,INT _DP)
```

Table 4-15 CYCLE979 call parameters 0)

No.	Screen form parameter	Cycle parameter	Meaning	
1		`S_MVAR`	Measuring variant	
			Values:	UNITS: Contour element 1 = Measure hole 2 = Measure spigot (shaft)
				TENS: Reserved
				HUNDREDS: Correction target 0 = Only measurement (no correction of the WO or no tool offset) 1 = Measurement and determination and correction of the WO (see `S_KNUM`) 2 = Measurement and tool offset (see `S_KNUM1`)
				THOUSANDS: Number of measurement points 0 = 3 measuring points 1 = 4 measuring points
				TEN THOUSANDS: Measurement with/without spindle reversal (differential measurement) or align probe in the switching direction 0 = Measurement without spindle reversal, without probe alignment 1 = Measurement with spindle reversal 2 = Align probe in switching direction
2	Selection	`S_KNUM`	Correction of work offset (WO) or basic WO or basic reference 2)	
			Values:	UNITS:
				TENS: 0 = No correction 1 to max. 99 numbers of the work offset or 1 to max. 16 numbers of the basic offset
				HUNDREDS: Reserved
				THOUSANDS: Correction of WO or basic or basic reference 0 = Correction of the adjustable WO 1 = Correction of the channel-specific basic WO 2 = Correction of the basic reference 3 = Correction of the global basic WO 9 = Correction of the active WO or for G500 in last active channel-specific basic WO
				TEN THOUSANDS: Coarse or fine correction in the WO, basic WO or basic reference 0 = Fine correction 6) 1 = Coarse correction

No.	Screen form parameter	Cycle parameter	Meaning	
3	Selection	`S_KNUM1`	Correction in tool offset [2)]	
			Values:	UNITS:
				TENS:
				HUNDREDS: 0 = No correction 1 to max. 999 D numbers (cutting edge numbers) for tool offset; for additive and setup offset, see also `S_DLNUM`
				THOUSANDS: 0 or unique D numbers
				TEN THOUSANDS: 0 or unique D numbers 1 to max. 32000 if unique D numbers in MDs have been set up
				HUNDRED THOUSANDS: Tool offset [2)] 0 = No specification (offset in tool radius) 1 = Offset of length L1 2 = Offset of length L2 3 = Offset of length L3 4 = Radius offset
				ONE MILLION: Tool offset [2)] 0 = No specification (offset of the tool radius wear) 1 = Tool offset, additive offset (AO) [5)] Tool offset value is added to the existing AO 2 = Tool offset, setup offset (SO) [5)] SO (new) = SO (old) + AO (old) offset value, AO (new) = 0 3 = Tool offset, setup offset (SO) [5)] Tool offset value is added to the existing SO 4 = Tool offset, geometry
				TEN MILLION: Tool offset [2)] 0 = No specification (offset in tool geometry normal, not inverted) 1 = Offset inverted
				HUNDRED MILLIONS: Tool offset 0 = Tool offset without replacement tools 1 = Tool offset in replacement tool (_DP)
4	Icon+ number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (value range 1 to 40)	
5	X0	`S_SETV`	Setpoint	
6	DFA	`S_FA`	Measurement path	
7	TSA	`S_TSA`	Safe area	
8	X0	`S_CPA`	Center point of the 1st axis of the plane (for G17 X)	
9	Y0	`S_CPO`	Center point of the 2nd axis of the plane (for G17 Y)	
10	XM	`S_XM`	Setpoint center point input geometry axis X	
11	YM	`S_YM`	Setpoint center point input geometry axis Y	
12	alpha 0	`S_STA1`	Starting angle [7)]	
13	alpha 1	`S_INCA`	Incremental angle [8)]	
14	Measure- ments	`S_NMSP`	Number of measurements at the same location [1)] (value range 1 to 9)	
15	ST	`_DP`	Number of the replacement tool (duplo number) to be corrected	

<table>
<tr><th>No.</th><th>Screen form parameter</th><th>Cycle parameter</th><th colspan="2">Meaning</th></tr>
<tr><td>16</td><td>T</td><td>S_TNAME</td><td colspan="2">Tool name 2)</td></tr>
<tr><td>17</td><td>DL</td><td>S_DLNUM</td><td colspan="2">Setup additive offset DL number 1), 4)</td></tr>
<tr><td>18</td><td>TZL</td><td>S_TZL</td><td colspan="2">Work offset 1), 2)</td></tr>
<tr><td>19</td><td>DIF</td><td>S_TDIF</td><td colspan="2">Dimensional difference check 1), 2)</td></tr>
<tr><td>20</td><td>TUL</td><td>S_TUL</td><td colspan="2">Upper tolerance limit (incremental to the setpoint) 2)</td></tr>
<tr><td>21</td><td>TLL</td><td>S_TLL</td><td colspan="2">Lower tolerance limit (incremental to the setpoint) 2)</td></tr>
<tr><td>22</td><td>TMV</td><td>S_TMV</td><td colspan="2">Offset range for averaging 1)</td></tr>
<tr><td>23</td><td>FW</td><td>S_K</td><td colspan="2">Weighting factor for averaging 1)</td></tr>
<tr><td>24</td><td></td><td>S_EVNUM</td><td colspan="2">Date set, empirical value memory 1), 6)</td></tr>
<tr><td>25</td><td></td><td>S_MCBIT</td><td colspan="2">Reserved</td></tr>
<tr><td rowspan="2">26</td><td rowspan="2"></td><td rowspan="2">_DMODE</td><td colspan="2">Display mode</td></tr>
<tr><td>Values:</td><td>UNITS: Machining plane G17/G18/G19
0 = Compatibility, the plane active before the cycle call remains active
1 = G17 (only active in the cycle)
2 = G18 (only active in the cycle)
3 = G19 (only active in the cycle)</td></tr>
<tr><td rowspan="2">27</td><td rowspan="2"></td><td rowspan="2">_AMODE</td><td colspan="2">Alternative mode</td></tr>
<tr><td>Values:</td><td>UNITS: Dimensional tolerance yes/no
0 = No
1 = Yes</td></tr>
</table>

0) All default values = 0 or marked as the range of values a to b.

1) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

2) Only for offset in tool, otherwise parameter = ""

3) Only for offset in tool and dimensional tolerance "Yes", otherwise parameter = 0

4) Only if the "Setup additive offset" function has been set-up in the general MD 18108 $MN_MM_NUM_SUMCORR .

5) If WO "fine" has not been set up in MDs, correction is according to WO "coarse"

6) Empirical averaging only possible for tool offset
Value range for empirical mean value memory:
1 to 20 numbers (n) of the empirical value memory, see channel-specific SD55623 $SCS_MEA_EMPIRIC_VALUE[n-1]
10000 to 200000 numbers (n) of the mean value memory, see channel-specific SD55625 $SCS_MEA_AVERAGE_VALUE[n-1]

7) Value range of starting angle -360 to +360 degrees

8) Value range of incremental angle >0 to ≤90 degrees for four measuring points or >0 to ≤120 degrees for three measuring points.

CYCLE997 measuring cycle parameters

```
PROC CYCLE997 (INT S_MVAR,INT S_KNUM,INT S_PRNUM,REAL S_SETV,REAL S_FA,REAL S_TSA,REAL
S_STA1,REAL S_INCA,REAL S_SETV0,REAL S_SETV1,REAL S_SETV2,REAL S_SETV3,REAL S_SETV4,REAL
S_SETV5,REAL S_SETV6,REAL S_SETV7,REAL S_SETV8,REAL S_TNVL,INT S_NMSP,INT S_MCBIT,INT
_DMODE,INT _AMODE)
```

Table 4-16 CYCLE997 call parameters [1), 2)]

No.	Screen form parameter	Cycle parameter	Meaning	
1		S_MVAR	Measuring variant (default =9)	
			Values:	UNITS: Contour element 9 = Measure ball
				TENS: Repeat measurement 0 = Without measurement repetition 1 = With measurement repetition
				HUNDREDS: Correction target 0 = Only measurement (no correction of WO) 1 = Measurement and determination and correction of the WO (see S_KNUM)
				THOUSANDS: Measuring strategy 0 = Paraxial measurement, without starting angle, probe alignment corresponding to SD 55740 $SCS_MEA_FUNCTION_MASK, bit 1 1 = Circling measurement, with starting angle, probe alignment corresponding to SD 55740 $SCS_MEA_FUNCTION_MASK, bit 1 2 = Circling measurement, with starting angle, align probe in the switching direction 3 = Paraxial measurement, with starting angle, probe alignment corresponding to SD 55740 $SCS_MEA_FUNCTION_MASK, bit 1 4 = Paraxial measurement, with starting angle, align probe in the switching direction
				TEN THOUSANDS: Number of balls to be measured 0 = Measure one ball 1 = Measure three balls
				HUNDRED THOUSANDS: Number of measuring points, only for measurement at an angle (note measuring strategy: THOUSANDS position > 0) 0 = Three measuring points for measurement at an angle (traversing around the ball) 1 = Four measuring points for measurement at an angle (traversing around the ball)
				ONE MILLION: Determination of the diameter setpoint of the ball 0 = No determination of the diameter setpoint of the ball 1 = Determination of the diameter setpoint of the ball

No.	Screen form parameter	Cycle parameter	Meaning	
2	Selection	`S_KNUM`	Correction in work offset (WO) or basic or basic reference [3]	
			Values:	UNITS:
				TENS: 0 = No correction 1 to max. 99 numbers of the work offset or 1 to max. 16 numbers of the basic offset
				HUNDREDS: Reserved
				THOUSANDS: Correction in WO or basic WO or basic reference 0 = Correction in adjustable WO 1 = Correction in channel-specific basic WO 2 = Correction in basic reference 3 = Correction in global basic WO [7] 9 = Correction in active WO or for G500 in last active channel-specific basic WO
				TEN THOUSANDS: Correction in WO or basic WO or basic reference coarse or fine 0 = Fine correction [6] 1 = Coarse correction
3	Icon+ number	`S_PRNUM`	Number of the field of the probe parameters (not probe number) (value range 1 to 40)	
4		`S_SETV`	Diameter of the ball(s) [4]	
5	DFA	`S_FA`	Measurement path	
6	TSA	`S_TSA`	Safe area	
7	alpha 0	`S_STA1`	Starting angle for measurement at an angle	
8	alpha 1	`S_INCA`	Incremental angle for measurement at an angle	
9	X1	`S_SETV0`	Set position of the 1st ball of the 1st axis of the plane (for G17 X) for measuring 3 balls	
10	Y1	`S_SETV1`	Set position of the 1st ball of the 2nd axis of the plane (for G17 Y) for measuring 3 balls	
11	Z1	`S_SETV2`	Set position of the 1st ball of the 3rd axis of the plane (for G17 Z) for measuring 3 balls	
12	X2	`S_SETV3`	Set position of the 2nd ball of the 1st axis of the plane for measuring 3 balls	
13	Y2	`S_SETV4`	Set position of the 2nd ball of the 2nd axis of the plane for measuring 3 balls	
14	Z2	`S_SETV5`	Set position of the 2nd ball of the 3rd axis of the plane for measuring 3 balls	
15	X3	`S_SETV6`	Set position of the 3rd ball of the 1st axis of the plane for measuring 3 balls	
16	Y3	`S_SETV7`	Set position of the 3rd ball of the 2nd axis of the plane for measuring 3 balls	
17	Z3	`S_SETV8`	Set position of the 3rd ball of the 3rd axis of the plane for measuring 3 balls	
18	TVL	`S_TNVL`	Limit value for distortion of the triangle (sum of the deviations) for measuring 3 balls [5]	
19	Measure- ments	`S_NMSP`	Number of measurements at the same location [2] (value range 1 to 9)	
20		`S_MCBIT`	Reserved	
21		`_DMODE`	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
22		`_AMODE`	Alternative mode	

1) All default values = 0 or marked as the range of values a to b

2) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

3) Intermediate positioning, circling around the ball at the equator

4) Measure 3 balls: The same diameter setpoint applies for all three balls (_SETV)

5) Default value for `S_TNVL=1.2`
Correction in WO: Correction is only performed in the WO when the determined distortion is below the `S_TNVL` limit value.

6) If WO "fine" has not been set up in MDs, correction is according to WO "coarse"

7) For measuring variant "Measure three balls", correction in a global basic frame is not possible (`S_KNUM` = 3001 to 3016), as the frame does not have a rotation component.

CYCLE995 measuring cycle parameters

```
PROC CYCLE995 (INT S_MVAR,INT S_KNUM,INT S_PRNUM,REAL S_SETV,REAL S_FA,REAL S_TSA,REAL
S_STA1,REAL S_INCA,REAL S_DZ,REAL S_SETV0,REAL S_SETV1,REAL S_SETV2,REAL S_TUL,REAL
S_TZL,INT S_NMSP,INT S_MCBIT,INT _DMODE,INT _AMODE)
```

Table 4-17 CYCLE995 call parameters 1)

No.	Screen form parameters	Cycle parameters	Meaning	
1		S_MVAR	Measuring variant (default=5)	
			Values:	UNITS: Contour element 5 = Spindle geometry (parallel to the tool axis)
				TENS: Repeat measurement 1 = with repeat measurement
				HUNDREDS: No offset target 0 = measurement only
				THOUSANDS: Measuring strategy 2 = measurement at an angle, align probe in direction of switching
				TEN THOUSANDS: Number of spheres to be measured 0 = measure a sphere
				HUNDRED THOUSANDS: Number of measurement points 1 = 4 measurement points when measuring at an angle (circle the sphere)
				ONE MILLION: Determination of the diameter setpoint of the sphere 0 = No determination of the diameter setpoint of the sphere 1 = Determination of the diameter setpoint of the sphere
2	Selection	S_KNUM	Correction target 0 = 0	
3	Icon+ number	S_PRNUM	Number of the field of the probe parameters (not probe number) (value range 1 to 40)	
4	DM	S_SETV	Diameter of the calibration sphere 4)	
5	DFA	S_FA	Measurement path	
6	TSA	S_TSA	Safe area 5)	
7	alpha 0	S_STA1	Starting angle for measurement at an angle 3)	

No.	Screen form parameters	Cycle parameters	Meaning	
8		S_INCA	Incremental angle for measurement at an angle [2)]	
9	DZ	S_DZ	Clearance 1st measurement P1 to the 2nd measurement P2 at the shaft of the probe	
10		S_SETV0	Setpoint position of the sphere of the 1st axis of the plane (for G17 X) [2)]	
11		S_SETV1	Setpoint position of the sphere of the 2nd axis of the plane (for G17 Y) [2)]	
12		S_SETV2	Setpoint position of the sphere of the 3rd axis of the plane (for G17 Z) [2)]	
13	TUL	S_TUL	Upper tolerance value of the angular deviation	
14	TZL	S_TZL	Zero offset range [1), 4)]	
15	Number	S_NMSP	Number of measurements at the same location [2)] (value range 1 to 9)	
16		S_MCBIT	Reserved[2)]	
17		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
18		_AMODE	Alternative mode	
			Values:	UNITS: Dimensional tolerance yes/no 0 = No 1 = Yes

All default values = 0 or marked as the range of values a to b

1) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE

2) Parameters are currently not used and also not displayed in the input screen.
The parameter incremental angle S_INCAis permanently set to 90 degrees.

3) Value range of starting angle -360 to +360 degrees

4) for dimensional tolerance yes:
If the measured angle is less than the value of the work offset range TZL, then the result parameters for the angle (_OVR[2], _OVR[3]) and deviations (_OVR[7], _OVR[8]) are set to zero.
DisplayTZL is realized using the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE bit25=1.
(enable selected zero offset when measuring angularity, spindle)

5) Parameter TSA refers to the 1st measurement of the calibration sphere.

CYCLE996 measuring cycle parameters

```
PROC CYCLE996(INT S_MVAR,INT S_TC,INT S_PRNUM,REAL S_SETV,REAL S_STA1,REAL S_SETV0,REAL
S_SETV1,REAL S_SETV2,REAL S_SETV3,REAL S_SETV4,REAL S_SETV5,REAL S_TNVL,REAL S_FA,REAL
S_TSA,INT S_NMSP,INT S_MCBIT,INT _DMODE,INT _AMODE)
```

Table 4-18 CYCLE996 call parameters 1)

No.	Screen form parameter	Cycle parameter	Meaning	
1		S_MVAR	Measurement version (default=1)	
			Values:	UNITS: Measuring sequence 0 = Calculate kinematics (selection with: Result display, protocol, change of the swivel data sets, where relevant with operator acknowledgment), see _AMODE 1 = 1st measurement 2 = 2nd measurement 3 = 3rd measurement
				TENS: Reserved 0 = 0
				HUNDREDS: Measurement version for 1st to 3rd measurement 0 = Measurement of the calibration ball paraxial 1 = Measurement of the calibration ball at an angle and no spindle correction 3) 2 = Measurement of the calibration ball and correction of the spindle in the switching direction of the probe 3) 3 = Paraxial measurement, with starting angle 8) 4 = Paraxial measurement, with starting angle, tracking spindle in the switching direction of the probe 8)
				THOUSANDS: Calculate correction target for kinematics 4) 0 = Measuring only. Swivel data sets are calculated, but remain unchanged 1 = calculate swivel data set. Swivel data sets are, if necessary, changed after acknowledgment by the operator 4)
				TEN THOUSANDS: Measuring axis (rotary axis 1 or 2) or vector chain open or closed for calculate kinematics 0 = Vector chain closed (only for calculate kinematics) 1 = Rotary axis 1 (only for 1st to 3rd measurement) 2 = Rotary axis 2 (only for the 1st to 3rd measurement) 5) 3 = Vector chain open (only for calculate kinematics)
				HUNDRED THOUSANDS: Normalizing of rotary axis 1 for calculate kinematics 0 = No scaling rotary axis 1 1 = Scaling in the direction of the 1st axis of the plane (for G17 X) 2 = Scaling in the direction of the 2nd axis of the plane (for G17 Y) 3 = Scaling in the direction of the 3rd axis of the plane (for G17 Z)
				ONE MILLION: Normalizing of rotary axis 2 for calculate kinematics 5) 0 = No scaling rotary axis 2 1 = Scaling in the direction of the 1st axis of the plane (for G17 X) 2 = Scaling in the direction of the 2nd axis of the plane (for G17 Y) 3 = Scaling in the direction of the 3rd axis of the plane (for G17 Z)
				TEN MILLION: Log file 0 = No protocol file 1 = Protocol file with the calculated vectors (tool carrier) and the 1st dynamic 5-axis transformation (TRAORI(1)), if set-up in MDs.
2		S_TC	Number of the swivel data record (tool carrier)	
3	Icon+ number	S_PRNUM	Number of the field of the probe parameters (not probe number) (default=1)	

No.	Screen form parameter	Cycle parameter	Meaning
4		`S_SETV`	Diameter of the calibration ball
5	alpha 0	`S_STA1`	Starting angle for measurement at an angle
6	alpha 0	`S_SETV0`	Position value of rotary axis 1 (if rotary axis is manual or semi-automatic)
7	alpha 1	`S_SETV1`	Position value of rotary axis 2 (if rotary axis is manual or semi-automatic) [6]
8	XN	`S_SETV2`	Position value for normalizing rotary axis 1
9	XN	`S_SETV3`	Position value for normalizing of rotary axis 2 [6]
10	Delta	`S_SETV4`	Tolerance value of the offset vectors I1 to I4
11	Delta	`S_SETV5`	Tolerance value of rotary axis vectors V1 and V2
12	TVL	`S_TNVL`	Limit value of angular segment of the rotary axis (value range 1 to 60 degrees) (default=20) [7]
13	DFA	`S_FA`	Measurement path
14	TSA	`S_TSA`	Safe area
15	Measurements	`S_NMSP`	Number of measurements at the same location [2] (default=1)
16		`S_MCBIT`	Reserved
17		`_DMODE`	Display mode Values: UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
18		`_AMODE`	Alternative mode Values: UNITS: Tolerance check yes/no 0 = No 1 = Yes: Evaluation of the tolerance values of the vectors `S_SETV4`, `S_SETV5` TENS: Acknowledgment by the operator when entering the calculated vectors in the swivel data set [4] 0 = yes: Operator must acknowledge the change 1 = no: calculated vectors are entered immediately (only effective if HUNDREDS and THOUSANDS position = 0) HUNDREDS: Measurement result display [5] 0 = No 1 = Yes THOUSANDS: Measurement result display can be edited 0 = No 1 = Yes, and can be edited (only effective, if the HUNDREDS position = 1)

[1] All default values = 0 or marked as default=x

[2] Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE.

[3] Using this version, for example, for 90 degree positions, the kinematics can be measured at the calibration ball, without colliding with the retaining shaft of the calibration ball. A starting angle `S_STA1` (0 to 360 degrees) can be entered. The incremental angle when circling the ball is equal to 90 degrees.
As feedrate along the circular path, the channel-specific SD55634 $SCS_MEA_FEED_PLANE_VALUE is used

[4] There is an operator prompt with M0 before entering. The vectors are only entered with NC start.
If the measuring program is aborted with RESET no calculated vectors are entered.
Vectors are only entered when the tolerance of the offset vectors has not been exceeded during the calculation.

5) Measurement result display only for the calculated kinematics measuring version.
If the measurement result should also be displayed after the 1st to the 3rd measurement, then this is realized by setting the channel-specific SD55613 $SCS_MEA_RESULT_DISPLAY.

6) Rotary axis 2 only for kinematics with two rotary axes

7) Limit value angular segment of the rotary axis. Value range of `S_TNVL` between 20 and 60 degrees. For values of `S_TNVL` < 20 degrees, inaccuracies can be expected as a result of the measuring inaccuracies in the micrometer range of the probe. If the limit value is violated, then error message 61430 is output – with a display of the minimum limit value.

8) Spindle is tracked in the probe switching direction if SD54760 $SNS_MEA_FUNCTION_MASK_PIECE bit 17 = 1

CYCLE9960 measuring cycle parameters

```
PROC CYCLE9960(INT S_MVAR,STRING[40] S_TNAME,INT S_PRNUM,REAL S_SETV,REAL S_SETV1,REAL
S_START_RA1,REAL S_END_RA1,INT S_CMEA_RA1,REAL S_POS_RA2,REAL S_SETV2,REAL
S_START_RA2,REAL S_END_RA2,INT S_CMEA_RA2,REAL S_POS_RA1,REAL S_SETV4,REAL S_FA,REAL
S_TSA,INT S_NMSP,INT S_DMODE,INT S_AMODE,INT S_KNUM)
```

Table 4-19 CYCLE9960 call parameters 1)

<table>
<tr><th>No.</th><th>Screen form parameter</th><th>Cycle parameter</th><th colspan="2">Meaning</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">Selection</td><td rowspan="6">S_MVAR</td><td colspan="2">Measurement version (default=1)</td></tr>
<tr><td rowspan="5">Values:</td><td>UNITS: Measuring variant
0 = measure and calculate kinematics (select with: Result display, protocol, change of swivel data set (see thousands S_MVAR)
1 = measure reference head
2 = adapt head to reference head
3 = measure and calculate interpolation points (E996)</td></tr>
<tr><td>TENS: Reserved</td></tr>
<tr><td>HUNDREDS: Ball measuring variant
2 = measurement of the calibration ball and tracking of the spindle in the switching direction of the probe
4 = paraxial measurement, with starting angle, tracking spindle in the switching direction of the probe 2)</td></tr>
<tr><td>THOUSANDS: Calculate correction target for kinematics 3)
0 = measure and calculate. Data sets are calculated and remain unchanged
1 = calculated data sets may be changed after operator acknowledgment
2 = previously measured kinematics are calculated and, if applicable, changed after operator acknowledgment</td></tr>
<tr><td>TENTHOUSANDS: Measuring axis (rotary axis 1 or 2)
1 = measure and calculate all existing rotary axes
4 = measure and calculate only rotary axis 1
5 = measure and calculate only rotary axis 2</td></tr>
<tr><td>2</td><td></td><td>S_TNAME</td><td colspan="2">Name of the transformation (swivel data set or transformation based on kinematic chains)</td></tr>
<tr><td>3</td><td>Icon+ number</td><td>S_PRNUM</td><td colspan="2">Number of the field of the probe parameters (not probe number)
(default=1)</td></tr>
<tr><td>4</td><td></td><td>S_SETV</td><td colspan="2">Diameter of the calibration ball</td></tr>
</table>

No.	Screen form parameter	Cycle parameter	Meaning	
5	alpha 1	S_SETV1	Starting angle for measuring at an angle for the 1st rotary axis	
6			S_START_RA1: Starting angle of the 1st rotary axis	
7			S_END_RA1: Final angle of 1st rotary axis	
8			S_CMEA_RA1: Number of measurements of the 1st rotary axis, 3 for measuring and calculating kinematics. Up to 12 measurements are possible for interpolation points.	
9			S_POS_RA2: Position of the 2nd rotary axis while measuring the 1st rotary axis [4]	
10	alpha 2	S_SETV2	Starting angle for measuring at an angle for the 2nd rotary axis [4]	
11			S_START_RA2 : Starting angle of the 2nd rotary axis [4]	
12			S_END_RA2 : Final angle of 2nd rotary axis [4]	
13			S_CMEA_RA2 : Number of measurements of the 2nd rotary axis, 3 for measuring and calculating kinematics. Up to 12 measurements are possible for interpolation points.[4]	
14			S_POS_RA1 : Position of the 1st rotary axis while measuring the 2nd rotary axis	
15	Delta	S_SETV4	Tolerance value of offset vectors	
16	DFA	S_FA	Measurement path	
17	TSA	S_TSA	Safe area	
18	Number	S_NMSP	Number of measurements at the same location [5] (default=1)	
19		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
20		_AMODE	Alternative mode	
			Values:	ONES: Tolerance check yes/no 0 = no 1 = yes: Evaluation of the tolerance values of the vectors -> S_SETV4, S_SETV5
				TENS: Automatic calibration 0 = without automatic calibration 1 = automatic calibration
				HUNDREDS: Automatic starting angle 0 = set starting angle is used 1 = automatic calculation of the starting angle, for each measuring point
21		S_KNUM	Number of the WO to be corrected for reference head measurement	
			Values	UNITS: The correction is always applied to the coarse offset, the fine offset is deleted
				TENS: 1 ... 99 number of the adjustable WO (1=G54)
				THOUSANDS: 9 correction active, adjustable WO

1) All default values = 0 or marked as default = xx

2) Spindle is tracked in the switching direction of the probe if SD54760 $SNS_MEA_FUNCTION_MASK_PIECE Bit17=1

3) There is an operator prompt with M0 before entering. The vectors cannot be entered until the NC has been started. If the measuring program is interrupted with RESET, no calculated vectors are entered. Vectors are entered only if the tolerance of the offset vectors is not exceeded in the calculation.

4) Rotary axis 2 only for kinematics with 2 rotary axes

5) Display depends on the general SD54760 $SNS_MEA_FUNCTION_MASK_PIECE.

CYCLE982 measuring cycle parameters

```
PROC CYCLE982(INT S_MVAR,INT S_KNUM,INT S_PRNUM,INT S_MA,INT S_MD,REAL S_ID,REAL S_FA,REAL
S_TSA,REAL S_VMS,REAL S_STA1,REAL S_CORA,REAL S_TZL,REAL S_TDIF,INT S_NMSP,INT S_EVNUM,INT
S_MCBIT,INT _DMODE,INT _AMODE)
```

Table 4-20 CYCLE982 call parameters [1)]

No.	Screen form parameter	Cycle parameter	Meaning	
1		S_MVAR	Measuring variant	
			Values:	UNITS: Calibration/measurement 0 = Calibrate tool probe 1 = Single tool measurement [3)] 2 = Multiple tool measurement, determine lengths and tool radius (for milling tools)
				TENS: Calibration or measurement in the MCS or WCS 0 = Machine-related [4)] 1 = Workpiece-related
				HUNDREDS: Measurement with or without reversal for milling tools 0 = Measurement without reversal 1 = Measurement with reversal
				THOUSANDS: Correction target for milling tools 0 = Determine length or length and radius (see S_MVAR 1st position) 1 = Determine radius if S_MVAR 1st position = 1 2 = Determine length and radius (face side) if S_MVAR 1st position = 1 or 2 3 = Determine side milling cutter upper cutting edge (rear side) and length and radius [5)]
				TEN THOUSANDS: Position of the milling tool or the drill 0 = Axial position of the milling tool or drill, radius in 2nd axis of the plane (for G18 X) [7)] 1 = Radial position of the milling tool or the drill, radius in 1st axis of the plane (for G18 Z) [7)]
				HUNDRED THOUSANDS: Incremental calibration or measurement 0 = No specification 1 = Incremental calibration or measurement
				ONE MILLION: Position spindle at starting angle S_STA1 (only for measurement of milling tools) 0 = Spindle is not positioned 1 = Spindle is positioned at the starting angle S_STA1
2	Selection	S_KNUM	Offset variant [2)]	
			Values:	UNITS: Tool offset 0 = No specification (tool offset in geometry) 1 = Tool offset in wear
3	Icon+ number	S_PRNUM	Number of the field of the probe parameters (not probe number) (default=1)	

No.	Screen form parameter	Cycle parameter	Meaning	
4	X0	S_MA	Measuring axis	
			Values:	1 = 1st axis of the plane (for G18 Z) 2 = 2nd axis of the plane (for G18 X)
5	+-	S_MD	Measuring direction	
			Values:	0 = No selection (measuring direction is determined from actual value) 1 = Positive 2 = Negative
6	Z2	S_ID	Offset	
7	DFA	S_FA	Measurement path	
8	TSA	S_TSA	Safe area	
9	VMS	S_VMS	Variable measuring velocity for calibration [2)]	
10	alpha1	S_STA1	Starting angle when measuring milling tools	
11	alpha2	S_CORA	Offset angle when measuring milling tools with reversal [8)]	
12	TZL	S_TZL	Work offset when measuring milling tools. When calibrating S_TZL = 0	
13	DIF	S_TDIF	Dimensional difference check	
14	Measurements	S_NMSP	Number of measurements at the same location [2)] (default=1)	
15	EVN	S_EVNUM	Number of the empirical mean value memory [2), 9)]	
16		S_MCBIT	Reserved	
17		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)
				TENS: Cutting edge position for turning and milling tools (only for display in the input screens 1 to 9)
				HUNDREDS: Tool type 0 = Turning tool 1 = Milling tool 2 = Drill
				THOUSANDS: The approach strategy with reference to the tool probe 0 = PLUS [X/Z]; X if tool position axial, Z if tool position radial 1 = MINUS [X/Z]; X if tool position axial, Z if tool position radial
18		_AMODE	Alternative mode	
			Values:	UNITS: Reserved
				TENS: Reserved
				HUNDREDS: Reserved
				THOUSANDS: approach starting position after measurement for calibration and single measurement (see S_MVAR - UNITS) 0 = Tool is located, offset by DFA with respect to the probe edge 1 = Approach starting position

1) All default values = 0 or marked as default=x

2) Display depends on the general SD54762 _MEA_FUNCTION_MASK_TOOL

3) Measure turning or milling tool or drill. Measuring axis in parameter `S_MA`
Specification for turning tools via cutting edge position 1...8, for milling tools via HUNDREDS to THOUSANDS position in parameter `S_MVAR`.

4) Measurement and calibration are performed in the basic coordinate system (MCS for kinematic transformation switched off).

5) Not for incremental measuring

6) Only for multiple measurements `S_MVAR=x2x02` or `x3x02` (example, disk-type or groove milling tools)

7) If the channel-specific SD42950 $SC_TOOL_LENGTH_TYPE = 2, then the tool length components are assigned just the same as for turning tools

8) Only for measurement with reversal `S_MVAR=xx1x1`

9) Empirical value generation
Value range of the empirical value memory: 1 to 20 numbers (n) of the empirical value memory, see channel-specific SD55623 $SCS_MEA_EMPIRIC_VALUE[n-1].

CYCLE971 measuring cycle parameters

```
PROC CYCLE971(INT S_MVAR,INT S_KNUM,INT S_PRNUM,INT S_MA,INT S_MD,REAL S_ID,REAL S_FA,REAL
S_TSA,REAL S_VMS,REAL S_TZL,REAL S_TDIF,INT S_NMSP,REAL S_F1,REAL S_S1,REAL S_F2,REAL
S_S2,REAL S_F3,REAL S_S3,INT S_EVNUM,INT S_MCBIT,INT _DMODE,INT _AMODE)
```

Table 4-21 CYCLE971 call parameters [1)]

No.	Screen form parameter	Cycle parameter	Meaning	
1		S_MVAR	Measuring variant	
			Values:	UNITS: 0 = Calibrate tool probe 1 = Measure tool with stationary spindle (length or radius) 2 = Measure tool with rotating spindle (length or radius), see parameters S_F1 to S_S4
				TENS: Measurement in the machine coordinate system or workpiece coordinate system 0 = Measurement in MCS (machine-related), measure tool or calibrate tool probe 1 = Measurement in WCS (workpiece-related), measure tool or calibrate tool probe
				HUNDREDS: Individually check teeth 0 = No 1 = Yes
				THOUSANDS: 0 = 0
				TEN THOUSANDS: Incremental calibration or measurement 0 = No specification 1 = Incremental calibration or measurement
				HUNDRED THOUSANDS: Calibrate tool probe automatically 0 = Do not calibrate tool probe automatically 1 = Calibrate tool probe automatically
				ONE MILLION: Calibrating in the plane with spindle reversal 0 = Calibrating in the plane without spindle reversal 1 = Calibrating in the plane with spindle reversal
2	Selection	S_KNUM	Offset variant [2)]	
			Values:	UNITS: Tool offset 0 = No specification (tool offset in geometry) 1 = Tool offset in wear
3	Icon+ number	S_PRNUM	Number of the field of the probe parameters (not probe number)	
4	X0	S_MA	Measuring axis, offset axis [4)]	
			Values:	UNITS: Number of the measuring axis 1 = 1st axis of the plane (for G17 X) 2 = 2nd axis of the plane (for G17 Y) 3 = 3rd axis of the plane (for G17 Z)
				TENS: 0 = 0
				HUNDREDS: Number of the offset axis 0 = Not an offset axis 1 = 1. axis of the plane (for G17 X) 2 = 2nd axis of the plane (for G17 Y)

No.	Screen form parameter	Cycle parameter	Meaning	
5	+-	S_MD	Measuring direction	
			Values:	0 = No selection (measuring direction is determined from actual value) 1 = Positive 2 = Negative
6	V	S_ID	Offset	
			Values:	0 = For tools without offset >0 = • Calibration: The offset is applied to the 3rd axis of the plane (for G17 Z) if the diameter of the calibration tool is greater than the upper diameter of the probe. The tool is offset by the tool radius from the center of the probe, minus the value of S_ID. The offset axis is also specified in S_MA . • Measure: With multiple cutting edges, the offset of tool length and the highest point of the cutting edge must be specified for radius measurement or the offset of tool radius to the highest point of the cutting edge must be specified when measuring the length.
7	DFA	S_FA	Measurement path	
8	TSA	S_TSA	Safe area	
9	VMS	S_VMS	Variable measuring velocity for calibration 2)	
10	TZL	S_TZL	Work offset (only for tool measurement)	
11	DIF	S_TDIF	Dimensional difference check for tool measurement (S_MVAR=xx1 or S_MVAR=xx2)	
12	Measurements	S_NMSP	Number of measurements at the same location 2)	
13	F1	S_F1	1st feedrate for contact with rotating spindle 2)	
14	S1	S_S1	1st speed for contact with rotating spindle 2)	
15	F2	S_F2	2nd feedrate for contact with rotating spindle 2)	
16	S2	S_S2	2nd speed for contact with rotating spindle 2)	
17	F3	S_F3	2nd feedrate for contact with rotating spindle 3)	
18	S3	S_S3	2nd speed for contact with rotating spindle 3)	
19	EVN	S_EVNUM	Number of the empirical value memory 2)	
20		S_MCBIT	Screen form of the _CBITs or _CHBITs	
21		_DMODE	Display mode	
			Values:	UNITS: Machining plane G17/G18/G19 0 = Compatibility, the plane active before the cycle call remains active 1 = G17 (only active in the cycle) 2 = G18 (only active in the cycle) 3 = G19 (only active in the cycle)

No.	Screen form parameter	Cycle parameter	Meaning	
22		_AMODE	Alternative mode	
			Values:	UNITS: Measuring the tool offset for radius 1 = No 2 = Yes
				TENS: Direction of the tool offset when measuring the radius in the 3rd axis of the plane (for G17 Z) 1 = Positive 2 = Negative
				HUNDREDS: Tool offset when measuring the length or when calibrating the probe in the 3rd axis 0 = Compatibility, auto 1 = No 2 = Yes
				THOUSANDS: Direction of the tool offset when measuring the length in the offset axis (see S_MA HUNDREDS) 1 = Positive 2 = Negative

1) All default values = 0 or marked as default=x

2) Display depends on the general SD54762 $MEA_FUNCTION_MASK_TOOL

3) Only for offset in tool and dimensional tolerance "Yes", otherwise parameter = 0

4) For automatic measurement (`S_MVAR=1x00xx`), no display of measuring axis, offset axis ⇒ `S_MA=0`.

