Industrial Hydraulics | **Electric Drives and Controls** | Linear Motion Assembly Technologies | Pneumatics | Service Automation | Mobile Hydraulics

Rexroth
Bosch Group

# Rexroth PNC
# DIN Programming Instructions

**1070 073 738**
Edition 11

Application description V7.3

**Title** Rexroth PNC
DIN Programming Instructions

**Type of Documentation** Application description

**Document Typecode** DOK-PNC***-DIN*PROG***-AW11-EN-P

**Purpose of Documentation** The present manual provides information on the operation, syntax and instruction set of the DIN programming language.

**Record of Revisions**

| Description | Release Date | Notes |
|---|---|---|
| DOK-PNC***-DIN*PROG***-AW11-EN-P | 06.2003 | Valid from V7.3 |
| | | |
| | | |
| | | |
| | | |

**Copyright** © Bosch Rexroth AG, 1993 – 2003

Copying this document, giving it to others and the use or communication of the contents thereof without express authority, are forbidden. Offenders are liable for the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design (DIN 34–1).

**Validity** The specified data is for product description purposes only and may not be deemed to be guaranteed unless expressly confirmed in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.

**Published by** Bosch Rexroth AG
Postfach 11 62
D-64701 Erbach
Berliner Straße 25
D-64711 Erbach
Tel.: +49 (0) 60 62/78-0
Fax: +49 (0) 60 62/78-4 28
Abt.: BRC/ESM11 (WE)

# Contents

See also the table of all G instructions and functions in the Annex.

# 1 Safety Instructions

Please read this manual before programming the PNC or modifying existing programs. Store this documentation in a place to which all users have access at any time.

## 1.1 Intended use

This manual contains all information required for the proper use of the control units. For reasons of clarity, however, it cannot contain each and every detail about each and all combinations of functions. Likewise, it is impossible to consider each and any aspect of integration or operation.

The PNC controls serve as

- activate feed drives, spindles and auxiliary axes of a machine tool via SERCOS interface for the purpose of guiding a processing tool along a programmed path to process a workpiece (CNC). Furthermore, I/O components are required for the integrated PLC which – in communication with the actual CNC – controls the machine processing cycles holistically and acts as a technical safety monitor.
- program contours and the processing technology (path feedrate, spindle speed, tool change) of a workpiece.

Any other application is deemed improper use!

The products described

- have been developed, manufactured, tested and documented in compliance with the safety standards. These products pose no danger to persons or property if they are used in accordance with the handling stipulations and safety notes prescribed for their configuration, mounting, and proper operation.
- comply with the requirements of
  - the EMC Directives (89/336/EEC, 93/68/EEC and 93/44/EEC)
  - the Low-Voltage Directive (73/23/EEC)
  - the harmonized standards EN 50081-2 and EN 50082-2
- are designed for operation in industrial environments, i.e.
  - no direct connection to public low-voltage power supply,
  - connection to the medium- or high-voltage system via a transformer.

  In residential environments, in trade and commerce as well as small enterprises class A equipment may only be used if the following warning is attached:

☞ **This is a Class A device. In a residential area, this device may cause radio interference. In such case, the user may be required to introduce suitable countermeasures, and to bear the cost of the same.**

The faultless, safe functioning of the product requires proper transport, storage, erection and installation as well as careful operation.

## 1.2 Qualified personnel

The requirements as to qualified personnel depend on the qualification profiles described by ZVEI (central association of the electrical industry) and VDMA (association of German machine and plant builders) in:
**Weiterbildung in der Automatisierungstechnik**
**edited by: ZVEI and VDMA**
**MaschinenbauVerlag**
**Postfach 71 08 64**
**D-60498 Frankfurt**.

The present manual is designed for NC programming personnel and NC project engineers.
These persons need special knowledge of the operation, syntax and instruction set of the DIN programming language.

Programming, start and operation as well as the modification of programs or program parameters may only be performed by properly trained personnel! This personnel must be able to judge potential hazards arising from programming, program changes and in general from the mechanical, electrical, or electronic equipment.
Interventions in the hardware and software of our products, unless described otherwise in this manual, are reserved to our specialized personnel.

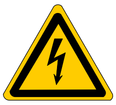
Tampering with the hardware or software, ignoring warning signs attached to the components, or non-compliance with the warning notes given in this manual may result in serious bodily injury or material damage.
Only electrotechnicians as recognized under IEV 826-09-01 (modified) who are familiar with the contents of this manual may install and service the products described.

Such personnel are

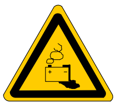
- those who, being well trained and experienced in their field and familiar with the relevant norms, are able to analyze the jobs being carried out and recognize any hazards which may have arisen.
- those who have acquired the same amount of expert knowledge through years of experience that would normally be acquired through formal technical training.

With regard to the foregoing, please note our comprehensive range of training courses. Please visit our website at
http://www.boschrexroth.com
for the latest information concerning training courses, teachware and training systems. Personal information is available from our Didactic Center Erbach,
Telephone: (+49) (0) 60 62 78-600.

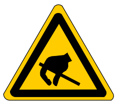
## 1.3 Safety markings on products



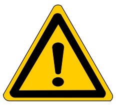
Warning of dangerous electrical voltage!



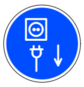
Warning of danger caused by batteries!



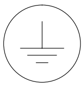
Components sensitive to electrostatic discharge!



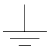
Warning of hazardous light emissions (optical fiber cable emissions)



Disconnect mains power before opening!



Lug for connecting PE conductor only!



Connection of shield conductor only

## 1.4 Safety instructions in this manual

**DANGEROUS ELECTRICAL VOLTAGE**
This symbol is used to warn of a **dangerous electrical voltage.** The failure to observe the instructions in this manual in whole or in part may result in **personal injury**.

**DANGER**
This symbol is used wherever insufficient or lacking compliance with instructions may result in **personal injury**.

**CAUTION**
This symbol is used wherever insufficient or lacking compliance with instructions may result in **damage to equipment or data files**.

☞ This symbol is used to draw the user’s attention to special circumstances.

★ This symbol is used if user activities are required.

## 1.5 Safety instructions for the described product

**DANGER**
**Danger of life through inadequate EMERGENCY-STOP devices!**
**EMERGENCY-STOP devices must be active and within reach in all system modes. Releasing an EMERGENCY-STOP device must not result in an uncontrolled restart of the system!**
**First check the EMERGENCY-STOP circuit, then switch the system on!**

**DANGER**
**Incorrect or undesired axis movement!**
**First, new programs should be tested carefully without axis movement! For this purpose, the control offers the possibility of inhibiting axis movements and/or auxiliary function outputs by appropriate softkeys in the 'Automatic' mode.**

**DANGER**
**Incorrect or undesired control unit response!**
**Rexroth accepts no liability for damage resulting from the execution of an NC program, an individual NC block or the manual movement of axes!**

**Furthermore, Rexroth accepts no liability for consequential damage which could have been avoided by programming the PLC appropriately!**

**DANGER**
**Retrofits or modifications may adversely affect the safety of the products described!**
**The consequences may include severe injury, damage to equipment, or environmental hazards. Possible retrofits or modifications to the system using third-party equipment therefore have to be approved by Rexroth.**

**DANGEROUS ELECTRICAL VOLTAGE**
**Unless described otherwise, maintenance works must be performed on inactive systems! The system must be protected against unauthorized or accidental reclosing.**

**Measuring or test activities on the live system are reserved to qualified electrical personnel!**

**DANGER**
**Tool or axis movements!**
**Feed and spindle motors generate very powerful mechanical forces and can accelerate very quickly due to their high dynamics.**

- **Always stay outside the danger area of an active machine tool!**
- **Never deactivate safety-relevant functions!**
- **Report any malfunction of the unit to your servicing and repairs department immediately!**

**CAUTION**
**Only spare parts approved by Rexroth may be used!**

**CAUTION**
**Danger to the module!**
**All ESD protection measures must be observed when using the module! Prevent electrostatic discharges!**

The following protective measures must be observed for modules and components sensitive to electrostatic discharge (ESD)!

- Personnel responsible for storage, transport, and handling must have training in ESD protection.
- ESD-sensitive components must be stored and transported in the prescribed protective packaging.
- ESD-sensitive components may only be handled at special ESD-workplaces.
- Personnel, working surfaces, as well as all equipment and tools which may come into contact with ESD-sensitive components must have the same potential (e.g. by grounding).
- Wear an approved grounding bracelet. The grounding bracelet must be connected with the working surface through a cable with an integrated 1 MΩ resistor.
- ESD-sensitive components may by no means come into contact with chargeable objects, including most plastic materials.
- When ESD-sensitive components are installed in or removed from equipment, the equipment must be de-energized.

## 1.6 Documentation, software release and trademarks

### Documentation

The present manual provides information on the operation, syntax and instruction set of the DIN programming language.

| Overview of available documentation | Part no. | | |
|---|---|---|---|
| | German | English | French |
| PNC-R – Connectivity Manual for project engineering and maintenance | 1070 073 704 | 1070 073 736 | – |
| PNC-R – Software installation | 1070 073 796 | 1070 073 797 | – |
| PNC-P – Connectivity Manual | 1070 073 880 | 1070 073 881 | – |
| PNC-P – BF2xxT/BF3xxT Control Panel Connectivity Manual | 1070 073 814 | 1070 073 824 | – |
| PNC-P – Software installation | 1070 073 882 | 1070 073 883 | – |
| Description of functions | 1070 073 870 | 1070 073 871 | – |
| MACODA<br>Operation and configuration of the machine parameters | 1070 073 705 | 1070 073 742 | – |
| Operating instructions – Standard operator interface | 1070 073 726 | 1070 073 739 | 1070 073 876 |
| Operating instructions – Diagnostics Tools | 1070 073 779 | 1070 073 780 | – |
| Error Messages | 1070 073 798 | 1070 073 799 | – |
| PLC project planning manual,<br>Software interfaces of the integrated PLC | 1070 073 728 | 1070 073 741 | – |
| iPCL system description and programming manual | 1070 073 874 | 1070 073 875 | – |
| ICL700 system description (PNC-R only),<br>Program structure of the integrated PLC ICL700 | 1070 073 706 | 1070 073 737 | – |
| DIN programming manual<br>for programming to DIN 66025 | 1070 073 725 | 1070 073 738 | – |
| CPL programming manual | 1070 073 727 | 1070 073 740 | 1070 073 877 |
| CPL Debugger Operating Instructions | 1070 073 872 | – | – |
| Tool Management – Parameterization | 1070 073 782 | 1070 073 793 | – |
| Software PLC<br>Development environment for Windows NT | 1070 073 783 | 1070 073 792 | – |
| Measuring cycles for<br>touch-trigger switching probes | 1070 073 788 | 1070 073 789 | – |
| Universal Milling Cycles | – | 1070 073 795 | – |

☞ **In this manual the floppy disk drive always uses drive letter A:, and the hard disk drive always uses drive letter C:.**

Special keys or key combinations are shown enclosed in pointed brackets:

- Named keys: e.g., <Enter>, <PgUp>, <Del>
- Key combinations (pressed simultaneously): e.g., <Ctrl> + <PgUp>

## Release

☞ **This manual refers to the following version:**
**Software release: V7.3**

The current release number of the individual software modules can be viewed by selecting the 'Control-Diagnostics' softkey in the 'Diagnostics' operating mode.

The software version of Windows may be displayed as follows:

1. Click the right mouse button on the **My Computer** icon on your desktop.
2. Select **Properties**.

## Trademarks

All trademarks of software installed on Rexroth products upon delivery are the property of the respective manufacturer.

Upon delivery, all installed software is copyright-protected. The software may only be reproduced with the approval of Rexroth or in accordance with the license agreement of the respective manufacturer.

MS-DOS® and Windows™ are registered trademarks of Microsoft Corporation.

PROFIBUS® is a registered trademark of the PROFIBUS Nutzerorganisation e.V. (user organization).

SERCOS interface™ is a registered trademark of Interessengemeinschaft SERCOS interface e.V.

# 2 Basics

## 2.1 Function and structure of an NC program

The NC program serves to provide the control unit with all information required for machining on the machine.

The structure of an NC program is variable. Only the guidelines are summarized in DIN 66025*). In this publication you can find the rules according to which the programming blocks are to be formed in the NC program.

☞ **The contents of DIN 66025, 'Program structure for numerically controlled machines' (Parts 1 and 2) correspond to the ISO/DIS 6983 and ISO/DP 6983 international standards, 'Numerical control of machines'.**

The PNC offers two different ways of programming:

- DIN 66025 programming
- CPL programming

In the present manual, you will find a description of the programming method according to **DIN 66025**. Machine-dependent cycles (machine manufacturer cycles) will not be described in this manual.

All NC programs (part programs) are maintained in the "File System" of the PNC. For the structure as well as for detailed explanations of the **File System** and the **File Protection** (**access rights**), please refer to the "Directories" section of the PNC operating instructions.
The operating instructions will also give you information about the new creation and editing of part programs.

### 2.1.1 Programming principles

Workpiece contours are divided into straight lines and circular arcs. The control unit is able to execute the respective movements required for each of these geometrically 'simple' contour elements in one machining step – a program block. As a prerequisite, all the machining steps must be determined in the correct sequence and with all necessary boundary conditions within the NC program.

The NC program consists of individual program blocks. These contain preparatory functions, positional data, auxiliary and special functions.
These blocks are used to enter details about the position, the technology and the program flow.

☞ **The memory available (e.g. for NC programs) depends on the control memory option.**

Basics

**Example:** Procedure for machining

- Breaking down the machining process into logical (and possibly recurrent) sections
- Breaking down the contour into ”simple” consecutive contour elements.
- Creation of the program (incl. subprograms, if any), program input into the CNC.
- Program start.

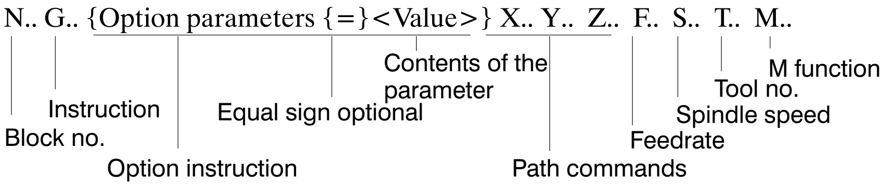
CNC controls the machining of the workpiece.

Program blocks

The control unit executes the program blocks one by one.

Each program block consists of a set of **program words** which, in turn, are made up of an **address letter** and a **string of digits**.

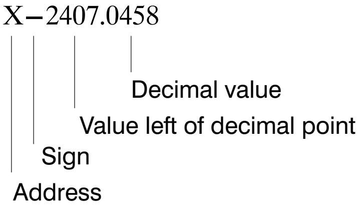
**Example:** A program block comprising 10 program words



Program words

Each program word of a block may consist of an address letter and a number (e.g. G00, X–23.450,Y40, M03, S250).

**Example:** Program word



- Leading zeros do not have to be programmed
- Non-integers are written with a decimal point; trailing zeros may be omitted (e.g. ”X100.500” corresponds to ”X100.5”).
- The CNC processes variable block lengths. The number of words per block may vary.
- Words containing positional data determine the tool path. These may also include a sign (+/–). If no sign is programmed, the positive value is assumed. In the case of a negative value, you must program a minus sign.

Some G functions include **optional** program words. These are set between **braces**, which are to be omitted for programming.

**Example:**
Syntax rule: G631 {SYM<s>} {ANG<a>}
Programming: G631 SYM2 ANG10

Setting the SYM<s> parameter as shown above between braces {..} is optional. If this parameter is omitted in the programmed instruction, the SYM parameter is automatically assigned a value(s) preset in MACODA.

**Example:**
Syntax rule: G612 <Axis name $_{\text{i}}$><Time$_{\text{Axis name i}}$>
Programming: G612 X10

“Axis name $_{\text{i}}$ ” denotes the i-th physical axis, e.g.
X axis = 1st physical axis
Y axis = 2nd physical axis

“Time$_{\text{axis name i}}$“ is 10 ms and refers only to the X axis.

Program words put between **square** brackets represent various parameters of one and the same category or are to be assigned specific values. If required, these program words must be set correspondingly when programming.

**Example:** Parameters for modal subprograms
Syntax rule: G81 [<Parameter 1>,<Parameter 2>,
{<Parameter 3>},{<Parameter 4>}]

Programming: G81 [Z,R1,P,R2]

Modal effect

Most words have a modal effect. This means that their effect remains in force until you program the same word with a different value, or until you switch off the word’s function.

**Example:**

As soon as you have programmed G1 (linear interpolation at feedrate) in a program block, the control unit will approach all subsequently specified positions at feedrate without the necessity of programming G1 again. G1 remains effective until you program a different interpolation type (e.g. ”G2”: circular interpolation or ”G0”: linear interpolation at rapid feedrate).

Nonmodal effect

Words of this type are only effective within the block in which they were programmed.

## Instructions and special functions

In terms of their effect, program words act either as instructions or special functions.

Instructions

For instance, the CNC must be told in which manner and to which position a tool is supposed to travel. This positional data is communicated to the CNC via the G address (*manner* of travelling) and the X, Y, Z, C addresses etc. (*where* to travel).

### Addresses X, Y, Z, C etc.

You use these addresses to determine the axis that is to travel to a specific position or over a specific distance.

☞ **With asynchronous axes (auxiliary axes), the input of positional/ path information alone will always initiate a motion. Normally, asynchronous axes are traversed in rapid mode. The speed can only be influenced by programming an FA address (refer to section 5.2).**

With various G functions, the value input is not interpreted as positional/ path information, but rather a parameter for the function, e.g.:

`N10 G60 X10 Y10 Z50` Specification of a new program zero point, does not cause any axis traversing.

The axis address (axis name) is specified by MACODA parameter 1003 00001. Axis addresses may also end in a numeral (e.g. ”X1”, ”X2”, ”B1”, ”PALLET1” etc.). If this is the case, the ”=” sign or a blank must be programmed between the axis address and any subsequent positional/ path information!

`G1 X1=90` or `G1 X1 90`

☞ **If a longer axis designation starts with another shorter one (there are axes “X” and “X2”), and a decimal point is programmed subsequently, the longer designation shall always apply (X2.5 → axis X2 travels to 0.5).**

### G address

G addresses are used to program the type of traversing movement (e.g. rapid feed, linear or circular interpolation etc.); this is also the reason why reference to ’preparatory functions’ is made.

All preparatory functions are ’sorted’ in **groups**. Preparatory functions from different groups do not interfere with each other. However, as preparatory functions contained within one and the same group act modally, not more than one G instruction **from any one group** may be used per program block.

From section 3on, you will find a list of all preparatory functions which are recognized by the CNC. It also describes their respective groups.

**Examples**:

```
N...G1 X20 Y50
```

Linear interpolation to a position

```
N...G60 X10 Y10 B1 35
```

Traversing movement of auxiliary axis

Zero shift

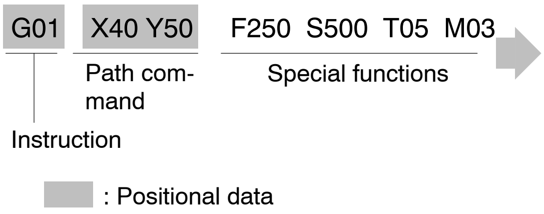
Special functions

Within the NC program, instructions may be supplemented by **special functions**. Some examples of important address letters representing special functions:

**F** Feed rate:
**S** Spindle speed
**M** M functions
(e.g. gear-stage selection, direction of spindle rotation)
**T** T word (tool selection)

For details, please refer to section 5, ”Auxiliary and special functions”.

**Example:** Positional data with special functions



Move at feedrate to X40, Y50 with programmed F value (feedrate) and S value (speed) with spindle rotating clockwise and prepare tool T05 in tool magazine.

## 2.1.2 Program design elements

Block numbers

You may identify each NC block by a block number. This improves the program’s readability. DIN block numbers are always on the left at the beginning of a program line and consist of the ”N” address letter and a number following directly behind (example: ”N10 ...”).

You should program the block numbers in ascending order and with an increment width of 10 (N10 ...; N20 ...; N30 ... etc.). This way, you can insert additional program lines between two blocks in the case of program changes without impairing the readability of the program.

If you wish to use branching instructions or jump markers containing block numbers as parameters, you must identify the target blocks with block numbers. Likewise, block numbers are required in subprograms and cycles.

Comments

Comments are used to provide program parts with explanations or to document them. Well-commented programs facilitate and accelerate subsequent familiarization for other programmers, e.g. if the program needs to be changed. However, each comment character will increase the size of the program file by 1 byte.

Comment text is usually put between parentheses, e.g. ”( )” or preceded by a semicolon “;” set before the comment text. The PNC will ignore text between the parentheses.

**Example:** Comment text

```
N50 (Pocket machining)
or
N50 ; Pocket machining
```

Notes

If you program notes, they are used to display text on the CNC screen during the execution of the program. You can use such notes to inform the operating staff about the current status of the program, or to give them instructions for action.

There are two kinds of notes:

- Channel-specific notes — **Syntax variants:** (MSG ...), (*MSG ...), (MSG, ...), (*MSG, ...), MSG (...)
  These notes are displayed in the MSG window of the “Automatic” mode for the **calling** channel. Additionally, these notes are displayed under “Messages” in the info dialog. They are deleted when the program is cancelled or by control reset.
- Channel-independent notes — **Syntax variants:** (GMSG ...), (GMSG, ...)
  These notes are displayed under the channel-independent notes in the info dialog. They are deleted by system control reset.

A programmed note may have up to 80 characters.

For an instruction to take action, you would program, e.g., an ”M0” in the very same line or in the following one. This ensures that the execution of the program will only be resumed after the note has been acknowledged by ”Cycle start” .

**Example:** Note text

```
N60  (MSG Measure workpiece!)
N70 M0
```

Program run

In the absence of instructions relating to the program flow, the program blocks will be processed one by one. However, you have the following options of influencing the running of the program:

- Subprogram calls (please refer to sections 2.1.3 and 5.4.1 )
- Repeat instructions (please refer to the CPL manual)
- Jump instructions (please refer to the CPL manual)
- ’Skip block’ instruction

Skip block

You can mark program blocks in such a manner that the control unit will simply ignore these blocks if the input signal ”I3.4 Skip block” is active. To do so, you program the ”/” sign at the beginning of the program line.

**Example:** `/N30`

| | |
|---|---|
| I3.4 is active: | N30 block will be ignored |
| I3.4 is not active: | N30 block will be processed. |

Channel designation

A program may contain a channel designation.
Syntax: $<Channel number>

If a program containing a channel designation is started on another channel, a runtime error will occur.

**Example:**

| | |
|---|---|
| `N10 S2` | The following program can be executed on channel 2 only. |
| `N20 G.. X.. Y..` | Program instructions on channel 2 |
| `N30 S1` | The following program can be executed on channel 1 only. |
| `N40 G.. X.. Y..` | Program instructions on channel 1 |
| `...` | |

## 2.1.3 Subprograms

If you need to repeat a specific processing operation within a program, it is recommended that you write this program part in the form of a subprogram and call it whenever it is needed.
This will save programming code and memory space. In addition, your programs will become clearer and easier to maintain.

### Subprogram call with P address

You call subprograms via the P address in the form of ”`P<SP name> DIN`”.

Explanation:

| | |
|---|---|
| `<SP name>` | stands for the subprogram name. |
| `DIN` | Optional parameter. Prevents the subprogram from being linked. You should not use this parameter unless the subprogram consists of DIN blocks only and does not call any other subprograms. If this is not the case (e.g., if CPL blocks are used in the subprogram), a program runtime error message will be displayed. For further information, please refer to the ”CALL command” of the ”CPL programming instructions”. |

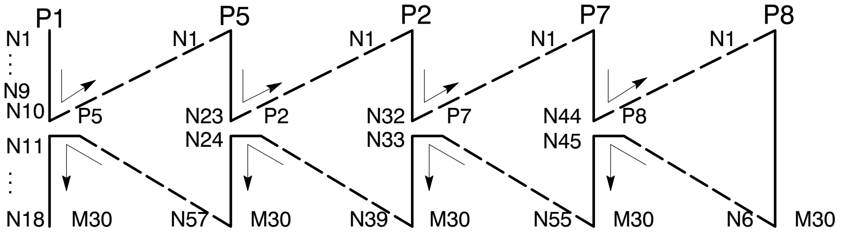
Traversing movements which are programmed in the same  line will be executed prior to the subprogram call.
(e.g. ”`N40 PTest1 X10 Y10 Z0`”).

The subprogram is executed unconditionally.
A subprogram can call further subprograms (nesting).

**Example:** Subprogram call

| | |
|---|---|
| `N...` | |
| `N40 PBohrbild1` | ”Bohrbild1” (drilling graph 1) is called and executed once. |
| `N50...` | Subsequently, the calling program will be further executed by the N50 block. |
| `N...` | |

**Example:** Nesting of subprograms



| | |
|---|---|
| P1: | Main program |
| P5, P2, P7, P8: | Subprograms |

☞ **Nesting is possible to a depth of 9 (incl. main program), i.e. with complete nesting, the main program can open a maximum of 8 subprograms.**

☞ **Subprograms can also be called via**
**G addresses (refer to section 2.1.4, page 2–11), and**
**M addresses (refer to section 5.4.1, page 5–5).**

### Subprogram call without P address

Subprograms can also be called directly **without** a P address. In this case, it is sufficient to state the name of the subprogram.

☞ **Any confusion with normal syntax must be avoided! Always give your subprograms unmistakable names to avoid misinterpretations by the control interpreter.**

Explanation:

<Sp name> stands for the subprogram name.

In both cases, the syntax is the same:

| | |
|---|---|
| N40 XSp | Subprogram call **without** P address |
| and | |
| N40 PXSp | Subprogram call **with** P address |

**Example:** Subprogram calls

| | |
|---|---|
| `N...` | |
| `N40 XSp` | Subprogram ”XSp” is called and executed. |
| `N50...` | Subsequently, the calling program will be further executed by the N50 block. |
| `N60 X0Sp` | **–>results in a syntax error!**<br>X0 is interpreted as the ’0’ coordinate in the X axis to which the X axis is to traverse, i.e. a subprogram called “X0Sp” will not even be recognized by the interpreter. |
| `N...` | |

## Subprogram call via nonmodal G instructions

Besides using various M functions (refer to page 5–5) or the P address (refer to page 2–8), subprograms can also be called via **16 nonmodal G instructions**.

You can use the MACODA to determine both the G instructions and the programs to be called via these G instructions. The subprogram called will be executed once.

☞ **The allocation of the G instruction to the program name is application-specific and can be defined in MACODA parameters 3090 00001 and 3090 00002.**
**Please contact your systems administrator for the G instructions defined as subprogram calls for your specific machine.**

Programming

As a rule, only **one** SP call may be programmed in a block with P, G or M functions. In the event of several identical address letters in one block (e.g. G or M), the address calling the subprogram must be programmed at the end of the line.

**Example:** Subprogram call via Gxx

```
N...   G0 X20 Y30 Z50 Gxx...
```

☞ **In addition to these 16 - non-modal - G instructions, an additional 16 modal G instructions may be defined as subprogram calls in MACODA parameters 3090 00005, 3090 00006, and 3090 00007.**
**These subprograms are processed by the PNC in each program block, until the modal effect is explicitly canceled by an appropriate command. This effect may be interesting, e.g., for drilling cycles. For example, you only need to travel to a new drilling position. After traversing to the new position, the hole is then drilled automatically by the subprogram.**

## End of subprogram

The end of a subprogram is reached

- at the file end.
  The NC returns to the calling program. All modal statuses are retained.
- in a program line containing ”M2”, ”M02”, or ”M30”.
  For details, refer to section 5.4.2.

## 2.1.4 Jump destinations and jump instructions

As a rule, main program and subprogram blocks and cycles are executed in the same order as they were programmed.

The processing sequence can be changed by program jumps.

The following instructions are available:

| | |
|---|---|
| ● Jump destinations (LABELS) | Stating jump destinations with user-defined names. |
| ● Jump destination (G23, G24) | Jump destinations dependent on an interface signal with a block number stated. |
| ● Jump instructions (GOTOF and GOTOB) | Allow branching from any point in the program to a jump destination. Program execution continues immediately upon arrival at the jump destination. |

Jump destination

Destination labelling (LABEL) within a program:

User-defined branches in a program can be programmed by defining jump destinations.

- Label names with a minimum of 2 and a maximum of 32 characters (letters, digits, underline) are assigned.
- The first two characters must be letters or underlines.
- The label name must always be followed by a colon.
- Labels are always written at the beginning of an NC block, directly after the block number.
- Jump destinations are addressed by means of jump instructions (GOTOF and GOTOB).

Jump instruction

Jump instruction (GOTOF) with a **forward** jump destination (towards the program end):

- must be programmed in a separate block.
- is programmed in combination with a LABEL.

Jump instruction

Jump instruction (GOTOB) with a **backward** jump destination (towards the beginning of the program):

- must be programmed in a separate block.
- is programmed in combination with a LABEL.

**Examples: Label, GOTOF, GOTOB**

| | |
|---|---|
| `N100 GOTOF TO_PART2` | Jump forward to jump destination |
| `N110..` | ”TO PART2” |
| `N120..` | |
| `N130 TO_PART1:` | Definition of jump destination ”TO PART1” |
| `N140..` | |
| `N150 GOTOB TO_PART1` | Jump backward to jump destination ”TO PART1” |
| `N160..` | |
| `N170 TO_PART2:` | Definition of the jump destination ”TO PART2” |

### Unconditional jump

The jump destination (G24) is a block number and is executed unconditionally. The jump destination is defined as an L address with a block number.

☞ **If an ”unconditional jump” is programmed incorrectly, an endless loop may occur.**

G24 L<block number>
with
<block number> = 15 digits, with an optional ”.”.

Please note for G24 L...:

- Jumps must never be programmed together with any other instructions in the same block.
- The syntax in the statement of the L address must be identical with the jump destination (N word) (also in the case of preceding zeros).

  **Example:**

  | | |
  |---|---|
  | `N020 G1 X200 Y300 F500` | |
  | `...` | |
  | `N500 G24 L20` | **Wrong!** |
  | `N500 G24 L020` | Correct! |

- Only DIN blocks can be jumped to. Blocks written in CPL may not be used as an L address.

### Conditional jump

The jump destination (G23) is conditioned by the status of the interface signal ”CONDITIONAL JUMP”. The interface signal is scanned while the G23 block is being prepared

☞ **Any interface signals between block preparation mode and block execution mode will be ignored!**
**Unless this can be ensured, block preparation must be interrupted by programming a WAIT instruction.**

The jump destination is defined as an L address with a block number.

G23 L<block number>
with
<block number> = 15 digits, with an optional ”.”.

Please note for G23 L...:

- Jumps must never be programmed together with any other instructions in the same block.
- The syntax in the statement of the L address must be identical with the jump destination (N word) (also in the case of preceding zeros).
- Only DIN blocks can be jumped to. Blocks written in CPL may not be used as an L address.

**Example:**

```
N68 X-250 Y20          Jump destination
...
N100 X100 Y200 Z50
N101 X0 Y0 Z10
102 WAIT               Waiting for an IF (IF = interface) signal, block
                       preparation interrupted.
N103 G23 L68           Jump to N68 is executed if an IF condition is
                       fulfilled.
N104 X200 Y-300
...
```

☞ **In CPL block 102, the programmed WAIT instruction ensures that any signal changes are recognized by the NC immediately before N103 processing.**

## 2.1.5 End of program

The **end of a (sub)program** is reached

- at the file end, or
- in a program line containing ”M2”, ”M02”, or ”M30”.
  For details concerning these M functions, refer to section 5.4.2.

If none of these M functions was used in the program, the control unit will interpret the end of file as the end of program.

At the end of a subprogram, execution returns to the calling program. All modal statuses are retained.

At the end of a main program, the system returns to the top of the program and waits for the next ”Cycle start”.  Again, all modal statuses are retained.

## 2.1.6 Standard programming formats

The standard format applies to metric input in terms of ”mm” and a measuring-system resolution of 0.0001 mm.

| Addresses | Preparatory function | Format | Meaning | Unit |
|---|---|---|---|---|
| variable, e.g. | | | **Positional data:** | |
| X,Y,Z,C | e.g.: G1, G2 | real | cartesian axis position | mm or degrees |
| X = AC(50) | e.g.: AC(..) | real | positional data with assignment | mm or degrees |
| X(p1,p2,p3, p4) | e.g.: G581 | real | positional data with parameter list | mm or degrees |
| I,J,K | G2, G3 | real | Circle parameters | mm |
| R | | | Circle radius | mm |
| | | | **Technological information** | |
| D | G41/G42 | int | cutter-radius compensation | compens. no. |
| F | G94 | real | feedrate (sync. axis) | mm/min |
| F | G4 | real | dwell time ( ” ) | sec |
| FA | | real | feedrate (auxil. axis) | mm/min |
| H | | int | tool length comp. | compens. no. |
| S | | real | spindle speed | rpm |
| T | | real | tool | tool no. |
| N (block no.) | | int | N1, N2, N3 etc. block address | |
| P,K,V | | str | Program, compensation, zero-shift address | |
| G | | int | G function | |
| M | | int | Special machine function | |

Meaning of the values in the ”Format” column:

| | |
|---|---|
| int: | Numerical string consisting of 9 digits max., without decimal point |
| real: | Numerical string. consisting of 15 digits max., with decimal point |
| str: | Character string |

☞ **Auxiliary functions (e.g. F, FA, S, ...etc.) can be bit- or bcd-coded (refer to Section 5).**

Notes:

# 3 G Instructions

☞ **For a tabular overview, please refer to the Annex.**

## 3.1 Linear interpolation at rapid travel G00

Effect

The programmed position is interpolated and approached at max. possible speed **on a straight line**.

At least one axis travels at max. speed or acceleration. The speed of the other axes is controlled in such a manner that they reach the target point at the same time.

- The speed can be influenced using the potentiometer.
- With active G0, the G0 ACTIVE channel IF (= interface) signal is output.
- With active G0, the system decelerates to V=0 **after each block**.

Use G161/G162 to determine whether G0 is to be active with or without ”In-position logic”.
If decelerating to V=0 after each block is not desired, you must use the G200 function instead of G0.

Programming

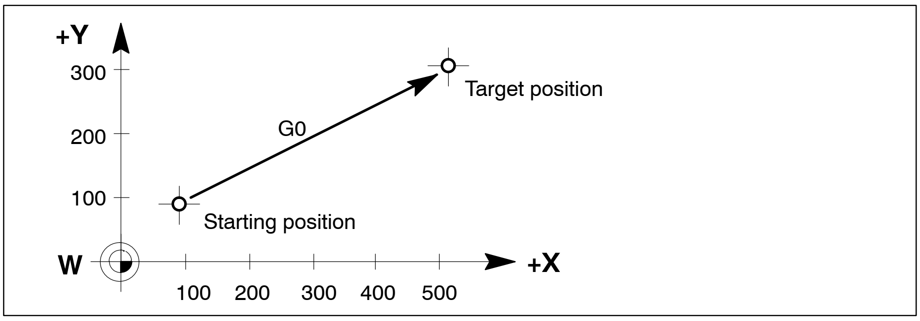
| | |
|---|---|
| `G0:` | Linear interpolation at rapid travel ON |

**Please note for G0:**

- Programmable with or without axis addresses.
- No feedrate value has to be programmed. The max. axis speed (1005 00002) is defined in MACODA.
- The feedrate of G0 rapid travel can be limited to the value set in MACODA parameter 7030 00110 by means of the channel-related interface signal ”Limit Rapid Travel” (NC I1.7).
- Acts modally until a new movement type is selected.
- G0 deletes G1, G2, G3, G5, G10–G13, G73, G200.

**Example:** rapid-travel programming

| | |
|---|---|
| `X100 Y100` | Starting position |
| `G0 X500 Y300` | Programmed target position |



## 3.2 Linear interpolation at rapid feed without decelerating to V=0 G200

Effect

If G0 is programmed, the system will – irrespective of G161/G162 – decelerate to V=0 at the block end. If this is not desired, use G200 instead.

This means that interpolation can continue without deceleration beyond the block limits. However, the following preconditions apply:

- G61 is not active and
- G163 is not active.

If G61 is actually active, the control unit will, despite G200, decelerate to V=0 after each block.

If G163 is active, the behavior depends on the respectively set ”In-position logic mode” (please refer to G164 to G166).

The effect of G200 corresponds to ”G1Fmax”.

Programming

`G200:` Linear interpolation at rapid feed without decelerating to V=0 ON

**Please note for G200:**

- Programmable with or without axis addresses.
- No feedrate value has to be programmed. The max. axis speed (1005 00002) is defined in MACODA.
- The feedrate of G200 rapid travel can be limited to the value set in MACODA parameter 7030 00110 by means of the channel-related interface signal ”Limit Rapid Travel” (NC I1.7).
- Acts modally until a new movement type is selected.
- G200 deletes the types of movement G0, G1, G2, G3, G5, G10–G13, G73.

## 3.3 Linear interpolation at feedrate G01

Effect

The programmed point is approached on **a straight line** at the effective feedrate (F word).

The movement is coordinated in such a manner that all axes involved arrive at the programmed end point simultaneously. At the end of the traversing movement, the control unit will perform a complete downslope down to speed V=0, unless a G8 is active.

The programmed feedrate value (F) acts as path feed; this means in the case of more than one moving axis that the portion of each individual axis is smaller than F.

The feedrate can be limited by MACODA parameters (as regards the axis or path).

The speed can be influenced using the feedrate potentiometer.

Use G61/G62 to determine whether G1 is to be active with or without ”In-position logic”.

Programming

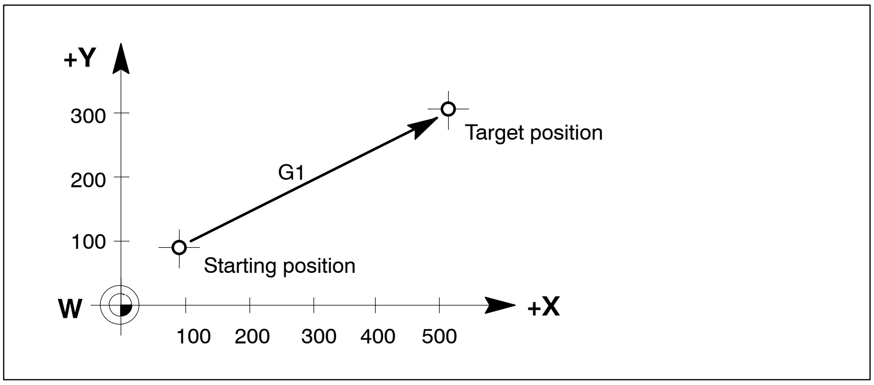
`G1:` Linear interpolation at feedrate ON

**Please note for G1:**

- G1 may be programmed with or without positional data.
- G1 has to be programmed with F word if no feedrate is yet active.
- The programmed feedrate remains effective until overwritten by a new one.
- G1 deletes G0, G2, G3, G5, G10–G13, G73 and G200.

**Example:** Linear programming

| | |
|---|---|
| X100 Y100 | Starting position |
| `G0 X500 Y300 F100` | Programmed target position |



## 3.4 Circular interpolation / helical interpolation G02 G03

Effect

The programmed end point is approached on a circular path at the active feedrate (F word).

The determination whether G2/G3 is to be active with or without ”In-position logic” is made using G61/G62.

The movement is coordinated in such a manner that all axes involved arrive at the programmed end point simultaneously. This applies also if an axis outside the circular plane is programmed within the block. In this case the PNC will interpolate this axis linearly together with the other axes. A helical movement will result (helical interpolation).

G2, G3 acts modally and deletes the G functions of the same group or is deleted by them.

The machine traverses at the programmed feedrate and in a circular motion in the selected plane:

- G2 clockwise
- G3 counter-clockwise.

A feedrate value has to be active. By G20, ”Plane selection 2 out of 6 axes”, it is possible to execute circles with two freely definable synchronous axes.

For programming, you can choose between

- radius programming and
- center-point programming.

☞ **Another option is G05 (circular interpolation with tangential entry).**

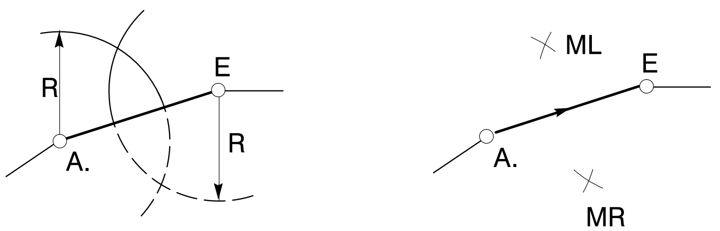
Depending on the type of programming, various parameters must be programmed in the G02/G03 block. Please refer to the following sections.

### 3.4.1 Radius programming

Using the **current position** as starting point, you define a circular movement with the **programmed radius** leading through the **programmed end point**.

The end point may be programmed in absolute or incremental terms. The radius will always act as an incremental value.

On the basis of the starting point, end point and radius, the PNC first calculates the circle center point. This results in two intersections, located to the left and to the right of the starting/end point distance:



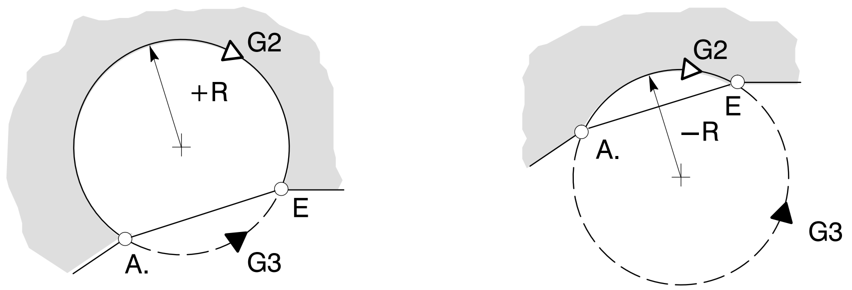
**A = Starting point**
**E = End point**
**R = Radius**

**ML** = center point left
**MR** = center point right

The applicable center point of the two is determined by the sign of the radius value:

- **Positive** radius value: center point **left**
- **Negative** radius value: center point **right**

The direction of rotation of the arc has already been specified by G2 or G3.



As the diagrams show, the radius must be at least half as large as the distance between the starting and end point, since otherwise no intersection point can be created.

If the radius is exactly half as large as the starting to end point path, this special case results in only one intersection point. This is only possible with a semi-circle. The sign of the radius value is then freely definable.

☞ **Radius programming cannot be used to create full circles. The smallest possible arc depends on the set MACODA parameters of the control unit (approx. 10 increments IN POS range).**

**Example:**

Programming

N... G17 G3 X... Y... R±... F... S ... M ...

where:

G17: Selection of the circular path in the X/Y plane
G3: Circle in counter-clockwise sense
X,Y: End point of the circle
R: Circle radius

### 3.4.2 Center-point programming

Using the current position as starting point, you define a circular movement through the **programmed end point** using the **programmed center point**.

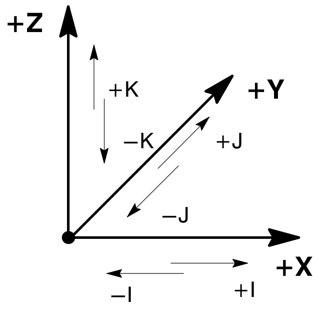
Imprecise entries can lead to two different radii (center point – starting point, center point – end point) in the internal calculation.
The control unit can compensate this by means of internal **center point correction**.

- Radius differences above the radius accuracy (MACODA parameter 7050 00010) are corrected automatically. Below this threshold, the programmed data are exclusively effective.
- The center point correction is effective at most up to the radius tolerance range (MACODA parameter 7050 00020). Greater differences trigger a runtime error.

Interpolation parameters

For circular interpolation, the interpolation parameters I, J and K are assigned to the axes involved in accordance with MACODA parameter 7010 00030 (axis classification).
They define the incremental distance between the **A circle starting point** and the **M circle center point** for each axis. Their sign results from the vector direction from A to M.

The standard assignment of the interpolation parameters is as follows:

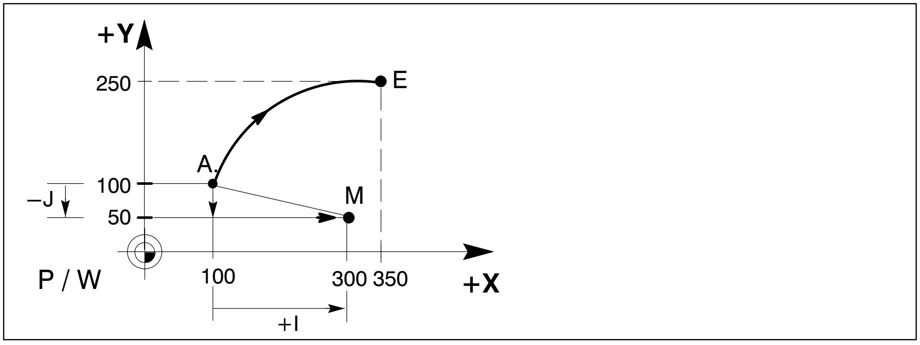


| | | |
|---|---|---|
| **I** | = M (X) – A (X) | **I, J, K** as interpolation parameters |
| **J** | = M (Y) – A (Y) | **X, Y, Z** axis portion of relevant coordinate |
| **K** | = M (Z) – A (Z) | **M** for circle center point<br>**A** for circle starting point. |

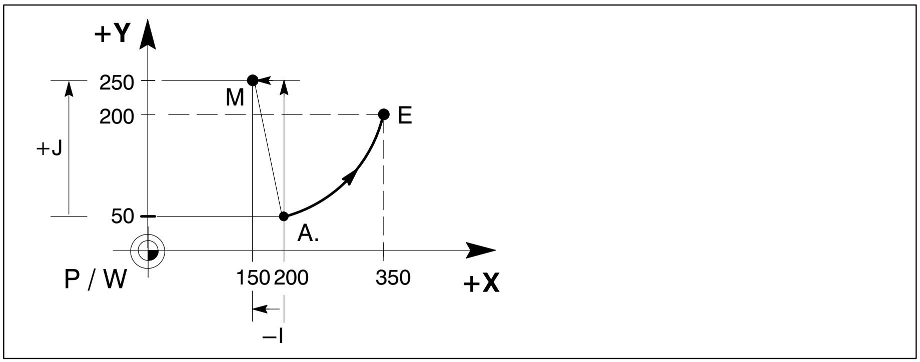
Programming

**Examples:**

N... G90 G17 G2 X350 Y250 I200 J−50 F... S... M...



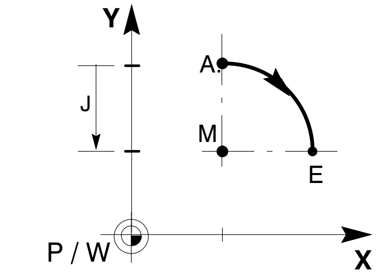
N.... G90 G17 G3 X350 Y200 I−50 J200 F... S... M...



Special case

**Quarter circle as a quadrant**

N... G17 G2 X... Y... J−... F... S... M...

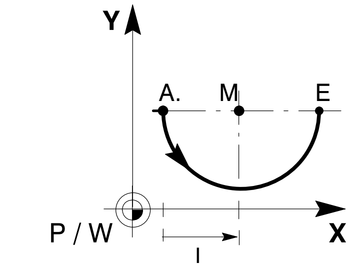


**Characteristic:**
One of the interpolation parameters is always zero and need not be written in the program. In this example, I can be omitted.

Special case

**Semicircle made up of two quadrants**

N... G17 G3 X... I... F... S... M...

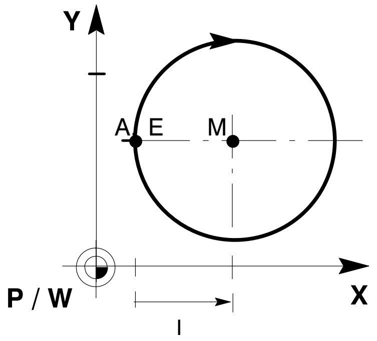


**Characteristics:**
The coordinates of the starting point and the end point are identical for one axis. The axis portion need not be specified as target. The interpolation parameter belonging to this axis is zero and may also be omitted. In this example, Y and J can be omitted.

Special case

**Full circle**

N... G17 G2 I... F... S... M...



**Characteristics:**
The coordinates of the starting point and the end point are identical. None of the two axis portions need to be specified as target. If the starting and end points are located exactly on a quadrant transition, one interpolation parameter is zero and hence need not be programmed. In this example, X, Y and J can be omitted.

☞ **If an interpolation parameter is programmed which does not correspond to the selected plane, the control unit will report the ”Programmed interpolation parameter outside the selected plane” runtime error.**

**Example:**

```
N... G17 G2 X5 I9 K7
```

K is not valid

☞ **If interpolation parameters and circle radius are programmed within one and the same block, only the radius will be used.**

## 3.5 Helical-N-Interpolation G202, G203

Effect

For a helical-N-movement, the travel to the programmed position is executed by the axes providing the working plane travel to this position along a circular arc while all other axes are moved along linearly with the effect that all axes arrive simultaneously. A maximum of 6 synchronous axes can be moved linearly, with permissible axis movement types being linear, endless, or rotary axis.

The function ”Helical-N-Interpolation” is a generalised version of the previous ”Helical Interpolation”, which is still available (refer to “Special case” below). With helical interpolation, only one linear feed axis can be moved along and this axis must be configured as a normal axis, i.e. perpendicularly to the selected working plane (MACODA parameter 7010 00030, axis classification as determined in version 108 or higher).

The axes travelling along a circular arc are clearly defined by the selected working plane (G17, G18, G19, G20). The maximum circular movement that can be programmed in one traversing block is a full circle.

The function ”Helical-N-Interpolation” allows the programming of “helical movements coupled with a change in orientation”.

Generally, the programmed feedrate applies to all the axes traversing in a block. Those axes that are moved along linearly, however, are controlled by the specific MACODA parameters set for functions G594 and G595, that are contributing to feedrate computing.

Any circular or helical movement can also be programmed as an equivalent helical-N-movement.

The Helical-N-Interpolation acts modally, i.e. it remains active until it is deselected through programming or another modal function that generates a movement.

All standard compensations like zero offset, tool length, workpiece position, or cutter compensation are also effective for helical-N path segments. Helical-N path segments can be programmed also for working in inclined planes.

Programming

| | |
|---|---|
| `G202:` | circular movement, turning clockwise |
| `G203:` | circular movement, turning counter-clockwise |

Both radius programming (R) and center-point programming (I, J, K) is possible:

- The sign of the radius determines whether the resulting center of the circle is located left (+) or right (–) of the line between the starting point and the end point.

- The size of the radius must be at least half the distance between the starting and the end point. If the radius is smaller than this value and if the balance lies within the tolerance window defined in MACODA parameter 7050 00030, the radius will be automatically corrected to half the above distance.
- If center-point programming is used, the coordinates of the center point refer to the starting point of the circular movement (center-point coordinates are incremental).
- If the starting point and the end point in the circle plane are identical, the control will automatically generate a full circle if center-point programming is used.
- MACODA parameters 7050 00010 and 7050 00020 allow to configure the required programming accuracy.
- If a center-point coordinate is programmed as lying outside the working plane, the control will display a runtime error.

Since the helical-N-interpolation is a modal function causing a movement, the active G code is shown on the MMI display of the respective modal function activated.

The behavior upon resetting the control or switching on/off is determined by the init strings configured in MACODA for control start-up or following control reset.

This function belongs to group 2.

☞ **Since the number of axes per channel is limited to a maximum of 8, coupled linear motion is limited to a maximum of 6 synchronous axes.**

Special case

**Helical interpolation**

If a third axis is programmed in addition to circular interpolation of two axes, this third axis will traverse linearly. The result is a helical movement (also refer to the figure below).

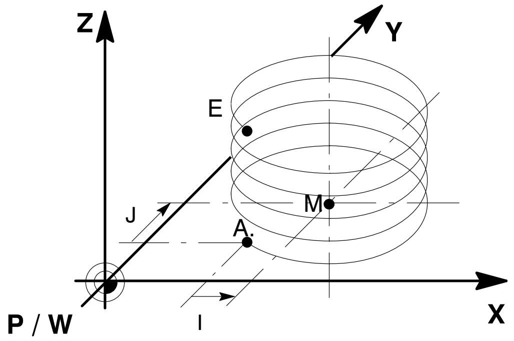
The tool-path compensation acts within the circular-path plane which can be freely selected via plane selection (G17...). The F feedrate corresponds to the real path speed.

**Example:**

Circular interpolation involving axes X and Y,
Linear interpolation of axis Z:

G Instructions G202 G203

N... G91 G17 G3 X... Y... Z... I... J... F... S... M...



**Characteristics:**
The coordinates of the starting point and the end point are identical for the X and Y coordinates. Interpolation parameter K is omitted, because the starting point is in the X–Y plane.

***Application:*** *e.g. thread cutting*

## 3.6 Dwell time in seconds G4<br>Dwell time in spindle revolutions G104

Effect

The “dwell time” can be programmed

- in **seconds** (G4) or
- in **spindle revolutions** (G104).

The dwell time is started when the G4/G104 dwell time block has been completely processed by the CNC and block execution takes place.
The program is stopped for the duration of the dwell time. A rotating spindle or traversing auxiliary axes are not stopped. Synchronous axes can reduce their lag, if necessary.
The block programmed subsequently is only executed when the programmed “dwell time” has elapsed.

The function G4/G104 is programmed with an F word for the dwell time duration in a separate block without positional data. Only auxiliary and special functions are permissible in this block.
The programmed dwell seconds or the number of spindle rotations, respectively, have to be programmed again in every G4/G104 block.

If G4/G104 is programmed with a dwell time F=0, programming G4/G104 will not cause the axes to decelerate within a G08 or G108 movement sequence. In this case, the G4/G104 block is deleted within the NC.

Programming G4/G104 without F word leads to a runtime error.

**Determination of the spindle revolutions with G104:**

To determine the spindle revolutions, the current actual revolutions are established cyclically from the main spindle, calculating the revolutions performed on that basis. In case of highly dynamic spindles, a certain deviation between the programmed and the actual spindle revolutions waited may therefore occur in the acceleration or deceleration phases.
If the configured main spindle is an analogous spindle (without speed feedback), the rpm setpoint is used for the calculation instead of the actual rpm.

The programmed spindle revolutions refer to the main spindle configured in MACODA parameter 7020 00010  or using the MAINSP function (refer to page 4–17).

Programming

G4 F<dwell time>...

where:

| | |
|---|---|
| dwell time | Dwell time in seconds |

Programming

G104 F<Number of spindle revolutions>...

where:

| | |
|---|---|
| Number of spindle revolutions | Dwell time in number of spindle revolutions |

## 3.7 Circular interpolation/helical interpolation with tangential entry G05

Effect

The control unit uses G5 to calculate a tangential circle entry. Only a transition involving no reversal of direction is referred to as 'tangential'.

The first entry tangent determines all following contour elements with G5 if several G5 movements take place consecutively.

The size and position of the arc formed are calculated by the control unit in accordance with the following:

Programming

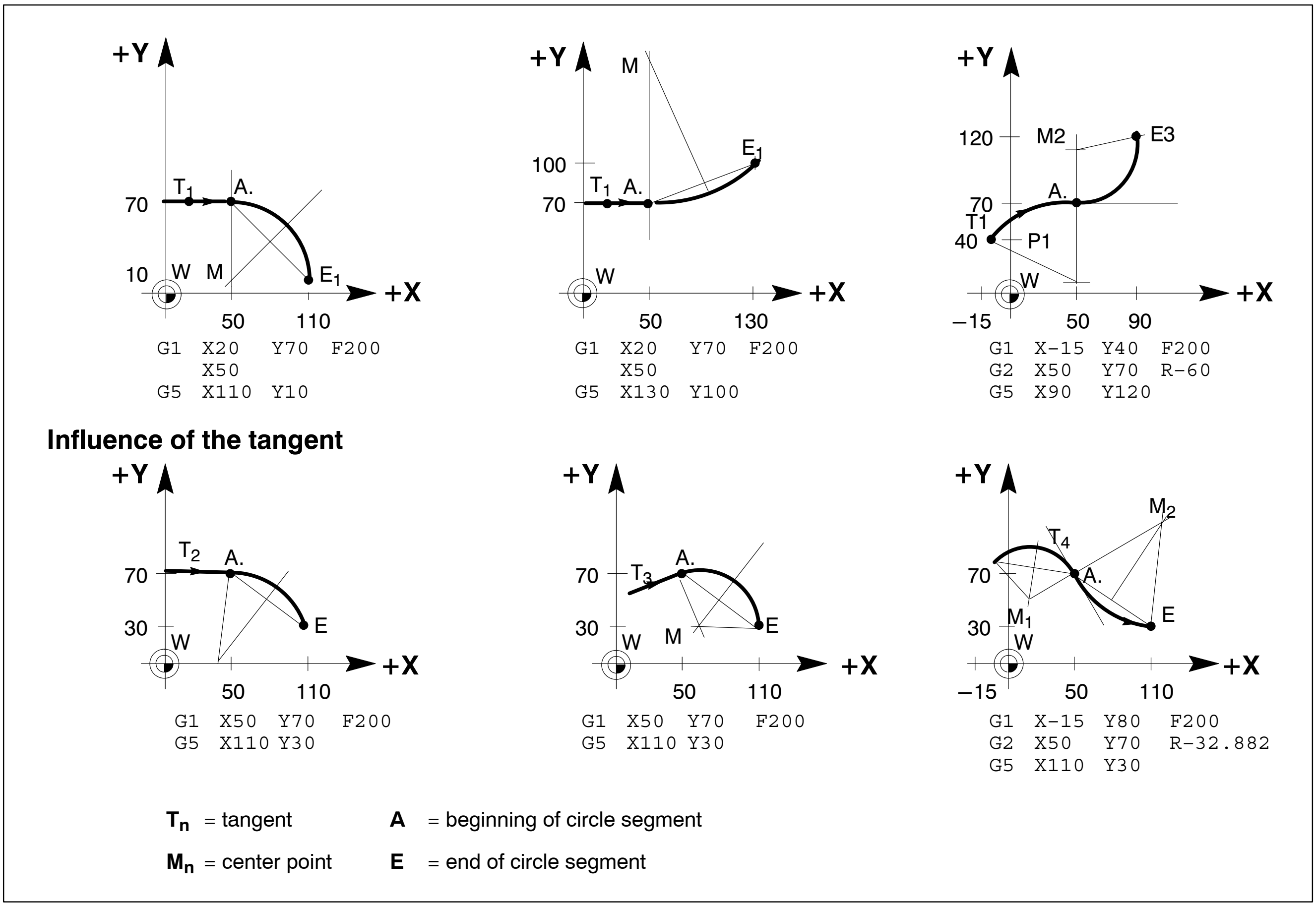
```
G5  X...  Y...
```

**No** radius is programmed.

**Restrictions:**

- Programming G5 in manual data input or as 1st block within the program is impossible, since no tangent can be calculated there.
- A block containing a traversing movement has to be programmed ahead of G5.
- The plane must not be switched over directly ahead of or during an active G5.

**CAUTION**
**When helical interpolation is used, machining marks may occur at the block transition!**
**The tangential transition refers only to the circle plane! The spatial tangent (with helical interpolation) may jump at the block transition!**



```
G1  X20   Y70  F200
    X50
G5  X110  Y10
```

```
G1  X20   Y70  F200
    X50
G5  X130  Y100
```

```
G1  X-15  Y40  F200
G2  X50   Y70  R-60
G5  X90   Y120
```

```
G1  X50  Y70   F200
G5  X110 Y30
```

```
G1  X50  Y70   F200
G5  X110 Y30
```

```
G1  X-15  Y80   F200
G2  X50   Y70   R-32.882
G5  X110  Y30
```

**T**$_n$ = tangent

**M**$_n$ = center point

**A** = beginning of circle segment

**E** = end of circle segment

*Influence of the end point*

## 3.8 Acceleration programming G06, G07, G206

Effect

The upper limits of the max. axis acceleration defined in MACODA (please refer to MACODA parameter 1010 00001) can be temporarily reduced within the part program via G6.

**DANGER**
**Incorrect axis addressing may cause inadvertent axis movements that may pose a hazard to the machine and personnel.**

**This programming refers directly to a real physical axis. A logical axis addressed, for instance, by a coordinate transformation (e.g. inclined plane) with the same axis address will lead to incorrect axis values. This might result in damage to the workpiece and/or the machine. There might even be danger to persons.**

Programming

**G06**
with axis information:

Supersedes the max. MACODA axis acceleration values defined in MACODA parameter 1010 00001 with the programmed values. Depending on the currently used measuring units (G71/G70), the control will interpret the programmed values as ”1000 inch/s$^2$” or ”m/s$^2$”.
You should program G6 preferably in a separate block.

**G06**
without axis information:

refer to G206.

**G07**

the maximum axis acceleration values specified in MACODA parameter 1010 000001 are applicable for all axes.
G7 may be programmed in connection with traversing data.

**G206**

Storing of the currently valid max. acceleration values of all axes in an internal memory. This memory is pre-initialized with the values from MACODA parameter 1010 00001 during program selection.
By programming G6 without axis information, all acceleration values stored in this memory are re-activated.

**Example 1:**

```
G6 X2 Y2
```

max. acceleration of axes X and Y is 2m/s$^2$, each.

**Example 2:**

Starting situation: The value of 8.0 m/s$^2$ is preassigned to axes X through Z in MACODA parameter 1010 00001.

| | |
|---|---|
| `G6 X1.0 Z2.1` | max. acceleration for X axis: 1.0 m/s$^2$ |
| `...` | max. acceleration for Y axis: 8.0 m/s$^2$ |
| | max. acceleration for Z axis: 2.1 m/s$^2$ |
| `G206` | storing all current axis acceleration values |
| `...` | |
| `G7` | reactivate values from MACODA parameter 1010 00001. |
| `...` | max. acceleration for X axis: 8.0 m/s$^2$ |
| | max. acceleration for Y axis: 8.0 m/s$^2$ |
| | max. acceleration for Z axis: 8.0 m/s$^2$ |
| `G6 Y5` | max. acceleration for X axis: 8.0 m/s$^2$ |
| `...` | max. acceleration for Y axis: 5.0 m/s$^2$ |
| | max. acceleration for Z axis: 8.0 m/s$^2$ |
| `G6` | max. acceleration for X axis: 1.0 m/s$^2$ |
| `...` | max. acceleration for Y axis: 8.0 m/s$^2$ |
| | max. acceleration for Z axis: 2.1 m/s$^2$ |

## 3.9 Programmable path acceleration G106, G107

Effect

Function G106 allows the upper limits entered in the MACODA parameters 7030 00210 and 7030 00220 for

- path acceleration and
- path deceleration

to be reduced in the part program. The two acceleration values can be switched over separately or together.

Irrespective of the currently effective path acceleration, the axis acceleration of the axes involved in the motion is always checked additionally, so that a programmed or preset path acceleration may be restricted.

G107 is used to switch back to the MACODA setting.

**Restriction:**

- The programmable acceleration values are restricted by the values set in MACODA.
- If an invalid value is programmed, a runtime error will occur.
- The programmed acceleration values are interpreted in dependence on G71 and G70 in m/s$^2$ or 1000 inch/s$^2$, respectively.

| | | |
|---|---|---|
| Programming | **G106** ACC<Value> | Setting the path acceleration and deceleration |
| | where: | |
| | <value > | Identical value for path acceleration and deceleration in m/s$^2$ or 1000 inch/s$^2$. |
| Programming | **G106** {UP<Value1>} {DOWN<Value2>} | Setting the path acceleration and deceleration separately. |
| | where: | |
| | UP<Value1> | **optional:** value for path acceleration in m/s$^2$ or 1000 inch/s$^2$. |
| | DOWN<Value1> | **optional:** value for path deceleration in m/s$^2$ or 1000 inch/s$^2$. |
| Programming | **G107** | Resetting of path acceleration and deceleration to MACODA setting. |

**Examples:**

```
G71
..
```

| | |
|---|---|
| `G106 UP 1.5` | Path acceleration is set at 1.5 m/s$^2$. |
| `G106 ACC 5` | Path acceleration and deceleration are set at 5 m/s$^2$. |
| `G107` | The acceleration values are reset to the MACODA setting. |
| `G106 ACC 3.5 DOWN 2` | Path acceleration is set at 3.5 m/s$^2$, path deceleration is set at 2 m/s$^2$. |

**Please note for G106, G107:**

- The G106 and G107 functions act modally and cancel each other mutually.
- The programmable acceleration values have to be programmed together with G106 in one and the same block.

☞ **For reasons of compatibility with the CC series, the alternative syntax for ”ACC” is the address letter ”E”.**

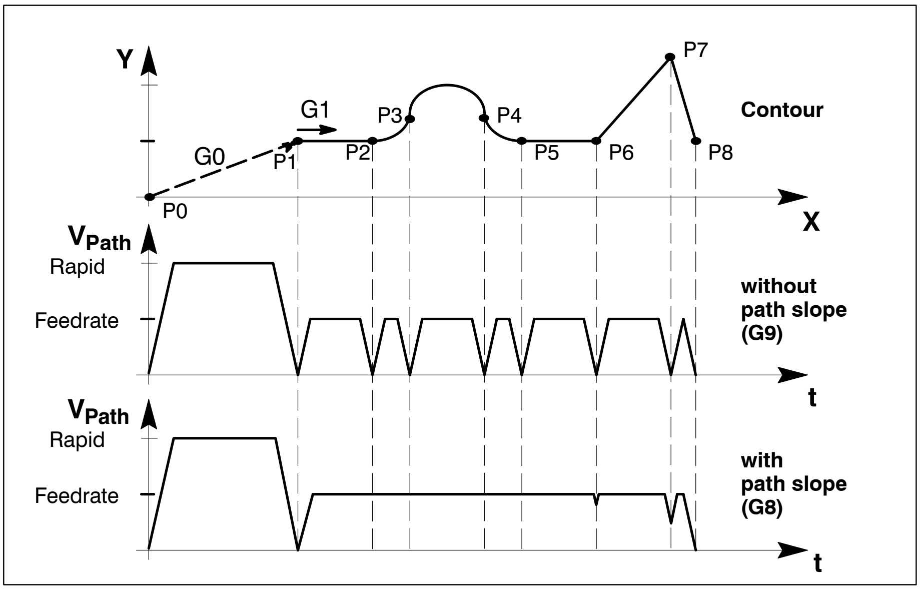
## 3.10 Path slope G08, G09

Effect

Using the ”path slope” function, the control unit attempts to generate a speed as constant as possible within the magnitude of the programmed feedrate during contour machining.

Without ”path slope”, the control unit performs a complete up and down slope (speed ramp) at the start and end of a traversing block.

With ”path slope”, this slope will – except for the beginning and end of machining – only take place to the extent required for going around a corner. In this process the PNC takes into account the value of the max. axis step change programmed in MACODA.
To limit contour deviations occurring at real corners, the maximum step change must not be set at too high a value in MACODA. On the other hand, if the maximum step change is set too low, this will result in undesirable deceleration at minor knees in the contour (quasi-continuous transitions). A solution is provided by G228 (refer to page 3–21).



Please note that the two time axes in the above illustration have different scalings.

With active G8, the P8 point will be approached within a shorter time than with active G9.

Decelerating to V=0 is performed after each G0 block!

After a G200 block, decelerating to V=0 is only performed if

- G61 or
- G163 are active!

Programming

G08: Path slope on

G09: Path slope off

The function has a modal effect. Path slope acts only on the machining axes.

**Example:** G08, path slope ON

| | |
|---|---|
| `N... G8` | (Path slope ON) |
| `N... G0 X100 Y50` | (at rapid to P1) |
| `N... G1 X150 F5000` | (continued at feedrate) |
| `N...` | |

When programming auxiliary functions while the path slope function is active, please make sure that the travel paths programmed are long enough for the amount of time required for interpolating the NC block is greater than the time required for executing the auxiliary function, including acknowledgement. (Basically, the time required for executing an NC block is defined by the travel path programmed and the feedrate.)

## 3.11 Limited-jerk velocity control G108

Effect

In contrast to function G08, G108 calculates a velocity profile for several blocks. The number of blocks can be configured in MACODA parameters 7060 00110 - 7060 00130.
This block look-ahead function accounts for longer deceleration distances, and ensures a smoother velocity profile.

Additional smoothing is provided by the optional Shape function which shares possible sudden changes in path acceleration out among several interpolation cycles, thus ensuring a continuous path acceleration profile (jerk limitation).
The number of cycles can be programmed.

Programming

Limited-jerk velocity control

| | |
|---|---|
| G108 {Shape<Shape order>} | ”Limited-jerk velocity control” on.<br>Without Shape, the order stored in MACODA parameter 7050 00320 will be active. |

where

| | |
|---|---|
| Shape<Shape order> | **optional:** Shape filter order (number of interpolation cycles):<br>0: Default (MP 7050 00320)<br>0 ..100: Programmable number of interpolation cycles |

Please note for G108:

- Functions G8, G9, G108, G408, and G608 form a modal group in each case and, therefore, cancel each other mutually.

## 3.12 Block transition without deceleration G228

Effect

The function ”Block transition without deceleration” limits the impact of the maximum step change on wide transition angles.By proper parameterisation, the behavior can be defined so as to take major knees in the contour into account in detail, whereas quasi-continuous contour transitions are rounded and thus smoothed due to the higher machining speed.

Programming

| | |
|---|---|
| G228 {K<transition angle>} | Activate function.<br>Without the K address, the transition angle stored in MACODA parameter 7030 00310 will be active. |

where:

| | |
|---|---|
| K<transition angle> | K address with<br>transition angle = 0° to 50° |

Please note for G228:

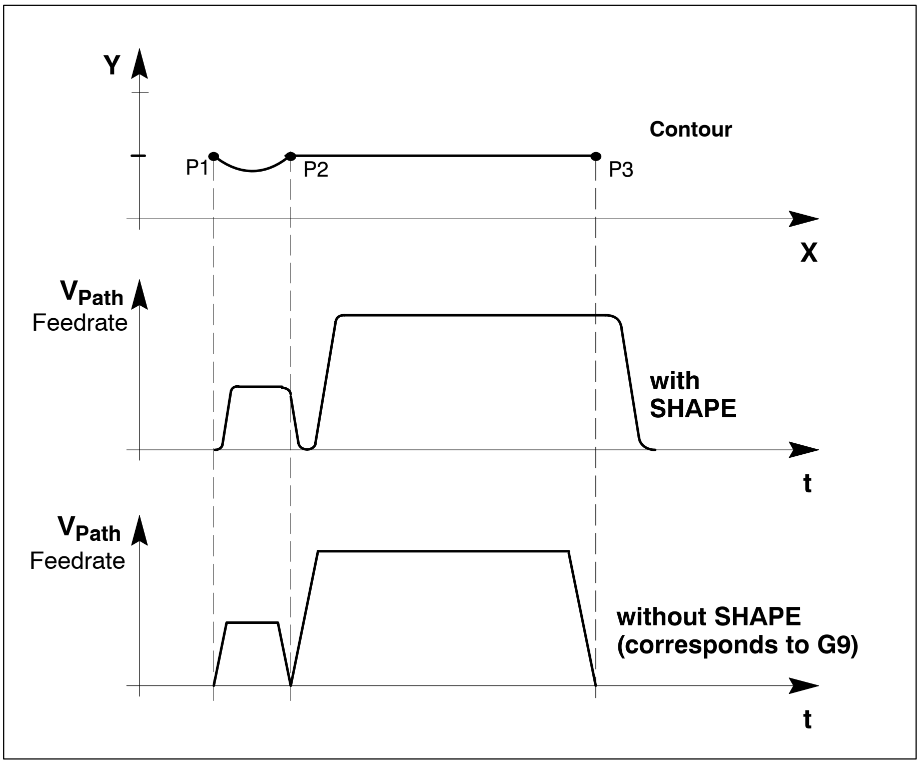
- G228 is not modal as such, but it acts modally.
- After a control reset, the respective init string setting is activated. If there is no G228 entered there, the previously activated setting remains active.

☞ **When configuring MACODA parameter 7030 00310, the desired traversing speed must be taken into account because at a high speed and a wide transition angle it is theoretically possible that a servo error occurs.**

## 3.13 Point-to-point movement using SHAPE G408

Effect

The shape function is used to share out jumps in the course of path acceleration between several interpolation cycles. This enables **“sin$^2$-shaped” path-speed behavior patterns**, i.e. jerk-free speed changes, to be made.



*Path-speed behavior patterns*

☞ **Compared to G9 (unchanged acceleration), the interpolation time (”with shape”) per point-by-point movement extends by order*interpolation cycle.**

Using the LIN and SIN parameters, the characteristics of the acceleration transition and thus a jerk-free path speed can be set.

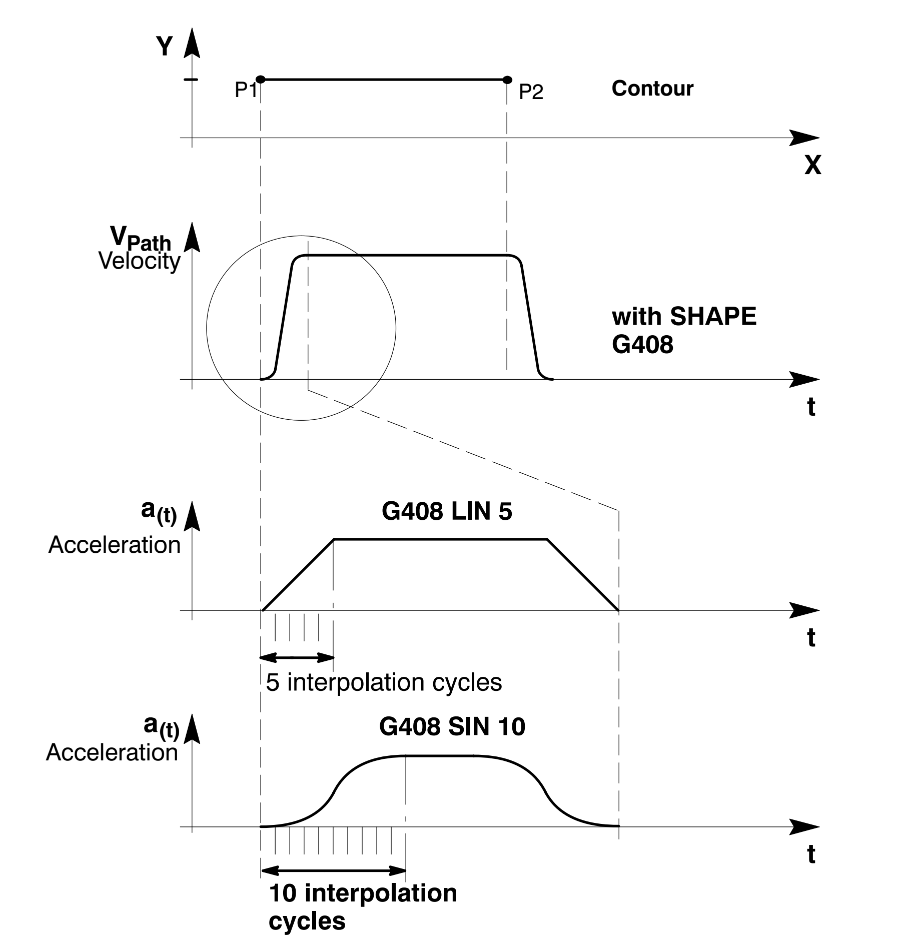
**LIN** <Number> Number of interpolation cycles (Settings: 2-41 cycles) between which an occurring path acceleration jump is to be shared out. The acceleration increase or decrease is **linear**.

**SIN** <Number> Activation of fixed acceleration-behavior pattern settings. The acceleration increase or decrease is **sin$^2$**-shaped.

The following **fixed** sin$^2$-shaped acceleration pattern settings are preset within the system:

- SIN 0: SHAPE is cancelled (=G9)
- SIN 3: 3 interpolation cycles in the ratio of 25% – 50% – 25%
- SIN 4: 4 interpolation cycles in the ratio of 12.5% – 37.5% – 37.5% – 12.5%
- SIN 5: subdivided into 5 interpolation cycles
- SIN 10: subdivided into 10 interpolation cycles
- SIN 15: subdivided into 15 interpolation cycles
- SIN 20: subdivided into 20 interpolation cycles
- SIN 40: subdivided into 40 interpolation cycles

☞ **The SIN parameter has priority over the LIN parameter.**



**Note:**
The velocity characteristic is a just a schematic diagram and may, of course, vary for the respective LIN and SIN acceleration values.

*Linear and sin$^2$-shaped acceleration transitions*

## Programming

| | |
|---|---|
| `G408` | Default setting corresponds to G408 LIN 2 (2 interpolation cycles) |
| `G408 SIN 3 LIN 5` | Acceleration jump with SIN 3 only (3 interpolation cycles with fixed acceleration characteristic setting) |
| `G408 LIN 5` | Acceleration jump with LIN 5 (5 interpolation cycles) |
| `G408 LIN 2` | corresponds to default setting (2 interpolation cycles) |

**”Invalid” programming:**

| | | |
|---|---|---|
| `G408 SIN 5` | **SIN valid,** therefore: | `G408 SIN 5` |
| `G408 LIN 41` | **LIN invalid** (value too high), therefore: | `G408 LIN 2` |
| `G408 SIN 3 LIN 5` | **SIN valid,** therefore: | `G408 SIN 3` |
| `G408 SIN 7 LIN 5` | **SIN invalid,** therefore: | `G408 LIN 5` |
| `G408 SIN 7 LIN 41` | **SIN and LIN invalid** (value too high), therefore: | `G408 LIN 2` |

The SIN parameter supersedes the LIN parameter.

The default setting will be selected if invalid values were programmed for LIN or SIN.

**Please note for G408:**

- G408 acts modally (belongs to the G8, G9, G108, G608 group)

## 3.14 Axis-by axis programmable SHAPE G608

Effect

Using the shape function, which can be programmed axis by axis, you can define a maximum permitted jerk for each synchronous axis that must not be exceeded in traversing.

Each axis is programmed to have one special **shape order** of its own. Internally, the control unit forms a **resultant shape order** based on all the axes involved for path interpolation purposes.

The **shape order** determines the sharing out of the path acceleration of one individual axis between a programmed number of interpolation steps (also ref. to G408).

For this purpose, the programmed shape order determines the max. shape order which may be effective for the respective axis.

If more than one axis is involved in the interpolation, the resulting shape order acting along the path is computed.

Programming

G608 <i axis>< Shape order i> <n axis><Shape order n>

where

| | |
|---|---|
| axis i | logical i-th axis |
| shape order i | the programmed max. Shape order of the i-th axis (= number of interpolation cycles (21 max.) among which a path acceleration jump of the i-th axis is to be shared out). |
| i | i=1...n ($n_{max}$ = 8 axes) |

**Example:**

```
N10 G608 X4 Y6 Z10
```

Shape order (X axis) = 4
Shape order (Y axis) = 6
Shape order (Z axis) = 10

**Please note for G608:**

- The G608, G8, G9, G408 functions act modally and cancel each other mutually.
- The shape order must only take on integer values (1<= shape order<=21)
- Unprogrammed axes will be preallocated with default values (also ref. to MACODA parameter 1003 00008).
- If G608 is programmed alone, all the axes are preallocated with default values (also ref. to MACODA parameter 1003 00008).
- An active G608 function will always lead to a block-transition speed = 0 and is therefore only suitable for positioning movements.
- In power-up condition, G09 is activated.

## Resulting shape order

The resulting path shape order $S_b$ is the maximum of the effective axis shape orders $S_i^{rms}$ of all the axes involved in the interpolation.

$$S_b = \max\{S_1^{rms}, \ldots, S_n^{rms}\}$$

The rms axis shape orders $S_i^{rms}$ are computed from the programmed shape orders using the formula:

$$S_i^{rms} = S_i^p \frac{a_i^{rms}}{a_i^{max}}$$

where:

$S_i^p$ Axis shape order programmed with G608

$a_i^{rms}$ rms axis acceleration in the current NC block. With linear interpolation, this depends on the current path segment of the axis. With other types of interpolation (e.g. circular, helical), it is the axis acceleration usually programmed with G06.
If the functions “inclined plane” or “axis coupling” are used, the rms axis acceleration is generally decreased once again as compared to the G06 value!

$a_i^{\max}$ Maximum axis acceleration (MACODA parameter).
**Attention:** G06 does not change this value!

## Relationship between Shape order and jerk

With axis shape orders $S_i^{rms}$, a maximum jerk $r_i^{\max}$ (derivative from acceleration after time) is defined as a limit not to be exceeded in any movement.

This jerk is defined by:

$$r_i^{max} = \frac{a_i^{max}}{S_i^p \, T_{ipo}}$$

$T_{ipo}$ is the interpolator cycle time

**Example:**
Axis X has a maximum acceleration (MACODA parameter) of 10 m/s$^2$. The programmed axis shape order is 5, and the interpolation cycle is 4 ms. According to the above formula, a maximum jerk of 500 m/s$^3$ is defined for axis X.

## 3.15 Polar coordinate programming G10 to G13

Effect

Within the polar coordinate system, and in contrast to Cartesian coordinate systems, you specify **points** by defining the radius and angle starting from a freely selectable pole.

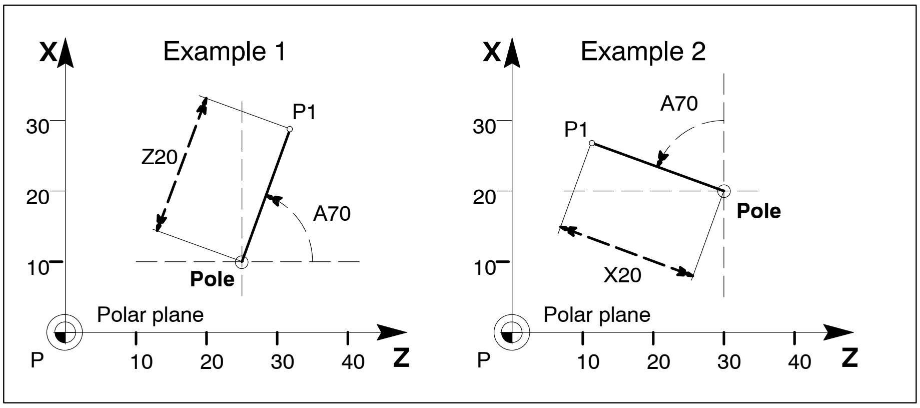
The pole corresponds to the point of reference of the polar coordinate system and can be defined via G20 within all admissible planes.

**Example:**

| | |
|---|---|
| `G20 X100 Z100` | The pole lies on the Z/X plane at the Cartesian coordinates X = 100 Z = 100. |

- If the pole is not programmed, the PNC will always use the coordinate origin as pole.
- The position of a point is described by the radius axis (one of the two axis forming the plane), the radius value and the angle. This angle relates to the programmed radius axis (refer to the example below). The syntax “A” of this angle may be declared differently in MACODA parameter 8005 0001.
- A positive axis direction of the radius axis always corresponds to the angle value of 0 degrees. All angular information refers to the positive axis direction.

| **Example 1** | **Example 2** | **Comment** |
|---|---|---|
| `N150 G20 Z25 X10` | `N150 G20 Z30 X20` | Pole determination |
| `N160 G10 Z20 A70` | `N160 G10 X20 A70` | Determination of the radius axis incl. radius value and angle |



You select the desired type of interpolation via the corresponding G instruction.

Explanation:

Programming

G10 Polar-coordinate programming at rapid travel (corresponds to G0)

G11 Polar-coordinate programming at feedrate (corresponds to G1)

G12 Polar-coordinate programming with circular clockwise interpolation (corresponds to G2, without helical)

G13 Polar-coordinate programming with circular counterclockwise interpolation (corresponds to G3, without helical)

The G0, G1, G2, G3, G5 and G10–G13 functions act modally and cancel each other mutually.

Unless explicitly switched over, the plane programmed via G20 during pole determination will also remain active after the cancellation of polar-coordinate programming.

## 3.16 Loop gain programming G14, G15

Effect

The function enables the **program-controlled change** of the KV values (loop gain) of individual axes.

This can be used for short-term increases of the rigidity of axes (e.g. for milling a bore). The KV values for the programmed axes, defined as MACODA parameters, are irrelevant during active KV programming.

where: $$KV = \frac{V\left[\frac{m}{min}\right]}{S\left[mm\right]}$$

V = Path feedrate
S = Lag

Decelerating to V=0 is performed ahead of each block containing a KV switch, since the KV value in the drive should only be switched over at standstill.
KV switching as such is always carried out **directly ahead of** the next traversing movement.

**DANGER**
**Incorrect axis addressing may cause inadvertent axis movements that may pose a hazard to the machine and personnel.**

**This programming refers directly to a real physical axis. A logical axis addressed, for instance, by a coordinate transformation (e.g. inclined plane) with the same axis address will lead to incorrect axis values. This might result in damage to the workpiece and/or the machine. There might even be danger to persons.**

Programming

G14 KV programming ON
G15 KV programming OFF

**Example:**

| | |
|---|---|
| `G14 X1.20 Y1.20 Z1.20`<br>`...` | for axes X, Y and Z a KV value of ”1.2” is specified |
| `G14 Z1.4`<br>`...` | KV value of ”1.4” defaulted for the Z axis |
| `G15 X200 Y300 Z–150`<br>`...` | The KV parameters (S-0-0104) defined in the SERCOS file apply again. |

The max. programmable KV value is ”655.35”.
G15 may also be programmed without axis information.

## 3.17 No plane G16

Effect

Function G16, “No plane”, is to be selceted for the following applications:

- If a main or secondary axis is taken out of a channel by means of “**Axis transfer**” (refer to page 3–177), the control unit automatically cancels the selected plane and activates function G16.
  In this case, circular or helical interpolation cannot be performed on this channel before a valid plane is selected.
- If no plane function (G17, G18, G19, G20) has been entered for an active channel after power-up (MACODA parameters 7060 00010 and 7060 00020), function G16, ”No plane”, will be activated for the respective channel by implication.
  In this case, no entry will appear on the channel-specific display of active functions.
- Some applications, types of machines, or processing functions do not require the definition of a plane because, e.g., no circular or helical interpolation is necessary (e.g., for channels with only one machining axis assigned).
  In this case, the axis classifications (MACODA parameter 7010 00030) have no function, either. Classification 999 – without processing function – can then be entered in MACODA for each axis.

Programming

**No plane:**

G16 Deactivate plane selection

**Please note for G16:**

- Function G16 is modal and forms a group together with G17...G20. These functions mutually cancel each other.
- After M30/control reset, the plane as defined in MACODA for power-up condition is automatically reactivated.

## 3.18 Plane selection<br>X/Y plane G17<br>Z/X plane G18<br>Y/Z plane G19

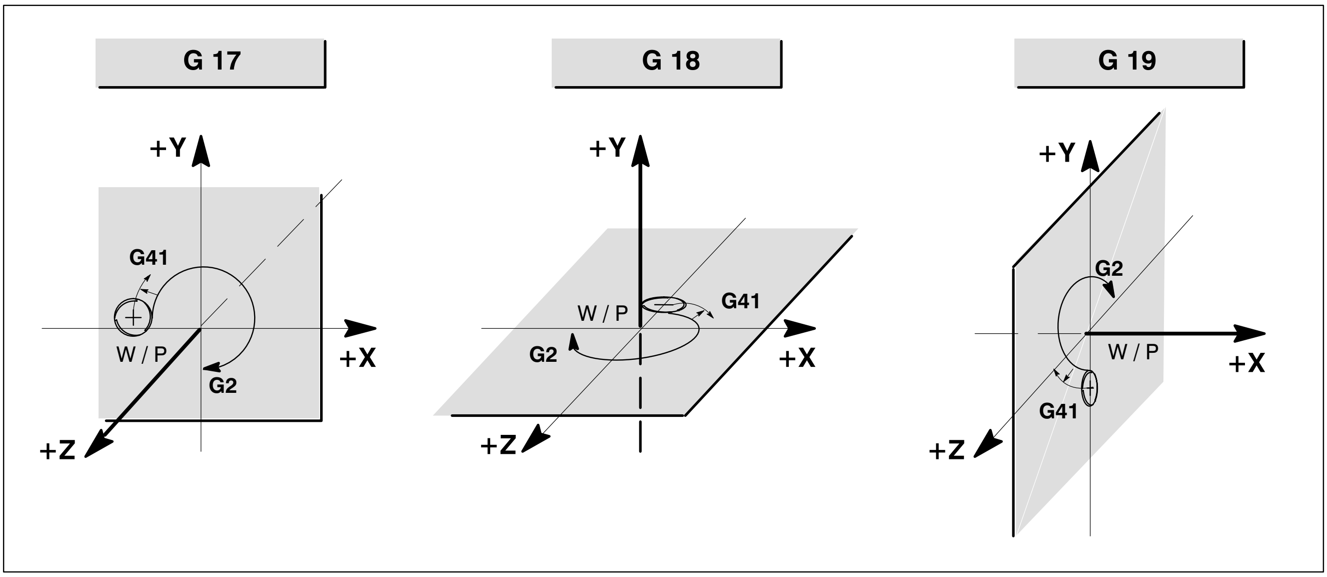
Effect

Used to define the working plane within the workpiece or program coordinate system. The effects of G2, G3, G5 as well as the polar-coordinate programming and the tool compensations are linked to this function.

Immediately after the activation of an axis, the PNC places the pole for polar-coordinate programming into the origin of the plane's coordinates. If an angle value is still active, it will be set to "0".

Within a Cartesian coordinate system, the three X, Y and Z axes form three different basic planes. These planes are characterized in that the respective third axis, as feed axis, is standing perpendicularly on these planes.

The following illustration shows the principle:



As the individual axes of the PNC may be assigned any desired name, machines with axis names other than X, Y, and Z may be configured. In this case, the individual planes are assigned the axes that correspond to the relevant "axis classification" (refer to MACODA parameter 7010 00030). The following assignment scheme applies here:

| | Classification of main axis | Classification of secondary axis |
|---|---|---|
| G17 | 1 | 2 |
| G18 | 3 | 1 |
| G19 | 2 | 3 |

If no axis meeting the above classification is defined for a selected plane, the ”Selected plane cannot be configured” runtime error will be displayed.

The axis classification also determines the **addresses of the interpolation parameters** for circular and helical interpolation in the case of center-point interpolation:

| Axis classification | Address of the interpolation parameter |
|---|---|
| 1 | I |
| 2 | J |
| 3 | K |

If more than 2 machining axes (= feed axes) are defined within the system, the **infeed axis** is determined according to the following assignment scheme:

| Plane | Classification of feed axis |
|---|---|
| G17 | 3 |
| G18 | 2 |
| G19 | 1 |

If no axis meeting the stated axis classification has been defined, the first axis within the system which does not have the axis classification of the main and secondary axis will always be selected by the system as feed axis.

**Example:**

| System axis index*) | Axis address | Axis classification |
|---|---|---|
| 0 | Y | 2 |
| 1 | B | 200 |
| 2 | C | 300 |
| 3 | X | 1 |

*) corresponds to the order specified in the MACODA parameters

In the case of a G17 plane, the B axis would be the feed axis, since it is the axis with the lowest system axis index (except for the axis having the 1 and 2 classification) and since, in addition, an axis with a classification of 3 does not exist.

Interaction of functions

From a functional perspective, the definition of a pole via G20 is equivalent to the selection of a plane.

Impact on tool compensation with standard axis classification being used:

| G instruction | Cutter-radius compensation Circular interpolation | Cutter-length compens. Feed axis for standard drilling cycles |
|---|---|---|
| G17 | X/Y plane | Z axis |
| G18 | Z/X plane | Y axis |
| G19 | Y/Z plane | X axis |

Programming

**Example:**

`N ... G19 ...` ( (Selection of the Y/Z plane)

- G16, G17, G18, G19 and G20 are modal and cancel each other mutually (for G16, “No plane”, please refer to section 3.17).
- After M30 the plane defined as power-up condition in the MACODA parameters will automatically become active.
- No plane change must be programmed with active cutter-path compensation (G41 or G42).

## 3.19 Plane selection 2 out of 8 axes G20<br>Pole programming for polar-coordinate programming

Effect

G20 allows the free selection of the circular interpolation and cutter-radius compensation plane. In addition, G20 is used to determine the pole for polar-coordinate programming (please refer to G10–G13).

**CAUTION**
**Danger of confusion through incorrect programming. Possibility of damaging the machine.**
**Whereas the X, Y and Z axes represent the 3 main axes of the current workpiece coordinate system, all the other axis addresses (e.g. the ”A” rotary axes) always designate real physical axes.**

Programming

In the G20 block you indicate the axes of the desired plane. The control unit will interpret programmed axis values (e.g. X**100** Y**40**) as **pole coordinates**.

`N... G20 X0 Y0` (Selection of the X/Y plane as interpolation plane. The pole for polar-coordinate programming is set to X=0 and Y=0)

`N... G20 Y100 Z200` (Selection of the Y/Z plane as interpolation plane. The pole for polar-coordinate programming is set to Y=100 and Z=200).

- G20 may only be programmed together with **two** axis addresses. If G20 is programmed without, with one or with more than two axis addresses, an error message is output and processing is stopped at the end of the previous block.
- G20 is latching. It will delete the G17, G18 and G19 functions. After M30 the plane defined as power-up condition in the MACODA parameters will automatically become active.
- After M30 the plane defined as power-up condition in the MACODA parameters will automatically become active.
- The axes of the selected plane will automatically have the cutter-path compensation assigned.
- You must **not program G20 during active cutter-path compensation**. Therefore, an active cutter-path compensation must be exited with G40 before selecting a new plane.
- With active helical interpolation, the programmed axis which does not lie within the circular interpolation plane will be moved along linearly.

☞ **Circular interpolation is only possible in the axis of the selected planes. As is described in section 3.4.2, center-point programming requires the specification of interpolation parameters. The assignment of interpolation parameters and the corresponding axis is defined in the ”axis classification” MACODA parameter 7010 00030.**

**Determination of the interpolation parameters:**

The table to the right shows the correlation between the axis classification and the required interpolation parameter. If both axes of the plane are within one and the same column, the parameters at the right margin apply, otherwise those at the bottom margin.

| | | | |
|---|---|---|---|
| 1 | 10 | 100 | I |
| 2 | 20 | 200 | J |
| 3 | 30 | 300 | K |
| I | J | K | |

## 3.20 Programming of axis classifications G21

Effect

Function G21, "Axis classification" (also refer to G17, G18, G19, plane selection, in section 3.18), defines the **functional significance** of an axis on a machining channel.

**Axis classifications** define the following:

- The axes defining the G17, G18 and G19 planes and which of these axes are the main, the secondary and the feed axes,
- which of the two axes programmed in G20 is the main axis and which one is the secondary axis, and
- which of the interpolation parameters I, J, and K is assigned to the respective main axis and the secondary axis for circular and helical interpolation.

You can specify the axis classification of all logical axes on each channel by using MACODA parameter 7010 00030.

**Transferring an axis** may **impact** the functional relevance of logical axes. Therefore, the following applies:

- If an axis not included in the power-up condition of a channel is transferred to this channel, this axis is assigned the "neutral axis classification" 999 (no functional significance) for the time being.
  Using G21, the axis classification is then definitely determined in the part program, so that these axes can co-define a plane in the further course and thus participate in a circular or helical interpolation.
  When programming G21, please note that no axis classification of functional significance (1, 2, 3, 10, 20, 30, 100, 200, 300) may be assigned more than **once** to any one channel. By contrast, axis classification '999' (no functional significance) may be assigned any number of times on any one channel.
- If an axis pertaining to a channel in its power-up condition is first transferred to another channel and then transferred back to its original channel, this axis is reassigned its original axis classification in the power-up condition.
- If an axis is removed from a group of axes on the currently active plane, i.e. this axis is either the main or secondary axis of the selected working plane, this selected plane is not available any more because one of its defining elements is missing.
  The control unit will then implicitly deactivate the selected plane and instead activate the **G16 function, "No plane"**.

**Example**:

| | |
|---|---|
| `N100 G17 X0 Y0 Z0` | Default axis classification: X=1, Y=2, Z=3. |
| `...` | |
| `N200 G512(Y)` | Y is removed from the group of axes. Implicit switch to G16.<br>Circular interpolation is rendered impossible. |
| `N210 G511(YA)` | Axis YA is included in the group of axes and assigned a neutral classification. |
| `N220 G21 YA2` | YA is assigned axis classification 2. |

| | |
|---|---|
| `N230 G17` | Switch to X/YA plane. |
| `N240 G2 X.. YA..` | Circular interpolation is possible again. |

Programming

**Programming of axis classifications:**

G21 (<LANi><Axis classification>,..,<LANn><Axis classification>)

where

| | |
|---|---|
| LAN | designation of logical axis/axes |
| Axis classification | programmable axis classification value.<br>Permitted values:<br>1, 2, 3, 10, 20, 30, 100, 200, 300, 999.<br>Please note that – with the exception of ’999’ – no axis classification may be assigned more than once on each channel! |

**Example**:

| | |
|---|---|
| `G21 X1 Y2 X3 B200` | The part program uses the axis classification X=1, Y=2, Z=3, B=200 |

## 3.21 Table activation G22

Effect

Use G22 to activate:

- zero-shift tables
- compensation tables
- tables for the ”inclined plane” function

These tables are stored as ASCII files in the file system of the PNC. The number of tables is limited by the storage capacity of this file system.

Programming

| | |
|---|---|
| N... G22 V {<path>}<file name> | activation of zero-shift table |
| N... G22 K {<path>}<file name> | activation of a compensation table |
| N... G22 ID {<path>}<file name> | activation of a compensation table “inclined plane” |

where:

| | |
|---|---|
| <file name> | freely defined file name |
| <path> | optional statement of path (directory) where the file is stored |

☞ **There has to be a blank before the “{<path>}<file name>”.**

**Examples:**

| | |
|---|---|
| `G22 V /mnt/npvtab1.npv` | Activates the zero-shift table ”npvtab1.npv” in the mounted directory ”/mnt”. |
| `G22 K geotab2.geo` | Activates the geometry compensation table ”geotab2.geo”. The file will be searched in the ”/database” directory. If the table you search is available there, it will be activated. Otherwise, the search will continue on the search path for subprograms and the first table with the name geotab2.geo that is found will be activated. |
| `G22 V npvtab3.npv`<br>`    K geotab3.geo` | Tables ”npvtab3.npv” and ”geotab3.geo” will be searched in the ”/database” directory and – if not found there – on the search path for subprograms and then activated. Several tables can be activated within one and the same block. |

**Please note for G22 and zero shift tables:**

- Table columns are assigned to the axes on a channel via the axis names entered on the table. These may be names of both logical and physical axes, with logical axis names taking precedence over physical axis names.
  You can select the option ”Strict assignment” for the table in the table editor (or when writing the table in CPL).
  If the option ”Strict assignment” is available for a zero-shift table and if this table is activated using **G22**, the system will check whether the current axis configuration of the respective channel matches the table entries. If this is not the case, an error message is displayed and program execution is aborted.
  If the option ”Strict assignment” has not been activated, any discrepancies between the current axis configuration and the table columns will not give rise to an error message. This allows to activate tables containing shift values of just some of the axes. Also, this allows to use tables containing additional columns with shift values of axes to be transferred to the respective channel at a later point in time.

  **Example 1:** G22 V npvtab1.npv (zero shift table on channel 1)
  Channel 1 contains 3 axes
  Strict assignment: 3 channel axes < – > 3 table axes

  **Example 2:** G22 V npvtab2.npv (zero shift table on channel 2)
  Channel 2 contains 4 axes
  No strict assignment: 4 channel axes < – > 2 table axes
  —> No error message as “strict assignment” has **not** been activated.

  **Example 3:** G22 V npvtab1.npv (zero shift table on channel 2)
  Channel 2 contains 4 axes
  Strict assignment is active
  —> Error message as “strict assignment” has been activated.

☞ **Please refer to the PNC Operating Manual on how to create or edit tables!**

## 3.22 Jump destinations: Unconditional jump (block number) G24 Conditional jump (interface signal) G23 Jump backwards GOTOB Jump forwards GOTOF

As a rule, main program and subprogram blocks and cycles are executed in the same order as they were programmed.

The processing sequence can be changed by program jumps. There are various jump destination types available for this purpose.

Refer to the explanation, section 2.1.4, p. 2–11 ff.

## 3.23 Tapping without compensation chuck G32

Effect

The PNC synchronizes linear interpolation of the drill axis with the spindle switched to C axis operation. This eliminates the need for a compensation chuck which would otherwise be required for taking up the speed difference between the drill axis and the spindle.

The ”G32 ACTIVE” interface signal will be output for the duration of the tapping process. During this time only the feed potentiometer is active.

Programming

G32 <Drill axis><Infeed depth> {F<Feedrate value>} M<3|4> S<Speed>|H<Thread pitch>

In addition to the infeed per cut, the following must be entered for programming a G32 block:

- spindle speed (S) or the thread pitch (H) and
- the sense of rotation (M3/M4)
  M and S act only within the programmed G32 block.

The PNC uses the active path feed (F word) if no other value is stated in the G32 block.

The thread pitch results from the ratio between the path feed and the speed, unless the thread pitch (H) is programmed.

**Example:**

```
N10 G0 X20 Y15 Z10 F1000 S5000     Positioning the axes
N20 G32 Z-20 F1000 M3 S1000        Drilling (Z drill axis)
N30 G32 Z5 F1000 M4 S1000          Retraction (Z drill axis)
```

In the case of direct programming of the H thread pitch, the pitch, if smaller than 1, is to be programmed as follows:

- H.5 instead of H0.5 or
- H 0.5 instead of H0.5

**Example:**

```
N10 G0 X30 Y5 Z0 F1500             Positioning the axes
N20 G32 Z-20 M3 H.75               Drilling (Z drill axis)
N30 G32 Z0 M4 H.75                 Retraction (Z drill axis)
```

**CAUTION**
**Possible damage to workpieces!**
**Drilling and retraction must always be programmed with identical thread pitch (F/S)!**

- G32 acts block by block.
- Neither M19, nor M5 are required ahead of G32. Switch-over to C axis operation is done automatically. Prior to starting, the PNC internally waits for ”INPOS” of all axes involved. In case an axis drifts out of its INPOS range, G32 will not be started (for INPOS range, please refer to MACODA parameters).

- The drilling and retraction blocks must be programmed directly one after another, otherwise the ”Retraction block not programmed” runtime error will appear.
- After the retraction block, the spindle will return to spindle operation.

☞ **Please refer to section 3.24 for tapping using several spindles, G532, and to section 3.83 for suppressing axes for calculating the feedrate, G594.**

## 3.24 Activation of tapping without compensation chuck for several spindles G532

Effect

You can tap threads in parallel without compensation chuck using up to 8 spindles. Programming and effect correspond to G32.
However, you use G532 to determine the spindles to which G32 is to refer.
If you do not program any G532, G32 will always refer to the 1st spindle.

Programming

| | |
|---|---|
| `G532 CAX<i>..{CAX<n>}` | tapping (G32) using the i-th spindle(s). |
| `G532 GRP<j>` | tapping (G32) using (a) spindle(s) from the j-th spindle group |
| `G532 GRP<j>..CAX<i>..{CAX<n>}` | tapping (G32) using (a) spindle(s) from the j-th spindle group and additionally the i-th spindle(s) |

where:

| | |
|---|---|
| CAX | spindle axis |
| i = 1 .. max. 8 (n) | number of the spindle |
| GRP | spindle group |
| j = 1 .. max. 4 | number of the spindle group |

**Example:**

| | |
|---|---|
| `G532 CAX1` | tapping (G32) using the 1st spindle |
| `G532 CAX2 CAX4 CAX7` | tapping (G32) using the 2nd, 4th and 7th spindle |
| `G532 GRP2` | tapping (G32) using (a) spindle(s) from spindle group 2 |
| `G532 GRP3 CAX4` | tapping (G32) using (a) spindle(s) from spindle group 3 and the additional 4th spindle |

**Please note for G532:**

- With G532 you cannot activate more than **one** spindle group.
- CAX1 through CAX8 can be combined in any way and in addition to any spindle group.

☞ **The information will remain active until a new G532 is programmed (i.e. even after CONTROL RESET!). After a control reset, this function can be entered in the MACODA start-up string.**

## 3.25 Retraction from tapped hole G9321, G9322

Effect

If function ”Tapping without compensation chuck” (G32) was cancelled (by ”control reset” or due to a voltage drop) while the screw tap was still in operation, G9321 or G9322 can be used to retract the tap from the tapped hole. Programming can be done by manual input or within a part program (cycle).

Programming

| | |
|---|---|
| `G9321` | Switches the spindle(s) running again in spindle speed mode after ”control reset” (control start-up) to Position mode. |
| `G9322 F` | Initiates the retraction motion proper (with F value). |

☞ **G9321 must be programmed before retracting!**

2 situations must be distinguished when applying the function ”Retraction from tapped hole”:

- **Retraction after ”control reset”** (automatic retraction)
  The data saved when starting the tapping process is retained.
  The spindles are switched over to position mode with G9321. If G9322 is programmed next, the tap moves out of the thread and on to the stored starting position. Only the desired feedrate (F value) must be programmed together with G9322.

- **Retraction after power failure** (manual retraction)
  The data saved when starting the tapping process is lost. In this case, the parameters have to be programmed explicitly together with G9322.

  G91 G9322 S<Speed> F<Feedrate> M3/M4 <Drill axis> <incremental path>

**Preconditions for the use of G9321/G9322:**

- The C axes involved must be defined as endless rotary axes (MACODA parameter 1001 0000 4 = 2 and SERCOS parameter S-0-0076 = Ob1xxxxxxx).
- In SERCOS secondary operation mode 1 (S-0-0033), bit 8 for drive-controlled change of operation mode must not be set: S-0-0033 = Ob000001011 or S-0-0033 = Ob000001100.

**Application of subprograms for retraction from tapping:**

To make the application of the function ”Retraction from tapping” easier, it is recommended that you write one subprogram (cycle), each, for automatic and manual retraction. With MACODA parameters 3090 00001 and 3090 00002, these subprograms can then be assigned to any G-code available.

**Example: Automatic** retraction cycle
AutoTR[feedrate]

```
N1
2    If P1=NUL THEN
N3     (MSG, ** P1 NO FEEDRATE PROGRAMMED **)
N4     M0
5      GOTO N3
6    ELSE
7      GVORSCH%=SD(1,7,2)
8      FVORSCH%=SD(5,1,2)
N9     G9321
N10    G94 G9322 F(P1)
N11    G[GFEED%]
12     IF GFEED%=94 THEN
N13      F[FFEED%]
14     ENDIF
15   ENDIF
M30
```

**Example: Manual** retraction cycle
ManTR[drill axis number, path, master pitch, feedrate]

```
N1
2    If P1=NUL THEN
N3     (MSG, ** P1 AXIS NOT PROGRAMMED **)
N4     M0
5      GOTO N3
6    ENDIF
7    If P2=NUL THEN
N8     (MSG, ** P2 PATH NOT PROGRAMMED **)
N9     M0
10     GOTO N8
11   ENDIF
12   If P3=NUL THEN
N13    (MSG, ** P3 THREAD PITCH NOT PROGRAMMED **)
N14    M0
15     GOTO N13
16   ENDIF
17   If P4=NUL THEN
N18    (MSG, ** P4 FEEDRATE NOT PROGRAMMED **)
N19    M0
20     GOTO N18
21   ENDIF
22   BAXIS%=ROUND(P1)
23   IF P3>0 THEN
24     MCODE1%=4
25   ELSE
26     MCODE1%=3
27   ENDIF
28   PITCH=ABS(P3)
29   GABS_INC%=SD(1,4,2)
30   GFEED%=SD(1,7,2)
```

```
31   FFEED%=SD(5,1,2)
N32  G9321
N33  G91 G94 G9322 [AXP(BAXIS%,P2)] H[PITCH] F[P4]
     M[MCODE1%]
N34  G[GABS_INC%] G[GFEED%]
35   IF GFEED%=94 THEN
N36    F[FFEED%]
37   ENDIF
M30
```

For the cycles to run, some configurations need to be set in MACODA.

**MACODA configuration:**

Parameter: 3090 00001

| | |
|---|---|
| 0 | 9032 |
| 1 | 9132 |
| 2 | ... |

Parameter: 3090 00002

| | |
|---|---|
| 0 | AutoTR |
| 1 | ManTR |
| 2 | ... |

**Example:**

| | |
|---|---|
| `G9032[1000]` | automatic retraction at a feedrate of F1000mm/min |
| `G9132[3,-100,0.5,500]` | manual retraction at a pitch of 0.5, a retraction path of –100mm, F500, with the 3rd logical axis of the channel as the drill axis |

## 3.26 Tapping G33

Effect

Activates tapping of

- Longitudinal threads
  (Cutting motion in parallel to the main axis of the active plane),
- Transversal threads
  (Cutting motion in parallel to the secondary axis of the active plane),
- Tapered threads
  (the main and the secondary axis of the active plane are involved in the cutting movement).

G33 can be programmed with a speed-controlled and position-controlled spindle.
The cutting motion is always linked to the main spindle active on the channel in question (refer to page 4–17).
The feedrate of the cutting motion results from the current spindle speed and the programmed pitch portions (fixed, variable – refer to ”Programming”).

Special features:

- single and multiple threads can be produced
- constant and variable thread pitches can be programmed
- special dynamics adjustment during the cutting process
- programmable quick retraction movement
- chained threads can be produced.

☞ **The feedrate potentiometer has no effect while G33 is active.**

☞ **G33 acts modally and belongs to the same group of NC functions as G0, G1, G2, G3, etc.**

☞ **Like circular interpolation (G2, G3), the tapping function is subject to the active plane (G17 ... G20).**

The behavior of the ”Tapping” function is normally defined in MACODA parameters 7050 006xx.
In individual cases, or during initial commissioning, it may be advantageous to adjust individual sections quickly. This requirement is satisfied by function G533 (for a description, refer to page 3–49 ff.).

G533 provides for

- the adjustment of the dynamics and retraction movement
- change-over of the spindle mode (speed control, position control)
- specification of the channel IF signal NC-O20.4, ”Tapping cycle active”.

Programming

G33 <End point> <fixed thread pitch> {<var. pitch>} {<starting angle>}

where

<End point>
Axis coordinates of the main and secondary axis of the active plane. The active plane is defined by G17, G18, G19 or G20.
Example:
The active plane for G18 is usually defined by axes Z (main axis) and X (secondary axis).

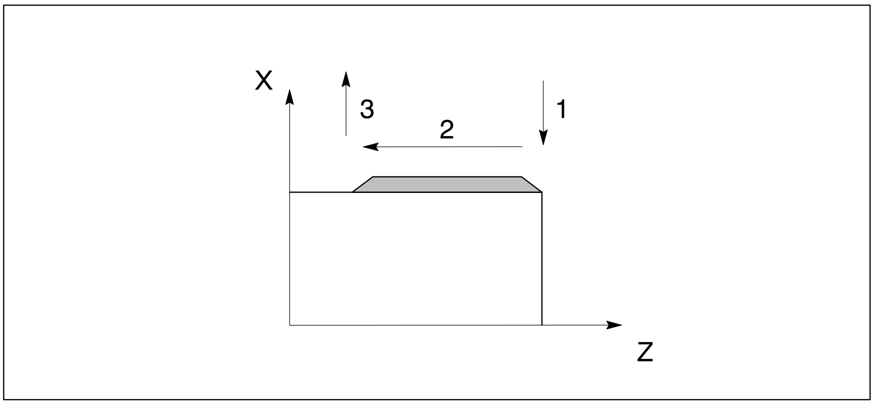
<fixed pitch>
Defines the path (in mm) traveled in the direction of the main or secondary axis with each spindle revolution.
This value is programmed by the interpolation parameter (I, J or K) valid in the respective active plane.
In the event of tapered threads, the pitch thread specified always has to relate to the main cutting direction.
Example:
For G18, parameter K is assigned to the main axis, and I to the secondary axis. For a longitudinal thread (pitch in direction of main axis), the fixed thread pitch is programmed with the K address.

<var. pitch>
Optional parameter with address DF.
Defines the pitch increase/decrease per spindle revolution in mm.
Programming: ”DF<Value>” with <Value> in mm.

<starting angle>
Optional parameter.
If no <Starting angle> has been programmed, it is assumed to be 0 degrees.
The starting angle (offset) is needed for multiple threads.
The interpolation parameter not assigned to the active plane is used as address.
Example:
Addresses I and K have been assigned to the plane for G18. Therefore, the address of the starting angle is J.

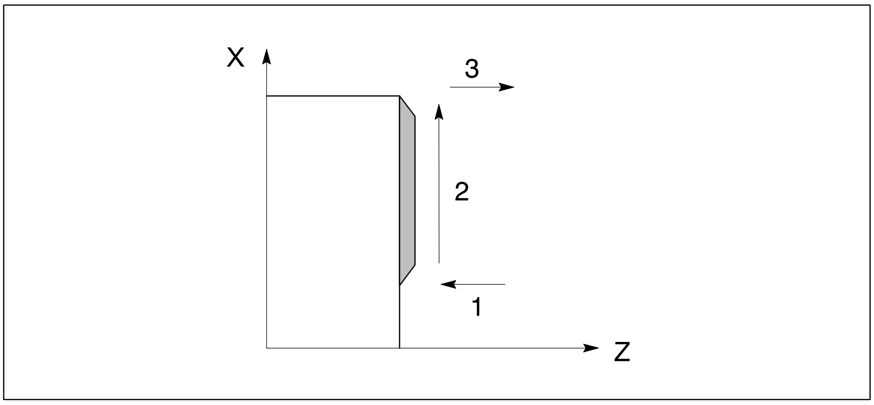
☞ **Alternatively, a syntax of the type used for CC220/Typ1 osa may be used. Details on request.**

**Programming example:** Longitudinal thread



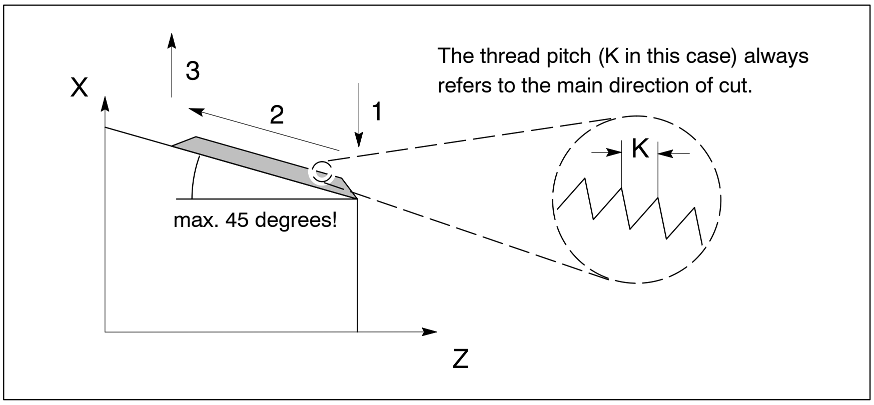
| | |
|---|---|
| `G91 G18 G8 M3 S1000` | Activate incremental data input.<br>Activate Z/X plane |
| `G0 X-10` | Infeed motion of the cutting tool (1). |
| `G33 Z-50 K2` | Tapping (2). End point: incremental by –50 mm in Z direction.<br>Fixed thread pitch: 2 mm/rev. Interpolation parameter: K in this case. |
| `G0 X10` | Move out cutting tool (3) |

**Programming example:** Transversal thread



| | |
|---|---|
| `G91 G18 G8 M3 S1000` | Activate incremental data input.<br>Activate Z/X plane |
| `G0 Z-10` | Infeed motion of the cutting tool (1). |
| `G33 X40 I2` | Tapping (2). End point: incremental by +40 mm in X direction.<br>Fixed thread pitch: 2 mm/rev. Interpolation parameter: I in this case. |
| `G0 Z10` | Move out cutting tool (3) |

**Programming example:** Tapered longitudinal thread



The thread pitch (K in this case) always refers to the main direction of cut.

| | |
|---|---|
| `G91 G18 G8 M3 S1000` | Activate incremental data input.<br>Activate Z/X plane |
| `G0 X-20` | Infeed motion of the cutting tool (1). |
| `G33 Z-50 X15 K2` | Tapping (2). End point: incremental by –50 mm in Z direction and +15 mm in X direction. Fixed thread pitch: 2 mm/rev. Interpolation parameter: K in this case. |
| `G0 X5` | Move out cutting tool (3) |

### Chained threads

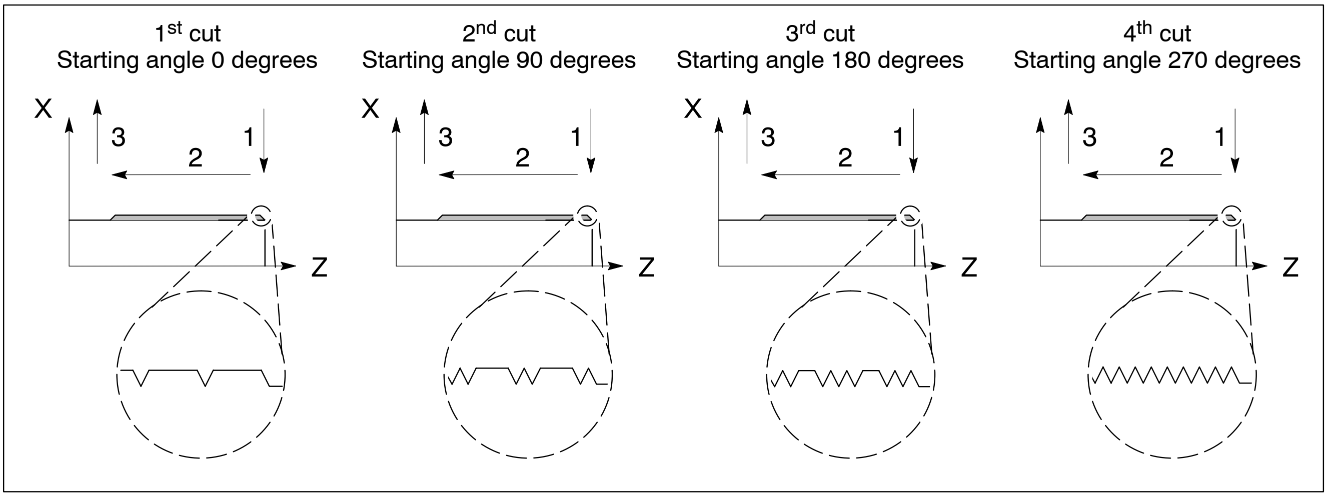
- can be produced from all types of thread.
- are programmed by several consecutive G33 blocks.