CYCLE150 measuring cycle parameters

```
PROC CYCLE150(INT S_PICT,INT S_PROT,STRING[160] S_PATH) SAVE
ACTBLOCNO DISPLOF
```

Table 4-22 CYCLE150 call parameters

No.	Screen parameters	Cycle parameters	Meaning	
1	Measuring result screen	S_PICT	Select result display (default = 0)	
			Values:	UNITS: 0 = Measuring result screen OFF 1 = Measuring result screen ON
				Tens: Select display mode (values as for SD55613 $SCS_MEA_RESULT_DISPLAY) 1 = Display measuring result screen - automatically deselect after 8 s 3 = Display measuring result screen - acknowledge using NC Start 4 = Display measuring result screen - only for alarms (61303 ... 61306)
2		S_PROT	Select logging (default = 0)	

No.	Screen parameters	Cycle parameters	Meaning	
	Log		Values:	UNITS: Select protocol off / on / last measurement 0 = Log OFF 1 = Log ON 2 = Log last measurement
	Log type			TENS: Select log type 0 = Standard log 1 = User log (can be freely defined)
	Log format			HUNDREDS: Select log format 0 = Text format 1 = Table format (for import to Excel)
	Log data			THOUSANDS: Rewrite or attach selection 0 = New 1 = Attach
	Log archive			TEN THOUSANDS: Select log archive 0 = As part program 1 = Directory
3		`S_PATH`	Path for the log file corresponding to the log archive selection (complete path name or only file name, e.g.: "//NC:/WKS.DIR/NAME.WPD or "MESSPROTOKOLL.TXT"	

Tables 5

5.1 Operations

Note

Cycles

The list of operations contains all cycles, which occur in the NC program (G code), i.e. can be programmed in the program editor using masks - or must be programmed for loops without programming support. Cycles, which for reasons of compatibility, are still available in the control, however can no longer be edited using the SINUMERIK Operate program editor ("compatibility cycles") are not taken into account.

5.1.1 Operations A ... C

Operation	Type [1)]	Meaning	W [2)]	TP [3)]	SA [4)]	Description see [5)]
1) 2) 3) 4) 5) for explanations, see legend (Page 1109).						
:	O	NC main block number, jump label termination, chaining operator		+		PM-NC
*	O	Operator for multiplication		+		PM-NC
+	O	Operator for addition		+		PM-NC
-	O	Operator for subtraction		+		PM-NC
<	O	Comparison operator, less than		+		PM-NC
<<	O	Chaining operator for strings		+		PM-NC
<=	O	Comparison operator, less than or equal to		+		PM-NC
=	O	Assignment operator		+		PM-NC
>=	O	Comparison operator, greater than or equal to		+		PM-NC
/	O	Operator for division		+		PM-NC
/0 ... ... /7		block is skipped (1st skip level) ... ... block is skipped (8th skip level)		+		PM-NC
A	A	Axis name	m/s	+		PM-NC
A2	A	Tool orientation: RPY or Euler angle	s	+		PM-NC
A3	A	Tool orientation: 1st component of the direction vector	s	+		PM-NC
A4	A	Tool orientation: 1st component of the surface normal vector at start of block	s	+		PM-NC
A5	A	Tool orientation: 1st component of the surface normal vector at end of block	s	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
A6	A	Tool orientation: 1st component of the direction vector for taper's axis of rotation	s	+		PM-NC
A7	A	Tool orientation: 1st vector component for intermediate orientation on peripheral surface of taper	s	+		PM-NC
ABS	F	Absolute value (amount)		+	+	PM-NC
AC	K	Absolute dimensions of coordinates/positions	s	+		PM-NC
ACC	K	Effect of current axial acceleration	m	+	+	PM-NC
ACCLIMA	K	Effect of current maximum axial acceleration	m	+	+	PM-NC
ACN	K	Absolute dimensions for rotary axes, approach position in negative direction	s	+		PM-NC
ACOS	F	Arc cosine (trigon. function)		+	+	PM-NC
ACP	K	Absolute dimensions for rotary axes, approach position in positive direction	s	+		PM-NC
ACTBLOCNO	P	Output of current block number of an alarm block, even if "current block display suppressed" (DISPLOF) is active!		+		PM-NC
ADDFRAME	F	Inclusion and possible activation of a measured frame		+	-	PM-NC, FM-B
ADIS	A	Smoothing clearance for path functions G1, G2, G3, ...	m	+		PM-NC
ADISPOS	A	Smoothing clearance for rapid traverse G0	m	+		PM-NC
ADISPOSA	P	Size of the tolerance window for IPOBRKA	m	+	+	PM-NC
AFISOF	P	Deactivate automatic filter chain switchover	m	+	-	PM-NC
AFISON	P	Activate automatic filter chain switchover	m	+	-	PM-NC
ALF	A	Rapid lift angle	m	+		PM-NC
AMIRROR	G	Programmable mirroring	s	+		PM-NC
AND	K	Logical AND		+		PM-NC
ANG	A	Contour angle	s	+		PM-NC
AP	A	Polar angle	m/s	+		PM-NC
APR	K	Read/show access protection		+		PM-NC
APRB	K	Read access right, OPI		+		PM-NC
APRP	K	Read access right, part program		+		PM-NC
APW	K	Write access protection		+		PM-NC
APWB	K	Write access right, OPI		+		PM-NC
APWP	K	Write access right, part program		+		PM-NC
APX	K	Definition of the access right for executing the specified language element		+		PM-NC
AR	A	Opening angle	m/s	+		PM-NC
AROT	G	Programmable rotation	s	+		PM-NC
AROTS	G	Programmable frame rotations with solid angles	s	+		PM-NC
AS	K	Macro definition		+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
ASCALE	G	Programmable scaling	s	+		PM-NC
ASIN	F	Arithmetic function, arc sine		+	+	PM-NC
ASPLINE	G	Akima spline	m	+		PM-NC
ATAN2	F	Arc tangent 2		+	+	PM-NC
ATOL	A	Axis-specific tolerance for compressor functions, orientation smoothing and smoothing types	m	+		PM-NC
ATRANS	G	Additive programmable work offset	s	+		PM-NC
AUXFUDEL	P	Delete auxiliary function channel-specifically from the global list		+	-	FM-B
AUXFUDELG	P	Delete all auxiliary functions of an auxiliary function group channel-specifically from the global list		+	-	FM-B
AUXFUMSEQ	P	Determine output sequence of M auxiliary functions		+	-	FM-B
AUXFUSYNC	P	Generate a complete part program block for the channel-specific SERUPRO end ASUB as string from the global list of auxiliary functions		+	-	FM-B
AX	K	Variable axis identifier	m/s	+		PM-NC
AXCTSWE	P	Rotate axis container		+	-	PM-NC
AXCTSWEC	P	Cancel enable for axis container rotation		+	+	PM-NC
AXCTSWED	P	Rotating axis container (command variant for commissioning!)		+	-	PM-NC
AXIS	K	Axis identifier, axis address		+		PM-NC
AXNAME	F	Converts input string into axis identifier		+	-	PM-NC
AXSTRING	F	Converts string spindle number		+	-	PM-NC
AXTOCHAN	P	Request axis for a specific channel. Possible from NC program and synchronized action.		+	+	PM-NC
AXTOSPI	F	Converts axis identifier into a spindle index		+	-	PM-NC
B	A	Axis name	m/s	+		PM-NC
B2	A	Tool orientation: RPY or Euler angle	s	+		PM-NC
B3	A	Tool orientation: component of the direction vector	s	+		PM-NC
B4	A	Tool orientation: 2nd component of the surface normal vector at start of block	s	+		PM-NC
B5	A	Tool orientation: 2nd component of the surface normal vector at end of block	s	+		PM-NC
B6	A	Tool orientation: 2nd component of the direction vector for taper's axis of rotation	s	+		PM-NC
B7	A	Tool orientation: 2nd vector component for intermediate orientation on peripheral surface of taper	s	+		PM-NC
B_AND	O	Bit-by-bit AND		+		PM-NC
B_OR	O	Bit-by-bit OR		+		PM-NC
B_NOT	O	Bit-by-bit negation		+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
B_XOR	O	Bit-by-bit exclusive OR		+		PM-NC
BAUTO	G	Definition of the first spline section by means of the next 3 points	m	+		PM-NC
BLOCK	K	Together with the keyword TO defines the program part to be processed in an indirect subprogram call		+		PM-NC
BLSYNC	K	Processing of interrupt routine is only to start with the next block change		+		PM-NC
BNAT 6)	G	Natural transition to first spline block	m	+		PM-NC
BOOL	K	Data type: Boolean value TRUE/FALSE or 1/0		+		PM-NC
BOUND	F	Tests whether the value falls within the defined value range. If the values are equal, the test value is returned.		+	+	PM-NC
BRISK 6)	G	Fast non-smoothed path acceleration	m	+		PM-NC
BRISKA	P	Switch on brisk path acceleration for the programmed axes		+	-	PM-NC
BSPLINE	G	B spline	m	+		PM-NC
BTAN	G	Tangential transition to first spline block	m	+		PM-NC
C	A	Axis name	m/s	+		PM-NC
C2	A	Tool orientation: RPY or Euler angle	s	+		PM-NC
C3	A	Tool orientation: 3rd component of the direction vector	s	+		PM-NC
C4	A	Tool orientation: 3rd component of the surface normal vector at start of block	s	+		PM-NC
C5	A	Tool orientation: 3rd component of the surface normal vector at end of block	s	+		PM-NC
C6	A	Tool orientation: 3rd component of the direction vector for taper's axis of rotation	s	+		PM-NC
C7	A	Tool orientation: 3rd vector component for intermediate orientation on peripheral surface of taper	s	+		PM-NC
CAC	K	Absolute position approach		+		PM-NC
CACN	K	Absolute approach of the value listed in the table in negative direction		+		PM-NC
CACP	K	Absolute approach of the value listed in the table in positive direction		+		PM-NC
CADAPTOF	P	Deactivate load adjustment		+	-	PM-NC
CADAPTON	P	Activate load adjustment		+	-	PM-NC
CALCDAT	F	Calculates radius and center point of circle from 3 or 4 points		+	-	PM-NC
CALCFIR	P	Adapting the FIR jerk filter to the dynamic response mode		+	-	PM-NC
CALCPOSI	F	Checking for protection area violation, working area limitation and software limits		+	-	PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
CALCTRAVAR	F	Calculating the angle for aligning the tool for TRAINT		+	-	PM-NC
CALL	K	Indirect subprogram call		+		PM-NC
CALLPATH	P	Programmable search path for subprogram calls		+	-	PM-NC
CANCEL	P	Cancel modal synchronized action		+	-	FM-SA
CANCELSUB	P	Cancel current subprogram level		+	+	FM-SA
CASE	K	Conditional program branch		+		PM-NC
CDC	K	Direct approach of a position		+		PM-NC
CDOF 6)	G	Switch off collision detection	m	+		PM-NC
CDOF2	G	Switch off collision detection during 3D circumferential milling	m	+		PM-NC
CDON	G	Activate collision detection	m	+		PM-NC
CFC 6)	G	Constant feedrate on contour	m	+		PM-NC
CFIN	G	Constant feedrate for internal radius only, not for external radius	m	+		PM-NC
CFINE	F	Assignment of fine offset to a FRAME variable		+	-	PM-NC
CFTCP	G	Constant feedrate in tool center point (center point path)	m	+		PM-NC
CHAN	K	Specify validity range for data		+		PM-NC
CHANDATA	P	Set channel number for channel data access		+	-	PM-NC
CHAR	K	Data type: ASCII character		+		PM-NC
CHF	A	Chamfer; value = length of chamfer	s	+		PM-NC
CHKDM	F	Uniqueness check within a magazine		+	-	FM-TM
CHKDNO	F	Check for unique D numbers		+	-	PM-NC
CHR	A	Chamfer; value = length of chamfer in direction of movement		+		PM-NC
CIC	K	Approach position by increments		+		PM-NC
CIP	G	Circular interpolation through intermediate point	m	+		PM-NC
CLEARM	P	Reset one/several markers for channel coordination		+	+	PM-NC
CLRINT	P	Deselect interrupt		+	-	PM-NC
CMIRROR	F	Mirror on a coordinate axis		+	-	PM-NC
COARSEA	K	Motion end when "Exact stop coarse" reached	m	+		PM-NC
COLLPAIR	F	Check whether part of a collision pair		+		PM-NC
COMPCAD	G	Activate the compressor function COMPCAD	m	+		PM-NC
COMPCURV	G	Activate the compressor function COMPCURV	m	+		PM-NC
COMPLETE		Control instruction for reading and writing data		+		PM-NC
COMPOF 6)	G	Deactivate NC block compression	m	+		PM-NC
COMPON	G	Activate the compressor function COMPON	m	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
COMPPATH	G	Activate compressor function COMPPATH	m	+		PM-NC
COMPSURF	G	Activate the compressor function COMPSURF	m	+		PM-NC
CONTDCON	P	Activate tabular contour decoding		+	-	PM-NC
CONTPRON	P	Activate reference preprocessing		+	-	PM-NC
CORROF	P	All active overlaid movements are deselected.		+	-	PM-NC
CORRTC	F	Modify offset vectors or direction vectors of orientable tool carriers according to machine measurement.		+	-	PM-NC
CORRTRAFO	F	Modifying offset vectors or direction vectors for the orientation axes in the kinematic model of the machine		+	-	PM-NC
COS	F	Cosine (trigon. function)		+	+	PM-NC
COUPDEF	P	Definition ELG group/synchronous spindle group		+	-	PM-NC
COUPDEL	P	Delete ELG group		+	-	PM-NC
COUPOF	P	Deactivate ELG group / synchronous spindle pair		+	-	PM-NC
COUPOFS	P	Deactivate ELG group/synchronous spindle pair with stop of following spindle		+	-	PM-NC
COUPON	P	Activate ELG group / synchronous spindle pair		+	-	PM-NC
COUPONC	P	Transfer activation of ELG group/synchronous spindle pair with previous programming		+	-	PM-NC
COUPRES	P	Reset ELG group		+	-	PM-NC
CP 6)	G	Path motion	m	+		PM-NC
CPBC	K	Generic coupling: Block change criterion		+	+	FM-A
CPDEF	K	Generic coupling: Creating a coupling module		+	+	FM-A
CPDEL	K	Generic coupling: Deletion of a coupling mod-ule		+	+	FM-A
CPFMOF	K	Generic coupling: Behavior of the following axis at complete switch-off		+	+	FM-A
CPFMON	K	Generic coupling: Behavior of the following axis when switching on		+	+	FM-A
CPFMSON	K	Generic coupling: Synchronization mode		+	+	FM-A
CPFPOS	K	Generic coupling: Synchronized position of the following axis		+	+	FM-A
CPFRS	K	Generic coupling: Coordinate reference system		+	+	FM-A
CPLA	K	Generic coupling: Definition of a leading axis		+	-	FM-A
CPLCTID	K	Generic coupling: Number of the curve table		+	+	FM-A
CPLDEF	K	Generic coupling: Definition of a leading axis and creation of a coupling module		+	+	FM-A
CPLDEL	K	Generic coupling: Deleting a leading axis of a coupling module		+	+	FM-A
CPLDEN	K	Generic coupling: Denominator of the cou-pling factor		+	+	FM-A

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
CPLINSC	K	Generic coupling: Scaling factor of the input value of a leading axis		+	+	FM-A
CPLINTR	K	Generic coupling: Offset value of the input value of a leading axis		+	+	FM-A
CPLNUM	K	Generic coupling: Numerator of the coupling factor		+	+	FM-A
CPLOF	K	Generic coupling: Switching off a leading axis of a coupling module		+	+	FM-A
CPLON	K	Generic coupling: Switching on a leading axis of a coupling module		+	+	FM-A
CPLOUTSC	K	Generic coupling: Scaling factor for the output value of a coupling		+	+	FM-A
CPLOUTTR	K	Generic coupling: Offset value for the output value of a coupling		+	+	FM-A
CPLPOS	K	Generic coupling: Synchronized position of the leading axis		+	+	FM-A
CPLSETVAL	K	Generic coupling: Coupling reference		+	+	FM-A
CPMALARM	K	Generic coupling: Suppression of special coupling-related alarm outputs		+	+	FM-A
CPMBRAKE	K	Generic coupling: Response of the following axis to certain stop signals and stop commands		+	-	FM-A
CPMPRT	K	Generic coupling: Coupling response at part program start under block search run via program test		+	+	FM-A
CPMRESET	K	Generic coupling: Coupling behavior for RESET		+	+	FM-A
CPMSTART	K	Generic coupling: Coupling behavior at part program start		+	+	FM-A
CPMVDI	K	Generic coupling: Response of the following axis to certain NC/PLC interface signals		+	+	FM-A
CPOF	K	Generic coupling: Switching off a coupling module		+	+	FM-A
CPON	K	Generic coupling: Switching on a coupling module		+	+	FM-A
CPRECOF 6)	G	Deactivate programmable contour accuracy	m	+		PM-NC
CPRECON	G	Activate programmable contour accuracy	m	+		PM-NC
CPRES	K	Generic coupling: Activates the configured data of the synchronous spindle coupling		+	-	FM-A
CPROT	P	Activate / deactivate channel-specific protection area		+	-	PM-NC
CPROTDEF	P	Definition of a channel-specific protection area		+	-	PM-NC
CPSETTYPE	K	Generic coupling: Coupling type		+	+	FM-A
CPSYNCOP	K	Generic coupling: Threshold value of position synchronism "Coarse"		+	+	FM-A
CPSYNCOP2	K	Generic coupling: Threshold value of position synchronism "Coarse" 2		+	+	FM-A

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
CPSYNCOV	K	Generic coupling: Threshold value of velocity synchronism "Coarse"		+	+	FM-A
CPSYNFIP	K	Generic coupling: Threshold value of position synchronism "Fine"		+	+	FM-A
CPSYNFIP2	K	Generic coupling: Threshold value of position synchronism "Fine" 2		+	+	FM-A
CPSYNFIV	K	Generic coupling: Threshold value of velocity synchronism "Fine"		+	+	FM-A
CR	A	Circle radius	s	+		PM-NC
CROT	F	Rotation of the current coordinate system		+	-	PM-NC
CROTS	F	Programmable frame rotations with solid angles (rotation in the specified axes)	s	+	-	PM-NC
CRPL	F	Frame rotation in any plane		+	-	FM-B
CSCALE	F	Scale factor for multiple axes		+	-	PM-NC
CSPLINE	F	Cubic spline	m	+		PM-NC
CT	G	Circle with tangential transition	m	+		PM-NC
CTAB	F	Define following axis position according to leading axis position from curve table		+	+	PM-NC
CTABDEF	P	Activate table definition		+	-	PM-NC
CTABDEL	P	Clear curve table		+	-	PM-NC
CTABEND	P	Deactivate table definition		+	-	PM-NC
CTABEXISTS	F	Checks the curve table with number n		+	+	PM-NC
CTABFNO	F	Number of curve tables still possible in the memory		+	+	PM-NC
CTABFPOL	F	Number of polynomials still possible in the memory		+	+	PM-NC
CTABFSEG	F	Number of curve segments still possible in the memory		+	+	PM-NC
CTABID	F	Returns table number of the n-th curve table		+	+	PM-NC
CTABINV	F	Define leading axis position according to following axis position from curve table		+	+	PM-NC
CTABISLOCK	F	Returns the lock state of the curve table with number n		+	+	PM-NC
CTABLOCK	P	Delete and overwrite, lock		+	+	PM-NC
CTABMEMTYP	F	Returns the memory in which curve table number n is created.		+	+	PM-NC
CTABMPOL	F	Max. number of polynomials still possible in the memory		+	+	PM-NC
CTABMSEG	F	Max. number of curve segments still possible in the memory		+	+	PM-NC
CTABNO	F	Number of defined curve tables in SRAM or DRAM		+	+	FM-A
CTABNOMEM	F	Number of defined curve tables in SRAM or DRAM		+	+	PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) for explanations, see legend (Page 1109).						
CTABPERIOD	F	Returns the table periodicity of curve table number n		+	+	PM-NC
CTABPOL	F	Number of polynomials already used in the memory		+	+	PM-NC
CTABPOLID	F	Number of the curve polynomials used by the curve table with number n		+	+	PM-NC
CTABSEG	F	Number of curve segments already used in the memory		+	+	PM-NC
CTABSEGID	F	Number of the curve segments used by the curve table with number n		+	+	PM-NC
CTABSEV	F	Returns the end value of the following axis of a segment of the curve table		+	+	PM-NC
CTABSSV	F	Returns the starting value of the following axis of a segment of the curve table		+	+	PM-NC
CTABTEP	F	Returns the value of the leading axis at the end of the curve table		+	+	PM-NC
CTABTEV	F	Returns the value of the following axis at the end of the curve table		+	+	PM-NC
CTABTMAX	F	Returns the maximum value of the following axis of the curve table		+	+	PM-NC
CTABTMIN	F	Returns the minimum value of the following axis of the curve table		+	+	PM-NC
CTABTSP	F	Returns the value of the leading axis at the start of the curve table		+	+	PM-NC
CTABTSV	F	Returns the value of the following axis at the start of the curve table		+	+	PM-NC
CTABUNLOCK	P	Revoke delete and overwrite lock		+	+	PM-NC
CTOL	A	Contour tolerance for compressor functions, orientation smoothing and smoothing types	m	+	-	PM-NC
CTOLG0	A	Contour tolerance for rapid traverse movements	m	+	-	PM-NC
CTRANS	F	Work offset for multiple axes		+	-	PM-NC
CUT2D 6)	G	2D TRC	m	+		PM-NC
CUT2DD	G	2½ D TRC in relation to the differential tool	m	+		PM-NC
CUT2DF	G	2D TRC relative to the current frame (inclined plane)	m	+		PM-NC
CUT2DFD	G	2½D TRC in relation to a differential tool relative to the current frame (inclined plane)	m	+		PM-NC
CUT3DC	G	3D TRC for circumferential milling	m	+		PM-NC
CUT3DCC	G	3D TRC for circumferential milling taking into account a limitation surface with 3D radius compensation: Contour on the machining surface	m	+		PM-NC

Operation	Type [1]	Meaning	W [2]	TP [3]	SA [4]	Description see [5]
[1) 2) 3) 4) 5)] **for explanations, see legend (Page 1109).**						
CUT3DCCD	G	3D TRC for circumferential milling taking into account a limitation surface with differential tool on the tool center-point path: Infeed to the limitation surface	m	+		PM-NC
CUT3DCD	G	3D TRC in relation to a differential tool for circumferential milling	m	+		PM-NC
CUT3DF	G	3D TRC for face milling with change in orientation	m	+		PM-NC
CUT3DFD	G	3D TRC in relation to a differential tool for face milling with change in orientation	m	+		PM-NC
CUT3DFF	G	3D TRC for face milling with constant orientation. The tool orientation is the direction defined by G17 - G19 and, in some cases, rotated by a frame.	m	+		PM-NC
CUT3DFS	G	3D TRC for face milling with constant orientation. The tool orientation is defined by G17 - G19 and is not influenced by frames.	m	+		PM-NC
CUTCONOF [6]	G	Deactivate tool radius compensation	m	+		PM-NC
CUTCONON	G	Activate tool radius compensation	m	+		PM-NC
CUTMOD	A	Activate Modification of the offset data for rotatable tools (in connection with orientable tool carriers)	m	+		PM-NC
CUTMODK	A	Activate Modification of the offset data for rotatable tools (in connection with orientation transformations defined by kinematic chains)	m	+		PM-NC
CYCLE60	C (T)	Engraving cycle		+		PM-NC
CYCLE61	C (T)	Face milling		+		PM-NC
CYCLE62	C (T)	Contour call		+		PM-NC
CYCLE63	C (T)	Contour pocket milling		+		PM-NC
CYCLE64	C (T)	Contour pocket predrilling		+		PM-NC
CYCLE70	C (T)	Thread milling		+		PM-NC
CYCLE72	C (T)	Path milling		+		PM-NC
CYCLE76	C (T)	Milling the rectangular spigot		+		PM-NC
CYCLE77	C (T)	Circular spigot milling		+		PM-NC
CYCLE78	C (T)	Mill cutting thread		+		PM-NC
CYCLE79	C (T)	Multiple edge		+		PM-NC
CYCLE81	C (T)	Drilling, centering		+		PM-NC
CYCLE82	C (T)	Drilling, counterboring		+		PM-NC
CYCLE83	C (T)	Deep-hole drilling		+		PM-NC
CYCLE84	C (T)	Tapping without compensating chuck		+		PM-NC
CYCLE85	C (T)	Reaming		+		PM-NC
CYCLE86	C (T)	Boring		+		PM-NC
CYCLE92	C (T)	Parting		+		PM-NC
CYCLE95	C (T)	Stock removal along the contour		+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
CYCLE98	C (T)	Thread chain		+		PM-NC
CYCLE99	C (T)	Thread cutting		+		PM-NC
CYCLE116	C (M)	Calculation of center point and radius of a circle		+		PM-MC
CYCLE119	C (M)	Determining position in space		+		PM-MC
CYCLE150	C (M)	Displaying/logging measurement results		+		PM-MC
CYCLE435	C (T)	Calculate dressing tool position		+		PM-NC
CYCLE495	C (T)	Form-truing		+		PM-NC
CYCLE750	C (A)	Internal operating cycle for CYCLE751 ... CY-CLE759 (contains the MMC command for the actual function call)		-		FM-A
CYCLE751	C (A)	Open / perform / close an optimization session		M		FM-A
CYCLE752	C (A)	Add axis to an optimization session		M		FM-A
CYCLE753	C (A)	Select optimization mode		M		FM-A
CYCLE754	C (A)	Add / remove language block		M		FM-A
CYCLE755	C (A)	Backing up/restoring data record		M		FM-A
CYCLE756	C (A)	Activate optimization results		M		FM-A
CYCLE757	C (A)	Store optimization data		M		FM-A
CYCLE758	C (A)	Changing the parameter value		M		FM-A
CYCLE759	C (A)	Read parameter value		M		FM-A
CYCLE782	C (T)	Adapt to load		+		PM-NC
CYCLE800	C (T)	Swiveling		+		PM-NC
CYCLE801	C (T)	Grid or frame		+		PM-NC
CYCLE802	C (T)	Arbitrary positions		+		PM-NC
CYCLE806	C (T)	Interpolation turning		+		PM-NC
CYCLE830	C (T)	Deep-hole drilling 2		+		PM-NC
CYCLE832	C (T)	High Speed Settings		+		PM-NC
CYCLE840	C (T)	Tapping with compensating chuck		+		PM-NC
CYCLE899	C (T)	Open slot milling		+		PM-NC
CYCLE930	C (T)	Groove		+		PM-NC
CYCLE940	C (T)	Undercut forms		+		PM-NC
CYCLE951	C (T)	Stock removal		+		PM-NC
CYCLE952	C (T)	Contour grooving		+		PM-NC
CYCLE961	C (M)	Determine the position of a workpiece corner (inner or outer) and insert as work offset		+		PM-MC
CYCLE971	C (M)	Adjust tool probe, measure tool length and/or tool radius (only for milling)		+		PM-MC
CYCLE973	C (M)	Adjust a workpiece probe on a surface on the workpiece or in a groove (only for turning)		+		PM-MC
CYCLE974	C (M)	Determine the workpiece zero in the selected measuring axis, determine tool offset with 1-point measurement (only for turning)		+		PM-MC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
CYCLE976	C (M)	Adjust a workpiece probe in a calibration ring or on a calibration ball completely in the working plane or at an edge for a particular axis and direction		+		PM-MC
CYCLE977	C (M)	Determine the center in the plane as well as the width or the diameter		+		PM-MC
CYCLE978	C (M)	Measure the position of an edge in the workpiece coordinate system		+		PM-MC
CYCLE979	C (M)	Determine center in the plane, measure radius of circle segment		+		PM-MC
CYCLE982	C (M)	Adjust tool probe, measure turning drilling and milling tools (only for turning)		+		PM-MC
CYCLE994	C (M)	Determine the workpiece zero in the selected measuring axis with 2-point measurement (only for turning)		+		PM-MC
CYCLE995	C (M)	Measure the angularity of the spindle on a machine tool		+		PM-MC
CYCLE996	C (M)	Determine transformation-relevant data for kinematic transformations with rotary axes		+		PM-MC
CYCLE997	C (M)	Determine center and diameter of a ball, measure center of three distributed balls		+		PM-MC
CYCLE998	C (M)	Determine the angular position of a surface (plane) referred to the working plane, determine angle of edges in the workpiece coordinate system		+		PM-MC
CYCLE4071	C (T)	Longitudinal grinding with infeed at the reversal point		+		PM-NC
CYCLE4072	C (T)	Longitudinal grinding with infeed at the reversal point and cancel signal		+		PM-NC
CYCLE4073	C (T)	Longitudinal grinding with continuous infeed		+		PM-NC
CYCLE4074	C (T)	Longitudinal grinding with continuous infeed and cancel signal		+		PM-NC
CYCLE4075	C (T)	Surface grinding with infeed at the reversal point		+		PM-NC
CYCLE4077	C (T)	Surface grinding with infeed at the reversal point and cancel signal		+		PM-NC
CYCLE4078	C (T)	Surface grinding with continuous infeed		+		PM-NC
CYCLE4079	C (T)	Surface grinding with intermittent infeed		+		PM-NC
CYCLE9960	C (M)	Measure kinematics completely		+		PM-MC

5.1.2 Operations D ... F

Operation	Type [1]	Meaning	W [2]	TP [3]	SA [4]	Description see [5]
[1] [2] [3] [4] [5] **for explanations, see legend (Page 1109).**						
D	A	Tool offset number		+		PM-NC
D0	A	With D0, offsets for the tool are ineffective		+		PM-NC
DAC	K	Absolute non-modal axis-specific diameter programming	s	+		PM-NC
DC	K	Absolute dimensions for rotary axes, approach position directly	s	+		PM-NC
DCI	K	Assign data class I (= Individual) (only SINUMERIK 828D)		+		PM-NC
DCM	K	Assign data class M (= Manufacturer) (only SINUMERIK 828D)		+		PM-NC
DCU	K	Assign data class U (= User) (only SINUMERIK 828D)		+		PM-NC
DEF	K	Variable definition		+		PM-NC
DEFAULT	K	Branch in CASE branch		+		PM-NC
DEFINE	K	Keyword for macro definitions		+		PM-NC
DELAYFSTOF	P	Define the end of a stop delay section	m	+	-	PM-NC
DELAYFSTON	P	Define the start of a stop delay section	m	+	-	PM-NC
DELDL	F	Delete additive offsets		+	-	PM-NC
DELDTG	P	Delete distance-to-go		-	+	FM-SA
DELETE	P	Delete the specified file. The file name can be specified with path and file identifier.		+	-	PM-NC
DELMLOWNER	F	Delete owner magazine location of the tool		+	-	FM-TM
DELMLRES	F	Delete magazine location reservation		+	-	FM-TM
DELMT	P	Delete multitool		+	-	FM-TM
DELOBJ	F	Deletion of elements from kinematic chains, protection areas, protection area elements, collision pairs and transformation data		+		PM-NC
DELT	P	Delete tool		+	-	FM-TM
DELTC	P	Delete tool carrier data record		+	-	FM-TM
DELTOOLENV	F	Delete data records describing tool environments		+	-	PM-NC
DIACYCOFA	K	Axis-specific modal diameter programming: OFF in cycles	m	+		FM-A
DIAM90	G	Diameter programming for G90, radius programming for G91	m	+		PM-NC
DIAM90A	K	Axis-specific modal diameter programming for G90 and AC, radius programming for G91 and IC	m	+		PM-NC
DIAMCHAN	K	Transfer of all axes from MD axis functions to diameter programming channel status		+		PM-NC
DIAMCHANA	K	Transfer of the diameter programming channel status		+		PM-NC

Operation	Type [1]	Meaning	W [2]	TP [3]	SA [4]	Description see [5]
[1] [2] [3] [4] [5] **for explanations, see legend (Page 1109).**						
DIAMCYCOF	G	Channel-specific diameter programming: OFF in cycles	m	+		FM-A
DIAMOF [6]	G	Diameter programming: OFF Initial setting, see machine manufacturer	m	+		PM-NC
DIAMOFA	K	Axis-specific modal diameter programming: OFF Initial setting, see machine manufacturer	m	+		PM-NC
DIAMON	G	Diameter programming: ON	m	+		PM-NC
DIAMONA	K	Axis-specific modal diameter programming: ON Activation, see machine manufacturer	m	+		PM-NC
DIC	K	Relative non-modal axis-specific diameter programming	s	+		PM-NC
DILF	A	Retraction path (length)	m	+		PM-NC
DISABLE	P	Interrupt OFF		+	-	PM-NC
DISC	A	Transition circle overshoot tool radius compensation	m	+		PM-NC
DISCL	A	Clearance between the end point of the fast infeed motion and the working plane		+		PM-NC
DISPLOF	PA	Suppress current block display		+		PM-NC
DISPLON	PA	Revoke suppression of the current block display		+		PM-NC
DISPR	A	Path differential for repositioning	s	+		PM-NC
DISR	A	Distance for repositioning	s	+		PM-NC
DISRP	A	Distance between the retraction plane and the working plane during smooth approach and retraction		+		PM-NC
DITE	A	Thread run-out path	m	+		PM-NC
DITS	A	Thread run-in path	m	+		PM-NC
DIV	K	Integer division		+		PM-NC
DL	A	Select location-dependent additive tool offset (DL, total set-up offset)	m	+		PM-NC
DO	K	Synchronized action: Triggering of actions when condition fulfilled		-	+	FM-SA
DRFOF	P	Deactivation of handwheel offsets (DRF)	m	+	-	PM-NC
DRIVE	G	Velocity-dependent path acceleration	m	+		PM-NC
DRIVEA	P	Activate knee-shaped acceleration characteristic for the programmed axes		+	-	PM-NC
DRVPRD	P	Read drive parameters		+	-	PM-NC
DRVPWR	P	Write drive parameters		+	-	PM-NC
DYNFINISH	G	Dynamic response for finishing	m	+		PM-NC
DYNNORM [6]	G	Standard dynamic response	m	+		PM-NC
DYNPOS	G	Dynamic response for positioning mode, tapping	m	+		PM-NC
DYNPREC	G	Dynamic response for smooth finishing	m	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
DYNROUGH	G	Dynamic response for roughing	m	+		PM-NC
DYNSEMIFIN	G	Dynamic response for semi-finishing	m	+		PM-NC
DZERO	P	Marks all D numbers of the TO unit as invalid		+	-	PM-NC
EAUTO	G	Definition of the last spline section by means of the last 3 points	m	+		PM-NC
EGDEF	P	Definition of an electronic gear		+	-	PM-NC
EGDEL	P	Delete coupling definition for the following axis		+	-	PM-NC
EGOFC	P	Turn off electronic gear continuously		+	-	PM-NC
EGOFS	P	Turn off electronic gear selectively		+	-	PM-NC
EGON	P	Turn on electronic gear		+	-	PM-NC
EGONSYN	P	Turn on electronic gear		+	-	PM-NC
EGONSYNE	P	Turn on electronic gear, with specification of approach mode		+	-	PM-NC
ELSE	K	NC program: Program branch if the IF condi-tion is not fulfilled		+	-	PM-NC
ELSE	K	Synchronized action: Triggering of actions when condition unfulfilled		-	+	FM-SA
ENABLE	P	Interrupt ON		+	-	PM-NC
ENAT 6)	G	Natural transition to next traversing block	m	+		PM-NC
ENDFOR	K	End line of FOR counter loop		+		PM-NC
ENDIF	K	End line of IF branch		+		PM-NC
ENDLABEL	K	End label for part program repetitions with RE-PEAT		+		PM-NC, FM-B
ENDLOOP	K	End line of endless program loop LOOP		+		PM-NC
ENDPROC	K	End line of program with start line PROC		+		
ENDWHILE	K	End line of WHILE loop		+		PM-NC
ESRR	P	Parameterizing drive-autonomous ESR retrac-tion in the drive		+		PM-NC
ESRS	P	Parameterizing drive-autonomous ESR shut-down in the drive		+		PM-NC
ETAN	G	Tangential transition to next traversing block at spline begin	m	+		PM-NC
EVERY	K	Execute synchronized action on transition of condition from FALSE to TRUE		-	+	FM-SA
EX	K	Keyword for value assignment in exponential notation		+		PM-NC
EXECSTRING	P	Transfer of a string variable with the executing part program line		+	-	PM-NC
EXECTAB	P	Execute an element from a motion table		+	-	PM-NC
EXECUTE	P	Program execution ON		+	-	PM-NC
EXP	F	Exponential function ex		+	+	PM-NC
EXTCALL	A	Execute external subprogram		+	+	PM-NC
EXTCLOSE	P	Closing external device / file that was opened for writing		+	-	PM-NC

Operation	Type [1]	Meaning	W [2]	TP [3]	SA [4]	Description see [5]
[1] [2] [3] [4] [5] **for explanations, see legend (Page 1109).**						
EXTERN	K	Declaration of a subprogram with parameter transfer		+		PM-NC
EXTOPEN	P	Opening external device / file for the channel for writing		+	-	PM-NC
F	A	Feedrate value (in conjunction with G4 the dwell time is also programmed with F)		+	+	PM-NC
FA	K	Axial feedrate	m	+	+	PM-NC
FAD	A	Infeed rate for soft approach and retraction		+		PM-NC
FALSE	K	Logical constant: Incorrect		+	+	PM-NC
FB	A	Non-modal feedrate		+		PM-NC
FCTDEF	P	Define polynomial function		+	-	PM-NC
FCUB	G	Feedrate variable according to cubic spline	m	+		PM-NC
FD	A	Path feedrate for handwheel override	s	+		PM-NC
FDA	K	Axis feedrate for handwheel override	s	+		PM-NC
FENDNORM [6]	G	Corner deceleration OFF	m	+		PM-NC
FFWOF [6]	G	Feedforward control OFF	m	+		PM-NC
FFWON	G	Feedforward control ON	m	+		PM-NC
FGREF	K	Reference radius for rotary axes or path reference factors for orientation axes (vector interpolation)	m	+		PM-NC
FGROUP	P	Definition of axis/axes with path feedrate		+	-	PM-NC
FI	K	Parameter for access to frame data: Fine offset		+		PM-NC
FIFOCTRL	G	Control of preprocessing buffer	m	+		PM-NC
FILEDATE	P	Returns date of most recent write access to file		+	-	PM-NC
FILEINFO	P	Returns summary information listing FILEDATE, FILESIZE, FILESTAT, and FILETIME		+	-	PM-NC
FILESIZE	P	Returns current file size		+	-	PM-NC
FILESTAT	P	Returns file status of rights for read, write, execute, display, delete (rwxsd)		+	-	PM-NC
FILETIME	P	Returns time of most recent write access to file		+	-	PM-NC
FINEA	K	End of motion when "Exact stop fine" reached	m	+		PM-NC
FL	K	Limit velocity for synchronized axis	m	+		PM-NC
FLIM	A	Adapting maximum path velocity	m	+	-	PM-NC
FLIN	G	Feed linear variable	m	+		PM-NC
FMA	K	Multiple feedrates axial	m	+		PM-NC
FNORM [6]	G	Feedrate normal to DIN 66025	m	+		PM-NC
FOC	K	Non-modal torque/force limitation	s	-	+	FM-SA
FOCOF	K	Switch off modal torque/force limitation	m	-	+	FM-SA
FOCON	K	Switch on modal torque/force limitation	m	-	+	FM-SA
FOR	K	Counter loop with fixed number of passes		+		PM-NC
FP	A	Fixed point: Number of fixed point to be approached	s	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
FPO	K	Feedrate characteristic programmed via a polynomial		+		PM-NC
FPR	P	Rotary axis identifier		+	-	PM-NC
FPRAOF	P	Deactivate revolutional feedrate		+	-	PM-NC
FPRAON	P	Activate revolutional feedrate		+	-	PM-NC
FRAME	K	Data type for the definition of coordinate systems		+		PM-NC
FRC	A	Feedrate for radius and chamfer	s	+		PM-NC
FRCM	A	Feedrate for radius and chamfer, modal	m	+		PM-NC
FROM	K	The action is executed if the condition is fulfilled once and as long as the synchronized action is active		-	+	FM-SA
FTOC	P	Change fine tool offset		-	+	FM-SA
FTOCOF 6)	G	Online fine tool offset OFF	m	+		PM-NC
FTOCON	G	Online fine tool offset ON	m	+		PM-NC
FXS	K	Travel to fixed stop ON	m	+	+	PM-NC
FXST	K	Torque limit for travel to fixed stop	m	+	+	PM-NC
FXSW	K	Monitoring window for travel to fixed stop		+	+	PM-NC
FZ	K	Tooth feedrate	m	+		PM-NC

5.1.3 Operations G ... L

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
G0	G	Linear interpolation with rapid traverse (rapid traverse motion)	m	+		PM-NC
G1 6)	G	Linear interpolation with feedrate (linear interpolation)	m	+		PM-NC
G2	G	Circular interpolation clockwise	m	+		PM-NC
G3	G	Circular interpolation counter-clockwise	m	+		PM-NC
G4	G	Dwell time, preset	s	+		PM-NC
G5	G	Oblique plunge-cut grinding	s	+		PM-NC
G7	G	Compensatory motion during oblique plunge-cut grinding	s	+		PM-NC
G9	G	Exact stop - deceleration	s	+		PM-NC
G17 6)	G	Selection of working plane X/Y	m	+		PM-NC
G18	G	Selection of working plane Z/X	m	+		PM-NC
G19	G	Selection of working plane Y/Z	m	+		PM-NC
G25	G	Lower working area limitation	s	+		PM-NC
G26	G	Upper working area limitation	s	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) for explanations, see legend (Page 1109).						