With each G33 block programmed, the NC checks whether a subsequent G33 block has been programmed with path information. If this is the case, the next block is processed without halting the axes.

### Multi-start threads

Multi-start threads are produced by a starting angle offset (for starting angle, refer to page 3–44).

Example: A four-start thread is produced by four cuts displaced by 90 degrees each (0,90, 180, 270).


1st cut
Starting angle 0 degrees
2nd cut
Starting angle 90 degrees
3rd cut
Starting angle 180 degrees
4th cut
Starting angle 270 degrees
X
3
2
1
Z

| | |
|---|---|
| `...G18...` | Activate Z/X plane |
| `...` | |
| `G33...J0` | First thread. Starting angle: 0 degrees |
| `...` | |
| `G33...J90` | Second thread: Starting angle: 90 degrees |
| `...` | |
| `G33...J180` | Third thread. Starting angle: 180 degrees |
| `...` | |
| `G33...J270` | Fourth thread. Starting angle: 270 degrees |
| `...` | |

### Dynamic behavior

In the beginning and at the end of a thread-cutting process, the axes involved have to be accelerated and stopped, respectively.

★ Therefore, you should always provide for a sufficiently long entry path (for accelerating the cutting axes) and discharge path (for stopping).

As a rule, a distinction is made between 2 process options:

- ”hard” start and ”hard” end of the cutting motion
  In the beginning of the G33 movement, the axis/axes jump(s) to the cutting speed (spindle speed * fixed pitch) when the starting angle is reached. At the end of the G33 movement the speed returns to 0.
- Start/end of the cutting motion with individual dynamics selection:
  Since the ”hard” solution is not always desirable, or cannot be performed due to restrictions existing in the area of axis dynamics, you may set the dynamic behavior concerning the speed jump, starting and deceleration speed individually.
  – statically with MACODA (7050 00610, 7050 00615 and 7050 00620)
  – dynamically in the part program using ”G533 DYN ...”
  (refer to page 3–49)).

  The control unit uses the programmed starting angle to compute a starting angle offset, taking into account the slope of the acceleration ramp. Thus, it can be ensured that the same thread is always cut, regardless of the size of the acceleration.

  At the end of the thread, the cutting axis/axes are uncoupled from the spindle and initially decelerated to the jump speed, depending on the deceleration setting, in order to be finally **decelerated to stop**.

  However, if the G33 block is followed by another traversing block with active G8 or G108, the motion of this block will start at the speed that would have resulted if the thread-cutting block had been a G1 block.

### Fast retract

The ”fast retract” function can be used in conjunction with G33.
If retract data has been

- configured
  (– statically with MACODA (7050 00645, 7050 00650), or
  – dynamically in the part program using ”G533 RD ...”
  (refer to page 3–49))
  **and**
- has been activated
  (– statically with MACODA (7050 00640), or
  – dynamically in the part program using ”G533 RON1 ... ”
  (refer to page 3–49)),

a positive edge at the channel IF signal NC-I7.4, ”fast retract”, will initiate the retraction with the following sequence:

1. The cutting motion is superimposed by a motion that is oriented vertically to the main cutting direction.
2. If more than 70% of the retract path have been traveled, the cutting axis/axes is/are uncoupled from the spindle and stopped at the configured deceleration (MACODA 7050 00620).

☞ **If the retract motion was initiated by NC-I7.4, this condition can only be canceled by ”Control reset” or ”Moving away from the contour”.**

Retract motions are always carried out perpendicularly to the main cutting direction of the secondary cutting axis.

Retract motions are automatically initiated if either of the events ”Channel reset”, ”System reset”, or ”Spindle reset” is triggered by the NC.

**Programming example:** Retraction from longitudinal thread

| | |
|---|---|
| `G18 G533 RON1 RD(0,5)` | Activate Z/X plane (G18)<br>Activate fast retract (RON1).<br>Retract motion (RD ...) by +5 mm in the secondary cutting direction (X in this case). |
| `G91 G33 Z–20 K1` | Incremental programming on (G91).<br>Tapping (G33). End point: incremental by –20 mm in Z direction.<br>Fixed thread pitch: 1 mm/rev. Interpolation parameter: K in this case. |

## 3.27 Additional tapping functions G533

Effect

Individual partial areas of G33 can be temporarily adjusted by programming G533.
In this case, the control unit superimposes the static values stored in MACODA.

G533 provides for

- the adjustment of the dynamics and retraction movement
- change-over of the spindle mode (speed control, position control)
- specification of the channel IF signal NC-O20.4, ”Tapping cycle active”.

Control reset or M30

- cancels the settings superimposed by G533
- clears an IF signal that had been set by G533
- switches the main spindle back to speed-controlled operation, if it had previously changed over to position-controlled spindle operation by ”G533 SPC1”.

☞ **All partial functions described below can be jointly programmed in a single G533 block.**

Programming

**Configuring the retract data:**

**G533 RD**(<MA value>**,**<SA value>{**,–1**})
where

<MA value> Retract path (incremental in mm) towards the main axis of the currently selected plane (G17, G18, G19, G20).
The value always has to be programmed, however, it is only relevant for longitudinal and tapered threads.

<SA value> Retract path (incremental in mm) towards the secondary axis of the currently selected plane.
The value always has to be programmed, however, it is only relevant for transversal and tapered threads.

–1 ”–1” is an optional third parameter.
In this case, the retract data from MP 7050 00645 and MP 7050 00650 will be active again.

**Activating retraction:**

**G533 RON**<Status>
where

<Status> 0: Deactivate fast retract.
1: Activate fast retract.

### Configuring the dynamics:

**G533 DYN**({<Jump>},{<Accel>}{,<Decel>})
where

| | |
|---|---|
| <Jump> | max. permitted jump speed in mm/min<br>Entering ”–1” will activate MP 7050 00610 again. |
| <Accel> | Acceleration in m/$s^2$<br>Entering ”–1” will activate MP 7050 00615 again. |
| <Decel> | Deceleration in m/$s^2$<br>Entering ”–1” will activate MP 7050 00620 again. |

### Changing over the spindle mode:

**G533 SPC**<Status>
where

<Status>

0:
Switching the main spindle into speed-controlled mode.

1:
Switching the main spindle into position-controlled mode, depending on the setting of 7050 00600 [3].

For details on main spindles, refer to page 4–17.

### Influencing the channel IF signal ”Tapping cycle active”:

**G533 TCI**<Status>
where

| | |
|---|---|
| <Status> | 0: clears the channel IF signal NC-O20.4 |
| | 1: sets the channel IF signal NC-O20.4 |

## 3.28 Corner rounding G34, G35, G36, G134

Effect

The ”corner rounding” function inserts tangential transition arcs between 2 linear blocks (G34, G134) and between circular and helical blocks (G134 only) in the principal plane. On the one side, this leads to a minor modification of the programmed contour at these corners, but on the other side, continuous speed and acceleration patterns are achieved during interpolation (refer to “PNC Description of Functions” manual).

Programming

| | |
|---|---|
| G34 | Switch on ”corner rounding” with max. admissible deviations |
| G35 | Switch off ”corner rounding” |
| Address E | The E word is used to program the ”maximum permitted deviation” (in mm) between the modified contour and the programmed values. Fractional parts are allowed. Programming ”E” is only possible if G34 is active. |
| G36 | Deletes a max. admissible deviation programmed via E word. The value activated in MACODA parameter 7050 00110 becomes active again. |
| G134 | Switch ”corner rounding” on with specification of the rounding radius. |
| R address: | You use an R word to program the radius of the transition arc. Fractional parts are allowed. ”R” has to be programmed together with G134 in one and the same block. |

**Please note for G34, G134:**

- the radius is programmed together with G134 in one block,
- the radius acts modally,
- in the case of helical blocks, only the components of the circular plane for rounding are taken into account.
- G34, G134, G234 (for chamfer programming, refer to Section 3.29) form a group.
- The functions G34, G134, G234 and G35 deselect each other, with G35 deselecting any type of insertion of transition segments.

With active G34, the control unit **will not** execute ”corner rounding” if:

- at least one of the two neighbouring blocks is no linear block.

With active G34, G134, the control unit **will not** execute ”corner rounding” if:

- at least one of the two neighbouring blocks has a path portion outside the selected principal plane, or
- at least one of the two neighbouring blocks has a traversing path which is smaller than the path set in MACODA parameter 7050 00120 (2 to 90 μm, default value: 2 μm), or
- no ”quasi-continual” block transition according to MACODA parameter 7050 00130 is present, i.e. the angle between the two blocks is greater than the value in 7050 00130 stated as maximum angle (default = 1).

## 3.29 Chamfer programming G234, G35

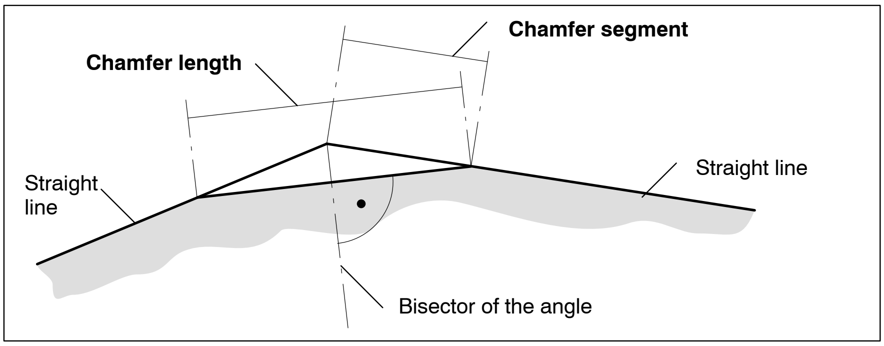
Effect

The function “chamfer programming” inserts a transition phase between two consecutive NC blocks of the type straight line or circle, the length of which can be specified as absolute **chamfer length** or as **length of the chamfer segment**. The chamfer is generated within the active working plane.

**The following chamfer transitions are possible**

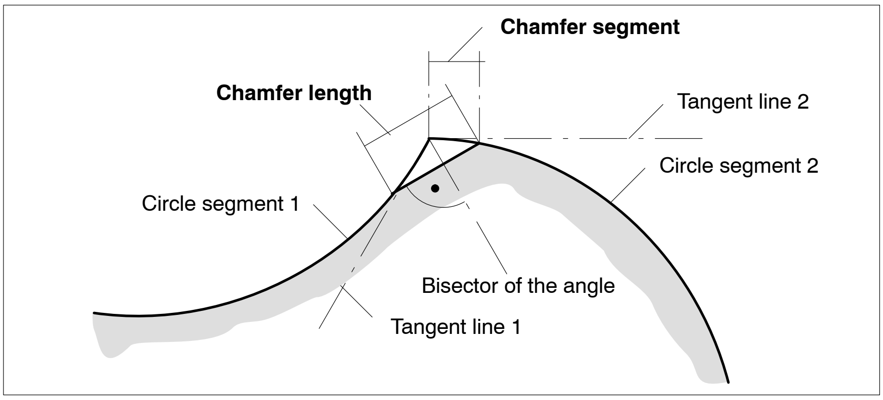
- Chamfer between two abutting straight lines:

  The chamfer runs at a right angle to the bisector between neighboring path segments. The length of the chamfer is automatically corrected (reduced) when there is no intersection with the neighboring programmed path segments.



- Chamfer between two abutting circle segments:

  In case of contour transitions involving circle segments, the dimensions of the chamfers refer to the respective end or starting tangent of the path segments involved in the contour transition. The actual resulting chamfer length is strongly dependent, among others, on the radii of the circles involved and thus deviates more or less from the programmed dimensions.



**Correction of the chamfer length in case of non-existing intersections**

Reasons why intersections with the neighboring contour segments do not exist due to the programmed geometry:

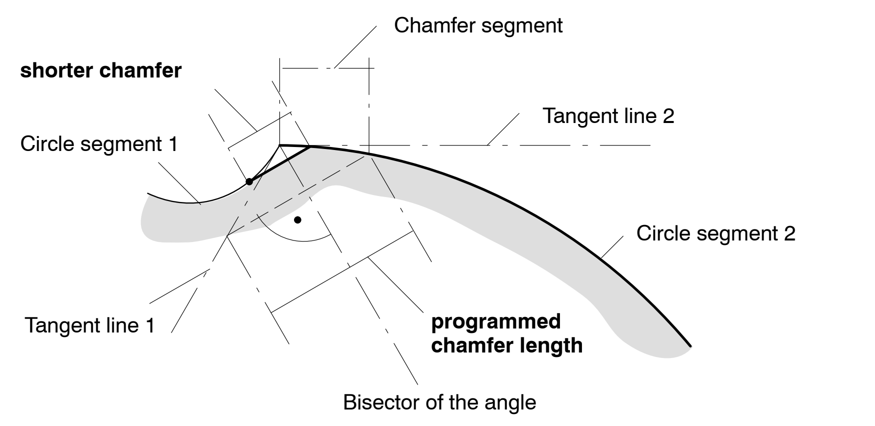
- Programmed chamfer length too long.

- Path length of the neighboring programmed path segments too short.
- Radius of the neighboring path segment is too small in relation to the chamfer length.

If intersections between the neighboring contour segments do not exist, the specified length of the chamfer is automatically reduced to the point where it can be inserted between the contour segments. The orientation of the chamfer is always perpendicular to the bisector between the two neighboring path segments.

If more than one-half of a programmed path segment would be cut off due to the reduced chamfer length, the chamfer length needs to be shortened. In this event, the following conditions are applicable:

- The transition from the previous path segment to the chamfer takes place at the earliest after one-half of the path of the **previous** path segment.
- The transition from the chamfer to the following path segment takes place at the latest after half of the path of the **following** path segment.



Programming G234... CHL<Chamfer length>|CHR<Chamfer length>|CHF<Chamfer segment>

| | |
|---|---|
| G234... CHL..\|CHR..\|CHF.. | Switch on ”chamfer programming” |

where

| | |
|---|---|
| CHL<Chamfer length> | Chamfer length (**Ch**amfer **L**ength) in mm (G71) or inches (G70) |
| CHR<Chamfer length> | alternative to CHL in unit mm (G71) or inches (G70), respectively. |
| CHF<Chamfer segment> | Chamfer segment alternative to CHL in unit mm (G71) or inches (G70), respectively. |

Programming G35 Switch off “chamfer programming” or “corner rounding” (refer to Section 3.28).

**Please note for G234, G35:**

- G234 has a modal effect
- G234 forms a group together with the two types of corner rounding G34 and G134.
- The functions G34, G134, G234 and G35 deselect each other, with G35 deselecting any type of insertion of transition segments.
- The chamfer length/chamfer segment parameters CHL, CHR or CHF have to be programmed together with G234 in one and the same block.
- The chamfer exclusively refers to the active working plane (G17, G18, G19, G20). If additional axes are involved in the movement, the chamfers are not influenced by this. As the coordinates of the programmed traversing blocks of the axes within the working plane are manipulated by the chamfers, but the values for the axes outside the working plane remain unchanged, the direction of straight lines in space may change, for example.
- The function is only effective in the ”Automatic” operating mode under automatic, single block and single step. Since the “program block” behaves like a manual input, chamfer programming is not effective here.
- The switch on/off procedure as well as the behavior upon control reset is exclusively determined by the two entries for the init strings in MACODA parameters 7060 00010 and 7060 00020.

## 3.30 Mirroring, scaling, rotating G37, G38, G39

Effect

**Mirroring:**

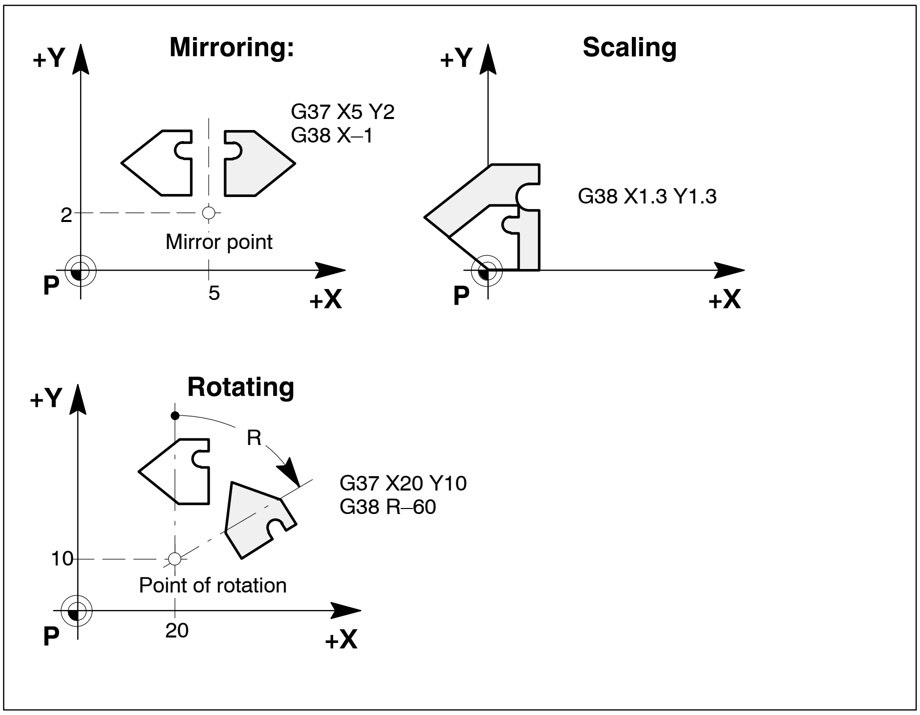
The control unit mirrors a programmed contour for machining. You do not need to change the programmed contour for this purpose.

**Scaling:**

The control unit scales the programmed contour up or down for machining. You need not change the programmed contour for this purpose.

**Rotating:**

The control unit rotates a programmed contour for machining. You need not change the programmed contour for this purpose.



The named functions can also be combined.

Programming

You can influence the **mirroring**, **scaling** and **rotating** functions jointly via the G37, G38 and G39 functions:

| | |
|---|---|
| G37 | Determination of the mirror or rotation point |
| G38 | Activate the mirroring, scaling or rotating function |
| G39 | Deactivate the mirroring, scaling or rotating function |

**CAUTION**
**The functions act only in the automatic, single-block and manual-input modes. The traversing direction during jogging will not change with active G38.**

For detailed explanations, please refer to the sections below.

## 3.30.1 Mirroring G37, G38, G39

Effect

The control unit will machine a programmed contour or, for instance, a bore-hole pattern in the form of a mirror image.
The ”scaling” and ”rotating” functions can be used simultaneously with ”mirroring”.

Programming

**G37 Pole definition (special case)**.
This is used to determine the position of the ”mirror point” for G38. This position has to be input as an absolute pair of coordinates referring to the program zero point.

G37 is **not** required if

- mirroring is supposed to be referring to the program zero point
- rotating is supposed to be referring to the program zero point (please refer to ”Rotating”).

The G37 function:

- acts modally The pole values remain effective until G39 or G37 is programmed (with other pole values)
- is only effective in conjunction with G38
- does not cause any axis traversing
- may be programmed together with other preparatory functions within the same block; auxiliary functions are allowed
- is not influenced by the factors programmed in G38 (in the case of the ”scaling” function) and by the angle of rotation (in the case of the ”rotating” function).

**Example:**

```
N...   G37   X100   Y–200
```

Pole coordinates
Calling the pole definition

Programming

**G38 Mirroring on.**
You switch ”mirroring” on by programming axis addresses (e.g. X) with the value of ”**–1**” in the same block as G38. By doing so you instruct the control unit to multiply all subsequently programmed path commands of the corresponding axis (e.g. X100) internally by the value ”–1”.
This means that the ”mirroring” function is exclusively realized by the minus sign. If you specify a value other than ”1”, you will furthermore change the size of the mirrored contour (please refer to ”Scaling”).
Mirroring will become effective together with the next traversing information.

Programming

**G38 O(<Sx>,<Sy>,<Sz>) Mirroring of an orientation vector**

An orientation vector is mirrored by components. Scaling or a pole determination have no influence.

where:

<Sx>,<Sy>,<Sz> Mirroring factors Sx, Sy, and Sz:
+1: no mirroring
–1: mirroring

Please note for the orientation vector:

- The polar coordinates φ and ϑ proper cannot be mirrored.
- Programming “G38 phi–1 theta–1” is not allowed.

The G38 function:

- acts modally. It remains active until G39 is programmed.
- must always be written into the same block as the axes to be mirrored
- may be written with other preparatory functions in the same block
- may be programmed with auxiliary functions.
- takes account of interpolation parameters in the case of circular interpolation
- influences the programmable contour shift G60
- does **not** affect the zero shifts G54–G259, G92 (set actual value) or cutter-radius and tool-length compensation.

**Example:**

N... G38 X–1 Y–1

X–1 Y–1: Any subsequently programmed axis values for the X and Y axis will be multiplied by the value of ”–1” within the control unit.

G38: Mirroring on (by next traversing motion)

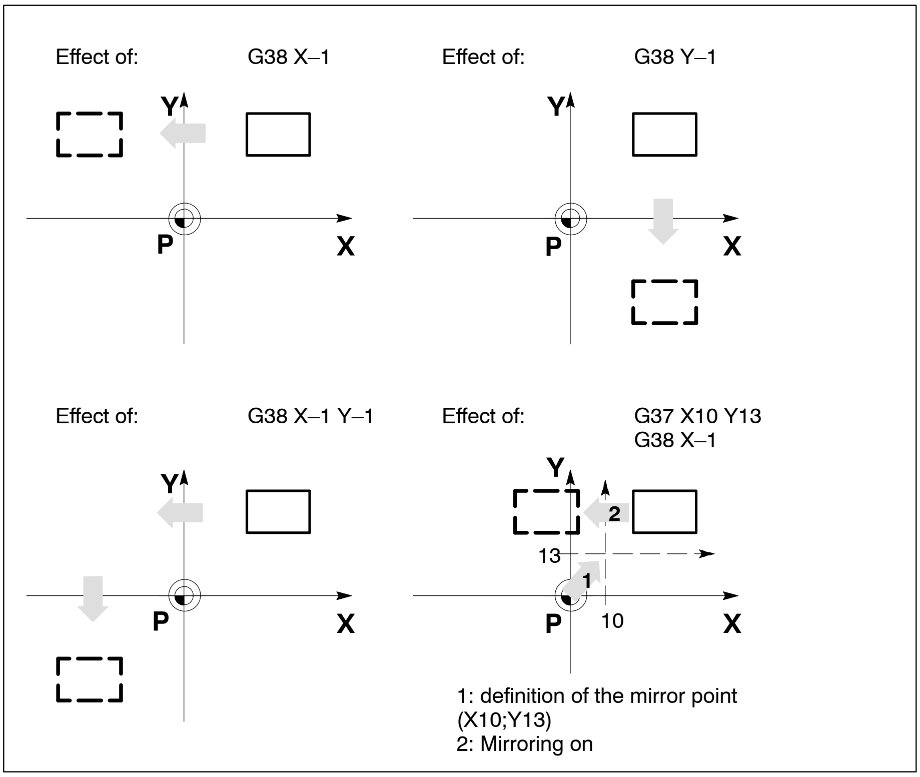
Programming

**G39 Switch mirroring, scaling, rotating off**.
Any subsequently programmed axis values will no longer be multiplied by the value of ”–1” within the control unit.
Approached axis positions are retained until re-programmed.

The G39 function:

- acts modally.
- deletes all mirror axes.
- deletes G37 and G38 and sets the pole coordinates to the value of ”0”.
- can be written into a block with preparatory functions, traverse information and auxiliary functions.

## Mirroring examples:



## 3.30.2 Scaling G38, G39

Effect

The control unit scales a programmed contour up or down for machining.

This can be used in part programs for programming contours using one fixed size (standard size). Then, prior to calling such a part program (e.g. as a subprogram), you use scaling factors for each axis to determine the scale of the programmed contour.
In this way it is, for instance, easy to compensate for the contraction of the workpieces in the manufacture of the moulds for cast and forged parts.

The ”scaling” function can be used together with ”mirroring” and ”rotating”.

- Scaling does not influence feed programming or the active feedrate.
- M2/M30 in a subprogram does not switch scaling off.

Particularities in the case of circular interpolation

In principle, different scaling factors can be stated for each axis. However, if you wish to use circular interpolation (or helical interpolation) with active scaling, the scaling factors must be the same for all axes involved! Otherwise an error message will be generated.

Scaling factors will also change the I, J, K interpolation parameters as well as the amount of the R address (for radius programming).

Interaction of functions

| | |
|---|---|
| ● **G0, G1, G2, G3, G5, G10, G11, G12, G13, G73, G200** | Scaling acts in combination with the programmed axis information. |
| ● **G20** | Scaling acts in combination with the programmed axis information. |
| ● **G37** | Pole values will not be scaled. |
| ● **G40, G41, G42, G43, G44** | Scaling is active; compensation values will not be scaled. |
| ● **G54–G59, G154–G159, G254–G259** | Zero-shift values will not be scaled. |
| ● **G60** | The programmable contour-shift values will be scaled. |
| ● **G70,G71** | Scaling acts independently of the active unit of measurement. |
| ● **G74,G76** | Scaling is not active. |
| ● **G90,G91** | Scaling acts with absolute and incremental data input. |
| ● **G92** | Offset will not be scaled. |

Programming

If the starting position of the workpiece contour is not to be influenced by a scale- down or scale-up of the programmed contour, you should select the starting point of the workpiece contour to be the program zero point, too.

`G38` Scaling on.
You switch ”scaling” on by programming axis addresses (e.g. X) with a positive factor in the same block as G38. By doing so you instruct the control unit to multiply all subsequently programmed path commands of the corresponding axis (e.g. X10) internally by this value.
If you specify a value other than ”1”, you will change the size of the contour:
Factor> 1 : the contour will be scaled up.
Factor< 1 : the contour will be scaled down.
If you program the factor with a negative sign, you switch on the ”mirroring” function in addition.

The G38 function:

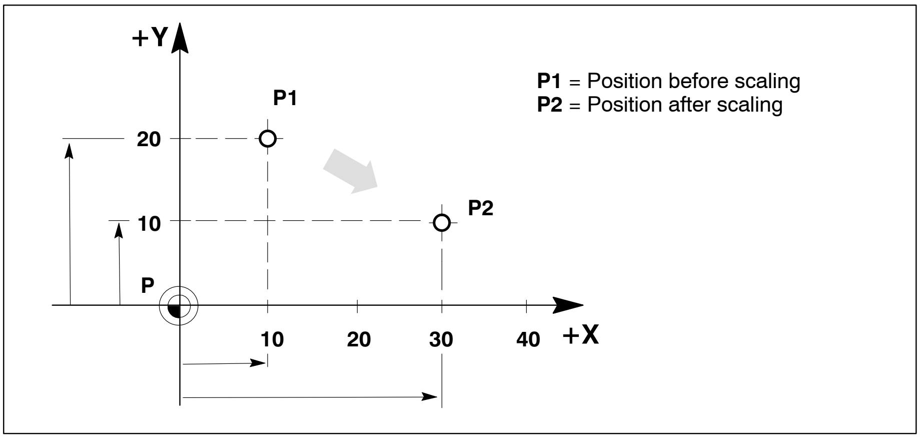
- acts modally. It remains active until G39 is programmed.
- always has to be written together with the axes to be scaled within one block.
- does not cause any axis traversing.
- may be written with other preparatory functions in the same block.
- may be programmed with auxiliary functions.

**Example:**

N... G38 X3 Y0.5

X3 Y0.5: All X coordinates programmed subsequently are multiplied by ”3”, and the Y coordinates by ”0.5”.

G38: Scaling on (upon next traversing motion)



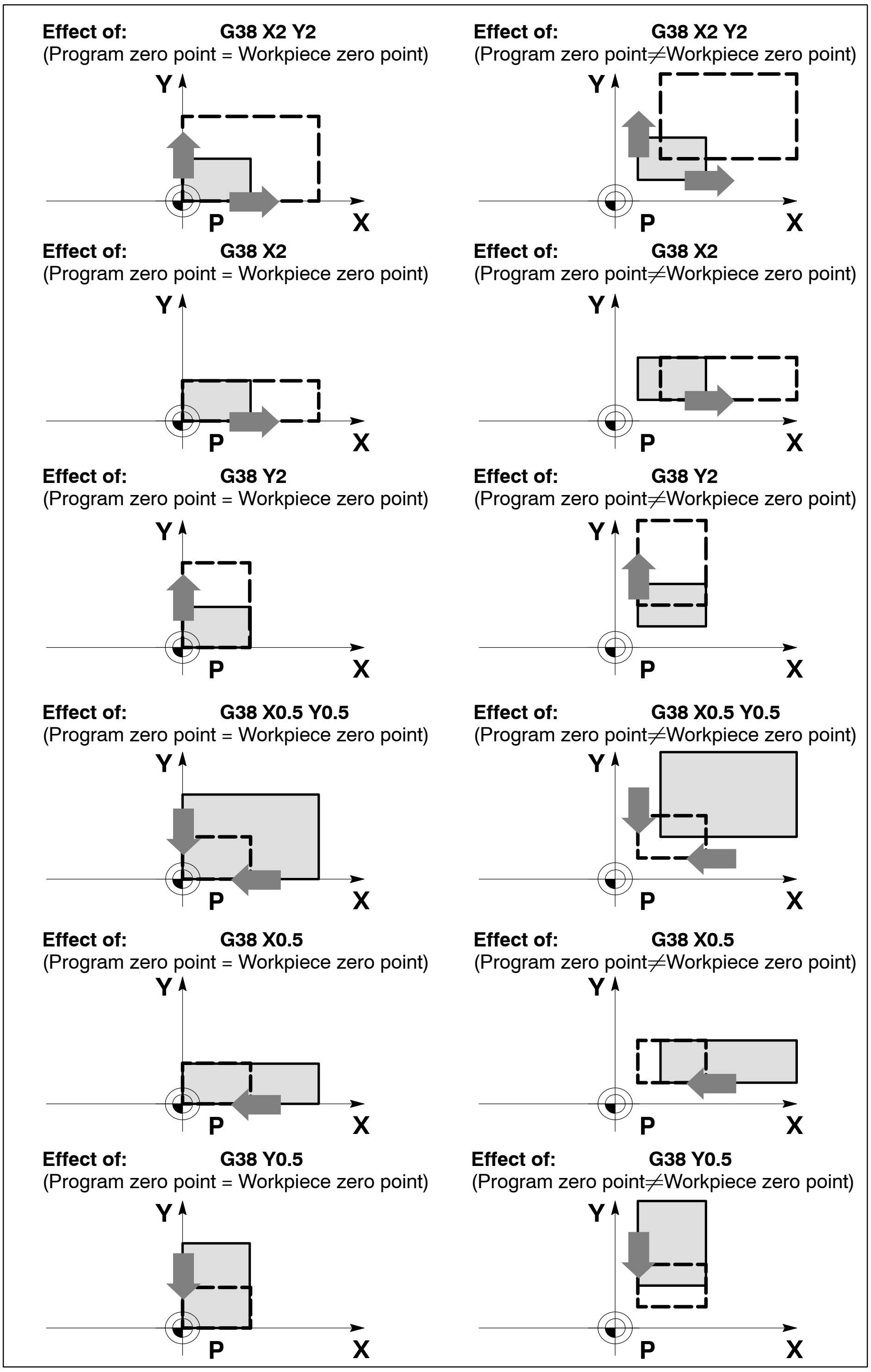
Programming

G39: Switch mirroring, scaling, rotating off.
Any subsequently programmed axis values will no longer be influenced.
Approached axis positions are retained until re-programmed.

The G39 function:

- acts modally.
- deletes all mirror axes.
- deletes G38 and sets the internal scaling factors to the value of ”1”.
- can be written into a block with preparatory functions, traverse information and auxiliary functions.

**Scaling examples:**


Effect of: G38 X2 Y2
(Program zero point = Workpiece zero point)
Effect of: G38 X2 Y2
(Program zero point≠Workpiece zero point)
Effect of: G38 X2
(Program zero point = Workpiece zero point)
Effect of: G38 X2
(Program zero point≠Workpiece zero point)
Effect of: G38 Y2
(Program zero point = Workpiece zero point)
Effect of: G38 Y2
(Program zero point≠Workpiece zero point)
Effect of: G38 X0.5 Y0.5
(Program zero point = Workpiece zero point)
Effect of: G38 X0.5 Y0.5
(Program zero point≠Workpiece zero point)
Effect of: G38 X0.5
(Program zero point = Workpiece zero point)
Effect of: G38 X0.5
(Program zero point≠Workpiece zero point)
Effect of: G38 Y0.5
(Program zero point = Workpiece zero point)
Effect of: G38 Y0.5
(Program zero point≠Workpiece zero point)
Y
X
P

## 3.30.3 Rotating G37, G38, G39

Effect

The control unit rotates a programmed contour in the active plane (please refer to G17, G18, G19 or G20).

This means that you must program only once such recurrent programming steps as are rotated around a specific angle.
In addition you do not have to convert the dimensions of angled workpieces to the machine coordinates; you simply take them over directly from a production drawing and specify the corresponding angle of rotation. The PNC will do the rest.

”Scaling” and ”mirroring” can be used in conjunction with the ”rotating” function.

Programming

**G37** **Pole definition.**
This is used to determine the position of the ”point of rotation” for G38. This position has to be input as an absolute pair of coordinates referring to the program zero point.

G37 is **not** required if

- rotating is supposed to be referring to the program zero point
- mirroring is supposed to be referring to the program zero point (please refer to ”Mirroring”).

The G37 function:

- acts modally. The pole values remain effective until G39 or G37 (with other pole values) is programmed
- is only effective in conjunction with G38
- does not cause any axis traversing.
- may be programmed together with other preparatory functions within the same block; auxiliary functions are allowed
- is not influenced by scaling factors programmed in G38 (in the case of the ”scaling” function) or their sign (in the case of the ”mirroring” function).

**Example:**

N...G17 G37 X200 Y100

X200 Y100: Pole coordinates

G37: Calling the pole definition

G17: Plane selection (may be omitted if rotating is to be performed in the active plane)

Programming

**G38** **Rotating on.**

Program the ”R” address with the desired angle of rotation in the same block as G38.

- Positive values: Counter-clockwise rotation
- Negative values: Clockwise rotation.

By doing so you instruct the control unit to rotate all subsequently programmed coordinates of the corresponding plane around the point of rotation (please refer to G37).
The rotation becomes active with the next traversing information. A programmed contour shift (G60) will be included in the calculation of the coordinate rotation

Programming

**G38 R<Angle>** **Rotating an orientation vector**

an orientation vector is rotated by the normal of the selected plane. Scaling or a pole determination have no influence.

where:

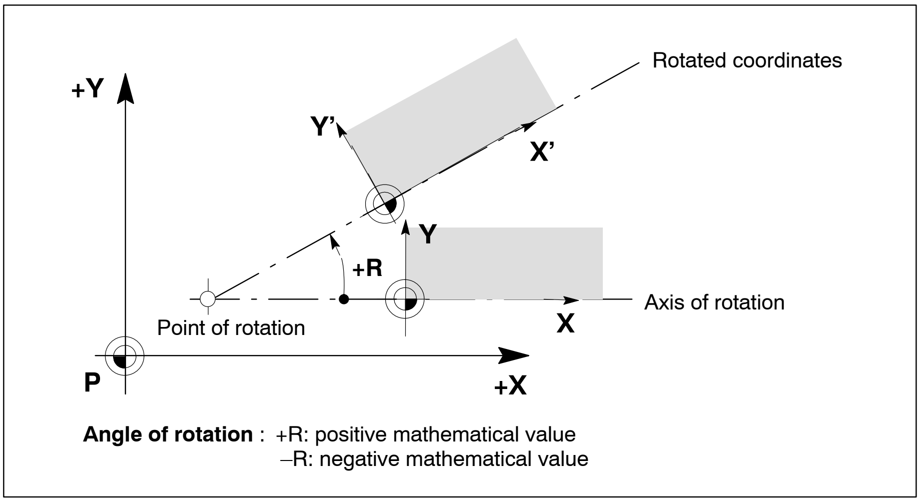
| | |
|---|---|
| <Angle> | Angle of rotation around the normal of the selected plane. |

The G38 function:

- acts modally. It remains active until G39 is programmed.
- must always be programmed with angle of rotation R in the same block.
- may be written with other preparatory functions in the same block.
- may be programmed with auxiliary functions.

**Example:**

```
N...    G38   R+30
        ‾‾‾     ‾‾
         |      |______________ Angle of rotation
         |_____________________ Rotating on
```



**Angle of rotation** : +R: positive mathematical value
–R: negative mathematical value

Programming

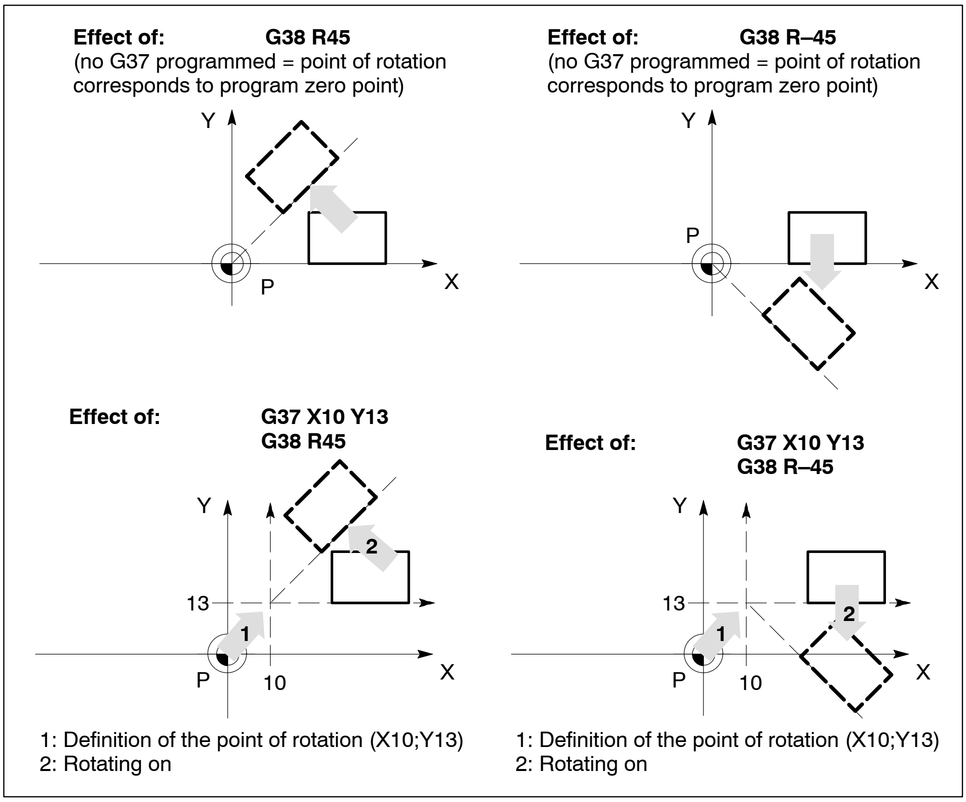
**G39** **Switch mirroring, scaling, rotating off.**

Any subsequently programmed coordinates will no longer be rotated.
Approached axis positions are retained until re-programmed.

The G39 function:

- acts modally.
- deletes G37 and sets the angle of rotation and the coordinates of the point of rotation to the value of ”0”
- can be written into a block with preparatory functions, traverse information and auxiliary functions.

**Rotation examples:**



**Effect of:** **G38 R45**
(no G37 programmed = point of rotation corresponds to program zero point)

**Effect of:** **G38 R–45**
(no G37 programmed = point of rotation corresponds to program zero point)

**Effect of:** **G37 X10 Y13**
**G38 R45**

**Effect of:** **G37 X10 Y13**
**G38 R–45**

1: Definition of the point of rotation (X10;Y13)
2: Rotating on

1: Definition of the point of rotation (X10;Y13)
2: Rotating on

## 3.30.4 Combining mirroring, scaling and rotating

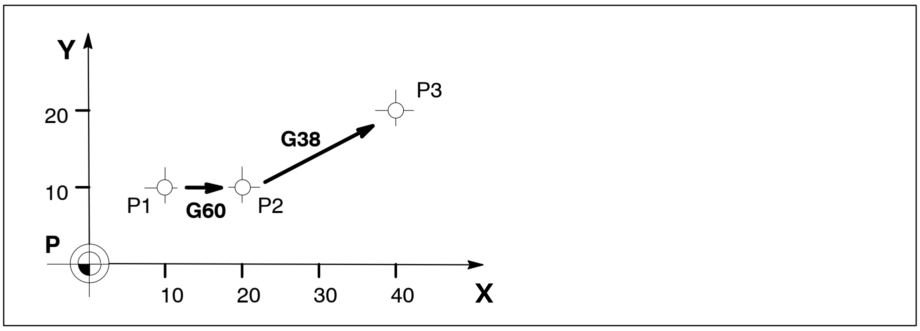
☞ **If rotating and mirroring or scaling are programmed simultaneously, rotating will be executed first, followed by mirroring or scaling.**

**Example:** Rotating + Mirroring + Scaling

| | |
|---|---|
| `N... G37 X100 Y-200` | determination of point of rotation and mirror point |
| `N... G38 X-3 Y-2 R115`<br>`...` | counter-clockwise rotation by 115 degrees; mirroring produced by the minus sign, and multiplication of the X and Y coordinates by ”3” or ”2”) |
| `N... G39` | (switch all off) |

## 3.30.5 Relationship between G37/G38 and G60 or G54..G259

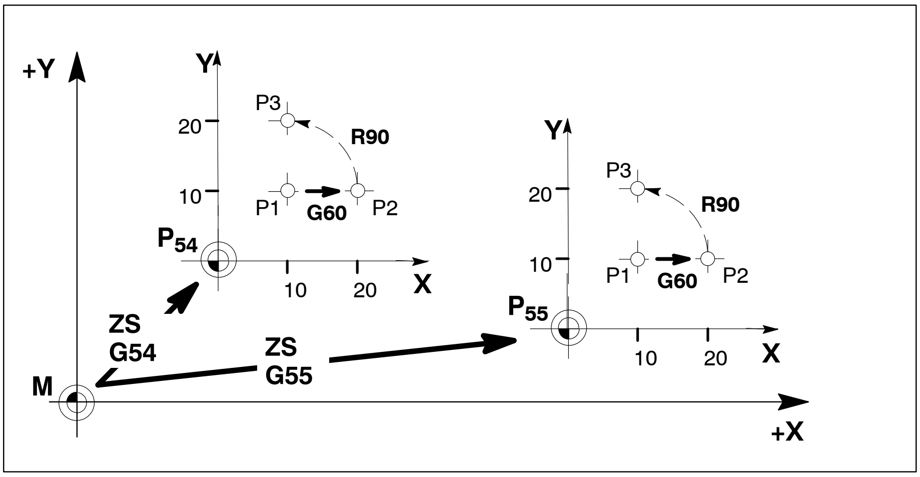
**Within the program coordinate system**, G37/G38 is influenced by G60:



**Example**: G60

| | |
|---|---|
| `N5...` | P1: current position |
| `N10 G60` | P2: G60 shift of P1 |
| `N20 G38 X2 Y2` | Scaling ON |
| `N30 G1 X10 Y10` | P3: scaled position of P2 |

As a principle, zero shifts (e.g. G54..G259) will shift the **entire program coordinate system** with regard to the machine coordinate system. Therefore they do not cause any change to the operations within the program coordinate system, triggered, for instance, by G37/G38 or G60:



| Example: G54 | Example: G55 | Comment |
|---|---|---|
| `N10 G54` | `N110 G55` | Call-up function |
| `N20 G37 X10 Y10` | `N120 G37 X10 Y10` | P1: point of rotation on X10 Y10 |
| `N30 G60 X10` | `N130 G60 X10` | P2: G60 shift of P1 |
| `N40 G38 R90` | `N140 G38 R90` | P3: Coodinate rotation of P2 |
| `N50 G1 X10 Y10` | `N150 G1 X10 Y10` | |

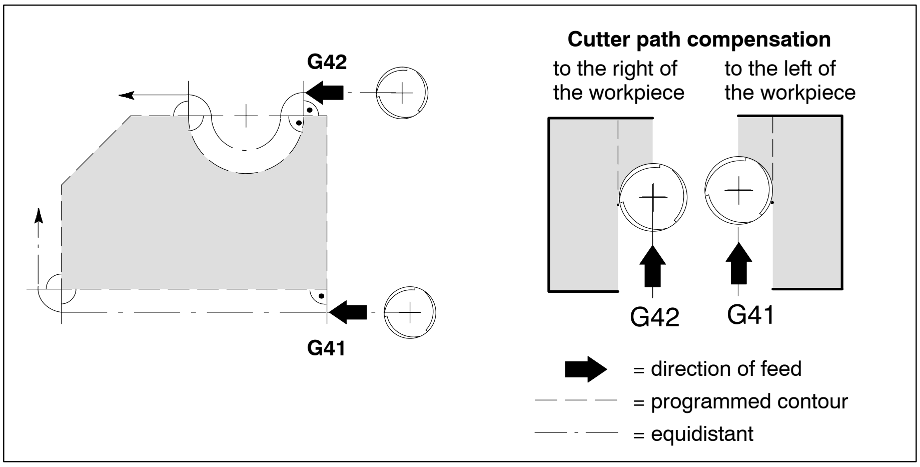
## 3.31 Cutter path compensation G40, G41, G42

☞ **The full functionality of the cutter path compensation includes:**
**– G40, G41, G42**
**– G68, G69**
**– G500, G543, G544**
**– G64, G65.**

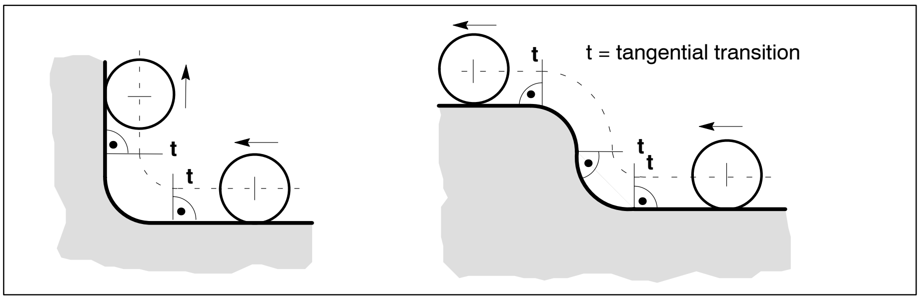
Effect

The cutter path compensation function causes the tool to move along an equidistant path parallel to the programmed path during execution of a part-specific program. (Equidistant path = path with right-angled, constant distance from the programmed contour.) The distance between the equidistant and the programmed path depends on the value of cutter-path compensation.

The following illustration shows the principle:



Along a contour element and in the case of tangential contour transitions, the equidistant – and thus the cutter path – is uniquely defined by the programmed contour:

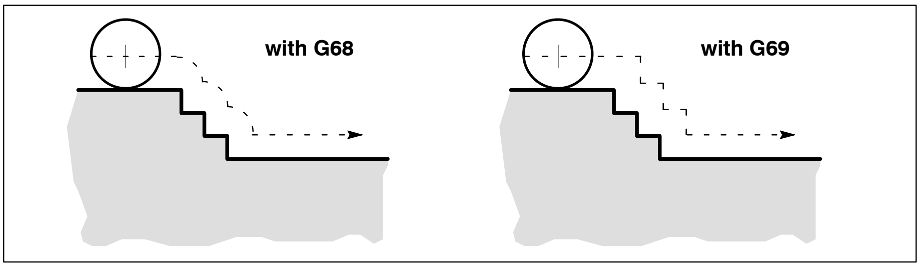


At unsteady contour transitions the control unit has to calculate a path independently in order to combine the equidistants of the contour elements involved.

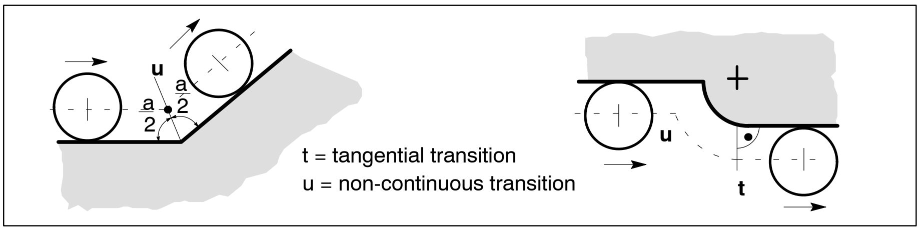
The following functions are available for this purpose for the contour transition **at outer corners** (refer to page 3–80):

- G68 (contour transition on circular arc) and

- G69 (contour transition through intersection of the equidistants)



At unsteady contour transitions in the case of **inner corners** the control unit will use the intersection of the equidistants to determine the required path.



With some contour patterns (e.g. indentations) this principle may lead to contour damage.

Therefore, for contour transition **at inner corners** the function

- ”Collision monitoring” (G543)

is available (refer to page 3–184).

Programming

**CAUTION**
**Compensation values may be immediately activated or deactivated without a traversing movement. This may result in damages of the workpiece or the tool.**
**Please note the information provided in this section in this context!**

☞ **With active G2, G3 or G5, functions G40, G41 or G42 may only be programmed without any traversing movement.**

G40 Cutter-path compensation off (power-up state).
If no traversing movement is programmed in the G40 block, the control unit will deactivate the cutter path compensation in position – vertically to the last traversing block!
If a traversing movement is programmed in the G40 block, the control unit will deactivate the cutter path compensation linearly while traveling to the end point of the traversing movement.

G41 Activating the cutter path compensation **to the left of the workpiece** (seen in the processing direction, if positive compensation values are used).
In addition to a D address containing the required radius compensation value, a linear traversing movement of the axes of the active plane may be programmed in the same block as G41. This will select the compensation while traveling to the end of the traversing movement.
If no traversing movement is programmed in the G41 block, the control unit will deactivate the cutter path compensation immediately – vertically to the last traversing block.

G42 Switching cutter-path compensation on **to the right of the workpiece**. Apart from that, identical to G41.

☞ **An active tool-length compensation will not be influenced by G40.**

**CAUTION**
**Possibility of damaging the workpiece or the machine!**
**The following is not allowed with active G41 or G42:**

- **Plane change-over (G17 to G20)**
- **Inch/metric change-over (G70, G71)**
- **G32, G74, G75, G76, G92**

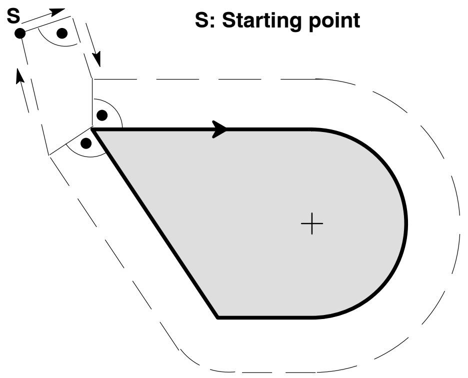
## Selection of cutter path compensation

In many cases, it is not possible to approach the contour directly from the tool change point. In most cases, machining is started from an intermediate position (cf. examples).

The selection of a suitable starting point helps avoid damage to the contour.

- The starting point should facilitate a tangential approach to the contour
- The starting point should be positioned in such a way that the direction of an axis does not change (relief cutting) at the first contour point.

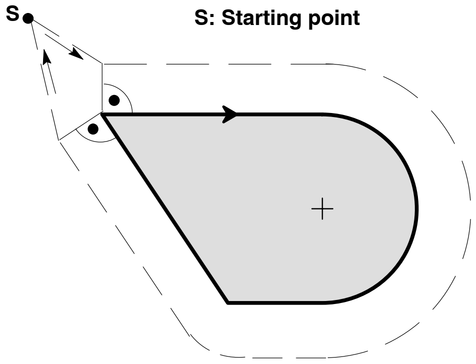
**Example:** Selecting the cutter path compensation with G41 without traversing movement:



S: Starting point

```
N1   G0    Z–10  T01M6
N2   G1    X–20  Y20F200
N3   S500  M3
N4   G41
N5   X0    Y0
N6   X20
N7   G2    Y–20  R20
N8   G1    X10
N9   X0    Y0
N10  G40   X–20  Y20
N11        Z300  T00
N12  M2
```

**Example:** Selecting the cutter path compensation with G41 with linear traversing movement:



S: Starting point

```
N1   G0    Z–10  T01M6
N2   G1    X–20  Y20F200
N3   G41   X0    Y0  S500  M3
N4   X20
N5   G2    Y–20  R20
N6   G1    X10
N7   X0    Y0
N8   G40   X–20  Y20
N9   Z300  T00
N10  M2
```

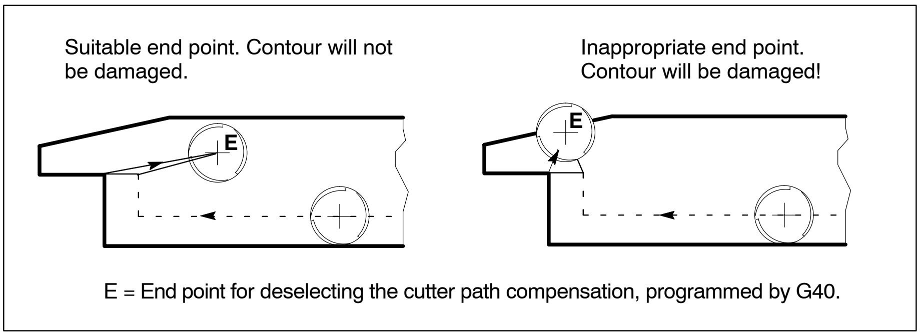
## Deselection of the cutter path compensation

As a rule, the tool does not directly move from the contour to the tool change point but rather via an intermediate position (end point).

The selection of a suitable starting point helps avoid damage to the contour. Therefore:

- The end point should facilitate a tangential contour departure when the radius compensation function is active.
- The end point should be positioned in such a way that relief cutting does not take place due to a change in direction when the tool moves away from the contour.

**CAUTION**
**Damage to the contour possible with inside contours!**
**If the control unit is in a circular mode (G2, G3, G5), no traversing movement may be programmed in the G40 block.**
**Consequently, the control unit will deselect the cutter path compensation in position, vertically to the last traversing block.**
**This may cause damage to the contour.**
**Therefore, before deactivating the cutter path compensation, you should move out of the inside contour (e.g. in Z direction), or program a suitable end point.**


Suitable end point. Contour will not be damaged.
E
Inappropriate end point. Contour will be damaged!
E

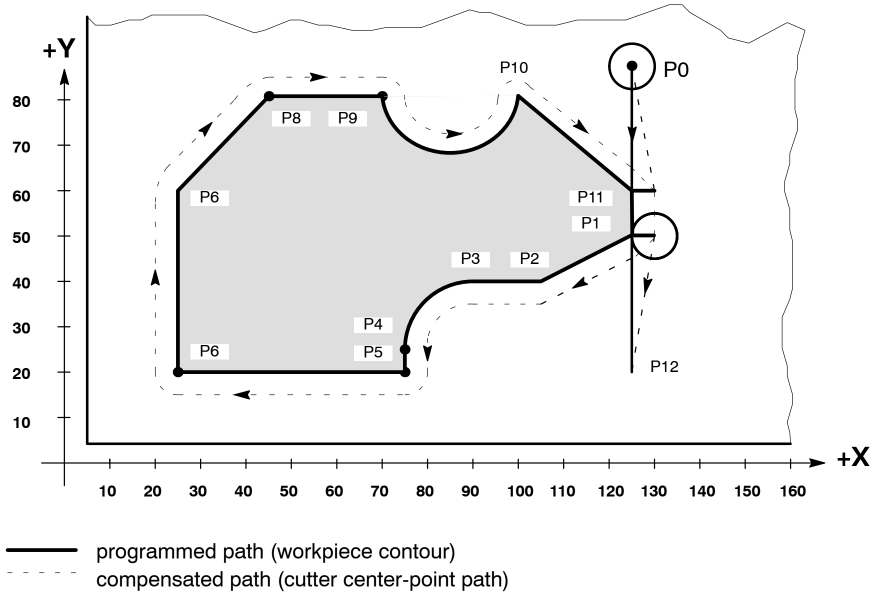
E = End point for deselecting the cutter path compensation, programmed by G40.

The following procedure is recommended (G41 active):

- last contour machining procedure (e.g. G2 active)
- tangential departure from contour with G1 (e.g. only X and Y programmed)
- relief cutting on Z... with G1 (e.g. program Z only)
- program G40 with X/Y movement in the extension of the last movement
- Z movement (e.g. program Z only)
- End of program

**Example 1:** Programming an outer contour with cutter path compensation.

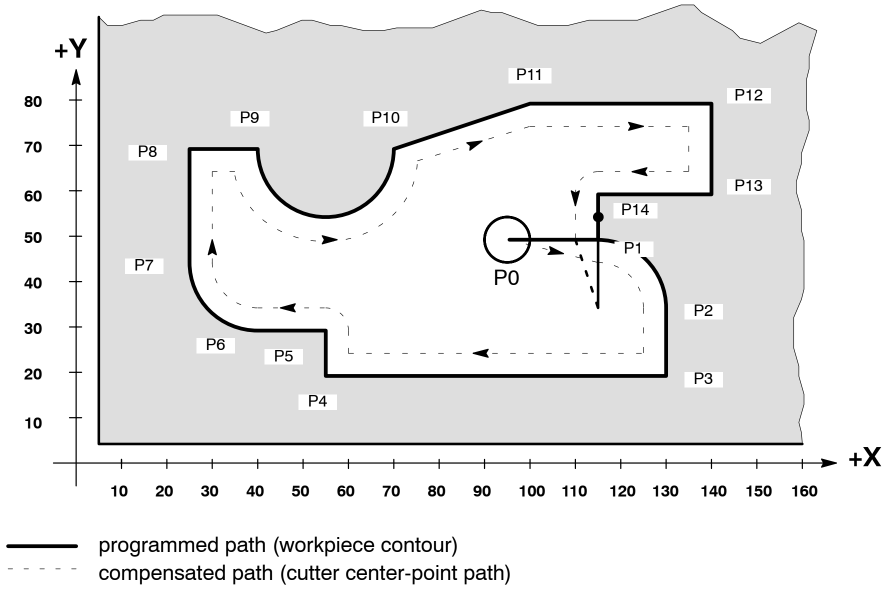
The cutter path compensation must be activated to the left of the workpiece (G41).
The respective geometry compensation table and the radius compensation value are already active.



```
N1    G1    F200
N2    G41   X125   Y60
N3                 Y50
N4          X105   Y40
N5          X90
N6    G5    X75    Y25
N7    G1           Y20
N8          X25
N9                 Y60
N10         X45    Y80
N11         X70
N12   G3    X100   R15
N13   G1    X125   Y60
N14                Y50
N15   G40          Y20
N16   M2
```

**Example 2:** Programming an inner contour with cutter path compensation.

The cutter path compensation must be activated to the right of the workpiece (G42).
The respective geometry compensation table and the radius compensation value are already active.



```
N2    G42   F300
N3    G1    X115   Y50
N4    G5    X130   Y35
N5    G1    Y20
N6          X55
N7                 Y30
N8          X40
N9    G5    X25    Y45
N10   G1           Y70
N11         X40
N12   G3    X70    R15
N13   G1    X100   Y80
N14         X140
N15                Y60
N16         X115
N17                Y50
N18   G40          Y35
N19   M2
```

## 3.32 Axis zero shift (ZS)
## Axis zero shift OFF G53
## Axis zero shift ON G54-G59
## 1st additive axis zero shift OFF G153
## 1st additive axis zero shift ON G154-G159
## 2nd additive axis zero shift OFF G253
## 2nd additive axis zero shift ON G254-G259

Effect

Using the axis ZS, it is possible to shift the program zero point to any point in reference to the machine coordinate system.