G33	G	Thread cutting with constant lead	m	+		PM-NC
G34	G	Thread cutting with linear increasing lead	m	+		PM-NC
G35	G	Thread cutting with linear decreasing lead	m	+		PM-NC
G40 6)	G	Tool radius compensation OFF	m	+		PM-NC
G41	G	Tool radius compensation left of contour	m	+		PM-NC
G42	G	Tool radius compensation right of contour	m	+		PM-NC
G53	G	Suppression of current zero offset (non-modal)	s	+		PM-NC
G54	G	1st settable zero offset	m	+		PM-NC
G55	G	2nd settable zero offset	m	+		PM-NC
G56	G	3rd settable zero offset	m	+		PM-NC
G57	G	4th settable zero offset	m	+		PM-NC
G58 (840D sl)	G	Absolute programmable work offset (coarse offset)	s	+		PM-NC
G58 (828D)	G	5th settable zero offset	m	+		PM-NC
G59 (840D sl)	G	Additive programmable work offset (fine off-set)	s	+		PM-NC
G59 (828D)	G	6th settable zero offset	m	+		PM-NC
G60 6)	G	Exact stop - deceleration	m	+		PM-NC
G62	G	Corner deceleration at inside corners when tool radius offset is active (G41, G42)	m	+		PM-NC
G63	G	Tapping with compensating chuck	s	+		PM-NC
G64	G	Continuous-path mode with reduced velocity as per the overload factor	m	+		PM-NC
G70	G	Inch dimensions for geometric specifications (lengths)	m	+	+	PM-NC
G71 6)	G	Metric dimensions for geometric specifications (lengths)	m	+	+	PM-NC
G74	G	Search for reference	s	+		PM-NC
G75	G	Fixed point approach	s	+		PM-NC
G90 6)	G	Absolute dimensions	m/s	+		PM-NC
G91	G	Incremental dimensions specification	m/s	+		PM-NC
G93	G	Inverse-time feedrate rpm	m	+		PM-NC
G94 6)	G	Linear feedrate F in mm/min or inch/min and degree/min	m	+		PM-NC
G95	G	Revolutional feedrate F in mm/rev or inch/rev	m	+		PM-NC
G96	G	Revolutional feedrate (as for G95) and con-stant cutting rate	m	+		PM-NC
G97	G	Revolutional feedrate and constant spindle speed (constant cutting rate OFF)	m	+		PM-NC
G110	G	Pole programming relative to the last program-med setpoint position	s	+		PM-NC
G111	G	Pole programming relative to zero of current workpiece coordinate system	s	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
G112	G	Pole programming relative to the last valid pole	s	+		PM-NC
G140 6)	G	SAR approach direction defined by G41/G42	m	+		PM-NC
G141	G	SAR approach direction to left of contour	m	+		PM-NC
G142	G	SAR approach direction to right of contour	m	+		PM-NC
G143	G	SAR approach direction tangent-dependent	m	+		PM-NC
G147	G	Soft approach with straight line	s	+		PM-NC
G148	G	Soft retraction with straight line	s	+		PM-NC
G153	G	Suppression of current frames including basic frame	s	+		PM-NC
G247	G	Soft approach with quadrant	s	+		PM-NC
G248	G	Soft retraction with quadrant	s	+		PM-NC
G290 6)	G	Switch over to SINUMERIK mode ON	m	+		FM-TM
G291	G	Switch over to ISO2/3 mode ON	m	+		FM-TM
G331	G	Rigid tapping, positive lead, clockwise	m	+		PM-NC
G332	G	Rigid tapping, negative lead, counter-clockwise	m	+		PM-NC
G335	G	Turning a convex thread in clockwise direction	m	+		PM-NC
G336	G	Turning a convex thread in counter-clockwise direction	m	+		PM-NC
G340 6)	G	Spatial approach block (depth and in plane at the same time (helix))	m	+		PM-NC
G341	G	Initial infeed on perpendicular axis (z), then approach in plane	m	+		PM-NC
G347	G	Soft approach with semicircle	s	+		PM-NC
G348	G	Soft retraction with semicircle	s	+		PM-NC
G450 6)	G	Transition circle	m	+		PM-NC
G451	G	Intersection of equidistances	m	+		PM-NC
G460 6)	G	Activation of collision detection for the approach and retraction block	m	+		PM-NC
G461	G	Insertion of a circle into the TRC block	m	+		PM-NC
G462	G	Insertion of a straight line into the TRC block	m	+		PM-NC
G500 6)	G	Deactivation of all adjustable frames, basic frames are active	m	+		PM-NC
G505 ... G599	G	5 ... 99 Settable work offset	m	+		PM-NC
G601 6)	G	Block change at exact stop fine	m	+		PM-NC
G602	G	Block change at exact stop coarse	m	+		PM-NC
G603	G	Block change at IPO block end	m	+		PM-NC
G621	G	Corner deceleration at all corners	m	+		PM-NC
G641	G	Continuous-path mode with smoothing as per distance criterion (= programmable smoothing clearance)	m	+		PM-NC
G642	G	Continuous-path mode with smoothing within the defined tolerances	m	+		PM-NC

Operation	Type [1)]	Meaning	W [2)]	TP [3)]	SA [4)]	Description see [5)]
[1) 2) 3) 4) 5)] for explanations, see legend (Page 1109).						
G643	G	Continuous-path mode with smoothing within the defined tolerances (block-internal)	m	+		PM-NC
G644	G	Continuous-path mode with smoothing with maximum possible dynamic response	m	+		PM-NC
G645	G	Continuous-path mode with smoothing and tangential block transitions within defined tolerances	m	+		PM-NC
G646	G	Extended continuous-path mode with reduced velocity as per the overload factor	m	+		PM-NC
G700	G	Inch dimensions for geometric and technological specifications (lengths, feedrate)	m	+	+	PM-NC
G710 [6)]	G	Metric dimensions for geometric and technological specifications (lengths, feedrate)	m	+	+	PM-NC
G810 [6)], ..., G819	G	G group reserved for the OEM user		+		PM-NC
G820 [6)], ..., G829	G	G group reserved for the OEM user		+		PM-NC
G931	G	Feedrate specified by means of traversing time, deactivate constant path velocity	m	+		
G942	G	Freeze linear feedrate and constant cutting rate or spindle speed	m	+		
G952	G	Freeze revolutional feedrate and constant cutting rate or spindle speed	m	+		
G961	G	Linear feedrate (as for G94) and constant cutting rate	m	+		PM-NC
G962	G	Linear feedrate or revolutional feedrate and constant cutting rate	m	+		PM-NC
G971	G	Linear feedrate and constant spindle speed (constant cutting rate OFF)	m	+		PM-NC
G972	G	Linear feedrate or revolutional feedrate and constant spindle speed (constant cutting rate OFF)	m	+		PM-NC
G973	G	Revolutional feedrate without spindle speed limitation and constant spindle speed (G97 without LIMS for ISO mode)	m	+		PM-NC
GEOAX	P	Assign new channel axes to geometry axes 1 - 3		+	-	PM-NC
GET	P	Replace enabled axis between channels		+	+	PM-NC
GETACTT	F	Gets active tool from a group of tools with the same name		+	-	FM-TM
GETACTTD	F	Gets the T number associated with an absolute D number		+	-	PM-NC
GETD	P	Replace axis directly between channels		+	-	PM-NC
GETDNO	F	Returns the D number of a cutting edge (CE) of a tool (T)		+	-	PM-NC
GETEXET	P	Reading of the loaded T number		+	-	FM-TM
GETFREELOC	P	Find a free space in the magazine for a given tool		+	-	FM-TM
GETSELT	P	Return selected T number		+	-	FM-TM

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
GETT	F	Get T number for tool name		+	-	FM-TM
GETTCOR	F	Read out tool lengths and/or tool length components		+	-	PM-NC
GETTENV	F	Read T, D and DL numbers		+	-	PM-NC
GETVARAP	F	Read access rights to a system/user variable		+	-	PM-NC
GETVARDFT	F	Read default value of a system/user variable		+	-	PM-NC
GETVARLIM	F	Read limit values of a system/user variable		+	-	PM-NC
GETVARPHU	F	Read physical unit of a system/user variable		+	-	PM-NC
GETVARTYP	F	Read data type of a system/user variable		+	-	PM-NC
GFRAME0 ... GFRAME100	G	Activation of the grinding frame <n> of the data management in channel	m	+		PM-NC
GOTO	K	Jump operation first forward then backward (direction initially to end of program and then to beginning of program)		+		PM-NC
GOTOB	K	Jump backward (toward the beginning of the program)		+		PM-NC
GOTOC	K	As GOTO, but suppress alarm 14080 "Jump destination not found"		+		PM-NC
GOTOF	K	Jump forward (toward the end of the program)		+		PM-NC
GOTOS	K	Jump back to beginning of program		+		PM-NC
GP	K	Keyword for the indirect programming of position attributes		+		PM-NC
GROUP_ ADDEND	C (T)	End of trial cut addition		+		PM-NC
GROUP_BEGIN	C (T)	Beginning of program group		+		PM-NC
GROUP_END	C (T)	End of program group		+		PM-NC
GWPSOF	P	Deselect constant grinding wheel peripheral speed (GWPS)	s	+	-	PM-NC
GWPSON	P	Select constant grinding wheel peripheral speed (GWPS)	s	+	-	PM-NC
H...	A	Auxiliary function output to the PLC		+	+	PM-NC, FM-B
HOLES1	C (T)	Row of holes		+		PM-NC
HOLES2	C (T)	Circle of holes		+		PM-NC
I	A	Interpolation parameters	s	+		PM-NC
I1	A	Intermediate point coordinate	s	+		PM-NC
IC	K	Incremental dimensions input	s	+		PM-NC
ICYCOF	P	All blocks of a technology cycle are processed in one interpolation cycle following ICYCOF		+	+	FM-SA
ICYCON	P	Each block of a technology cycle is processed in a separate interpolation cycle following ICYCON		+	+	FM-SA
ID	K	Identifier for modal synchronous actions	m	-	+	FM-SA
IDS	K	Identifier for modal static synchronous actions		-	+	FM-SA

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
IF	K	Introduction of a conditional jump in the part program/technology cycle		+	+	PM-NC
INDEX	F	Define index of character in input string		+	-	PM-NC
INICF	K	Initialization of variables for NEWCONF		+		PM-NC
INIPO	K	Initialization of variables at POWER ON		+		PM-NC
INIRE	K	Initialization of variables at reset		+		PM-NC
INIT	P	Selection of a particular NC program for execution in a particular channel		+	-	PM-NC
INITIAL		Generation of an INI file across all areas		+		PM-NC
INT	K	Data type: Integer with sign		+		PM-NC
INTERSEC	F	Calculate intersection between two contour elements		+	-	PM-NC
INVCCW	G	Trace involute, counter-clockwise	m	+		PM-NC
INVCW	G	Trace involute, clockwise	m	+		PM-NC
INVFRAME	F	Calculate the inverse frame from a frame		+	-	FM-B
IP	K	Variable interpolation parameter		+		PM-NC
IPOBRKA	P	Motion criterion from braking ramp activation	m	+	+	
IPOENDA	K	End of motion when "IPO stop" reached	m	+		PM-NC
IPTRLOCK	P	Freeze start of the untraceable program section at next machine function block.	m	+	-	PM-NC
IPTRUNLOCK	P	Set end of untraceable program section at current block at time of interruption.	m	+	-	PM-NC
IR	A	Center of circle coordinate (X axis) when turning a convex thread		+		PM-NC
ISAXIS	F	Check if geometry axis 1 specified as parameter		+	-	PM-NC
ISD	A	Insertion depth	m	+		PM-NC
ISFILE	F	Check whether the file exists in the NC application memory		+	-	PM-NC
ISNUMBER	F	Check whether the input string can be converted to a number		+	-	PM-NC
ISOCALL	K	Indirect call of a program programmed in an ISO language		+		PM-NC
ISVAR	F	Check whether the transfer parameter contains a variable declared in the NC		+	-	PM-NC
J	A	Interpolation parameters	s	+		PM-NC
J1	A	Intermediate point coordinate	s	+		PM-NC
JERKA	P	Activate acceleration response set via MD for programmed axes		+	-	
JERKLIM	K	Adapt maximum axis jerk	m	+		PM-NC
JERKLIMA	K	Reduction or overshoot of the maximum slave axis jerk	m	+	+	PM-NC
JR	A	Center of circle coordinate (Y axis) when turning a convex thread		+		PM-NC
K	A	Interpolation parameters	s	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
K1	A	Intermediate point coordinate	s	+		PM-NC
KONT	G	Travel around contour on tool offset	m	+		PM-NC
KONTC	G	Approach/retract with continuous-curvature polynomial	m	+		PM-NC
KONTT	G	Approach/retract with continuous-tangent polynomial	m	+		PM-NC
KR	A	Center of circle coordinate (Z axis) when turning a convex thread		+		PM-NC
L	A	Subprogram number	s	+	+	PM-NC
LEAD	A	Lead angle 1st basic tool orientation 2nd orientation polynomials	m	+		PM-NC
LEADOF	P	Axial master value coupling OFF		+	+	PM-NC
LEADON	P	Axial master value coupling on		+	+	PM-NC
LENTOAX	F	Provides information about the assignment of tool lengths L1, L2, and L3 of the active tool to the abscissa, ordinate and applicate		+	-	PM-NC
LFOF 6)	G	Fast retraction for thread cutting OFF	m	+		PM-NC
LFON	G	Fast retraction for thread cutting ON	m	+		PM-NC
LFPOS	G	Retraction of the axis declared with POLFMASK or POLFMLIN to the absolute axis position programmed with POLF	m	+		PM-NC
LFTXT 6)	G	The plane of the retraction movement for fast retraction is determined from the path tangent and the current tool direction	m	+		PM-NC
LFWP	G	The plane of the retraction movement for fast retraction is determined by the current working plane (G17/G18/G19)	m	+		PM-NC
LIFTFAST	K	Fast retraction		+		PM-NC
LIMS	K	Speed limitation for G96/G961 and G97	m	+		PM-NC
LLI	K	Lower limit value of variables		+		PM-NC
LN	F	Natural logarithm		+	+	PM-NC
LOCK	P	Disable synchronous action with ID (stop technology cycle)		-	+	FM-SA
LONGHOLE	C (T)	Elongated hole		+		PM-NC
LOOP	K	Introduction of an endless loop		+		PM-NC

5.1.4 Operations M ... R

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
M0		Programmed stop		+	+	PM-NC
M1		Optional stop		+	+	PM-NC
M2		End of program, main program (as M30)		+	+	PM-NC
M3		CW spindle rotation		+	+	PM-NC
M4		CCW spindle rotation		+	+	PM-NC
M5		Spindle stop		+	+	PM-NC
M6		Tool change		+	+	PM-NC
M17		End of subprogram		+	+	PM-NC
M19		Spindle positioning to the position entered in SD43240		+	+	PM-NC
M30		End of program, main program (as M2)		+	+	PM-NC
M40		Automatic gear change		+	+	PM-NC
M41 ... M45		Gear stage 1 ... 5		+	+	PM-NC
M70		Transition to axis mode		+	+	PM-NC
MASLDEF	P	Define main/sub-coupling group		+	+	PM-NC
MASLDEL	P	Disconnect main/sub-coupling group and delete definition of linkage		+	+	PM-NC
MASLOF	P	Deactivation of a temporary main/sub-coupling		+	+	PM-NC
MASLOFS	P	Switch off a temporary main/sub-coupling with automatic stop of the sub-axis		+	+	PM-NC
MASLON	P	Activation of a temporary main/sub-coupling		+	+	PM-NC
MATCH	F	Search for string in string		+	-	PM-NC
MAXVAL	F	Larger value of two variables (arithm. function)		+	+	PM-NC
MCALL	K	Activate/deactivate modally effective subprogram call	m	+		PM-NC
MCALLOF	K	Suppress modally effective subprogram call block by block	s	+		PM-NC
MEAC	A	Axis-specific, continuous measurement without delete distance-to-go	s	+	+	PM-NC
MEAFRAME	F	Frame calculation from measuring points		+	-	PM-NC
MEAS	A	Channel-specific measurement with delete distance-to-go	s	+		PM-NC
MEASA	A	Axis-specific measurement with deletion of distance-to-go	s	+	+	PM-NC
MEASF	A	Channel-specific high-speed measurement with delete distance-to-go	s	+		PM-NC
MEASURE	F	Calculation method for workpiece and tool measurement		+	-	FM-TE
MEAW	A	Channel-specific measurement without delete distance-to-go	s	+		PM-NC

Operation	Type [1]	Meaning	W [2]	TP [3]	SA [4]	Description see [5]
[1] [2] [3] [4] [5] **for explanations, see legend (Page 1109).**						
MEAWA	A	Axes-specific measurement without delete distance-to-go	s	+	+	PM-NC
MI	K	Access to frame data: Mirroring		+		PM-NC
MINDEX	F	Define index of character in input string		+	-	PM-NC
MINVAL	F	Smaller value of two variables (arithm. function)		+	+	PM-NC
MIRROR	G	Programmable mirroring	s	+		PM-NC
MMC	P	Call the dialog window interactively from the part program on the HMI		+	-	PM-NC
MOD	K	Modulo division		+		PM-NC
MODAXVAL	F	Determine modulo position of a modulo rotary axis		+	-	PM-NC
MOV	K	Start positioning axis		-	+	FM-SA
MOVT	A	Specify end point of a traversing motion in the tool direction				FM-B
MSG	P	Programmable messages	m	+	-	PM-NC
MVTOOL	P	Language command to move tool		+	-	FM-TM
N	A	NC auxiliary block number		+		PM-NC
NAMETOINT	F	Determining the system variable index		+		PM-NC
NC	K	Specify validity range for data		+		PM-NC
NEWCONF	P	Apply modified machine data (corresponds to "Activate machine data")		+	-	PM-NC
NEWMT	F	Create new multitool		+	-	FM-TM
NEWT	F	Create new tool		+	-	FM-TM
NORM [6]	G	Standard setting in starting point and end point with tool offset	m	+		PM-NC
NOT	K	Logic NOT (negation)		+		PM-NC
NPROT	P	Machine-specific protection area ON/OFF		+	-	PM-NC
NPROTDEF	P	Definition of a machine-specific protection area		+	-	PM-NC
NUMBER	F	Convert input string to number		+	-	PM-NC
OEMIPO1	G	OEM interpolation 1	m	+		PM-NC
OEMIPO2	G	OEM interpolation 2	m	+		PM-NC
OF	K	Keyword in CASE branch		+		PM-NC
OFFN	A	Allowance on the programmed contour	m	+		PM-NC
OMA1	A	OEM address 1	m	+		PM-NC
OMA2	A	OEM address 2	m	+		PM-NC
OMA3	A	OEM address 3	m	+		PM-NC
OMA4	A	OEM address 4	m	+		PM-NC
OMA5	A	OEM address 5	m	+		PM-NC
OR	K	Logic operator, OR operation		+		PM-NC
ORIANGLE	G	Interpolation via the virtual kinematics defined by G group 50	m	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
ORIAXES	G	Linear interpolation of the orientation axes via the shortest path	m	+		PM-NC
ORIAXESFR	G	Linear interpolation of the orientation axes to the programmed positions without consideration of the shortest path	m	+		PM-NC
ORIAXPOS	G	Orientation angle via virtual orientation axes with rotary axis positions	m	+		PM-NC
ORIC 6)	G	Orientation changes at outside corners are superimposed on the circle block to be inserted	m	+		PM-NC
ORICONCCW	G	Interpolation on a circular peripheral surface in CCW direction	m	+		PM-NC, FM-TR
ORICONCW	G	Interpolation on a circular peripheral surface in CW direction	m	+		PM-NC, FM-TR
ORICONIO	G	Interpolation on a circular peripheral surface with intermediate orientation setting	m	+		PM-NC, FM-TR
ORICONTO	G	Interpolation on circular peripheral surface in tangential transition (final orientation)	m	+		PM-NC, FM-TR
ORICURINV	G	Interpolation with additional space curve for orientation (orientation is defined inversely to ORICURVE)	m	+		PM-NC
ORICURVE	G	Interpolation of orientation with specification of motion of two contact points of tool	m	+		PM-NC, FM-TR
ORID	G	Orientation changes are performed before the circle block	m	+		PM-NC
ORIEULER 6)	G	Orientation angle via Euler angle	m	+		PM-NC
ORIMKS	G	Tool orientation in the machine coordinate system	m	+		PM-NC
ORIPATH	G	Tool orientation in relation to path	m	+		PM-NC
ORIPATHS	G	Tool orientation in relation to path, blips in the orientation characteristic are smoothed	m	+		PM-NC
ORIPLANE	G	Interpolation in a plane (corresponds to ORIVECT), large-radius circular interpolation	m	+		PM-NC
ORIRESET	P	Initial tool orientation with up to 3 orientation axes		+	-	PM-NC
ORIROTA 6)	G	Angle of rotation to an absolute direction of rotation	m	+		PM-NC
ORIROTC	G	Tangential rotational vector in relation to path tangent	m	+		PM-NC
ORIROTR	G	Angle of rotation relative to the plane between the start and end orientation	m	+		PM-NC
ORIROTT	G	Angle of rotation relative to the change in the orientation vector	m	+		PM-NC
ORIRPY	G	Orientation angle via RPY angle (XYZ)	m	+		PM-NC
ORIRPY2	G	Orientation angle via RPY angle (ZYX)	m	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
ORIS	A	Change in orientation	m	+		PM-NC
ORISOF 6)	G	Smoothing of the orientation characteristic OFF	m	+		PM-NC
ORISOLH	F	Calculate orientations		+		PM-NC
ORISON	G	Smoothing of the orientation characteristic ON	m	+		PM-NC
ORIVECT 6)	G	Large-circle interpolation (identical to ORI-PLANE)	m	+		PM-NC
ORIVIRT1	G	Orientation angle via virtual orientation axes (definition 1)	m	+		PM-NC
ORIVIRT2	G	Orientation angle via virtual orientation axes (definition 1)	m	+		PM-NC
ORIWKS 6)	G	Tool orientation in the workpiece coordinate system	m	+		PM-NC
OS	K	Oscillation on/off		+		PM-NC
OSB	K	Oscillating: Starting point	m	+		PM-NC
OSC	G	Continuous tool orientation smoothing	m	+		PM-NC
OSCILL	K	Axis: 1 - 3 infeed axes	m	+		PM-NC
OSCTRL	K	Oscillation options	m	+		PM-NC
OSD	G	Smoothing of tool orientation by specifying smoothing distance with SD	m	+		PM-NC
OSE	K	Oscillation end point	m	+		PM-NC
OSNSC	K	Oscillating: Number of spark-out cycles	m	+		PM-NC
OSOF 6)	G	Tool orientation smoothing OFF	m	+		PM-NC
OSP1	K	Oscillating: Left reversal point	m	+		PM-NC
OSP2	K	Oscillation right reversal point	m	+		PM-NC
OSS	G	Tool orientation smoothing at end of block	m	+		PM-NC
OSSE	G	Tool orientation smoothing at start and end of block	m	+		PM-NC
OST	G	Smoothing of tool orientation by specifying angular tolerance in degrees with SD (maximum deviation from programmed. orientation characteristic)	m	+		PM-NC
OST1	K	Oscillating: Stopping point in left reversal point	m	+		PM-NC
OST2	K	Oscillating: Stopping point in right reversal point	m	+		PM-NC
OTOL	A	Orientation tolerance for compressor functions, orientation smoothing and smoothing types	m	+	-	PM-NC
OTOLG0	A	Orientation tolerance for rapid traverse movements	m	+	-	PM-NC
OVR	K	Speed offset	m	+		PM-NC
OVRA	K	Axial speed offset	m	+	+	PM-NC
OVRRAP	K	Rapid traverse override	m	+		PM-NC
P	A	Number of subprogram repetitions		+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
PACCLIM	A	Adapting maximum path acceleration	m	+	-	PM-NC
PAROT	G	Align workpiece coordinate system on workpiece	m	+		PM-NC
PAROTOF 6)	G	Deactivate frame rotation in relation to workpiece	m	+		PM-NC
PCALL	K	Call subprograms with absolute path and parameter transfer		+		PM-NC
PDELAYOF	G	Punching with delay OFF	m	+		PM-NC
PDELAYON 6)	G	Punching with delay ON	m	+		PM-NC
PHI	K	Angle of rotation of the orientation around the direction axis of the taper		+		PM-NC
PHU	K	Physical unit of a variable		+		PM-NC
PL	A	1. B spline: Node clearance 2. Polynomial interpolation Length of the parameter interval for polynomial interpolation	s	+		PM-NC
PM	K	Per minute		+		PM-NC
PO	K	Polynomial coefficient for polynomial interpolation	s	+		PM-NC
POCKET3	C (T)	Milling the rectangular pocket		+		PM-NC
POCKET4	C (T)	Milling the circular pocket		+		PM-NC
POLF	K	LIFTFAST retraction position	m	+		PM-NC
POLFA	P	Start retraction position of single axes with $AA_ESR_TRIGGER	m	+	+	PM-NC
POLFMASK	P	Enable axes for retraction without a connection between the axes	m	+	-	PM-NC
POLFMLIN	P	Enable axes for retraction with a linear connection between the axes	m	+	-	PM-NC
POLY	G	Polynomial interpolation	m	+		PM-NC
POLYPATH	P	Polynomial interpolation can be selected for the AXIS or VECT axis groups	m	+	-	PM-NC
PON	G	Punching ON	m	+		PM-NC
PONS	G	Punching ON in interpolation cycle	m	+		PM-NC
POS	K	Axis positioning		+	+	PM-NC
POSA	K	Position axis across block boundary		+	+	PM-NC
POSM	P	Position magazine		+	-	FM-TM
POSMT	P	Position multitool on toolholder at location number		+	-	FM-TM
POSP	K	Positioning axis in parts (oscillation)		+		PM-NC
POSRANGE	F	Determine whether the currently interpolated position setpoint of an axis is located in a window at a predefined reference position		+	+	FM-SA
POT	F	Power function		+	+	PM-NC
PR	K	Per revolution		+		PM-NC
PREPRO	PA	Identify subprograms with preparation		+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
PRESETON	P	Actual value setting with loss of the referencing status		+	+	PM-NC
PRESETONS	P	Actual value setting without loss of the referencing status		+	+	PM-NC
PRIO	K	Keyword for setting the priority for interrupt processing		+		PM-NC
PRLOC	K	Initialization of variables at reset only after local change		+		PM-NC
PROC	K	First operation in a program		+		PM-NC
PROTA	P	Request for a recalculation of the collision model		+		PM-NC
PROTD	F	Calculating the distance between two protection areas		+		PM-NC
PROTS	P	Setting the protection area status		+		PM-NC
PSI	K	Opening angle of the taper		+		PM-NC
PTP	G	Point-to-point motion (PTP travel)	m	+		PM-NC
PTPG0	G	Point-to-point motion only with G0, otherwise path motion CP	m	+		PM-NC
PTPWOC	G	Point-to-point motion without compensation movements caused by changes in orientation	m	+		PM-NC
PUNCHACC	P	Travel-dependent acceleration for nibbling		+	-	PM-NC
PUTFTOC	P	Tool fine offset for parallel dressing		+	-	PM-NC
PUTFTOCF	P	Tool fine offset dependent on a function for parallel dressing defined with FCTDEF		+	-	PM-NC
PW	A	B spline, point weight	s	+		PM-NC
QU	K	Fast additional (auxiliary) function output		+		PM-NC
R...	A	Arithmetic parameter also as settable address identifier and with numerical extension		+		PM-NC
RAC	K	Absolute non-modal axis-specific radius programming	s	+		PM-NC
RDISABLE	P	Read-in disable		-	+	FM-SA
READ	P	Reads one or more lines in the specified file and stores the information read in the array		+	-	PM-NC
REAL	K	Data type: Floating-point variable with sign (real numbers)		+		PM-NC
REDEF	K	Redefinition of system variables, user variables, and NC language commands		+		PM-NC
RELEASE	P	Release machine axes for axis exchange		+	+	PM-NC
REP	K	Keyword for initialization of all elements of an array with the same value		+		PM-NC
REPEAT	K	Repetition of a program loop		+		PM-NC
REPEATB	K	Repetition of a program line		+		PM-NC
REPOSA	G	Linear repositioning with all axes	s	+		PM-NC
REPOSH	G	Repositioning with semicircle	s	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
REPOSHA	G	Repositioning with all axes; geometry axes in semicircle	s	+		PM-NC
REPOSL	G	Linear repositioning	s	+		PM-NC
REPOSQ	G	Repositioning in a quadrant	s	+		PM-NC
REPOSQA	G	Linear repositioning with all axes, geometry axes in quadrant	s	+		PM-NC
RESETMON	P	Language command for setpoint activation		+	-	FM-TM
RET	P	End of subprogram		+	+	PM-NC
RETB	P	End of subprogram		+	+	PM-NC
RIC	K	Relative non-modal axis-specific radius programming	s	+		PM-NC
RINDEX	F	Define index of character in input string		+	-	PM-NC
RMB	G	Repositioning to start of block	m	+		PM-NC
RMBBL	G	Repositioning to start of block	s	+		PM-NC
RME	G	Repositioning to end of block	m	+		PM-NC
RMEBL	G	Repositioning to end of block	s	+		PM-NC
RMI 6)	G	Repositioning to interrupt point	m	+		PM-NC
RMIBL 6)	G	Repositioning to interrupt point	s	+		PM-NC
RMN	G	Repositioning to the nearest path point	m	+		PM-NC
RMNBL	G	Repositioning to the nearest path point	s	+		PM-NC
RND	A	Round the contour corner	s	+		PM-NC
RNDM	A	Modal rounding	m	+		PM-NC
ROT	G	Programmable rotation	s	+		PM-NC
ROTS	G	Programmable frame rotations with solid angles	s	+		PM-NC
ROUND	F	Rounding of decimal places		+	+	PM-NC
ROUNDUP	F	Rounding of an input value		+	+	PM-NC
RP	A	Polar radius	m/s	+		PM-NC
RPL	A	Rotation in the plane	s	+		PM-NC
RT	K	Parameter for access to frame data: Rotation		+		PM-NC
RTLIOF	G	G0 without linear interpolation (single-axis interpolation)	m	+		PM-NC
RTLION 6)	G	G0 with linear interpolation	m	+		PM-NC

5.1.5 Operations S ... Z

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
S	A	Spindle speed (with G4, G96/G961 different meaning)	m/s	+	+	PM-NC
SAVE	PA	Attribute for saving information when subprograms are called		+		PM-NC
SBLOF	P	Suppress single block		+	-	PM-NC
SBLON	P	Revoke suppression of single block		+	-	PM-NC
SC	K	Parameter for access to frame data: Scaling		+		PM-NC
SCALE	G	Programmable scaling	s	+		PM-NC
SCC	K	Selective assignment of transverse axis to G96/ G961/G962. Axis identifiers may take the form of geometry, channel or machine axes.		+		PM-NC
SCPARA	K	Program servo parameter set		+	+	PM-NC
SD	A	Spline degree	s	+		PM-NC
SET	K	Keyword for initialization of all elements of an array with listed values		+		PM-NC
SETAL	P	Set alarm		+	+	PM-NC
SETDNO	F	Assign the D number of a cutting edge (CE) of a tool (T)		+	-	PM-NC
SETINT	K	Define which interrupt routine is to be activated when an NC input is present		+		PM-NC
SETM	P	Setting of markers in dedicated channel		+	+	PM-NC
SETMS	P	Reset to the master spindle defined in machine data		+	-	PM-NC
SETMS(n)	P	Set spindle n as master spindle		+		PM-NC
SETMTH	P	Set master toolholder number		+	-	FM-TM
SETPIECE	P	Set piece number for all tools assigned to the spindle		+	-	FM-TM
SETTA	P	Activate tool from wear group		+	-	FM-TM
SETTCOR	F	Modification of tool components taking all supplementary conditions into account		+	-	PM-NC
SETTIA	P	Deactivate tool from wear group		+	-	FM-TM
SF	A	Starting point offset for thread cutting	m	+		PM-NC
SIN	F	Sine (trigon. function)		+	+	PM-NC
SIRELAY	F	Activate the safety functions parameterized with SIRELIN, SIRELOUT, and SIRELTIME		-	+	FM-SI
SIRELIN	P	Initialize input variables of function block		+	-	FM-SI
SIRELOUT	P	Initialize output variables of function block		+	-	FM-SI
SIRELTIME	P	Initialize timers of function block		+	-	FM-SI
SLOT1	C (T)	Longitudinal groove		+		PM-NC
SLOT2	C (T)	Circumferential groove		+		PM-NC
SOFT	G	Soft path acceleration	m	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
SOFTA	P	Activate jerk-limited axis acceleration for the programmed axes		+	-	PM-NC
SON	G	Nibbling ON	m	+		PM-NC
SONS	G	Nibbling ON in interpolation cycle	m	+		PM-NC
SPATH 6)	G	Path reference for FGROUP axes is arc length	m	+		PM-NC
SPCOF	P	Switch master spindle or spindle(s) from position control to speed control	m	+	-	PM-NC
SPCON	P	Switch master spindle or spindle(s) from speed control to position control	m	+	-	PM-NC
SPI	F	Converts spindle number into axis identifier		+	-	PM-NC
SPIF1 6)	G	Fast NC inputs/outputs for punching/nibbling byte 1	m	+		FM-TE
SPIF2	G	Fast NC inputs/outputs for punching/nibbling byte 2	m	+		FM-TE
SPLINEPATH	P	Define spline grouping		+	-	PM-NC
SPN	A	Number of path sections per block	s	+		PM-NC
SPOF 6)	G	Stroke OFF, nibbling, punching OFF	m	+		PM-NC
SPOS	K	Spindle position	m	+	+	PM-NC
SPOSA	K	Spindle position across block boundaries	m	+		PM-NC
SPP	A	Length of a path section	m	+		PM-NC
SPRINT	F	Returns an input string formatted		+		PM-NC
SQRT	F	Square root (arithmetic function)		+	+	PM-NC
SR	A	Oscillation retraction path for synchronous action	s	+		PM-NC
SRA	K	Oscillation retraction path with external input axial for synchronous action	m	+		PM-NC
ST	A	Oscillation sparking-out time for synchronous action	s	+		PM-NC
STA	K	Oscillation sparking-out time axial for synchronous action	m	+		PM-NC
START	P	Start selected programs simultaneously in several channels from current program		+	-	PM-NC
STARTFIFO 6)	G	Execute; fill preprocessing memory simultaneously	m	+		PM-NC
STAT		Position of joints	s	+		PM-NC
STOLF	A	G0 tolerance factor	m	+		PM-NC
STOPFIFO	G	Stop machining; fill preprocessing memory until STARTFIFO is detected, preprocessing memory is full or end of program	m	+		PM-NC
STOPRE	P	Preprocessing stop until all prepared blocks in the main run are executed		+	-	PM-NC
STOPREOF	P	Revoke preprocess stop		-	+	FM-SA
STRING	K	Data type: Character string		+		PM-NC

Operation	Type [1)]	Meaning	W [2)]	TP [3)]	SA [4)]	Description see [5)]
[1) 2) 3) 4) 5)] **for explanations, see legend (Page 1109).**						
STRINGIS	F	Checks the present scope of NC language and the NC cycle names, user variables, macros, and label names belonging specifically to this command to establish whether these exist, are valid, defined or active.		+	-	PM-NC
STRLEN	F	Define string length		+	-	PM-NC
SUBSTR	F	Define index of character in input string		+	-	PM-NC
SUPA	G	Suppression of current work offset, including programmed offsets, system frames, handwheel offsets (DRF), external work offset, and overlaid movement	s	+		PM-NC
SUPD	G	Suppression of the active tool offsets	s	+	-	PM-NC
SVC	K	Tool cutting rate	m	+		PM-NC
SYNFCT	P	Evaluation of a polynomial as a function of a condition in the motion-synchronous action		-	+	FM-SA
SYNR	K	The variable is read synchronously, i.e. at the time of execution		+		PM-NC
SYNRW	K	The variable is read and written synchronously, i.e. at the time of execution		+		PM-NC
SYNW	K	The variable is written synchronously, i.e. at the time of execution		+		PM-NC
T	A	Call tool (only change if specified in machine data; otherwise M6 command necessary)		+		PM-NC
TAN	F	Tangent (trigon. function)		+	+	PM-NC
TANG	P	Tangential control: Define coupling		+	-	PM-NC
TANGDEL	P	Tangential control: Delete coupling		+	-	PM-NC
TANGOF	P	Tangential control: Deactivate coupling		+	-	PM-NC
TANGON	P	Tangential control: Activate coupling		+	-	PM-NC
TCA (828D: _TCA)	P	Tool selection/tool change irrespective of tool status		+	-	FM-TM
TCARR	A	Request toolholder (number "m")		+		PM-NC
TCI	P	Load tool from buffer into magazine		+	-	FM-TM
TCOABS [6)]	G	Determine tool length components from the current tool orientation	m	+		PM-NC
TCOFR	G	Determine tool length components from the orientation of the active frame	m	+		PM-NC
TCOFRX	G	Determine tool orientation of an active frame on selection of tool, tool points in X direction	m	+		PM-NC
TCOFRY	G	Determine tool orientation of an active frame on selection of tool, tool points in Y direction	m	+		PM-NC
TCOFRZ	G	Determine tool orientation of an active frame on selection of tool, tool points in Z direction	m	+		PM-NC
THETA	A	Angle of rotation	s	+		PM-NC
TILT	A	Tilt angle	m	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
TLIFT	P	Tangential control: Activate intermediate block generation		+	-	PM-NC
TML	P	Tool selection with magazine location number		+	-	FM-TM
TMOF	P	Deselect tool monitoring		+	-	PM-NC
TMON	P	Activate tool monitoring		+	-	PM-NC
TO	K	Designates the end value in a FOR counter loop		+		PM-NC
TOFF	A	Tool length offset in the direction of the tool length component that is effective parallel to the geometry axis specified in the index.	m	+		PM-NC
TOFFL	A	Tool length offset in the direction of the tool length component L1, L2 or L3	m	+		PM-NC
TOFFLR	A	Simultaneous tool length offset and tool radius offset	m	+		PM-NC
TOFFOF	P	Deactivate online tool offset		+	-	PM-NC
TOFFON	P	Activate online tool length offset		+	-	PM-NC
TOFFR	A	Tool radius offset	m	+		PM-NC
TOFRAME	G	Align Z axis of the WCS by rotating the frame parallel to the tool orientation	m	+		PM-NC
TOFRAMEX	G	Align X axis of the WCS by rotating the frame parallel to the tool orientation	m	+		PM-NC
TOFRAMEY	G	Align Y axis of the WCS by rotating the frame parallel to the tool orientation	m	+		PM-NC
TOFRAMEZ	G	As TOFRAME	m	+		PM-NC
TOLOWER	F	Convert the letters of a string into lowercase		+	-	PM-NC
TOOLENV	F	Save current states which are of significance to the evaluation of the tool data stored in the memory		+	-	PM-NC
TOOLGNT	F	Determine number of tools of a tool group		+	-	FM-TM
TOOLGT	F	Determine T number of a tool from a tool group		+	-	FM-TM
TOROT	G	Align Z axis of the WCS by rotating the frame parallel to the tool orientation	m	+		PM-NC
TOROTOF 6)	G	Frame rotations in tool direction OFF	m	+		PM-NC
TOROTX	G	Align X axis of the WCS by rotating the frame parallel to the tool orientation	m	+		PM-NC
TOROTY	G	Align Y axis of the WCS by rotating the frame parallel to the tool orientation	m	+		PM-NC
TOROTZ	G	As TOROT	m	+		PM-NC
TOUPPER	F	Convert the letters of a string into uppercase		+	-	PM-NC
TOWBCS	G	Wear values in the basic coordinate system (BCS)	m	+		PM-NC
TOWKCS	G	Wear values in the coordinate system of the tool head for kinetic transformation (differs from machine coordinate system through tool rotation)	m	+		PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
TOWMCS	G	Wear values in machine coordinate system (MCS)	m	+		PM-NC
TOWSTD 6)	G	Initial setting value for offsets in tool length	m	+		PM-NC
TOWTCS	G	Wear values in the tool coordinate system (toolholder ref. point T at the tool holder)	m	+		PM-NC
TOWWCS	G	Wear values in workpiece coordinate system (WCS)	m	+		PM-NC
TR	K	Offset component of a frame variable		+		PM-NC
TRAANG	P	Transformation inclined axis		+	-	PM-NC
TRACON	P	Cascaded transformation		+	-	PM-NC
TRACYL	P	Cylinder: Peripheral surface transformation		+	-	PM-NC
TRAFOOF	P	Deactivate active transformations in the channel		+	-	PM-NC
TRAFOON	P	Activate a transformation defined with kinematic chains		+	-	PM-NC
TRAILOF	P	Asynchronous coupled motion OFF		+	+	PM-NC
TRAILON	P	Asynchronous coupled motion ON		+	+	PM-NC
TRANS	G	Absolute programmable work offset	s	+		PM-NC
TRANSMIT	P	Pole transformation (face machining)		+	-	PM-NC
TRAORI	P	4-axis, 5-axis transformation, generic transformation		+	-	PM-NC
TRUE	K	Logical constant: True		+		PM-NC
TRUNC	F	Truncation of decimal places		+	+	PM-NC
TU		Axis angle	s	+		PM-NC
TURN	A	Number of turns for helix	s	+		PM-NC
ULI	K	Upper limit value of variables		+		PM-NC
UNLOCK	P	Enable synchronous action with ID (continue technology cycle)		-	+	FM-SA
UNTIL	K	Condition for end of REPEAT loop		+		PM-NC
UPATH	G	Path reference for FGROUP axes is curve parameter	m	+		PM-NC
VAR	K	Keyword: Type of parameter transfer		+		PM-NC
VELOLIM	K	Adapt maximum axis velocity or spindle speed	m	+	-	PM-NC
VELOLIMA	K	Reduction or overshoot of the maximum slave axis velocity	m	+	+	PM-NC
WAITC	P	Wait for the coupling block change criterion to be fulfilled for the axes/spindles		+	-	PM-NC
WAITE	P	Wait for end of program in another channel.		+	-	PM-NC
WAITENC	P	Wait for synchronized or restored axis positions		+	-	PM-NC
WAITM	P	Wait for marker in specified channel; terminate previous block with exact stop.		+	-	PM-NC
WAITMC	P	Wait for marker in specified channel; exact stop only if the other channels have not yet reached the marker.		+	-	PM-NC

Operation	Type 1)	Meaning	W 2)	TP 3)	SA 4)	Description see 5)
1) 2) 3) 4) 5) **for explanations, see legend (Page 1109).**						
WAITP	P	Wait for end of travel of the positioning axis		+	-	PM-NC
WAITS	P	Wait for spindle position to be reached		+	-	PM-NC
WALCS0 6)	G	Workpiece coordinate system working area limitation deselected	m	+	-	PM-NC
WALCS1	G	WCS working area limitation group 1 active	m	+	-	PM-NC
WALCS2	G	WCS working area limitation group 2 active	m	+	-	PM-NC
WALCS3	G	WCS working area limitation group 3 active	m	+	-	PM-NC
WALCS4	G	WCS working area limitation group 4 active	m	+	-	PM-NC
WALCS5	G	WCS working area limitation group 5 active	m	+	-	PM-NC
WALCS6	G	WCS working area limitation group 6 active	m	+	-	PM-NC
WALCS7	G	WCS working area limitation group 7 active	m	+	-	PM-NC
WALCS8	G	WCS working area limitation group 8 active	m	+	-	PM-NC
WALCS9	G	WCS working area limitation group 9 active	m	+	-	PM-NC
WALCS10	G	WCS working area limitation group 10 active	m	+	-	PM-NC
WALIMOF	G	BCS working area limitation OFF	m	+	-	PM-NC
WALIMON 6)	G	BCS working area limitation ON	m	+	-	PM-NC
WHEN	K	The action is executed once whenever the condition is fulfilled.		-	+	FM-SA
WHENEVER	K	The action is executed cyclically in each interpolator cycle when the condition is fulfilled.		-	+	FM-SA
WHILE	K	Start of WHILE program loop		+		PM-NC
WRITE	P	Write text to file system. Appends a block to the end of the specified file.		+	-	PM-NC
WRTPR	P	Write string in OPI variable		+	-	PM-NC
X	A	Axis name	m/s	+	+	PM-NC
XOR	O	Logic exclusive OR		+		PM-NC
Y	A	Axis name	m/s	+	+	PM-NC
Z	A	Axis name	m/s	+	+	PM-NC

5.1.6 Legend

[1] Type of operation:

A — Address
Identifier to which a value is assigned (e.g. OVR=10). There are also some addresses that switch on or off a function without value assignment (e.g. CPLON and CPLOF).

C (A) — AST cycle
Predefined NC program for automatic post optimization (tuning) with AST (= Automatic Servo Tuning). Parameters are used to adapt to the specific optimization situation; these parameters are transferred at the call.

C (M) — Measuring cycle
Predefined NC program in which a specific, generally valid, measuring operation, such as determining the inner diameter of a cylindrical workpiece, is programmed. Parameters are used to adapt to the specific measurement situation; these parameters are transferred at the call.

C (T) — Technological cycle
Predefined NC program in which a specific, generally valid, machining operation, such as tapping of a thread or milling a pocket, is programmed. The adaptation to a specific machine situation is realized via parameters that are transferred to the cycle during the call.

F — Predefined function (supplies a return value)
The call of the predefined function can be an operand in an expression.

G — G command
The G commands are divided into G groups. Only one G command of a group can be programmed in a block. A G command can be either modal (until it is canceled by another command of the same group) or only effective for the block in which it is programmed (non-modal).

K — Keyword
Identifier that defines the syntax of a block. No value is assigned to a keyword, and no NC function can be switched on/off with a keyword.
Examples: Control structures (IF, ELSE, ENDIF, WHEN, ...), program execution (GOTOB, GOTO, RET ...)

O — Operator
Operator for a mathematical, comparison or logical operation

P — Predefined procedure (does not supply a return value)

PA — Program attribute
Program attributes are at the end of the definition line of a subprogram:
```
PROC <program name>(...) <program attribute>
```
They determine the behavior during execution of the subprogram.

[2] Effectiveness of the operation:

m — Modal

s — Non-modal

[3] Programmability in part program:

\+ — Programmable

\- — Not programmable

M — Programmable only by the machine manufacturer

4) Programmability in synchronous actions:

+	Programmable
-	Not programmable
T	Programmable only in technology cycles

5) Reference to the document containing the detailed description of the operation:

FM-A	Function Manual Axes and Spindles
FM-B	Function Manual Basic Functions
FM-SA	Function Manual Synchronous Actions
FM-SI	Function Manual, Safety Integrated
FM-TE	Function Manual Technologies
FM-TM	Function Manual, Tool Management
FM-TR	Function Manual Transformations
PM-MC	Programming Manual Measuring Cycles
PM-NC	Programming Manual NC Programming

6) Default setting at beginning of program (factory settings of the control, if nothing else programmed).

5.2 Operations: Availability for SINUMERIK 828D

Note

Cycles

Cycles are marked as "optional" if they depend on the following options that require a license:

- Extended technology functions (article number: 6FC5800-0AP58-0YB0)
- Measuring cycles (article number: 6FC5800-0AP28-0YB0)
- Measuring kinematics (article number: 6FC5800-0AP18-0YB0)
- SINUMERIK Grinding Advanced (article number: 6FC5800-0AS35-0YB0)

Not marked, if cycles only contain partial functionalities as a result of option "Extended technology functions".

5.2.1 Control version milling / turning

Operations A ... C

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
:	●	●	●	●	●	●
*	●	●	●	●	●	●
+	●	●	●	●	●	●
-	●	●	●	●	●	●
<	●	●	●	●	●	●
<<	●	●	●	●	●	●
<=	●	●	●	●	●	●
=	●	●	●	●	●	●
>=	●	●	●	●	●	●
/	●	●	●	●	●	●
/0 ... /7	●	●	●	●	●	●
A	●	●	●	●	●	●
A2	-	-	-	-	-	-
A3	-	-	-	-	-	-
A4	-	-	-	-	-	-
A5	-	-	-	-	-	-
A6	-	-	-	-	-	-
A7	-	-	-	-	-	-
ABS	●	●	●	●	●	●
AC	●	●	●	●	●	●
ACC	●	●	●	●	●	●
ACCLIMA	●	●	●	●	●	●
ACN	●	●	●	●	●	●
ACOS	●	●	●	●	●	●
ACP	●	●	●	●	●	●
ACTBLOCNO	●	●	●	●	●	●
ADDFRAME	●	●	●	●	●	●
ADIS	●	●	●	●	●	●
ADISPOS	●	●	●	●	●	●
ADISPOSA	●	●	●	●	●	●
AFISOF	●	●	●	●	●	●
AFISON	●	●	●	●	●	●
ALF	●	●	●	●	●	●
AMIRROR	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
AND	●	●	●	●	●	●
ANG	●	●	●	●	●	●
AP	●	●	●	●	●	●
APR	●	●	●	●	●	●
APRB	●	●	●	●	●	●
APRP	●	●	●	●	●	●
APW	●	●	●	●	●	●
APWB	●	●	●	●	●	●
APWP	●	●	●	●	●	●
APX	●	●	●	●	●	●
AR	●	●	●	●	●	●
AROT	●	●	●	●	●	●
AROTS	●	●	●	●	●	●
AS	●	●	●	●	●	●
ASCALE	●	●	●	●	●	●
ASIN	●	●	●	●	●	●
ASPLINE	○	○	○	○	○	○
ATAN2	●	●	●	●	●	●
ATOL	●	●	●	●	●	●
ATRANS	●	●	●	●	●	●
AUXFUDEL	●	●	●	●	●	●
AUXFUDELG	●	●	●	●	●	●
AUXFUMSEQ	●	●	●	●	●	●
AUXFUSYNC	●	●	●	●	●	●
AX	●	●	●	●	●	●
AXCTSWE	-	-	-	-	-	-
AXCTSWEC	-	-	-	-	-	-
AXCTSWED	-	-	-	-	-	-
AXIS	●	●	●	●	●	●
AXNAME	●	●	●	●	●	●
AXSTRING	●	●	●	●	●	●
AXTOCHAN	●	●	●	●	●	●
AXTOSPI	●	●	●	●	●	●
B	●	●	●	●	●	●
B2	-	-	-	-	-	-
B3	-	-	-	-	-	-
B4	-	-	-	-	-	-
B5	-	-	-	-	-	-
B6	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
B7	-	-	-	-	-	-
B_AND	●	●	●	●	●	●
B_OR	●	●	●	●	●	●
B_NOT	●	●	●	●	●	●
B_XOR	●	●	●	●	●	●
BAUTO	○	○	○	○	○	○
BLOCK	●	●	●	●	●	●
BLSYNC	●	●	●	●	●	●
BNAT	○	○	○	○	○	○
BOOL	●	●	●	●	●	●
BOUND	●	●	●	●	●	●
BRISK	●	●	●	●	●	●
BRISKA	●	●	●	●	●	●
BSPLINE	○	○	○	○	○	○
BTAN	○	○	○	○	○	○
C	●	●	●	●	●	●
C2	-	-	-	-	-	Channel axis name
C3	-	-	-	-	-	-
C4	-	-	-	-	-	-
C5	-	-	-	-	-	-
C6	-	-	-	-	-	-
C7	-	-	-	-	-	-
CAC	●	●	●	●	●	●
CACN	●	●	●	●	●	●
CACP	●	●	●	●	●	●
CADAPTOF	○	○	○	○	○	○
CADAPTON	○	○	○	○	○	○
CALCDAT	●	●	●	●	●	●
CALCFIR	○	○	○	○	○	○
CALCPOSI	●	●	●	●	●	●
CALCTRAVAR	●	●	●	●	●	●
CALL	●	●	●	●	●	●
CALLPATH	●	●	●	●	●	●
CANCEL	●	●	●	●	●	●
CANCELSUB	●	●	●	●	●	●
CASE	●	●	●	●	●	●
CDC	●	●	●	●	●	●
CDOF	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
CDOF2	-	-	-	-	-	-
CDON	-	-	-	-	-	-
CFC	●	●	●	●	●	●
CFIN	●	●	●	●	●	●
CFINE	●	●	●	●	●	●
CFTCP	●	●	●	●	●	●
CHAN	●	●	●	●	●	●
CHANDATA	●	●	●	●	●	●
CHAR	●	●	●	●	●	●
CHF	●	●	●	●	●	●
CHKDM	●	●	●	●	●	●
CHKDNO	●	●	●	●	●	●
CHR	●	●	●	●	●	●
CIC	●	●	●	●	●	●
CIP	●	●	●	●	●	●
CLEARM	-	-	-	-	○	○
CLRINT	●	●	●	●	●	●
CMIRROR	●	●	●	●	●	●
COARSEA	●	●	●	●	●	●
COLLPAIR	●	●	●	●	●	●
COMPCAD	●	-	●	-	●	-
COMPCURV	●	-	●	-	●	-
COMPLETE	●	●	●	●	●	●
COMPOF	●	-	●	-	●	-
COMPON	●	-	●	-	●	-
COMPPATH	-	-	○	-	○	-
COMPSURF	-	-	○	-	○	-
CONTDCON	●	●	●	●	●	●
CONTPRON	●	●	●	●	●	●
CORROF	●	●	●	●	●	●
CORRTC						
CORRTRAFO	-	-	-	-	-	-
COS	●	●	●	●	●	●
COUPDEF	-	○	-	○	-	○
COUPDEL	-	○	-	○	-	○
COUPOF	-	○	-	○	-	○
COUPOFS	-	○	-	○	-	○
COUPON	-	○	-	○	-	○
COUPONC	-	○	-	○	-	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
COUPRES	-	○	-	○	-	○
CP	●	●	●	●	●	●
CPBC	○	○	○	○	○	○
CPDEF	○	○	○	○	○	○
CPDEL	○	○	○	○	○	○
CPFMOF	○	○	○	○	○	○
CPFMON	○	○	○	○	○	○
CPFMSON	○	○	○	○	○	○
CPFPOS	○	○	○	○	○	○
CPFRS	○	○	○	○	○	○
CPLA	○	○	○	○	○	○
CPLCTID	○	○	○	○	○	○
CPLDEF	○	○	○	○	○	○
CPLDEL	○	○	○	○	○	○
CPLDEN	○	○	○	○	○	○
CPLINSC	○	○	○	○	○	○
CPLINTR	○	○	○	○	○	○
CPLNUM	○	○	○	○	○	○
CPLOF	○	○	○	○	○	○
CPLON	○	○	○	○	○	○
CPLOUTSC	○	○	○	○	○	○
CPLOUTTR	○	○	○	○	○	○
CPLPOS	○	○	○	○	○	○
CPLSETVAL	○	○	○	○	○	○
CPMALARM	○	○	○	○	○	○
CPMBRAKE	○	○	○	○	○	○
CPMPRT	○	○	○	○	○	○
CPMRESET	○	○	○	○	○	○
CPMSTART	○	○	○	○	○	○
CPMVDI	○	○	○	○	○	○
CPOF	○	○	○	○	○	○
CPON	○	○	○	○	○	○
CPRECOF	●	●	●	●	●	●
CPRECON	●	●	●	●	●	●
CPRES	○	○	○	○	○	○
CPROT	●	●	●	●	●	●
CPROTDEF	●	●	●	●	●	●
CPSETTYPE	○	○	○	○	○	○
CPSYNCOP	○	○	○	○	○	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
CPSYNCOP2	○	○	○	○	○	○
CPSYNCOV	○	○	○	○	○	○
CPSYNFIP	○	○	○	○	○	○
CPSYNFIP2	○	○	○	○	○	○
CPSYNFIV	○	○	○	○	○	○
CR	●	●	●	●	●	●
CROT	●	●	●	●	●	●
CROTS	●	●	●	●	●	●
CRPL	●	●	●	●	●	●
CSCALE	●	●	●	●	●	●
CSPLINE	○	○	○	○	○	○
CT	●	●	●	●	●	●
CTAB	-	-	-	-	-	-
CTABDEF	-	-	-	-	-	-
CTABDEL	-	-	-	-	-	-
CTABEND	-	-	-	-	-	-
CTABEXISTS	-	-	-	-	-	-
CTABFNO	-	-	-	-	-	-
CTABFPOL	-	-	-	-	-	-
CTABFSEG	-	-	-	-	-	-
CTABID	-	-	-	-	-	-
CTABINV	-	-	-	-	-	-
CTABISLOCK	-	-	-	-	-	-
CTABLOCK	-	-	-	-	-	-
CTABMEMTYP	-	-	-	-	-	-
CTABMPOL	-	-	-	-	-	-
CTABMSEG	-	-	-	-	-	-
CTABNO	-	-	-	-	-	-
CTABNOMEM	-	-	-	-	-	-
CTABPERIOD	-	-	-	-	-	-
CTABPOL	-	-	-	-	-	-
CTABPOLID	-	-	-	-	-	-
CTABSEG	-	-	-	-	-	-
CTABSEGID	-	-	-	-	-	-
CTABSEV	-	-	-	-	-	-
CTABSSV	-	-	-	-	-	-
CTABTEP	-	-	-	-	-	-
CTABTEV	-	-	-	-	-	-
CTABTMAX	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
CTABTMIN	-	-	-	-	-	-
CTABTSP	-	-	-	-	-	-
CTABTSV	-	-	-	-	-	-
CTABUNLOCK	-	-	-	-	-	-
CTOL	●	●	●	●	●	●
CTOLG0	●	●	●	●	●	●
CTRANS	●	●	●	●	●	●
CUT2D	●	●	●	●	●	●
CUT2DD	●	●	●	●	●	●
CUT2DF	●	●	●	●	●	●
CUT2DFD	●	●	●	●	●	●
CUT3DC	-	-	-	-	-	-
CUT3DCC	-	-	-	-	-	-
CUT3DCCD	-	-	-	-	-	-
CUT3DCD	-	-	-	-	-	-
CUT3DF	-	-	-	-	-	-
CUT3DFD	-	-	-	-	-	-
CUT3DFF	-	-	-	-	-	-
CUT3DFS	-	-	-	-	-	-
CUTCONOF	●	●	●	●	●	●
CUTCONON	●	●	●	●	●	●
CUTMOD	●	●	●	●	●	●
CUTMODK	-	-	-	-	-	-
CYCLE60	○	○	●	●	●	●
CYCLE61	●	●	●	●	●	●
CYCLE62	●	●	●	●	●	●
CYCLE63	○	○	●	●	●	●
CYCLE64	○	○	●	●	●	●
CYCLE70	○	○	●	●	●	●
CYCLE72	●	●	●	●	●	●
CYCLE76	●	●	●	●	●	●
CYCLE77	●	●	●	●	●	●
CYCLE78	○	○	●	●	●	●
CYCLE79	○	○	●	●	●	●
CYCLE81	●	●	●	●	●	●
CYCLE82	●	●	●	●	●	●
CYCLE83	●	●	●	●	●	●
CYCLE84	●	●	●	●	●	●
CYCLE85	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
CYCLE86	●	●	●	●	●	●
CYCLE92	●	●	●	●	●	●
CYCLE95	●	●	●	●	●	●
CYCLE98	●	●	●	●	●	●
CYCLE99	●	●	●	●	●	●
CYCLE116	○	○	○	○	○	○
CYCLE119	○	○	○	○	○	○
CYCLE150	○	○	○	○	○	○
CYCLE435	-	-	-	-	-	-
CYCLE495	-	-	-	-	-	-
CYCLE750	●	●	●	●	●	●
CYCLE751	●	●	●	●	●	●
CYCLE752	●	●	●	●	●	●
CYCLE753	●	●	●	●	●	●
CYCLE754	●	●	●	●	●	●
CYCLE755	●	●	●	●	●	●
CYCLE756	●	●	●	●	●	●
CYCLE757	●	●	●	●	●	●
CYCLE758	●	●	●	●	●	●
CYCLE759	●	●	●	●	●	●
CYCLE782	○	○	○	○	○	○
CYCLE800	-	-	●	●	●	●
CYCLE801	●	●	●	●	●	●
CYCLE802	●	●	●	●	●	●
CYCLE806	○	○	○	○	○	○
CYCLE830	○	○	●	●	●	●
CYCLE832	●	●	●	●	●	●
CYCLE840	●	●	●	●	●	●
CYCLE899	○	○	●	●	●	●
CYCLE930	●	●	●	●	●	●
CYCLE940	●	●	●	●	●	●
CYCLE951	●	●	●	●	●	●
CYCLE952	○	○	●	●	●	●
CYCLE961	○	○	○	○	○	○
CYCLE971	○	○	○	○	○	○
CYCLE973	○	○	○	○	○	○
CYCLE974	○	○	○	○	○	○
CYCLE976	○	○	○	○	○	○
CYCLE977	○	○	○	○	○	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
CYCLE978	○	○	○	○	○	○
CYCLE979	○	○	○	○	○	○
CYCLE982	○	○	○	○	○	○
CYCLE994	○	○	○	○	○	○
CYCLE995	○	○	○	○	○	○
CYCLE996	○	-	○	-	○	-
CYCLE997	○	○	○	○	○	○
CYCLE998	○	○	○	○	○	○
CYCLE4071	-	-	-	-	-	-
CYCLE4072	-	-	-	-	-	-
CYCLE4073	-	-	-	-	-	-
CYCLE4074	-	-	-	-	-	-
CYCLE4075	-	-	-	-	-	-
CYCLE4077	-	-	-	-	-	-
CYCLE4078	-	-	-	-	-	-
CYCLE4079	-	-	-	-	-	-
CYCLE9960	-	-	-	-	-	-

Operations D ... F

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
D	●	●	●	●	●	●
D0	●	●	●	●	●	●
DAC	●	●	●	●	●	●
DC	●	●	●	●	●	●
DCI	●	●	●	●	●	●
DCM	●	●	●	●	●	●
DCU	●	●	●	●	●	●
DEF	●	●	●	●	●	●
DEFINE	●	●	●	●	●	●
DEFAULT	●	●	●	●	●	●
DELAYFSTON	●	●	●	●	●	●
DELAYFSTOF	●	●	●	●	●	●
DELDL	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
DELDTG	●	●	●	●	●	●
DELETE	●	●	●	●	●	●
DELMLOWNER	●	●	●	●	●	●
DELMLRES	●	●	●	●	●	●
DELMT	●	●	●	●	●	●
DELOBJ	-	-	-	-	-	-
DELT	●	●	●	●	●	●
DELTC	●	●	●	●	●	●
DELTOOLENV	●	●	●	●	●	●
DIACYCOFA	●	●	●	●	●	●
DIAM90	●	●	●	●	●	●
DIAM90A	●	●	●	●	●	●
DIAMCHAN	●	●	●	●	●	●
DIAMCHANA	●	●	●	●	●	●
DIAMCYCOF	●	●	●	●	●	●
DIAMOF	●	●	●	●	●	●
DIAMOFA	●	●	●	●	●	●
DIAMON	●	●	●	●	●	●
DIAMONA	●	●	●	●	●	●
DIC	●	●	●	●	●	●
DILF	●	●	●	●	●	●
DISABLE	●	●	●	●	●	●
DISC	●	●	●	●	●	●
DISCL	●	●	●	●	●	●
DISPLOF	●	●	●	●	●	●
DISPLON	●	●	●	●	●	●
DISPR	●	●	●	●	●	●
DISR	●	●	●	●	●	●
DISRP	●	●	●	●	●	●
DITE	●	●	●	●	●	●
DITS	●	●	●	●	●	●
DIV	●	●	●	●	●	●
DL	-	-	-	-	-	-
DO	●	●	●	●	●	●
DRFOF	●	●	●	●	●	●
DRIVE	●	●	●	●	●	●
DRIVEA	●	●	●	●	●	●
DRVPRD	●	●	●	●	●	●
DRVPWR	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
DYNFINISH	●	●	●	●	●	●
DYNNORM	●	●	●	●	●	●
DYNPOS	●	●	●	●	●	●
DYNPREC	●	●	●	●	●	●
DYNROUGH	●	●	●	●	●	●
DYNSEMIFIN	●	●	●	●	●	●
DZERO	●	●	●	●	●	●
EAUTO	○	○	○	○	○	○
EGDEF	-	○	-	○	-	○
EGDEL	-	○	-	○	-	○
EGOFC	-	○	-	○	-	○
EGOFS	-	○	-	○	-	○
EGON	-	○	-	○	-	○
EGONSYN	-	○	-	○	-	○
EGONSYNE	-	○	-	○	-	○
ELSE	●	●	●	●	●	●
ENABLE	●	●	●	●	●	●
ENAT	○	○	○	○	○	○
ENDFOR	●	●	●	●	●	●
ENDIF	●	●	●	●	●	●
ENDLABEL	●	●	●	●	●	●
ENDLOOP	●	●	●	●	●	●
ENDPROC	●	●	●	●	●	●
ENDWHILE	●	●	●	●	●	●
ESRR	○	○	○	○	○	○
ESRS	○	○	○	○	○	○
ETAN	○	○	○	○	○	○
EVERY	●	●	●	●	●	●
EX	●	●	●	●	●	●
EXECSTRING	●	●	●	●	●	●
EXECTAB	●	●	●	●	●	●
EXECUTE	●	●	●	●	●	●
EXP	●	●	●	●	●	●
EXTCALL	●	●	●	●	●	●
EXTCLOSE	●	●	●	●	●	●
EXTERN	●	●	●	●	●	●
EXTOPEN	●	●	●	●	●	●
F	●	●	●	●	●	●
FA	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
FAD	●	●	●	●	●	●
FALSE	●	●	●	●	●	●
FB	●	●	●	●	●	●
FCTDEF	●	●	●	●	●	●
FCUB	●	●	●	●	●	●
FD	●	●	●	●	●	●
FDA	●	●	●	●	●	●
FENDNORM	●	●	●	●	●	●
FFWOF	●	●	●	●	●	●
FFWON	●	●	●	●	●	●
FGREF	●	●	●	●	●	●
FGROUP	●	●	●	●	●	●
FI	●	●	●	●	●	●
FIFOCTRL	●	●	●	●	●	●
FILEDATE	●	●	●	●	●	●
FILEINFO	●	●	●	●	●	●
FILESIZE	●	●	●	●	●	●
FILESTAT	●	●	●	●	●	●
FILETIME	●	●	●	●	●	●
FINEA	●	●	●	●	●	●
FL	●	●	●	●	●	●
FLIM	●	●	●	●	●	●
FLIN	●	●	●	●	●	●
FMA	●	●	●	●	●	●
FNORM	●	●	●	●	●	●
FOCOF	○	○	○	○	○	○
FOCON	○	○	○	○	○	○
FOR	●	●	●	●	●	●
FP	●	●	●	●	●	●
FPO	-	-	-	-	-	-
FPR	●	●	●	●	●	●
FPRAOF	●	●	●	●	●	●
FPRAON	●	●	●	●	●	●
FRAME	●	●	●	●	●	●
FRC	●	●	●	●	●	●
FRCM	●	●	●	●	●	●
FROM	●	●	●	●	●	●
FTOC	●	●	●	●	●	●
FTOCOF	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
FTOCON	●	●	●	●	●	●
FXS	●	●	●	●	●	●
FXST	●	●	●	●	●	●
FXSW	●	●	●	●	●	●
FZ	●	●	●	●	●	●

Operations G ... L

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
G0	●	●	●	●	●	●
G1	●	●	●	●	●	●
G2	●	●	●	●	●	●
G3	●	●	●	●	●	●
G4	●	●	●	●	●	●
G5	●	●	●	●	●	●
G7	●	●	●	●	●	●
G9	●	●	●	●	●	●
G17	●	●	●	●	●	●
G18	●	●	●	●	●	●
G19	●	●	●	●	●	●
G25	●	●	●	●	●	●
G26	●	●	●	●	●	●
G33	●	●	●	●	●	●
G34	●	●	●	●	●	●
G35	●	●	●	●	●	●
G40	●	●	●	●	●	●
G41	●	●	●	●	●	●
G42	●	●	●	●	●	●
G53	●	●	●	●	●	●
G54	●	●	●	●	●	●
G55	●	●	●	●	●	●
G56	●	●	●	●	●	●
G57	●	●	●	●	●	●
G58	→ G505					

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
G59	→ G506					
G60	●	●	●	●	●	●
G62	●	●	●	●	●	●
G63	●	●	●	●	●	●
G64	●	●	●	●	●	●
G70	●	●	●	●	●	●
G71	●	●	●	●	●	●
G74	●	●	●	●	●	●
G75	●	●	●	●	●	●
G90	●	●	●	●	●	●
G91	●	●	●	●	●	●
G93	●	●	●	●	●	●
G94	●	●	●	●	●	●
G95	●	●	●	●	●	●
G96	●	●	●	●	●	●
G97	●	●	●	●	●	●
G110	●	●	●	●	●	●
G111	●	●	●	●	●	●
G112	●	●	●	●	●	●
G140	●	●	●	●	●	●
G141	●	●	●	●	●	●
G142	●	●	●	●	●	●
G143	●	●	●	●	●	●
G147	●	●	●	●	●	●
G148	●	●	●	●	●	●
G153	●	●	●	●	●	●
G247	●	●	●	●	●	●
G248	●	●	●	●	●	●
G290	●	●	●	●	●	●
G291	●	●	●	●	●	●
G331	●	●	●	●	●	●
G332	●	●	●	●	●	●
G335	●	●	●	●	●	●
G336	●	●	●	●	●	●
G340	●	●	●	●	●	●
G341	●	●	●	●	●	●
G347	●	●	●	●	●	●
G348	●	●	●	●	●	●
G450	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
G451	●	●	●	●	●	●
G460	●	●	●	●	●	●
G461	●	●	●	●	●	●
G462	●	●	●	●	●	●
G500	●	●	●	●	●	●
G505 ... G599	●	●	●	●	●	●
G601	●	●	●	●	●	●
G602	●	●	●	●	●	●
G603	●	●	●	●	●	●
G621	●	●	●	●	●	●
G641	●	●	●	●	●	●
G642	●	●	●	●	●	●
G643	●	●	●	●	●	●
G644	●	●	●	●	●	●
G645	●	●	●	●	●	●
G646	○	○	○	○	○	○
G700	●	●	●	●	●	●
G710	●	●	●	●	●	●
G810 ... G819	-	-	-	-	-	-
G820 ... G829	-	-	-	-	-	-
G931	●	●	●	●	●	●
G942	●	●	●	●	●	●
G952	●	●	●	●	●	●
G961	●	●	●	●	●	●
G962	●	●	●	●	●	●
G971	●	●	●	●	●	●
G972	●	●	●	●	●	●
G973	●	●	●	●	●	●
GEOAX	●	●	●	●	●	●
GET	●	●	●	●	●	●
GETACTT	●	●	●	●	●	●
GETACTTD	●	●	●	●	●	●
GETD	-	-	-	-	○	○
GETDNO	●	●	●	●	●	●
GETEXET	●	●	●	●	●	●
GETFREELOC	●	●	●	●	●	●
GETSELT	●	●	●	●	●	●
GETT	●	●	●	●	●	●
GETTCOR	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
GETTENV	●	●	●	●	●	●
GETVARAP	●	●	●	●	●	●
GETVARDFT	●	●	●	●	●	●
GETVARLIM	●	●	●	●	●	●
GETVARPHU	●	●	●	●	●	●
GETVARTYP	●	●	●	●	●	●
GFRAME0 ... GFRAME100	-	-	-	-	-	-
GOTO	●	●	●	●	●	●
GOTOB	●	●	●	●	●	●
GOTOC	●	●	●	●	●	●
GOTOF	●	●	●	●	●	●
GOTOS	●	●	●	●	●	●
GP	●	●	●	●	●	●
GWPSOF	●	●	●	●	●	●
GROUP_ ADDEND	●	●	●	●	●	●
GROUP_BEGIN	●	●	●	●	●	●
GROUP_END	●	●	●	●	●	●
GWPSON	●	●	●	●	●	●
H...	●	●	●	●	●	●
HOLES1	●	●	●	●	●	●
HOLES2	●	●	●	●	●	●
I	●	●	●	●	●	●
I1	●	●	●	●	●	●
IC	●	●	●	●	●	●
ICYCOF	●	●	●	●	●	●
ICYCON	●	●	●	●	●	●
ID	●	●	●	●	●	●
IDS	●	●	●	●	●	●
IF	●	●	●	●	●	●
INDEX	●	●	●	●	●	●
INIPO	●	●	●	●	●	●
INIRE	●	●	●	●	●	●
INICF	●	●	●	●	●	●
INIT	-	-	-	-	○	○
INITIAL						
INT	●	●	●	●	●	●
INTERSEC	●	●	●	●	●	●
INVCCW	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
INVCW	-	-	-	-	-	-
INVFRAME	●	●	●	●	●	●
IP	●	●	●	●	●	●
IPOBRKA	●	●	●	●	●	●
IPOENDA	●	●	●	●	●	●
IPTRLOCK	●	●	●	●	●	●
IPTRUNLOCK	●	●	●	●	●	●
IR	●	●	●	●	●	●
ISAXIS	●	●	●	●	●	●
ISD	-	-	-	-	-	-
ISFILE	●	●	●	●	●	●
ISNUMBER	●	●	●	●	●	●
ISOCALL	●	●	●	●	●	●
ISVAR	●	●	●	●	●	●
J	●	●	●	●	●	●
J1	●	●	●	●	●	●
JERKA	●	●	●	●	●	●
JERKLIM	●	●	●	●	●	●
JERKLIMA	●	●	●	●	●	●
JR	●	●	●	●	●	●
K	●	●	●	●	●	●
K1	●	●	●	●	●	●
KONT	●	●	●	●	●	●
KONTC	●	●	●	●	●	●
KONTT	●	●	●	●	●	●
KR	●	●	●	●	●	●
L	●	●	●	●	●	●
LEAD Tool orientation Orientation polyn.	- -	- -	- -	- -	- -	- -
LEADOF	-	-	-	●	-	●
LEADON	-	-	-	●	-	●
LENTOAX	●	●	●	●	●	●
LFOF	●	●	●	●	●	●
LFON	●	●	●	●	●	●
LFPOS	●	●	●	●	●	●
LFTXT	●	●	●	●	●	●
LFWP	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
LIFTFAST	●	●	●	●	●	●
LIMS	●	●	●	●	●	●
LLI	●	●	●	●	●	●
LN	●	●	●	●	●	●
LOCK	●	●	●	●	●	●
LONGHOLE	●	●	●	●	●	●
LOOP	●	●	●	●	●	●

Operations M ... R

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
M0	●	●	●	●	●	●
M1	●	●	●	●	●	●
M2	●	●	●	●	●	●
M3	●	●	●	●	●	●
M4	●	●	●	●	●	●
M5	●	●	●	●	●	●
M6	●	●	●	●	●	●
M17	●	●	●	●	●	●
M19	●	●	●	●	●	●
M30	●	●	●	●	●	●
M40	●	●	●	●	●	●
M41 ... M45	●	●	●	●	●	●
M70	●	●	●	●	●	●
MASLDEF	○	○	○	○	○	○
MASLDEL	○	○	○	○	○	○
MASLOF	○	○	○	○	○	○
MASLOFS	○	○	○	○	○	○
MASLON	○	○	○	○	○	○
MATCH	●	●	●	●	●	●
MAXVAL	●	●	●	●	●	●
MCALL	●	●	●	●	●	●
MCALLOF	●	●	●	●	●	●
MEAC	○	○	○	○	○	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
MEAFRAME	●	●	●	●	●	●
MEAS	●	●	●	●	●	●
MEASA	○	○	○	○	○	○
MEASF	●	●	●	●	●	●
MEASURE	●	●	●	●	●	●
MEAW	●	●	●	●	●	●
MEAWA	○	○	○	○	○	○
MI	●	●	●	●	●	●
MINDEX	●	●	●	●	●	●
MINVAL	●	●	●	●	●	●
MIRROR	●	●	●	●	●	●
MMC	●	●	●	●	●	●
MOD	●	●	●	●	●	●
MODAXVAL	●	●	●	●	●	●
MOV	●	●	●	●	●	●
MOVT	●	●	●	●	●	●
MSG	●	●	●	●	●	●
MVTOOL	●	●	●	●	●	●
N	●	●	●	●	●	●
NAMETOINT	●	●	●	●	●	●
NC	●	●	●	●	●	●
NEWCONF	●	●	●	●	●	●
NEWMT	●	●	●	●	●	●
NEWT	●	●	●	●	●	●
NORM	●	●	●	●	●	●
NOT	●	●	●	●	●	●
NPROT	●	●	●	●	●	●
NPROTDEF	●	●	●	●	●	●
NUMBER	●	●	●	●	●	●
OEMIPO1	-	-	-	-	-	-
OEMIPO2	-	-	-	-	-	-
OF	●	●	●	●	●	●
OFFN	●	●	●	●	●	●
OMA1	-	-	-	-	-	-
OMA2	-	-	-	-	-	-
OMA3	-	-	-	-	-	-
OMA4	-	-	-	-	-	-
OMA5	-	-	-	-	-	-
OR	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
ORIANGLE	-	-	-	-	-	-
ORIAXES	-	-	-	-	-	-
ORIAXESFR	-	-	-	-	-	-
ORIAXPOS	-	-	-	-	-	-
ORIC	-	-	-	-	-	-
ORICONCCW	-	-	-	-	-	-
ORICONCW	-	-	-	-	-	-
ORICONIO	-	-	-	-	-	-
ORICONTO	-	-	-	-	-	-
ORICURINV	-	-	-	-	-	-
ORICURVE	-	-	-	-	-	-
ORID	-	-	-	-	-	-
ORIEULER	-	-	-	-	-	-
ORIMKS	-	-	-	-	-	-
ORIPATH	-	-	-	-	-	-
ORIPATHS	-	-	-	-	-	-
ORIPLANE	-	-	-	-	-	-
ORIRESET	-	-	-	-	-	-
ORIROTA	-	-	-	-	-	-
ORIROTC	-	-	-	-	-	-
ORIROTR	-	-	-	-	-	-
ORIROTT	-	-	-	-	-	-
ORIRPY	-	-	-	-	-	-
ORIRPY2	-	-	-	-	-	-
ORIS	-	-	-	-	-	-
ORISOF	-	-	-	-	-	-
ORISOLH	-	-	-	-	-	-
ORISON	-	-	-	-	-	-
ORIVECT	-	-	-	-	-	-
ORIVIRT1	-	-	-	-	-	-
ORIVIRT2	-	-	-	-	-	-
ORIWKS	-	-	-	-	-	-
OS	-	-	-	-	-	-
OSB	-	-	-	-	-	-
OSC	-	-	-	-	-	-
OSCILL	-	-	-	-	-	-
OSCTRL	-	-	-	-	-	-
OSD	-	-	-	-	-	-
OSE	-	-	-	-	-	-

Operation	**SINUMERIK 828D**					
● Standard **○ Option** **- not available**	**SW24x (5) CNC SW Milling Export (me42)**	**SW24x (5) CNC SW Turning Export (te42)**	**SW26x (3) CNC SW Milling Export (me62)**	**SW26x (3) CNC SW Turning Export (te62)**	**SW28x (1) CNC SW Milling Export (me82)**	**SW28x (1) CNC SW Turning Export (te82)**
OSNSC	●	●	●	●	●	●
OSOF	-	-	-	-	-	-
OSP1	●	●	●	●	●	●
OSP2	●	●	●	●	●	●
OSS	-	-	-	-	-	-
OSSE	-	-	-	-	-	-
OST	-	-	-	-	-	-
OST1	●	●	●	●	●	●
OST2	●	●	●	●	●	●
OTOL	●	●	●	●	●	●
OTOLG0	●	●	●	●	●	●
OVR	●	●	●	●	●	●
OVRA	●	●	●	●	●	●
OVRRAP	●	●	●	●	●	●
P	●	●	●	●	●	●
PACCLIM	○	○	○	○	○	○
PAROT	●	●	●	●	●	●
PAROTOF	●	●	●	●	●	●
PCALL	●	●	●	●	●	●
PDELAYOF	-	-	-	-	-	-
PDELAYON	-	-	-	-	-	-
PHI	-	-	-	-	-	-
PHU	●	●	●	●	●	●
PL	-	-	-	-	-	-
PM	●	●	●	●	●	●
PO	-	-	-	-	-	-
POCKET3	●	●	●	●	●	●
POCKET4	●	●	●	●	●	●
POLF	●	●	●	●	●	●
POLFA	●	●	●	●	●	●
POLFMASK	●	●	●	●	●	●
POLFMLIN	●	●	●	●	●	●
POLY	-	-	-	-	-	-
POLYPATH	-	-	-	-	-	-
PON	-	-	-	-	-	-
PONS	-	-	-	-	-	-
POS	●	●	●	●	●	●
POSA	●	●	●	●	●	●
POSM	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
POSMT	●	●	●	●	●	●