The corresponding distance values are stored in the axis zero-shift tables. Each zero-shift table contains a maximum of 3 groups with 6 axis zero shifts, each (G54..G59; G154..G159; G254..G259).

☞ **For details about editing the axis ZS tables, please refer to the operating instructions.**

To activate an axis ZS, you first select the desired axis ZS table (please refer to G22). Subsequently, you simply program the corresponding G instruction. Programming the G instruction without additional positional data will not result in a traversing movement.

Axis zero shifts **from different groups** always act additively to each other (e.g. G54 + G156 + G 259).

Axis zero shifts **within the individual groups** overwrite each other.

Programming

Axis zero shift

```
N... G22 V1            (activate axis ZS table V1)
N... G54               (axis ZS activated; no traversing movement)
(or)
N... G54  X...Y...Z..  (shift applies already to the position programmed
                       here)
N...
N... G154  X...Y...Z.. (1st add. axis ZS activated; with traversing move-
                       ment)
N...
N... G254  X...Y...Z.. (2nd add. axis ZS activated; with traversing move-
                       ment)
N...
N... G253              (only 2nd add. axis ZS off)
N...
N... G53               (all still active axis ZS off)
```

**Please note for G53, G54...G59:**

- G54 to G59 act modally and cancel each other mutually.
- They are switched off by G53.
- G53 switches the 1st and 2nd additive axis ZS off, **too**.
- G53 will **not** influence a ”programmed contour shift” (G60).

**Please note for G153, G154...G159:**

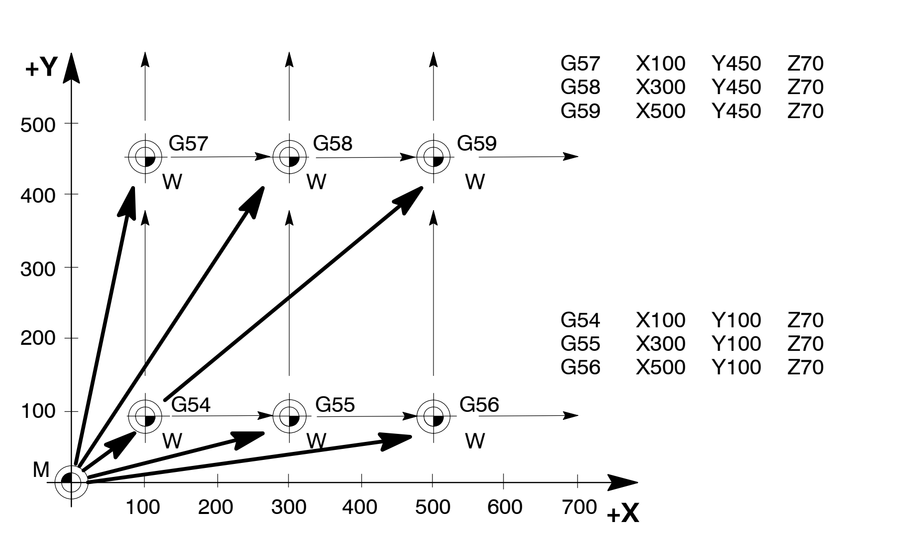
- G154 to G159 act modally and cancel each other mutually.
- They are switched off by G153 and G53.

**Please note for G253, G254...G259:**

- G254 to G259 act modally and cancel each other mutually.
- They are switched off by G253 and G53.

**Example:**

Axis ZS principle (in the example, the program and the workpiece zero point are identical)



☞ **The axis ZS function is not permitted in combination with the ”inclined plane” function.**

## 3.33 Programmed contour shift G60, G67

Effect

G60 does not shift the program coordinate system with regard to the machine coordinate system, but only the contour **within** the program coordinate system.

From this program block on, all programmed coordinates will be offset correspondingly. If no coordinate rotation is active (please refer to G38, ”rotating”), the programmed shift values act directly in addition to other compensations.
G60 does not cause any traversing movement.

**CAUTION**
**Incorrect programming may cause damage to the workpiece and the machine!**
**G60 is influenced by G38 (mirroring, scaling, rotating), i.e. the coordinates of the new zero point which were programmed in the G60 block will also be mirrored, scaled or rotated!**

Programming

| | |
|---|---|
| G60 | Programmed contour shift on |
| G67 | Programmed contour shift off |

**Example:** Programmed contour shift

```
N10 G60 X10 Y10 Z50      new program zero point at X10 Y10 Z50.
...                      The axes are not traversed in this block.
N100 G1 X... Y... Z...   traversing of axes with shift values taken
...                      into account
N210 G67 X... Y...       axis traversing without taking the G60 val-
Z...                     ues into account
...
or
N210 G67                 reset of G60
```

When reprogramming G60, any axis not reprogrammed in the process will retain its previously active contour shift values.

**Example:** Programmed contour shift
(repeated programming of G60)

```
N10 G60 X10 Y10 Z50      new program zero point at X10 Y10 Z50.
...                      The axes are not traversed in this block.
N100 G1 X... Y... Z...   traversing of axes with shift values taken
...                      into account
N110 G60 X20 Y20         new program zero point at X20 Y20 Z50 (Z
...                      shift is retained as programmed pre-
                         viously in G60!). The axes are not tra-
                         versed in this block.
N120 G1 X... Y... Z...   traversing of axes with shift values taken
...                      into account
N210 G67                 reset of G60
```

## 3.34 In-position logic G61, G62, G163

Effect

During the control of a tool movement, a time offset between the set and the actual values of the individual axes occurs during the movement owing to the dynamics of the machine. This ’lag effect’ causes a following distance error during machining, the size of which depends on the feedrate speed and the KV factor (axis dynamics). In the case of unsteady contour transitions (corners) this following distance error becomes noticeable in the form of a ”slurring” of the corner.

This effect can be avoided using G61. Functions G164 to G166 can be used to set 3 different In-position options.

G61 acts **only on movements at feedrate** (for In-position logic at rapid travel, please refer to G161/G162).

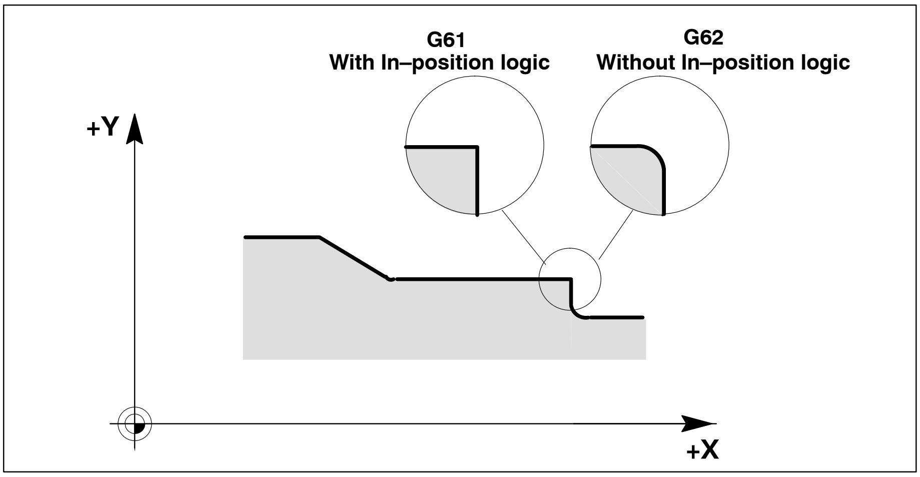
The action of G163 is overriding and affects **movements both at feedrate as well as at rapid** (G0, G200). G163 supersedes G161/G162.

Programming

| | |
|---|---|
| G61 | In-position logic at feedrate on. |
| G62 | In-position logic off. |
| G163 | Switch In-position logic on at feedrate and at rapid travel. |

Please note the following for G61, G62 and G163:

- G61, G62 and G163 act modally. M2/M30 sets the power-up state.
- G61, G62 or G163 must be programmed at the latest within the block to which the respective function applies.



**Example**: Programming of G61/G62

| | |
|---|---|
| `N10 G61` | no movement; In-position logic ON |
| `N11 G1 Y200` | interpolation with In-position logic |
| `(or)` | |
| `N10 G62` | interpolation without In-position logic |
| `N11 G1 Y200` | |
| `N50 G61 X200` | interpolation with In-position logic already in this block |

## 3.35 Feedrate 100% G63, G66

Effect

You use program control to influence the function of the feedrate potentiometer (for feedrate and rapid travel). Both functions are of effect in the ”manual data input” and ”automatic” operating modes.

☞ **Considering that the G63 function also sets the ”G63” output signal, the spindle potentiometer can be influenced as well. To do so, you must link the ”G63” output signal to the ”spindle override 100%” input signal.**

Programming

G63 feedrate 100% on.
Deactivates the feedrate potentiometer. The feedrate is set to 100% of the programmed value irrespective of the position of the feedrate potentiometer.

G66 feedrate 100% off.
Activates the feedrate potentiometer. The feedrate depends on the position of the feedrate potentiometer.

**Please note for G63 and G66:**

- Both functions are modal and cancel each other mutually.
- G63 and G66 can also be programmed together with other preparatory functions.
- M2/M30 sets the power-up condition.

**Example:** Programmed ”feedrate 100% ON”

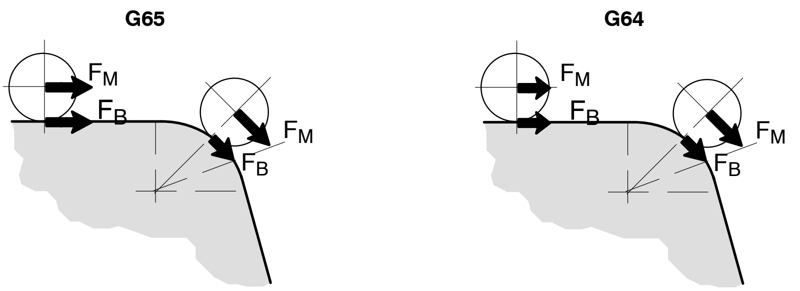
```
N... G63 G1 X120.675 Y34.896 Z-34.765 F200 S1000
M04
```

# 3.36 Feedrate compensation: Cutter contact point G64 Cutter center point G65

Effect

You define whether the PNC is supposed to keep the programmed feedrate constant

- either at the cutter contact point (cutter cutting path), or
- along the cutter center point path



$F_M$ = Feed along the center point path

$F_B$ = Feed along the cutting path

Programming

G64 The control unit keeps constant the feedrate $F_B$ along the cutting path. This calculation is only possible when **G41/G42 are active** for arcs G2/03/05.

G65 The control unit keeps constant the feedrate $F_M$ along the cutter center point path.

☞ **G64 should be used only for finish milling because the feedrate speed can increase considerably along circular contours.**

**Please note for G64 and G65:**

- G64 and G65 act modally and cancel each other mutually.
- The power-up state can be set via MACODA parameters.

Influence on compensating circles

The effective feedrate in the case of a **compensating circle** in combination with cutter-path compensation depends on the point where an F word is programmed:

**Example 1:**

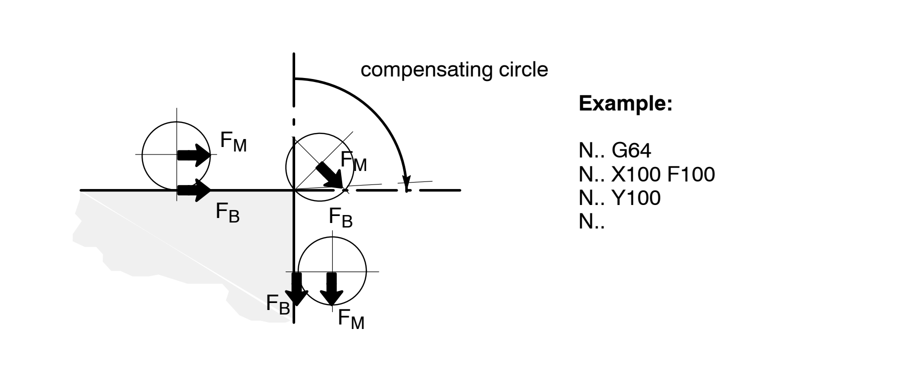
```
N10 G64 X100 F100
N20 Y100 F200         The compensating circle is traversed at F100
```

**Example 2:**

```
N10 G64 X100 F100
N20 F200
N30 Y100              The compensating circle is traversed at F200
```

**Example 3:**

```
N10 G64 X100 F100
N20 Z50
N30 Y100 F200         The compensating circle is traversed at F100
```



## 3.37 Contour transitions: Circular arc G68 Intersection point G69

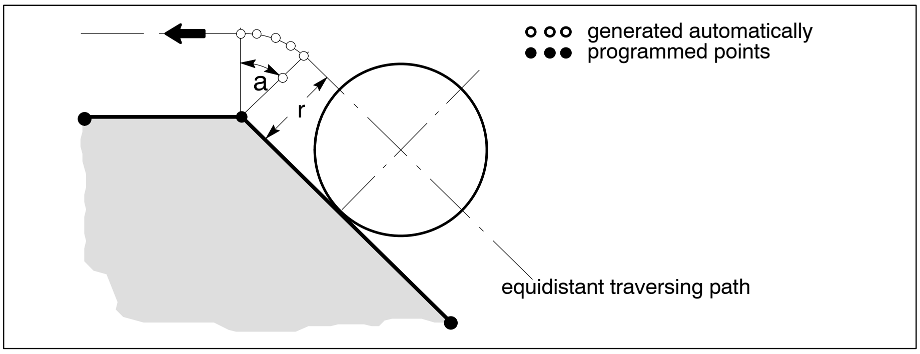
Effect

Function for active cutter-path compensation (G41, G42). In this case, the control unit realizes a transition between two contour elements **on outer corners**, optionally by an automatically created arc (G68) or by an intersection of the equidistants (G69).

☞ **The full functionality of the cutter path compensation includes:**
**G40, G41, G42**
**G68, G69**
**G500, G543, G544.**

**G68: Circular arc**

The path gap is closed by a tangential arc with the radius r:

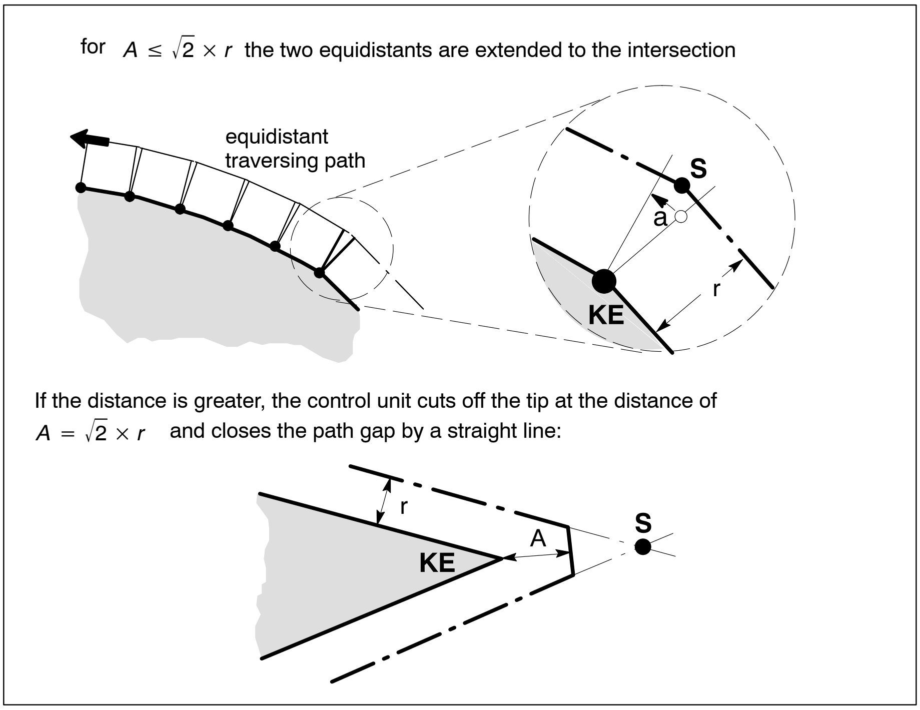


**G69: Intersection**

The control unit tries to close the path gap by determining the intersection of the two equidistants.

1$^{st}$ case: An intersection **exists**.

Depending on the ”A” distance between the ”KE” contour corner and the ”S” intersection, the control unit will proceed as follows:

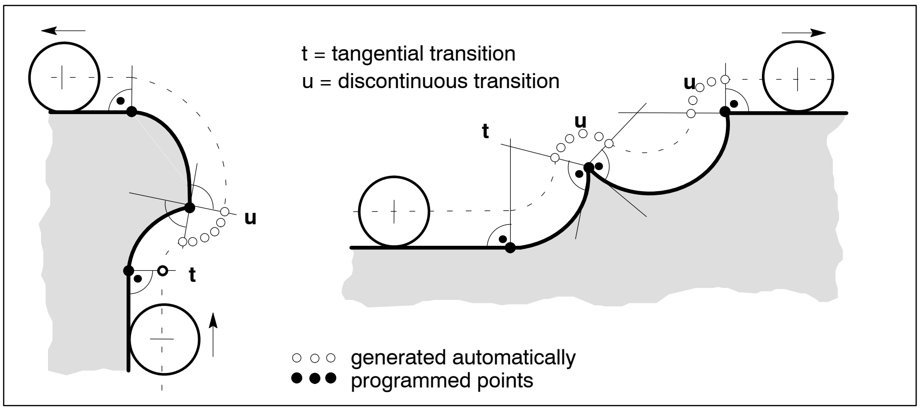


for $A \leq \sqrt{2} \times r$ the two equidistants are extended to the intersection

If the distance is greater, the control unit cuts off the tip at the distance of $A = \sqrt{2} \times r$ and closes the path gap by a straight line:

2nd case: An intersection **does not exist**.

This case can occur in the case of an unsteady contour transition between a straight line and an arc or between 2 arcs. In such a case, as with G68, the path gap will be closed by a tangential arc with the radius of r.



t = tangential transition
u = discontinuous transition
u
t
t
u
u
○ ○ ○ generated automatically
● ● ● programmed points

Programming

G68
G69

G68 and G69 are programmed **without** any preparatory functions.

**Please note for G68 and G69:**

- G68 or G69 act modally.
- The power-up state can be defined in MACODA.

## 3.38 Inch programming G70

Effect

Positional and feed data after G70 must be entered **in inches**. All effective metric values and zero shifts are automatically converted to inches. G70 is modal and cancels the G71 function.

Programming

Programming of G70

| | | |
|---|---|---|
| N... | G70 | from here on all positional and feed information is interpreted in terms of inch |
| N... | X... Y... Z... | all path and feed data to be entered in inch units |

## 3.39 Metric programming G71

Effect

Positional and feed data after G71 must be entered in **metric measures**. All effective inch values and zero shifts are automatically converted to metric values. G71 is modal and cancels the G70 function.

Programming

Programming of G71

| | | |
|---|---|---|
| N... | G71 | from here on all positional and feed information is interpreted in terms of metric units |
| N... | X... Y... Z... | all path and feed data to be entered in metric units |

## 3.40 Linear interpolation with In-position logic G73

Effect

In contrast to G1, a G73 block is **always executed with In-position logic** – irrespective of G61/G62. The In-position logic option is globally determined via the G164 to G166 functions.

Programming

G73 X... Y... Z... F....

**Please note for G73:**

- G73 may be programmed with or without positional data.
- G73 has to be programmed with F word if no feedrate is yet active.
- The programmed feedrate remains effective until overwritten by a new one.
- G73 cancels G0, G1, G2, G3, G5, G10–G13 and G200.

## 3.41 Approach reference point coordinates G74

Effect

The axes programmed in the same block as G74 traverse simultaneously and at rapid to the reference point(s).

With G74 neither reference point cams nor markers are taken into account. G74 is purely a positioning process for the absolute axis positions (i.e. it also applies to axes with distance-coded encoders).

**CAUTION**
**Possibly active compensations will remain unconsidered in the course of this positioning process!**

- G74 is effective only on a block-by-block basis.
- When the reference point is approached with G74 the axis actual values are not set or reset, i.e. the programmed shift values are not affected.
- G74 is canceled when the machine axes programmed with G74 in the block have reached the reference point.
- Any compensations still active, ZS etc. are not taken account of in the G74 block for the programmed axes.
- G74 does **not** set the interface signal ’RAPID’.

Programming

G74 is programmed in a separate block together with the axes to be traversed. The axis addresses must be programmed together with a numerical value (e.g. ”X0”, ”Y0”, ”Z0”). This numerical value has no effect on the position of the reference point. It is only needed to form a complete word.

Auxiliary and special functions may be programmed within the same block.

**Example:** Programming G74

| | |
|---|---|
| `N100 G74 X0 Y0 Z0` | The X, Y and Z axes approach the reference point positions simultaneously linearly. |

## 3.42 Traverse to reference point G374

Effect

The function ”Traverse to reference point” allows to initiate axis referencing in the part program.
For a detailed description, also refer to “PNC Description of Functions” manual.

Traversing to the reference point is a drive-controlled function, i.e. the interpolation takes place in the drive.
For this purpose, the SERCOS command **drive-controlled referencing** (S-0-0148) is triggered by the PNC for each of the programmed axes.
As a consequence, the drive generates its own position inputs for referencing using SERCOS parameters S-0-0147 (referencing parameter), S-0-0041 (referencing velocity) and S-0-0042 (referencing acceleration).

☞ **In order to ensure that the axis potentiometer can work with these drive-controlled movements, it must be transferred to the drive by SERCOS parameter S-0-0108 (feedrate override):**

- **by the PLC through the service channel in 500 ms intervals, or**
- **within the cyclic telegram (MDT). For this purpose, S-0-0108 must be entered in the MDT configuration list (S-0-0024).**

Programming

G374 <Axis address 1> <Integer number>...
<Axis address n> <Integer number>

**Example:**

```
N..
N.. G374 X1 Y1 Z1        referencing of axes X, Y and Z
N..
```

**Please note for G374:**

- For axes with absolute measuring systems, you can define in reference point parameter S-0-0147, whether or not these axes shall reference to the set reference point.
- If several axes have been programmed in one G374 block, these axes will approach their respective reference points independently of each other (no continuous-path operation!).
- With G374, you can program both **synchronous and asynchronous** axes. Block processing is suspended until all axes have traversed to their respective reference points (implicit WAIT).
- For matters of compatibility with the Typ1 osa control, a numerical value must be entered behind the respective axis address which, however, will not be evaluated. (The approach to the reference point is configured statically via SERCOS parameter S-0-0147.)
- There is no difference between the axes traversing to their reference points initiated this way and the operation mode ”Manual”: –Activate traversing to reference point–.

## 3.43 Probe input G75

Effect

The control unit drives the measuring axis **at feedrate** in the direction of the position programmed via G75 and checks in this process whether the probe was triggered.
The axes for which the measuring probe function G75 is to be activated are entered in MACODA parameter 1003 00012.

As soon as the edge defined with machine parameters is recognised (probe comes into contact with the measuring surface; the edge evaluation can be set via MACODA parameter 1003 00011: all axes involved must have identical edge), the control unit performs the following:

- the actual position is stored
- initiates an axis stop with Down Slope function
- clears the distance to go
- deletes G75 (effective block-by-block)

☞ **This function should only be used in combination with a CPL program (e.g. with measuring cycles).**

Programming

Programming G75

```
N100 G75   X400
```

**Please note for G75:**

- G75 must be programmed together with at least one axis position. This value represents the maximum search depth to which the probe must have switched at the latest.
- No auxiliary functions may be programmed in the G75 block. G75 can be programmed with other preparatory functions as far as required.

☞ **You do not have to program WAIT in the part program.**

Evaluation of G75

Continuation of program, evaluation of axis information, safety monitoring, generation of error messages etc. must be realised in the CPL program.

**Example:** Measuring program

| | |
|---|---|
| `N100 G75 Y250 F500` | Approach the contour to be measured (at F500) |
| `110 IF SD(9)=0 THEN` | Query whether probe was displaced |
| `120 YPOS=PPOS(2)` | Store the position in the 'YPOS' variable upon switching of the 2nd axis (Y axis) |
| `N130 (MSG, PROBE DISPLACED)` | |
| `140 GOTO N180` | |
| `150 ENDIF` | |
| `N160 (MSG, PROBE NOT DISPLACED)` | |
| `N170 M0` | Program stop |
| `N180 ...` | |

## 3.44 On-the-fly measurement G175, G275

Effect

With the ”On-the-fly measurement” NC functionality it is possible to use the SERCOS measuring-probe function for measurements without cancelling the programmed movement. Possible collisions are avoided by a suitable measuring probe (contactless or suitable mechanical design).

The function acts as follows:

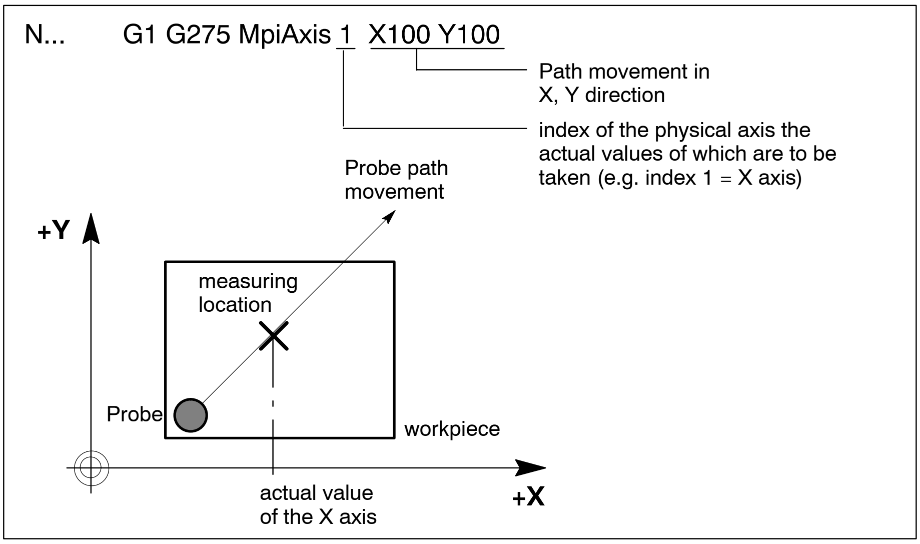
The probe logic of the drive has to be initialized via G175 first (one time after each SERCOS phase startup). Subsequently, the initialized drive (= physical axis) is ready to recognize and store measuring (actual) values.

G275 is newly programmed for each measurement.

The measuring probe traverses on a programmed path to its end point. Via the probe signal generated by mechanical positioning or a contactless system, the actual value of the predefined axis is recorded and stored in the drive (probe logic in the drive). **In contrast to G75, a successful measurement will neither cause a cancellation nor a ramping-down (decelerating to standstill) of the movement.**

**Example:**

| | |
|---|---|
| Path movement in X and Y direction: | actual-value recording of the $1^{st}$ physical axis = X axis; for further details, please refer to MACODA parameter 1003 00001 |



```
N...    G1 G275 MpiAxis 1  X100 Y100
```

Path movement in X, Y direction
index of the physical axis the actual values of which are to be taken (e.g. index 1 = X axis)
Probe path movement
+Y
measuring location
Probe
workpiece
actual value of the X axis
+X

Programming

**Initialization:**

G175 MpiAxis <i>

where

G175: **Intialization** of the probe logic in the SERCOS drive. It is called once following the SERCOS startup via G175.

MpiAxis probe axis parameter

i index of the physical axis the actual values of which are to be taken

Programming

**Starting the measuring cycle:**

G275 MpiAxis <i> <axis1> ..<axes n>

where

| | |
|---|---|
| G275: | **starting** the measuring cycle proper |
| MpiAxis | probe axis parameter |
| i | index of the physical axis the actual values of which are to be taken with G175/G275 |
| axis 1 ... n | probe positioning axes |

**Please note for G175 and G275:**

- G175 and G275 act block by block.
- G175 and G275 act in parallel with the active interpolation.
- G175 and G275 can be programmed together with all interpolation types.
- If no measurement is made at G275 during interpolation, the system will wait at the block end until the measurement is completed (to be ended by program cancellation).
- The measuring position can be inquired via **PPOS** (CPL).
- **PPOS** (CPL) will supply the measuring value referred to the respectively valid zero point for linear modulo axes.
- The information to know whether a measurement was carried out can be inquired via **SD(9**) (CPL).
- The block used to further process the probe information should be preceded by a **WAIT** (CPL) to avoid that further blocks are edited (50 or more blocks in the PNC).
- As an alternative to WAIT, the number of edited blocks can be determined in advance using the **PREPNUM** function.

For further details about the CPL commands, please refer to the CPL manual available on this topic.

☞ **Since both functions involve communication via the SERCOS service channel, you must ensure that the programmed synchronous traversing paths are sufficiently long so that SERCOS communication can be completed within the course of interpolation (at present several 100 msec). If communication was not complete, or if no measurement was performed, the speed will sharply drop at the end of the block and the control unit will wait for the end of SERCOS communication.**
**WAIT may be omitted if, for instance, ”creeping errors” are to be corrected and the main focus is not on the latest measuring results.**

## 3.45 Switching NC blocks via high-speed signal G575

Effect

Function G575 allows early jumping to the next block via the high-speed inputs (HS signal) of the DC/IO.

The traversing blocks in a program are programmed beyond the actual contour. As a result, the end position of an NC block switched by an HS signal is never actually reached. Instead, the motion is measured “on-line”, i.e. continuously, and the block end is reported by an HS signal. The NC block currently being executed is cancelled and the next one is executed.

This function is linked with NC blocks with linear motion.

A linear motion (G0, G1, G10, G11, G73, G200) can be terminated early via a high-speed input signal by programming function G575, which acts block by block.

With G575, there are two options for changing blocks early:

- on-the-fly change of blocks: early block change without changing the programmed end points but with changed geometry,
- change of blocks with abortion: early block change with cancellation of the remaining distance to go.

### 3.45.1 On-the-fly change of blocks G575 HS

Whenever the voltage level y is reached at HS input x, an early change is made to the next block, which must also contain a linear motion. Blocks are changed without checking the maximum velocity step change.

With the function “On-the-fly change of blocks”, the change from one block to the next is made at the current velocity.
Please note the exceptions described for operation mode “Automatic” and the special features in operation modes “Program Block” and “Manual Data Input”.

Programming

G575 HS<x>=<y>

where

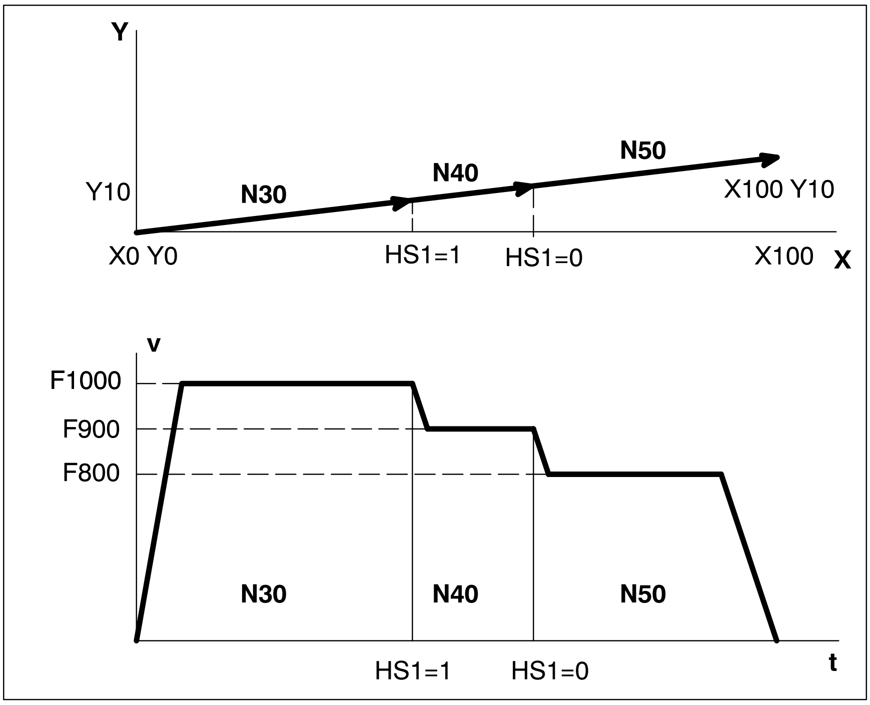
| | |
|---|---|
| x | 1..8 |
| Y | 0..1 |
| HS1–HS8 | designate the HS input at the DC/IO module. |
| HSx=0 or HSx=1 | designate the voltage level required for a change of blocks:<br>HSx=0 (corresponds to 0 V)<br>HSx=1 (corresponds to 24 V) |

**Example:** On-the-fly change of blocks - G575 HSx=y

Depending on external events, up to 3 different feedrates are to be used for traversing on a straight path.

The end points of the programmed blocks must be different because the respective next block (from the point of view of the part program) must contain a path to be traversed.

| | |
|---|---|
| `N20 G0 X0 Y0` | |
| `N30 G575 G1 X70 Y7 F1000 HS1=1` | X axis traverses at F1000 until HS input No. 1 is “High” or X70 Y7 is reached. |
| `N40 G575 X90 Y9 F900 HS1=0` | X axis traverses at F900 until HS input No. 1 is “Low” or X90 Y9 is reached. |
| `N50 X100 Y10 F800` | The remaining distance to go to X100 Y10 is traversed at F800. |

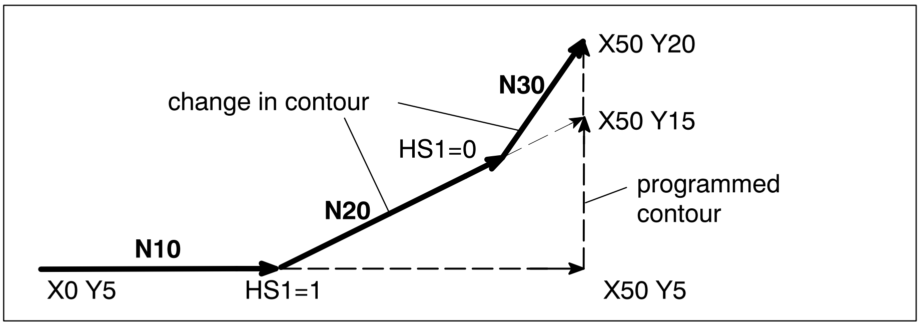


**DANGER**
**The function “On-the-fly change of blocks” only changes the contour, not the programmed end point of an NC block.**

Refer to the following example:

**Example:** Changes in contour caused by “On-the-fly change of blocks”

| | |
|---|---|
| `N05 G1 F100 X0 Y5` | End point: X0 Y5 |
| `N10 G575 HS1=1 G90 X50` | End point: X50 Y5 |
| `N20 G575 HS1=0 G91 Y10` | End point: X50 Y15 (abs. Y5+incr. Y10) |
| `N30 G91 Y5` | End point: X50 Y20 (abs. Y15+incr. Y5) |



## Operation mode “Automatic”

Early changes of blocks are usually made without deceleration to zero axis velocity.

Exceptions:

- Change of blocks at 0 velocity due to a knee in the contour: The change from one block to the next is made at zero velocity if there is a knee > 90 degrees in the contour between the block to be cancelled and the next block.
- Change of blocks at 0 velocity due to G function:
  - The block marked with G575 must end at 0 velocity (G0, G10, G73).
  - In-position function is generally activated (e.g. G61, G161, G163).
  - The next block must begin with 0 velocity due to additional information programmed (e.g. G14, G15, G114, G115).
  - Block-by-block interpolation with path shape is active (G408, G608).

End point information of axes not programmed in the next block is always taken from the cancelled block.

☞ **The feedrate of the block following G575 HSx=y is limited by its original block length.**

## Operation modes “Single block” and “Single step”

In these operation modes, just one block of a part program is executed at a time. Since this precludes an on-the-fly change of blocks, the velocity is reduced to 0 when the external event for a block change occurs.

The end points of axes not programmed in the next block are taken from the cancelled blocks. This allows testing the ”behavior in the Automatic operation mode” by approximations also in the operation modes “Single block” and “Single step”.

### Operation mode “Program block”

In this operation mode, each block of a part program is executed as if it were a complete part program. As this precludes any next blocks, the velocity is reduced to 0 when the external event for a block change occurs.

**DANGER**
**The remaining distance to go at the end of the block is cancelled. Any next NC block with incremental programming will result in traversing to an incorrect end point.**

### Operation mode “Manual data input”

In this operation mode, the velocity is reduced to 0 when the external event for a change of blocks occurs. The same applies when a part program is executed with manual data input.

**DANGER**
**The remaining distance to go at the end of the block is cancelled. Any next NC block with incremental programming will result in traversing to an incorrect end point.**

## 3.45.2 Change of blocks with cancellation

If the level y is reached at the HS input x, the velocity in the current block is reduced to 0 (HSSTOP=0) or a velocity step change is made (HSSTOP=–1) and the remaining distance to go is cancelled (same behavior as with G75).

The end point of any NC block and thus also the starting point of the next NC block are determined by the occurrence of an external event.

Programming

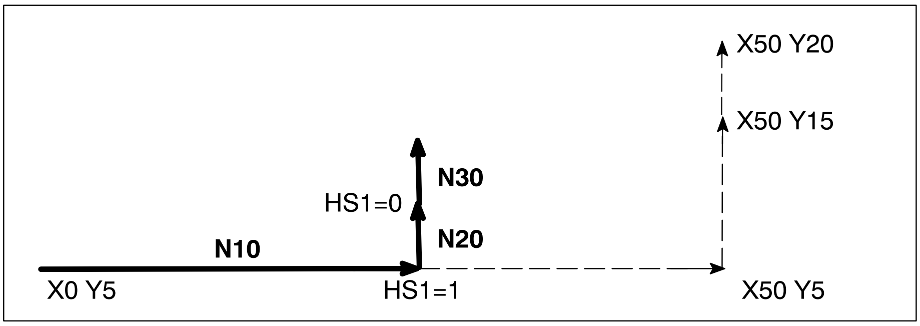
G575 HS<x>=<y> HSSTOP=<z>

where

| | |
|---|---|
| x | 1..8 |
| Y | 0..1 |
| z | –1 or 0 |
| HS1–HS8 | designate the HS input at the DC/IO module. |
| HSx=0 or HSx=1 | designate the voltage level required for a change of blocks:<br>HSx=0 (corresponds to 0V)<br>HSx=1 (corresponds to 24V) |
| HSSTOP=–1 | designates the cancellation with a velocity step change to 0. |
| HSSTOP=0 | designates the cancellation with a velocity down-slope to 0. |

**Example:** Change of blocks with cancellation

| | |
|---|---|
| `N05 G1 F1000 X0 Y5` | End point: X0 Y5 |
| `N10 G575 HS1=1 HSSTOP=0 G90 X50` | End point: external event at X≤ 50 |
| `N20 G575 HS1=0 HSSTOP1=0 G91 Y10` | End point: external event at Y≤ 10 |
| `N30 G91 Y5` | End point: incremental by Y5 from the last occurrence of external event (Y≤ 10) |



In blocks N20 and N30, the Y axis traverses by increments. The end point of the Y axis results from the current position of the Y axis upon cancellation of the previous block to which the programmed incremental step is added.

With the function “Change of blocks with cancellation”, any NC block ends at 0 velocity (even if the programmed event does not occur). Additionally, the remaining distance to go in the aborted block is cancelled.

As the starting point of the next NC block is unknown after a cancellation, this has to be a linear block.

**Example:**
Traversing to a dead stop (e.g. pressure- or torque-driven)

| | |
|---|---|
| `N20 G1 F1000 X0 Y0` | |
| `N30 G575 F10 X10 HS1=1`<br>`    HSSTOP=-1` | X axis traverses until HS input no. 1 is “High” or X10 is reached. Upon the occurrence of HS1=1, the motion is cancelled with a velocity step to 0 and the remaining distance to go is cancelled. |
| `N40 G575 F200 Y100 HS2=1`<br>`    HSSTOP=0` | Y axis traverses until HS input no. 2 is “High” or Y100 is reached. Upon the occurrence of HS2=1, the motion is decreased to 0 at the current acceleration and the remaining distance to go is cancelled. |
| `N50 G0 X0 Y0` | |

## 3.46 Traverse to machine-oriented absolute axis position G76

Effect

G76 is used to traverse to a machine-oriented absolute axis position (absolute position referring to the machine coordinate system) e.g. to change tools, check tools for damage, run measuring cycles, change pallets, etc.

G76 acts only block by block. Prior to any traversing movement, the control unit deselects various compensations and cancels some active functions block by block:

- length compensations (Hxx)
- radius compensations (G41, G42)
- zero shifts (G54 . . . G259)
- inclined plane (G352, G354 . . . G359)
- incremental data input (G91)
- set actual value (G92)
- mirror function (G38)
- workpiece position compensation (G138)

Subsequently, the programmed axes travel simultaneously at **active** speed (at feedrate **or** at rapid travel!) to the programmed machine position.

If deleted by G76, the following functions or compensations will become active again **in the next block**:

- length compensations (Hxx)
- zero shifts
- incremental data input (G91)
- set actual value (G92)
- mirror function (G38)

Programming

G76 is programmed together with positional data

**Please note for G76:**

- G76 can be written together with other preparatory functions (e.g. G93, G94, G95, G0, G1)
- G76 can be written together with the F word
- G76 is effective in conjunction with G93, G94, G95 and the F word

## 3.47 Compensation switchover (drill axis switching) G78 Activate presetting for compensation directions G79

Effect

Machining on machine tools may be performed using tools of different sizes, with the tool clamped in different directions, depending on the machine, or oriented in any direction in space in case of the corresponding machine kinematics.

The function compensation switchover G78 assigns the different **length compensations** of the individual functions to the geometry compensation of either

- the individual directions of the current workpiece coordinate system (WCS) or
- the directions of the tool coordinate system (TCS).

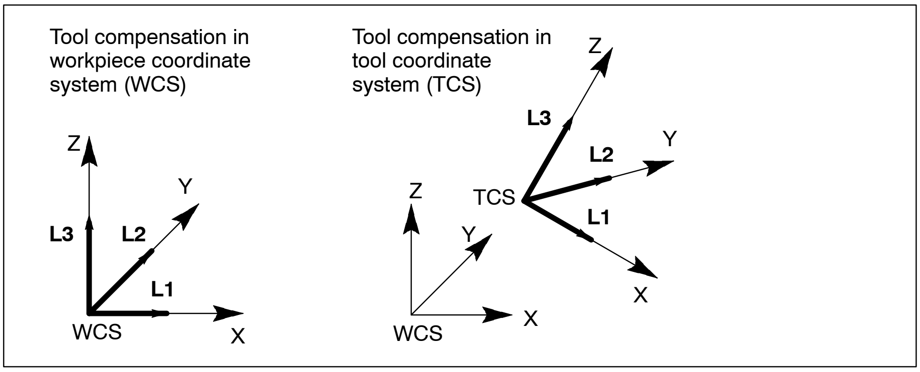
An assignment with respect to the directions of the workpiece coordinate system (WCS) is possible if the tool is aligned perpendicularly to the current working plane and its orientation with regard to the working plane remains constant during machining.
Examples: drilling and turning, milling of plane surfaces.

An assignment of compensations with reference to the directions of the tool coordinate system (TCS) is necessary if the orientation of the tool changes during machining, e.g. milling of freeform surfaces. This allows for a tool length compensation at variable tool orientation. An active axis transformation (e.g. 5 axis or 6 axis transformation) is necessary for this compensation. The compensation values are taken into account within the axis transformation.

The switchover may be activated either in the manual data input mode or in the part program.

Tool length compensations in the PNC are organized by 2 compensation groups as follows:

- **1$^{st}$ compensation group** refers to axes, on which the following compensations have effect:
  - H and $H_{ext}$/$L_{(1)3}$ of the “External tool compensation” (G145–G845)
- **2$^{nd}$ compensation group** refers to axes on which the compensations
  - $L_{(2)1}$, $L_{(2)2}$, $L_{(2)3}$ of the General Tool Compensation (G147–G847) take effect.


Tool compensation in workpiece coordinate system (WCS)
Z
Y
L3
L2
L1
WCS
X
Tool compensation in tool coordinate system (TCS)
Z
L3
L2
Y
Z
TCS
L1
Y
X
WCS
X

Programming

**G78** Coordinate name $_i$ <Compensation $_i$>..{Coordinate name $_n$ <Compensation $_n$>}

where

G78 ... Activate compensation switchover.

Coordinate name i..n  i.. n-th logical axis name (WCS related). Logical axis names are synonyms of the coordinate directions of the WCS.
The logical axis names are determined channel-specific in MACODA parameter 7010 00010 (e.g. 1st logical axis name for X, 2nd for Y, 3rd for Z).

Coordinate name 1..3  XTR, YTR, ZTR (TCS related).
The designations XTR, YTR, ZTR are fixed names for the axes of the tool coordinate system (TCS).
The compensation is only taken into account if a corresponding axis transformation is active.

| | | |
|---|---|---|
| <Compensation $_i$> | ± 1 or ± 13: | **1st** compensation group (H and $H_{ext}/L_{(1)3}$ of the i-th axis. |
| | ± 21: | **2nd** Compensation group, **1st** length compensation ($L_{(2)1}$) of the i-th axis |
| | ± 22: | **2nd** Compensation group, **2nd** length compensation ($L_{(2)2}$) of the i-th axis |
| | ± 23: | **2nd** Compensation group, **3rd** length compensation ($L_{(2)3}$) of the i-th axis |

When double-digit compensation values are entered, the 1st digit stands for the compensation group and the 2nd digit for the compensation index.

The **+ or – sign** before the compensation defines the direction of the compensation. This ensures that the proper direction is taken into account in the computation of length compensations. A positive sign increases the tool (the tool length) vis-à-vis an assumed zero tool. A negative sign decreases the tool vis-à-vis an assumed zero tool.

Compensations relating to several coordinates can be switched simultaneously in a single G78 block (max. 4, because 4 compensations exist). Compensations relating to the WCS and compensations relating to the TCS can be switched together in a single block.

The individual length compensation values always have to be assigned to different coordinate directions.

Programming

**G79** {CG< i>}

where

G79 CG.. Activate presetting for compensation directions

CG<i> **optional**: i=1,2
For the compensation group i specified by CG, the coordinate assignment is reset to the presetting entered in MACODA parameter 7050 00420. G79 without option parameter resets all compensation groups.

**Please note for G78 and G79:**

- G78 and G79 act modally and deselect each other.
- The axis addresses programmed under G78 refer to workpiece coordinates or tool coordinates (XTR, YTR, ZTR).
- G79 may be programmed with other preparatory functions, traversing information or auxiliary functions.
- G78 may be programmed together with other preparatory functions or auxiliary functions in one block.
- The behavior after control start-up is determined by MACODA parameter 7050 00420 or the init string, respectively.
- The individual length compensation values always have to be assigned to different coordinate directions.
- The external tool compensations have to be applied in MACODA parameter 7050 00410.

**Examples:**

| | |
|---|---|
| `G78 X21 ZTR13` | The $L_{(2)1}$ compensation is assigned to the X axis of the workpiece coordinate system (WCS), the $L_{(1)3}$ compensation to the Z axis of the tool coordinate system (TCS). Both compensations are to be taken into account in the positive direction. |
| `G78 Y-1` | The $L_{(1)3}$ compensation is assigned to the Y axis of the workpiece coordinate system (WCS) and taken into account in the negative direction. |
| `G78 YA21 YB22` | The YA and the YB axis of the workpiece coordinate system are assigned the $L_{(2)1}$ and $L_{(2)2}$ compensation, respectively. Both compensations are taken into account in the positive direction. |
| `G79 CG1` | The compensations of the 1st compensation group are assigned their default axes from MACODA. |
| `G79` | All compensations are assigned their default coordinates. |

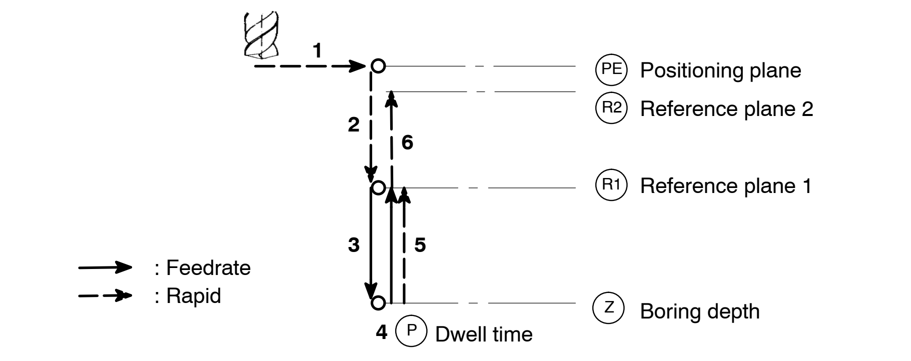
## 3.48 Boring cycles G80-G86, G184

☞ **The boring cycles are programmed in CPL. The assignment between the individual CPL program names and the G functions to be used as well as the number of the transfer parameters are entered as a standard in MACODA parameter blocks 3090 00005 to 3090 00007 and must not be changed.**
**For further details about the parameter blocks, please refer to the MACODA manual.**
**The FEED HOLD interface signal also acts in boring cycles. If required, it has to be blocked by appropriate PLC programming for the duration of a cycle.**
**For more information, refer to the ”PLC Configuration” manual.**

Effect

If you program a boring cycle (G81 to G86 or G184) including the required boring parameters within a program block, the control unit will automatically perform the currently active boring cycle upon reaching the programmed position in the same or in all following blocks, if a machining axis has been programmed there.

General movement sequence of a boring cycle:



1. Positioning in the active plane at rapid
2. Infeed to the programmed R1 reference plane at rapid
3. Infeed to the programmed Z drilling depth at the active feedrate
4. Waiting over a programmed P dwell time (the dwell time serves to compensate for a possibly required deceleration or acceleration to command speed of the spindle in the reversing point)
5. Retraction movement at feedrate or rapid to the programmed R1 reference plane
6. Optional approach of the R2 reference plane at rapid

☞ **All boring cycles can be performed in positive (Z > R1) or negative (Z < R1) boring direction.**
**The active boring axis can be switched over using G78 (please refer to page 3–96).**

☞ **The optionally programmed R2 reference plane may also be located underneath the R1 reference plane. Please note, however, that the distance between the two reference planes is always traversed at rapid. The tool should therefore not be in contact with the workpiece within this range.**

**Boring-cycle overview:**

| Preparatory function | Machining cycle | Infeed to drilling depth at | Action when drilling depth is reached | Retraction movement to reference plane 1 | Application examples |
|---|---|---|---|---|---|
| G80 | no | – | – | – | Delete machining cycle |
| G81 | yes drilling 1 | feedrate | spindle turning (dwell time) | rapid | drilling, boring |
| G82 | yes drilling 2 | feedrate | spindle turning (dwell time) | feedrate | facing, centering |
| G83 | yes deep-hole drilling | step-by-step feedrate | spindle turning (dwell time) | rapid | deep-hole drilling |
| G84 | yes tapping 1 | working feedrate | spindle reversion (dwell time) | feedrate | tapping with compensation chuck |
| G85 | yes boring 1 | feedrate | spindle hold (dwell time) | rapid | boring 1 |
| G86 | yes boring 2 | feedrate | spindle hold (dwell time) | feedrate | boring 2 |
| G184 | yes tapping 2 | working feedrate | spindle reversion | feedrate | tapping without compensation chuck |

Programming

You specify parameters directly following the corresponding G function (except for G80). The number of parameters depends on the selected boring cycle. The order of the individual parameters is **not** arbitrary.

All the parameters must be programmed between the ”[” and ”]” signs and separated from each other by commas.

You can program parameters either as numerical values or as variable names. If you are using variable names, the variables must have valid values at the latest at the time the block is being processed.

**Overview of the parameters used:**

| | | |
|---|---|---|
| **G80:** | N... G80 | Switch off active boring cycle |
| **G81:** | N... X... Y... G81 [Z,R1,P,R2] | G81 on |
| **G82:** | N... X... Y... G82 [Z,R1,P,R2] | G82 on |
| **G83:** | N... X... Y... G83 [Z,R1,K,k,P,R2] | G83 on |
| **G84:** | N... X... Y... G84 [Z,R1,P,R2] | G84 on |
| **G85:** | N... X... Y... G85 [Z,R1,P,R2] | G85 on |
| **G86:** | N... X... Y... G86 [Z,R1,P,R2] | G86 on |
| **G184:** | N... X... Y... G184 [Z,R1,P,R2,GS,U1,U2] | G184 on |

Some parameters may, but do not have to be specified.

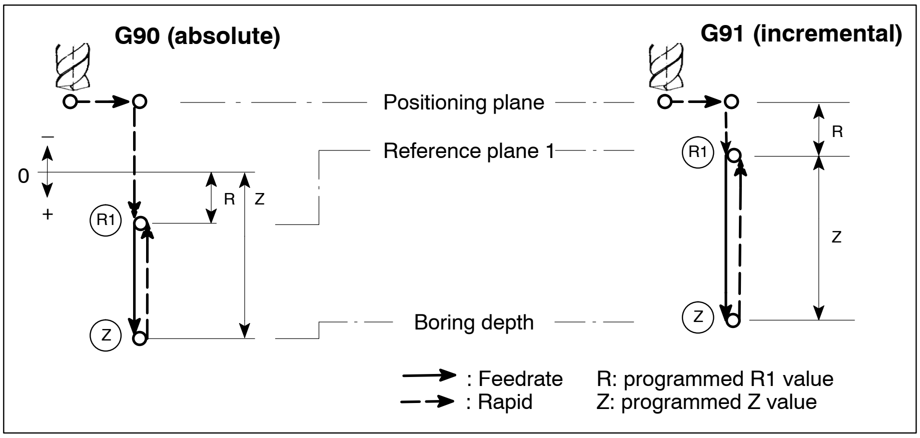
The following applies:

- The parameter value as such may be omitted, but the neighbouring commas must be written.
  **Example:** G81 [Z,R1,,R2] (P missing)
- Commas ahead of the "]" sign are omitted.
  **Examples:** G81 [Z,R1] (P and R2 missing) or
  G81 [Z,R1,P] (R2 missing)

☞ **For details about the meaning of the individual parameters and the movement patterns of each boring cycles, please refer to the sections below. The Z, R1, P and R2 parameters, however, were already introduced to you in the figure on page 3–99.**

The following applies in general:

- "Control reset" always cancels active boring cycles. "M02" or "M30" cancels active boring cycles only if the value "G80" is also entered in MP 7060 00020. If you wish to switch an active boring cycle off, you should at best program G80.
- No expression between brackets must be programmed in a G80 block.
- In the case of more than one program word within one and the same line the parameters must be programmed directly following the boring-cycle call:
  **Correct:** N10 G55 G81 [..] **Wrong:** N10 G81 G55 [..]
- The desired boring position within the positioning plane has to be in the cycle-calling block or after it and may also contain a positioning of the boring axis in infeed or retraction direction. The last position of the boring axis prior to programming a boring cycle, however, has to be on the positioning plane.
- G80 has to be programmed ahead of a change of the boring cycles.
- The entire cycle has to be called for a change of active parameter values.
- Boring cycles may be used both with active absolute data input (G90) as well as with active incremental data input (G91). Please note, however, that the parameters transferred will be interpreted differently:



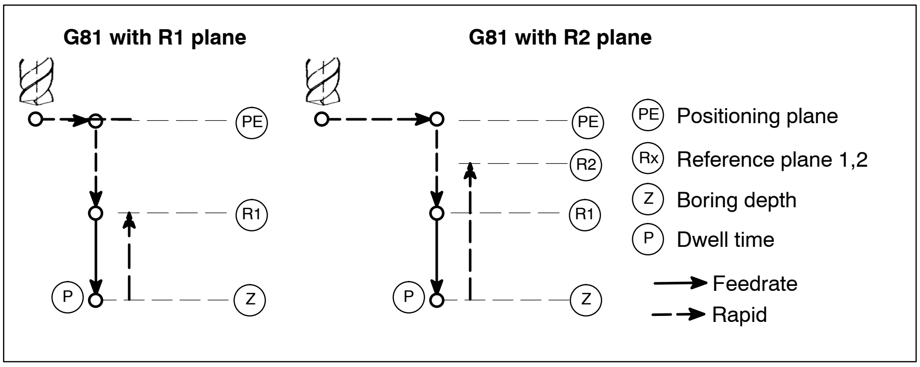
## 3.48.1 Boring cycle G81

Application

Centering and simple drilling operation, facing, boring.

Effect

Upon reaching the Z drilling depth a dwell time becomes active, depending on the programming. Subsequently, retraction at rapid will take place.



Programming

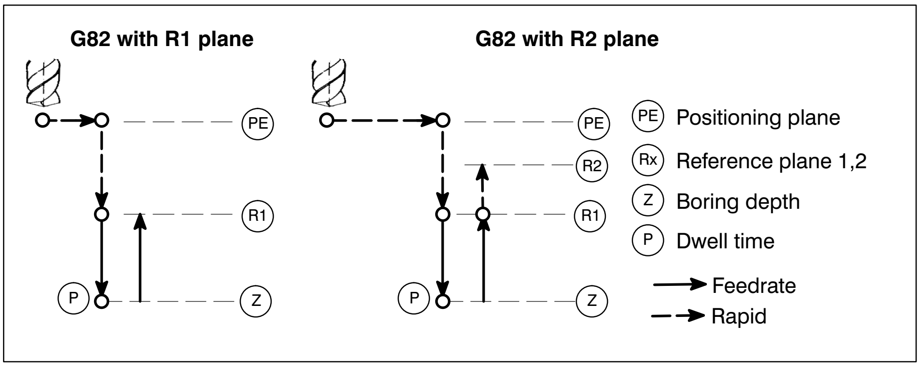
N100 X... Y... G81 [**Z, R1,** P, R2]

Z, R1 must be programmed

P, R2 may be programmed

## 3.48.2 Boring cycle with retraction movement G82

Like G81. However, retraction to R1 is at feedrate.