POSP	●	●	●	●	●	●
POSRANGE	●	●	●	●	●	●
POT	●	●	●	●	●	●
PR	●	●	●	●	●	●
PREPRO	●	●	●	●	●	●
PRESETON	●	●	●	●	●	●
PRESETONS	●	●	●	●	●	●
PRIO	●	●	●	●	●	●
PRLOC	●	●	●	●	●	●
PROC	●	●	●	●	●	●
PROTA	●	●	●	●	●	●
PROTD	●	●	●	●	●	●
PROTS	●	●	●	●	●	●
PSI	-	-	-	-	-	-
PTP	●	●	●	●	●	●
PTPG0	●	●	●	●	●	●
PTPWOC	●	●	●	●	●	●
PUNCHACC	-	-	-	-	-	-
PUTFTOC	●	●	●	●	●	●
PUTFTOCF	●	●	●	●	●	●
PW	○	○	○	○	○	○
QU	●	●	●	●	●	●
R...	●	●	●	●	●	●
RAC	●	●	●	●	●	●
RDISABLE	●	●	●	●	●	●
READ	●	●	●	●	●	●
REAL	●	●	●	●	●	●
RELEASE	●	●	●	●	●	●
REP	●	●	●	●	●	●
REPEAT	●	●	●	●	●	●
REPEATB	●	●	●	●	●	●
REPOSA	●	●	●	●	●	●
REPOSH	●	●	●	●	●	●
REPOSHA	●	●	●	●	●	●
REPOSL	●	●	●	●	●	●
REPOSQ	●	●	●	●	●	●
REPOSQA	●	●	●	●	●	●
RESETMON	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
RET	●	●	●	●	●	●
RETB	●	●	●	●	●	●
RIC	●	●	●	●	●	●
RINDEX	●	●	●	●	●	●
RMB	●	●	●	●	●	●
RME	●	●	●	●	●	●
RMI	●	●	●	●	●	●
RMN	●	●	●	●	●	●
RND	●	●	●	●	●	●
RNDM	●	●	●	●	●	●
ROT	●	●	●	●	●	●
ROTS	●	●	●	●	●	●
ROUND	●	●	●	●	●	●
ROUNDUP	●	●	●	●	●	●
RP	●	●	●	●	●	●
RPL	●	●	●	●	●	●
RT	●	●	●	●	●	●
RTLIOF	●	●	●	●	●	●
RTLION	●	●	●	●	●	●

Operations S ... Z

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
S	●	●	●	●	●	●
SAVE	●	●	●	●	●	●
SBLOF	●	●	●	●	●	●
SBLON	●	●	●	●	●	●
SC	●	●	●	●	●	●
SCALE	●	●	●	●	●	●
SCC	●	●	●	●	●	●
SCPARA	●	●	●	●	●	●
SD	○	○	○	○	○	○
SET	●	●	●	●	●	●
SETAL	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
SETDNO	●	●	●	●	●	●
SETINT	●	●	●	●	●	●
SETM	-	-	-	-	○	○
SETMS	●	●	●	●	●	●
SETMS(n)	●	●	●	●	●	●
SETMTH	●	●	●	●	●	●
SETPIECE	●	●	●	●	●	●
SETTA	●	●	●	●	●	●
SETTCOR	●	●	●	●	●	●
SETTIA	●	●	●	●	●	●
SF	●	●	●	●	●	●
SIN	●	●	●	●	●	●
SIRELAY	-	-	-	-	-	-
SIRELIN	-	-	-	-	-	-
SIRELOUT	-	-	-	-	-	-
SIRELTIME	-	-	-	-	-	-
SLOT1	●	●	●	●	●	●
SLOT2	●	●	●	●	●	●
SOFT	●	●	●	●	●	●
SOFTA	●	●	●	●	●	●
SON	-	-	-	-	-	-
SONS	-	-	-	-	-	-
SPATH	●	●	●	●	●	●
SPCOF	●	●	●	●	●	●
SPCON	●	●	●	●	●	●
SPI	●	●	●	●	●	●
SPIF1	-	-	-	-	-	-
SPIF2	-	-	-	-	-	-
SPLINEPATH	○	○	○	○	○	○
SPN	-	-	-	-	-	-
SPOF	-	-	-	-	-	-
SPOS	●	●	●	●	●	●
SPOSA	●	●	●	●	●	●
SPP	-	-	-	-	-	-
SPRINT	●	●	●	●	●	●
SQRT	●	●	●	●	●	●
SR	●	●	●	●	●	●
SRA	●	●	●	●	●	●
ST	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
STA	●	●	●	●	●	●
START	-	-	-	-	○	○
STARTFIFO	●	●	●	●	●	●
STAT	●	●	●	●	●	●
STOLF	●	●	●	●	●	●
STOPFIFO	●	●	●	●	●	●
STOPRE	●	●	●	●	●	●
STOPREOF	●	●	●	●	●	●
STRING	●	●	●	●	●	●
STRINGFELD	●	●	●	●	●	●
STRINGIS	●	●	●	●	●	●
STRLEN	●	●	●	●	●	●
SUBSTR	●	●	●	●	●	●
SUPA	●	●	●	●	●	●
SUPD	●	●	●	●	●	●
SVC	●	●	●	●	●	●
SYNFCT	●	●	●	●	●	●
SYNR	●	●	●	●	●	●
SYNRW	●	●	●	●	●	●
SYNW	●	●	●	●	●	●
T	●	●	●	●	●	●
TAN	●	●	●	●	●	●
TANG	-	-	-	-	-	-
TANGDEL	-	-	-	-	-	-
TANGOF	-	-	-	-	-	-
TANGON	-	-	-	-	-	-
TCA (828D: _TCA)	●	●	●	●	●	●
TCARR	●	-	●	-	●	-
TCI	●	●	●	●	●	●
TCOABS	●	-	●	-	●	-
TCOFR	●	-	●	-	●	-
TCOFRX	●	-	●	-	●	-
TCOFRY	●	-	●	-	●	-
TCOFRZ	●	-	●	-	●	-
THETA	-	-	-	-	-	-
TILT	-	-	-	-	-	-
TLIFT	-	-	-	-	-	-
TML	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
TMOF	●	●	●	●	●	●
TMON	●	●	●	●	●	●
TO	●	●	●	●	●	●
TOFF	●	●	●	●	●	●
TOFFL	●	●	●	●	●	●
TOFFLR	●	●	●	●	●	●
TOFFOF	-	-	-	-	-	-
TOFFON	-	-	-	-	-	-
TOFFR	●	●	●	●	●	●
TOFRAME	●	●	●	●	●	●
TOFRAMEX	●	●	●	●	●	●
TOFRAMEY	●	●	●	●	●	●
TOFRAMEZ	●	●	●	●	●	●
TOLOWER	●	●	●	●	●	●
TOOLENV	●	●	●	●	●	●
TOOLGNT	●	●	●	●	●	●
TOOLGT	●	●	●	●	●	●
TOROT	●	●	●	●	●	●
TOROTOF	●	●	●	●	●	●
TOROTX	●	●	●	●	●	●
TOROTY	●	●	●	●	●	●
TOROTZ	●	●	●	●	●	●
TOUPPER	●	●	●	●	●	●
TOWBCS	●	-	●	-	●	-
TOWKCS	●	-	●	-	●	-
TOWMCS	●	-	●	-	●	-
TOWSTD	●	-	●	-	●	-
TOWTCS	●	-	●	-	●	-
TOWWCS	●	-	●	-	●	-
TR	●	●	●	●	●	●
TRAANG	-	-	-	-	-	○
TRACON	-	-	-	-	-	○
TRACYL	○	○	○	○	○	○
TRAFOOF	●	●	●	●	●	●
TRAFOON	-	-	-	-	-	-
TRAILOF	●	●	●	●	●	●
TRAILON	●	●	●	●	●	●
TRANS	●	●	●	●	●	●
TRANSMIT	○	○	○	○	○	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
TRAORI	-	-	-	-	-	-
TRUE	●	●	●	●	●	●
TRUNC	●	●	●	●	●	●
TU	●	●	●	●	●	●
TURN	●	●	●	●	●	●
ULI	●	●	●	●	●	●
UNLOCK	●	●	●	●	●	●
UNTIL	●	●	●	●	●	●
UPATH	●	●	●	●	●	●
VAR	●	●	●	●	●	●
VELOLIM	●	●	●	●	●	●
VELOLIMA	●	●	●	●	●	●
WAITC	●	●	●	●	●	●
WAITE	-	-	-	-	○	○
WAITENC	●	●	●	●	●	●
WAITM	-	-	-	-	○	○
WAITMC	-	-	-	-	○	○
WAITP	●	●	●	●	●	●
WAITS	●	●	●	●	●	●
WALCS0	●	●	●	●	●	●
WALCS1	●	●	●	●	●	●
WALCS2	●	●	●	●	●	●
WALCS3	●	●	●	●	●	●
WALCS4	●	●	●	●	●	●
WALCS5	●	●	●	●	●	●
WALCS6	●	●	●	●	●	●
WALCS7	●	●	●	●	●	●
WALCS8	●	●	●	●	●	●
WALCS9	●	●	●	●	●	●
WALCS10	●	●	●	●	●	●
WALIMOF	●	●	●	●	●	●
WALIMON	●	●	●	●	●	●
WHEN	●	●	●	●	●	●
WHENEVER	●	●	●	●	●	●
WHILE	●	●	●	●	●	●
WORKPIECE	●	●	●	●	●	●
WRITE	●	●	●	●	●	●
WRTPR	●	●	●	●	●	●
X	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW Milling Export (me42)	SW24x (5) CNC SW Turning Export (te42)	SW26x (3) CNC SW Milling Export (me62)	SW26x (3) CNC SW Turning Export (te62)	SW28x (1) CNC SW Milling Export (me82)	SW28x (1) CNC SW Turning Export (te82)
XOR	●	●	●	●	●	●
Y	●	●	●	●	●	●
Z	●	●	●	●	●	●

5.2.2 Control versions grinding

Operations A ... C

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
:	●	●	●	●	●	●
*	●	●	●	●	●	●
+	●	●	●	●	●	●
-	●	●	●	●	●	●
<	●	●	●	●	●	●
<<	●	●	●	●	●	●
<=	●	●	●	●	●	●
=	●	●	●	●	●	●
>=	●	●	●	●	●	●
/	●	●	●	●	●	●
/0 ... /7	●	●	●	●	●	●
A	●	●	●	●	●	●
A2	-	-	-	-	-	-
A3	-	-	-	-	-	-
A4	-	-	-	-	-	-
A5	-	-	-	-	-	-
A6	-	-	-	-	-	-
A7	-	-	-	-	-	-
ABS	●	●	●	●	●	●
AC	●	●	●	●	●	●
ACC	●	●	●	●	●	●
ACCLIMA	●	●	●	●	●	●
ACN	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
ACOS	●	●	●	●	●	●
ACP	●	●	●	●	●	●
ACTBLOCNO	●	●	●	●	●	●
ADDFRAME	●	●	●	●	●	●
ADIS	●	●	●	●	●	●
ADISPOS	●	●	●	●	●	●
ADISPOSA	●	●	●	●	●	●
AFISOF	●	●	●	●	●	●
AFISON	●	●	●	●	●	●
ALF	●	●	●	●	●	●
AMIRROR	●	●	●	●	●	●
AND	●	●	●	●	●	●
ANG	●	●	●	●	●	●
AP	●	●	●	●	●	●
APR	●	●	●	●	●	●
APRB	●	●	●	●	●	●
APRP	●	●	●	●	●	●
APW	●	●	●	●	●	●
APWB	●	●	●	●	●	●
APWP	●	●	●	●	●	●
APX	●	●	●	●	●	●
AR	●	●	●	●	●	●
AROT	●	●	●	●	●	●
AROTS	●	●	●	●	●	●
AS	●	●	●	●	●	●
ASCALE	●	●	●	●	●	●
ASIN	●	●	●	●	●	●
ASPLINE	○	○	○	○	○	○
ATAN2	●	●	●	●	●	●
ATOL	●	●	●	●	●	●
ATRANS	●	●	●	●	●	●
AUXFUDEL	●	●	●	●	●	●
AUXFUDELG	●	●	●	●	●	●
AUXFUMSEQ	●	●	●	●	●	●
AUXFUSYNC	●	●	●	●	●	●
AX	●	●	●	●	●	●
AXCTSWE	-	-	-	-	-	-
AXCTSWEC	-	-	-	-	-	-
AXCTSWED	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
AXIS	●	●	●	●	●	●
AXNAME	●	●	●	●	●	●
AXSTRING	●	●	●	●	●	●
AXTOCHAN	●	●	●	●	●	●
AXTOSPI	●	●	●	●	●	●
B	●	●	●	●	●	●
B2	-	-	-	-	-	-
B3	-	-	-	-	-	-
B4	-	-	-	-	-	-
B5	-	-	-	-	-	-
B6	-	-	-	-	-	-
B7	-	-	-	-	-	-
B_AND	●	●	●	●	●	●
B_OR	●	●	●	●	●	●
B_NOT	●	●	●	●	●	●
B_XOR	●	●	●	●	●	●
BAUTO	○	○	○	○	○	○
BLOCK	●	●	●	●	●	●
BLSYNC	●	●	●	●	●	●
BNAT	○	○	○	○	○	○
BOOL	●	●	●	●	●	●
BOUND	●	●	●	●	●	●
BRISK	●	●	●	●	●	●
BRISKA	●	●	●	●	●	●
BSPLINE	○	○	○	○	○	○
BTAN	○	○	○	○	○	○
C	●	●	●	●	●	●
C2	-	-	Channel axis name	-	-	-
C3	-	-	-	-	-	-
C4	-	-	-	-	-	-
C5	-	-	-	-	-	-
C6	-	-	-	-	-	-
C7	-	-	-	-	-	-
CAC	●	●	●	●	●	●
CACN	●	●	●	●	●	●
CACP	●	●	●	●	●	●
CADAPTOF	○	○	○	○	○	○
CADAPTON	○	○	○	○	○	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
CALCDAT	●	●	●	●	●	●
CALCFIR	○	○	○	○	○	○
CALCPOSI	●	●	●	●	●	●
CALCTRAVAR	●	●	●	●	●	●
CALL	●	●	●	●	●	●
CALLPATH	●	●	●	●	●	●
CANCEL	●	●	●	●	●	●
CANCELSUB	●	●	●	●	●	●
CASE	●	●	●	●	●	●
CDC	●	●	●	●	●	●
CDOF	-	-	-	-	-	-
CDOF2	-	-	-	-	-	-
CDON	-	-	-	-	-	-
CFC	●	●	●	●	●	●
CFIN	●	●	●	●	●	●
CFINE	●	●	●	●	●	●
CFTCP	●	●	●	●	●	●
CHAN	●	●	●	●	●	●
CHANDATA	●	●	●	●	●	●
CHAR	●	●	●	●	●	●
CHF	●	●	●	●	●	●
CHKDM	●	●	●	●	●	●
CHKDNO	●	●	●	●	●	●
CHR	●	●	●	●	●	●
CIC	●	●	●	●	●	●
CIP	●	●	●	●	●	●
CLEARM	-	-	●	-	-	●
CLRINT	●	●	●	●	●	●
CMIRROR	●	●	●	●	●	●
COARSEA	●	●	●	●	●	●
COLLPAIR	●	●	●	●	●	●
COMPCAD	●	●	●	●	●	●
COMPCURV	●	●	●	●	●	●
COMPLETE	●	●	●	●	●	●
COMPOF	●	●	●	●	●	●
COMPON	●	●	●	●	●	●
COMPPATH	-	-	-	-	-	-
COMPSURF	-	-	-	-	-	-
CONTDCON	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
CONTPRON	●	●	●	●	●	●
CORROF	●	●	●	●	●	●
CORRTC						
CORRTRAFO	-	-	-	-	-	-
COS	●	●	●	●	●	●
COUPDEF	○	○	○	○	○	○
COUPDEL	○	○	○	○	○	○
COUPOF	○	○	○	○	○	○
COUPOFS	○	○	○	○	○	○
COUPON	○	○	○	○	○	○
COUPONC	○	○	○	○	○	○
COUPRES	○	○	○	○	○	○
CP	●	●	●	●	●	●
CPBC	○	○	○	○	○	○
CPDEF	○	○	○	○	○	○
CPDEL	○	○	○	○	○	○
CPFMOF	○	○	○	○	○	○
CPFMON	○	○	○	○	○	○
CPFMSON	○	○	○	○	○	○
CPFPOS	○	○	○	○	○	○
CPFRS	○	○	○	○	○	○
CPLA	○	○	○	○	○	○
CPLCTID	○	○	○	○	○	○
CPLDEF	○	○	○	○	○	○
CPLDEL	○	○	○	○	○	○
CPLDEN	○	○	○	○	○	○
CPLINSC	○	○	○	○	○	○
CPLINTR	○	○	○	○	○	○
CPLNUM	○	○	○	○	○	○
CPLOF	○	○	○	○	○	○
CPLON	○	○	○	○	○	○
CPLOUTSC	○	○	○	○	○	○
CPLOUTTR	○	○	○	○	○	○
CPLPOS	○	○	○	○	○	○
CPLSETVAL	○	○	○	○	○	○
CPMALARM	○	○	○	○	○	○
CPMBRAKE	○	○	○	○	○	○
CPMPRT	○	○	○	○	○	○
CPMRESET	○	○	○	○	○	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
CPMSTART	○	○	○	○	○	○
CPMVDI	○	○	○	○	○	○
CPOF	○	○	○	○	○	○
CPON	○	○	○	○	○	○
CPRECOF	●	●	●	●	●	●
CPRECON	●	●	●	●	●	●
CPRES	○	○	○	○	○	○
CPROT	●	●	●	●	●	●
CPROTDEF	●	●	●	●	●	●
CPSETTYPE	○	○	○	○	○	○
CPSYNCOP	○	○	○	○	○	○
CPSYNCOP2	○	○	○	○	○	○
CPSYNCOV	○	○	○	○	○	○
CPSYNFIP	○	○	○	○	○	○
CPSYNFIP2	○	○	○	○	○	○
CPSYNFIV	○	○	○	○	○	○
CR	●	●	●	●	●	●
CROT	●	●	●	●	●	●
CROTS	●	●	●	●	●	●
CRPL	●	●	●	●	●	●
CSCALE	●	●	●	●	●	●
CSPLINE	○	○	○	○	○	○
CT	●	●	●	●	●	●
CTAB	-	-	-	-	-	-
CTABDEF	-	-	-	-	-	-
CTABDEL	-	-	-	-	-	-
CTABEND	-	-	-	-	-	-
CTABEXISTS	-	-	-	-	-	-
CTABFNO	-	-	-	-	-	-
CTABFPOL	-	-	-	-	-	-
CTABFSEG	-	-	-	-	-	-
CTABID	-	-	-	-	-	-
CTABINV	-	-	-	-	-	-
CTABISLOCK	-	-	-	-	-	-
CTABLOCK	-	-	-	-	-	-
CTABMEMTYP	-	-	-	-	-	-
CTABMPOL	-	-	-	-	-	-
CTABMSEG	-	-	-	-	-	-
CTABNO	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
CTABNOMEM	-	-	-	-	-	-
CTABPERIOD	-	-	-	-	-	-
CTABPOL	-	-	-	-	-	-
CTABPOLID	-	-	-	-	-	-
CTABSEG	-	-	-	-	-	-
CTABSEGID	-	-	-	-	-	-
CTABSEV	-	-	-	-	-	-
CTABSSV	-	-	-	-	-	-
CTABTEP	-	-	-	-	-	-
CTABTEV	-	-	-	-	-	-
CTABTMAX	-	-	-	-	-	-
CTABTMIN	-	-	-	-	-	-
CTABTSP	-	-	-	-	-	-
CTABTSV	-	-	-	-	-	-
CTABUNLOCK	-	-	-	-	-	-
CTOL	●	●	●	●	●	●
CTOLG0	●	●	●	●	●	●
CTRANS	●	●	●	●	●	●
CUT2D	●	●	●	●	●	●
CUT2DD	●	●	●	●	●	●
CUT2DF	●	●	●	●	●	●
CUT2DFD	●	●	●	●	●	●
CUT3DC	-	-	-	-	-	-
CUT3DCC	-	-	-	-	-	-
CUT3DCCD	-	-	-	-	-	-
CUT3DCD	-	-	-	-	-	-
CUT3DF	-	-	-	-	-	-
CUT3DFD	-	-	-	-	-	-
CUT3DFF	-	-	-	-	-	-
CUT3DFS	-	-	-	-	-	-
CUTCONOF	●	●	●	●	●	●
CUTCONON	●	●	●	●	●	●
CUTMOD	●	●	●	●	●	●
CUTMODK	-	-	-	-	-	-
CYCLE60	-	-	-	-	-	-
CYCLE61	-	-	-	-	-	-
CYCLE62	●	●	●	●	●	●
CYCLE63	-	-	-	-	-	-
CYCLE64	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
CYCLE70	-	-	-	-	-	-
CYCLE72	-	-	-	-	-	-
CYCLE76	-	-	-	-	-	-
CYCLE77	-	-	-	-	-	-
CYCLE78	-	-	-	-	-	-
CYCLE79	-	-	-	-	-	-
CYCLE81	-	-	-	-	-	-
CYCLE82	-	-	-	-	-	-
CYCLE83	-	-	-	-	-	-
CYCLE84	-	-	-	-	-	-
CYCLE85	-	-	-	-	-	-
CYCLE86	-	-	-	-	-	-
CYCLE92	-	-	-	-	-	-
CYCLE95	-	-	-	-	-	-
CYCLE98	-	-	-	-	-	-
CYCLE99	-	-	-	-	-	-
CYCLE116	-	-	-	-	-	-
CYCLE119	-	-	-	-	-	-
CYCLE150	-	-	-	-	-	-
CYCLE435	○	○	○	○	○	○
CYCLE495	○	○	○	○	○	○
CYCLE750	●	●	●	●	●	●
CYCLE751	●	●	●	●	●	●
CYCLE752	●	●	●	●	●	●
CYCLE753	●	●	●	●	●	●
CYCLE754	●	●	●	●	●	●
CYCLE755	●	●	●	●	●	●
CYCLE756	●	●	●	●	●	●
CYCLE757	●	●	●	●	●	●
CYCLE758	●	●	●	●	●	●
CYCLE759	●	●	●	●	●	●
CYCLE782	○	○	○	○	○	○
CYCLE800	○	○	○	○	○	○
CYCLE801	-	-	-	-	-	-
CYCLE802	-	-	-	-	-	-
CYCLE806	-	-	-	-	-	-
CYCLE830	-	-	-	-	-	-
CYCLE832	●	●	●	●	●	●
CYCLE840	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
CYCLE899	-	-	-	-	-	-
CYCLE930	-	-	-	-	-	-
CYCLE940	-	-	-	-	-	-
CYCLE951	-	-	-	-	-	-
CYCLE952	-	-	-	-	-	-
CYCLE961	-	-	-	-	-	-
CYCLE971	-	-	-	-	-	-
CYCLE973	-	-	-	-	-	-
CYCLE974	-	-	-	-	-	-
CYCLE976	-	-	-	-	-	-
CYCLE977	-	-	-	-	-	-
CYCLE978	-	-	-	-	-	-
CYCLE979	-	-	-	-	-	-
CYCLE982	-	-	-	-	-	-
CYCLE994	-	-	-	-	-	-
CYCLE995	-	-	-	-	-	-
CYCLE996	-	-	-	-	-	-
CYCLE997	-	-	-	-	-	-
CYCLE998	-	-	-	-	-	-
CYCLE4071	●	●	●	●	●	●
CYCLE4072	●	●	●	●	●	●
CYCLE4073	●	●	●	●	●	●
CYCLE4074	●	●	●	●	●	●
CYCLE4075	●	●	●	●	●	●
CYCLE4077	●	●	●	●	●	●
CYCLE4078	●	●	●	●	●	●
CYCLE4079	●	●	●	●	●	●
CYCLE9960	-	-	-	-	-	-

Operations D ... F

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
D	●	●	●	●	●	●
D0	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
DAC	●	●	●	●	●	●
DC	●	●	●	●	●	●
DCI	●	●	●	●	●	●
DCM	●	●	●	●	●	●
DCU	●	●	●	●	●	●
DEF	●	●	●	●	●	●
DEFINE	●	●	●	●	●	●
DEFAULT	●	●	●	●	●	●
DELAYFSTON	●	●	●	●	●	●
DELAYFSTOF	●	●	●	●	●	●
DELDL	●	●	●	●	●	●
DELDTG	●	●	●	●	●	●
DELETE	●	●	●	●	●	●
DELMLOWNER	●	●	●	●	●	●
DELMLRES	●	●	●	●	●	●
DELMT	-	-	-	-	-	-
DELOBJ	-	-	-	-	-	-
DELT	●	●	●	●	●	●
DELTC	●	●	●	●	●	●
DELTOOLENV	●	●	●	●	●	●
DIACYCOFA	●	●	●	●	●	●
DIAM90	●	●	●	●	●	●
DIAM90A	●	●	●	●	●	●
DIAMCHAN	●	●	●	●	●	●
DIAMCHANA	●	●	●	●	●	●
DIAMCYCOF	●	●	●	●	●	●
DIAMOF	●	●	●	●	●	●
DIAMOFA	●	●	●	●	●	●
DIAMON	●	●	●	●	●	●
DIAMONA	●	●	●	●	●	●
DIC	●	●	●	●	●	●
DILF	●	●	●	●	●	●
DISABLE	●	●	●	●	●	●
DISC	●	●	●	●	●	●
DISCL	●	●	●	●	●	●
DISPLOF	●	●	●	●	●	●
DISPLON	●	●	●	●	●	●
DISPR	●	●	●	●	●	●
DISR	●	●	●	●	●	●

Operation	**SINUMERIK 828D**					
● Standard **○ Option** **- not available**	**SW24x (5) CNC SW G-Tech Export (gce42)**	**SW26x (3) CNC SW G-Tech Export (gce62)**	**SW28x (1) CNC SW G-Tech Export (gce82)**	**SW24x (5) CNC SW G-Tech Export (gse42)**	**SW26x (3) CNC SW G-Tech Export (gse62)**	**SW28x (1) CNC SW G-Tech Export (gse82)**
DISRP	●	●	●	●	●	●
DITE	●	●	●	●	●	●
DITS	●	●	●	●	●	●
DIV	●	●	●	●	●	●
DL	-	-	-	-	-	-
DO	●	●	●	●	●	●
DRFOF	●	●	●	●	●	●
DRIVE	●	●	●	●	●	●
DRIVEA	●	●	●	●	●	●
DRVPRD	●	●	●	●	●	●
DRVPWR	●	●	●	●	●	●
DYNFINISH	●	●	●	●	●	●
DYNNORM	●	●	●	●	●	●
DYNPOS	●	●	●	●	●	●
DYNPREC	●	●	●	●	●	●
DYNROUGH	●	●	●	●	●	●
DYNSEMIFIN	●	●	●	●	●	●
DZERO	●	●	●	●	●	●
EAUTO	○	○	○	○	○	○
EGDEF	○	○	○	○	○	○
EGDEL	○	○	○	○	○	○
EGOFC	○	○	○	○	○	○
EGOFS	○	○	○	○	○	○
EGON	○	○	○	○	○	○
EGONSYN	○	○	○	○	○	○
EGONSYNE	○	○	○	○	○	○
ELSE	●	●	●	●	●	●
ENABLE	●	●	●	●	●	●
ENAT	○	○	○	○	○	○
ENDFOR	●	●	●	●	●	●
ENDIF	●	●	●	●	●	●
ENDLABEL	●	●	●	●	●	●
ENDLOOP	●	●	●	●	●	●
ENDPROC	●	●	●	●	●	●
ENDWHILE	●	●	●	●	●	●
ESRR	○	○	○	○	○	○
ESRS	○	○	○	○	○	○
ETAN	○	○	○	○	○	○
EVERY	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
EX	●	●	●	●	●	●
EXECSTRING	●	●	●	●	●	●
EXECTAB	●	●	●	●	●	●
EXECUTE	●	●	●	●	●	●
EXP	●	●	●	●	●	●
EXTCALL	●	●	●	●	●	●
EXTCLOSE	●	●	●	●	●	●
EXTERN	●	●	●	●	●	●
EXTOPEN	●	●	●	●	●	●
F	●	●	●	●	●	●
FA	●	●	●	●	●	●
FAD	●	●	●	●	●	●
FALSE	●	●	●	●	●	●
FB	●	●	●	●	●	●
FCTDEF	●	●	●	●	●	●
FCUB	●	●	●	●	●	●
FD	●	●	●	●	●	●
FDA	●	●	●	●	●	●
FENDNORM	●	●	●	●	●	●
FFWOF	●	●	●	●	●	●
FFWON	●	●	●	●	●	●
FGREF	●	●	●	●	●	●
FGROUP	●	●	●	●	●	●
FI	●	●	●	●	●	●
FIFOCTRL	●	●	●	●	●	●
FILEDATE	●	●	●	●	●	●
FILEINFO	●	●	●	●	●	●
FILESIZE	●	●	●	●	●	●
FILESTAT	●	●	●	●	●	●
FILETIME	●	●	●	●	●	●
FINEA	●	●	●	●	●	●
FL	●	●	●	●	●	●
FLIM	●	●	●	●	●	●
FLIN	●	●	●	●	●	●
FMA	●	●	●	●	●	●
FNORM	●	●	●	●	●	●
FOCOF	○	○	○	○	○	○
FOCON	○	○	○	○	○	○
FOR	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
FP	●	●	●	●	●	●
FPO	-	-	-	-	-	-
FPR	●	●	●	●	●	●
FPRAOF	●	●	●	●	●	●
FPRAON	●	●	●	●	●	●
FRAME	●	●	●	●	●	●
FRC	●	●	●	●	●	●
FRCM	●	●	●	●	●	●
FROM	●	●	●	●	●	●
FTOC	●	●	●	●	●	●
FTOCOF	●	●	●	●	●	●
FTOCON	●	●	●	●	●	●
FXS	●	●	●	●	●	●
FXST	●	●	●	●	●	●
FXSW	●	●	●	●	●	●
FZ	●	●	●	●	●	●

Operations G ... L

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
G0	●	●	●	●	●	●
G1	●	●	●	●	●	●
G2	●	●	●	●	●	●
G3	●	●	●	●	●	●
G4	●	●	●	●	●	●
G5	●	●	●	●	●	●
G7	●	●	●	●	●	●
G9	●	●	●	●	●	●
G17	●	●	●	●	●	●
G18	●	●	●	●	●	●
G19	●	●	●	●	●	●
G25	●	●	●	●	●	●
G26	●	●	●	●	●	●
G33	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard **○ Option** **- not available**	**SW24x (5) CNC SW G-Tech Export (gce42)**	**SW26x (3) CNC SW G-Tech Export (gce62)**	**SW28x (1) CNC SW G-Tech Export (gce82)**	**SW24x (5) CNC SW G-Tech Export (gse42)**	**SW26x (3) CNC SW G-Tech Export (gse62)**	**SW28x (1) CNC SW G-Tech Export (gse82)**
G34	●	●	●	●	●	●
G35	●	●	●	●	●	●
G40	●	●	●	●	●	●
G41	●	●	●	●	●	●
G42	●	●	●	●	●	●
G53	●	●	●	●	●	●
G54	●	●	●	●	●	●
G55	●	●	●	●	●	●
G56	●	●	●	●	●	●
G57	●	●	●	●	●	●
G58	→ G505					
G59	→ G506					
G60	●	●	●	●	●	●
G62	●	●	●	●	●	●
G63	●	●	●	●	●	●
G64	●	●	●	●	●	●
G70	●	●	●	●	●	●
G71	●	●	●	●	●	●
G74	●	●	●	●	●	●
G75	●	●	●	●	●	●
G90	●	●	●	●	●	●
G91	●	●	●	●	●	●
G93	●	●	●	●	●	●
G94	●	●	●	●	●	●
G95	●	●	●	●	●	●
G96	●	●	●	●	●	●
G97	●	●	●	●	●	●
G110	●	●	●	●	●	●
G111	●	●	●	●	●	●
G112	●	●	●	●	●	●
G140	●	●	●	●	●	●
G141	●	●	●	●	●	●
G142	●	●	●	●	●	●
G143	●	●	●	●	●	●
G147	●	●	●	●	●	●
G148	●	●	●	●	●	●
G153	●	●	●	●	●	●
G247	●	●	●	●	●	●
G248	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
G290	●	●	●	●	●	●
G291	-	-	-	-	-	-
G331	●	●	●	●	●	●
G332	●	●	●	●	●	●
G335	●	●	●	●	●	●
G336	●	●	●	●	●	●
G340	●	●	●	●	●	●
G341	●	●	●	●	●	●
G347	●	●	●	●	●	●
G348	●	●	●	●	●	●
G450	●	●	●	●	●	●
G451	●	●	●	●	●	●
G460	●	●	●	●	●	●
G461	●	●	●	●	●	●
G462	●	●	●	●	●	●
G500	●	●	●	●	●	●
G505 ... G599	●	●	●	●	●	●
G601	●	●	●	●	●	●
G602	●	●	●	●	●	●
G603	●	●	●	●	●	●
G621	●	●	●	●	●	●
G641	●	●	●	●	●	●
G642	●	●	●	●	●	●
G643	●	●	●	●	●	●
G644	●	●	●	●	●	●
G645	●	●	●	●	●	●
G646	○	○	○	○	○	○
G700	●	●	●	●	●	●
G710	●	●	●	●	●	●
G810 ... G819	-	-	-	-	-	-
G820 ... G829	-	-	-	-	-	-
G931	●	●	●	●	●	●
G942	●	●	●	●	●	●
G952	●	●	●	●	●	●
G961	●	●	●	●	●	●
G962	●	●	●	●	●	●
G971	●	●	●	●	●	●
G972	●	●	●	●	●	●
G973	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
GEOAX	●	●	●	●	●	●
GET	●	●	●	●	●	●
GETACTT	●	●	●	●	●	●
GETACTTD	●	●	●	●	●	●
GETD	-	-	●	-	-	●
GETDNO	●	●	●	●	●	●
GETEXET	●	●	●	●	●	●
GETFREELOC	●	●	●	●	●	●
GETSELT	●	●	●	●	●	●
GETT	●	●	●	●	●	●
GETTCOR	●	●	●	●	●	●
GETTENV	●	●	●	●	●	●
GETVARAP	●	●	●	●	●	●
GETVARDFT	●	●	●	●	●	●
GETVARLIM	●	●	●	●	●	●
GETVARPHU	●	●	●	●	●	●
GETVARTYP	●	●	●	●	●	●
GFRAME0 ... GFRAME100	< 50	< 100	< 100	< 50	< 100	< 100
GOTO	●	●	●	●	●	●
GOTOB	●	●	●	●	●	●
GOTOC	●	●	●	●	●	●
GOTOF	●	●	●	●	●	●
GOTOS	●	●	●	●	●	●
GP	●	●	●	●	●	●
GWPSOF	●	●	●	●	●	●
GROUP_ADDEND	●	●	●	●	●	●
GROUP_BEGIN	●	●	●	●	●	●
GROUP_END	●	●	●	●	●	●
GWPSON	●	●	●	●	●	●
H...	●	●	●	●	●	●
HOLES1	-	-	-	-	-	-
HOLES2	-	-	-	-	-	-
I	●	●	●	●	●	●
I1	●	●	●	●	●	●
IC	●	●	●	●	●	●
ICYCOF	●	●	●	●	●	●
ICYCON	●	●	●	●	●	●
ID	●	●	●	●	●	●
IDS	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
IF	●	●	●	●	●	●
INDEX	●	●	●	●	●	●
INIPO	●	●	●	●	●	●
INIRE	●	●	●	●	●	●
INICF	●	●	●	●	●	●
INIT	-	-	●	-	-	●
INITIAL						
INT	●	●	●	●	●	●
INTERSEC	●	●	●	●	●	●
INVCCW	-	-	-	-	-	-
INVCW	-	-	-	-	-	-
INVFRAME	●	●	●	●	●	●
IP	●	●	●	●	●	●
IPOBRKA	●	●	●	●	●	●
IPOENDA	●	●	●	●	●	●
IPTRLOCK	●	●	●	●	●	●
IPTRUNLOCK	●	●	●	●	●	●
IR	●	●	●	●	●	●
ISAXIS	●	●	●	●	●	●
ISD	-	-	-	-	-	-
ISFILE	●	●	●	●	●	●
ISNUMBER	●	●	●	●	●	●
ISOCALL	-	-	-	-	-	-
ISVAR	●	●	●	●	●	●
J	●	●	●	●	●	●
J1	●	●	●	●	●	●
JERKA	●	●	●	●	●	●
JERKLIM	●	●	●	●	●	●
JERKLIMA	●	●	●	●	●	●
JR	●	●	●	●	●	●
K	●	●	●	●	●	●
K1	●	●	●	●	●	●
KONT	●	●	●	●	●	●
KONTC	●	●	●	●	●	●
KONTT	●	●	●	●	●	●
KR	●	●	●	●	●	●
L	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
LEAD						
Tool orientation	-	-	-	-	-	-
Orientation polynomial	-	-	-	-	-	-
LEADOF	-	-	-	-	-	-
LEADON	-	-	-	-	-	-
LENTOAX	●	●	●	●	●	●
LFOF	●	●	●	●	●	●
LFON	●	●	●	●	●	●
LFPOS	●	●	●	●	●	●
LFTXT	●	●	●	●	●	●
LFWP	●	●	●	●	●	●
LIFTFAST	●	●	●	●	●	●
LIMS	●	●	●	●	●	●
LLI	●	●	●	●	●	●
LN	●	●	●	●	●	●
LOCK	●	●	●	●	●	●
LONGHOLE	-	-	-	-	-	-
LOOP	●	●	●	●	●	●

Operations M ... R

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
M0	●	●	●	●	●	●
M1	●	●	●	●	●	●
M2	●	●	●	●	●	●
M3	●	●	●	●	●	●
M4	●	●	●	●	●	●
M5	●	●	●	●	●	●
M6	●	●	●	●	●	●
M17	●	●	●	●	●	●
M19	●	●	●	●	●	●
M30	●	●	●	●	●	●
M40	●	●	●	●	●	●
M41 ... M45	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
M70	●	●	●	●	●	●
MASLDEF	○	○	○	○	○	○
MASLDEL	○	○	○	○	○	○
MASLOF	○	○	○	○	○	○
MASLOFS	○	○	○	○	○	○
MASLON	○	○	○	○	○	○
MATCH	●	●	●	●	●	●
MAXVAL	●	●	●	●	●	●
MCALL	●	●	●	●	●	●
MCALLOF	●	●	●	●	●	●
MEAC	○	○	○	○	○	○
MEAFRAME	●	●	●	●	●	●
MEAS	●	●	●	●	●	●
MEASA	○	○	○	○	○	○
MEASF	●	●	●	●	●	●
MEASURE	●	●	●	●	●	●
MEAW	●	●	●	●	●	●
MEAWA	○	○	○	○	○	○
MI	●	●	●	●	●	●
MINDEX	●	●	●	●	●	●
MINVAL	●	●	●	●	●	●
MIRROR	●	●	●	●	●	●
MMC	●	●	●	●	●	●
MOD	●	●	●	●	●	●
MODAXVAL	●	●	●	●	●	●
MOV	●	●	●	●	●	●
MOVT	●	●	●	●	●	●
MSG	●	●	●	●	●	●
MVTOOL	●	●	●	●	●	●
N	●	●	●	●	●	●
NAMETOINT	●	●	●	●	●	●
NCK	●	●	●	●	●	●
NEWCONF	●	●	●	●	●	●
NEWMT	●	●	●	●	●	●
NEWT	-	-	-	-	-	-
NORM	●	●	●	●	●	●
NOT	●	●	●	●	●	●
NPROT	●	●	●	●	●	●
NPROTDEF	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