Programming

N100 X... Y... G82 [**Z, R1,** P, R2]

Z, R1 must be programmed

P, R2 may be programmed

## 3.48.3 Deep-hole drilling cycle G83

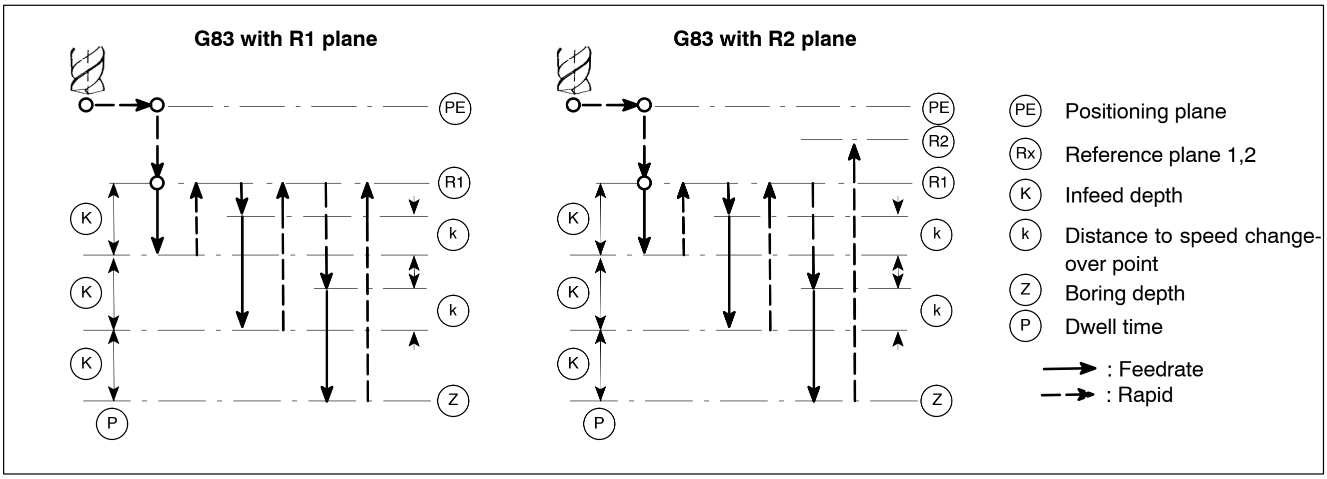
Application

Deep-hole drilling with complete removal of the drilling chips.

Effect

After each arrival at the programmed K infeed depth per cut, one retraction movement at rapid to the reference R1 plane will be performed.

Renewed infeed to the programmed k distance (speed-change point) will also be performed at rapid. Subsequently, the PNC will switch back to feedrate.
Stepwise infeed with corresponding retraction to the reference plane will be repeated until the programmed Z total drilling depth is reached.



Programming

N100 X... Y... G83 [**Z, R1, K, k,** P, R2]

Z, R1, K, k must be programmed

P, R2 may be programmed

The K infeed depth has to be programmed without sign in incremental dimensions, irrespective of the drilling direction.
If the Z max. drilling depth is exceeded owing to erroneous programming of the K infeed depth, the control unit will first interrupt the boring cycle via M0 and display the ”K DRILLING DEPTH TOO LARGE” error message.

After a new start, the boring cycle is cancelled (M30).

## 3.48.4 Tapping with compensation chuck G84

Application

Tapping (left and right) **with** compensation chuck.

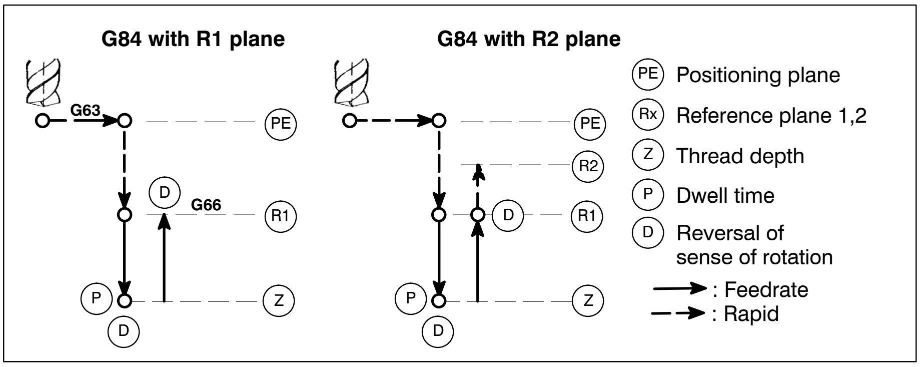
Condition: One **internal spindle** has to be used **as drilling axis**. External spindles are not allowed.

Effect

Tool infeed is done at programmed M3 cw rotating spindle or M4 ccw rotating spindle (right- or left-handed thread).

As soon as the Z drilling depth (thread depth) has been reached, the sense of rotation is reversed, and the P dwell time (if programmed) starts to run.

Subsequently, the retraction movement to the reference plane is performed at feedrate. As soon as it has been reached, the reversal of the sense of rotation is cancelled again.



Programming

N100 X... Y... G84 [**Z, R1,** P, R2]

Z, R1 must be programmed

P, R2 may be programmed

**CAUTION**
**Possible damage to tools or workpieces!**

**During the cycle, any active single-block processing will not be suppressed! This means that the spindle will keep on running after a positioning process within the cycle. This may lead to damage to the tool and the workpiece.**
**You should therefore ensure that the control unit executes the cycle in the AUTOMATIC mode only!**

## 3.48.5 Tapping without compensation chuck G184

Application

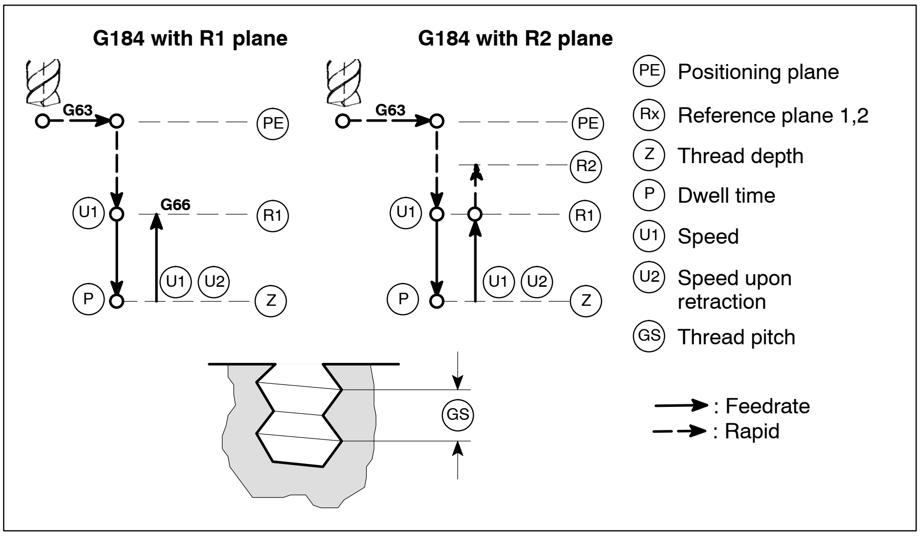
Tapping (left and right) **without** compensation chuck.

Condition: controlled spindle; G32 exact tapping without compensation chuck.

Effect

Tool infeed is calculated internally on the basis of the product of ”speed x thread pitch”. You select the sense of rotation (right- or left-handed thread) via the sign of the GS parameter (thread pitch).

As soon as the Z drilling depth (thread depth) is reached, the sense of rotation is reversed. Subsequently, the retraction movement to the reference plane is performed at feedrate. The sense of rotation of the spindle remains effective until you program a new boring cycle.



Programming

N100 X... Y... G184 [**Z,R1,**P,R2,**GS,U1,**U2,RP$^{*)}$] right-handed thread

N100 X... Y... G184 [**Z,R1,**P,R2,**−GS,U1,**U2,RP$^{*)}$] left-handed thread

Z, R1, GS, U1 must be programmed

R2, U2, RP may be programmed

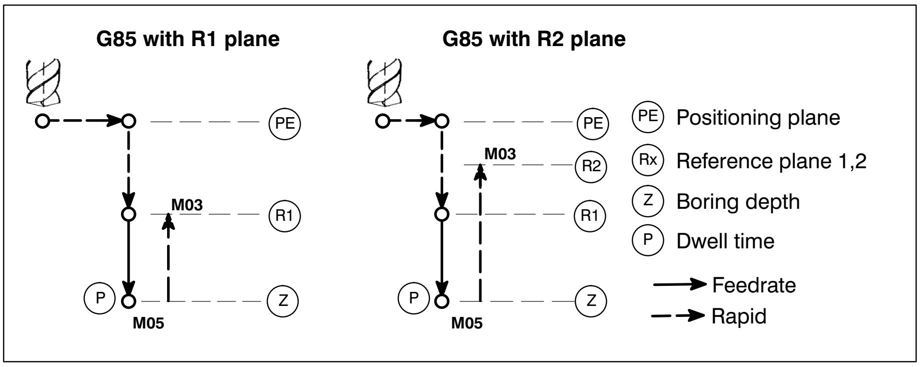
$^{*)}$ Optionally, it is also possible to program the **RP** parameter. RP determines the **orientation position** of the spindle.

☞ **For matters of compatibility, P may be assigned a value. However, P dwell times programmed are not evaluated any more!**
**The syntax entered would read as follows:**
**N100 X... Y... G184 [Z,R1,,R2,−GS,U1,U2,RP$^{*)}$**
**Example: “left-handed thread”)**

## 3.48.6 Boring G85

Application Boring

Effect Upon reaching the Z drilling depth, the spindle stops. Depending on programming, a dwell time will start to run. Subsequently, retraction at rapid will take place.



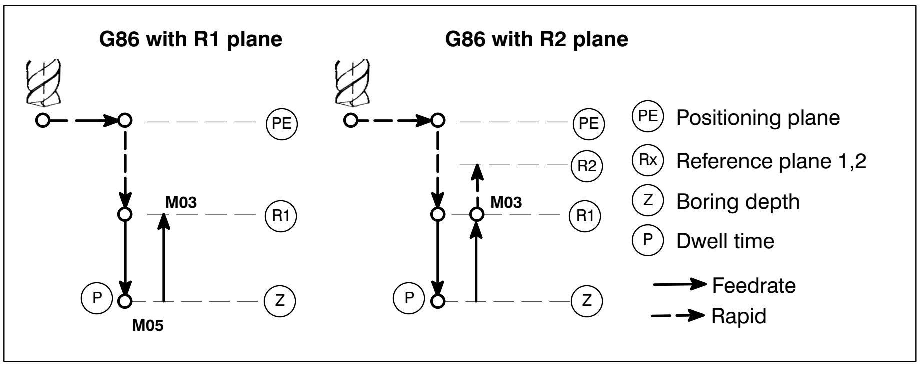
Programming N100 X... Y... G85 [**Z, R1,** P, R2]

Z, R1 must be programmed

P, R2 may be programmed

## 3.48.7 Boring with retraction movement G86

Like G85. However, retraction to R1 is at feedrate.



Programming N100 X... Y... G86 [**Z, R1,** P, R2]

Z, R1 must be programmed

P, R2 may be programmed

## 3.48.8 Programming examples

**Example 1:** General programming.

```
N90 G1 M3 S1050 F400
N91 G81 [-1000,-800]                 cycle call without positioning
N92 X600 Y800                        drilling starts from this block
N95 X500 Y700 G81 [-1000,-800]       Cycle call with positioning.
                                     From this position, drilling is al-
                                     ready being performed.
N96 X600 Y800
N100 X800 Y700 G81                   retraction to R2 plane; no dwell
    [-1000,-800,,-600]               time
N110 X0 Y0 G81 [-1000,-800]          retraction to R1 plane only; no
                                     dwell time
N111 X-100 Y-500
N150 X-400 Y200 G81                  retraction to R1 plane only;
    [-1000,-800,1]                   dwell time 1s
N151 X200 Y300
```

**Example 2:** Programming the boring-cycle parameters via CPL variables.

```
N5 X200 Y400 M3
10 Z=1000                            Definition of the CPL variables
20 R1=800
30 P=2
40 R2=900
N50 X... Y... G84 [Z,R1,P,R2]
```

**Example 3:** Calling the boring cycle in the main program. The positions to be approached have been programmed in a subprogram.

```
N05 X100 Y100 Z200
N10 G91 G81 [100,10]
N20 P1000   ------------------------->  P1000 Subprogram
N30 G80                            |->
N40 G90                            |      N10 G91 X10 Y10
N50 X500 Y100 Z200                 |      N20     X20 Y20
N10 G91 G81 [100,10]               |      N30     X30 Y30
N20 P1000   -----------------------|      M30
N30 G80
N40 G90
```

**Example 4:** Drill axis switching; X axis with positive compensation

```
N10 G78 X1
N20 G1 M3 S1050 F400
N30 G81 [-100,-800]
N40 Y500 Z700
N50 G80
N60 G79
```

## 3.49 Absolute data input 1 G90 Incremental programming G91 Absolute data input 2 G189 Local absolute data input AC(...) Local incremental data input IC(...)

Effect

Workpiece drawings may be dimensioned in absolute or relative (incremental) dimensions.

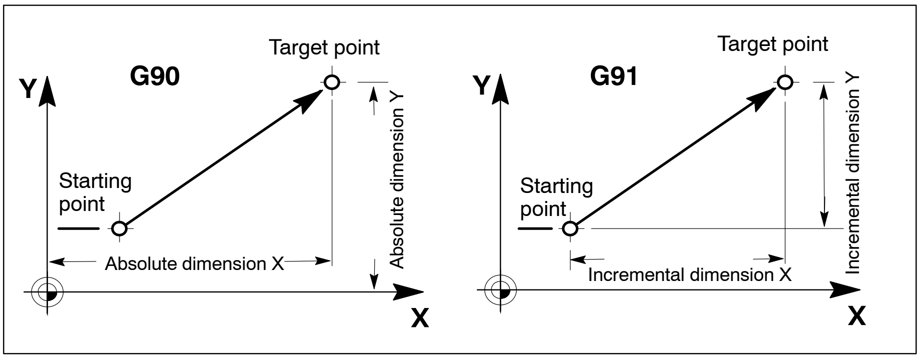
The PNC may be set in both formats. The two variants of absolute programming differ as to the treatment of endless axes (modulo axes) which have been configured via MACODA parameter 1003 00005.

**G90:** The control unit will interpret dimensions as **absolute values** referring to the active zero point. Endless axes, entered as ”can be switched over via G90/G189” in MP 1003 00005, will traverse with sign logic (also ref. to section 3.64).

**G91:** The control unit will interpret dimensions as **incremental values** referring to the position last approached (relative or incremental dimension).

**G189:** The control unit will interpret dimensions as **absolute values** referring to the active zero point. Endless axes, entered as ”can be switched over via G90/G189” in the MP 1003 00005, will traverse with ”shortest-path” logic (also ref. to section 3.64).

The following diagram illustrates the difference between G90 and G91:



**Please note for G90, G91, and G189:**

- All functions act modally and cancel each other mutually.
- While G90 is active, the length compensation is added to or subtracted from – depending on the sign before the correction value – any new displacement entered for the spindle axis when the **H word** is called with the next **position data for the spindle axis**. While G91 is active, the compensation value is taken into account only in the computation of the first displacement.

Programming

G90, G91

| | |
|---|---|
| `N10 G90` | all subsequent dimensions will be interpreted as absolute values referring to the active zero point. |
| `N20 X100 Y100` | current machine position: X100 Y100 |

| | |
|---|---|
| `N30 G91` | all subsequent dimensions will be interpreted as relative values referring to the position last approached. |
| `N40 X50 Y10` | current machine position: X150 Y110 |

☞ **For further details on the positioning types of endless axes, also refer to G150/G151.**

## Local absolute / incremental data input

With G90/G189 or G91 active, using the AC and IC address attributes, it is possible to program individual axes block by block in absolute or incremental terms.

- **AC(...)**: the programmed axis value is to be treated as **absolute**.
- **IC(...)**: the programmed axis value is to be treated as **incremental**.

Programming

<logical axis address> = <address attribute>(<value>)

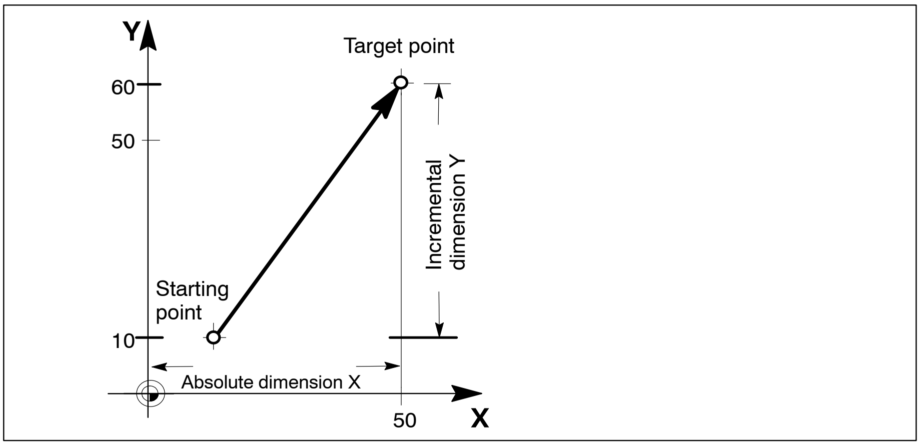
| | |
|---|---|
| X = AC (50) | Irrespective of the presetting via G90/G189 or G91, the X axis will travel to the absolute position 50 (referred to the current coordinate system). |

Within one block it is possible to program different attributes for different axes.

**Example:**

G91 X=AC(50) Y50

| | |
|---|---|
| X: | X travels to the absolute position X=50 |
| Y: | Y travels by increments of 50 mm (to Y=60) |



☞ **Local incremental data input using the IC address attribute in the context of function G76, ”Traverse to machine-oriented absolute axis position” is not permitted because it would lead to a runtime error.**

## 3.50 Set actual value G92

Effect

The effect of G92 differs depending on the programmed axis information:

Programming G92 **without** axis information:

The current actual value of all axes is **set to machine coordinates** without taking into account compensations and zero shifts.

Programming G92 **with** axis information:

The current actual value of an axis is set to the programmed value.

**CAUTION**
**This option must not be used as long as a ZS is active. If necessary, you must program the G53 instruction ahead of G92.**

No axis movement will occur with G92. The new position values will be displayed.

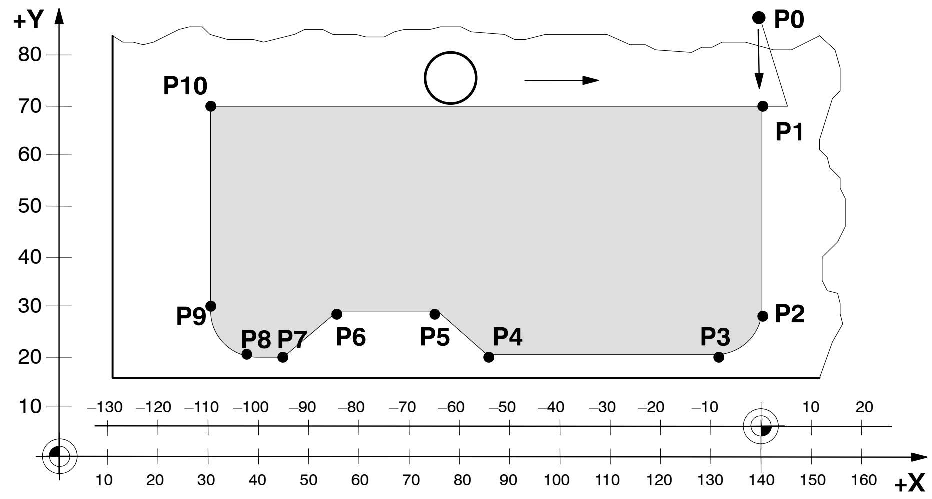
Programming

| | |
|---|---|
| `N... G92 X0 Y0`<br>`N...`<br>`N...` | The current actual values of the X and Y axes are set to ”0” (point-of-reference shift). The actual values of the Z axis remain unchanged. |
| `N... G92` | Set all actual values to machine coordinates (cancel point-of-reference shift). |

**Please note for G92:**

- G92 acts block by block.
- Other functions may be programmed jointly with G92 in the same block provided these functions do not require an axis address.
- Whether a G92 shift is to be deleted after a control reset or whether this shift is to be retained can be entered for each channel in MACODA parameter 7050 00510. If the preset option remains unchanged, the shift values will be deleted upon control reset.

**Example:**



```
G90   F200
G1    X140     Y70
G92   X0
G1             Y30
G2    X–10     Y20    I–10
G1    X–55
      X–65     Y30
      X85
      X95      Y20
      X–100
G2    X–110    Y30    J10
G1             Y70
      X5
      X0       Y80
G92
M30
```

## 3.51 Time programming (feedrate programming as block duration) G93

Effect

The control unit will interpret subsequent F words as machining time for a programmed linear (G1) or circular (G2, G3, G5) distance.

The same applies in the case of polar coordinate programming.

Programming

**Example:**

```
N10 G93 G1 X300 Z400 A50 B120
    F60
```

The programmed linear interpolation will last **60 seconds**.

**Please note for G93:**

- acts modally
- **remains stored internally** in the case of a switch-over to G94 or G95 and becomes active again if G93 is selected again
- The power-up condition can be set in MACODA parameters 7060 00010 and 7060 00020.

**CAUTION**
**Upon ”power off (7060 00010) and/or ”reset” or ”control reset” (7060 00020), the machining time as set in the MACODA parameters will become active (default value = F0)!**

☞ **In addition, please keep taking MACODA parameter 8004 00001 into account!**
**The control unit will automatically calculate the required feedrate on the basis of the path length of the block and the programmed machining time.**
**This feedrate, however, can be restricted according to the programmed path and the max. values of the axes involved!**

## 3.52 Feedrate programming G94

The programmed feedrate refers to the sum of all programmed axes involved in the movement. If a maximum of 3 axes standing perpendicularly to each other (Cartesian system, **no** working range coordinates!) are moved, the programmed feedrate corresponds to the expected path feedrate of the tool in space.

If other axes are traversed as well, the path feedrate is reduced because the other axes also contribute to the feedrate.
The weighting of the ratio “rotary axes to linear axes” is entered in MACODA parameter 7040 00110, ”scaling of the rotary axis feedrate at G70 or G71” in dependence on the active measuring system (inch/metric).

Axes which are not programmed and whose movement is not derived implicitly from the programmed axis movement, are generally not included in the feedrate computing, i.e. the feedrate exclusively refers to the programmed axes (examples: C axes in case of tapping, tool axis in case of tangential tool guidance, etc.)
In MACODA parameter 1003 00020 it is furthermore possible to explicitly select axes which are removed from feedrate computing when G594 (“suppressing axes for feedrate computing and separate programming of rotary axis feedrate”) is activated. The axes are then synchronously moved along with the feedrate contributing axes, monitoring the dynamic values of all axes involved in the movement and reducing the feedrate if necessary.

If axes not contributing to the feedrate are involved in the movement exclusively, the programmed feedrate refers to their path.

In case of **active working range coordinate programming**, the active feedrate refers to (maximally 3) linear working range coordinates. A simultaneously programmed orientation motion as well as additional programmed synchronous axes (pseudo-coordinates) are guided along synchronously.
If no linear working range coordinates have been programmed, the effectiveness of the feedrate is transmitted to a programmed orientation motion. In this case too, programmed pseudo-coordinates are guided along synchronously.

If none of the working range coordinates has been programmed, the resulting behavior will be analogous to the case of the deselected axis transformation (no working range coordinate programming active).

Effect

The programmed feedrate is interpreted as being stated in **mm/min** (with active G71) or in **inch/min** (with active G70). In case of rotary axes, the programmed feedrate corresponds to °/min.

☞ **You can adapt the unit of measure to your specific requirements in MACODA parameter 7040 00010!**

The feedrate can be programmed with the following parameters:

- **F**
  F is generally used for feedrate programming.
- **Omega**
  2nd Feedrate value:
  - if working range coordinates are active and no linear working range coordinates have been programmed but an orientation motion instead.
  - for axes suppressed from feedrate computing (G594) and for axes not contributing to the feedrate if no feedrate contributing axis has been programmed. Alternatively, Omega may be specified instead of the last active F.

Restrictions:

- The limit of F and Omega depends on the maximum speeds of the axes involved.
- In case of some NC functions (e.g. G4, G104), “F” has a different meaning.

Programming

**G94... F**<value> Feedrate programming with F

where

| | |
|---|---|
| F<value> | Feedrate value for axes contributing to the feedrate and axes not contributing to the feedrate activated. |
| value | Magnitude of the feedrate. Unit corresponding to the presetting G70/G71 in case of linear axes. For rotary axes, in °/min. |

Example: F in mm/min

```
N10  G71                          Feedrate in mm/min
N10  G1 G94 X200 Z300 F200        programmed feedrate 200 mm/min
N11  G4 F40                       dwell time 40 seconds
N12  X300 Z400                    the 200 mm/min feedrate
                                  is active again
```

Example: F in inch/min

```
N10  G70                          Feedrate in inch/min
N10  G1 G94 X200 Z300 F200        programmed feedrate 200 inch/min
N11  G4 F40                       dwell time 40 seconds
N12  X300 Z400                    the 200 inch/min feedrate
                                  is active again
```

## Programming

**G94... Omega**<value> Feedrate programming with Omega

where

| | |
|---|---|
| Omega<value> | Feedrate value for axes not contributing to the feedrate activated. |
| value | Magnitude of the feedrate.<br>Unit (linear axes): always in mm/min, irrespective of G70/G71.<br>Unit (rotary axes): in °/min |
| Omega 0 | active Omega value is deactivated. |

**Example**: X,Y,Z = feedrate contributing axes; B,C, not contributing to the feedrate

| | |
|---|---|
| `N1 G70 G1 G594 F100` | The feedrate is 100 inch/min (suppressing axes B and C from feedrate computing). |
| `N3 Y200 B200` | The Y axis traverses at 100 inch/min to position 200 inch, the B axis is synchronously moved along to position 200° (B is not feedrate contributing!). |
| `N4 C200` | The C axis traverses at 100 °/min to position 200° (C moves with the last programmed F value because no feedrate contributing axis is programmed). |
| `N5 Omega 1000` | Omega is 1000°/min (1000 inch/min) |
| `N6 X0 C0` | The X axis traverses at 100 inch/min to position 0 inch. The C axis is carried along to position 0° (C is not feedrate contributing!) |
| `N7 B300` | The B axis traverses at 1000 °/min to position 300° (Omega acts on not feedrate contributing axes). |
| `N11 Omega 0` | Omega is deactivated |
| `N10 B10` | The B axis moves at 100 °/min to position 10° (F is effective again). |

**Please note for G94:**

- acts modally
- **remains internally stored** in the case of a switch-over to G93 or G95 and becomes active again if G94 is selected again
- the power-up condition can be determined in MACODA parameters 7060 00020 and 7060 00010.

**DANGER**
**Changing the feedrate may pose a risk to the machine and personnel!**

**Upon ”power off” (7060 00010) and/or ”reset” or ”control reset” (7060 00020), the feedrate as set in MACODA parameters will become active (default value = F0)!**

## 3.53 Incremental speed programming with acceleration adaptation G194

Effect

Using the G194 function it is possible to increase or decrease the active feedrate in increments. Within a block in which G194 was programmed the acceleration will be adapted in such a manner that the resulting speed will only be reached at the end of the block. This can be used to automatically achieve a very soft acceleration behavior. The existing acceleration and deceleration limits are monitored. (if necessary, the final speed is not reached before the next block).

Additionally, you can change the spindle speed by increments for a specific path. Within the block where this function is programmed, the spindle speed is adjusted linearly along the path so that the desired spindle speed is reached at the end of the block.

Any number of blocks containing G194 may be programmed. The programmed speed change will always refer to the existing speed of the preceding block.

Programming

| | |
|---|---|
| N.. G194 F100 X.. Y.. Z.. | The path speed will increase within the block by 100 mm/min. |
| N.. G194 F−50 X.. Y.. Z.. | The path speed will decrease within the block by 50 mm/min. |
| N.. G194 S1=100 X.. Y.. Z.. | The specified speed of the 1st spindle is increased within the block by 100 rev./min. |
| N.. G194 F100 S2=150 X.. Y.. Z.. | Within the block, the path speed is increased by 100 mm/min and the specified speed of the 2nd spindle by 150 rev./min. |

**Please note for G194:**

- The function is not modal. However, the resulting feedrate will have a modal effect on the subsequent blocks.
- The calculated acceleration acts only block by block.
- The unit of incremental feedrate corresponds to the F value programmed via G94.

☞ **If the program is cancelled within a G194 block, deceleration at the calculated acceleration will occur.**

## 3.54 Feedrate programming in mm/rev G95

Effect

The control unit will interpret all subsequent F words as feedrate in **mm/rev**.

☞ **You can adapt the unit of measure to your specific requirements in MACODA parameter 7040 00020!**

The main spindle must be activated before the first traversing movement is to be performed with G95. The main spindle is defined in MACODA parameter 7020 00010 or using the MAINSP function (refer to page 4–17).
With the main spindle running, the following axis data is interpolated for the feedrate in mm/rev in accordance with the F word programmed.

The limit of the F word depends on the max. feedrate values set in the parameters of the axes involved.

**DANGER**
**Changing the feedrate may pose a risk to the machine and personnel!**

**Upon ”power off” (7060 00010) and/or ”reset” or ”control reset” (7060 00020), the feedrate as set in the MACODA parameters will become active (default value = F0)!**

☞ **Considering that the current feedrate is derived from the spindle speed, the active feedrate will be influenced by both the spindle as well as by the feedrate potentiometer!**

Programming

Feedrate programming in mm/rev with dwell time

```
N9   S2000 M4                 spindle speed 2000 rev/min, ccw run
N10  G1 G95 X200 Z300 F0.2    programmed feedrate 0.2 mm/rev
N11  G104 F4                  dwell time 4 spindle revolutions
N12  X300 Z400                the 0.2 mm/rev feedrate is active again
```

**Please note for G95:**

- acts modally
- **remains internally stored** in the case of a switch-over to G93 or G94 and becomes active again if G95 is selected again
- the power-up condition can be determined in MACODA parameters 7060 00020 and 7060 00010.

## 3.55 Constant cutting speed (rotating function) G96<br>Direct speed programming (rotating function) G97

Effect

**G96: Constant cutting speed**

Using G96 and the S word, a constant cutting speed is programmed for a specific spindle in the unit m/min (G71) or feet/min (G70). For this purpose, the control unit automatically changes the speed of the axis entered in MACODA parameter 7010 00110, ”Reference axis for constant cutting speed”.

If the cutting speed is to be changed in the further course of the program, it is sufficient to program the S word of the respective spindle. In case of a changeover from G96 to G97 and back to G96, it may possible to do without programming of the S word. In this case, the last cutting speed programmed under G96 becomes effective again.

Together with G96, it is also possible to switch several spindles to individual constant cutting speed. The spindles not programmed then continue to run in the operating mode ”direct speed programming”.

In case of active G96, the spindle override is effective as in G97. A lower or upper speed limit can be set using the speed limiting functions G192 or G292.

A possibly desired gear-range change has to be carried out ahead of G96. If the function automatic gear-range switchover is active, the gear range remains active until G97 is selected again.

To determine the tool contact point, G96 takes the tool length from the length compensation of the ”general tool compensation” (MACODA parameter 7050 00420[6]) assigned to the reference axis.

The function G196 (compatibility with earlier versions) exists in addition to G96. It differs from G96 as follows:

- G196 interprets the programmed S word in unit mm/min (G71) or inch/min (G70).
- To determine the tool contact point, the tool length from the 1st additive ZS group (G154–G159) is subtracted from the current value of the 1st axis internally at G196.

**G97: Direct speed programming**

An S word programmed with G97 causes a constant speed, irrespective of the position of the 1st axis.

Programming

**G97**.. S<i>=<Speed>.. {S<n>=<Speed>}

where

| | |
|---|---|
| S<i> | S word of the i-th spindle in the same block as G97 (i=1..n). |
| Speed | Determine the speed of the i-th spindle. If G97 precedes G96/G196 in the beginning section of the program, or if a specific speed is supposed to be reached, the S word **must** be written.<br>The S word may be omitted if a change from G96/G196 to G97 is made, in this case the currently active speed is taken over. |

Programming

**G96**.. S<i>=<Cutting speed>.. {S<n>=<Cutting speed>}

where

| | |
|---|---|
| S<i> | S word of the i-th spindle (i=1..n). |
| cutting speed | Determine cutting speed for the i-th spindle. Using G96 and the S word, the desired cutting speed is programmed for the corresponding spindle in the unit m/min (G71) or feet/min (G70).<br>If the cutting speed is to be changed in the further course of the program, it is sufficient to reprogram the S word of the respective spindle.<br>In case of a changeover from G96 to G97 and back to G96, the last cutting speed programmed under G96 will become effective again. |

**Please note the following for G96, G97 and G196:**

- G96, G97 and G196 act modally and cancel each other mutually.

**Example:** Behavior of the function with 2 spindles configured G97 S... –and– G96 S...

| | |
|---|---|
| `G96 S1=...`<br>`.`<br>`.` | Only the 1st spindle runs at constant cutting speed. The 2nd spindle has speed programming. |
| `G96 S1=... S2=...`<br>`.` | The 1st and 2nd spindle run at constant cutting speed. |
| `G97 S1=... S2=...`<br>`.`<br>`.` | Both spindles run under speed programming (the speed is calculated internally, unless programmed). |
| `G96` | The last setting active under G96 is valid again. |

## 3.56 Spline programming

**G99**
**SplineDef**

Effect

The NC supports programming of 4 splines (for a detailed description, refer to “PNC Description of Functions” manual).

- Spline with coefficient programming (type 0)
  (polynomial coefficients of CAD/CAM system)
- $C^1$-continuous, cubic splines with interpolation point programming (type 1)
  (tangential transitions at the interpolation points)
- $C^2$-continuous, cubic splines with interpolation point programming (type 2)
  (continuous-curvature transitions at the interpolation points)
- B splines with checkpoint programming (type 3)
  (curve shape near the interpolation points)

Programming

- In the program, the spline type first has to be initialised by programming “**SplineDef**”.
- Then the spline is activated by **G99**.

The following functions can **not** be programmed with splines:

- Tensor orientation this refers to the tensor syntax Ox(..), Oy(..) and Oz(..) as well as the Eulerian angles phi, theta, psi.
- Polar coordinate interpolation (end face)
- Path compensation G41/G42.
- Cancel distance to go.
- Punching and nibbling with path separation
- Circle and helix with tangential entry to previous spline element
- Chamfers and curvatures
- Tangential tool guidance
- Precision programming
- Area control

### Spline with coefficient programming (type 0)

Initialization

SplineDef(<Id>) Initialize type 0 spline

where

<Id> Spline degree: 1,..., 5
**Example**: SplineDef(5)

## Activation

| | |
|---|---|
| G99 | Activating the “Spline” type of path |

## Modal parameters for G99

- **Coordinate/axis programming**

  Each channel coordinate may be moved if desired

  - as a spline by specifying the polynomial coefficients

    | | |
    |---|---|
    | <CoordName>(<c0>,<c1>,...,<cn>) | Programming individual coordinates with polynomial coefficients |

  - or linearly by specifying the end positions

    | | |
    |---|---|
    | <CoordName>(<EndPos>) | Programming the end position of individual coordinates/axes |

  where

  | | |
  |---|---|
  | <CoordName> | Name of coordinate or axis |
  | <c0>,<c1>,...,<cn> | Polynomial coefficient of a coordinate. n=1,..,5 corresponds to the spline degree defined in SplineDef |
  | <EndPos> | End position of the coordinate |

  **Example**:

  ```
  SplineDef(3)
  G99 X(0.1,1.25,0.5,0.73) Y30 B(0.0,-1.0,0.1,-0.2)
  ```

- **Denominator polynomial programming:**

  | | |
  |---|---|
  | DN(<$g_0$>,<$g_1$>,...,<$g_n$>) | Common denominator polynomial for all spline coordinates.<br>Exact description of rational Bezier splines, rational B splines (NURBS) and all conics. |

  where

  | | |
  |---|---|
  | <$g_0$>,<$g_1$>,...,<$g_n$> | Polynomial coefficient of the denominator polynomial.<br>n=1,..,5 corresponds to the spline degree defined in SplineDef. |

  **Example**:

  ```
  SplineDef(3)
  G99 X(0.1,1.25,0.5,0.73) B(0.0,-1.0,0.1,-0.2)
  DN(1,0,1)
  ```

- **Orientation vector programming:**
  Active working range coordinate programming (**Coord(..)**) is required for this programming method.

| | |
|---|---|
| O1($<o_{10}>,< o_{11}>,...,< o_{1n}>$) | x component of the orientation vector |
| O2($<o_{20}>,< o_{21}>,...,< o_{2n}>$) | y component of the orientation vector |
| O3($<o_{30}>,< o_{31}>,...,< o_{3n}>$) | z component of the orientation vector |

where

| | |
|---|---|
| $<o_{10}>,< o_{11}>,...,< o_{1n}>$ | Spline coefficients of the x component of the orientation vector. |
| $<o_{20}>,< o_{21}>,...,< o_{2n}>$ | Spline coefficients of the y component of the orientation vector. |
| $<o_{30}>,< o_{31}>,...,< o_{3n}>$ | Spline coefficients of the z component of the orientation vector. |
| | n=1,..,5 corresponds to the spline degree defined in SplineDef. |

☞ **The common denominator polynomial (DN) does not apply to vector orientation.**

**Example**:

```
N00 ;Spline-coefficients for vector orientation
001 PI=3.14159 : PIH = PI/2 : PIHQ = PIH*PIH :
    PIHC = PIHQ*PIH
N10 G1 F30000 X0 Y0 Z0 B90 C0
N20 SplineDef(3)
N30 Coord(1) ;5-Axis transformation with vector
    orientation ON
N40 G99 PL[PIH]
N50 O1(1,0,-3/PIHQ,2/PIHC) O3(0,1,(3-PI)/PIHQ,(-2+PIH)
    /PIHC)
N60 O1(0,0,3/PIHQ,-2/PIHC) O3(1,0,(-3+PIH)/PIHQ,
    (2-PIH)/PIHC)
N70 O(0,1,0) ;Normal vector orientation
N80 G1
N90 Coord(0)
```

- **Spline parameter length programming:**

  The spline parameter length is the length of the interval defined for w, where w is active between 0...$w_e$.
  The value $w_e$ acts modally and remains valid for all NC blocks until G99 is deselected. PL must be programmed in the first traversing block after G99, otherwise, a runtime error will occur.

  | | |
  |---|---|
  | {PL<$w_e$>} | **optional:** Programming the spline parameter length. |

  where

  | | |
  |---|---|
  | <$w_e$> | any value > 0 |

  **Example**:

  ```
  G99 X(0.1,1.25,0.5,0.75) B(0.0,-1.0,0.1,-0.2)
       PL0.6
  ```

  (X = 0.1 + 1.25 w + 0.5 $w^2$ + 0.75 $w^3$ and
  B = 0.0 – 1.0 w + 0.1 $w^2$ – 0.2 $w^3$ where
  w active between 0...0.6)

## $C^1$- and $C^2$-continuous cubic spline (type 1 and type 2)

Initialization

| | |
|---|---|
| SplineDef(<Id>,<Members>) | Initialize type 1 or type 2 spline |

where

<Id> four-digit integer, consisting of <Type> <Parameter settings><Tangent computation><Degrees>

| | |
|---|---|
| <Type>: | 1= continuous **$C^1$**-spline with tangential transition |
| | 2= continuous **$C^2$**-spline with continuous-curvature-transition |
| <Parameter setting>: | 1= equidistant |
| | 2= chordal |
| | 3= centripetal |
| <Tangent computation>: | 1= Bessel |
| | 2= Akima |
| | 3= chords |
| <Degrees>: | 1,..., 5 |

Leading zeros may be omitted.

<Members> defines the coordinate or axis names involved in the spline motion.

- If working range coordinate programming (COORD(..)) is active, the orientation motion can also be programmed as a spline with:
  – Orientation ”O” or
  – Polar coordinates ”phi” and ”theta”
- Coordinates/axes not listed as <Members> can only be moved on a straight line.

  **Examples**:
  SplineDef(2203,**”x”,”y”,”z”**)
  SplineDef(2203,”x”,”y”,”z”,”O”)
  SplineDef(2203,”x”,”y”,”z”,”phi”,”theta”)

## Activation

G99 Activating the “Spline” type of path

## Modale parameters for G99

- **Coordinate/axis programming**

  The end points of the channel coordinates are programmed. All members listed in SplineDef are moved along the spline curve, the remaining coordinates move on a straight line.

| | |
|---|---|
| <CoordName>(<EndPos>) and/or<br><AxisName>(<EndPos>) and/or<br><Orientation coordinate>(<EndOrientation>) | Programming of individual coordinates/ axes/orientation coordinates and their values. |

where

| | |
|---|---|
| <EndPos> | End position of the coordinate/axis. |
| <EndOrientation> | Orientation in polar angles or Cartesian coordinates. |

**Example**: coordinates: x, y, z and orientation coordinate theta

```
SplineDef(2203,”x”,”y”,”z”,”theta”)
G99 x10 y20 theta30
```

**Example**: Axes: X,Y, U

```
SplineDef(1213,”X”,”Y”)
G99 X10 Y10 U20
```
(X, Y move as Spline, U linearly)

- **Starting conditions:**

| | |
|---|---|
| SBC(<Type>[,<Values>]) | Boundary conditions for the starting point of a spline sequence<br>$C^1$ with 3 starting conditions<br>$C^2$ with 5 starting conditions |

where

<type> **Default**: 2

1 (Valid for $C^1$ and $C^2$)
Specification of tangential direction at the starting point of the spline sequence. A value must be entered in the <Values> list for each spline member.

2 (Valid for $C^1$ and $C^2$)
Specification of second derivative at the starting point of the spline sequence. A value must be entered in the <Values> list for each spline member.

3 (Valid for $C^2$)
De Boor's boundary condition, links the second derivatives at the first two interpolation points.
<Values> usually 1.

4 (Valid for $C^2$)
Periodic boundary condition: the last and the first point of the spline sequence are matching. SBC(4) necessarily requires EBC(4). The entire spline sequence must be in the look-ahead range, otherwise, a runtime error will be generated.

11 (Valid for $C^1$ and $C^2$)
First spline starts tangentially relative to the previous linear block.

<Values> **Default:** 0,...,0

All information entered in <Values> taken together indicates the direction and size of the starting tangent or the second derivative at the starting point. A positive or negative value may be entered for each spline member.
Type 11 does not need any <Values>.

**Example:**
SBC(1,**1.0,1.0,0.2**), if SplineDef(1213,”X”,”Y”,”B”)

- **End conditions:**

EBC(<Type>[,<Values>]) Boundary conditions for the end point of a spline sequence
$C^1$ with 3 end conditions
$C^2$ with 5 end conditions

where

<type> **Default**: 2

1 (Valid for $C^1$ and $C^2$)
Specification of tangential direction at the end point of the spline sequence. A value must be entered in the <Values> list for each spline member.

2 (Valid for $C^1$ and $C^2$)
Specification of second derivative at the end point of the spline sequence. A value must be entered in the <Values> list for each spline member.

3 (Valid for $C^2$)
De Boor’s boundary condition, links the second derivatives at the last two interpolation points. <Values> usually 1.

4 (Valid for $C^2$)
Periodic boundary condition: the last and the first point of the spline sequence are matching.
EBC(4) necessarily requires SBC(4). The entire spline sequence must be in the look-ahead range, otherwise, a runtime error will be generated.

11 (Valid for $C^1$ and $C^2$)
The last spline leads tangentially to the subsequent linear block.

<Values> **Default:** 0,...,0

All information entered in <Values> taken together indicates the direction and size of the end tangent or the second derivative at the end point. A positive or negative value may be entered for each spline member.
Type 11 does not need any <Values>.

**Example:**
EBC(1,**1.0,1.0,0.2**), if SplineDef(1213,”X”,”Y”,”B”)

- **Spline parameter length:**

The spline parameter length is calculated by the NC from the specified interpolation points. The process (parameter setting) defined in the spline ID is used for this purpose. The spline parameter length may also be programmed, if necessary.

{PL<$w_e$>} **optional:** Programming the spline parameter length if the selection of the “parameter setting” in SplineDef(..) is to be overwritten.

where

<$w_e$> any value > 0

## B-Splines (Type 3, NURBS)

### Initialization

SplineDef(<Id>,<Members>) Definition of the spline ID and the spline members.

where

<Id> four-digit integer, consisting of <Type> <Parameter settings><Tangent computation><Degrees>

| | |
|---|---|
| <Type>: | 3= B-Spline |
| <Parameter setting>: | 1= equidistant<br>(= uniform B-Spline: is often used in practice)<br>2= chordal<br>3= centripetal |
| <Tangent computation>: | 0=irrelevant |
| <Degrees>: | 1,..., 5 |

Leading zeros may be omitted.

<Members> defines the coordinate or axis names involved in the spline motion.

- If working range coordinate programming (COORD(..)) is active, the orientation motion can also be programmed as a spline with:
  – Orientation ”O” or
  – Polar coordinates ”phi” and ”theta”
- Coordinates/axes not listed as <Members> can only be moved on a straight line.

  **Examples**:
  SplineDef(3103,**”x”,”y”,”z”**)
  SplineDef(3103,”x”,”y”,”z”,”O”)
  SplineDef(3103,”x”,”y”,”z”,”phi”,”theta”)

## Activation

G99 Activating the “Spline” type of path

## Modal parameters for G99

- **Coordinate/axis programming**

  The end points of the channel coordinates (checkpoints) are programmed. All members listed in SplineDef are moved along the spline curve, the remaining coordinates not defined in SplineDef move on a straight line.

| | |
|---|---|
| <CoordName>(<EndPos>) and/or<br><AxisName>(<EndPos>) and/or<br><Orientation coordinate>(<EndOrientation>) | Programming individual checkpoints (coordinates/axes) and their values. |

  where

| | |
|---|---|
| <EndPos> | End position of a checkpoint (coordinate/axes) |
| <EndOrientation> | Orientation in polar angles or Cartesian coordinates. |

  **Example**: coordinates: x, y, z and orientation coordinates
  SplineDef(3103,”x”,”y”,”z”,”O”)
  G99 x10 y20 z30 O(0.1,0,1.0)

  **Example**: Axes: X,Y, U
  SplineDef(3102,”X”,”Y”)
  G99 X10 Y10 U20 (X, Y move as splines, U on a straight line)

☞ **It is not possible to program start or end conditions.**

- **Spline parameter length:**

  The spline parameter length is automatically calculated by the NC from the specified checkpoints. The process (parameter setting: =1) defined in the spline ID is used for this purpose. The spline parameter length may also be programmed, if necessary.

| | |
|---|---|
| {PL<$w_e$>} | **optional:** Programming the spline parameter length if the selection of the “parameter setting” in SplineDef(..) is to be overwritten. |

  where

| | |
|---|---|
| <$w_e$> | any value > 0 |

- **Spline point weighting for checkpoint of B-splines:**

{PW<$w_e$>} **optional:** Programming point weightings (the splines may be modified in the vicinity of a checkpoint).

where

<$w_e$> **Default:** 1

$0 < w_e < 1$: **pressing** the spline away from the checkpoint

$w_e >$ 1: **pulling** the spline towards the checkpoint

**Example**: coordinates: x, y, z and orientation coordinates

```
SplineDef(3103,”x”,”y”,”z”,”O”)
G99 x10 y20 O(0.1,0,1.0) PW2.3
```

## 3.57 Zerosetting of modulo axis (linear endless axis) G105

Effect

Using the G105 ”modulo axis zerosetting” function, the point of reference (program zero point) of a **linear endless axis** can be determined. As soon as the modulo value is reached, the actual value of the linear endless axis is automatically set to **zero**. This modulo calculation prevents an overflow of the axis values and enables the axis to travel at ”endless”.

G105 determines the program zero point. Using this point the control unit calculates the distance from the zero point of the command-value system. The resulting offset is internally added to all subsequent values.

### Modulo value

The modulo value should be as long as possible (e.g. 20 m) in order to have a large programming range available. The modulo value is taken over by the drive using ID no. S-0-0103 **already** during SERCOS run-up. A modulo value change can only take effect after rebooting SERCOS.

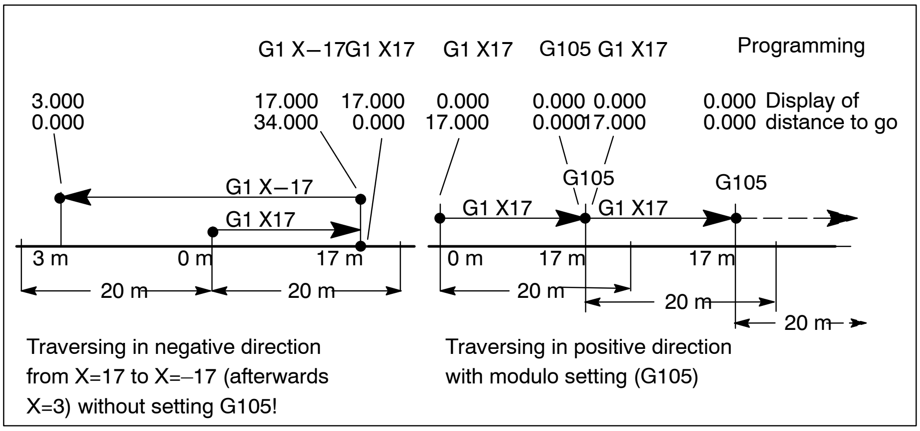
### Traversing range

The control unit will **not** permit any programming of positions greater than the modulo value.

A linear endless axis can also travel backwards. Negative input is possible as long as the amount is smaller than the modulo value.

If an endless axis traverses with a negative value (e.g. X–17), the end point will automatically be transformed into a positive X=3 end position as soon as it is reached.

**Example:** Linear endless axis with modulo value = 20 m



Traversing in negative direction from X=17 to X=–17 (afterwards X=3) without setting G105!

Traversing in positive direction with modulo setting (G105)

☞ **The on-the-fly measurement (G175/G275) can be used for linear endless axes if the programmed positions have a positive sign. Backward travelling using the probe (programming of negative positions) will not provide unique values.**

## Programming

G105

G105 sets the program zero point for all internal linear endless axes configured in MACODA parameter 1003 00004.

G105 X..

Sets the program zero point and programs a traversing movement (e.g. X..) which refers already to the new zero point. One or more axes can be traversed.

G105 LinModAxis<physical axis index>

G105 sets the program zero point only for the linear endless axis with the physical axis index (1..n) configured in MACODA parameter 1003 00004.

G105 LinModAxis<physical axis index>X..

G105 sets the program zero point only for the linear endless axis with the physical axis index (1..n) configured in MACODA parameter 1003 00004 and programs a traversing movement (e.g. X..) for one or more axes.

**Example**: Programming the linear endless axis

- N.. G105 — Setting the program zero point of all linear endless axes.
- N.. G105 X200 — Setting the program zero point of all linear endless axes, and traversing the X axis to position 200 following zerosetting.
- N.. G105 LinModAxis1 — Setting the program zero point of the linear endless axis with the physical axis index of 1.
- N.. G105 LinModAxis1 X–200
  Setting the program zero point of the linear endless axis with the physical axis index of 1, and traversing the X axis to –200.

**Please note for G105:**

- The workpiece position is always calculated in terms of modulo:
  0 <= X < Xmod
- If the modulo range is exceeded, Xmod on the actual value display will jump to 0 or from 0 to Xmod. The setpoint value jumps only from 0 to Xmod because Xmod cannot be exceeded in the other direction.
- The stored axis offset is deleted upon control reset, i.e. the program zero point coincides with the axis zero point.
- The program value always indicates the position last programmed.

☞ **For linear endless axes, MACODA parameter 1003 00004 must be set to 4.**

## 3.58 Consideration of the existing braking distance with active path slope G112, G113

Effect

The "**Consideration of the existing braking distance with active path slope**" function looks ahead at the respective following block and lowers the final speed of the current block of block preparation to such an extent that a speed of V = 0 can be reached at the end of the following block.

Programming

| | |
|---|---|
| G112 | Deactivate braking-distance consideration. |
| G113 | Activate braking-distance consideration. |

**Please note for G112 and G113:**

- Programming G112 requires an active G8. The G112 and G113 functions act modally and cancel each other mutually.

☞ **In the case of short blocks, the restricted lookahead may cause a feedrate reduction although this may not be necessary from a geometry perspective.**

## 3.59 Feed forward control G114, G115

Effect

Contour errors are primarily caused by system dependent lag. The current lag depends on the feedrate in the steady state of the axis, and in the acceleration phase on the acceleration as well.

The feed forward control will correct interpolator command values of the CNC in such a manner that the lag is reduced. This enables a more accurate contour to be achieved.

☞ **The feed forward control function is integrated in the drive software according to the manufacturer's specifications and is only activated by the PNC via SERCOS interface. For a detailed description of the function, please refer to the drive documentation.**
**If the drives being used support feed forward control the function has to be released for the corresponding axes via MACODA parameter 1003 00009.**

Upon activation, the drive will switch to ”secondary mode 1” (ID no. S-0-0033; bit 3 set = position control excluding following error).

Programming

G114: Activate feed forward control.

G114 programmed **without** axis addresses; **all** axes will be switched to secondary mode 1 if MACODA parameter 1003 00009 is set for these axes.

G114 X.. Y.. G114 programmed **with** axis addresses; **the programmed** axes will be switched to secondary mode 1 if MACODA parameter 1003 00009 is set for these axes.

G115 Deactivate feed forward control.

G115 programmed **without** axis addresses; **all** axes will be switched back to the main operating mode.

G114 X0 Y0 The **programmed** axes will be switched back to the main operating mode.

☞ **The feed forward control parameters (e.g. P-0-0500 or P-0-0501 for Servodyn-D drives) can be defined using function G900.**

**Example**: Programming the feed forward control

```
N10 G114                    activate feed forward control of all axes

N20 F1000 S500
N30 G1 X1800 Y800
.
N160 X1500 Y1500
N170 G2 I50
N180 G114 Z0                deactivate feed forward control for Z
.
N210 G115                   deactivate feed forward control of all axes
M30
```

---

⚠

**DANGER**
**This programming might result in damage to the workpiece and/or the machine! There might even be danger to persons!**

**This programming refers directly to a real physical axis. A logical axis addressed, for instance, by a coordinate transformation (e.g. inclined plane) with the same axis address will lead to incorrect axis values.**

---

## 3.60 Tangential tool guidance ON G131<br>Tangential tool guidance OFF G130

Effect

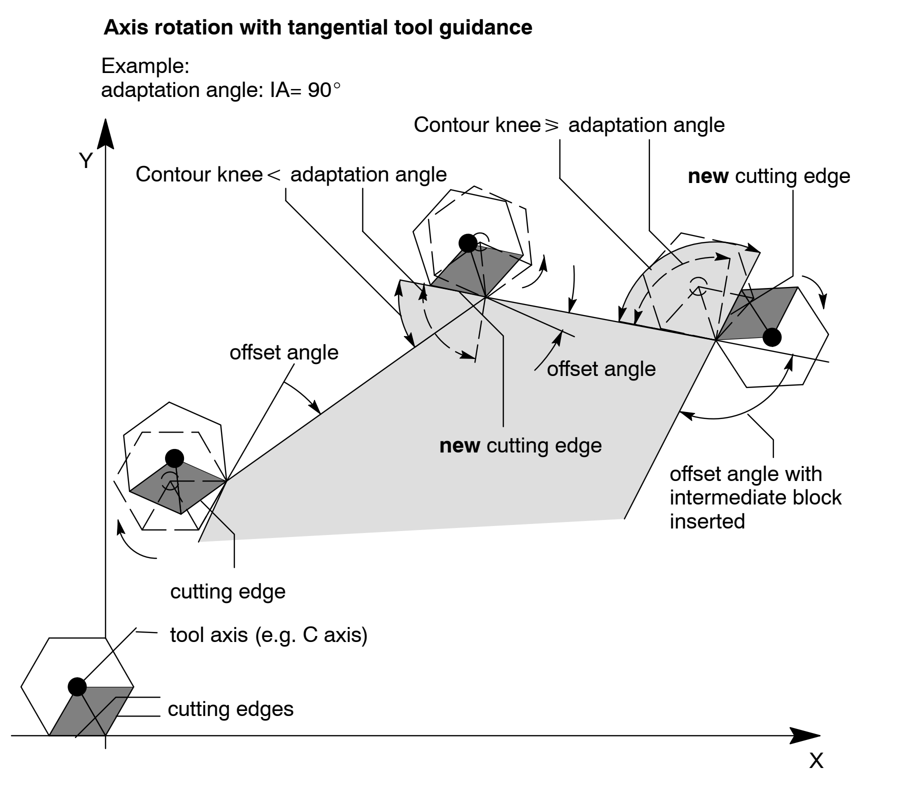
Tangential tool guidance allows to approach a path on a selected plane with a **tool axis** at a specified **offset angle** with the path. The tool axis is in 0° position if it is set tangentially (offset angle = 0°) to the main axis traversing in a positive direction.

In the case of a circular path, the offset angle with the tangent to the circular path is calculated in accordance with the interpolator clock pulse. Consequently, the tool axis keeps turning by the respective offset angle calculated with each interpolator clock pulse.

In the course of the execution of all blocks or block segments, the tool axis reaches its full tangential angle at the **starting point** of the path (unlike in the case of the function ”Tangential tool orientation”, refer to section 3.85).

Depending on the **adaptation angle** parameter, an intermediate (adaptation) block is automatically inserted between two NC blocks:

- If the angle at the contour knee is **wider** than the adaptation angle programmed, a tool rotation block (adaptation block) is automatically inserted, which rotates the tool axis towards the new starting tangent.
- If the angle at the contour knee is **smaller** than the adaptation angle programmed, the tool axis jumps to its new position at the beginning of the block.


Axis rotation with tangential tool guidance
Example:
adaptation angle: IA= 90°
Contour knee ≥ adaptation angle
Y
Contour knee < adaptation angle
new cutting edge
offset angle
offset angle
new cutting edge
offset angle with intermediate block inserted
cutting edge
tool axis (e.g. C axis)
cutting edges
X

## Programming

**G131:** Tangential tool guidance ON

```
G131 {TAX{=}<Axis>} {SYM{=}<s>} {ANG{=}<a>}
     {IA{=}<Aa>} {PLC{=}<p>}
```

where:

TAX TAX (tool axis) is used for programming the axis which is to approach the path.

Axis Designation of the axis to which the function ”Tangential tool guidance” is to apply. You may enter either the **logical** axis name, or the **physical** axis name, or the **logical axis number**. CPL terms are also permitted.

SYM SYM defines the symmetry value s. Indicates the tool symmetry (number of cutting edges). A tool with the symmetry value s returns to its original position after a rotation of 360°/s.

s Symmetry value: any integer except 0.

**s=1**: The tool is asymmetric having 1 tool edge. The tool edge is run along the contour with the offset angle taken into account.

**s>1**: The tool is symmetric having several, equally spaced tool edges. If there is a knee in the contour, the tool is rotated just enough for the nearest tool edge to be positioned at the offset angle with the contour. “s” is dependent on the kind of tool being used, i.e. with a rectangular tool, s = 2, with a square tool, s = 4, etc.