NUMBER	●	●	●	●	●	●
OEMIPO1	-	-	-	-	-	-
OEMIPO2	-	-	-	-	-	-
OF	●	●	●	●	●	●
OFFN	●	●	●	●	●	●
OMA1	-	-	-	-	-	-
OMA2	-	-	-	-	-	-
OMA3	-	-	-	-	-	-
OMA4	-	-	-	-	-	-
OMA5	-	-	-	-	-	-
OR	●	●	●	●	●	●
ORIANGLE	-	-	-	-	-	-
ORIAXES	-	-	-	-	-	-
ORIAXESFR	-	-	-	-	-	-
ORIAXPOS	-	-	-	-	-	-
ORIC	-	-	-	-	-	-
ORICONCCW	-	-	-	-	-	-
ORICONCW	-	-	-	-	-	-
ORICONIO	-	-	-	-	-	-
ORICONTO	-	-	-	-	-	-
ORICURINV	-	-	-	-	-	-
ORICURVE	-	-	-	-	-	-
ORID	-	-	-	-	-	-
ORIEULER	-	-	-	-	-	-
ORIMKS	-	-	-	-	-	-
ORIPATH	-	-	-	-	-	-
ORIPATHS	-	-	-	-	-	-
ORIPLANE	-	-	-	-	-	-
ORIRESET	-	-	-	-	-	-
ORIROTA	-	-	-	-	-	-
ORIROTC	-	-	-	-	-	-
ORIROTR	-	-	-	-	-	-
ORIROTT	-	-	-	-	-	-
ORIRPY	-	-	-	-	-	-
ORIRPY2	-	-	-	-	-	-
ORIS	-	-	-	-	-	-
ORISOF	-	-	-	-	-	-
ORISOLH	-	-	-	-	-	-
ORISON	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
ORIVECT	-	-	-	-	-	-
ORIVIRT1	-	-	-	-	-	-
ORIVIRT2	-	-	-	-	-	-
ORIWKS	-	-	-	-	-	-
OS	○	○	○	○	○	○
OSB	○	○	○	○	○	○
OSC	-	-	-	-	-	-
OSCILL	○	○	○	○	○	○
OSCTRL	○	○	○	○	○	○
OSD	-	-	-	-	-	-
OSE	○	○	○	○	○	○
OSNSC	●	●	●	●	●	●
OSOF	-	-	-	-	-	-
OSP1	●	●	●	●	●	●
OSP2	●	●	●	●	●	●
OSS	-	-	-	-	-	-
OSSE	-	-	-	-	-	-
OST	-	-	-	-	-	-
OST1	●	●	●	●	●	●
OST2	●	●	●	●	●	●
OTOL	●	●	●	●	●	●
OTOLG0	●	●	●	●	●	●
OVR	●	●	●	●	●	●
OVRA	●	●	●	●	●	●
OVRRAP	●	●	●	●	●	●
P	●	●	●	●	●	●
PACCLIM	○	○	○	○	○	○
PAROT	●	●	●	●	●	●
PAROTOF	●	●	●	●	●	●
PCALL	●	●	●	●	●	●
PDELAYOF	-	-	-	-	-	-
PDELAYON	-	-	-	-	-	-
PHI	-	-	-	-	-	-
PHU	●	●	●	●	●	●
PL	-	-	-	-	-	-
PM	●	●	●	●	●	●
PO	-	-	-	-	-	-
POCKET3	-	-	-	-	-	-
POCKET4	-	-	-	-	-	-

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
POLF	●	●	●	●	●	●
POLFA	●	●	●	●	●	●
POLFMASK	●	●	●	●	●	●
POLFMLIN	●	●	●	●	●	●
POLY	-	-	-	-	-	-
POLYPATH	-	-	-	-	-	-
PON	-	-	-	-	-	-
PONS	-	-	-	-	-	-
POS	●	●	●	●	●	●
POSA	●	●	●	●	●	●
POSM	●	●	●	●	●	●
POSMT	-	-	-	-	-	-
POSP	●	●	●	●	●	●
POSRANGE	●	●	●	●	●	●
POT	●	●	●	●	●	●
PR	●	●	●	●	●	●
PREPRO	●	●	●	●	●	●
PRESETON	●	●	●	●	●	●
PRESETONS	●	●	●	●	●	●
PRIO	●	●	●	●	●	●
PRLOC	●	●	●	●	●	●
PROC	●	●	●	●	●	●
PROTA	●	●	●	●	●	●
PROTD	●	●	●	●	●	●
PROTS	●	●	●	●	●	●
PSI	-	-	-	-	-	-
PTP	●	●	●	●	●	●
PTPG0	●	●	●	●	●	●
PTPWOC	●	●	●	●	●	●
PUNCHACC	-	-	-	-	-	-
PUTFTOC	●	●	●	●	●	●
PUTFTOCF	●	●	●	●	●	●
PW	○	○	○	○	○	○
QU	●	●	●	●	●	●
R...	●	●	●	●	●	●
RAC	●	●	●	●	●	●
RDISABLE	●	●	●	●	●	●
READ	●	●	●	●	●	●
REAL	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
RELEASE	●	●	●	●	●	●
REP	●	●	●	●	●	●
REPEAT	●	●	●	●	●	●
REPEATB	●	●	●	●	●	●
REPOSA	●	●	●	●	●	●
REPOSH	●	●	●	●	●	●
REPOSHA	●	●	●	●	●	●
REPOSL	●	●	●	●	●	●
REPOSQ	●	●	●	●	●	●
REPOSQA	●	●	●	●	●	●
RESETMON	●	●	●	●	●	●
RET	●	●	●	●	●	●
RETB	●	●	●	●	●	●
RIC	●	●	●	●	●	●
RINDEX	●	●	●	●	●	●
RMB	●	●	●	●	●	●
RME	●	●	●	●	●	●
RMI	●	●	●	●	●	●
RMN	●	●	●	●	●	●
RND	●	●	●	●	●	●
RNDM	●	●	●	●	●	●
ROT	●	●	●	●	●	●
ROTS	●	●	●	●	●	●
ROUND	●	●	●	●	●	●
ROUNDUP	●	●	●	●	●	●
RP	●	●	●	●	●	●
RPL	●	●	●	●	●	●
RT	●	●	●	●	●	●
RTLIOF	●	●	●	●	●	●
RTLION	●	●	●	●	●	●

Operations S ... Z

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
S	●	●	●	●	●	●
SAVE	●	●	●	●	●	●
SBLOF	●	●	●	●	●	●
SBLON	●	●	●	●	●	●
SC	●	●	●	●	●	●
SCALE	●	●	●	●	●	●
SCC	●	●	●	●	●	●
SCPARA	●	●	●	●	●	●
SD	○	○	○	○	○	○
SET	●	●	●	●	●	●
SETAL	●	●	●	●	●	●
SETDNO	●	●	●	●	●	●
SETINT	●	●	●	●	●	●
SETM	-	-	●	-	-	●
SETMS	●	●	●	●	●	●
SETMS(n)	●	●	●	●	●	●
SETMTH	●	●	●	●	●	●
SETPIECE	●	●	●	●	●	●
SETTA	●	●	●	●	●	●
SETTCOR	●	●	●	●	●	●
SETTIA	●	●	●	●	●	●
SF	●	●	●	●	●	●
SIN	●	●	●	●	●	●
SIRELAY	-	-	-	-	-	-
SIRELIN	-	-	-	-	-	-
SIRELOUT	-	-	-	-	-	-
SIRELTIME	-	-	-	-	-	-
SLOT1	-	-	-	-	-	-
SLOT2	-	-	-	-	-	-
SOFT	●	●	●	●	●	●
SOFTA	●	●	●	●	●	●
SON	-	-	-	-	-	-
SONS	-	-	-	-	-	-
SPATH	●	●	●	●	●	●
SPCOF	●	●	●	●	●	●
SPCON	●	●	●	●	●	●
SPI	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard **○ Option** **- not available**	**SW24x (5) CNC SW G-Tech Export (gce42)**	**SW26x (3) CNC SW G-Tech Export (gce62)**	**SW28x (1) CNC SW G-Tech Export (gce82)**	**SW24x (5) CNC SW G-Tech Export (gse42)**	**SW26x (3) CNC SW G-Tech Export (gse62)**	**SW28x (1) CNC SW G-Tech Export (gse82)**
SPIF1	-	-	-	-	-	-
SPIF2	-	-	-	-	-	-
SPLINEPATH	○	○	○	○	○	○
SPN	-	-	-	-	-	-
SPOF	-	-	-	-	-	-
SPOS	●	●	●	●	●	●
SPOSA	●	●	●	●	●	●
SPP	-	-	-	-	-	-
SPRINT	●	●	●	●	●	●
SQRT	●	●	●	●	●	●
SR	●	●	●	●	●	●
SRA	●	●	●	●	●	●
ST	●	●	●	●	●	●
STA	●	●	●	●	●	●
START	-	-	●	-	-	●
STARTFIFO	●	●	●	●	●	●
STAT	●	●	●	●	●	●
STOLF	●	●	●	●	●	●
STOPFIFO	●	●	●	●	●	●
STOPRE	●	●	●	●	●	●
STOPREOF	●	●	●	●	●	●
STRING	●	●	●	●	●	●
STRINGFELD	●	●	●	●	●	●
STRINGIS	●	●	●	●	●	●
STRLEN	●	●	●	●	●	●
SUBSTR	●	●	●	●	●	●
SUPA	●	●	●	●	●	●
SUPD	●	●	●	●	●	●
SVC	●	●	●	●	●	●
SYNFCT	●	●	●	●	●	●
SYNR	●	●	●	●	●	●
SYNRW	●	●	●	●	●	●
SYNW	●	●	●	●	●	●
T	●	●	●	●	●	●
TAN	●	●	●	●	●	●
TANG	○	○	○	○	○	○
TANGDEL	○	○	○	○	○	○
TANGOF	○	○	○	○	○	○
TANGON	○	○	○	○	○	○

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
TCA (828D: _TCA)	●	●	●	●	●	●
TCARR	●	●	●	●	●	●
TCI	●	●	●	●	●	●
TCOABS	●	●	●	●	●	●
TCOFR	●	●	●	●	●	●
TCOFRX	●	●	●	●	●	●
TCOFRY	●	●	●	●	●	●
TCOFRZ	●	●	●	●	●	●
THETA	-	-	-	-	-	-
TILT	-	-	-	-	-	-
TLIFT	○	○	○	○	○	○
TML	●	●	●	●	●	●
TMOF	●	●	●	●	●	●
TMON	●	●	●	●	●	●
TO	●	●	●	●	●	●
TOFF	●	●	●	●	●	●
TOFFL	●	●	●	●	●	●
TOFFLR	●	●	●	●	●	●
TOFFOF	-	-	-	-	-	-
TOFFON	-	-	-	-	-	-
TOFFR	●	●	●	●	●	●
TOFRAME	●	●	●	●	●	●
TOFRAMEX	●	●	●	●	●	●
TOFRAMEY	●	●	●	●	●	●
TOFRAMEZ	●	●	●	●	●	●
TOLOWER	●	●	●	●	●	●
TOOLENV	●	●	●	●	●	●
TOOLGNT	●	●	●	●	●	●
TOOLGT	●	●	●	●	●	●
TOROT	●	●	●	●	●	●
TOROTOF	●	●	●	●	●	●
TOROTX	●	●	●	●	●	●
TOROTY	●	●	●	●	●	●
TOROTZ	●	●	●	●	●	●
TOUPPER	●	●	●	●	●	●
TOWBCS	●	●	●	●	●	●
TOWKCS	●	●	●	●	●	●
TOWMCS	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
TOWSTD	●	●	●	●	●	●
TOWTCS	●	●	●	●	●	●
TOWWCS	●	●	●	●	●	●
TR	●	●	●	●	●	●
TRAANG	○	○	○	-	-	-
TRACON	○	○	○	-	-	-
TRACYL	○	○	○	○	○	○
TRAFOOF	●	●	●	●	●	●
TRAFOON	-	-	-	-	-	-
TRAILOF	●	●	●	●	●	●
TRAILON	●	●	●	●	●	●
TRANS	●	●	●	●	●	●
TRANSMIT	○	○	○	○	○	○
TRAORI	-	-	-	-	-	-
TRUE	●	●	●	●	●	●
TRUNC	●	●	●	●	●	●
TU	●	●	●	●	●	●
TURN	●	●	●	●	●	●
ULI	●	●	●	●	●	●
UNLOCK	●	●	●	●	●	●
UNTIL	●	●	●	●	●	●
UPATH	●	●	●	●	●	●
VAR	●	●	●	●	●	●
VELOLIM	●	●	●	●	●	●
VELOLIMA	●	●	●	●	●	●
WAITC	●	●	●	●	●	●
WAITE	-	-	●	-	-	●
WAITENC	●	●	●	●	●	●
WAITM	-	-	●	-	-	●
WAITMC	-	-	●	-	-	●
WAITP	●	●	●	●	●	●
WAITS	●	●	●	●	●	●
WALCS0	●	●	●	●	●	●
WALCS1	●	●	●	●	●	●
WALCS2	●	●	●	●	●	●
WALCS3	●	●	●	●	●	●
WALCS4	●	●	●	●	●	●
WALCS5	●	●	●	●	●	●
WALCS6	●	●	●	●	●	●

Operation	SINUMERIK 828D					
● Standard ○ Option - not available	SW24x (5) CNC SW G-Tech Export (gce42)	SW26x (3) CNC SW G-Tech Export (gce62)	SW28x (1) CNC SW G-Tech Export (gce82)	SW24x (5) CNC SW G-Tech Export (gse42)	SW26x (3) CNC SW G-Tech Export (gse62)	SW28x (1) CNC SW G-Tech Export (gse82)
WALCS7	●	●	●	●	●	●
WALCS8	●	●	●	●	●	●
WALCS9	●	●	●	●	●	●
WALCS10	●	●	●	●	●	●
WALIMOF	●	●	●	●	●	●
WALIMON	●	●	●	●	●	●
WHEN	●	●	●	●	●	●
WHENEVER	●	●	●	●	●	●
WHILE	●	●	●	●	●	●
WORKPIECE	●	●	●	●	●	●
WRITE	●	●	●	●	●	●
WRTPR	●	●	●	●	●	●
X	●	●	●	●	●	●
XOR	●	●	●	●	●	●
Y	●	●	●	●	●	●
Z	●	●	●	●	●	●

5.3 Addresses

5.3.1 Address letters

Letter	Meaning	Numeric extension
A	Settable address identifier	x
B	Settable address identifier	x
C	Settable address identifier	x
D	Selection/deselection of tool length compensation, tool cutting edge	
E	Settable address identifier	x
F	Feedrate Dwell time in seconds	x
G	G command	
H	H function	x
I	Settable address identifier	x
J	Settable address identifier	x
K	Settable address identifier	x

Letter	Meaning	Numeric extension
L	Subprogram name, subprogram call	
M	M function	x
N	Subblock number	
O	Unassigned	
P	Number of program runs	
Q	Settable address identifier	x
R	Variable identifier (R parameter) Settable address identifier (without numeric extension)	x
S	Spindle value Dwell time in spindle revolutions	x x
T	Tool number	x
U	Settable address identifier	x
V	Settable address identifier	x
W	Settable address identifier	x
X	Settable address identifier	x
Y	Settable address identifier	x
Z	Settable address identifier	x
%	Start character and separator for file transfer	
:	Main block number	
/	Skip identifier	

5.3.2 Fixed addresses

Fixed addresses without axial extension

Address identifier	Address type	Modal/ non-modal	G70/ G71	G700/ G710	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	QU	Data type of the assigned value
D	Offset number	m								x	Unsigned INT
F	Feed, dwell time	m, s	x							x	Unsigned REAL
FLIM	Maximum path velocity	m									Unsigned REAL
G	G command	See list of the G functions									Unsigned INT

Address identifier	Address type	Modal/ non-modal	G70/ G71	G700/ G710	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	QU	Data type of the assigned value
H	Auxiliary functions	s								x	M: unsigned INT H: REAL
L	Subprogram number	s									Unsigned INT
M	Auxiliary functions	s								x	M: unsigned INT H: REAL
N	Block number	s									Unsigned INT
OVR	Override	m									Unsigned REAL
OVRRAP	Override for rapid traverse velocity	m									Unsigned REAL
P	Number of subprogram repetitions	s									Unsigned INT
PACCLIM	Maximum path acceleration	m									Unsigned REAL
S	Spindle, dwell time	m, s								x	Unsigned REAL
SCC	Assignment of a transverse axis to G96 /G961/G962	m									REAL
SPOS	Spindle position	m				x	x	x			REAL
SPOSA	Spindle position across block boundaries	m				x	x	x			REAL
T	Tool number	m								x	Unsigned INT

Fixed addresses with axial extension

Address identifier	Address type	Modal/ non-modal	G70/ G71	G700/ G710	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	QU	Data type of the assigned value
ACC	Axial acceleration	m									Unsigned REAL
ACCLIMA	Axial acceleration limitation of following axis	m									Unsigned REAL
AX	Variable axis identifier	1)	x	x	x	x	x	x			REAL
FA	Axial feedrate	m	x							x	Unsigned REAL
FDA	Axis feedrate for handwheel override	s	x								Unsigned REAL
FGREF	Reference radius	m	x	x							Unsigned REAL
FL	Axial feedrate limit	m	x								Unsigned REAL
FMA	Axial synchronized feedrate	m									Unsigned REAL
FOC	Non-modal traversing with limited torque	s									REAL
FOCOF	Modal traversing with limited torque OFF	m									REAL
FOCON	Modal traversing with limited torque ON	m									REAL
FXS	Travel to fixed stop ON	m									Unsigned INT
FXST	Torque limit for travel to fixed stop	m									REAL
FXSW	Monitoring window for travel to fixed stop	m									REAL
IP	Variable interpolation parameter	s	x	x	x	x	x				REAL

Address identifier	Address type	Modal/ non-modal	G70/ G71	G700/ G710	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	QU	Data type of the assigned value
JERKLIM	Axial jerk limitation	m									Unsigned REAL
JERKLIMA	Axial jerk limitation of following axis	m									Unsigned REAL
MEAC	Axis-specific continuous measurement without delete distance-to-go	s									INT mode, FIFO No. and 1 - 4 trigger events
MEASA	Axis-specific measurement with delete distance-to-go	s									INT Mode and 1 - 4 trigger events
MEAWA	Axis-specific measurement without delete distance-to-go	s									INT Mode and 1 - 4 trigger events
MOV	Start positioning axis	m	x	x	x	x	x	x	x		REAL
OS	Oscillation ON/OFF	m									Unsigned INT
OSB	Oscillation starting point	m	x	x	x	x	x	x			REAL
OSCILL	Axis assignment for oscillation, activate oscillation	m									Axis: 1 - 3 infeed axes
OSCTRL	Oscillation options	m									Unsigned INT: Setting options, unsigned INT: Reset options
OSE	Oscillation end point	m	x	x	x	x	x	x			REAL
OSNSC	Number of spark-out cycles (oscillation)	m									Unsigned INT

Address identifier	**Address type**	**Modal/ non-modal**	**G70/ G71**	**G700/ G710**	**G90/ G91**	**IC**	**AC**	**DC, ACN, ACP**	**CIC, CAC, CDC, CACN, CACP**	**QU**	**Data type of the assigned value**
OSP1	Left reversal point (oscillation)	m	x	x	x	x	x	x			REAL
OSP2	Right reversal point (oscillation)	m	x	x	x	x	x	x			REAL
OST1	Stopping time at left reversal point (oscillation)	m									REAL
OST2	Stopping time at right reversal point (oscillation)	m									REAL
OVRA	Axial override	m	x								Unsigned REAL
PO	Polynomial coefficient	s	x	x		x	x	x			Unsigned REAL
POLF	LIFTFAST position	m	x	x							Unsigned REAL
POS	Positioning axis	m	x	x	x	x	x	x	x		REAL
POSA	Positioning axis across block boundaries	m	x	x	x	x	x	x	x		REAL
POSP	Positioning axis in parts (oscillation)	m	x	x	x	x	x	x			REAL: End position Real: Partial length INT: Option
STA	Sparking out time (axial)	m									Unsigned REAL
SRA	Retraction path on external input (axial)	m									Unsigned REAL
VELOLIM	Axial velocity limitation	m									Unsigned REAL
VELOLIMA	Axial velocity limitation of following axis	m									Unsigned REAL

[1] Absolute end points: Modal, incremental end points: Non-modal, otherwise modal/non-modal depending on the G function that determines the syntax.

5.3.3 Settable addresses

Address identifier (default setting)	Address type	Modal/ non-modal	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	PR, PM	QU	Max. number	Data type of the assigned value
Axis values and end points											
X, Y, Z, A, B, C	Axis	[1]	x	x	x	x				8	REAL
AP	Polar angle	m/s [1]	x	x	x					1	REAL
RP	Polar radius	m/s [1]	x	x	x					1	Unsigned REAL
Tool orientation											
A2, B2, C2	Euler angle or RPY angle	s								3	REAL
A3, B3, C3	Components of the directional vector	s								3	REAL
A4, B4, C4	Components of the surface normal vector at the start of the block	s								3	REAL
A5, B5, C5	Components of the surface normal vector at the end of the block	s								3	REAL
A6, B6, C6	Components of the direction vector for the rotary axis of the taper	s								3	REAL
A7, B7, C7	Vector components for the intermediate orientation on the peripheral surface of a taper	s								3	REAL
LEAD	Lead angle	m								1	REAL
THETA	Angle of rotation, rotation around the tool direction	m		x	x					1	REAL
TILT	Tilt angle	m								1	REAL

Address identifier (default setting)	Address type	Modal/ non-modal	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	PR, PM	QU	Max. number	Data type of the assigned value
ORIS	Orientation change (in relation to the path)	m								1	REAL
Interpolation parameters											
I, J, K	Interpolation parameter intermediate point coordinate	s		x2)	x2)					3	REAL
I1, J1, K1		s	x	x	x					3	REAL
RPL	Rotation in the plane	s								1	REAL
CR	Circle radius	s								1	Unsigned REAL
AR	Opening angle	s								1	Unsigned REAL
TURN	Number of turns for helix	s								1	Unsigned INT
PL	Parameter interval length	s								1	Unsigned REAL
PW	weight	s								1	Unsigned REAL
SD	Spline degree	m								1	Unsigned INT
TU	Axis angle	s								1	Unsigned INT
STAT	Position of joints	m								1	Unsigned INT
SF	Starting point offset for thread cutting	m								1	REAL
DISCL	Safety clearance SAR	s								1	Unsigned REAL
DISR	Repositioning clearance / SAR clearance	s								1	Unsigned REAL
DISPR	Path differential for repositioning	s								1	Unsigned REAL
ALF	Rapid lift angle	m								1	Unsigned INT
DILF	Rapid lift length	m								1	REAL
FP	Fixed point: Number of fixed point to be approached	s								1	Unsigned INT

Address identifier (default setting)	Address type	Modal/ non-modal	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	PR, PM	QU	Max. number	Data type of the assigned value
RNDM	Modal rounding	m								1	Unsigned REAL
RND	Non-modal rounding	s								1	Unsigned REAL
CHF	Chamfer non-modal	s								1	Unsigned REAL
CHR	Chamfer in original direction of motion	s								1	Unsigned REAL
ANG	Contour angle	s								1	REAL
ISD	Insertion depth	m								1	REAL
DISC	Transition circle overshoot tool radius compensation	m								1	Unsigned REAL
OFFN	Offset contour normal	m								1	REAL
DITS	Thread run-in path	m								1	REAL
DITE	Thread run-out path	m								1	REAL
Corner rounding criteria											
ADIS	Rounding clearance	m								1	Unsigned REAL
ADISPOS	Rounding clearance for rapid traverse	m								1	Unsigned REAL
Measurement											
MEAS	Channel-specific measurement with delete distance-to-go	s								1	INT
MEASF	Channel-specific fast measurement with delete distance-to-go	s								1	INT
MEAW	Channel-specific measurement without delete distance-to-go	s								1	INT
Axis, spindle behavior											
LIMS	Spindle speed limitation	m								1	Unsigned REAL

Address identifier (default setting)	Address type	Modal/non-modal	G90/G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	PR, PM	QU	Max. number	Data type of the assigned value
COARSEA	Block change behavior: Exact stop coarse axial	m									
FINEA	Block change behavior: Exact stop fine axial	m									
IPOENDA	Block change behavior: Interpolator stop axial	m									
DIACYCOFA	Transverse axis: Axial diameter programming OFF in cycles	m									
DIAM90A	Transverse axis: Axial diameter programming for G90	m									
DIAMCHAN	Transverse axis: Transfer of all transverse axes in the diameter programming channel status	m									
DIAMCHANA	Transverse axis: Transfer of the diameter programming channel status	m									
DIAMOFA	Transverse axis: Axial diameter programming OFF	m									
DIAMONA	Transverse axis: Axial diameter programming ON	m									
GP	Position: Indirect programming of position attributes	m									
Feedrates											
FAD	Speed of the slow feed movement	s						x		1	Unsigned REAL
FD	Path feedrate for handwheel override	s								1	Unsigned REAL

Address identifier (default setting)	Address type	Modal/ non-modal	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	PR, PM	QU	Max. number	Data type of the assigned value
FRC	Feedrate for radius and chamfer	s								1	Unsigned REAL
FRCM	Feedrate for radius and chamfer, modal	m								1	Unsigned REAL
FB	Non-modal feedrate	s								1	Unsigned REAL
Nibbling/punching											
SPN	Number of path sections per block	s								1	INT
SPP	Length of a path section	m								1	REAL
Grinding											
ST	Sparking-out time	s								1	Unsigned REAL
SR	Retraction path	s								1	Unsigned REAL
Tool selection											
TCARR	Tool carrier	m								1	INT
Tool management											
DL	Total tool offset	m								1	INT
OEM addresses											
OMA1	OEM address 1	m		x	x	x				1	REAL
OMA2	OEM address 2	m		x	x	x				1	REAL
OMA3	OEM address 3	m		x	x	x				1	REAL
OMA4	OEM address 4	m		x	x	x				1	REAL
OMA5	OEM address 5	m		x	x	x				1	REAL
Miscellaneous											
CUTMOD	Modification of the offset data for rotatable tools (in combination with orientable tool carriers)	m									INT

Address identifier (default setting)	Address type	Modal/ non-modal	G90/ G91	IC	AC	DC, ACN, ACP	CIC, CAC, CDC, CACN, CACP	PR, PM	QU	Max. number	Data type of the assigned value
CUTMODK	Modification of the offset data for rotatable tools (in combination with orientation transformations that have been defined by means of kinematic chains)	m									STRING
TOFF	Tool length offset parallel to the specified geometry axis	m									REAL
TOFFL	Tool length offset in the direction of the tool length component L1, L2 or L3	m									REAL
TOFFR	Tool radius offset	m									REAL
TOFFLR	Simultaneous tool length offset and tool radius offset	m									REAL

1) Absolute end points: Modal, incremental end points: non-modal, otherwise modal/non-modal depending on the G command that determines the syntax.

2) As circle center points, IPO parameters act incrementally. They can be programmed in absolute mode with AC. The address modification is ignored when the parameters have other meanings (e.g. thread lead).

5.4 G commands

5.4.1 G commands

The G commands are divided into G groups. In part programs or synchronized actions, in a block, only a G command of a G group can be written. A G command can be modal or non-modal.

Modal: up to programming another G command of the same G group.