**s<0**: A negative symmetry has the effect that the tool is not rotated if a reversal of direction of motion (180° knee) occurs, irrespective of the offset angle. In every other respect tool operation is the same as in the case of a positive symmetry.

ANG ANG is used for programming the offset angle.

a Offset angle [–180° .. 180°] The offset angle indicates the angular offset between the path and the tool.

IA IA is used to program the adaptation (intermediate) angle (Aa)

Aa Adaptation angle [0° .. 180°]: The adaptation angle specifies from how many degrees upwards of a contour knee angle an intermediate block is inserted to rotate the tool axis. If the angle at the controur knee is **smaller** than the limit thus specified, **no** intermediate block is inserted to rotate the tool axis. Instead, at the start of the next block, the tool jumps to its new position.

PLC PLC is used to switch NC-PLC communication on and off while an intermediate block is being executed.

p **p=0**: NC-PLC communication is switched off while an intermediate block is being executed. The NC executes the rotation block unconditionally.
**p=1**: Execution of a rotation block is controlled via NC-PLC communication.

The programmed parameters may be omitted. In this case, they will be initialized by the following MACODA parameters:

| | | |
|---|---|---|
| ● 7050 00210: | number of the tool axis | (TAX) |
| ● 7050 00220: | symmetry value | (SYM) |
| ● 7050 00230: | adaptation angle | (IA) |
| ● 7050 00240: | offset angle | (ANG) |
| ● 7050 00260: | NC-PLC communication | (PLC) |

Programming

**G130:** Tangential tool guidance OFF

**Please note for G130, G131:**

- The functions ”Tangential tool guidance” and ”Tangential tool orientation” (G630, G631 or TTON/TTOFF) must **never** be active **simultaneously**.
- G131 does not produce a traversing motion after power-up.
- G131 must never be programmed together with an axis motion in the same block (error message!).
- Approach motions of tool axes programmed with G131 are executed only together with the next traversing movement to be carried out. Depending on the adaptation angle,
  - a rotation block is executed first, or
  - the tool jumps to its new position at the beginning of the next block.
- If no offset angle is entered when programming G131, please note the following:
  - the current rotary axis angle is taken to be the offset angle, or
  - the angle preset in MACODA parameter 7050 00250 is applied as the offset angle.

  In MACODA parameter 7050 00250, you may select one of the above options.

**Syntax examples:**

| | |
|---|---|
| G131 | Approach movements of all axes are executed with the SYM, ANG and IA initialization values of MACODA. |
| G131 TAX=C SYM1 ANG90 IA20 PLC0 | Programming with logical axis name |
| G131 TAX3 SYM=1 ANG90 IA20 PLC1 | Programming with logical axis number |
| G131 TAX[NAME$] SYM1 ANG=90 IA20 | Programming with CPL variable |

## NC-PLC interface signals

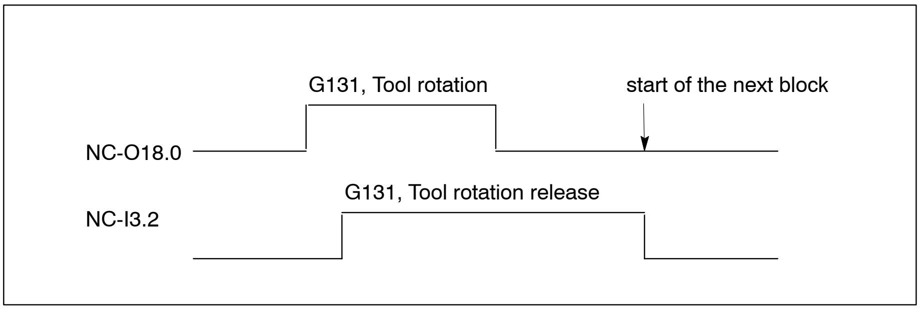
The PLC is able to control the execution of the intermediate block if NC-PLC communication is active (PLC=1).

Channel output signal NC–>PLC:

- **NC-O18.0, "G131, Tool rotation"**
  A signal is sent to the PLC indicating that the current angle between two blocks is wider than the " adaptation angle" (IA). The NC does not execute the intermediate block before it receives an acknowledgement from the PLC. The signal is not reset before the intermediate block is executed.

Channel input signal PLC–>NC:

- **NC-I3.2, "G131, Tool rotation release":1**
  The PLC signals the release of the execution of the intermediate block to the NC. After the execution of the intermediate block, the NC will not continue to execute any of the following blocks before the signal is reset.



## 3.61 Workpiece position compensation G138, G139

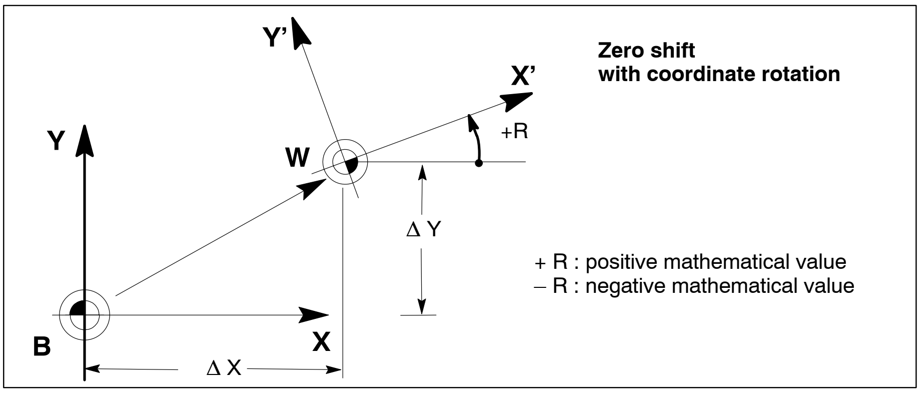
Effect

The ’workpiece position compensation’ function unlinks coordinates of part programs P from the basis workpiece coordinate system B. This function acts on the **first 3** logical axes on the respective channel.

In contrast to a zero shift function, also the **1st** and **2nd coordinates** can be rotated here around the 3rd coordinate. This allows you to adapt the coordinate system to any workpiece position.

In the course of the execution of a part program, all the programmed traversing movements will then be referring to the ”new” – offset and rotated – coordinate system.

Workpiece position compensation is only possible in the valid working area of the machine and acts additively to active zero shifts!


Y’
X’
Zero shift
with coordinate rotation
+R
Y
W
Δ Y
+ R : positive mathematical value
– R : negative mathematical value
B
X
Δ X

Programming

G138 Switch on workpiece position compensation.
In the beginning of the part program, the following is programmed in the same block as G138:
**Offset** of the workpiece zero point in X, Y and Z direction including the corresponding axis address, and
**Angle of rotation** of the 1st and 2nd axis (standard axis addresses: X and Y) as R address (value range: –360° < angle of rotation <360°).
All programming values must be **absolute machine coordinates**.

G139 Switch off workpiece position compensation.

**Please note for G138 and G139:**

- G138/G139 act modally and cancel each other mutually.
- If workpiece position compensation is active, G37, G38, G54 – G59, G154 – G159, G254 – G259, G60, G160 – G360, G168, G268, G145 – G845, G147 – G847 as well as tool-length compensation Hx will be taken into account.
- You can program the ”Inclined plane” function, G352, G354..G359, together with ”Workpiece position compensation”. ”Inclined plane” acts additively on the workpiece position compensation.

- G38, ”Scaling”, has no impact on the parameters of the inclined plane function.
- The axis addresses under G138 refer to (basis) workpiece coordinates.
- G138/G139 must never be programmed in combination with a traversing motion.
- Programming G138/G139 will interrupt the look-ahead function. Therefore, G138/G139 must never be programmed while the cutter compensation function, G41/G42, is active. If required, the workpiece position must be programmed before activating the cutter compensation function.
- The current workpiece position is taken into account in the display of workpiece coordinates.

**Example:** Calling workpiece position compensation

```
N... G90 G17 F1000 S250
...
N... G138 X50 Y300 Z10   Set workpiece zero point to machine coor-
     R1.23               dinates X50 Y300 Z10 and rotate X/Y
N...                     plane counter-clockwise by 1.23 degrees.
N...
N...
N... G139                Switch off workpiece position compensa-
                         tion.
```

## 3.62 External tool compensation G145, G146, G245- G845

Effect

Activation of one out of 8 external compensation pairs for radius and length compensation.
For this purpose, the respective compensation values must be imported from the PLC (application, e.g., for multiple compensation in the case of combination tools).

The effective radius or length compensation value is then equal to the total of any active geometry compensation table values plus the activated external tool compensation pair.

If the PLC changes the currently active external compensation values in the course of the execution of a part program, this change will only become active in the **block being under preparation** as the **next** block. Under certain circumstances this can mean that even more blocks are due to execution without this change.

In order to avoid this effect you must program the ”**WAIT**” CPL command directly after the block causing the PLC to hand over the new compensation values. By doing so you hold the block preparation of the PNC until all program blocks ahead of ”WAIT” have been executed.

Subsequently, in the program block following ”WAIT”, the new compensation values will already be active (please refer to Example 2).

Programming

| | |
|---|---|
| G145..G845 | External tool compensation on. |
| G146 | External tool compensation off. |

**Please note for G145 ... G845 and G146:**

- G145 ... G845 / G146 act modally and cancel each other mutually.
- G145 ... G845 / G146 can be programmed in the same block as other preparatory functions, axis information and auxiliary function.
- G145 ... G845 / G146 do not cause a traversing movement if they have been programmed in a separate block.

**Example 1:**

| | |
|---|---|
| `N... G0 X0 Y0 Z0` | |
| `N... H0` | table length compensation OFF |
| `N... G146` | external tool compensation OFF |
| `N... G1` | |
| `N... H1` | table length compensation 1 ON |
| `N... X10 Y10 Z10` | traversing movement with table length compensation 1 |
| `N... G145` | External tool compensation G145 ON |
| `N... X20 Y20 Z20` | traversing movement with table length compensation 1 plus external tool compensation G145 |
| `N... G345` | external tool compensation G145 OFF and G345 ON |
| `N... X30 Y30 Z30` | traversing movement with table length compensation 1 plus external tool compensation G345 |

| | |
|---|---|
| `N... H0` | table length compensation OFF |
| `N... X40 Y40 Z40` | traversing movement with external tool compensation G345 |
| `N... G146` | external tool compensation OFF |
| `N... X0 Y0 Z0` | traversing movement without compensation |

**Example 2:**

| | |
|---|---|
| `N... G145` | external tool compensation G145 ON |
| `N... M...` | The M function will cause the PLC to perform the following cycle:<br>1. Transfer new compensation values<br>2. Send acknowledgement to CNC<br>(the CNC will interpret this acknowledgement as ”Block or subprogram with ’M’ has been executed”). |
| `10 WAIT` | Block preparation will be suspended until ”M” has been executed. |

Length compensation will be in the direction of the selected drilling axis (please refer to G78/G79).

The active tool compensation is displayed at the ”External tool compensation bit 0 ... bit 3” channel interface. (please refer to “PLC Project Planning” manual).

## 3.63 General tool compensation G147, G148, G247- G847

Effect

**General tool compensation** is available as the 2nd external tool compensation function (2nd compensation group). It may be used with drilling, milling, turning and anglehead tools and may be activated in addition to and independent of the “external tool compensation” function (G145, G146, G245–G845).

The compensation data is stored in a **compensation data set** that may include the following parameters as a maximum:

| | |
|---|---|
| ● $\mathbf{L_{2(1)}, L_{2(2)}, L_{2(3)}}$: | Length compensation or offset |
| ● **R:** | Radius |
| ● **TO:** | Tool orientation (edge position) |
| ● $\varphi$ (phi), $\vartheta$ (theta), $\psi$ (psi)**:** | Eulerian angles, for orientation compensation, e.g. for gripping devices |

You can program the selection of **one** compensation data set from a total of 8.

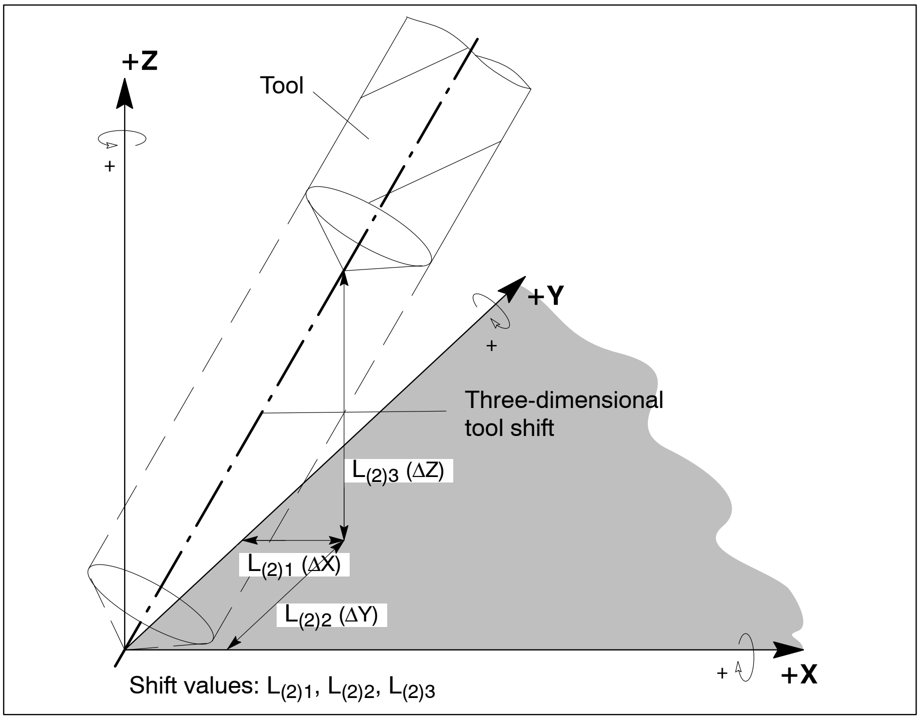
**L1, L2 and L3 length compensation parameters and/or shift parameters:**

With a total of 3 shift values, L1, L2 and L3, you can perform both constant three-dimensional tool shifts and parallel length compensations of 3 different tools as a maximum.

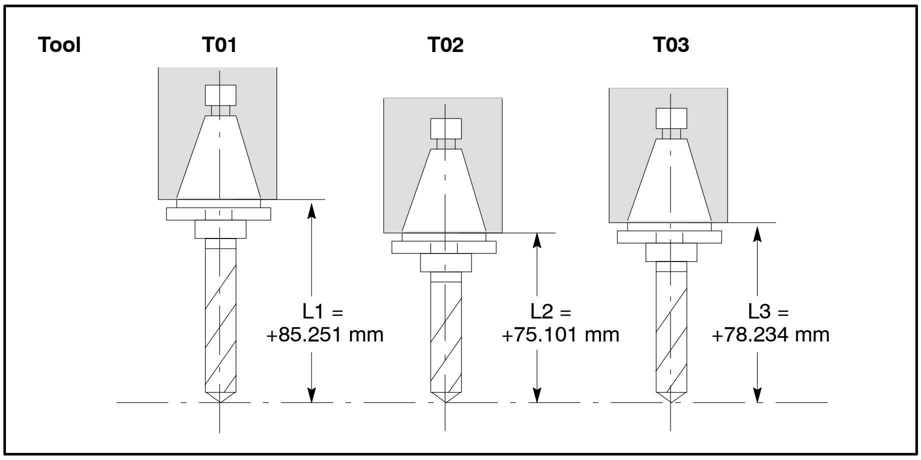
**Example 1:** Three-dimensional tool shift

L1, L2 and L3 shift values are assigned to the respective axes via MACODA parameter 7050 00420 (also refer to G78, G79 in sect. 3.47). The control unit will check internally whether or not the assignments are correct.

| Shift parameter | Assignment to logical axis names |
|---|---|
| L1 | X |
| L2 | Y |
| L3 | Z |


+Z
Tool
+
+Y
+
Three-dimensional tool shift
$L_{(2)3}$ (ΔZ)
$L_{(2)1}$ (ΔX)
$L_{(2)2}$ (ΔY)
+
+X
Shift values: $L_{(2)1}$, $L_{(2)2}$, $L_{(2)3}$

**Example 2:** Parallel length compensations of up to 3 different tools



Assignments, which will be internally checked for correctness, are to be entered in MACODA parameter 7050 00420 as follows:

| Shift parameter | Assignment to logical axis names |
|---|---|
| L1 | Z1 (drill axis 1) |
| L2 | Z2 (drill axis 2) |
| L3 | Z3 (drill axis 3) |

☞ **These axes must be different from the drill axis on which the external tool compensation function (G145 ...) or the length compensation function H act (external tool compensations and length compensations always act additively on one and the same axis).**

### Parameter R for radius compensation:

If the general tool compensation and the external tool compensation (G145 ... G845) have been activated in the same NC block, the radius set in the general tool compensation function will be applied invariably (refer to Example 1 below).

While the cutter compensation function G41/G42 is active, the radius values set in general tool compensation (G147 ff.) and those set in external tool compensation (G145 ff.) automatically cancel each other, i.e. only the value that was activated last takes effect (refer to Example 2 below). The radius value that was activated last always acts additively on any D word (radius compensation) that may have been programmed.

The required compensation values must be specified by the PLC. Any changes in the currently active compensation values made by the PLC in the course of the execution of a part program will take effect only in the next block to be prepared. Therefore, it may happen that a number of blocks are executed before a change in a compensation value takes effect.

In order to avoid this effect you must program the ”**WAIT**” CPL command directly after the block causing the PLC to hand over the new compensation values. Block preparation by the PNC will be suspended until all program blocks before the ”WAIT” command have been executed. In the first NC block after the ”WAIT” command, the new compensation values will be taken into account.

### Example 1:

| | |
|---|---|
| `N10 G147 G145 X.. Y..` | The radius value set in the general tool compensation function will take effect! |

### Example 2:

| | |
|---|---|
| `N10 G147` | The radius value set in the general tool compensation function is effective! |
| `N20 G145  X.. Y..` | The radius value set in the external tool compensation function will take effect! |

### TO tool orientation (edge position) parameter:

The tool orientation (edge position) parameter (TO) describes the principal orientation of the tool. The tool edge position compensation is defined in conjunction with the cutter radius (R). The cutter position compensation is needed in connection with path compensation (G41/G42) in order to ensure a faultless contour when processing the workpiece with milling tools and movements that are not in parallel to the machine axis.
For details, please refer to “PNC Description of Functions” manual.

**Eulerian angles φ (phi), ϑ (theta), ψ (psi):**

For special tools (e.g. certain gripper types), orientation compensation may be required in addition to length compensation. The gripper coordinate system can thus be offset and rotated in any way with respect to the flange coordinate system (tool holder).
For details, please refer to "PNC Description of Functions" manual.

☞ **Orientation compensation is not possible unless the appropriate axis transformation is active which takes the rotation internally into account.**

Programming

| | |
|---|---|
| G147..G847 | general tool compensation ON |
| G148 | general tool compensation OFF |

**Please note for G147... G847and G148:**

- G147 .. G847/G148 act modally and cancel each other mutually.
- G147 .. G847/G148 can be programmed in the same block as other preparatory functions, axis information and auxiliary function.
- G147 .. G847/G148 do not cause a traversing movement if they have been programmed in a separate block.
- If the general tool compensation and the external tool compensation (G145 ... G845) have been activated in the same NC block, the radius set in the general tool compensation function will be applied invariably. If the radius of the external tool compensation is to be taken into account, this function must be programmed separately in the next NC block.
- The general tool compensation is displayed at the channel interface "General tool compensation", bit 0 ... bit 3 (please refer to the "PLC Project Planning" manual).

## 3.64 Changing the positioning type for endless axes G150, G151 Local setting of the positioning type for endless axes DC(..), ACP(..), ACN(..)

Effect

The PNC allows the positioning type for endless axes (type: **rotary** or **endless**) to be configured in a very flexible manner (please refer to G90/G189, G151/G150 and MACODA parameters 1003 00005, 1003 00050).

The following table shows various settings and switchover options for the positioning type of an endless axis:

| Presetting in MACODA parameter **1003 00005** (how does endless axis translate the programmed value into a motion) | Presetting in MACODA parameter **1003 00050** (changeover with **G151**:<br>1= yes<br>0 = no) | Switchover via **G90/G189** | Switchover via **G150/G151** e.g. B axis<..><br>0=no special logic<br>1=shortest path<br>2=sign logic | Block-by-block switchover via **DC(..), ACP(..), ACN(..)** (applies only to linear interpolation G0, G1) | **Axis** traverses using positioning type: |
|---|---|---|---|---|---|
| 0 | 0 | | | | no special logic |
| 1 | 0 | | | | shortest path |
| 2 | 0 | | | | sign logic |
| 3 | 0 or 1 | G90 | | | sign logic |
| 3 | 0 or 1 | G189 | | | shortest path |
| unequal to 3 | 1 | | G150: Switchover according to presetting in 1003 00005 | | – no special logic<br>or<br>– shortest path<br>or<br>– sign logic |
| unequal to 3 | 1 | | G151 B0 | | no special logic |
| unequal to 3 | 1 | | G151 B1 | | shortest path |
| unequal to 3 | 1 | | G151 B2 | | sign logic |
| 0, 1, 2 or 3 | 0 or 1 | | | DC(..) | shortest path |
| 0, 1, 2 or 3 | 0 or 1 | | | ACP(..) | sign logic<br>(in positive direction) |
| 0, 1, 2 or 3 | 0 or 1 | | | ACN(..) | sign logic<br>(in negative direction) |

## Switching over the positioning type for endless axes

Programming

G151 Change positioning type.
The axis address of the axis the positioning type of which is to be switched over is specified in the same block.
In addition to the axis address the desired positioning type is defined by the 0, 1 and 2 numerals:

0: no logic. The axis will subsequently always traverse without positioning logic to the respective last programmed position.

1: shortest path. The axis will always use the shortest path for traversing to the respective programmed position (traversing movement is always smaller than 180 degrees).

2: sign logic: The programmed sign determines the sense of rotation of the axis, the numerical value defines its position.

G150 The positioning type is switched back to the state programmed in MP 1003 00005.

**Example:**

| | |
|---|---|
| `G151 B0` | axis B: no logic. |
| `G151 A1 C2` | axis B: logic according to MP 1003 00005<br>axis A: shortest path<br>axis C: sign<br>. |
| `G150` | axes A, B and C according to MP 1003 00005 |

**Please note for G150 and G151:**

- G150/G151 act modally and cancel each other mutually.

## Local setting of the positioning type for endless axes

With local setting of the positioning type for endless axes you have the possibility of determining or switching over the positioning type of an endless axis block by block regardless of the MACODA parameter setting or active (modal) NC functions.

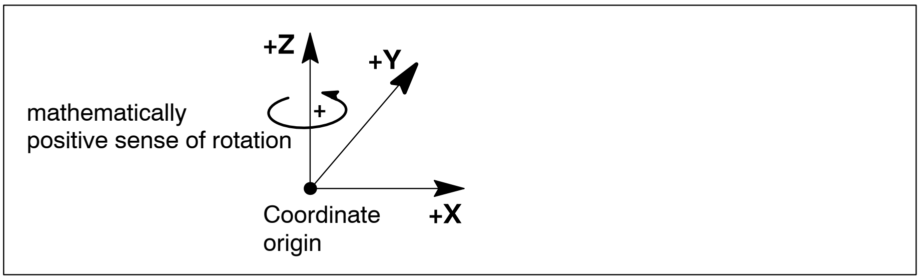
Programming

DC(...): the programmed position is approached on the shortest path.

ACP(...): the programmed position is approached in mathematically positive direction.

ACN(...) the programmed position is approached in mathematically negative direction.

Note: mathematically positive direction = counter-clockwise sense of rotation seen from a coordinate axis in the direction of the coordinates origin.



Programming

<physical axis address> = <address attribute>(<value>)

B = ACP (258) Irrespective of the presetting by G150/G151, the B axis will traverse in mathematically positive direction to the position 258 degrees.

Please note for the ACP function:

- The address attributes only act **block by block**.
- It is possible to program different attributes for different endless axes within one block.
- The evaluation of the address attributes is only performed for endless axes. They are ignored for other types of axis movement.
- The positioning type of endless axes only applies to G00, G01 linear interpolations (quasi-positioning mode). For other interpolation types, interpolation will include endless axes in analogy to a rotary axis.
- Only the amount of the axis value will be evaluated (negative sign will be ignored).
- The positioning type of endless axes will only be evaluated with absolute programming (G90).

**DANGER**
**This programming of the ACP function might result in damage to the workpiece and/or the machine! There might even be danger to persons!**

**This programming refers directly to a real physical axis. A logical axis addressed, for instance, by a coordinate transformation (e.g. inclined plane) with the same axis address will lead to incorrect axis values.**

## 3.65 External axis zero shift G160..G360, G167

Effect

You can perform one out of max. 3 external axis zero shifts for each applied machining axis (= synchronous axis).

For this purpose the PLC has to default the corresponding values. The **effective** axis zero shift will then correspond to the total of

- the active axis zero-shift values of the **axis ZS tables**, if any,
- the activated **external axis zero shift**.

If the PLC changes the currently active external offset values in the course of the execution of a part program, this change will only become active in the **block under preparation** as the **next** block. Under certain circumstances this can mean that several blocks are still due to execution without this change.

In order to avoid this effect you must program the ”WAIT” CPL command directly after the block causing the PLC to hand over the new shift values. By doing so you suspend block preparation by the PNC until all program blocks ahead of ”WAIT” have been executed.
Subsequently, in the program block following ”WAIT”, the new values will already be active.

Programming

| | |
|---|---|
| G160 | External axis zero shift no. 1 on. |
| G260 | External axis zero shift no. 2 on. |
| G360 | External axis zero shift no. 3 on. |
| G167 | External axis zero shift off. |

**Please note for G160, G260, G360 and G167:**

- G160, G260, G360, G167 act modally and cancel each other mutually.
- G160, G260, G360, G167 will not cause any traversing movement if programmed alone in a block.
- The compensation values under G160, G260, G360 refer to **machine or axis coordinate values**.
- The active ”external zero shift” will be displayed at the **External axis zero shift bit 0..bit 1** channel interface (also ref. to “PLC Project Planning” manual).

## 3.66 In-position at rapid travel G161, G162

Effect

During the control of a tool movement, an offset between the set and the actual values of the individual axes occurs during the movement owing to the dynamics of the machine.

In the case of positioning movements this effect has to be avoided if an accurate position is to be reached prior to the start of machining.

Using G161 you activate the ”In-position logic” especially for movements at rapid (for movement at feedrate, please refer to G61/62). Functions G164 to G166 can be used to set 3 different In-position logic options.

☞ **Please note that G161/G162 are superseded by an active G163!**

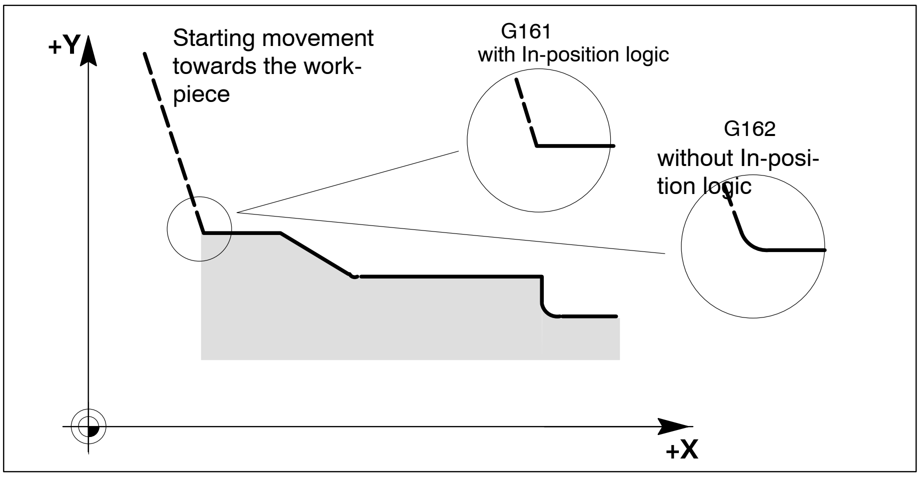
Programming

G161 In-position logic at rapid travel on.

G162 In-position logic at rapid travel off (only if G163 is not active).

Please note for G161 and G162:

- G161 and G162 act modally. M2/M30 sets the power-up state.
- G161 or G162 has to be programmed at the latest in the block to which the respective function is supposed to apply.


+Y
Starting movement towards the work-piece
G161
with In-position logic
G162
without In-position logic
+X

**Example**: Programming of G161/G162

| | |
|---|---|
| `N10 G161` | no movement; In-position logic ON |
| `N11 G0 Y200` | rapid travel with In-position logic |
| (or) | |
| `N10 G162` | rapid travel without In-position logic |
| `N11 G0 Y200` | |
| `N50 G161 X200` | rapid travel with In-position logic already in this block |

## 3.67 In-position logic mode G164, G165, G166

Effect

Using G164 to G166 you first determine the behavior of the ”In-position logic”. Subsequently, you activate ”In-position logic” via the G functions

- G61 (for movements at feedrate)
- G161 (for movements at rapid travel)
- G163 (for movements at feedrate **and** rapid travel).

Programming

G164 At the block end, the PNC reduces the path speed to V=0. It checks via the SERCOS interface whether the ”positioning window fine” (SERCOS ID no.: S-0-0057) has been reached for all axes involved. For this purpose, the ID no. S-0-0336 is assigned to the real-time bit 2 S-0-0307.
Only when this positioning window has been reached for all axes involved will the traversing movement of the next block be executed.

G165 At the block end, the PNC reduces the path speed to V=0. It checks via the SERCOS interface whether the ”positioning window rough” (SERCOS ID no.: S-0-0261) has been reached for all axes involved. For this purpose, the ID no. S-0-0341 is assigned to the real-time bit 2 S-0-0307.
Only when this positioning window has been reached for all axes involved will the traversing movement of the next block be executed.

G166 At the block end, the PNC reduces the path speed to V=0. Subsequently, the traversing movement of the next block is executed without performing a positioning-window check.

**Please note the following for G164, G165 and G166:**

- G164, G165 and G166 act modally. M2/M30 sets the power-up state.
- As long as the ’positioning window rough’ (G165) is selected, this is indicated at the **Inpos range 2 activated** channel interface (also ref. to “PLC Project Planning” manual).

☞ **The ”positioning window fine” and ”positioning window rough” parameters can be determined in the SERCOS files for Phase 3. For further details about the SERCOS files, please refer to the ”Configuration parameters and MACODA parameter description” manual under ”SERCOS initialization”.**

The following table shows the In-position logic behavior as a function of the different **interpolation types:**

- G1,G2 linear and circular interpolation
- G73 linear interpolation with In-position logic
- G0 rapid travel with In-position logic (with deceleration to V=0)
- G200 rapid travel without In-position logic (without deceleration to V=0)

as a function of the **modal functions**:

- G61 In-position logic at feedrate
- G62 In-position logic at feedrate off
- G161 In-position logic at rapid travel
- G162 In-position logic at rapid travel off (only if G163 is not active)
- G163 In-position logic at feedrate and rapid travel
- G164 positioning window, fine (V=0)
- G165 positioning window, rough (V=0)
- G166 without positioning window (V=0)

**InPos table:**

**Example cf. below**

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| InPos | 61 | 61 | 61 | 61 | 61 | 61 | 62 | 62 | 62 | 62 | 62 | 62 | 163 | 163 | 163 | 163 | 163 | 163 |
| **rapid travel InPos** | 161 | 161 | 161 | 162 | 162 | 162 | 161 | 161 | 161 | 162 | 162 | 162 | 161 | 161 | 161 | 162 | 162 | 162 |
| InPos window mode | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 |
| G1,G2 | 164 | 165 | 166 | 164 | 165 | 166 | - - - | - - - | - - - | - - - | - - - | - - - | 164 | 165 | 166 | 164 | 165 | 166 |
| **G73** | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 | 164 | 165 | 166 |
| G0 | 164 | 165 | 166 | 166 | 166 | 166 | 164 | 165 | 166 | 166 | 166 | 166 | 164 | 165 | 166 | 166 | 166 | 166 |
| G200 | 166 | 166 | 166 | 166 | 166 | 166 | - - - | - - - | - - - | - - - | - - - | - - - | 164 | 165 | 166 | 164 | 165 | 166 |

**Examples:** Using the table

The following In-position logic defaults are activated:

InPos 62: G62 (deactivate In-position logic function at normal feedrate) is selected

Rapid travel InPos 162: G162 (deactivate In-position logic function at rapid travel) is selected

InPos window mode165: G165 (positioning window, rough (V=0)) is selected

The above InPos settings determine the **behavior at block transition** for the following active G functions:

| | | |
|---|---|---|
| G1,G2 | - - -: | No ramping-down (exception: max. axis step change) |
| G73 | 165: | reaching the ”InPos window rough” is being waited for |
| G0 | 166: | ramping-down to speed V=0 |
| G200 | - - -: | No ramping-down (exception: max. axis step change) |

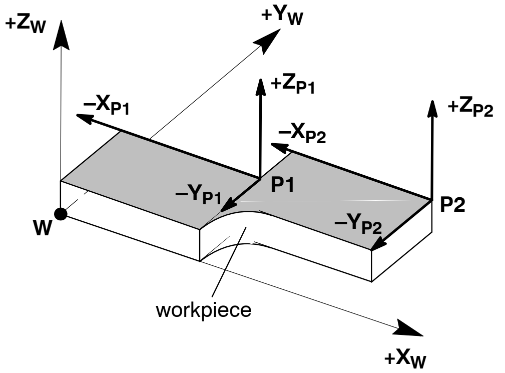
Despite G162 being active, ramping-down will be performed with G0. With regard to G0, G162 has the same effect as the combination of G161, G166.

## 3.68 Program coordinate shift G168, G169<br>Additive program coordinate shift G268, G269

Effect

All programmed coordinates of the feed axes (synchronous axes on a channel) of a part program or an MDI block refer to the program coordinate system (PCS or P). Therefore, a program zero point can be offset relative to a freely defined workpiece zero (WCS or W).

Shifting the program coordinate zero point allows the execution of part programs without making any changes in any position within the working range of the machine.



**P1** = Program zero point of the 1st program coordinate shift
**P2** = Program zero point of the 2nd program coordinate shift
**W** = Workpiece zero point

Additive shifting of the program coordinates allows to describe several successive coordinate systems and thus to design a part program that is equivalent to the dimensioning of a design drawing.

If one program coordinate shift is already active while a new one is being programmed, any axes for which no new values are entered will retain their previous shift values. This is the same behavior as with the ”Programmed contour shift” function, G60.

The program coordinate shift function may be used in the context of all NC functions defining coordinate system transformations and in particular together with zero shifts (G54 ... G259) or inclined plane (G352 ... G359).

However, the program coordinate shift function must always be deactivated when an inclined plane function is activated or deactivated.

**Difference between this function and “Programmed contour shift, G60/G67”:**

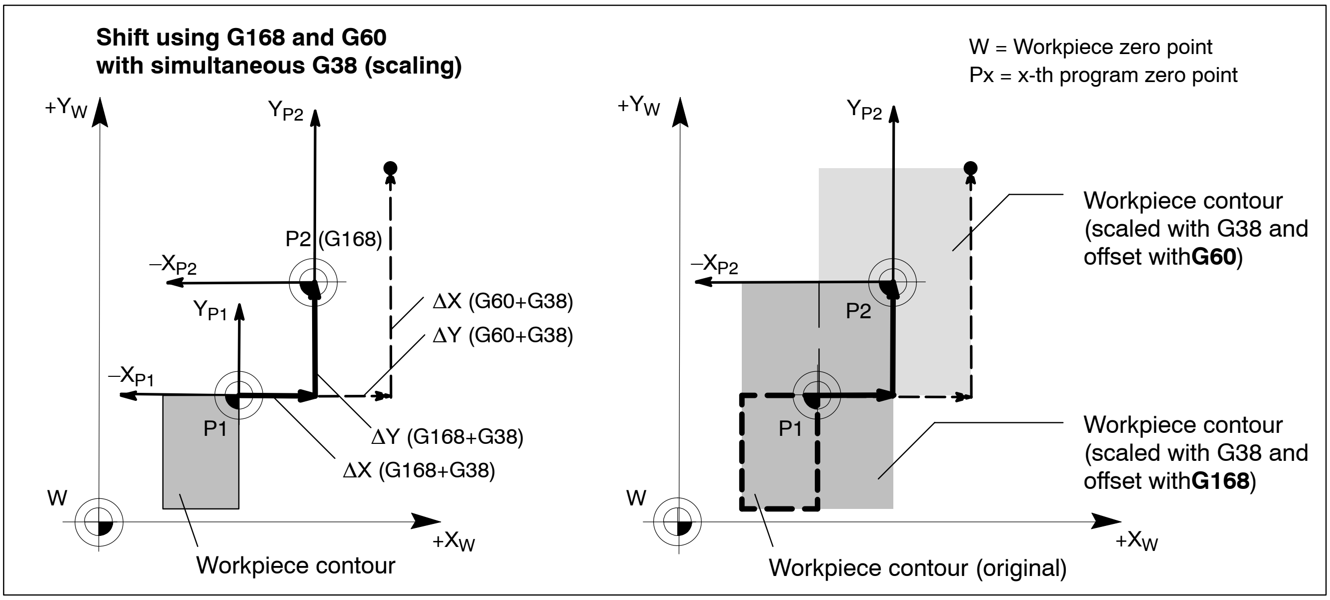
The functionality of ”Program coordinate shift” and ”Programmed contour shift”, G60, is the same with the **exception** of their behavior in combination with function **G38, ”Mirroring, scaling, rotating”**:

Whereas shift values programmed with G60 are impacted by G38 (shift values are also scaled, mirrored and rotated), G38 has no effect on any shifts made with the program coordinate shift function. Unlike G60, the program coordinate shift function has the effect of shifting coordinate systems.

**Example**: G168 versus G60, both times in combination with G38 (scaling)

The following shifts result with the scaling factor=2:

| **Axis** | **Scaling factor** | **Shift using G168 and G38** | **Shift using G60 and G38** |
|---|---|---|---|
| X | 2 | ΔX=1 unit | ΔX=2 units |
| Y | 2 | ΔY=1 unit | ΔY=2 units |


Shift using G168 and G60
with simultaneous G38 (scaling)
W = Workpiece zero point
Px = x-th program zero point
+YW
YP2
P2 (G168)
–XP2
YP1
–XP1
P1
ΔX (G60+G38)
ΔY (G60+G38)
ΔY (G168+G38)
ΔX (G168+G38)
W
+XW
Workpiece contour
+YW
YP2
–XP2
P2
P1
W
+XW
Workpiece contour (scaled with G38 and offset withG60)
Workpiece contour (scaled with G38 and offset withG168)
Workpiece contour (original)

Programming

G168 program coordinate shift ON

G169 **all** program coordinate shifts OFF

G268 **additive** program coordinate shift ON

G269 **additive** program coordinate shift **only** OFF

The desired shift values must be entered together with the pertinent G code (G168 or G268) and the axis addresses in one block that must not include any traversing motions.

Functions G168/G169 and G268/G269 form a modal group in each case and, therefore, cancel each other mutually.

**Example:** Programmed contour shift

| | |
|---|---|
| `N10 G168 X10 Y10 Z50`<br>`...` | Setting program zero at X10, Y10, Z50 of the current workpiece coordinate system. There is no traversing motion included in this block. |
| `N100 G1 X... Y... Z...` | Programmed positions refer to the program coordinate system defined above. |
| `N110 G268 X20 Y10` | Now, the program coordinate system is set at X30, Y20, Z50 relative to the workpiece coordinate system. There is no traversing motion included in this block. |
| `N200 G169` | The program coordinate system entered previously is deleted. Now, the program coordinate system is identical with the workpiece coordinate system. |

## 3.69 Torque reduction G177

Effect

The positive edge of the TORQUE REDUCTION axis interface signal can be used to set the max. torque of an axis to a value defined in MACODA parameter 1003 00010. The TORQUE LIMIT output signal indicates that the reduced torque has become effective.

G177 offers the possibility of overwriting the value as preset via MACODA parameter on an axis-per-axis basis within the value range from 0 to 500 (0 to 50% of the standstill torque) using program control. The value of the MACODA parameter itself is not changed.

Programming

`G177 X5` With the next positive edge of the ”torque reduction” signal the max. torque for the X axis in the drive will be limited to 0.5% of the standstill torque.

`G177 Y7` With the next positive edge of the ”torque reduction” signal the max. torque for the Y axis in the drive will be limited to 0.7% of the standstill torque.

`G177` With the respective next positive edge of the ”torque reduction” signal the max. torque for the individual axes will again be limited to the value determined in MACODA parameter 1003 00010.

When entering an invalid value the control unit will limit the input value to the admissible range (0 to 500) and output a warning.

☞ **In the case of a negative edge of TORQUE REDUCTION the max. torque will be set back to the value which was active in the drive after the last SERCOS phase startup. If this value is to be changed, it has to be entered in the corresponding SERCOS file under the SERCOS ID no. S-0-0092, and subsequently a phase startup has to be performed.**
**For further details about the SERCOS files, please refer to the ”Configuration parameters and MACODA parameter description” manual under ”SERCOS initialization”.**

**DANGER**
**Programming function G177 might result in damage to the workpiece and/or the machine! There might even be danger to persons!**

**This programming refers directly to a real physical axis. A logical axis addressed by a coordinate transformation (e.g. inclined plane) with the same axis address will lead to incorrect axis values.**

## 3.70 Speed limitation G192, G292

Effect

To make sure that the spindle speed does not rise or fall excessively (constant cutting speed G96), the upper or lower limit of the admissible speed range can be programmed in the part program. A speed change – even if initiated by the spindle potentiometer – will not be performed unless it is within the absolute limits specified.

The limit values apply to all speed ranges, however, they are effective only if they are within the speed range limits.

Programming

G192 Determine **lower limit** of admissible speed.
The spindle minimum speed is programmed in the same block as G192 with the S word.
G192 with an S word ≤ 0 cancels the limit value.

G292 Determine **upper limit** of admissible speed.
The spindle maximum speed is programmed in the same block as G292 with the S word.
G292 with an S word ≤ 0 cancels the limit value.

**Please note for G192 and G292:**

- G192 and G292 can be replaced by programming new limit values.
- The speed limits influence the direct speed programmed in G97 and the constant cutting speed in G96.
- The S values programmed with a speed limit do not influence the speeds in connection with M3/M4 programming. They remain stored and effective until M2/M30, ”Control reset”, ”Reset” or one of the cancelling functions is activated.

**Example:**Programming a speed limit

```
N...
N100 X... Y... G192 S1500    minimum speed: 1500 rpm
N101 X... Y... G292 S2500    maximum speed: 2500 rpm
N... X... Y... G292 S-1      cancel maximum speed
N... X... Y... G192 S0       cancel minimum speed
```

## 3.71 Oscillating axis G301, G350

Effect

Using the ”oscillating axis” function, an oscillating movement can be performed with any synchronous axis while linear interpolation is carried out for the other synchronous axes of the channel (e.g. flat grinding).

Any synchronous axis can be defined as oscillating axis:

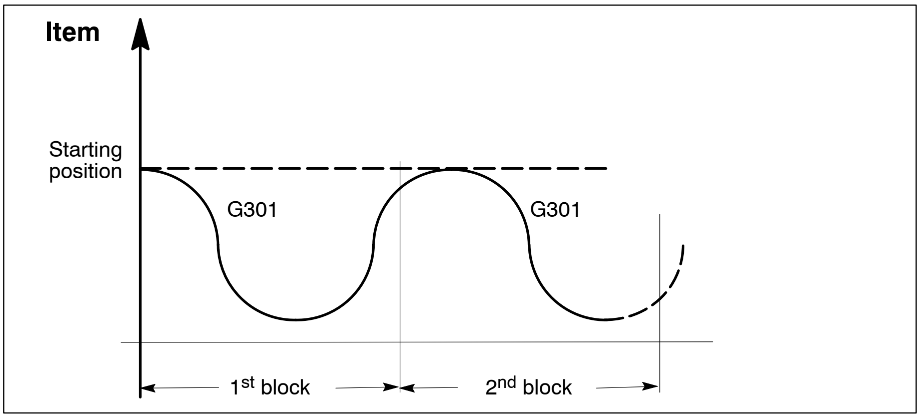
- linear axis
- rotary axis
- C axis

The parameters of the oscillating movement (initialization) are set by G350 and saved modally.

This initialization must be programmed prior to the actual oscillating movement:

- selection of the oscillating axis
- starting and end position as reversing points of the oscillating movement
- frequency or speed of oscillating movement

The oscillating axis is implemented as modal function G301, ”oscillation with linear movement”. The transition of the oscillating movement between 2 consecutive oscillation blocks is steady (also in terms of speed).



Programming

Initialization of the oscillating axis function

```
G350   OscAxis<physical axis index> URP<axis position>
       LRP<axis position> F<speed>|OF<oscillation frequency>
       R<reversing range>
```

where:

| | |
|---|---|
| OscAxis | selection of the oscillating axis (physical axis index) |
| URP | upper reversing point of the oscillating axis (mm) |
| LRP | lower reversing point of the oscillating axis (mm) |
| F | speed of the oscillating axis (mm/min)<br>-**alternatively to OF**- |
| OF | oscillation frequency (Hz in 1/sec) - **alteratively to F** - |
| R | reversing range (not yet available, will be implemented in a later option) |

Programming

Starting the oscillating movement:

```
G301   X<axis position> Y<axis position> F<speed> Time<duration>
```

where:

| | |
|---|---|
| X | synchronous axis, interpolating linearly with Y |
| Y | synchronous axis, interpolating linearly with X |
| F | path feed of axes (X, Y) |
| Time | duration (sec) of oscillating movement for blocks without traversing movement |

**Example:**

| | |
|---|---|
| `G350 OscAxis4 URP200 LRP100 OF5` | Oscillating axis is the 4th physical axis (e.g. U axis) |
| `G301 X100 Y10 F20 Time 200` | |

☞ **If the axis address of the oscillating axis is programmed with a traversing path, an error message will be generated. None of the explicitly programmed axes may be defined as an oscillating axis.**

☞ **The programmed time (Time) only refers to the block in which ”Time” was programmed. The oscillating movement takes at least as long as the programmed time. A synchronous traversing movement programmed in the same block whose execution takes more time than the programmed oscillating time, will let the oscillating movement take longer. If oscillating is still active in the next block, although no time has been programmed, the execution time of this block will be solely determined by the synchronous axis movement.**

The **display** is in workpiece coordinates. If a zero point shift is selected for the oscillating axis, the application of the shift will be postponed for as long as the oscillating axis is active. However, it will be applied to the workpiece position display.

The last position prior to the beginning of the oscillating movement will be displayed as end position. This position also acts as starting position of the oscillating movement.

The distance to go is defined as the difference between the end point and the current machine position setpoint. It oscillates between 0 and the distance between the reversing points.

**Please note for G301 and G350:**

- G301 is a modal function (group G1, G2, ...)
- G350 is not modal (therefore not in display)
- G350 sets the parameters modally, i.e. the old parameters can only be overwritten by programming G350 with suitable parameters again.
- When oscillating has been activated, the oscillating axis first traverses to the reversing point which can be reached within the shortest distance starting from its current position.
- Oscillating will remain active until a new modal movement function (e.g. G1, G2, ...) is programmed.
- For as long as oscillating is active, the oscillating movement will be steady and can be differentiated across block limits.
- When oscillating is turned off, the oscillating axis will return to the reversing point from where it started.
- After Control Reset, the oscillating movement will not be cancelled before the next reversing point (speed = 0) has been reached. The modal function will be cancelled in accordance with the default status.
- Programming of NC functions (loop gain (KV) programming, feed forward control, etc.) which **act** on the physical address of the oscillating axis should be avoided within the machining section because sudden speed losses may be the consequence.
- By programming the address of the oscillating axis in connection with a function internal to the NC (e.g. G60 oscillating axis address value), this function will be activated, however, it will not take effect on the oscillating axis for as long as oscillating is active. The compensation (e.g. G60) will not be selected before the oscillating movement has stopped.
- The program value equals zero during the oscillating movement.
- If a zero offset is activated while G301 is being programmed, first make a query of the current data of the oscillating axis using CPL function “FXC” because this data must be taken into account when programming the “URP” and the “LRP” (G350).

**Example:**

```
1  A=FXC(4)
N2 G350 OSCAxis4 URP[200+A] LRP [100+A] OF5
N3 G301 X100 Y10 F20 Time200
```

### Restrictions:

Function G301, ”Oscillating axis”, is not permitted in combination with functions

- In-position logic G61 or G163
- Switching NC blocks via high-speed signal G575

While G301 is active, the following functions must **not** be programmed (because otherwise interpolation would be aborted abruptly which may result in a servo error):

G4, G14/G15, G32, G75, G114/G115, G374, G590/G591, G900

**Auxiliary functions** may be programmed together with G301 only if the time required for interpolating the NC block is longer than the time required for the execution of the auxiliary function, including acknowledgement. Basically, the time required for executing an NC block is determined by the path and the feedrate programmed for this block as well as by the duration of the oscillating movement, ”Time”. Any G301 blocks for which no duration of the oscillating movement nor a traversing motion has been entered are executed within one interpolation cycle.

To ensure that the oscillating movement is properly terminated, **WAIT** needs to be programmed before any M0/M1. If the execution of the program is continued upon a cycle start command, the oscillating movement will also be resumed.

### Example:

```
N50 G301 ...
.
.
N60 WAIT N70 M0
N80 ...
```

## 3.72 Ramp functions G310 - G316

Effect

Provides for the definition of own velocity profiles. The following features are available for this purpose:

- 3 speed interpolators
  (for linear, sinusoidal and $\sin^2$-shaped velocity rise),
- 3 deceleration interpolators
  (for linear, sinusoidal and $\sin^2$-shaped deceleration),
- 1 constant-speed interpolator

**Acceleration interpolators**

With all these options, the control unit will accelerate, starting from velocity $V_0$ in the beginning of the movement, across the entire programmed path length, to the target velocity $V_1$.

The target velocity $V_1$ is reched together with the programmed end point, and results from the programmed feedrate as a function of the current override value. It is limited by

- the maximum path acceleration and
- the maximum permitted path velocity.

Both quantities are calculated by the control unit specifically for each path segment and each NC block, and a 1-block look-ahead is performed in connection with the maximum permitted velocity. This prevents a violation of the maximum axis velocities in the respective next block.

Behavior in the event of override changes:

- for acceleration interpolator with linear velocity increase:
  - If the override is reduced to final values below the start velocity $V_0$, the NC will determine a deceleration ramp that lasts until the programmed end point.
  - Increasing the override will result in a re-computation of the acceleration ramp.
- for acceleration interpolator with sinusoidal and $\sin^2$-shaped velocity rise:
  - An override reduction to final values below the start velocity $V_0$ will be ignored.
  - Increasing the override will result in a re-computation of the acceleration ramp.

**Deceleration interpolators**

With all these options, the control unit will decelerate, starting from velocity $V_0$ (in the beginning of the movement), across the entire path length programmed in the NC block, **always** to zero speed ($V_1$=0).

Override changes have no effect, with the following exception:

If the override was set to 0% in the previous block and afterwards the command velocity of 0 was reached exactly at the block transition to the deceleration interpolator, the control unit will maintain the deceleration interpolator until the override is increased to a value > 0!
The velocity will be increased by one acceleration step (depending on

the permitted path acceleration).
The control unit will calculate the necessary deceleration ramp on the basis of the velocity value resulting from this function. Afterwards, the actual override value will be of no effect until the end of the block.

### Constant-speed interpolator

The control unit tries to reach the programmed velocity, taking the maximum permitted path velocity and the current override value into account.

Behavior in the event of override changes:

- Velocity changes are carried out at the respective permitted path acceleration and path deceleration.

Programming

| | |
|---|---|
| G310 | Constant-speed interpolator on |
| G311 | activate acceleration interpolator with linear velocity rise |
| G312 | activate deceleration interpolator with linear velocity decrease |
| G313 | activate acceleration interpolator with sinusoidal velocity rise |
| G314 | activate deceleration interpolator with sinusoidal velocity decrease |
| G315 | activate acceleration interpolator with $\sin^2$-shaped velocity rise |
| G316 | activate deceleration interpolator with $\sin^2$-shaped velocity decrease |

★ In addition to the G function, the desired end point coordinate has to be indicated for the type of interpolator in the NC block.

☞ **When using deceleration interpolators in connection with extremely short traversing paths, excessive acceleration is possible, which may lead to a servo error.**
**Therefore, please note the maximum possible machine dynamics already when creating the part program.**

☞ **G310 to G316 each act modally, and form a group together with G8, G9, G108, G408 and G608.**

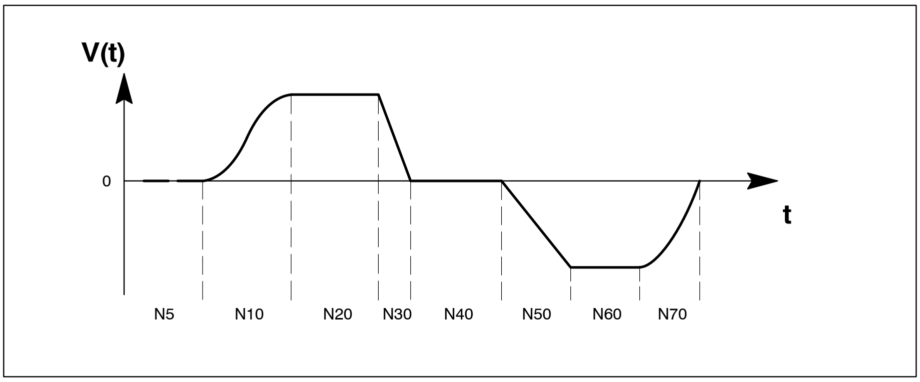
- G310 to G316 can only be invoked in operation mode automatic.
  Other operation modes (manual data input, single block, single step, or program block) will lead to a runtime error.
- No auxiliary functions may be programmed, and no functions such as ”In-position logic” may be active in conjunction with G310, G311, G313 and G315 (constant-speed or acceleration interpolators) because otherwise sudden speed drops may be experienced.
  Prohibited functions: G0, G4, G14, G15, G32, G33, G61, G73, G75, G161, G163, G374, G575, G900.
- No velocity lower than the one active at the beginning of the ramp should be programmed within a movement sequence (comprising acceleration, constant-speed and deceleration).

A lower programmed velocity will be simply ignored by the acceleration interpolator (same behavior as with override change).

- Auxiliary functions or a dwell time may be programmed at the end of each processing sequence.

**Application example:**

Programming an oscillation cycle for the U axis.



| | |
|---|---|
| `:` | |
| `N5 G0 U10` | U axis is traversed to starting position (U=10mm) |
| `N10 G315 U17 F500` | Sin$^2$-shaped acceleration up to position U=17.<br>Feedrate setpoint at the end point:<br>F=500 mm/min |
| `N20 G310 U23` | Constant speed until position U=23. |
| `N30 G312 U29` | Linear deceleration until position U=29.<br>End speed: 0 mm/min |
| `N40 G4 F0.5` | Dwell time in the point of reversal. |
| `N50 G311 U20` | Linear acceleration until position U=29. |
| `N60 G310 U17` | Constant speed until position U=17. |
| `N70 G314 U10` | Sinusoidal deceleration until position U=10. End speed: 0 mm/min |
| `:` | |

## 3.73 Precision programming G328, G329

Effect

The **precision programming** function automatically reduces the feedrate for contour transitions or circular contour segments (circles, helical, helicalN) to ensure compliance with **accuracy specifications** (please refer to the figure below).
For this purpose, a feedrate value is calculated based on a control system model (closed position control loop in steady-state condition, with **feed forward control** taken into account). This feedrate value will ensure that the programmed tolerance does not fall below the required precision at the block transition.

☞ **In contrast to the “in-position” function, the feedrate is not reduced to 0 at a block transition with the ”precision programming” function. With the “in-position” function, the actual following errors of all axes of the respective channel are checked after deceleration to 0 speed.**

The precision tolerance range is set by selecting one of 2 different parameters:

- **Deviation from the contour ε**: Maximum permissible deviation for contour transitions or maximum permissible deviation from radius for circular arcs.
- **Overtravel δ**: Maximum overtravel (distance from corners) not to be exceeded at the actual transition point of a contour transition.

Programming

| | |
|---|---|
| G328 | Activation of precision programming with the value of deviation from contour ε as preset in MP 8003 00001. |
| G328 EPS<contour error> | Activation of precision programming |
| G328 DIST<corner distance> | Activation of precision programming |
| G329 | Deactivation of precision programming |

where:

| | |
|---|---|
| EPS | **Contour transition**: A deviation from the contour at a block transition is the minimum deviation of the actual contour at the transition from the programmed position.<br>**Circular arc**: A deviation from radius is the difference between the programmed radius and the resulting actual radius, which is dependent on path velocity. |
| Contour error | Distance (ε) to be entered in mm or inch (depending on the unit set with G70 or G71). |

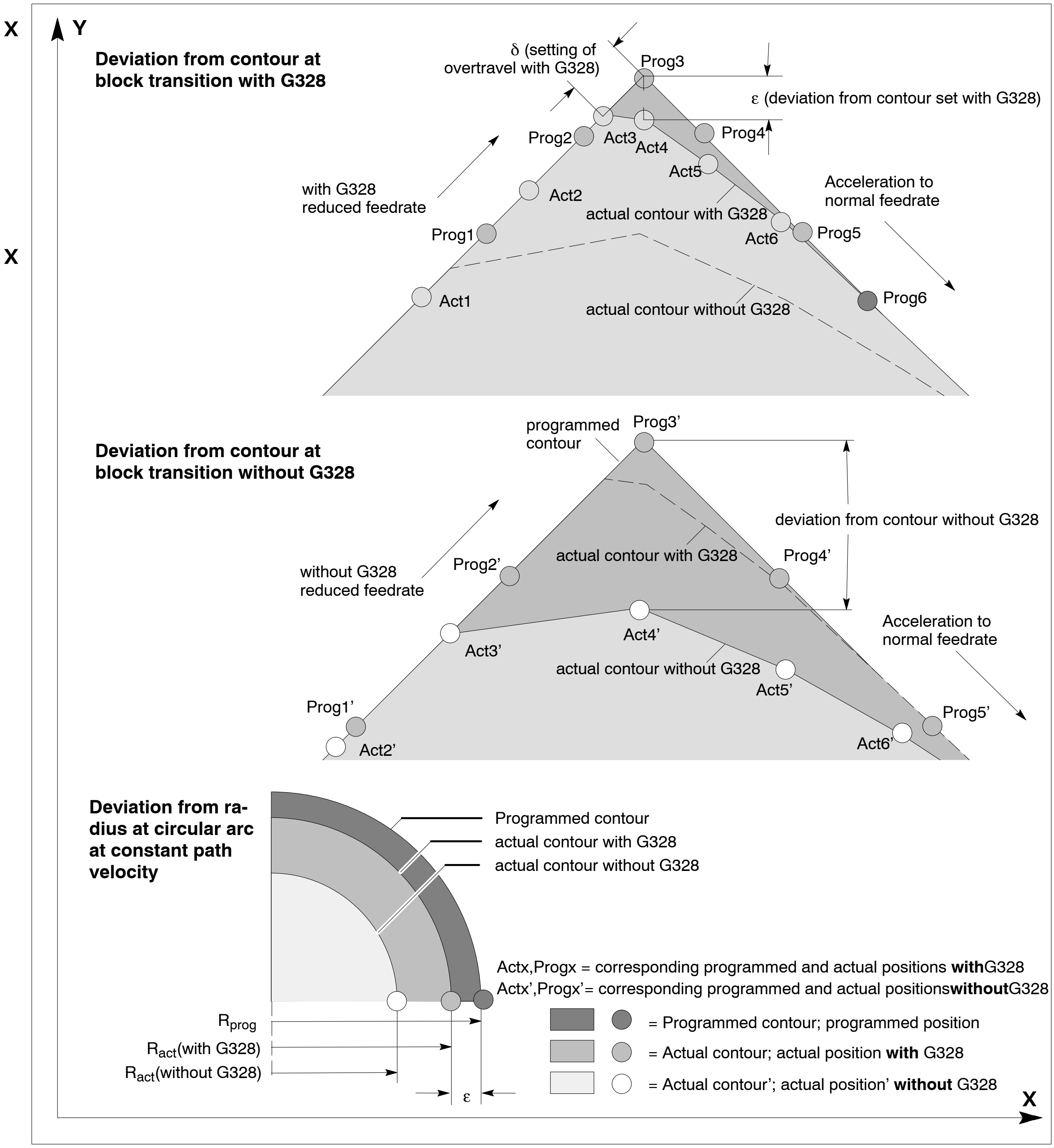
DIST The overtravel is the distance between the position where the actual contour first deviates from the programmed contour before a contour transition and the block position programmed.
When programming circles with DIST, the ε value set in MACODA parameter 8003 00001 is taken into account.

Corner distance Distance (δ) to be entered in mm or inch (depending on the unit set with G70 or G71).

☞ **To avoid the need for entering the deviation from contour ε explicitly each time, a default value may be entered in MACODA parameter 8003 00001.**

**Please note for G328 and G329:**

- Identical dynamics must be set for all axes.
  If drives other than Bosch drives are used, active feed forward control functions will be taken into account only in terms of quality (following error reduced by 50%). The configuration of operation without any following errors in SERCOS varies with the manufacturer.
- A path slope function, G8 or G108, must be active. Otherwise, the speed is decelerated to 0 at every contour transition.
- In-position (G61, G163) must be deactivated to prevent deceleration to 0 at every contour transition.

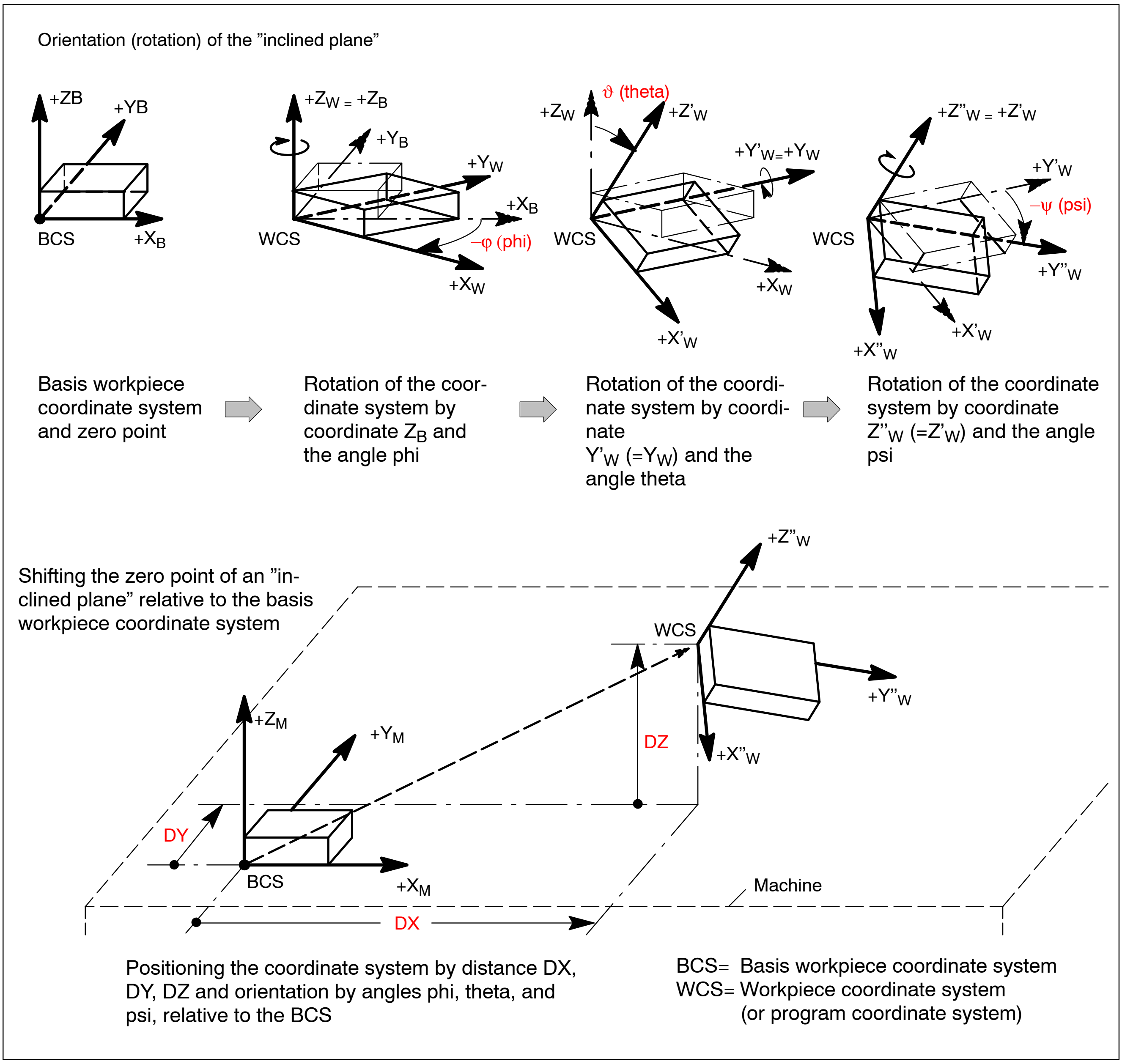

X
Y
X
Deviation from contour at block transition with G328
δ (setting of overtravel with G328)
Prog3
ε (deviation from contour set with G328)
Prog2
Act3
Act4
Prog4
Act5
with G328 reduced feedrate
Act2
Acceleration to normal feedrate
actual contour with G328
Prog1
Act6
Prog5
Act1
actual contour without G328
Prog6
Deviation from contour at block transition without G328
programmed contour
Prog3'
deviation from contour without G328
actual contour with G328
without G328 reduced feedrate
Prog2'
Prog4'
Act4'
Act3'
Acceleration to normal feedrate
actual contour without G328
Act5'
Prog1'
Prog5'
Act2'
Act6'
Deviation from radius at circular arc at constant path velocity
Programmed contour
actual contour with G328
actual contour without G328
Actx,Progx = corresponding programmed and actual positions with G328
Actx',Progx'= corresponding programmed and actual positions without G328
$R_{prog}$
$R_{act}$(with G328)
$R_{act}$(without G328)
ε
= Programmed contour; programmed position
= Actual contour; actual position with G328
= Actual contour'; actual position' without G328
X

## 3.74 Inclined plane G352, G353, G354..G359

Effect

The workpiece (WCS) or program (PCS) coordinate system can be **offset and oriented** in space at the user's discretion using the "Inclined plane" function. The underlying workpiece coordinate system is the reference quantitiy. In addition to a workpiece or program zero point shift, the WCS/PCS can be rotated by several coordinates.

Since there are 3 degrees of freedom for orientation, every orientation can be represented by 3 consecutive basis rotations.



Orientation (rotation) of the "inclined plane"

Basis workpiece coordinate system and zero point

Rotation of the coordinate system by coordinate $Z_B$ and the angle phi

Rotation of the coordinate system by coordinate $Y'_W$ (=$Y_W$) and the angle theta

Rotation of the coordinate system by coordinate $Z''_W$ (=$Z'_W$) and the angle psi

Shifting the zero point of an "inclined plane" relative to the basis workpiece coordinate system

Positioning the coordinate system by distance DX, DY, DZ and orientation by angles phi, theta, and psi, relative to the BCS

BCS= Basis workpiece coordinate system
WCS= Workpiece coordinate system
(or program coordinate system)

Since the inclined plane can be offset and rotated in space relative to the BCS, it is necessary to define the precise location and the orientation of the program or workpiece zero point.

Programming

The **zero point** of the workpiece or program coordinate system of the ”inclinced plane” relative to the BCS basis workpiece coordinate system

- can be entered **directly** using **G352**

  G352 X<X-Offset> Y<Y-Offset> Z<Z-Offset> PHI<1st Eulerian angle> THETA<2nd Eulerian angle> PSI<3rd Eulerian angle>

  where:

  | | |
  |---|---|
  | X | Offset value in X direction relative to BCS zero point |
  | Y | Offset value in Y direction relative to BCS zero point |
  | Z | Offset value in Z direction relative to BCS zero point |
  | PHI | Angle of rotation by Z axis relative to BCS (syntax: PHI, Phi, phi) |
  | THETA | Angle of rotation by the **new** Y coordinate (relative to the position of the coordinate system after rotation by PHI) (syntax: THETA, Theta, theta, The, the) |
  | PSI | Angle of rotation by the **new** Z coordinate (relative to the position of the coordinate system after rotation by THETA) (syntax: PSI, Psi, psi) |

- or called **indirectly**
  - with **G354..G359** (internal call of a table containing all position and orientation parameters). The table has the format of an ASCII file: ID<filename>. **G22** will activate the ID table.
  - turn active ”inclined plane” off with **G353**.

**Example: G354..G359**

| | |
|---|---|
| `N... G22 IDTab1` | activate Inclined plane table Tab1 |
| `N... G354` | ”inclined plane” active; no traversing movement |
| (or) | |
| `N... G354 X... Y... Z..`<br>`N...` | offset and angle of rotation already apply to position programmed in this block |
| `N... G353` | cancel active inclined plane |

**Please note for G352, G353, G354..G359:**

- The ”inclined plane” function is retained after Control Reset provided that no appropriate function is included in the default (power-up) status.
- Functions G352, G353, G354..G359 act modally and cancel each other mutually.
- G352...G359 must not be programmed together with a traversing motion.

- G352...G359 must not be programmed while the ”cutter path compensation” function is active. Consequently, the inclined plane must be selected before the ”cutter path compensation” function is activated.
- Any workpiece coordinates displayed refer to the ”inclined plane”.
- The “inclined plane” function acts additively on function G138, “Workpiece position compensation”.
- When the “inclined plane” function is active, functions G37, G38, G60, G168, G268, G145 – G845, G147 – G847 and Hx are also taken into account.
- The ”inclined plane” function always refers to the **first three ”coordinates” (=directions of the BCS)”** of a channel.
- Within an active ”inclined plane” function, a plane can be selected with G17, G18, G19, G20. Its coordinates refer to the coordinate system of the ”inclined plane”.

  If the appropriate axis classifications have been set in MACODA, the following coordinates make up the respective plane:
  - G17: Xprog Yprog
  - G18: Zprog Xprog
  - G19: Yprog Zprog.

- A contour offset programmed with G60 (with active ”inclined plane” function) refers to the coordinate system of the ”inclined plane”. Programmed axis addresses specify the coordinate directions relative to the ”inclined plane”.
- Axis addresses acting **directly** on the axes are **not affected** by the ”inclined plane” (e.g. G14 X2: In this case, the loop gain (Kv) value acts on the X axis of the machine coordinate system).
- The axis display is in machine and/or workpiece coordinates.

☞ **For details concerning the structure of the table ID <...> of the ”inclined plane”, please refer to the “PNC Description of Functions” manual.**

## 3.75 Measuring fixed stop G375

Effect

In order to use this function, you have to ensure that

- the drive in question supports the SERCOS command S-0-0149, ”Move to fixed stop”, and
- this function has been enabled for the relevant axis by MACODA parameter 1003 00030.

The control unit will traverse all programmed synchronous axes by way of linear interpolation at the specified feedrate to the programmed end point.

During this time period, the control unit will output the ”Move to fixed stop active” signal (NC-O17.0) at the corresponding axis interface, and wait for the feedback of the executed command.
The drive monitors the current torque. If a configurable limit value is exceeded during the motion, the drive will generate a message that will trigger the following activities by the control unit:

- Output of the axis ”Fixed stop reached” IF signal (NC-O17.1)
- the actual position is stored
- Stop motion
- the distance to go and G375 (effective block-by-block) are deleted

The NC will generate an error message if no ”fixed stop” has been reached at the end of the path (specified torque threshold is exceeded).

☞ **The G375 function should only be used in combination with a CPL program for analysis.**

☞ **While G375 is active (G375 is effective block-by-block), the following functions are not permitted: G75, G175, G177, G475.**

Programming

**G375** <end point> <feedrate> **MfsAxis** <axis no>
– or –
**G375** <end point> <feedrate> **MfsAxis** <Var%>
– or –
**G375** <end point> <feedrate> **MfsAxis**(<axis no.>**,**<threshold>)

where

<End point> desired end point coordinates of synchronous axes (e.g. ”X100 Y100 Z100”).
Is traversed to by linear interpolation of all axes involved, taking into account the <feedrate> and MP 1010 00030 (maximum acceleration ”move to fixed stop”).

<Feedrate> desired path feed.
Limited by MP 1005 00030 (maximum feedrate ”move to fixed stop”) and MP 1005 00002 (maximum axis speed and rapid velocity).

<axis no.> index of the axis the torque of which is to be monitored.

<Var%> integer variable containing the <axis no.>.

<threshold> Torque limit value. Input as a percentage of the maximum torque. If not programmed, MACODA parameter 1003 00031, ”torque limit fixed stop” will be effective.

**Example:** Evaluation by CPL program

```
:
N10 G375 X100 F500     ”Measuring fixed stop” for the first axis (in this
    MfsAxis(1,30)      case X) is activated, and end point X=100 is
                       traversed to at a path feed of 500 mm/min.
                       Torque limit value: 30% of the maximum
                       torque.
10  IF SD(9)=0 THEN    Query: fixed stop reached (torque limit exceeded)?
                       If logic TRUE:
20  XPOS=PPOS(1)       Save position of X axis and output message.
N30 (MSG, POSITION     Else: Jump to ENDIF.
     MEASURED)
50  ENDIF
:
```

## 3.76 Move to fixed stop G475

Effect

In order to use this function, you have to ensure that

- the drive in question supports the SERCOS command S-0-0149, ”Move to fixed stop”, and
- this function has been enabled for the relevant axis by MACODA prameter 1003 00030.

The control unit will traverse all programmed synchronous and asynchronous axes at the specified feedrate to the programmed end points and monitor the current torque of a desired drive. During this time, the ”move to fixed stop active” interface signal (NC-O17.0) is output at the corresponding axis interfaces.

During the motion, the control unit tries not to exceed the following torque values:

- Torque limit of fixed stop (MP 1003 00031) or
- torque configured in G477 (refer to page 3–176).

The following activities are carried out for this purpose:

- Output of the axis ”Fixed stop reached” IF signal (NC-O17.1)
- Stop motion
- Setting the programmed position to the actual position +0.1 mm (or 0.1 degrees)

- Monitoring of the axis position for:
  Position of fixed stop + MP 1003 00032 (”Monitoring window fixed stop in mm or degrees”)
- Enable block change.

If the specified torque has not been reached until the programmed end point, the NC will generate an error message.

☞ **G475 remains active beyond the block and will not be terminated before G476!**

☞ **If G475 is active, the following functions are not permitted: G75, G175, G177, G375.**

Programming

**G475** <end point_S> <feedrate_S> <end point_A> <feedrate_A>

where

| | |
|---|---|
| <end point_S> | desired end point coordinates of synchronous axes (e.g. ”X100 Y100 Z100”). Is traversed to by linear interpolation of all axes involved, taking into account the <feedrate_S> and MP 1010 00030 (maximum acceleration ”move to fixed stop”). |
| <feedrate_S> | desired path feed.<br>Programming by ”F” address, limited by MP 1005 00030 (maximum feedrate ”move to fixed stop”) and MP 1005 00002 (maximum axis and rapid velocity). |
| <end point_A> | Desired end point coordinates of asynchronous axes. The coordinates are traversed to taking into account the <feedrate_A> and MP 1010 00030 (maximum acceleration ”move to fixed stop”). |
| <feedrate_A> | Desired feedrate of asynchronous axes.<br>Programming by ”FA” address, limited by MP 1005 00030 (maximum feedrate ”move to fixed stop”) and MP 1005 00002 (maximum axis and rapid velocity). |

The following functions are used in conjunction with G475:

- G476: deactivates G475
- G477: defines the desired torque at the fixed stop

## 3.77 Cancel fixed stop G476

Effect

Terminates the ”Move to fixed stop” function.

- If synchronous and/or asynchronous axes have been programmed in the G476 block, the control unit will traverse all axes at the specified feedrate to the programmed end points.
  The torque value active in the drive parameter S-0-0092 will be effective for moving away (refer to page 3–176).
- If no axes have been programmed in the G476 block, only the synchronous axes will released. In this case, asynchronous axes for which ”move to fixed stop” is still active can only be released by an interface signal.

Programming

**G476** <end point_S> <feedrate_S> <end point_A> <feedrate_A>

where

| | |
|---|---|
| <end point_S> | desired end point coordinates of synchronous axes (e.g. ”X100 Y100 Z100”). Is traversed to by linear interpolation of all axes involved, taking into account the <feedrate_S> and MP 1010 00030 (maximum acceleration ”move to fixed stop”). |
| <feedrate_S> | desired path feed.<br>Programming by ”F” address, limited by MP 1005 00030 (maximum feedrate ”move to fixed stop”) and MP 1005 00002 (maximum axis and rapid velocity). |
| <end point_A> | Desired end point coordinates of asynchronous axes. The coordinates are traversed to taking into account the <feedrate_A> and MP 1010 00030 (maximum acceleration ”move to fixed stop”). |
| <feedrate_A> | Desired feedrate of asynchronous axes.<br>Programming by ”FA” address, limited by MP 1005 00030 (maximum feedrate ”move to fixed stop”) and MP 1005 00002 (maximum axis and rapid velocity). |

## 3.78 Torque reduction fixed stop G477

Effect

G477 overwrites the drive's internal parameter S-0-0092, ”Torque limit value” of the programmed axis. In this way, the ”clamping torque” of the axis in question (refer to G476) can be set in the part program.

☞ **ID number S-0-0092, ”Torque limit value”, must be available in the relevant drive, and must have been configured for ”% weighting”!**

☞ **Using the ”torque reduction” function via the interface (G177) is not permitted while functions G375, G475 and G476 are effective!**

Programming

**G477** <axis><torque>

where

| | |
|---|---|
| <Axis> | Axis whose parameter S-0-0092, ”Torque limit value” is to be influenced. |
| <Torque> | Desired torque limit value as a percentage. |

**Example:**

| | |
|---|---|
| `N50 G477 X20` | Parameter S-0-0092, ”Torque limit value”, is set to 20 in the drive that is assigned to the X axis. |

## 3.79 Axis transfer G510..G513, G515 G516 G517 G518

The axis transfer function influences the assignment of an axis to an axis group (interpolation group within a channel):

- Transferring axes between axis groups, i.e.
  a synchronous axis remains a synchronous axis
- Removing an axis from an axis group, i.e.
  a synchronous axis becomes an asynchronous axis
- Taking over an axis to an axis group, i.e.
  an asynchronous axis becomes a synchronous axis
- Renaming axes within a group
- Changing over the axis classification (functional significance).

☞ **For details, please refer to “PNC Description of Functions” manual.**

### Overview

The following G functions are available:

**G510 (..)** Integrate axis.
Error message if the axis has not been released from its previous axis group.

**G511 (..)** Integrate axis with WAIT until axis has been released

**G512 (..)** Remove an axis from an axis group

**G513** Accept the axis configuration from MACODA

**G515 (..)** Assign new logical axis name
This logical name must have been predefined in MACODA parameters 7010 00010, ”Logical axis designation”, or 7010 00020, ”Optional axis designation”.

**G516 (..)** Remove logical axis names from the calling axis group.

**G517** C axis off

**G518** C axis on

**G16** Plane selection off.
Circular interpolation is rendered impossible. Main or secondary axes can be removed from the axis group. (for a description, refer to page 3–29).

**G21 (..)** Change over the axis classification (for a description, refer to page 3–34)

Parameters

The following syntax is applicable to G510 to G518:

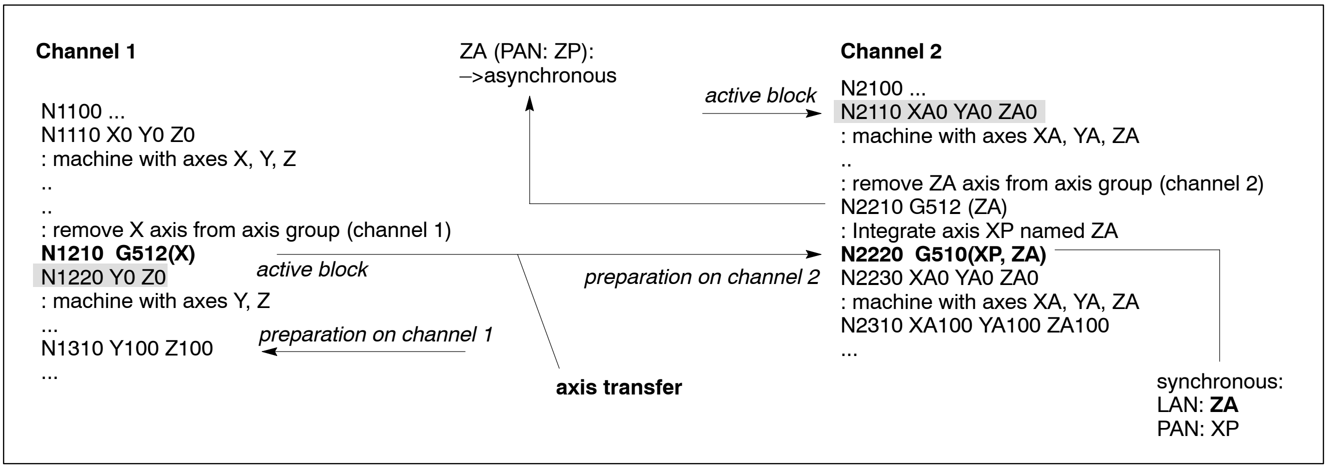
| | |
|---|---|
| PAN | physical axis name |
| PAI | physical axis index |
| LAN | logical axis name |
| | optional: |
| PANi \| PAIi \| LANi | i-th physical axis name, axis index, or logical axis name |
| i, n | number of axes applied (i=1..n; currently available: $n_{max.}$=8) |

☞ **The functions G510 to G513 have to be programmed ahead of an axis in the NC block.**
Example: Correct: N10 G512(Y) X100
Wrong: N10 X100 G512(Y) will trigger an error message.

## Transferring axes between axis groups

- Channel of the“ source axis group” is not active.
  An axis can be transferred to a second axis group at any time (borrowing of axes).

- Channel of the “source axis group” is active.
  The axis first has to be removed from the active channel, and integrated into the other channel in a second step.
  **Example:**
  The X axis of channel 1 is transferred to channel 2 (refer to Fig. below).
  On channel 1, preparation has advanced to block N1310 and block N1220 is active. Thus, the release of the X axis is concluded.
  At this point in time, block N2110 is active on channel 2. Preparation has advanced to block N2220 and is going to take over physical axis XP (previously the X axis on channel 1).
  Axis XP is assigned the name ZA in the process. Since this name is already known on channel 2, no waiting time will occur in the transfer process.



**Channel 1**

```
N1100 ...
N1110 X0 Y0 Z0
: machine with axes X, Y, Z
..
..
: remove X axis from axis group (channel 1)
N1210  G512(X)
N1220 Y0 Z0
: machine with axes Y, Z
...
N1310 Y100 Z100
...
```

**Channel 2**

```
N2100 ...
N2110 XA0 YA0 ZA0
: machine with axes XA, YA, ZA
..
: remove ZA axis from axis group (channel 2)
N2210 G512 (ZA)
: Integrate axis XP named ZA
N2220  G510(XP, ZA)
N2230 XA0 YA0 ZA0
: machine with axes XA, YA, ZA
N2310 XA100 YA100 ZA100
...
```

## Removing an axis from an axis group

G512 is used to remove an axis from an axis group. This turns a synchronous channel axis into an **asynchronous** axis.
Block preparation is not interrupted by this action.

Programming G512 (<PANi | PAIi |LANi>,..,<PAN n | PAIn | LANn>)

where

PAN | PAI | LAN Defines the axis/axes to be removed.

☞ **An error message is output if an invalid axis name is programmed. However, if the axis no longer exists in the channel, there will be no error message.**

**Example**:

| | |
|---|---|
| `G512 (XP,2,Z)` | Physical axis XP, the physical axis assigned the index 2, and logical axis Z are removed from this axis group. |

## Taking over an axis to an axis group

Using G510 or G511, an asynchronous axis is integrated into an axis group, thus turning it into a synchronous channel axis.

- G510 requires a stopped axis, otherwise, an error message will be output, and block preparation will be stopped.
- G511 implicitly waits for the axis to stop.
- A new logical axis name can be optionally input with this function

Programming G510 (<PANi | PAIi>,{<LANi>},..,<PAN n | PAIn>,{<LANn>})

where

PAN | PAI Defines the axis/axes to be integrated in the receiving channel.

LAN Programming is **optional** and defines the “logical name” by which the axis to be integrated is to be addressed on the receiving channel. This logical name must have been predefined in one of channel-specific MACODA parameters 7010 00010, “Logical axis designation”, or 7010 00020, “Optional axis designation”. If you choose not to assign a logical name, enter two commas.

**Example**:

| | |
|---|---|
| `G510 (YP,,ZP,Z)` | Physical axes YP and ZP are integrated in the receiving channel. ZP can be addressed by its logical name Z. YP only by the name YP. If either one of these axes has not been released, a runtime error will occur. |

Programming G511 (<PANi | PAIi>,{<LANi>},..,<PANn | PAIn>,{<LANn>})

where

| | |
|---|---|
| PAN \| PAI | same as G510 |
| LAN | same as G510 |

**Example**:

`G511 (YP,ZP,Z)` same as G510. However, block preparation will wait until YP and ZP have been released.

### Accepting the axis configuration from MACODA

If an axis that is to be transferred has not been released yet, this will cause a runtime error. Therefore, G513 should be entered in a suitable position in the init string.

- Behind the **#Reset** keyword in the init string:
  G513 will only be executed with **Control reset**.
- Behind the **#SysRes** keyword in the init string:
  G513 will only be executed with **System reset**.

Programming G513

### Assigning a logical axis name

With function **G515**, you can assign a new logical axis name on a channel. Again, the same conditions apply as in the case of an axis transfer, i.e. the “new logical axis name” must have been predefined in one of the two channel-specific MACODA parameters, either in 7010 00010, “Logical axis designation”, or in 7010 00020, “Optional axis designation”.

Using **G516**, the assignment of the name can be reversed.

Programming G515 (<PAN1 | PAI1 |LAN1$_1$>,<LAN1$_2$>,..,
<PANn | PAIn |LANn$_1$>,<LANn$_2$>)

where

| | |
|---|---|
| PAN \| PAI \| LAN$_1$ | Defines the axis/axes that are to be assigned a new “Logical axis name” (LAN$_2$). |
| LAN$_2$ | Specifies the ”new” logical axis names. |

Axis designations are programmed by pairs. A parameter list may include several such pairs.

**Example**:

`G515 (YP,X,3,Y,B,Z)` Physical axis YP is assigned the logical axis name X, the 3rd physical axis is assigned the logical name Y, and logical axis B is assigned the logical name Z. Programming B will generate a runtime error.

Programming

G516 (<PANi | PAIi |$LANi_1$>,..,<PANn | PAIn |$LANn_1$>)

where

| | |
|---|---|
| PAN \| PAI \| $LAN_1$ | Defines the axis/axes to be removed from the receiving channel. |

**Example**:

| | |
|---|---|
| `G516 (YP,3,Z)` | The logical names of physical axis YP, the 3rd physical axis and logical axis Z are removed from the receiving channel. |

**Switchover from spindles to asynchronous axes and back:**

Effect

When a spindle is switched to C axis operation, the spindle turns into an asynchronous axis. The display shows an asynchronous axis. Following a switchover, the axis is located on a freely definable position between 0 and 359.9999 degrees.

To switch the asynchronous axis back to spindle operation, it has to stand still before the switchover is performed. Furthermore, the axis may not be active in any axis group.

Programming

**Switch off ”C axis”** (switching back to spindle operation):

G517 (<PANi | PAIi>,..,<PAN n | PAIn>)

where

| | |
|---|---|
| PAN \| PAI | Defines the axis/axes to be switched to spindle operation. |

**Example**:

| | |
|---|---|
| `G517 (CH)` | The physical axis CH (i.e. the spindle with the name CH during axis operation) is switched back to spindle operation. |

**Please note for G517:**

- When switched off, the ”C axis” may not be part of any axis group (group of coupled axes).

Programming

**C axis on**

G518 (<PANi | PAIi>,..,<PAN n | PAIn>)

where

| | |
|---|---|
| PAN \| PAI | Defines the spindle(s) to be switched over to asynchronous axis operation. |

**Example**:

| | |
|---|---|
| `G518 (CH)` | The physical axis CH (i.e. the spindle with the name CH during axis operation) is switched over to an asynchronous axis. |

**Please note for G518:**

- Each spindle entered as spindle/C axis in MACODA parameter 1001 00001 (drive function type) and as SERCOS spindle in MACODA parameter 1040 00001 (selection of spindle type), can be switched to ”C axis operation”.

## 3.80 Drive-controlled interpolation G520..G524

Effect

Using the "**Drive-controlled interpolation**" function, a synchronous axis can be switched to drive-controlled operation and controlled in parallel to synchronous operation.

In the case of drive-controlled interpolation, a synchronous axis can be given **position data** under its physical axis address through any channel although it continues to be permanently assigned to one channel.

At a given moment of time, an axis can only process position data received from one channel. If another channel wants to access the axis, and if a drive-controlled interpolation is already being performed for this axis by another channel, an error message will be output. The position data of the requesting channel will not become active for this axis.

If a channel specifies new position data although the last data specified by the same channel has not been completely interpolated yet, the old data will be replaced by the new one.

Programming

| | |
|---|---|
| N.. G522 X1 Z1 | X and Z are switched to drive-controlled interpolation. (Only possible on the channel to which axes X and Z have been assigned.) |
| N.. G521 | All drive-controlled axes of the channel are returned to NC operation. |
| N.. G520 X100 | Drive-controlled interpolation of the X axis shall be performed for position X100 (can be programmed from any other channel only if the axis is not occupied and has been switched over accordingly).<br>G520 cannot be programmed from the channel to which the axis is assigned. |
| N.. G523 X1000 | The positioning speed of the drive-controlled X axis is to be 1000 mm/min (unit specified in MACODA parameter 7040 00010). This can be programmed from any channel. The positioning speed will be entered under SERCOS ID no. S-0-0259. |
| N.. G524 X3 | The acceleration of the drive-controlled X axis is to be 3m/sec. This can be programmed from any channel. The positioning speed will be entered under SERCOS ID no. S-0-0260. |

**Please note for G521:**

- During drive-controlled interpolation, the NC's internal override function for the axis in question is not active. However, the PLC can directly specify the Feedrate Override (SERCOS ID no. S-0-0108) through the SERCOS service channel in the same way as for referencing.

- The drive responds to the Drive Hold control signal, i.e. drive-controlled interpolation is halted by Feed Hold for the channel to which the axis has been assigned in MACODA parameter 1003 00002. The axis will continue to run after Cycle start. Control Reset for this channel will cancel drive-controlled interpolation and resume NC-controlled position control (corresponding to G522).
- Axis reset will not have any effect on any synchronous axis.
- Only the channel to which the axis is assigned can switch over the drive mode. The ”drive-controlled interpolation” drive mode is output at the axis interface of the axis in question.
- Secondary modes 2 and 3 of the SERCOS drives must be assigned appropriate parameters for ”drive-controlled interpolation”:
  If main operation mode was active previously, the mode is switched to secondary mode 2.
  If secondary mode 1 was active previously, the mode is switched to secondary mode 3.
  Bit 4 (interpolation in the drive) must be set in the respective SERCOS configuration parameters for secondary modes 2 and 3.
- Functions G521 and G522 are active modally and deselect each other mutually.
- Functions G520, G523 and G524 are active block-by-block.
- G520, G523 and G524 have an effect on synchronous and asynchronous axes.

**CAUTION**
**When programming functions G523 or G524 of release 4.x.x, incorrect data may sometimes be written to the drive!**
**If this happens, use G900 for entering positioning speed and acceleration data.**

# 3.81 Collision monitoring ON G543<br>Collision monitoring OFF G544<br>Look-ahead for collision monitoring G500

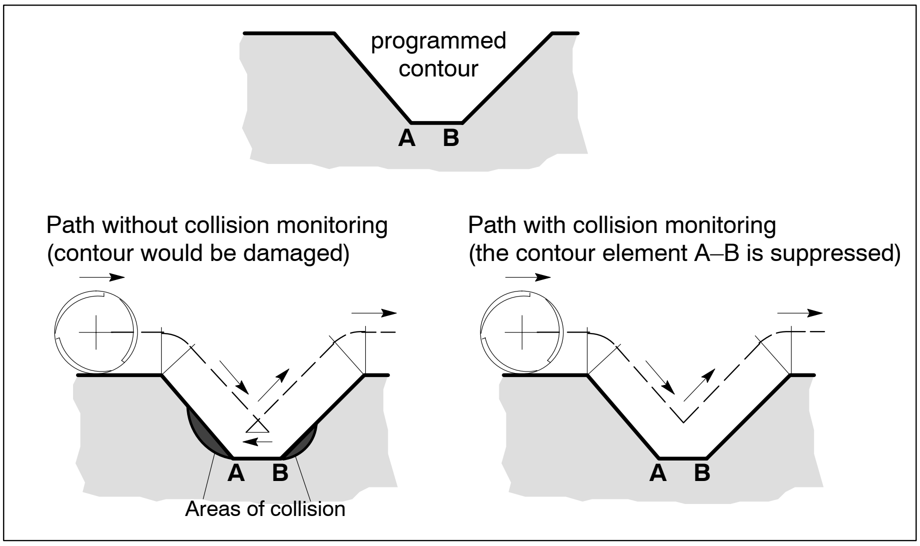
Effect

In connection with G41 and G42, these functions are designed to monitor the contour offset by the cutter path compensation for potential collisions.

A collision occurs if the look-ahead function of the collision monitoring function detects a point of intersection or contact of two path segments on the offset contour path computed by the cutter path compensation function. If an intersection is detected, the offset path will always form a loop. In particular, any point where the contour intersects with itself results in a collision. Only the two coordinates (axes) of the active working plane are taken into account for detection of any contour loops. Any changes that may be caused by the infeed per cut are ignored because the control has no information of how deep the tool plunges into the workpiece.

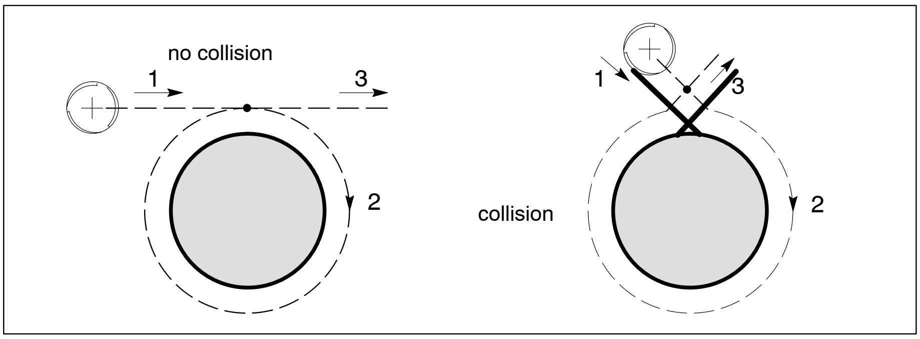
The effective look-ahead range for collision monitoring can be adjusted (default=2 blocks).

If the machining tool radius does not permit machining of individual contour elements, the control unit will try to modify the related path so as to avoid damages to the contour.


programmed contour
A
B
Path without collision monitoring (contour would be damaged)
A
B
Areas of collision
Path with collision monitoring (the contour element A–B is suppressed)
A
B

G Instructions G543 G544 G500

Although full circles always constitute a programmed loop in itself of the programmed contour, they are excluded from collision monitoring provided they are retained as full circles if a compensation value is taken into account.



Generally, it is impossible for the control to recognize whether or not a detected collision may have been programmed on purpose. Therefore, the control functions can be adjusted to individual machining segments.

For this purpose, the PNC offers the following options:

**G543:** Activate collision monitoring. With collision monitoring activated, you can state whether or not you want collisions to be indicated by a runtime error or a warning message.

**G544:** Deactivate collision monitoring.

**G500:** With G500, you can set the look-ahead range for collision monitoring, either generally by changing the default setting, or just locally, for a specific segment.

The PNC performs collision monitoring even if the value of the active D word of the tool radius compensation is set to ”0” (e.g. D1=0).

Programming

G543 CollErr 0: Activate collision monitoring; if an anticipated collision is detected, neither runtime errors nor warning messages are displayed.

G543 CollErr 1: Activate collision monitoring; if a collision is detected, a runtime error is displayed and machining is suspended.

G543 CollErr 2: Activate collision monitoring; if a collision is detected, a warning message is displayed but machining continues.

G543 Activate collision monitoring; behavior in terms of messages displayed remains unchanged. Unless the response in the event of a collision has been programmed previously (or entered in the init string), G543 = G543 CollErr 0!

G544 Deactivate collision monitoring.

☞ **Optionally, the CollErr variable may also be spelt COLLERR.**

Programming

| | |
|---|---|
| G500 VS 3 | The look-ahead range for collision monitoring is set locally to 3 blocks. When G41 or G42 are programmed the next time, the set default value becomes effective again. |

☞ **In release V4.3.8.1 or later, ”G500 VS n” may be programmed in one block with G41 or G42. The look-ahead range can be selected between 1 and 5 blocks.**

| | |
|---|---|
| G500 DVS 1: | The DVS variable changes the default setting for the look-ahead range. The look-ahead range thus programmed becomes effective the next time G41 or G42 is programmed. The preset value can be overridden locally with ”G500 VS n”. ”G500 DVS n” must be programmed in a block before G41 or G42. It is recommended that you preset a value between 1 and 5 blocks. |
| G500 | The preset value of the look-ahead range is reset to the default value of 2 blocks. The effect of G500 is the same as if you program ”G500 DVS 2”. |

**Please note the following for G543, G544 and G500:**

- You can select the power-up condition (monitoring response following control reset) – Monitoring ON/OFF – by setting MACODA parameters 7060 00020 and 7060 00010 for a specific channel.
- The response in terms of **messages displayed** in the event of a collision (G543, CollErr) programmed last remains effective until it is reprogrammed or a new default is set via the init string at control reset. If the response in terms of messages displayed shall always be the same, you can program this simply by making the respective entry in the init string. In the part programs, you may only have to program G543 and G544.
- The **DVS** default value programmed last remains effective until a new presetting or just G500 is programmed or activated via the init string at control reset. If you want to apply the preset look-ahead range (i.e. a range other than 2 blocks) all the time, you just need to program ”G500 DVS n” in the init string. With G500 DVS 1, e.g., you can set the look-ahead range for collision monitoring to 1 block (like with CC220 or Typ1 osa).
- If the look-ahead function finds a traversing block for which the collision monitoring function is deactivated (G544), collision monitoring, which was activated by a previous block, is terminated in this traversing block. When collision monitoring is activated again (G543), a new look-ahead process is started for collision monitoring.
  In order to suspend the collision monitoring function temporarily, it is not enough just to program G544 in two subsequent blocks and then to program G543 in the next block. In addition, a traversing motion must be programmed between G544 and G543. This traversing motion may be programmed together with G544 in one and the same block.

### Collision monitoring with reverse compensation direction

In order to move backwards along the contour while the cutter path compensation function is active, a reversal in compensation direction (G41 becomes G42, or G42 becomes G41) must be programmed. In this case, the collision monitoring function will not signal a collision of the forward motion with the subsequent backward motion.
To this end, the look-ahead function for collision monitoring is canceled in the block in which the compensation direction is reversed. Subsequently, a new look-ahead is started in the block reversing the compensation direction.

**Example:**

| | |
|---|---|
| `N10 G41 G500 DVS10 H1` | Traversing forward with cutter path compensation on the left |
| `N20 X10` | |
| `N30 X20` | |
| `N40 X30` | |
| `N50 G42` | Reversing motion with cutter path compensation on the right. Although the look-ahead function is set to 10 blocks ahead, collision monitoring is canceled and restarted in block N50. |
| `N60 X20` | |
| `N70 X10` | |
| `N80 X0` | |
| `N90 G40` | |
| `M30` | |

# 3.82 Axis coupling G581, G580

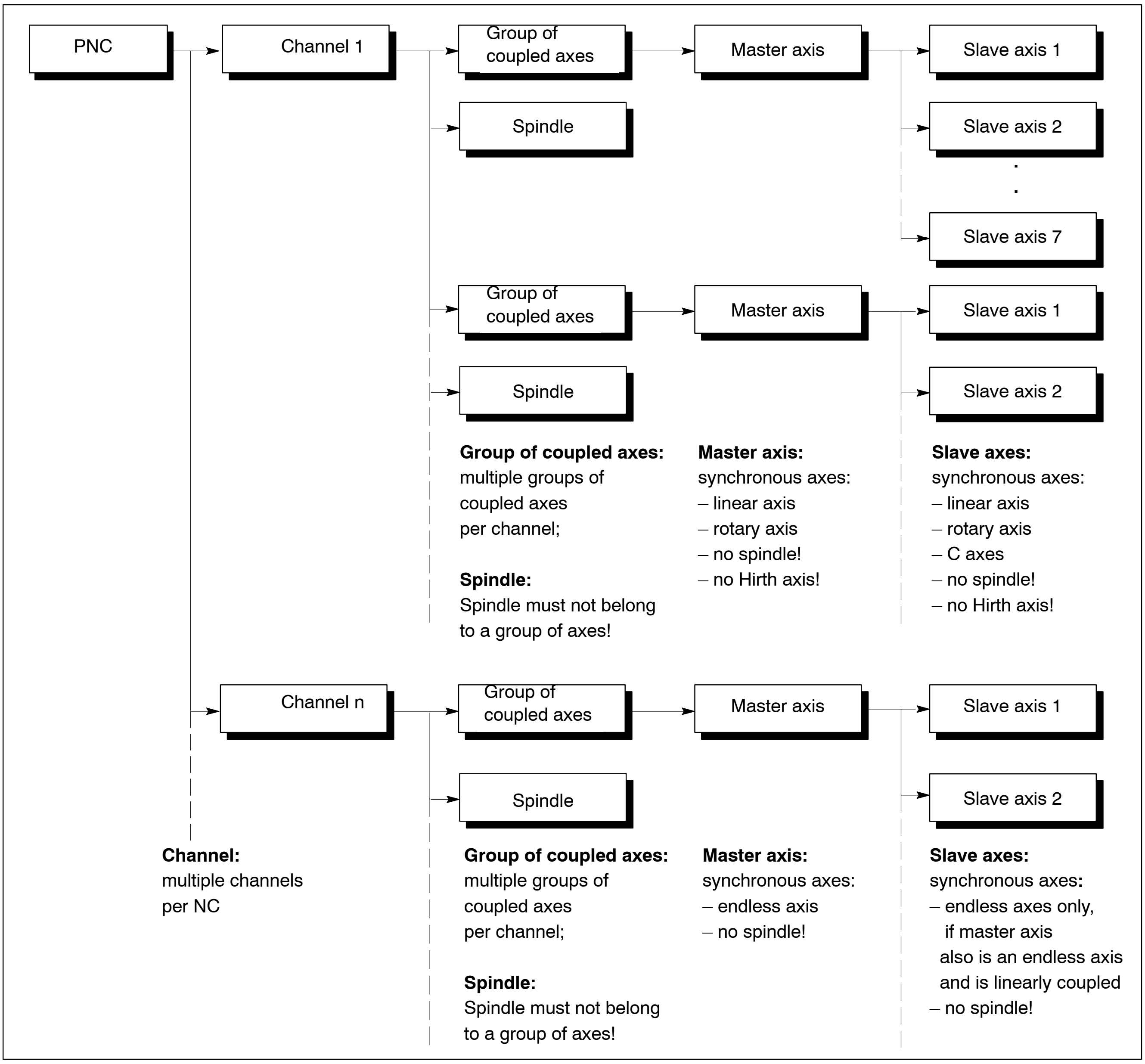
Effect

The axis coupling function establishes a fixed relationship between the positions of a **master axis** and a **slave axis**.

**Group of coupled axes:**

Master axes and slave axes are linked to form a **group of coupled axes**. Every group of coupled axes consists of just **one** master axis and up to **seven** slave axes. All axes belonging to a group of coupled axes must be on one and the **same channel**. A channel may be assigned more than one group of coupled axes.

The following overview shows the structure of parallel axes in terms of groups of coupled axes and channel assignment:



**Group of coupled axes:**
multiple groups of coupled axes per channel;

**Spindle:**
Spindle must not belong to a group of axes!

**Master axis:**
synchronous axes:
– linear axis
– rotary axis
– no spindle!
– no Hirth axis!

**Slave axes:**
synchronous axes:
– linear axis
– rotary axis
– C axes
– no spindle!
– no Hirth axis!

**Channel:**
multiple channels per NC

**Group of coupled axes:**
multiple groups of coupled axes per channel;

**Spindle:**
Spindle must not belong to a group of axes!

**Master axis:**
synchronous axes:
– endless axis
– no spindle!

**Slave axes:**
synchronous axes**:**
– endless axes only, if master axis also is an endless axis and is linearly coupled
– no spindle!

In a group of coupled axes, parallel axes, electronically controlled gears and other freely definable coupling characteristics are possible simultaneously.

**Coupling characteristics:**

Linear coupling characteristics are distinguished from freely defined coupling relationships of the axis positions relative to each other.

**Linear coupling:**The relationship between the position of the master axis, $p_m$, and the position of the slave axis, $p_s$, can be linear:

$$\mathbf{p_s = p_m * \underbrace{k}_{\text{Coupling factor}} + \underbrace{o}_{\text{Offset}}} \quad \text{(formula 1)}$$

k=1 —> parallel axes

$k \neq 1$ —> electronic gearbox

**Freely defined coupling:** Function $\mathbf{f(p_m)}$ is stored as a function table (coupling table) in the file system of the PNC.

$$\mathbf{p_s = f\,(\underbrace{p_m}_{\text{Coupling function (in coupling table format)}} - \underbrace{\overset{o}{p}_m}_{\text{Shift of the master axis}}) * \underbrace{k}_{\text{Coupling factor}} + \underbrace{o}_{\text{Offset}}} \quad \text{(formula 2)}$$

A group of coupled axes consists of a master axis and one or more slave axes: **Each** slave axis has **its own** specific coupling relationship with the master axis as defined by formula (1) or (2).

The second members of equations (1) and (2) represent reference values in the form of parameters that are dependent on the position of the master axis and that are used in every interpolation cycle as a position input for the slave axes.

Example of coupling characteristics:

- **parallel axes** (layout of machine tables in parallel)
- **electronically controlled gear box** (axes moving at certain ratios)

## 3.82.1 Types of axes

The following types of axes may be used both as master and slave axes:

- synchronous axes
- axes that can be switched from synchronous to asynchronous (if they belong to a group of axes, they must be switched to synchronous)
- modulo axes

The following types of axes **must not be used** as master or slave axes:

- asynchronous axes
- Hirth axes

**Restrictions applying to modulo axes:**

- Linear coupling characteristics:

  If the master axis is a modulo axis (linear or endless: refer to MACODA parameter 1003 00004), the slave axis must be a modulo axis, too. The following restriction applies in respect of the modulo value (drive parameter):

$$\mathbf{m_m * k \bmod m_s = 0}$$

modulo value, slave axis

modulo value, master axis

- Coupling via coupling table:

  A modulo master axis can be coupled with a non-modulo slave axis via the coupling table. The restrictions in respect of the coupling table (refer to sect. 3.82.6, parameter #20) apply to the master axis (modulo axis). The slave axis is not subject to any restrictions.

## 3.82.2 Forming a group of axes

Programming

**G581:** Forming a group of axes with **linear** coupling

G581<Master name>0 <Slave name 1>({<$o_{s1}$>,<$k_{s1}$>}) ...
{<Slave name n>({<$o_{sn}$>,<$k_{sn}$>})}

where:

| | |
|---|---|
| master name | logical axis address of the master axis |
| slave name i | logical axis address of the i-th slave axis |
| $o_{si}$ | shift of the i-th slave axis |
| $k_{si}$ | coupling factor of the i-th slave axis |
| i = 1 .. max. 7 (n) | max. number of slave axes per channel and group of coupled axes |

**Example:**

```
G581 Z0 A(4,2) B(2,1)
```

Z= master axis,
A and B = slave axes

Programming

**G581:** Forming a group of axes with **any** type of coupling (via coupling table)

G581<Master name>0 <Slave name i>({<$o_{si}$>,<$k_{si}$>,<$p^0$>,<$f_{si}$>})
...{<Slave name n>({<$o_{sn}$>,<$k_{sn}$>,<$p^0$>,<$f_{sn}$>})}

where:

| | |
|---|---|
| master name | logical axis address of the master axis |
| slave name i | logical axis address of the i-th slave axis |
| $o_{si}$ | shift of the i-th slave axis |
| $k_{si}$ | coupling factor of the i-th slave axis |
| $p^0$ | Shift of the master axis |
| $f_{si}$ | name of the coupling table for the i-th slave axis |
| i = 1 .. max. 7 (n) | max. number of slave axes per channel and group of coupled axes |

**Example:**

```
G581 X0 B(-3,0.5,0,"Ftab_B")
```

Ftab_B= coupling table for coupling slave axis B and master axis X

For any parameters not stated in the syntax, **default values** are set:
$k_S = 1$; $o_S = 0$; $p^0 = 0$

In a group of coupled axes, it is also possible to define mixed (linear and freely defined) couplings simultaneously.

**Example:**

```
G581 X0 A(4,2)
B(–3,0.5,0,”Ftab_B”)
```

- – coupling master axis X and A
- – coupling master axis X and slave axis B via the Ftab_B = Coupling table.

For any parameters not stated in the syntax, **default values** are set: $k_S = 1$; $o_S = 0$; $p^0 = 0$

☞ **There is no default available for the coupling table.**

Therefore, the following terms are available for the coupling syntax (slave axis B):

| | |
|---|---|
| B() | linear coupling with o = 0, k = 1 |
| B(2) | linear coupling with o = 2, k = 1 |
| B(,–1) | linear coupling with o = 0, k = –1 |
| B(,,,Tab) | coupling via table with o = 0, k = 1, $p^0$ = 0, f=Tab |
| B(,,4.5,Tab) | coupling via table with o = 0, k = 1, $p^0$ = 4.5, f=Tab |
| B(–3.2,,4.5,Tab) | coupling via table with o = –3.2, k = 1, $p^0$ = 4.5, f=Tab |

The following **actions** are carried out with G581 <master name> 0 ...:

- Deletion of existing group of coupled axes <master name>
- Slave axes 1 through n stated in the syntax are deleted from other groups of coupled axes
- Forming the <master name> group of axes with the slave axes 1 through n.

**CAUTION**
**Forming a group of coupled axes with NC block ”G581 <master name>0 ...” will cause a traversing motion of all slave axes programmed in this block.**

**Each slave axis will traverse to its specific coupling position (reference value) defined by the position of the master axis and the coupling characteristics.**

☞ **The syntax used previously for axis coupling with G functions**

- **G590 MASTER = formation of a group of coupled axes**
- **G591 deletion of all groups of coupled axes**

**is still supported.**

## 3.82.3 Adding more axes to a group

An existing group of axes may be expanded by one or several slave axes, or existing axis couplings may be changed:

Programming

**G581:** Expanding an existing group of axes with **linear or freely defined** coupling (via the coupling table)

G581<master name>1 <slave name i>({<$o_{si}$>,<$k_{si}$>,<$p^0$>,<$f_{si}$>})
...{<slave name n>({<$o_{sn}$>,<$k_{sn}$>,<$p^0$>,<$f_{sn}$>})}

The coupling syntax for slave axes is the same as shown in sect. 3.82.2.

With G581 <master name>1..., you can expand the group of axes defined by the master name by slave axes 1 through n. You may also use this syntax to change the coupling characteristics of an existing slave axis in this group.

**CAUTION**
**Expanding a group of coupled axes with NC block ”G581 <master name>1 ...” will cause a traversing motion of all slave axes programmed in this block.**

**Each slave axis will traverse to its specific coupling position (reference value) defined by the position of the master axis and the coupling characteristics.**

The following **actions** are carried out with G581 <master name> 1 ...:

- Slave axes 1 through n stated in the syntax are deleted from other groups of coupled axes
- Expanding the <master name> group of axes with the slave axes 1 through n.

## 3.82.4 Removing axes from a group

One or several slave axes can be removed from an existing group of axes.

Programming

**G581:** Reducing an existing group of axes with **linear** or **freely defined** coupling

G581<master name>–1<slave name i>()...<slave name n>()

The group of axes defined by <master name> is reduced by slave axes “slave name i” through “slave name n” (where i = 1...n).
You do not have to enter the coupling characteristics in the syntax for the slave axes.

**Example:**

| | |
|---|---|
| `G581 Z-1 A() B()` | Slave axes A and B are removed from the Z group of coupled axes. |
| `G581 Z-1` | If no slave axes are programmed, the **whole** group of coupled axes is disbanded. |

## 3.82.5 Disbanding all groups of axes

Programming

**G580:** With G580, **all** existing groups of coupled axes are disbanded.

Apart from G580, disbanding of all groups of axes is possible by **repeated** programming of ”**G581 <master name>–1**”. In the latter case, however, block preparation of axis coupling will remain active. G581 will remain the active G function displayed on the operator interface.

**Example:**

| | |
|---|---|
| `G581 Z0 A(4,2) B(2,1)` | Z group of axes |
| `G581 Y0 C(3,1)` | Y group of axes |
| `..` | |
| `G580:`<br>`or:` | Disbanding the **whole** group of coupled axes. |
| `G581 Z-1` | Disband Z group of axes |
| `G581 Y-1` | Disband Y group of axes. |

## 3.82.6 Coupling table

A coupling table contains the coupling characteristics in the form of a coupling function between slave and master axes. The coupling relationship is described in the form of a coupling function:

$$p_s = f\,(p_m - \overset{o}{p}_m) * k + o$$

Master shift

Coupling function (in **coupling table** format)

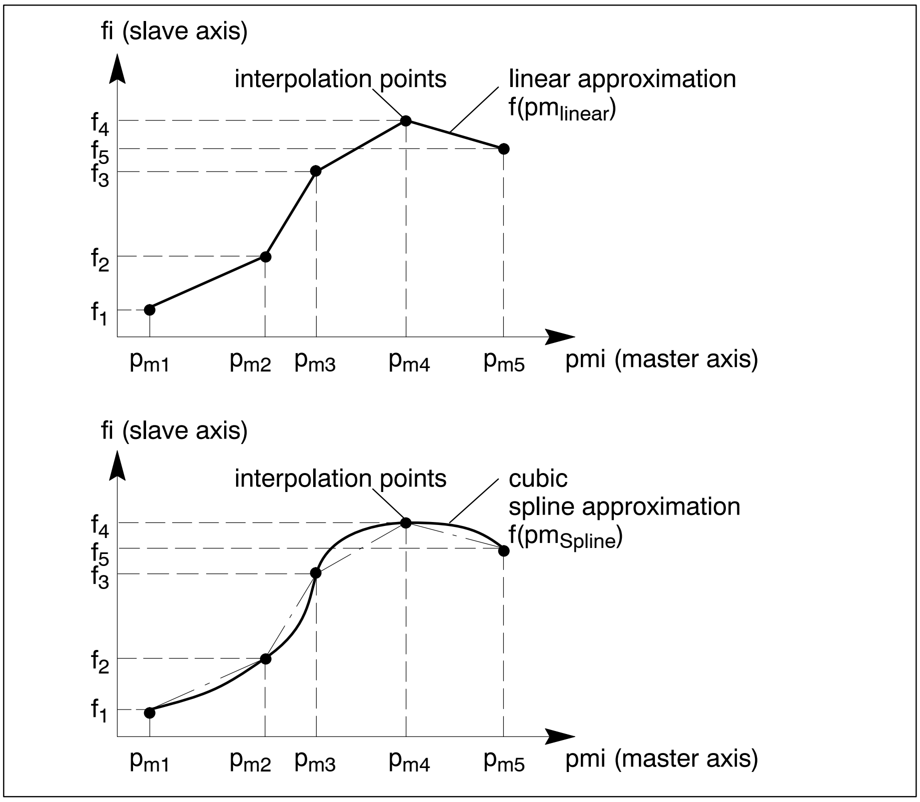
calculated position of the slave axis

The user defines the coupling function f($p_m$) in the **form of a table** with pairs of interpolation points, ($p_{mi}$, $f_i$) (i=1,...,n).

Based on these pairs of interpolation points, the NC interpolator computes the values of the function between the interpolation points and thus the position of the slave axis.

The following approximations may be selected for the computation of the positions between the interpolation points:

- **linear** – as a line between two interpolation points, or
- **cubic spline** – as a spline curve between two interpolation points with the previous and the following interpolations points taken into account (also refer to sect. 3.82.7).



Spline approximation is to be preferred whenever a **curved** shape is desired and no data on the exact shape is available. Cubic spline approximation allows a curve shape with smooth transitions at the interpolation points.

### Structure of the coupling table

| | | |
|---|---|---|
| #1 | <type of interpolation> | 1 ≙ linear,<br>3 ≙ cubic spline |
| #11 | <Unit of the $p_{mi}$ values> | –3 mm, –2 cm, –1 dm, 0 m,<br>1 inch, 2 degrees, 3 rad |
| #12 | <Unit of the $f_i$ values> | –3 mm, –2 cm, –1 dm, 0 m,<br>1 inch, 2 degrees, 3 rad |
| #20 | <periodic> | 0 ≙ non-periodic,<br>1 ≙ periodic |
| #100 | <$p_{m1}$> <$f_1$> | 1st pair of interpolation points |
| #100 | <$p_{m2}$> <$f_2$> | 2nd pair of interpolation points |
| . | . | . |
| . | . | . |
| #100 | <$p_{mn}$> <$f_n$> | nth pair of interpolation points |

Any tables addressed by the NC program will be searched on the subprogram path (compiled table: e.g. /usr/lnk/cam.fct.s).