5.4.2 G group 1: Modally valid motion commands

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G0	1	Rapid traverse	+	m		
G1	2	Linear interpolation (linear interpolation)	+	m	x	
G2	3	Circular interpolation clockwise	+	m		
G3	4	Circular interpolation counter-clockwise	+	m		
CIP	5	Circular interpolation through intermediate point	+	m		
ASPLINE	6	Akima spline	+	m		
BSPLINE	7	B spline	+	m		
CSPLINE	8	Cubic spline	+	m		
POLY	9	Polynomial interpolation	+	m		
G33	10	Thread cutting with constant lead	+	m		
G331	11	Tapping	+	m		
G332	12	Retraction (tapping)	+	m		
OEMIPO1	13	Reserved	+	m		
OEMIPO2	14	Reserved	+	m		
CT	15	Circle with tangential transition	+	m		
G34	16	Thread cutting with linear increasing lead	+	m		
G35	17	Thread cutting with linear decreasing lead	+	m		
INVCW	18	Involute interpolation clockwise	+	m		
INVCCW	19	Counter-clockwise involute interpolation	+	m		
G335	20	Turning a convex thread in clockwise direction	+	m		
G336	21	Turning a convex thread in counter-clockwise direction	+	m		

5.4.3 G group 2: Non-modally valid motion, dwell time

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G4	1	Dwell time, preset	-	s		
G63	2	Tapping without synchronization	-	s		
G74	3	Reference point approach with synchronization	-	s		
G75	4	Fixed-point approach	-	s		
REPOSL	5	Linear repositioning	-	s		
REPOSQ	6	Repositioning in a quadrant	-	s		
REPOSH	7	Repositioning in semicircle	-	s		
REPOSA	8	Linear repositioning with all axes	-	s		
REPOSQA	9	Linear repositioning with all axes, geometry axes in quadrant	-	s		
REPOSHA	10	Repositioning with all axes; geometry axes in semicircle	-	s		
G147	11	Approach contour with straight line	-	s		

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G247	12	Approach contour with quadrant	-	s		
G347	13	Approach contour with semicircle	-	s		
G148	14	Leave contour with straight line	-	s		
G248	15	Leave contour with quadrant	-	s		
G348	16	Leave contour with semicircle	-	s		
G5	17	Oblique plunge-cut grinding	-	s		
G7	18	Compensatory motion during oblique plunge-cut grinding	-	s		

5.4.4 G group 3: Programmable frame, working area limitation and pole programming

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
TRANS	1	TRANSLATION: Programmable offset	-	s		
ROT	2	ROTATION: Programmable rotation	-	s		
SCALE	3	SCALE: Programmable scaling	-	s		
MIRROR	4	MIRROR: Programmable mirroring	-	s		
ATRANS	5	Additive TRANSLATION: Additive programmable translation	-	s		
AROT	6	Additive ROTATION: Programmable rotation	-	s		
ASCALE	7	Additive SCALE: Programmable scaling	-	s		
AMIRROR	8	Additive MIRROR: Programmable mirroring	-	s		
-	9	Unassigned	-	-		
G25	10	Minimum working area limitation/spindle speed limitation	-	s		
G26	11	Maximum working area limitation/spindle speed limitation	-	s		
G110	12	Pole programming relative to the last programmed setpoint position	-	s		
G111	13	Polar programming relative to origin of current workpiece coordinate system	-	s		
G112	14	Pole programming relative to the last valid pole	-	s		
G58	15	5th settable work offset	-	s		
G59	16	6th settable work offset	-	s		
ROTS	17	Rotation with solid angle	-	s		
AROTS	18	Additive rotation with solid angle	-	s		

5.4.5 G group 4: FIFO

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
STARTFIFO	1	Start FIFO Execute and simultaneously fill preprocessing memory	+	m	x	
STOPFIFO	2	STOP FIFO Stop machining; fill preprocessing memory until START-FIFO is detected, FIFO is full or end of program	+	m		
FIFOCTRL	3	Activation of automatic preprocessing memory control	+	m		

5.4.6 G group 6: Plane selection

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G17	1	Plane selection 1. – 2. Geometry axis	+	m	x	
G18	2	Plane selection 3. – 1. Geometry axis	+	m		
G19	3	Plane selection 2. – 3. Geometry axis	+	m		

5.4.7 G group 7: Tool radius compensation

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G40	1	No tool radius compensation	+	m	x	
G41	2	Tool radius compensation left of the contour	-	m		
G42	3	Tool radius compensation right of the contour	-	m		

5.4.8 G group 8: Settable work offset

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G500	1	Deactivation of settable work offset (G54 to G59, G507 to G599)	+	m	x	
G54	2	1st settable work offset	+	m		
G55	3	2nd settable work offset	+	m		
G56	4	3rd settable work offset	+	m		
G57	5	4th settable work offset	+	m		
G58	6	5th settable work offset	+	m		
G59	7	6th settable work offset	+	m		

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G507	8	7th settable work offset	+	m		
...	...	...	+	m		
G599	100	99th settable work offset	+	m		
Each of the G commands in this G group is used to activate an adjustable user frame $P_UIFR[ ]. G54 corresponds to frame $P_UIFR[1], G507 corresponds to frame $P_UIFR[7]. The number of adjustable user frames and therefore the number of G commands in this G group can be set using machine data MD28080 $MC_MM_NUM_USER_FRAMES.						

5.4.9 G group 9: Frame and tool offset suppression

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G53	1	Suppression of current frames: Programmable frame including system frame for TOROT and TOFRAME and active adjustable frame (G54 to G57, G505 to G599)	-	s		
SUPA	2	As for G153 including suppression of system frames for actual value setting, scratching, external work offset, PAROT including handwheel offsets (DRF), [external work offset], overlaid movement	-	s		
G153	3	As for G53 including suppression of all channel-specific and/or NCU-global basic frames	-	s		
SUPD	4	Suppression of the active tool offsets	-	s		

5.4.10 G group 10: Exact stop - continuous-path mode

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G60	1	Exact stop	+	m	x	
G64	2	Continuous-path mode with reduced velocity as per the overload factor	+	m		
G641	3	Continuous-path mode with smoothing according to distance criterion (= programmable smoothing clearance)	+	m		
G642	4	Continuous-path mode with smoothing within the defined tolerances	+	m		
G643	5	Continuous-path mode with smoothing within the defined tolerances (block-internal)	+	m		
G644	6	Continuous-path mode with smoothing with maximum possible dynamic response	+	m		

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G645	7	Continuous-path mode with smoothing and tangential block transitions within defined tolerances	+	m		
G646	8	Extended continuous-path mode with reduced velocity as per the overload factor	+	m		

5.4.11 G group 11: Exact stop, non-modal

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G9	1	Exact stop	-	s		

5.4.12 G group 12: Block change criteria at exact stop (G60/G9)

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G601	1	Block change at exact stop fine	+	m	x	
G602	2	Block change at exact stop coarse	+	m		
G603	3	Block change at IPO block end	+	m		

5.4.13 G group 13: Workpiece measuring inch/metric

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G70	1	Input system inches (length)	+	m		
G71	2	Input system metric mm (lengths)	+	m	x	
G700	3	Input system inch, inch/min (lengths + velocity + system variable)	+	m		
G710	4	Input system metric mm, mm/min (lengths + velocity + system variable)	+	m		

5.4.14 G group 14: Workpiece measuring absolute/incremental

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G90	1	Absolute dimension	+	m	x	
G91	2	Incremental dimensions	+	m		

5.4.15 G group 15: Feed type

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G93	1	Inverse-time feedrate rpm	+	m		
G94	2	Linear feedrate in mm/min, inch/min	+	m	x	
G95	3	Revolutional feedrate in mm/rev, inch/rev	+	m		
G96	4	Revolutional feedrate (as for G95) and constant cutting rate	+	m		
G97	5	Revolutional feedrate and constant spindle speed (constant cutting rate OFF)	+	m		
G931	6	Feedrate specified by means of traversing time, deactivate constant path velocity	+	m		
G961	7	Linear feedrate (as for G94) and constant cutting rate	+	m		
G971	8	Linear feedrate and constant spindle speed (constant cutting rate OFF)	+	m		
G942	9	Linear feedrate and constant cutting rate or constant spindle speed	+	m		
G952	10	Revolutional feedrate and constant cutting rate or constant spindle speed	+	m		
G962	11	Linear feedrate or revolutional feedrate and constant cutting rate	+	m		
G972	12	Linear feedrate or revolutional feedrate and constant spindle speed (constant cutting rate OFF)	+	m		
G973	13	Revolutional feedrate without spindle speed limitation and constant spindle speed (G97 without LIMS for ISO mode)	+	m		

5.4.16 G group 16: Feedrate override at inside and outside curvature

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
CFC	1	Constant feedrate at contour effective for internal and external radius	+	m	x	
CFTCP	2	Constant feedrate in tool center point (center point path)	+	m		
CFIN	3	Constant feedrate for internal radius only, acceleration for external radius	+	m		

5.4.17 G group 17: Approach and retraction response, tool offset

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
NORM	1	Normal position at starting and end points	+	m	x	
KONT	2	Travel around contour at starting and end points	+	m		
KONTT	3	Approach/retraction with constant tangent	+	m		
KONTC	4	Approach/retraction with constant curvature	+	m		

5.4.18 G group 18: Corner behavior, tool offset

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G450	1	Transition circle (tool travels around workpiece corners on a circular path)	+	m	x	
G451	2	Intersection of equidistant paths (tool backs off from the workpiece corner)	+	m		

5.4.19 G group 19: Curve transition at beginning of spline

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
BNAT	1	Natural transition to first spline block	+	m	x	
BTAN	2	Tangential transition to first spline block	+	m		
BAUTO	3	Definition of the first spline section by means of the next 3 points	+	m		

5.4.20 G group 20: Curve transition at end of spline

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
ENAT	1	Natural transition to next traversing block	+	m	x	
ETAN	2	Tangential transition to next traversing block	+	m		
EAUTO	3	Definition of the last spline section by means of the last 3 points	+	m		

5.4.21 G group 21: Acceleration profile

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
BRISK	1	Fast non-smoothed path acceleration	+	m	x	
SOFT	2	Soft smoothed path acceleration	+	m		
DRIVE	3	Velocity-dependent path acceleration	+	m		

5.4.22 G group 22: Tool offset type

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
CUT2D	1	2½D TRC	+/-	m	x	
CUT2DF	2	2½D TRC relative to the current frame (inclined plane)	+/-	m		
CUT2DD	9	2½D TRC referred to a differential tool	+/-	m		
CUT2DFD	10	2½D TRC referred to a differential tool relative to the current frame (inclined plane)	+/-	m		

5.4.23 G group 24: Precontrol

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
FFWOF	1	Feedforward control OFF	+	m	x	
FFWON	2	Feedforward control ON	+	m		

5.4.24 G group 26: Repositioning mode for REPOS (modal)

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
RMB	1	Repositioning to start of block	-	m		
RMI	2	Repositioning to interrupt point	-	m	x	
RME	3	Repositioning to end of block	-	m		
RMN	4	Repositioning to the nearest path point	-	m		

5.4.25 G group 28: Working area limitation

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
WALIMON	1	Working area limitation ON	+	m	x	
WALIMOF	2	Working area limitation OFF	+	m		

5.4.26 G group 29: Radius/diameter programming

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
DIAMOF	1	Modal channel-specific diameter programming OFF Deactivation activates channel-specific radius programming.	+	m	x	
DIAMON	2	Modal independent channel-specific diameter programming ON The effect is independent of the programmed dimensions mode (G90/G91).	+	m		
DIAM90	3	Modal dependent channel-specific diameter programming ON The effect is dependent on the programmed dimensions mode (G90/G91).	+	m		
DIAMCYCOF	4	Modal channel-specific diameter programming during cycle processing OFF	+	m		

5.4.27 G group 30: NC block compression

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
COMPOF	1	NC block compression off	+	m	x	
COMPON	2	Compressor function COMPON on	+	m		
COMPCURV	3	Compressor function COMPCURV on	+	m		
COMPCAD	4	Compressor function COMPCAD on	+	m		
COMPSURF	5	Compressor function COMPSURF on	+	m		
COMPPATH	6	Compressor function COMPPATH on	+	m		

5.4.28 G group 33: Settable fine tool offset

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
FTOCOF	1	Online fine tool offset OFF	+	m	x	
FTOCON	2	Online fine tool offset ON	-	m		

5.4.29 G group 37: Feedrate profile

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
FNORM	1	Feedrate normal to DIN 66025	+	m	x	
FLIN	2	Feed linear variable	+	m		
FCUB	3	Feedrate variable according to cubic spline	+	m		

5.4.30 G group 39: Programmable contour accuracy

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
CPRECOF	1	Programmable contour accuracy OFF	+	m	x	
CPRECON	2	Programmable contour accuracy ON	+	m		

5.4.31 G group 40: Tool radius compensation constant

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
CUTCONOF	1	Constant tool radius compensation OFF	+	m	x	
CUTCONON	2	Constant tool radius compensation ON	+	m		

5.4.32 G group 41: Interruptible thread cutting

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
LFOF	1	Interruptible thread cutting OFF	+	m	x	
LFON	2	Interruptible thread cutting ON	+	m		

5.4.33 G group 42: Tool carrier

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
TCOABS	1	Determine tool length components from the current tool orientation	+	m	x	
TCOFR	2	Determine tool length components from the orientation of the active frame	+	m		
TCOFRZ	3	Determine tool orientation of an active frame on selection of tool, tool points in Z direction	+	m		
TCOFRY	4	Determine tool orientation of an active frame on selection of tool, tool points in Y direction	+	m		
TCOFRX	5	Determine tool orientation of an active frame on selection of tool, tool points in X direction		m		

5.4.34 G group 43: SAR approach direction

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G140	1	SAR approach direction defined by G41/G42	+	m	x	
G141	2	SAR approach direction to left of contour	+	m		
G142	3	SAR approach direction to right of contour	+	m		
G143	4	SAR approach direction tangent-dependent	+	m		

5.4.35 G group 44: SAR path segmentation

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G340	1	Spatial approach block; in other words, infeed depth and approach in plane in one block	+	m	x	
G341	2	Start with infeed on perpendicular axis (Z), then approach in plane	+	m		

5.4.36 G group 45: Path reference for FGROUP axes

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
SPATH	1	Path reference for FGROUP axes is arc length	+	m	x	
UPATH	2	Path reference for FGROUP axes is curve parameter	+	m		

5.4.37 G group 46: Plane selection for fast retraction

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
LFTXT	1	The plane is determined from the path tangent and the current tool orientation	+	m	x	
LFWP	2	The plane is determined by the current working plane (G17/G18/G19)	+	m		
LFPOS	3	Axial retraction to a position	+	m		

5.4.38 G group 47: Mode switchover for external NC code

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G290	1	Activate SINUMERIK language mode	+	m	x	
G291	2	Activate ISO language mode	+	m		

5.4.39 G group 48: Approach and retraction response with tool radius compensation

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
G460	1	Collision detection for approach and retraction block ON	+	m	x	
G461	2	Extend border block with arc if no intersection in TRC block	+	m		
G462	3	Extend border block with straight line if no intersection in TRC block	+	m		

5.4.40 G group 49: Point-to-point motion

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
CP	1	Path motion	+	m	x	
PTP	2	Point-to-point motion (synchronized axis motion)	+	m		
PTPG0	3	Point-to-point motion only with G0, otherwise path motion CP	+	m		
PTPWOC	4	Point-to-point motion without compensationary motion, which is caused by orientation changes	+	m		

5.4.41 G group 52: Frame rotation in relation to workpiece

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
PAROTOF	1	Frame rotation in relation to workpiece OFF	+	m	x	
PAROT	2	Frame rotation in relation to workpiece ON The workpiece coordinate system is aligned on the workpiece.	+	m		

5.4.42 G group 53: Frame rotation in relation to tool

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
TOROTOF	1	Frame rotation in relation to tool OFF	+	m	x	
TOROT	2	Align Z axis of the WCS by rotating the frame parallel to the tool orientation	+	m		
TOROTZ	3	As TOROT	+	m		

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
TOROTY	4	Align Y axis of the WCS by rotating the frame parallel to the tool orientation	+	m		
TOROTX	5	Align X axis of the WCS by rotating the frame parallel to the tool orientation	+	m		
TOFRAME	6	Align Z axis of the WCS by rotating the frame parallel to the tool orientation	+	m		
TOFRAMEZ	7	As TOFRAME	+	m		
TOFRAMEY	8	Align Y axis of the WCS by rotating the frame parallel to the tool orientation	+	m		
TOFRAMEX	9	Align X axis of the WCS by rotating the frame parallel to the tool orientation	+	m		

5.4.43 G group 55: Rapid traverse with/without linear interpolation

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
RTLION	1	Rapid traverse motion with linear interpolation ON	+	m	x	
RTLIOF	2	Rapid traverse motion with linear interpolation OFF Rapid traverse motion is achieved with single-axis interpolation.	+	m		

5.4.44 G group 56: Taking into account tool wear

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
TOWSTD	1	Initial setting value for offsets in tool length	+	m	x	
TOWMCS	2	Wear values in the machine coordinate system (MCS)	+	m		
TOWWCS	3	Wear values in the workpiece coordinate system (WCS)	+	m		
TOWBCS	4	Wear values in the basic coordinate system (BCS)	+	m		
TOWTCS	5	Wear values in the tool coordinate system (toolholder ref. point T at the toolholder)	+	m		
TOWKCS	6	Wear values in the coordinate system of the tool head for kinetic transformation (differs from machine coordinate system through tool rotation)	+	m		

5.4.45 G group 57: Corner deceleration

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
FENDNORM	1	Corner deceleration OFF	+	m	x	
G62	2	Corner deceleration at inside corners when tool radius compensation is active (G41/G42)	+	m		
G621	3	Corner deceleration at all corners	+	m		

5.4.46 G group 59: Dynamic response mode for path interpolation

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
DYNNORM	1	Standard dynamic response	+	m	x	
DYNPOS	2	Positioning mode, tapping	+	m		
DYNROUGH	3	Roughing	+	m		
DYNSEMIFIN	4	Rough finishing	+	m		
DYNFINISH	5	Finishing	+	m		
DYNPREC	6	Smooth finishing	+	m		

5.4.47 G group 60: Working area limitation

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
WALCS0	1	Workpiece coordinate system working area limitation OFF	+	m	x	
WALCS1	2	WCS working area limitation group 1 active	+	m		
WALCS2	3	WCS working area limitation group 2 active	+	m		
WALCS3	4	WCS working area limitation group 3 active	+	m		
WALCS4	5	WCS working area limitation group 4 active	+	m		
WALCS5	6	WCS working area limitation group 5 active	+	m		
WALCS6	7	WCS working area limitation group 6 active	+	m		
WALCS7	8	WCS working area limitation group 7 active	+	m		
WALCS8	9	WCS working area limitation group 8 active	+	m		
WALCS9	10	WCS working area limitation group 9 active	+	m		
WALCS10	11	WCS working area limitation group 10 active	+	m		

5.4.48 G group 62: Repositioning mode for REPOS (non-modal)

G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MH
RMBBL	1	Repositioning to start of block	-	s		
RMIBL	2	Repositioning to interrupt point	-	s	x	
RMEBL	3	Repositioning to end of block	-	s		
RMNBL	4	Repositioning to the nearest path point	-	s		

Legend

1) Internal number (e.g. for PLC interface)

2) Configurability of the G command as a reset setting for the G group on power up, reset or end of part program (with MD20150 $MC_GCODE_RESET_VALUES):

+ Configurable

- Not configurable

3) Effectiveness of the G command:

m modal

s Non-modal

4) Reset setting, see the following machine data:

- MD20149GCODE_RESET_S_VALUES (reset position of G groups (fix))
- MD20150 $MC_GCODE_RESET_VALUES (reset position of the G groups)
- MD20151GCODE_RESET_S_MODE (reset behavior of G groups (fix))
- MD20152 $MC_GCODE_RESET_MODE (reset behavior of G groups)
- MD20154 $MC_EXTERN_GCODE_RESET_VALUES (reset position of the G groups in ISO mode)
- MD20156 $MC_EXTERN_GCODE_RESET_MODE (reset behavior of external G groups)

SAG Default setting **S**iemens **AG**

MM Default setting **M**achine **M**anufacturer (see machine manufacturer's specifications)

5.5 Predefined procedures

The call of a predefined procedure triggers the execution of a predefined NC function. A predefined procedure does **not** supply a return value in contrast to a predefined function.

Coordinate system					
Identifier	**Parameter**				**Explanation**
	1st	**2nd**	**3rd - 15th**	**4th - 16th**	
PRESETON	AXIS *): Axis identifier of machine axis	REAL: Preset offset G700/G710 context	As 1 ...	As 2 ...	Set actual value for the programmed axes with loss of the referencing status

Coordinate system					
Identifier	**Parameter**				**Explanation**
PRESETONS	AXIS *): Axis identifier of machine axis	REAL: Preset offset G700/G710 context	As 1 ...	As 2 ...	Set actual value for the programmed axes without loss of the referencing status
DRFOF					Deletes the DRF offset for all axes assigned to the channel.

) As a general rule, geometry or special axis identifiers can also be used instead of the machine axis identifier, as long as the reference is unambiguous.

Axis groupings					
Identifier	**Parameter**				**Explanation**
GEOAX	**1st**	**2nd**	**3rd / 5th**	**4th / 6th**	Selection of a parallel coordinate system
	INT: Geometry axis number 1 ... 3	AXIS: Channel axis identifier	As 1	As 2	
FGROUP	**1st – 8th**				Variable F value reference: Definition of the axes to which the path feedrate refers Maximum number of axes: 8 The default setting for the F value reference is activated with FGROUP ( ) without parameters
	AXIS: Channel axis identifier				
SPLINEPATH	**1st**	**2nd – 9th**			Definition of the spline grouping Maximum number of axes: 8
	INT: Spline grouping (must be 1)	AXIS: Geometry of additional identifier			
POLYPATH	**1st**	**2nd**			Activation of the polynomial interpolation for selective axis groups
	STRING	STRING			

Coupled motion							
Identifier	**Parameter**						**Explanation**
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
TANG	AXIS: Axis name following axis	AXIS: Leading axis 1	AXIS: Leading axis 2	REAL: Coupling factor	CHAR: Option: "B": Tracking in the BCS "W": Tracking in the WCS	CHAR optimization: "S": Standard "P": Autom. with smoothing clearance, angle tolerance	Tangential control: Define coupling The tangent for the follow-up is determined by the two master axes specified. The coupling factor specifies the relationship between a change in the angle of tangent and the following axis. It is usually 1.
TANGON	AXIS: Axis name following axis	REAL: Offset angle	REAL: Smoothing clearance	REAL: Angular tolerance			Tangential control: Activate coupling
TANGOF	AXIS: Axis name following axis						Tangential control: Deactivate coupling
TLIFT	AXIS: Tracked axis						Tangential control: Activate intermediate block generation
TRAILON	AXIS: Following axis	AXIS: Leading axis	REAL: Coupling factor				Asynchronous coupled motion OFF
TRAILOF	AXIS: Following axis	AXIS: Leading axis					Deactivate coupled-axis motion
TANGDEL	AXIS: Following axis						Tangential control: Delete coupling

Axial acceleration profile		
Identifier	**Parameter**	**Explanation**
	1st – 8th	
BRISKA	AXIS	Activate stepped axis acceleration for the programmed axes
SOFTA	AXIS	Activate jerk-limited axis acceleration for the programmed axes

Axial acceleration profile		
Identifier	**Parameter**	**Explanation**
	1st – 8th	
DRIVEA	AXIS	Activate knee-shaped acceleration characteristic for the programmed axes
JERKA	AXIS	The acceleration behavior set in machine data $MA_AX_JERK_ENABLE is active for the programmed axes

Revolutional feedrate			
Identifier	**Parameter**		**Explanation**
FPRAON	**1st**	**2nd**	Axial revolutional feedrate ON
	AXIS: Axis for which revolutional feedrate is activated	AXIS: Axis/spindle from which revolutional feedrate is derived. If no axis has been programmed, the revolutional feedrate is derived from the master spindle.	
FPRAOF	**1st - nth**		Axial revolutional feedrate OFF
	AXIS: Axes for which revolutional feedrate is deactivated		The revolutional feedrate can be deactivated for several axes simultaneously. You can program as many axes as are permitted in a block.
FPR	**1st**		Selection of a rotary axis or spindle from which the feed rate per revolution of the path is derived for G95. The setting made with FPR is modal.
	AXIS: Axis/spindle from which revolutional feedrate is derived. If no axis has been programmed, the revolutional feedrate is derived from the master spindle.		

Transformations				
Identifier	**Parameter**			**Explanation**
	1st	**2nd**	**3rd**	
TRACYL	REAL: Working diameter	INT: Number of the transformation		Cylinder: Peripheral surface transformation Several transformations can be set per channel. The transformation number specifies which transformation is to be activated. If the 2nd parameter is omitted, the transformation grouping set via MD is activated.
TRANSMIT	INT: Number of the transformation			Transmit: Polar transformation Several transformations can be set per channel. The transformation number specifies which transformation is to be activated. If the parameter is omitted, the transformation group defined in the MD is activated.
TRAANG	REAL: Angle	INT: Number of the transformation		Transformation inclined axis Several transformations can be set per channel. The transformation number specifies which transformation is to be activated. If the 2nd parameter is omitted, the transformation grouping set via MD is activated. If the angle is not programmed (TRAANG (,2) or TRAANG) the last angle applies modally.
TRACON	INT: Number of the transformation	REAL: Further parameters, MD-dependent		Cascaded transformation The meaning of the parameters depends on the type of cascading.
TRAFOOF				Deactivate transformation
TRAFOON	STRING: Name of the transformation data set	REAL: Reference or working diameter (TRACYL only)	BOOL: With/without groove side offset (TRACYL only)	Activate a transformation defined with kinematic chains

Spindle			
Identifier	**Parameter**		**Explanation**
	1	**2nd - nth**	
SPCON	INT: Spindle number	INT: Spindle number	Switch to position-controlled spindle operation.
SPCOF	INT: Spindle number	INT: Spindle number	Switch to speed-controlled spindle operation.
SETMS	INT: Spindle number		Declaration of spindle as master spindle for the current channel With SETMS( ), the machine data default applies automatically without any need for parameterization.

Grinding		
Identifier	**Parameter**	**Explanation**
	1st	
GWPSON	INT: Spindle number	Constant grinding wheel peripheral speed ON If the spindle number is not programmed, the grinding wheel peripheral speed for the spindle of the active tool is selected.
GWPSOF	INT: Spindle number	Constant grinding wheel peripheral speed OFF If the spindle number is not programmed, the grinding wheel peripheral speed for the spindle of the active tool is deselected.
TMON	INT: T number	Grinding-specific tool monitoring ON If no T number is programmed, monitoring is activated for the active tool.
TMOF	INT: T number	Tool monitoring OFF If no T number is programmed, monitoring is deactivated for the active tool.

Stock removal					
Identifier	**Parameter**				**Explanation**
	1st	**2nd**	**3rd**	**4th**	
CONTPRON	REAL [,11]: Contour table	CHAR: Machining type	INT: Number of relief cuts	INT: Status of the calculation	Activate reference preprocessing The contour programs or NC blocks which are called in the following steps are divided into individual movements and stored in the contour table. The number of relief cuts is returned.
CONTDCON	REAL [, 6]: Contour table	INT: Machining direction			Contour decoding The blocks for a contour are stored in a named table with one table line per block and coded to save memory.
EXECUTE	INT: Error status				Activate program execution This switches back to normal program execution from reference point editing mode or after setting up a protection area.

Execute table		
Identifier	**Parameter**	**Explanation**
	1st	
EXECTAB	REAL [11]: Element from motion table	Execute an element from a motion table

Protection areas						
Identifier	**Parameter**					**Explanation**
	1st	**2nd**	**3rd**	**4th**	**5th**	
CPROTDEF	INT: Number of the protection area	BOOL: TRUE: Tool-related protection area	INT: 0: 4th and 5th parameters are not evaluated 1: 4th parameter is evaluated 2: 5th parameter is evaluated 3: 4th and 5th parameters are evaluated	REAL: Limit in plus direction	REAL: Limit in minus direction	Definition of a channel-specific protection area
NPROTDEF	INT: Number of the protection area	BOOL: TRUE: Tool-related protection area	INT: 0: 4th and 5th parameters are not evaluated 1: 4th parameter is evaluated 2: 5th parameter is evaluated 3: 4th and 5th parameters are evaluated	REAL: Limit in plus direction	REAL: Limit in minus direction	Definition of a machine-specific protection area

Protection areas						
Identifier	**Parameter**					**Explanation**
	1st	**2nd**	**3rd**	**4th**	**5th**	
CPROT	INT: Number of the protection area	INT: Option 0: Protection area OFF 1: Preactivate protection area 2: Protection area ON 3: Preactivate protection area with conditional stop, only with protection areas active	REAL: Offset of the protection area in the first geometry axis	REAL: Offset of the protection area in the second geometry axis	REAL: Offset of the protection area in the third geometry axis	Channel-specific protection area ON/OFF
NPROT	INT: Number of the protection area	INT: Option 0: Protection area OFF 1: Preactivate protection area 2: Protection area ON 3: Preactivate protection area with conditional stop, only with protection areas active	REAL: Offset of the protection area in the first geometry axis	REAL: Offset of the protection area in the second geometry axis	REAL: Offset of the protection area in the third geometry axis	Machine-specific protection area ON/OFF

Preprocessing / single block		
Identifier	**Parameter**	**Explanation**
STOPRE		Preprocessing stop until all prepared blocks in the main run are executed
SBLOF		Suppress single-block processing
SBLON		Cancel suppression of the single-block processing

Interrupts		
Identifier	**Parameter**	**Explanation**
	1st	
DISABLE	INT: Number of the interrupt input	Deactivates the interrupt routine assigned to the specified hardware input. Fast retraction is not executed. The assignment between the hardware input and the interrupt routine made with SETINT remains valid and can be reactivated with ENABLE.
ENABLE	INT: Number of the interrupt input	Reactivation of the interrupt routine assignment deactivated with DISABLE.
CLRINT	INT: Number of the interrupt input	Delete assignment of interrupt routines and attributes to an interrupt input. The interrupt routine is deactivated and no reaction occurs when the interrupt is generated.

Synchronized actions		
Identifier	**Parameter**	**Explanation**
	1st - nth	
CANCEL	INT: Number of the synchronized action	Aborts the modal synchronized action with the specified ID. Several IDs, separated by commas, can be specified.
CANCELSUB		Cancel current subprogram level

Function definition					
Identifier	**Parameter**				**Explanation**
	1st	**2nd**	**3rd**	**4th-7th**	
FCTDEF	INT: Function number	REAL: Lower limit value	REAL: Upper limit value	REAL: Coefficients a0 - a3	Define polynomial function This is evaluated in SYFCT or PUTFTOCF.

Communication				
Identifier	**Parameter**			**Explanation**
	1st	**2nd**		

Communication						
Identifier	**Parameter**					**Explanation**
MMC	STRING: Command	CHAR: Acknowledgement mode*) "N": Without acknowledgement "S": Synchronous acknowledgement "A": Asynchronous acknowledgement				Command to HMI command Interpreter for the configuration of windows via NC program
	1st	**2nd**	**3rd**	**4th**	**5th**	
DRVPRD	VAR INT: Status	AXIS: Machine axis name	INT: Number of the drive parameter	INT: Index of the drive parameter	VAR REAL: Read value of the drive parameter	Reading of drive parameters on part program level
DRVPWR	VAR INT: Status	AXIS: Machine axis name	INT: Number of the drive parameter	INT: Index of the drive parameter	REAL: Value of the drive parameter to be written	Writing of drive parameters on part program level

) Commands are acknowledged on request from the executing component (channel, NC, etc.).

Program coordination					
Identifier	**Parameter**				**Explanation**
INIT	**1st**	**2nd**	**3rd**		Selection of an NC program for execution in a channel
	INT: Channel number **or** channel name from MD20000*)	STRING: Path specification	CHAR: Acknowledgement mode**)		
	1st - nth				
START	INT: Channel number **or** channel name from MD20000 *)				Start selected programs simultaneously in several channels from current program This command has no effect for the own channel
WAITE	INT: Channel number **or** channel name from MD20000 *)				Wait for end of program in one or more other channels

Program coordination			
Identifier	**Parameter**		**Explanation**
	1st	**2nd - nth**	
WAITM	INT: Marker number	INT: Channel number **or** channel name from MD20000 *)	Wait until a marker is reached in the specified channels The previous block is terminated with exact stop
WAITMC	INT: Marker number	INT: Channel number **or** channel name from MD20000 *)	Wait until a marker is reached in the specified channels An exact stop is initiated only if the other channels have not yet reached the marker
	1st - nth		
SETM	INT: Marker number		Set one or more markers for the channel coordination The processing in own channel is not affected by this.
CLEARM	INT: Marker number		Delete one or more markers for the channel coordination The processing in own channel is not affected by this.
	1st - nth		
WAITP	AXIS: Axis identifier		Wait until the specified positioning axes that were previously programmed with POSA, reach their programmed end point
WAITS	INT: Spindle number		Wait until the specified spindles that were previously programmed with SPOSA, reach their programmed end point

<table>
<tr><th colspan="6">Program coordination</th></tr>
<tr><th>Identifier</th><th colspan="4">Parameter</th><th>Explanation</th></tr>
<tr><td rowspan="2">RET</td><td>1st</td><td>2nd</td><td>3rd</td><td>4th</td><td rowspan="3">End of subprogram with no function output to the PLC
If the 1st parameter (jump destination) is specified, the return jump is first made to the block after the calling block. The target is then sought depending on the programming (RET or RETB) according to the following strategy:
• RET:
Search in the direction of the end of the program. A search is made toward the start of the program if the search was not successful.
• RETB:
Search in the direction of the start of the program. A search is made toward the end of the program if the search was not successful.</td></tr>
<tr><td>INT (or STRING):
Jump target (block no./ marker) for return</td><td>INT:
0:
Return to jump destination from 1st par.
> 0:
Return to the following block</td><td>INT:
Number of subprogram levels to be skipped</td><td>BOOL:
Return to first block in the main program</td></tr>
<tr><td>RETB</td><td>INT (or STRING):
Jump target (block no./ marker) for return</td><td>INT:
0:
Return to jump destination from 1st par.
> 0:
Return to the following block</td><td>INT:
Number of subprogram levels to be skipped</td><td>BOOL:
Return to first block in the main program</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td></td><td colspan="4">1st - nth</td><td></td></tr>
<tr><td>GET</td><td colspan="4">AXIS:
Axis identifier ***)</td><td>Assign machine axis(axes)
The specified axes must be released in the other channel with RELEASE</td></tr>
<tr><td>GETD</td><td colspan="4">AXIS:
Axis identifier ***)</td><td>Assign machine axis(axes) directly
The specified axes must not be released with RELEASE</td></tr>
<tr><td>RELEASE</td><td colspan="4">AXIS:
Axis identifier ***)</td><td>Release machine axis(axes)</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td></td><td>1st</td><td>2nd</td><td>3rd</td><td>4th</td><td></td></tr>
<tr><td>PUTFTOC</td><td>REAL:
Offset value</td><td>INT:
Parameter number</td><td>INT:
Channel number
or
channel name from MD20000*)</td><td>INT: Spindle number</td><td>Change of fine tool compensation</td></tr>
<tr><td>PUTFTOCF</td><td>INT:
No. of the function</td><td>VAR REAL:
Reference value</td><td>INT: Parameter number</td><td>INT:
Channel number
or
channel name from MD20000*)</td><td>Change of fine tool compensation depending on a function defined with FCTDEF
(max. 3rd degree polynomial)
The number used here must be specified in FCTDEF</td></tr>
<tr><td colspan="6"></td></tr>
</table>

Program coordination					
Identifier	**Parameter**				**Explanation**
AXTOCHAN	**1st**	**2nd**	**3rd - nth**	**4th - mth**	Axes transferred to other channels
	AXIS: Axis identifi- er	INT: Channel number **or** channel name from MD20000*)	As 1 ...	As 2 ...	

) Instead of channel numbers, the channel names defined via MD20000 $MC_CHAN_NAME can also be programmed.

**) Commands are acknowledged on request from the executing component (channel, NC, etc.).

***) The SPI function can be used to program a spindle instead of an axis. E.g. GET(SPI(1))

Data access		
Identifier	**Parameter**	**Explanation**
CHANDATA	**1st**	Set channel number for channel data access (only permitted in the initialization block). The following access refers to the channel set with CHANDATA.
	INT: Channel num- ber	
NEWCONF		Accept changed machine data

Messages			
Identifier	**Parameter**		**Explanation**
	1st	**2nd**	
MSG	STRING: Message	INT: Execution	Output arbitrary character string as message on the user in- terface
WRTPR	STRING: Character string	INT: Execution	Write string in OPI variable

File access						
Identifier	**Parameter**					**Explanation**
READ	**1st**	**2nd**	**3rd**	**4th**	**5th**	Read blocks from file system
	VAR INT: Error	CHAR[160]: File name	INT: Start line of the file section to be read	INT: Number of lines to be read	VAR CHAR[255]: Variable array in which the read informa- tion is stored	

File access						
Identifier	**Parameter**					**Explanation**
WRITE	**1st**	**2nd**	**3rd**	**4th**		Write block to file system (or to an external device/file)
	VAR INT: Error	CHAR[160]: File name	STRING: Device/file for external output	CHAR[200]: Block		
DELETE	**1st**	**2nd**				Delete file
	VAR INT: Error	CHAR[160]: File name				

Alarms			
Identifier	**Parameter**		**Explanation**
	1st	**2nd**	
SETAL	INT: Alarm number (cycle alarms)	STRING: Character string	Set alarm A character string with up to four parameters can be specified in addition to the alarm number. The following predefined parameters are available: %1 = channel number %2 = block number, label %3 = text index for cycle alarms %4 = additional alarm parameters

Tool management							
Identifier	**Parameter**						**Explanation**
	1st	**2nd**					
DELDL	INT: T no.	INT: D no.					Delete all additive offsets of the tool edge (or of a tool if D is not specified)
DELT	STRING[32]: Tool identifier	INT: Duplo no.					Delete tool Duplo number can be omitted
DELTC	INT: Data set no. n	INT: Data set no. m					Delete tool carrier data set number n to m
DZERO							Set D numbers of all tools of the TO unit assigned to the channel to invalid
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	

Tool management							
Identifier	**Parameter**						**Explanation**
GETFREELOC	VAR INT: Magazine no. (return value)	VAR INT: Location no. (return value)	INT: T no.	INT: Reference magazine no.	CHAR: Specification dep. on 4th parameter	INT: Reservation mode	Find empty location for a tool
	1st	**2nd**					
GETSELT	VAR INT: T no. (return value)	INT: Spindle no.					Returns the T number of the tool preselected for the spindle
GETEXET	VAR INT: T no. (return value)	INT: Spindle no.					Returns the T number of the tool active from the point of view of the NC program
GETTENV	STRING: Name of the tool environment	INT ARRAY[3]: Return values					Reads the T, D and DL numbers stored in a tool environment
	1st	**2nd**	**3rd**	**4th**			
POSM	INT: No. of the location for positioning	INT: No. of the magazine to be moved	INT: Location no. of the internal magazine	INT: Magazine no. of the internal magazine			Position magazine
RESETMON	VAR INT: Status = result of the operation (return value)	INT: Internal T no.	INT: D no. of the tool	INT: Optional bit-coded parameter			Set actual value of tool to setpoint
SETDNO	**1st**	**2nd**	**3rd**				Set offset number (D) of the cutting edge of the tool (T)
	INT: T no.	INT: Cutting edge no.	INT: D no.				
SETMTH	**1st**						Set tool carrier no.
	INT: Tool carrier no.						
SETPIECE	**1st**	**2nd**					Decrement workpiece counter of the spindle Update the count monitoring data of the tools associated with the machining process
	INT: Value used when decrementing	INT: Spindle no.					

Tool management							
Identifier	**Parameter**						**Explanation**
	1st	**2nd**	**3rd**	**4th**			
SETTA	VAR INT: Status = result of the operation (return value)	INT: Magazine no.	INT: Wear group no.	INT: Tool subgroup			Activate tool from wear group
SETTIA	VAR INT: Status = result of the operation (return value)	INT: Magazine no.	INT: Wear group no.	INT: Tool subgroup			Deactivate tool from wear group
TCA	**1st**	**2nd**	**3rd**				Tool selection/change irrespective of the tool status
	STRING[32]: Tool identifier	INT: Duplo no.	INT: Tool carrier no.				
TCI	**1st**	**2nd**					Load tool from buffer into the magazine
	INT: No. of the buffer	INT: Tool carrier no.					
MVTOOL	**1st**	**2nd**	**3rd**	**4th**	**5th**		Language command to move tool
	INT: Status	INT: Magazine no.	INT: Location no.	INT: Magazine no. after moving	INT: Target location no. after moving		

Synchronous spindle							
Identifier	**Parameter**						**Explanation**
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
COUPDEF	AXIS: Following spindle	AXIS: Leading spindle	REAL: Numerator of transmission ratio	REAL: Denominator of transmission ratio	STRING[8]: Block change behavior	STRING[2]: Coupling type	Define synchronous spindle grouping
COUPDEL	AXIS: Following spindle	AXIS: Leading spindle					Delete synchronous spindle grouping
COUPRES	AXIS: Following spindle	AXIS: Leading spindle					Reset coupling parameters to configured MD and SD values

Synchronous spindle							
Identifier	**Parameter**						**Explanation**
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
COUPON	AXIS: Following spindle	AXIS: Leading spindle	REAL: Switch-on position of the following spindle				Switch-on synchronous spindle coupling If a switch-on position is specified for the following spindle (angular misalignment between FS and LS that refers -- absolutely or incrementally -- to the zero degree position of the LS in the positive direction of rotation), the coupling is only switched on when the specified position is crossed.
COUPONC	AXIS: Following spindle	AXIS: Leading spindle					Switch-on synchronous spindle coupling With COUPONC, the currently active speed of the following spindle is taken over when switching on the coupling (M3/M4 S...).
COUPOF	AXIS: Following spindle	AXIS: Leading spindle	REAL: Switch-off position of the following spindle (absolute)	REAL: Switch-off position of the leading spindle (absolute)			Switch-off synchronous spindle coupling If positions are specified, the coupling is only cancelled when all the specified positions have been overtraveled The following spindle continues to revolve at the last speed programmed before deactivation of the coupling

Synchronous spindle							
Identifier	**Parameter**						**Explanation**
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
COUPOFS	AXIS: Following spindle	AXIS: Leading spindle	REAL: Switch-off position of the following spindle (absolute)				Switch off the synchronous spindle coupling with stop of the following spindle If a position is specified, the coupling is only cancelled when the specified position is crossed
WAITC	AXIS: Following spindle	STRING [8]: Block change behavior	AXIS: Following spindle	STRING[8]: Block change behavior			Wait until the coupling block change criterion for the spindles (max. 2) has been fulfilled If the block change behavior is not specified, the block change behavior specified in the definition with COUPDEF applies

Electronic gear							
Identifier	**Parameter**						**Explanation**
EGDEL	**1st**						Delete coupling definition for the following axis
	AXIS: Following axis						
EGDEF	**1st**	**2nd / 4th / 6th / 8th / 10th**	**3rd / 5th / 7th / 9th / 11th**				Definition of an electronic gear
	AXIS: Following axis	AXIS: Leading axis	INT: Coupling type				
EGON	**1st**	**2nd**	**3rd / 6th / 9th / 12nd / 15th**	**4th / 7th / 10th / 13rd / 16th**	**5th / 8th / 11st / 14th / 17th**		Electronic gear ON without synchronization
	AXIS: Following axis	STRING: Block change behavior	AXIS: Leading axis	REAL: Numerator of the coupling factor	REAL: Denominator of the coupling factor		

Electronic gear									
Identifier	**Parameter**								**Explanation**
EGONSYN	**1st**	**2nd**	**3rd**	**4th / 8th / 12nd / 16th / 20th**	**5th / 9th / 13rd / 17th / 21st**	**6th / 10th / 14th / 18th / 22nd**	**7th / 11st / 15th / 19th / 23rd**		Electronic gear ON with synchronization
	AXIS: Following axis	STRING: Block change behavior	REAL: Synchron ized position of the following axis	AXIS: Leading axis	REAL: Synchron ized position of the leading axis	REAL: Numerator of the coupling factor	REAL: Denominator of the coupling factor		
EGONSYNE	**1st**	**2nd**	**3rd**	**4th**	**5th / 9th / 13rd / 17th / 21st**	**6th / 10th / 14th / 18th / 22nd**	**7th / 11st / 15th / 19th / 23rd**	**8th / 12nd / 16th / 20th / 24th**	Electronic gear ON with synchronization and specification of the approach mode
	AXIS: Following axis	STRING: Block change behavior	REAL: Synchron ized position of the following axis	STRING: Approach mode	AXIS: Leading axis	REAL: Synchron ized position of the leading axis	REAL: Numerator of the coupling factor	REAL: Denominator of the coupling factor	
EGOFS	**1st**	**2nd - nth**							Turn off electronic gear selectively
	AXIS: Following axis	AXIS: Leading axis							
EGOFC	**1st**								Switch off electronic gear (version only for spindles)
	AXIS: Following spindle								

Information functions in the passive file system				
Identifier	**Parameter**			**Explanation**
	1st	**2nd**	**3rd**	
FILEDATE	VAR INT: Error message	CHAR[160]: File name	VAR CHAR[8]: Date in the format "dd.mm.yy"	Returns the date of the last write access to a file
FILETIME	VAR INT: Error message	CHAR[160]: File name	VAR CHAR[8]: Time in the format "hh.mm.ss"	Returns the time of the last write access to a file

Information functions in the passive file system				
Identifier	**Parameter**			**Explanation**
	1st	**2nd**	**3rd**	
FILESIZE	VAR INT: Error message	CHAR[160]: File name	VAR INT: File size	Returns the current size of a file
FILESTAT	VAR INT: Error message	CHAR[160]: File name	VAR CHAR[5]: Date in the format "rwxsd"	Returns the status of a file with respect to the following rights: • Read (r: read) • Write (w: write) • Execute (x: execute) • Show (s: show) • Delete (d: delete)
FILEINFO	VAR INT: Error message	CHAR[160]: File name	VAR CHAR[32]: Date in the format "rwxsd nnnnnnnn dd.mm.yy hh:mm:ss"	Returns the sum of the information for a file that can be read out via FILEDATE, FILETIME, FILESIZE, and FILESTAT

Main/sub-coupling		
Identifier	**Parameter**	**Explanation**
	1st - nth	
MASLON	AXIS: Axis identifier	Switch on main/sub-coupling
MASLOF	AXIS: Axis identifier	Disconnect main/sub-coupling
MASLOFS	AXIS: Axis identifier	Disconnect main/sub-coupling and brake sub-spindles automatically
MASLDEF	AXIS: Axis identifier	Define main/sub-coupling The last axis is the main axis.
MASLDEL	AXIS: Axis identifier	Disconnect main/sub-coupling and clear grouping definition

Online tool length compensation			
Identifier	**Parameter**		**Explanation**
	1st	**2nd**	
TOFFON	AXIS: Offset direction	REAL: Offset value in offset direction	Activate online tool length compensation in the specified offset direction
TOFFOF	AXIS: Offset direction		Reset online tool length compensation in the specified offset direction

SERUPRO		
Identifier	**Parameter**	**Explanation**
IPTRLOCK		Start of untraceable program section
IPTRUNLOCK		End of search-suppressed program section

Retraction				
Identifier	**Parameter**			**Explanation**
	1st - nth			
POLFMASK	AXIS: Geometry or machine axis name			Enable axes for rapid retraction (without a connection between the axes)
POLFMLIN	AXIS: Geometry or machine axis name			Enable axes for linear rapid retraction
POLFA	**1st**	**2nd**	**3rd**	Retraction position for single axes
	AXIS: Channel axis identifier	INT: Type	REAL: Value	

Collision avoidance			
Identifier	**Parameter**		**Explanation**
	1st		
PROTA	STRING: "R"		Request for a recalculation of the collision model
PROTS	**1st**	**2nd - nth**	Set protection area status
	CHAR: Status	STRING: Protection area name	

Jerk adaptation		
Identifier	**Parameter**	**Explanation**
AFISON		Activate automatic filter chain switchover
AFISOF		Deactivate automatic filter chain switchover

5.6 Predefined procedures in synchronized actions

The following predefined procedures are only available in synchronized actions.

Synchronous procedures		
Identifier	**Parameter**	**Explanation**
STOPREOF		Revoke preprocessing stop A synchronized action with a STOPREOF command causes a preprocessing stop after the next output block (= block for the main run). The preprocessing stop is canceled with the end of the output block or when the STOPREOF condition is fulfilled. All synchronized action operations with the STOPREOF command are therefore interpreted as having been executed.
RDISABLE		Read-in disable
DELDTG	**1.** AXIS: Axis for axial delete distance-to-go (optional). If the axis is omitted, delete distance-to-go is triggered for the path distance.	Delete distance-to-go A synchronized action with a DELDTG command causes a preprocessing stop after the next output block (= block for the main run). The preprocessing stop is canceled with the end of the output block or when the first DELDTG condition is fulfilled. The axial distance to the destination point on an axial delete distance-to-go is stored in $AA_DELT[axis]; the distance-to-go is stored in $AC_DELT.

Program coordination of technology cycles		
Identifier	**Parameter**	**Explanation**
	1.	
LOCK	INT: ID of the synchronized action to be disabled	Lock synchronized action with ID or stop technology cycle One or more IDs can be programmed
UNLOCK	INT: ID of the synchronized action to be unlocked	Unlock synchronized action with ID or continue technology cycle One or more IDs can be programmed
ICYCON		Each block of a technology cycle is processed in a separate interpolation cycle following ICYCON
ICYCOF		All blocks of a technology cycle are processed in one interpolation cycle following ICYCOF

Polynomial functions						
Identifier	**Parameter**					**Explanation**
SYNFCT	**1.**	**2.**	**3.**			If the condition in the motion-synchronous action is fulfilled, the polynomial determined by the first expression is evaluated at the input variable. The upper and lower range of the value is limited and the result variable is assigned.
	INT: Number of the polynomial function defined with FCTDEF	VAR REAL: Result variable *)	VAR REAL: Input variable **)			
FTOC	**1.**	**2.**	**3.**	**4.**	**5.**	Change of fine tool compensation depending on a function defined with FCTDEF (max. 3rd degree polynomial). The number used here must be specified in FCTDEF.
	INT: Number of the polynomial function defined with FCTDEF	VAR REAL: Input variable **)	INT: Length 1, 2, 3	INT: Channel number	INT: Spindle number	

*) Only special system variables are permissible as a result variable (see Function Manual Synchronized Actions).

**) Only special system variables are permissible as input variable (see Function Manual Synchronized Actions).

5.7 Predefined functions

The call of a predefined function triggers the execution of a predefined NC function, which in contrast to the predefined procedure, supplies a return value. The call of the predefined function can be an operand in an expression.

Coordinate system						
Identifier	**Return value**	**Parameter**				**Explanation**
		1st	**2nd**	**3rd - 15th**	**4th - 16th**	
CTRANS	FRAME	AXIS: Axis identifier	REAL: Offset	AXIS: Axis identifier	REAL: Offset	Translation: Zero offset COARSE for multiple axes
CFINE	FRAME	AXIS: Axis identifier	REAL: Offset	AXIS: Axis identifier	REAL: Offset	Translation: Zero offset for FINE multiple axes
CSCALE	FRAME	AXIS: Axis identifier	REAL: Scale factor	AXIS: Axis identifier	REAL: Scale factor	Scale: Scale factor for multiple axes
		1st	**2nd**	**3. and 5.**	**4. and 6.**	

Coordinate system						
Identifier	**Return value**	**Parameter**				**Explanation**
CROT	FRAME	AXIS: Axis identifier	REAL: Rotation	AXIS: Axis identifier	REAL: Rotation	Rotation: Rotation of the current coordinate system Maximum number of parameters: 6 (one axis identifier and one value per geometry axis)
CROTS	FRAME	AXIS: Axis identifier	REAL: Rotation with solid angle	AXIS: Axis identifier	REAL: Rotation with solid angle	Rotation: Rotation of the current coordinate system with solid angle Maximum number of parameters: 6 (one axis identifier and one value per geometry axis)
CMIRROR		**1st**	**2nd – 8th**			Mirror: Mirror on a coordinate axis
	FRAME	AXIS	AXIS			
		1st	**2nd**			
CRPL	FRAME	INT: Rotary axis	REAL: Angle of rotation			Frame rotation in any plane
ADDFRAME	INT: 0: OK 1: Specified target (string) is wrong 2: Target frame is not configured 3: Rotation in frame is not permitted	FRAME: Additively measured or calculated frame	STRING: Specified target frame			Calculates the target frame specified by the string The target frame is calculated in such a way that the new complete frame appears as a chain of the old complete frame and the transferred frame.
INVFRAME	FRAME	**1st**				Calculates the inverse frame from a frame The frame chaining of a frame with its inverse frame always results in a zero frame
		FRAME				

Coordinate system						
Identifier	**Return value**	**Parameter**				**Explanation**
		1st	**2nd**	**3rd**		
MEAFRAME	FRAME	REAL[3,3]: Coordinates of the measured spatial points	REAL[3,3]: Coordinates of the specified points	VAR REAL: Variable with which the information on the quality of FRAME calculation is returned		Frame calculation from 3 measuring points in space

Geometry functions					
Identifier	**Return value**	**Parameter**			**Explanation**
		1st	**2nd**	**3rd**	
CALCDAT	BOOL: Error status	VAR REAL [n, 2]: Table (abscissa, ordinate) of points 1 to n	INT: Number of points	VAR REAL [3]: Result: Abscissa, ordinate and radius of calculated circle center point	Calculates the center point coordinates and the radius of the circle from 3 or 4 points The points must be different.
INTERSEC	BOOL: Error status	VAR REAL [11]: First contour element	VAR REAL [11]: Second contour element	VAR REAL [2]: Result vector for the intersection coordinates: Abscissa and ordinate	Calculates the intersection coordinates between two contour elements. The error status indicates whether an intersection was found.

Curve table functions								
Identifier	**Return value**	**Parameter**						**Explanation**
		1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
CTAB	REAL: Following axis position	REAL: Leading axis position	INT: Table number	VAR REAL[ ]: Pitch result	AXIS: Following axis for scaling	AXIS: Leading axis for scaling		Determines the following axis position to the specified leading axis position from the curve table. If parameters 4/5 are not programmed, calculation is with standard scaling.
CTABINV	REAL: Leading axis position	REAL: Following axis position	REAL: Leading position	INT: Table number	VAR REAL[ ]: Pitch result	AXIS: Following axis for scaling	AXIS: Leading axis for scaling	Determines the leading axis position to the specified following axis position from the curve table. If parameters 5/6 are not programmed, calculation is with standard scaling.

Curve table functions								
Identifier	**Return value**	**Parameter**						**Explanation**
		1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
CTABID	INT: Curve table number	INT: Entry number in memory	STRING: Storage location: "SRAM", "DRAM"					Determines the curve table number entered under the specified number in the memory.
CTABISLOCK	INT: Lock state	INT: Table number						Determines the lock state of the curve table: > 0: Table is locked 1: CTABLOCK 2: Active coupling 3: CTABLOCK and active coupling 0: Table is not locked -1: Table does not exist
CTABEXISTS	INT: Existence	INT: Table number						Determines the existence of the curve table in the static or dynamic NC memory: 0: FALSE 1: TRUE
CTABMEMTYP	INT: Storage location	INT: Table number						Determines the storage location of the curve table: 1: DRAM 0: SRAM -1: Table does not exist
CTABPERIOD	INT: Periodicity	INT: Table number						Determines the periodicity of the curve table: 0: Not periodic 1: Periodic in leading axis 2: Periodic in leading and following axis -1: Table does not exist
CTABNO	INT: Number of curve tables							Determines the number of defined curve tables (in static and dynamic NC memory)
CTABNOMEM	INT: Number of curve tables	STRING: Storage location: "SRAM", "DRAM"						Determines the number of defined curve tables in the specified memory

Curve table functions								
Identifier	**Return value**	**Parameter**						**Explanation**
		1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
CTABFNO	INT: Number of tables	STRING: Storage location: "SRAM", "DRAM"						Determines the number of curve tables still possible in the specified memory
CTABSEG	INT: Number of curve segments	STRING: Storage location: "SRAM", "DRAM"	STRING: Segment type: "L": Linear "P": Polynomial					Determines the number of curve segments used of the specified segment type in the specified memory >=0: Number -1: Invalid memory type If parameter 2 is not programmed, the sum of the linear and polynomial segments is output.
CTABFSEG	INT: Number of curve segments	STRING: Storage location: "SRAM", "DRAM"	STRING: Segment type: "L": Linear "P": Polynomial					Determines the number of still possible curve segments of the specified segment type in the specified memory >=0: Number -1: Invalid memory type
CTABSEGID	INT: Number of curve segments	INT: Table number	STRING: Segment type: "L": Linear "P": Polynomial					Determines the number of curve segments of the specified segment type that are used by the curve table >=0: Number -1: Table does not exist
CTABMSEG	INT: Number of curve segments	STRING: Storage location: "SRAM", "DRAM"	STRING: Segment type: "L": Linear "P": Polynomial					Determines the maximum possible number of curve segments of the specified segment type in the specified memory >=0: Number -1: Table does not exist
CTABPOL	INT: Number of curve polynomials	STRING: Storage location: "SRAM", "DRAM"						Determines the number of used curve polynomials in the specified memory >=0: Number -1: Table does not exist

Curve table functions								
Identifier	**Return value**	**Parameter**						**Explanation**
		1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
CTABPOLID	INT: Number of curve polynomials	INT: Table number						Determines the number of curve polynomials used by the curve table >=0: Number -1: Table does not exist
CTABFPOL	INT: Number of curve polynomials	STRING: Storage location: "SRAM", "DRAM"						Determines the maximum possible number of curve polynomials in the specified memory: >=0: Number -1: Table does not exist
CTABMPOL	INT: Number of curve polynomials	STRING: Storage location: "SRAM", "DRAM"						Determines the maximum possible number of curve polynomials in the specified memory: >=0: Number -1: Table does not exist
CTABSSV	REAL: Following axis position	REAL: Leading axis position	INT: Table number	VAR REAL[ ]: Pitch result	AXIS: Following axis for scaling	AXIS: Leading axis for scaling		Determines the following axis position at the start of the curve segment belonging to the specified leading axis value
CTABSEV	REAL: Following axis position	REAL: Leading axis position	INT: Table number	VAR REAL[ ]: Pitch result	AXIS: Following axis for scaling	AXIS: Leading axis for scaling		Determines the following axis position at the end of the curve segment belonging to the specified leading axis value
CTABTSV	REAL: Following axis position	INT: Table number	VAR REAL[ ]: Pitch result at start of the table	AXIS: Following axis				Determines the following axis position at the start of the curve table.
CTABTEV	REAL: Following axis position	INT: Table number	VAR REAL[ ]: Pitch result at end of the table	AXIS: Following axis				Determines the following axis position at the end of the curve table.

Curve table functions								
Identifier	Return value	Parameter						Explanation
		1st	2nd	3rd	4th	5th	6th	
CTABTSP	REAL: Leading axis position	INT: Table number	VAR REAL[ ]: Pitch result at start of the table	AXIS: Leading axis				Determines the leading axis position at the start of the curve table.
CTABTEP	REAL: Leading axis position	INT: Table number	VAR REAL[ ]: Pitch result at end of the table	AXIS: Leading axis				Determines the leading axis position at the end of the curve table.
CTABTMIN	REAL: Minimum value	INT: Table number	REAL: Leading value interval lower limit	REAL: Leading value interval upper limit	AXIS: Following axis	AXIS: Leading axis		Determines the minimum value of the following axis in the entire definition range of the curve table or in a defined interval
CTABTMAX	REAL: Maximum value	INT: Table number	REAL: Leading value interval lower limit	REAL: Leading value interval upper limit	AXIS: Following axis	AXIS: Leading axis		Determines the maximum value of the following axis in the entire definition range of the curve table or in a defined interval
Note: The curve table functions can also be programmed in synchronized actions.								

Axis functions						
Identifier	Return value	Parameter				
		1st	2nd	3rd	4th	Explanation
AXNAME	AXIS: Axis identifier	STRING [ ]: Input string				Converts input string into axis identifier
AXSTRING	STRING[ ]: Axis name	AXIS: Axis identifier				Converts axis identifier into string
ISAXIS	BOOL: Axis present (TRUE) or not (FALSE)	INT: Number of the geometry axis (1 to 3)				Checks whether the geometry axes 1 to 3 specified as parameters are present in accordance with machine data MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB
SPI	AXIS: Axis identifier	INT: Spindle number				Converts spindle number into axis identifier

Axis functions						
Identifier	**Return value**	**Parameter**				
		1st	**2nd**	**3rd**	**4th**	**Explanation**
AXTOSPI	INT: Spindle number	AXIS: Axis identifier				Converts axis identifier into spindle number
MODAXVAL	REAL: modulo value	AXIS: Axis identifier	REAL: Axis position			From the entered axis position, calculates the modulo rest If the specified axis is not a modulo axis, the axis position is returned unchanged.
POSRANGE	BOOL: Position setpoint within the position window (TRUE) or not (FALSE)	AXIS: Axis identifier	REAL: Reference position in the coordinate system	REAL: Position window width	INT: Coordinate system	Determines whether the position setpoint of an axis is located in a window at a predefined reference position

Tool management					
Identifier	**Return value**	**Parameter**			**Explanation**
		1st	**2nd**	**3rd**	
CHKDM	INT: Status: Result of the check:	INT: Magazine number	INT: D number		Checks the uniqueness of the D number within a magazine
CHKDNO	INT: Status: Result of the check:	INT: T number of the 1st tool	INT: T number of the 2nd tool	INT: D number	Checks the uniqueness of the D number
GETACTT	INT: Status	INT: T number	STRING [32]: Tool name		Determines the active tool from a group of tools with the same name
GETACTTD	INT: Status: Result of the check:	VAR INT: T number found (return value)	INT: D number		Determines the T number associated with an absolute D number
GETDNO	INT: D number	INT: T number	INT: Cutting edge number		Determines the D number of the cutting edge of tool T
GETT	INT: T number	STRING [32]: Tool name	INT: Duplo number		Determines the T number for the tool name
NEWT	INT: T number	STRING [32]: Tool name	INT: Duplo number		Sets up a new tool (provides the tool data) The duplo number can be omitted.
TOOLENV	INT: Status	STRING: Name			Stores the tool environment with the specified name in the static NC memory

Tool management					
Identifier	**Return value**	**Parameter**			**Explanation**
		1st	**2nd**	**3rd**	
DELTOOLENV	INT: Status	STRING: Name			Deletes the tool environment with the specified name in the static NC memory Deletes all tool environments if no name is specified.
GETTENV	INT: Status	STRING: Name	VAR INT: T number [0] D number [1] DL number [2]		Determines the T number, D number, and DL number from a tool environment with the specified name

Arithmetic					
Identifier	**Return value**	**Parameter**			**Explanation**
		1st	**2nd**	**3rd**	
SIN	REAL	REAL			Sine
ASIN	REAL	REAL			Arc sine
COS	REAL	REAL			Cosine
ACOS	REAL	REAL			Arc cosine
TAN	REAL	REAL			Tangent
ATAN2	REAL	REAL	REAL		Arc tangent 2
SQRT	REAL	REAL			Square root
POT	REAL	REAL	REAL		Power function
TRUNC	REAL	REAL			Integer component
ROUND	REAL	REAL			Round to an integer number
ROUNDUP	REAL	REAL			Round up
ABS	REAL	REAL			Absolute value
LN	REAL	REAL			Natural logarithm
EXP	REAL	REAL			Exponential function e^x
MINVAL	REAL	REAL	REAL		Determines the smaller value of two parameters
MAXVAL	REAL	REAL	REAL		Determines the larger value of two parameters
BOUND	REAL: Check status	REAL: Lower limit	REAL: Upper limit	REAL: Reference value	Determines whether the reference value is within the limits.
Note: The arithmetic functions can also be programmed in synchronized actions. These arithmetic functions are calculated and evaluated in the main run. The synchronized action parameter $AC_PARAM[<n>] can also be used for calculations and as buffer.					

String functions					
Identifier	**Return value**	**Parameter**			**Explanation**
		1st	**2nd**	**3rd**	
ISNUMBER	BOOL	STRING: Input string			Checks whether the input string can be conver-ted to a number.
NUMBER	REAL	STRING: Input string			Converts the input string into a number.
TOUPPER	STRING	STRING: Input string			Converts the input string into upper case
TOLOWER	STRING	STRING: Input string			Converts the input string into lower case
STRLEN	INT	STRING: Input string			Determines the length of the input string up to the end of the string (/0)
INDEX	INT	STRING: Input string	CHAR: Search char-acters		Determines the position of the character in the input string from left to right. The 1st character of the string from the left has the index 0.
RINDEX	INT	STRING: Input string	CHAR: Search char-acters		Determines the position of the character in the input string from right to left. The 1st character of the string from the right has the index 0.
MINDEX	INT	STRING: Input string	STRING: Search char-acter		Determines the position of a character specified in the 2nd parameter in the input string from left to right. The 1st character of the input string from the left has the index 0.
SUBSTR	STRING	STRING: Input string	INT	INT	Determines the substring of the input string, defined by the start character (2nd parameter) and number of characters (3rd parameter).
SPRINT	STRING	STRING: Input string			Determines the formatted input string

Functions for measuring cycles								
Identifier	**Return value**	**Parameter**						**Explanation**
		1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
CALCPOSI	INT: Status	REAL[3]: Starting position in the WCS	REAL[3]: Incremental path specification in relation to the starting position	REAL[5]: Minimum distances to the monitoring limits	REAL[3]: Return array for the poss. incr. distance	BOOL: Conversion of the measuring system Yes/No	INT: Type of limit monitoring	Checks whether the geometry axes can traverse a defined path without violating the axis limits starting from a specified starting point. If the defined path cannot be traversed without violating limits, the maximum permissible value is returned.
GETTCOR	INT: Status	REAL [11]:	STRING: Tool length component: Coordinate system	STRING: Name of the tool environment	INT: Internal T no. of the tool	INT: Cutting-edge number (D no.) of the tool	INT: Number of the location-dependent offset (DL no. of the tool)	Determines the tool lengths and tool length components from tool environment or current environment
LENTOAX	INT: Status	INT[3]: Axis assignment of the geometry axes	REAL[3]: Matrix for mapping the tool lengths in the coordinate system	STRING: Coordinate system for the assignment				Determines information about the assignment of the tool lengths L1, L2, L3 of the active tool to abscissa, ordinate, applicate. The assignment to the geometry axes is affected by frames and the active plane (G17 - 19).

SETTCOR	INT: Status	**1st**	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**	
		REAL [3]: Offset vector in space	STR.: Component identifier	INT: Component(s) to be corrected 0 - 11	INT: Type of write operation 0 - 3	INT: Index of the geometry axis	STRING: Name of the tool environment	INT: int. T No. of the tool	INT: D no. of the tool	INT: DL no. of the tool	Changes the tool components, considering all supplementary conditions that are included in the evaluation of the individual components

Other functions								
Identifier	Return value	Parameter						Explanation
		1st	2nd	3rd	4th	5th	6th	
STRINGIS	INT: Information about the string	STRING: Name of the element to be checked						Checks whether the specified string is available as element of the NC programming language in the current language scope.
ISVAR	BOOL: Variable known Yes/No	STRING: Name of the variable						Checks whether the transfer parameter contains a variable known in the NC (machine data, setting data, system variable, general variables such as GUDs).
GETVARTYP	INT: Data type	STRING: Name of the variable						Determines the data type of a system/user variable
GETVARPHU	INT: Numeric value of the physical unit	STRING: Name of the variable						Determines the physical unit of a system/user variable
GETVARAP	INT: Protection level for access	STRING: Name of the variable	STRING: Type of access					Determines the access right to a system/user variable
GETVARLIM	INT: Status	STRING: Name of the variable	CHAR: Specifies which limit value should be read out	VAR REAL: Return of the limit value				Determines the lower/upper limit value of a system/user variable
GETVARDFT	INT: Status	STRING: Name of the variable	VAR REAL /STRING/ FRAME: Return of the default value	INT: Index to the first dimension (optional)	INT: Index to the second dimension (optional)	INT: Index to the third dimension (optional)		Determines the default value of a system/user variable
COLLPAIR	INT: Check result	STRING: Name of the 1st protection area	STRING: Name of the 2nd protection area	BOOL: Alarm suppression (optional)				Checks whether part of a collision pair

Other functions								
Identifier	Return value	Parameter						Explanation
		1st	2nd	3rd	4th	5th	6th	
PROTD	REAL: Clearance of the two protection zones	STRING: Name of the 1st protection area	STRING: Name of the 2nd protection area	VAR REAL: Return value: 3-dimensional clearance vector	BOOL: Measuring system for clearance and clearance vector (optional)			Determines the clearance of the two specified protection areas.
DELOBJ	INT: Error number	STRING: Component type to be deleted	INT: Start index of the components to be deleted (optional)	INT: End index of the components to be deleted (optional)	BOOL: Alarm suppression (optional)			Deletes elements from kinematic chains, protection areas, protection area elements, collision pairs and transformation data
NAMETOINT	INT: System variable index	STRING: Name of the system variable array	STRING: Character string / name	BOOL: Alarm suppression (optional)				Determines the associated system variable index based on the character string
ORISOLH	INT: Error number	INT: Controls the behavior of the function	REAL: First angle	REAL: Second angle				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. Requirement: A 6-axis transformation is active that has been parameterized with kinematic chains.
CORRTRAFO	INT: Error number	REAL: Correction vector	INT: Element to be modified	INT: Correction mode	BOOL: Alarm suppression (optional)			Modifies offset vectors or direction vectors of the orientation axes in the kinematic model of the machine.

Other functions								
Identifier	**Return value**	**Parameter**						**Explanation**
		1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	
CORRTC	INT: Error number	REAL: Correction vector	INT: Element to be modified	INT: Correction mode	BOOL: Alarm suppression (optional)			Modify offset vectors or direction vectors of orientable tool carriers according to machine measurement.
CALCTRAVAR	INT: Error number	VAR REAL: [0]: Correct position of the spindle for selection of transformation TRAINT [1]: Correct frame rotation for selecting the transformation TRAINT	STRING: Name of the TRAINT transformation data set to be activated later	INT: Alignment of the tool cutting edge to the circular tangent				Calculate angle for aligning the tool for TRAINT

5.8 Currently set language in the HMI

The table below lists all of the languages available at the user interface.

The currently set language can be queried in the part program and in the synchronized actions using the following system variable:

$AN_LANGUAGE_ON_HMI = <value>

<value>	Language	Language code
1	German (Germany)	GER
2	French	FRA
3	English (Great Britain)	ENG
4	Spanish	ESP
6	Italian	ITA
7	Dutch	NLD
8	Simplified Chinese	CHS
9	Swedish	SVE
18	Hungarian	HUN

<value>	Language	Language code
19	Finnish	FIN
28	Czech	CSY
50	Portuguese (Brazil)	PTB
53	Polish	PLK
55	Danish	DAN
57	Russian	RUS
68	Slovakian	SKY
72	Rumanian	ROM
80	Traditional Chinese	CHT
85	Korean	KOR
87	Japanese	JPN
89	Turkish	TRK

Note

$AN_LANGUAGE_ON_HMI is updated:

- after the system boots.
- after NC and/or PLC reset.
- after switching over to another NC within the scope of M2N.
- after changing over the language on the HMI.

Appendix

A

A.1 List of abbreviations

A	
A	Output
AFIS	Automatic Filter Switch: Automatic filter switch
ASCII	American Standard Code for Information Interchange: American coding standard for the exchange of information
ASIC	Application Specific Integrated Circuit: User switching circuit
ASUP	Asynchronous subprogram
AUTO	Operating mode "Automatic"
AUXFU	Auxiliary Function: Auxiliary function
AWL	Statement list

B	
BAG	Mode group
BCD	Binary Coded Decimals: Decimal numbers encoded in binary code
BICO	Binector Connector
BIN	Binary Files: Binary files
BKS	Basic coordinate system
BM	Operating alarm
BO	Binector Output
BTSS	Operator panel interface

C	
CLC	Clearance control
CNC	Computerized Numerical Control: Computerunterstützte numerische Steuerung
COM	Communication
CP	Communication Processor
CPU	Central Processing Unit: Central processing unit
CST	Configured Stop: Configured stop

D	
DDS	Drive Data Set: Drive data set
DIR	Directory: Directory
DO	Drive Object
DRF	Differential Resolver Function: Differential resolver function (handwheel)

D	
DRY	Dry Run: Dry run feedrate
DWORD	Double word (currently 32 bits)

E	
E	Input
EES	Execution from External Storage
E/A	Input/Output
ESR	Extended stop and retract
ETC	ETC key ">"; Softkey bar extension in the same menu

F	
FDD	Feed Disable: Feed disable
FdStop	Feed Stop: Feed stop
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
FM	Error message
FUP	Control system flowchart (PLC programming method)
FW	Firmware

G	
GEO	Geometry, e.g. geometry axis
GP	Basic program (PLC)
GUD	Global User Data: Global user data

H	
HEX	Abbreviation for hexadecimal number
HiFu	Auxiliary function
HMI	Human Machine Interface: SINUMERIK user interface
HSA	Main spindle drive
HT	Handheld Terminal
HW	Hardware

I	
IBN	Commissioning
INC	Increment: Increment
INI	Initializing Data: Initializing data
IPO	Interpolator

J	
JOG	JOGging: Setup mode

K	
KOP	Ladder diagram (PLC programming method)

L	
LED	Light Emitting Diode: Light-emitting diode
LMS	Position measuring system
LR	Position controller

M	
Main	Main program: Main program (OB1, PLC)
MCP	Machine Control Panel: Machine control panel
MD	Machine Data
MDA	Manual Data Automatic: Manual input
MDS	Motor Data Set: Motor data set
MELDW	Message word
MKS	Machine coordinate system
MM	Motor Module
MPF	Main Program File: Main program (NC)
MPI	Multi Point Interface
MSTT	Machine control panel

N	
NC	Numerical Control: Numerical control with block preparation, traversing range, etc.
NCU	Numerical Control Unit: NC hardware unit
NCK	Numerical Control Kernel
NCSD	NC Start Disable
NST	Interface signal
NV	Work offset
NX	Numerical Extension: Axis expansion module

O	
OB	Organization block in the PLC
OEM	Original Equipment Manufacturer
OP	Operation Panel: Operating equipment

P	
PCU	PC Unit: PC box (computer unit)
PG	Programming device
PI	Program instance
PLC	Programmable Logic Control: Controller
PN	PROFINET
PO	Power On
POS	Position/positioning
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
PTP	Point to Point: Point-to-point
PZD	Process data: Process data part of a PPO

R	
REF	REFerence point approach function
REPOS	REPOSition function
RESU	Retrace support
RID	Read In Disable
RP	R Parameter, arithmetic parameter, predefined user variable

S	
SA	Synchronized action
SBL	Single Block: Single block
SBT	Safe Brake Test
SCC	Safety Control Channel
SCL	Structured Control Language
SD	Settingdatum or setting data
SDI	Safe Direction
SERUPRO	Search-Run by Program Test: Block search via program test
SIC	Safety Info Channel
SKP	Skip: Function for skipping a part program block
SLP	Safe Limited Position
SLS	Safely Limited Speed
SMI	Sensor Module Integrated
SOS	Safe Operating Stop
SPF	Sub Program File: Subprogram (NC)
SS1	Safe Stop 1
SS2	Safe Stop 2
STO	Safe Torque Off

S	
STW	Control word
SUG	Grinding wheel peripheral speed
SW	Software

T	
TCU	Thin Client Unit
TIA	Totally Integrated Automation
TM	Terminal Module (SINAMICS)
TO	Tool Offset: Tool offset
TOA	Tool Offset Active: Identifier (file type) for tool offsets
TOFF	Online tool length offset
TRANSMIT	Transform Milling Into Turning: Coordination transformation for milling operations on a turning machine

U	
UP	Subprogram
USB	Universal Serial Bus

V	
VDI	Internal communication interface between NC and PLC

W	
WKS	Workpiece coordinate system
WPD	Work Piece Directory: Workpiece directory
WZ	Tool
WZV	Tool management

Z	
ZSW	Status word (of drive)

Index

-

$

B

C

D

E

F

G

H

I

J

K

L

M

P

Q

R

S

T

U

V

W

X

Y

Z