**Coupling table parameters:**

#1 defines the type of interpolation to be used between the interpolation points. The types available are linear interpolation (value = 1) or cubic spline interpolation (value = 3). The default value is 1, i.e. linear interpolation.

#11 defines the unit of the $p_m$ values. If you specify units of length (values –3 through +1) here, the table may be used for linear master axes only. Accordingly, if you specify angular units (values 2 and 3), the table may be used for rotary master axes only. The default value is –3 (mm).

#12 defines the unit of the f values (the same units as with #11). In tables to be used for linear or rotary slave axes, no other values than –3 through +1 or 2 and 3 may be specified. The default value is –3 (mm).

#20 defines whether the coupling function is to be periodic or non-periodic. If the coupling function is to be periodic (value = 1), the last $p_m$ value defines the period. If the position of the master axis exceeds the period, function value $f(p_m)$ is determined by means of a modulo calculation for $p_m$ to fall within the periodic interval. For non-periodic coupling functions, modulo calculation is deactivated. The default value is 0, i.e. non-periodic.

Please note the following rules:

- #20 = 0, **Non-periodic**:
  The limit switch range of the master axis is restricted to the periodic interval [$p_m$1, $p_{mn}$]. No modulo axes (linear modulo axes or endless axes) may be used as master axes.
- #20 = 1, **Periodic**:
  The $p_m$ values must begin with 0, i.e. $p_{m1}$ = 0. The last $p_m$ value defines the period. The f values of the first and the last pairs of interpolation points must be identical, i.e. $f(p_{m1})$ = $f(p_{mn})$. If the master axis is a modulo axis, the AxModVal mod cycle must be 0, with AxModVal being the axis-specific modulo value (drive parameter).

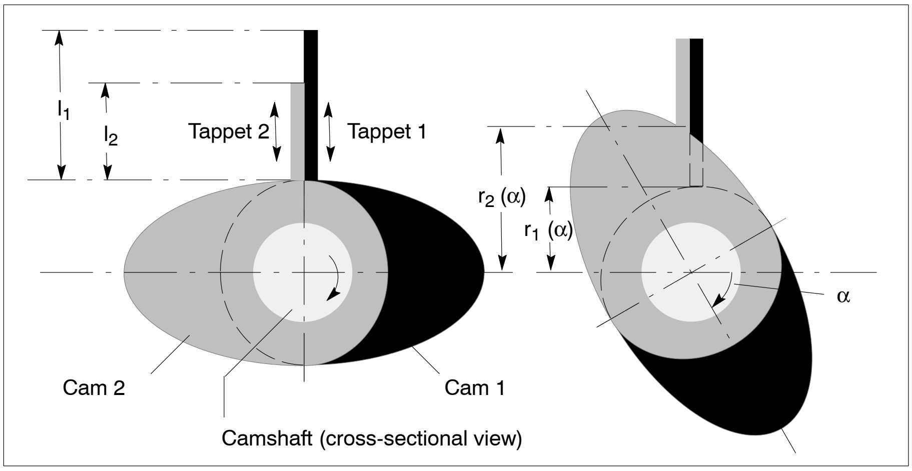
#100 defines a pair of interpolation points. You may enter any number of interpolation points in the table. The $p_m$ values must be entered in ascending order.

; The comment indicator to be used in the table is the semicolon.

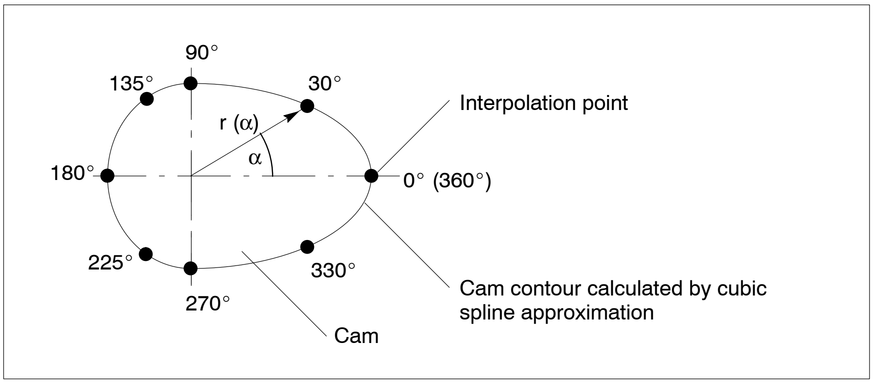
**Example:**

A camshaft with two cams of identical shape and with 180° rotational offset from each other is to move two tappets.

Mechanical design of the camshaft:



Interpolation points of the cam coupling table:



The shape of the two cams is described by eight pairs of interpolation points (refer to coupling table “cam.fct”).

Coupling table:

Various angular positions $\alpha_i$ of the camshaft are assigned cam radius values $r_i$. This results in a coupling table “cam.fct” with the following contents:

```
#1 3                  ;  cubic spline approximation
#11 2                 ;  Unit of the pm values is degrees
#20 1                 ;  periodic coupling function
#100 0.0       30.0   ;  1st pair of interpolation points (α1, r)
#100 30.0      28.0   ;  2nd pair of interpolation points (α2, r)
#100 90.0      24.0
#100 135.0     22.0
#100 180.0     20.0
#100 225.0     22.0
#100 270.0     24.0
#100 330.0     28.0
#100 360.0     30.0   ;  last pair of interpolation points
```

☞ **Syntax: A blank or a TAB is to be inserted between the values.**

Establishing the coupling relationship:

The two tappets with lengths $l_1$ and $l_2$ are represented by two linear axes Z1 and Z2. The camshaft proper is assumed to be an endless axis, called A. The zero point of the linear axes is assumed to be located in the center of the shaft.

**Note:** There is no need to have the camshaft physically represented by an axis. Since the control unit does not support virtual axes, axis A must be entered in MACODA. However, it may be suppressed in the SERCOS interface ring.

Thus, the coupling relationships within the group of axes are as follows:
The following applies to position $p_s$ of axis Z1:

$$\mathbf{p_s = f\ (p_{m+}\ 90) + l_1}$$

The master shift in this case is –90° because, with camshaft A in an angular position $p_m = 0$, slave axis Z1 is to be in this position ($p_s = f(90) + l_1$) (refer to figures).

The following applies to position $p_s$ of axis Z2:

$$\mathbf{p_s = f\ (p_{m-}\ 90) + l_2}$$

The master shift in this case is +90° because the two cams show a rotational offset of 180°from each other.

The axis coupling is generated in the NC program with the following syntax:
**G581 A0 Z1(<l1>,,–90.0,"cam.fct") Z2(<l2>,,90.0,"cam.fct")**

When this coupling syntax is interpreted during block preparation, the corresponding spline table /**<link directory>**/**cam.fct.s** is created (refer to next section).

## 3.82.7 Creating a spline table file

Effect

The spline table is a **separate** table. It is created on the basis of the pairs of interpolation points contained in the coupling table.
The NC block interpolator accesses an image of the spline table and -on the basis of this table - calculates the function values between the various interpolation points and thus the position of the slave axis.

**Creating a spline table:-**

Spline tables are created during block preparation when the coupling syntax is being interpreted. Spline tables are stored as a file in the link table directory.

In the following cases the spline table is **created**:

- Spline table does not exist in the link directory.
- One of the entries #–1, #–2 or #–3 in an already existing spline table does not correspond with the respective attribute of the active coupling table:

  #–1: Size of the coupling table incorrect.

  #–2: Time stamp of the coupling table in the existing spline table is not identical with time stamp of the currently active coupling table.

  #–3: Time stamp of the coupling table in the existing spline table is not identical with time stamp of the currently active coupling table.
- When programming G582 STAB(<name>,**1)**, the creation of the spline table is explicitly mandatory.
- When programming G582 STAB(<name>,**0)**, the creation is mandatory if no spline table exists or if it is older than the coupling table.

☞ **The default link table directory is /usr/lnk. However, it may also be freely defined using MACODA parameter 3080 00004.**

**Name of the spline table:**

The extension ”.s” is added to the name of the currently active coupling table.
**Example:** the name of the spline table is derived from the name of the coupling table **curve.fct**: **curve.fct.s**

**Structure of the spline table:**

| | | |
|---|---|---|
| #–3 | <Size of the coupling table> | |
| #–2 | <Path of the coupling table> | |
| #–1 | <Time stamp of the coupling table> | |
| #1 | <Type of interpolation of the interpolation points> | same as coupling table |
| #10 | <Number of splines> | Number of splines |
| #20 | <Cycle> | Unit increments |
| #30 | <Master axis type> | Permitted master axis types |
| #31 | <Slave axis type> | Permitted slave axis types |
| #200 | <p$_{m1}$> | 1st master axis position |
| #201 | <c10> <c11> <c12> <c13> | Coefficients for 1st spline |
| #200 | <p$_{m2}$> | 2nd master axis position |
| #201 | <c20> <c21> <c22> <c23> | Coefficients for 2nd spline |
| . | . . . . . | . |
| . | . . . . . | . |
| #200 | <p$_{mn}$> | nth master axis position |
| #201 | <cn0> <cn1> <n2> <n3> | Coefficients for 2nd spline |

**Parameters of the spline table:**

#–3 Entering the size of the coupling table in bytes

#–2 Entering the complete name of the coupling table including path.
Example:
#–2 /mnt/AxCo/cam.fct

#–1 Entering the time stamp (time of last change) of the coupling table.

Programming

Creating a spline table file **while** the program is **running**:

**1st step**:

Creating coupling table(s) “fsi” ... ”fsn”:

| | | | | |
|---|---|---|---|---|
| **#1** | 3 | | ; | **cubic spline approximation** |
| **#11** | 2 | | ; | Unit of the $p_m$ values is degrees |
| **#20** | 1 | | ; | periodic coupling function |
| **#100** | 0.0 | 30.0 | ; | 1st pair of interpolation points ($\alpha_1$, $r_1$) |
| **#100** | 30.0 | 28.0 | ; | 2nd pair of interpolation points ($\alpha_2$, $r_2$) |
| **#100** | 90.0 | 24.0 | | |

**2nd step**:

Creating a group of axes with **freely defined** coupling (via coupling table):

G581<master name>0 <slave name i>({<$\mathbf{o}_{si}$>,<$k_{\mathbf{si}}$>,<$p^0$>,<$f_{si}$>})
...{<slave name n>({<$o_{sn}$>,<$k_{sn}$>,<$p^0$>,<$\mathbf{f}_{\mathbf{sn}}$>})}

**3rd step**:

Program start

**4th step** (automatic):

With the program running and while G581 is interpreted, the NC automatically creates one or several **spline table files** on the basis of the coupling table(s) “fsi” ... ”fsn”.

Programming

**G582:** Creating a spline table with **G582**

G582 STAB(<”Coupling table name”>,{<1|**0**>})

where:

coupling table name — The coupling table is searched on the subprogram search path and the corresponding spline table is created in the link table directory.

1|**0** — **optional, default: 0**
”**0**”: The spline table is not created, unless it does not exist or is older than the coupling table.
”**1**”: A new spline table is created.

With G582, you can create a spline table even without an existing group of axes (e.g. in Manual Data Input mode).

**Example:**

| | |
|---|---|
| `G582 STAB(”curve.fct”)`<br>equivalent to:<br>`G582 STAB(”curve.fct”,0)` | creates the spline table /<link directory>/curve.fct.s, if required |
| `G582 STAB(”curve.fct”,1)` | creates the /<link directory>/curve.fct.s spline table irrespective of the time stamp and whether or not the spline table already exists |

**Please note for G580 and G581:**

- All axes within a group of coupled axes must be on the same channel and may be assigned to this group of coupled axes only.
- A maximum of 7 slave axes is permissible for each group of coupled axes.
- You may define and program additional groups of coupled axes. Slave axes are automatically integrated in a new group of axes or deleted from an existing group of coupled axes.
- No slave axis may be the master axis of any third axis at the same time (multi-stage coupling is not permitted!).
- **Referencing:** Deactivate coupling prior to referencing to facilitate referencing of each axis individually.
- Existing axis couplings are retained after the end of a program.
- **Program value display:** The slave axis display depends on the values of the master axis, taking the coupling relationship into account. The display of the slave axes can be suppressed by MACODA settings.
- **Limit switches:** When a group of axes is coupled, the permitted traversing range of the master axis may be reduced if a slave axis has a smaller traversing range than the master axis, or if the shift $o_S$ or the coupling factor $k_S$ (electronic gearbox) are set accordingly. In this case, the traversing range of the master axis will be given **new limit switch values**.
- **Limit switch suppression:** Limit switches can be suppressed both for master axes and for slave axes. In this case, the limit switches have no effect in the coupled condition.
- **Axis dynamics:** When coupling is active, the dynamics of the weakest axis and/or the velocity relationships resulting from the coupling relationships between master and slave axes (e.g. $_m$=25, $V_S$=100, with a coupling factor of 1:4) will determine the dynamics of the whole group. The maximum admissible speed of the group equals the lowest maximum admissible speed of all axes within the group of coupled axes. The same applies to the maximum admissible acceleration.

- **Axis inhibit/Test mode:** All axes within a group of coupled axes must be in the same mode. Axis inhibit of individual axes within a group of coupled axes is not permitted. Activating a couple in test mode is only permitted if test mode is switched on with active coupling (since the slave axis might not be in coupling position when test mode is switched off, a setpoint jump may occur which may cause a drive error!). Therefore, the PLC must ensure that the interface signals act on **all** axes within a group of coupled axes.
  Axes coupled in test mode must be uncoupled before test mode is switched off.
- Programming a traversing motion of slave axes is **not** permitted.

## 3.83 Explicit suppression of axes for feedrate computing G594/G595

Effect

By programming G594, axes can be explicitly removed from feedrate computing. These axes have to be defined in MACODA parameter 1003 00020 as axes not contributing to the feedrate.

**Axes contributing to feedrate computing**

- In principle, all programmed axes are taken into account for feedrate computing. If a maximum of 3 axes standing perpendicular to each other are programmed (Cartesian system), the programmed feedrate corresponds to the **physical path speed**.
  If additional axes have been programmed, the control unit assumes in the feedrate division, that all axes are standing perpendicularly on each other.
- For rotary axes, it may be stated in MACODA parameter 7040 00110 how these are weighted against linear axes with regard to their feedrate. The resulting physical path speed then no longer corresponds to the programmed feedrate.

**Axes not contributing to feedrate computing**

- Axis movements created internally by specific NC functions (e.g. C axes during tapping G32, tangential tool guidance G131) are implicitly removed from feedrate computing. The programmed feedrate thus refers to the programmed axes only.
- Axes defined in MACODA parameter 1003 00020 which are explicitly removed from feedrate computing using G594.
- The movement of axes not contributing to feedrate computing is carried along **synchronously** to the movement of the axes contributing to feedrate computing, i.e. all axes start and finish traversing at the same time. Likewise, all acceleration and deceleration processes are locked synchronously.

### Effect of the programmed feedrate at G594

**F value**

If axes not contributing to feedrate computing are programmed in an NC block without other feedrate contributing axes, the last programmed F value for the axes not contributing to feedrate computing is used (if no Omega value has been programmed, refer to below).

**Example**: In machines with parallel, but uncoupled (independent) axis structures, the parallel axes (except for one) may be suppressed for feedrate computing. The feedrate then remains constant, irrespective of whether the ”master axis” alone has been programmed or the parallel axes have been programmed as well.

**Example**: In the function ”tapping without compensation chuck”, it is made sure that for calculating the path of the C axes, only the master axis is taken into consideration for feedrate computing and thus the programmed thread pitch is taken into account correctly.

**Omega** As an option to a programmed F value, the parameter ”Omega” can be used with active G94 to program an independent second feedrate value. This value is used whenever exclusively those axes are moved which have been removed from feedrate computing. Following deselection of the Omega parameter (Omega 0), the last programmed F value is again applicable to movements of exclusively those axes that do not contribute to feedrate computing.

**Example**: The speed of the tool axis in case of ”tangential tool guidance” can be parametrized in an inserted intermediate block using ”Omega”.

**Weighting** If no feedrate contributing axes have been programmed, the feedrate is always interpreted in mm/min or in °/min, irrespective of the currently active measures (G70/G71). This applies both when Omega has been programmed and to the last programmed F value.

**Example**: All rotary/endless axes of a Cartesian machine tool are suppressed for feedrate computing. The programmed F value then corresponds to the physical path feedrate. In this case, the movement of the rotary axes is carried along synchronously. If only axes not contributing to the feedrate are programmed in a block, their feedrate (either the last programmed F value or Omega) is interpreted in °/min, irrespective of G70/G71.

☞ **If the specified feedrate of a feedrate contributing or a non-feedrate contributing axis exceeds an axis limit, there will be a block-by-block reduction of the programmed feedrate or the configured acceleration.**
**The result is a real reduction of the physical path speed or the path acceleration.**

## G594 in case of working range coordinate programming

In case of active working range coordinate programming (Coord(i)), the programmed feedrate refers exclusively to the programmed **linear** position coordinates. Orientation and further pseudo-coordinates are carried along synchronously.

If orientation coordinates have been programmed only, the programmed feedrate refers to the orientation coordinates; however, if pseudo-coordinates have been programmed, these will be carried along synchronously.
With the programming of Omega (G94 active), it is possible to program a second feedrate value alternatively which refers to the orientation coordinates.

If none of the working range coordinates has been programmed, but exclusively pseudo-coordinates, the behavior with regard to the feedrate will be identical with the case of inactive working range coordinates. In this case the pseudo-coordinates correspond to the axis coordinates.

☞ **Working range coordinates cannot be suppressed for feedrate computing (MACODA parameter 1030 00020 is an axis parameter).**

The following table shows the effect of feedrate computing when programming feedrate contributing or non-feedrate-contributing axes or working range coordinates.

| **Working range coordinate programming** | | | | | |
|---|---|---|---|---|---|
| **Position coordinates programmed** | **Orientation coordinates programmed** | **Feedrate contributing axes programmed** | **Non-feedrate-computing axes programmed** | **Programmed F value refers to** | **Omega effective (only with G94)** |
| ● | are carried along if applicable | are carried along if applicable | are carried along if applicable | position coordinates | – |
| | ● | are carried along if applicable | are carried along if applicable | orientation coordinates | ● |
| | | ● | are carried along if applicable | feedrate contributing axes | – |
| | | | ● | non-feedrate-contributing axes | ● |

Programming

**G594** All non-feedrate-contributing axes defined in MACODA are suppressed for feedrate computing.

**G595** If they have been programmed, all axes removed from feedrate computing using G594 will be taken into account in feedrate computing again.
(For working range coordinate programming, refer to page 3–206)

**Example:**
Axes B and C are set as non-feedrate-contributing axes in MP 1003 00020:

| **Axis designation** | **Axis movement type MP 1003 00004** | **Axis not contributing to feedrate computing MP 1003 00020** |
|---|---|---|
| X | linear | no |
| Y | linear | no |
| Z | linear | no |
| B | rotary | yes |
| C | rotary | yes |

Programming example with these settings:

| | |
|---|---|
| `N1 G70 G1 G94 G594 F100` | The feedrate is 100 inch/min (suppressing axes B and C from feedrate computing). |
| `N2 X100` | The X axis traverses at 100 inch/min to position 100 inch. |
| `N3 Y200 B200` | The Y axis traverses at 100 inch/min to position 200 inch, the B axis is synchronously moved along to position 200°<br>(B is not feedrate contributing!). |
| `N4 C200` | The C axis traverses at 100 °/min to position 200° (C moves with the last programmed F value because no feedrate contributing axis is programmed). |
| `N5 Omega 1000` | Omega is 1000°/min (1000 inch/min) |
| `N6 X0 C0` | The X axis traverses at 100 inch/min to position 0 inch. The C axis is carried along to position 0° (C is not feedrate contributing!) |
| `N7 B300` | The B axis traverses at 1000 °/min to position 300° (Omega acts on non-feedrate-contributing axes). |
| `N8 G595` | Deselection of G594 |
| `N9 Y0 B0` | The Y axis traverses to position 0 inch, the B axis to 0°<br>The feedrate of 100 inch/min acts on both axes, the weighting of the rotary axis is defined in MACODA parameter 7040 00110. |
| `N10 B180` | The B axis moves at 100 °/min to position 180° (B axis contributes to feedrate computation). |

**Please note for G594 and G595:**

- Functions G594 and G595 act modally and deselect each other mutually.
- If neither G594 nor G595 is selected, the behavior G595 corresponds to the current MACODA parameter setting.
- G594 and G595 may be programmed together with other preparatory functions, traversing information and auxiliary functions in one block.

## Speed weighting for rotary axes

MACODA parameter 7040 00110 may be used to determine globally for axes removed from feedrate computing how the speed ratio between translational and rotary axes is weighted. This means, a separate entry is made for G70 (inch programming) and G71 (metric programming) whether **1 degree** in feedrate computing is weighted as:

- 1 mm or
- 1 inch.

This allows for the control unit to be adapted to typically European or American behavior.

### European setting

The following must be entered in MACODA parameter 7040 00110:

| Definition | Parameter | Comment |
|---|---|---|
| 000000 (G70)<br>Inch programming: | **1** | 1 degree = 1 mm = 0.03937 inch<br>(or 25.4 degrees = 1 inch) |
| 000001 (G71)<br>metric programming: | **1** | 1 degree = 1 mm |

For metric and for inch programming, **internally** 1 degree always corresponds to 1 mm.

- If the rotary axis alone has been programmed with G71, the feedrate is interpreted as being stated in degrees/min (F100 = 100 mm/min or 100 degrees/min).
- If the rotary axis alone has been programmed with G70, the rotary axis will traverse too fast by the factor 25.4 (feedrate value is interpreted as being stated in degrees/min: e.g. F100=100 inches/ min=2540 mm/min or 2540 degrees/min)
- If the rotary axis and the linear axis traverse jointly, both are **assigned the same weighting** with G71.

  **Example:** G71 is preset.

  The linear axis and the rotary axis have the same distance to traverse of 100 mm or 100 degrees. Therefore, according to the internal computation, both axes traverse at an axis feedrate of 0.7071 x the programmed path feedrate, each, if both distances to be traversed have the same length.

- If with G70 (inch setting) the linear axes traverse by 100 inches and the rotary axis by 100 degrees, the linear axis traverses at almost 100% of the programmed path feedrate because the traversed distance of the linear axis is longer by the inch-to-metric translation factor of 25.4 than the distance to be traversed by the rotary axis (refer to Example: G594)

### U.S. setting

The following must be entered in MACODA parameter 7040 00110:

| Definition | Parameter | Comment |
|---|---|---|
| 000000 (G70)<br>Inch programming: | **2** | 1 degree = 1 inch |
| 000001 (G71)<br>metric programming: | **2** | 1 degree = 1 inch = 25.4 mm<br>(0.03937 degree=1mm) |

For metric and for inch programming, **internally**
1 degree always corresponds to 1 inch

- If the rotary axis alone has been programmed with G70, the feedrate is interpreted as being stated in degrees/min (F100 = 100 inches/min or 100 degrees/min).
- If the rotary axis alone has been programmed with G71, the rotary axis will traverse too slowly by the factor 25.4 (feedrate value is interpreted as being stated in degrees/min: e.g. F100=100 mm/ min=3.937 inches/min or 3.937 degrees/min)
- If the rotary axis and the linear axis traverse jointly, both are **assigned the same weighting** with G70.

  **Example:** G70 is preset.

  The linear axis and the rotary axis have the same distance to traverse of 100 inches or 100 degrees. Therefore, according to the internal computation, both axes traverse at an axis feedrate of 0.7071 x the programmed path feedrate, each, if both distances to be traversed have the same length.

- If with G71 (metric setting) the linear axes traverse by 100 mm and the rotary axis by 100 degrees, the rotary axis traverses at almost 100% of the programmed path feedrate because the traversed distance of the rotary axis is longer by the inch-to-metric translation factor of 25.4 than the distance to be traversed by the linear axes (refer to Example: G594)

### General setting

For both **US and European** programming patterns, the following must be entered in MACODA parameter 7040 00110:

| Definition | Parameter | Comment |
|---|---|---|
| 000000 (G70)<br>Inch programming: | **2** | 1 degree = 1 inch |
| 000001 (G71)<br>metric programming: | **1** | 1 degree = 1 mm |

With metric programming, **internally** 1 degree exactly corresponds to 1 mm,
with inch programming, **internally** 1 degree exactly corresponds to 1 inch.

- Both with G70 and G71, the feedrate is interpreted as being stated in degrees/min if only the rotary axis has been programmed.
- If the rotary axis and the linear axis traverse jointly, they are **assigned the same weighting**, both with G70 and G71.

  **Example:** G70 is preset.

  Both axes have the same distance to traverse: 100 inches or 100 degrees. Therefore, according to the internal computation, both axes traverse at an axis feedrate of 0.7071 x the programmed path feedrate, each, if both distances to be traversed have the same length.

  **Example:** G71 is preset.

  Both axes have the same distance to traverse: 100 mm or 100 degrees. Therefore, according to the internal computation, both axes traverse at an axis feedrate of 0.7071 x the programmed path feedrate, each, if both distances to be traversed have the same length.

## 3.84 Stroke release time (default values) G610
## Stroke release time (interpolation end point reached) G611
## Stroke release time (Inpos window reached) G612

Effect

The point in time the stroke is to be released in a punching or nibbling process (refer to sect. 3.86) can be influenced with G610 through G612. This is a way to activate the stroke release early (which is the rule), e.g.

- to achieve faster machining by releasing the stroke while the positioning axes are still moving,

or to release a stroke not before the axes have come to a safe standstill, e.g.:

- to achieve increased positioning accuracy for axes with poor dynamic characteristics
- for punching with holding-down appliances.

The **stroke release time** required for the above purposes can be programmed as a waiting time which starts to run as soon as the respective axis reaches a specific zero point in time (time reference point).

Two different **zero points** in time (time reference points) may be used:

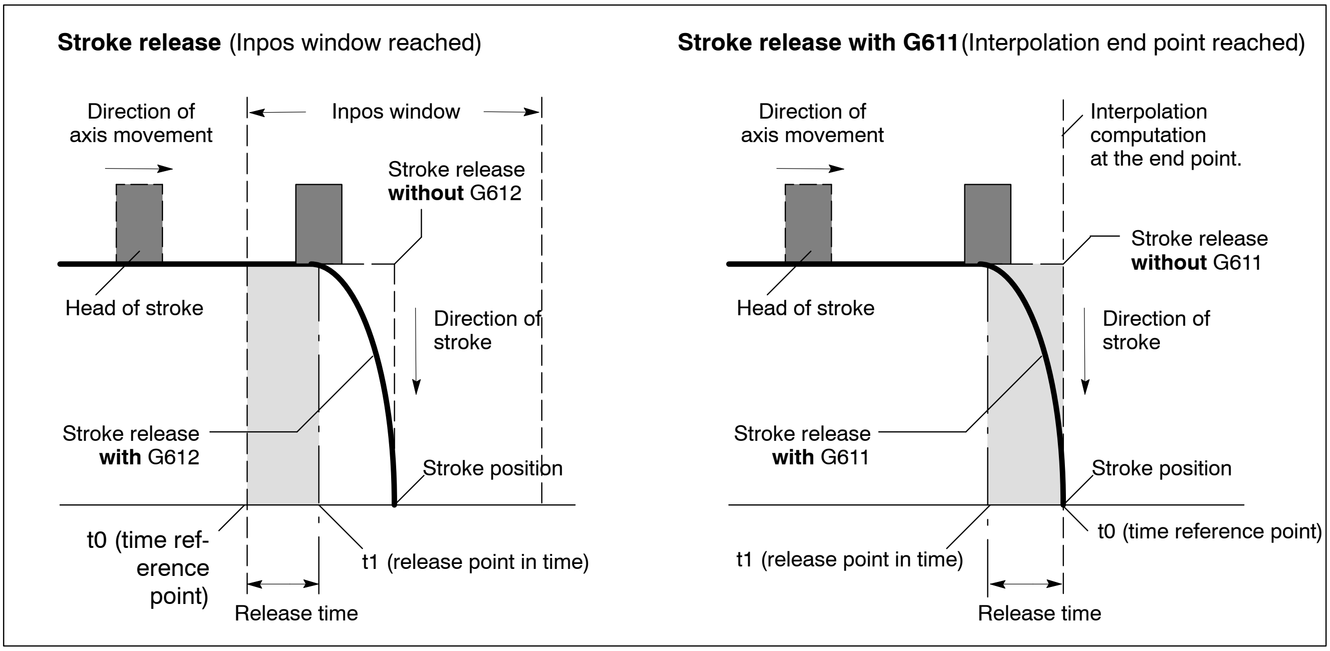
- zero point in time = **time when all axes have reached the inpos window**

  Only when **all** axes involved in the traversing motion have reached their respective inpos windows and the **programmed waiting time** has elapsed will the stroke be released.

- zero point in time = **time when the interpolation has reached its end point**

  The time when the NC interpolation has reached the end point of the traversing motion is taken to be the zero point in time. In this case, you can program both positive times (delayed stroke release) and negative times (early stroke release).

G Instructions G610 G611 G612


**Stroke release** (Inpos window reached)
Direction of axis movement
Inpos window
Stroke release **without** G612
Head of stroke
Direction of stroke
Stroke release **with** G612
Stroke position
t0 (time reference point)
t1 (release point in time)
Release time
**Stroke release with G611** (Interpolation end point reached)
Direction of axis movement
Interpolation computation at the end point.
Stroke release **without** G611
Head of stroke
Direction of stroke
Stroke release **with** G611
Stroke position
t0 (time reference point)
t1 (release point in time)
Release time

What you need to program is the **release point in time** and the **time reference point** for each axis. With every punching block, the ”weakest” criterion of all axes involved in terms of release point in time and time reference point determines the stroke release.

Programming

**G611:** Stroke release time (interpolation has reached the end point)

G611 <Axis name i><$\text{Time}_{\text{Axis name i}}$>...<Axis name n><$\text{Time}_{\text{Axis name n}}$>
where:

axis name i — Name of the i-th logical axis to which a stroke release time is to apply with “interpolation has reached the end point” as the time reference point.

$\text{Time}_{\text{Axis name i}}$ — Waiting time (in ms) of the ith axis referenced to the interpolation reaching the end point where a stroke is to be released.

i — i=1 .. n

Programming

**G612:** Stroke release time (inpos reached)

G612 <Axis name i><$\text{Time}_{\text{Axis name i}}$>...<Axis name n><$\text{Time}_{\text{Axis name n}}$>
where:

axis name i — Name of the i-th logical axis to which a stroke release time is to apply with “inpos window reached” as the time reference point.

$\text{Time}_{\text{Axis name i}}$ — Waiting time (in ms) of the i-th axis referenced to the time the inpos window is reached, where a stroke is to be released.
**Condition:** $\text{Time}_{\text{Axis name i}} \geqslant 0$

i — i=1 .. n

Programming

G610 Setting the MACODA stroke release times (parameters 8001 00010, 8001 00020, 8001 00021) for all axes (default values of times and time reference points).

**Please note the following for G610, G611 and G612:**

- The times programmed are not subject to any restrictions. With negative values in G611, it is possible to achieve an **early stroke release**.
- All time values entered are brought up to match the SERCOS cycle time slots. For each traversing motion, the stroke release behavior is determined by the ”weakest” axis. First the **maximum reference** and then the **maximum time** is computed for all axes involved in the motion.

  Please note for the maximum reference:
  ”Inpos window reached” is **greater** than ”Interpolation has reached the end point”.

**Example:**

- Axis configuration: X, Y, C
- SERCOS cycle time: 3 ms

Initializing the reference axis:

| **Code** | **Time reference point** | **Programmed release time** | **Release time with SERCOS** |
|---|---|---|---|
| `N10 G612 Y10 C2` | ”inpos reached” | X axis: 10 ms<br>C axis:  2 ms | Y axis: 12 ms<br>C axis:  3 ms |
| `N20 G611 X-10`<br>`..` | ”Ipo end point reached” | X axis:–10 ms | X axis: –9 ms |

During machining:

| **Code** | **Stroke release** | **Time reference point and release time** |
|---|---|---|
| `N30 G1 Y20 C10` | Y determines the stroke release behavior (maximum time!). | Time reference point = ”inpos reached”<br>Release time = 12 ms |
| `N40 X20 C20` | C determines the stroke release behavior (maximum reference!). | Time reference point = ”inpos reached”<br>Release time = 3 ms |
| `N50 X30` | X determines the stroke release behavior<br>(only X axis is moved!) | Time reference point = ”inpos reached”<br>Release time = –9 ms |
| `N60 G610` | All axes are assigned MACODA values | All axes are assigned MACODA values |

## 3.85 Tangential tool orientation ON G631<br>Tangential tool orientation OFF G630<br>Tangential tool orientation ON TTON<br>Tangential tool orientation OFF TTOFF

Effect

Function G631, “Tangential tool orientation” is designed to set a punching die **tangentially** to the path with every stroke (possible only when G661, punching, or G662, nibbling, is active).

In all blocks or block segments, the orientation axis of the tool reaches its full tangential angle normally at the **end point** of the path (unlike with ”Tangential tool guidance”, refer to sect. 3.60). The orientation axis moves synchronously with the linear axes, and both arrive at the end point simultaneously.

However, if an additional stroke is released (refer to para 1.2, ”Nibbling”) at the beginning of the first path segment, a block inserted specifically for this purpose turns the orientation axis to the tangential angle of the first block segment. This is to ensure correct tool orientation with every punching stroke.

Together with every orientation motion, the path search logic ”shortest path” is active. If symmetry values greater than 1 occur, the nearest equivalent angle will be approached.

Programming

**G631:** Tangential tool orientation ON

G631 {SYM<s>} {ANG<a>} or

TTON {SYM<s>} {ANG<a>} (Alternative syntax)

where:

SYM SYM defines the symmetry value s. A tool with the symmetry value s returns to its original position after a rotation of 360°/s.

s Any positive integer may be entered for the symmetry value s. It depends on the type of tool used, i.e. a rectangular tool has the symmetry value s = 2, a square tool the symmetry value s = 4, etc.

**Default**: If no entry is made for SYM, the default value for s=1.

ANG ANG is used to enter the offset angle a with the path tangent.

a The offset angle may be in between 180° ≤ a ≤ 180°.

**Default**: If no entry is made for ANG, the default value for a=0.

Programming

## G630: Tangential tool orientation OFF

G630 or TTOFF (Alternative syntax)

**Please note for G630, G631 or TTON/TTOFF:**

- G631 and TTON/TTOFF can be applied only in combination with punching, G661, or nibbling, G662 (also refer to sect. 3.86).
- Although function G631 may be programmed while G660 is active, tangential axis setting will be executed only for punching/nibbling blocks.
- The number of the orientation axis is defined by MACODA parameter 7050 00210, i.e. it is not programmable.
- For circular blocks with block splitting, the tangent to the circular contour is taken as the basis of orientation with each stroke, while the traversing motion from one stroke to the next is linear.
- Functions ”Tangential tool orientation” and ”Tangential tool guidance”, G130, G131, must **never** be active **simultaneously**.

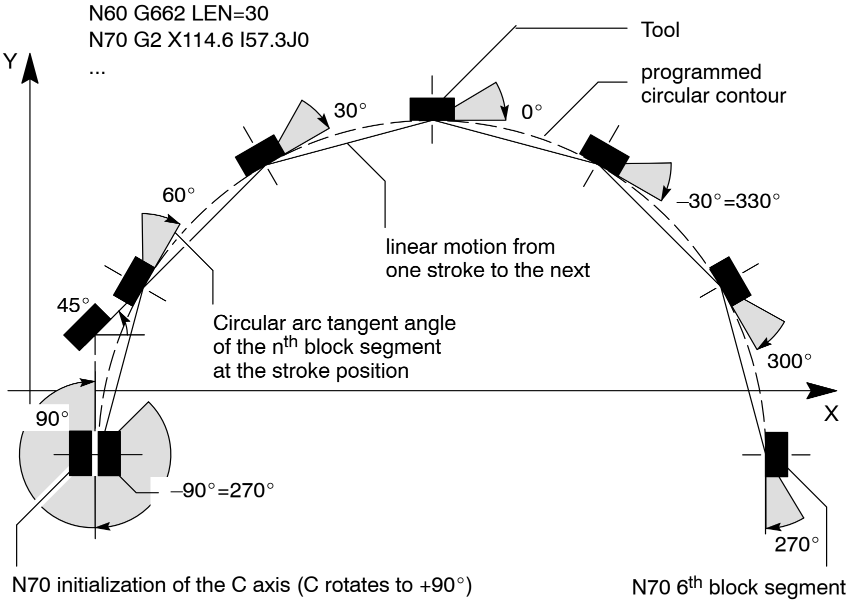
**Example:**

| | |
|---|---|
| `N10 G660 X0Y0C0` | |
| `N10 G631` | Tang. tool orientation ON, symmetry 1, offset angle 0 |
| `N20 G1 G91 X10 Y10` | G660 is still active –> no orientation of the C axis |
| `N30 G661 X10 Y10` | C end point = 45° |
| `N40 Y-10` | C end point = –90°<br>After modulo calculation, C is positioned at 270° |
| `N50 G660` | Punching OFF |
| `N60 G662 LEN=30` | Nibbling ON, with 30 mm path segments |
| `N70 G2 X114.6 I57.3J0` | Semicircle with 180 mm length of arc:<br>– C rotates from 270° to 90°; stroke at starting point<br>– 1st block segment, C from 90° to 60°<br>– 2nd block segment, C from 60° to 30°<br>– 3rd block segment, C from 30° to 0°<br>– 4th block segment, C from 0° to –30° (=330°)<br>– 5th block segment, C from 330° to 300°<br>– 6th block segment, C from 300° to 270°<br>(also refer to fig. below) |
| `N80 G630` | Tangential tool orientation OFF |
| `N90 X200 NUM=2` | C remains at 270° |

**C axis rotation with tangential tool orientation**

Example:

```
...
N30 G661 X10 Y10
N40 Y–10
N50 G660
N60 G662 LEN=30
N70 G2 X114.6 I57.3J0
...
```



## 3.86 Punching/Nibbling OFF G660<br>Punching ON G661<br>Nibbling ON G662

Effect

When G661 or G662 is active, a punching stroke is released at the end of every path segment. The subsequent traversing motion is not started before the stroke is finished.

Basically, punching and nibbling have the same functionality. They differ in the following respects:

- **Stroke release**
  - For punching (G661), strokes are always released at the end of the path or path segment.
  - For nibbling (G662), there are two cases where an additional stroke is released at the beginning of the first path segment:
    – if no traversing movement was programmed in the previous block
    – if G660 is active in the previous block
- **Block splitting**
  - Block splitting need not be active for punching. In this case, every NC block is concluded at its end with a stroke (individual stroke operation).
  - For nibbling, it is mandatory that you program block splitting – either by entering a modal LEN value or a local NUM value. Individual stroke operation is not possible for nibbling.
    Block splitting means that the length of a path on the selected plane is split in block segments of equal size. Any additional axes programmed outside this plane will reach their respective end point not before the last block segment (refer to Example 3).

**No** stroke is released in the following cases:

- No axis coordinate on the selected plane has been programmed (e.g. N..C60). In this case, execution continues without a stroke being released.
- Stroke release is suppressed by the PLC. Execution is suspended at the point of the stroke until the stroke is released by the PLC.

☞ **The point in time when a stroke is to be released can be influenced via G611 and G612 (refer to sect. 3.84).**

Programming

G661 Punching ON

Together with G661, path splitting may be programmed by entering LEN and NUM values.

G662 Nibbling ON

Together with G662, path splitting must be programmed by entering LEN and NUM values.

G660 Punching/Nibbling OFF

LEN=< value> where “value” = **length** of the path segment of an NC block

NUMN=<value> where “value” = **number** of path segments of an NC block

**Please note the following for G660, G661 and G662:**

These three NC functions form a modal group, i.e. at any given point in time, no more than one of these three NC functions is active.
Punching or nibbling functions can be activated only if the desired function is set in MACODA parameter 8001 00010 and if the path shape (G408 or G608) in the NC program is active.

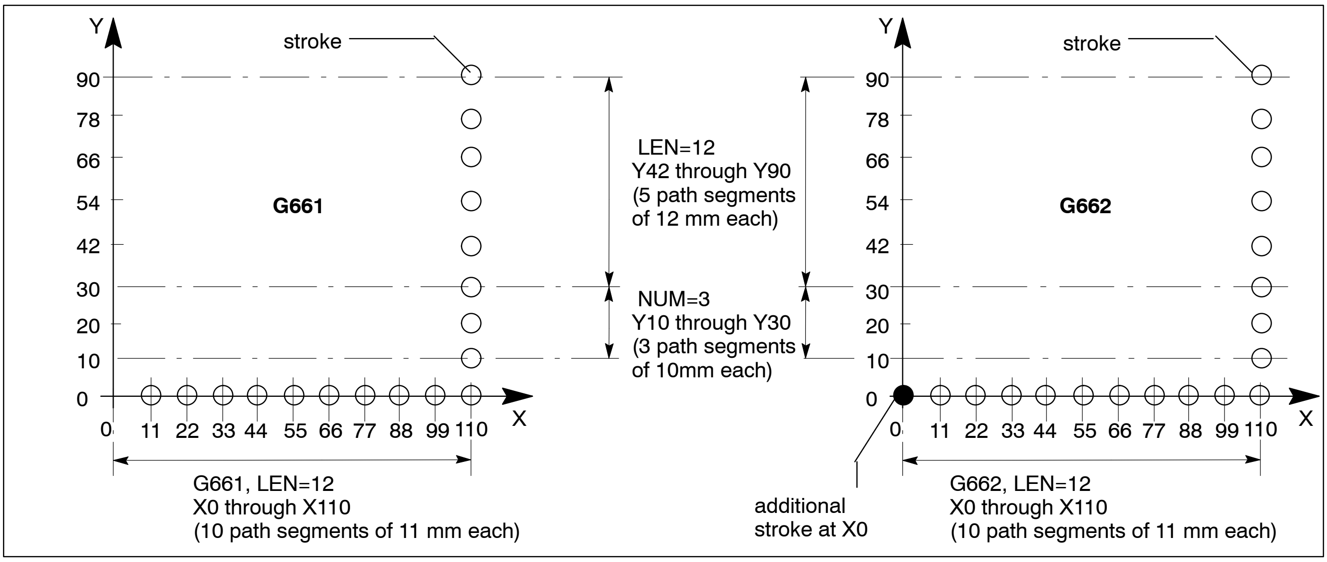
**Please note for LEN:**

- Defines the lengths of the path segments of each NC block.
- May be any positive number. When LEN=0, block splitting is deactivated.
- You do not have to enter an integral divisor of the programmed length of path. Internally, the NC computes an effective LEN value that is lower than/equal to the programmed LEN value with the effect that the effective path segments are integral divisors of the programmed length of path.
- The unit for programming is the same as for the axis coordinates.
- Applies both to linear and to circular blocks. In the latter case, the LEN value refers to the length of the arc. However, the traversing motion from one stroke to the next is always linear.
- May be overwritten block-by-block by NUM (refer to below).
- Can be programmed while G660 is active. However, block splitting begins only after punching or nibbling have been activated.
- Acts modally, i.e. the segmentation of the path applies to all subsequent NC blocks as long as G661 or G662 is active.

**Please note for NUM:**

- Defines the number of path segments of an NC block.
- May be any positive integer.
  NUM=0 is not permitted. NUM=1 does not produce any block splitting.
- Acts locally, i.e. only in the NC block where it is programmed.
- Overwrites block-by-block any path segmentations programmed with an LEN value.
- May be programmed only while either punching or nibbling is active.

## Example: Punching and nibbling



### Punching:

| | |
|---|---|
| `N10 C10 G661` | **Punching** is activated. No stroke is released because X and Y are not programmed. |
| `N20 C60` | No stroke |
| `N30 X0` | Stroke because X is programmed (no traversing motion) |
| `N40 LEN=12` | Path segmentation, 12 mm |
| `N50 X110` | Block is split in 10 path segments of 11 mm, ea.<br>Strokes are executed in positions X11, X22, ..., X99, X110 |
| `N60 Y30 NUM=3` | LEN=12 is overwritten block by block.<br>Strokes at Y10, Y20, Y30 |
| `N70 Y90` | Modal LEN is active again.<br>Strokes at Y42, Y54, Y66, Y78, Y90 |
| `N80 X50Y50 G660` | Punching off. No stroke is released. |

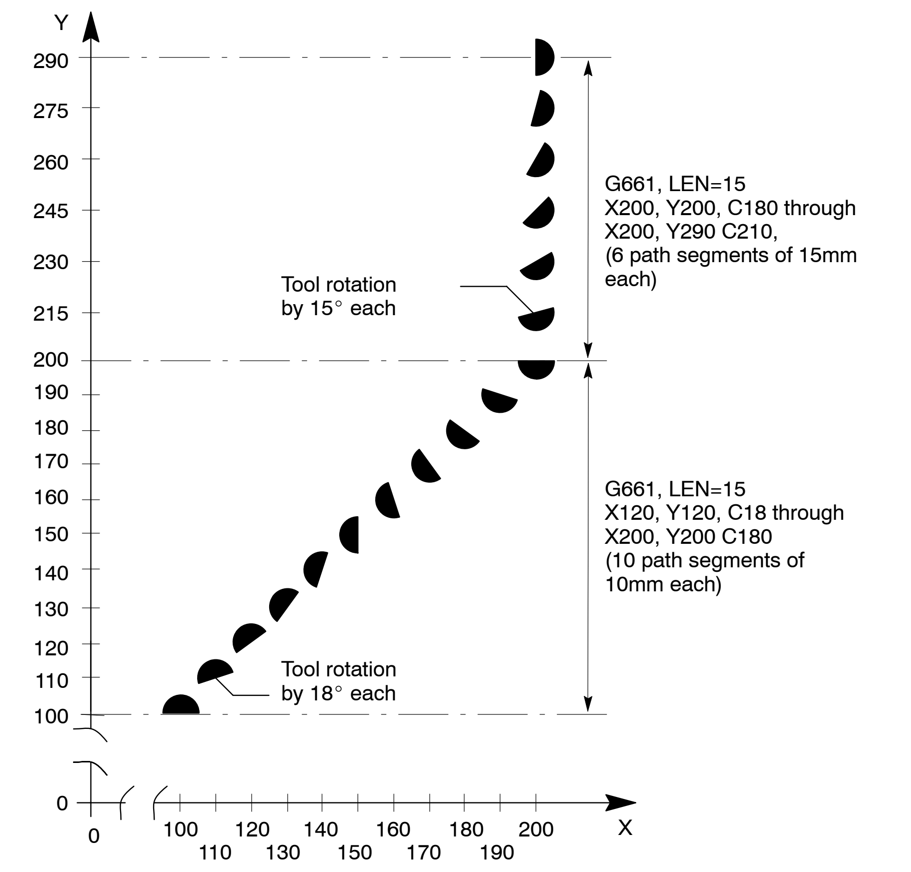
### Nibbling:

| | |
|---|---|
| `N01 X0 Y0 C0 G90 G1 LEN=12` | |
| `N30 C10 G662` | **Nibbling** is activated.<br>No stroke because X and Y are not programmed. |
| `N40 X0` | Stroke because X is programmed. No traversing motion. |
| `N50 X110` | Block is split in 10 path segments of 11 mm, ea.<br>**Additional stroke at X0** because there is no traversing motion in N40.<br>Further strokes at X11, X22, ..., X99, X110 |
| `N60 Y30 NUM=3` | LEN=12 is overwritten block by block.<br>Strokes at Y10, Y20, Y30 |

| | |
|---|---|
| `N70 Y90` | Modal LEN is active again.<br>Strokes at Y42, Y54, Y66, Y78, Y90 |
| `N80 X50Y50 G660` | Nibbling OFF. No stroke is released. |

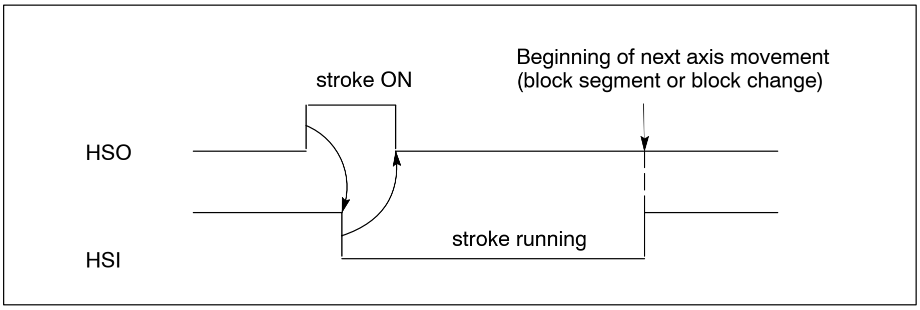
**Example:** Block splitting

| | |
|---|---|
| `N10 G1 X0 Y0 C0 G660` | |
| `N20 X100 Y100 LEN=15` | Length of path segments: 15 mm<br>No block splitting because G660 is active. |
| `N30 X200 Y200 C180 G661` | Punching is activated. Path length of 141.42 mm is split in 10 block segments.<br>Stroke positions at (110,110,18), (120,120,36),...,(200,200,180) |
| `N40 Y290 C210` | Path length of 90 mm is split in 6 block segments.<br>Stroke positions at (200,215,185), (200,230,190),...,(200,290,210) |
| `N50 G660` | Punching off.<br>Subsequently, no more block splitting. |



## 3.86.1 Stroke release

By setting the high-speed output 0 on the DCIO (PNC-R power supply unit) (HSO = 1), the NC instructs the punching control to release a stroke. The punching control acknowledges the job after releasing the stroke by resetting the high-speed input 0 on the DCIO (HSI = 0). Subsequently, the NC resets the HSO.
There is no traversing motion while the stroke is being executed (HSI = 0). The NC starts the next traversing block when HSI = 1.


Beginning of next axis movement
(block segment or block change)
stroke ON
HSO
stroke running
HSI

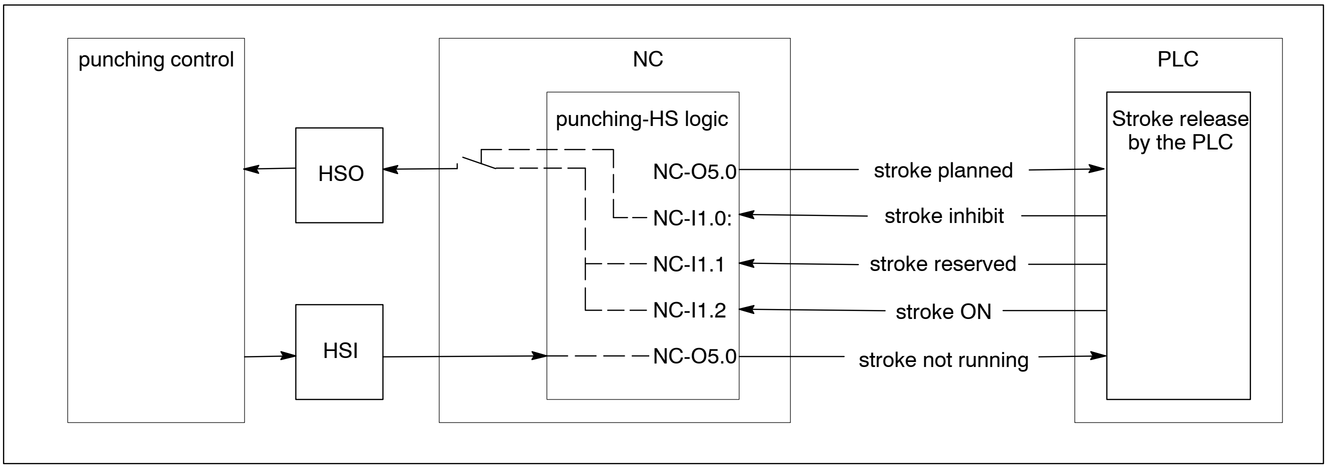
## 3.86.2 Interface signals used in the punching process

It may sometimes be necessary to release individual strokes (if metal sheets get jammed, working position is not properly aligned yet, etc.). The logic required for this can be applied by means of the PLC.

Effect

The PLC can release a stroke by sending an instruction to this effect to the NC via the NC-PLC interface.

Bit signals used between the NC, the punching control and the PLC are as follows:



All signals are available on the “general interface”.

Programming

## NC Outputs

| | | |
|---|---|---|
| NC-O5.0 | ”stroke planned” | With this signal, the NC indicates to the PLC that a stroke is to be released. |
| NC-O5.1 | ”stroke not running” | The high-speed input HSI-0 is relayed by the punch-HS logic to the PLC. |

## NC Inputs

| | | |
|---|---|---|
| NC-I1.0 | ”stroke inhibit” | This allows the PLC to prevent HSO-0 setting. |
| NC-I1.1 | ”stroke reserved” | This allows the PLC to reserve the high-speed output HSO-0 for a stroke release directly by the PLC. |
| NC-I1.2 | ”stroke ON” | The PLC instructs the NC to release a stroke. |

The following signal conditions must be fulfilled at the input of the Ipo-HS logic for a stroke to be released (HSO-0 = 1):

| **Stroke to be released by the NC:** | | **Stroke to be released by the PLC:** | |
|---|---|---|---|
| NC-O5.0: | ”stroke planned”=1 | NC-I1.0: | ”stroke inhibit”=0 |
| NC-I1.0: | ”stroke inhibit”=0 | NC-I1.1: | ”stroke reserved”=1 |
| NC-I1.1: | ”stroke reserved”=0 | MC-O1–2: | ”stroke ON”=1 |

## 3.87 Programming SERCOS ID numbers while in a part program G900

Effect

In a part program, you can specify

- **manufacturer-independent** drive parameters (S-0-..) or
- **manufacturer-specific** drive parameters (P-0-..).

The SERCOS parameters in question must be capable of being modified in SERCOS phase 4.

☞ **Read-only access to SERCOS parameters is possible via the CPL command SCS.**

★ Please note:

- Drive parameters are not written before the individual axis/spindle has come to a stop.
- All SERCOS parameters modified by G900 will be overwritten when the drive is started up for the next time.
- Please note the drive manufacturer-specific weightings of the individual parameters! You can display this weighting in the group mode “Diagnostics” on the SERCOS monitor!

Programming

| | |
|---|---|
| G900-0-... X... Y... Z...<br>G900-0-... X... DRIVE(”VA”,...) | for manufacturer-independent parameters |
| G900 P-0-... X... Y... Z...<br>G900 P-0-... X... DRIVE(5,...) | for manufacturer-specific parameters |

where:

DRIVE(<drive_1>,<value_1>,<drive_2>,<value_2>,...,<drive_8>,<value_8>)

| | |
|---|---|
| | Describes the drive parameters of auxiliary axes or spindles for max. 8 drives |
| <drivex> | Number of the drive or physical axis name |
| <valuex> | Value written to the parameter specified |

**Example**:

| | |
|---|---|
| N.. G900 P-0-0500 X100 Y100 Z99 | SERCOS parameter P-0-0500 is assigned the following values:<br>in the drive of the X axis: 100<br>in the drive of the Y axis: 100<br>in the drive of the Z axis: 99 |

**DANGER**
**Improper changes in SERCOS parameters may cause damage to workpieces and/or the machine and pose a hazard to personnel due to unexpected machine reactions.**

Improper changes in SERCOS parameters may lead to conditions that can only be corrected by a control restart or a drive reset.

**CAUTION**
**The G900 function must not be programmed in machining sections because sudden speed reductions (downslope) may occur.**

Incorrect programming of a SERCOS ID number will produce a runtime error and may have the following causes:

- Invalid SERCOS parameter
- SERCOS parameter cannot be programmed in SERCOS phase 4 or is currently write-protected
- The permitted limit values of the parameter have been exceeded.
- An attempt was made to program a non-SERCOS axis.
- Several SERCOS parameters were entered for each G900 block.

## 3.88 Control area — AREA..
## Area definition — AREADEF
## Activate/deactivate area — AREAVALID

Effect

The **Control area** function monitors traversing motions of machine axes as to whether or not their positions are within the specified rectangular, two-dimensional **areas** with paraxial boundaries. In the event of a violation of a specified area, this function generates a runtime error.

Area

Up to 10 areas can be set in MACODA (NC function parameters/control areas 8002 00001 – 8002 00033) and may be defined as

- **Dead ranges**: These areas must not be crossed over or touched upon by a traversing motion. No starting or end points must be located in a dead range. The area of a dead range extends up to and includes its boundaries!
- **Working range**: The boundaries of the working range must not be exceeded by any traversing motion. Starting and end points must be within the working range. The area of a working range extends up to and includes its boundaries!

**Validity of areas:**

- Every area used is to be assigned **two** system axes in MACODA. These axes define the plane on which the control area is located. Numbering of system axes starts with 1 for the 1$^{st}$ system axis!
  The designation of these axes is **system**-specific as opposed to channel-specific and, therefore, unique within the whole system!
- Every area is unique throughout the system. On every channel, an area can be monitored only if the system axes of both monitoring dimensions are located on the very same channel.
- The axes of an area that is activated must be on the channel because otherwise a runtime error will occur.
- In the case of an **axis transfer** it must be ensured that every area containing one of the transferred axes is **deactivated** on the channel from which the axes were transferred.
- The transfer of an axis from an active area to another channel will cause a runtime error.
- If the axes of an area are transferred to another channel, this area is assigned the default values of the new channel as preset in MACODA and the area is deactivated! The values set on the original channel are not transferred in the process. (This applies also when the axes are retransferred to their original channel!)
- When you activate the jogging mode, the values last activated apply to every area.

**Default status (start-up):**

- The control area data is input as default values from MACODA. The control areas are not active.

**Control reset:**

- The data and status of the control areas are retained unless they are expressly modified by a syntax in the init string.

Monitoring

All data required to monitor an area can be set in MACODA with the effect that all you have to do is activate the respective area in the part program or the cycle.

The areas are monitored in terms of their absolute machine positions.

☞ **Monitoring is only effective for axes with known reference points.**

**Monitoring in automatic mode:**

- In automatic mode, active areas are monitored only if their system axis dimensions on the channel are located on the selected plane.
- Both linear and circular traversing motions are monitored. Blocks not containing any traversing motion are not monitored.

**Monitoring in jog mode:**

- In jog mode, all active areas are monitored. Monitoring is based on the physical axes.
- Only one of the axes defining a dead range may be jogged at a time. If both axes are jogged, an error will occur.
- If an axis is about to touch upon an area boundary, it stops and a warning is displayed.
- There is no monitoring of handwheel operation.
- While the monitoring function is active, use of the ”Inclined plane” function is not permitted if this would involve axes defining the control area.

Programming

**Programming the area:**

A control area is programmed and takes effect on **one** channel only.

Every area can be programmed individually with the following parameters:

- Position
- Expansion
- Type
- Activation of the area programmed
- Deactivation of the area programmed

Dimensions must be entered in accordance with the unit of measurement valid in each case.

Programming

**Modifying area i and programming “activation” or “deactivation” of monitoring at the same time:**

N.. AREADEF ( i,k,type, position1,position2,expansion1,expansion2)

where

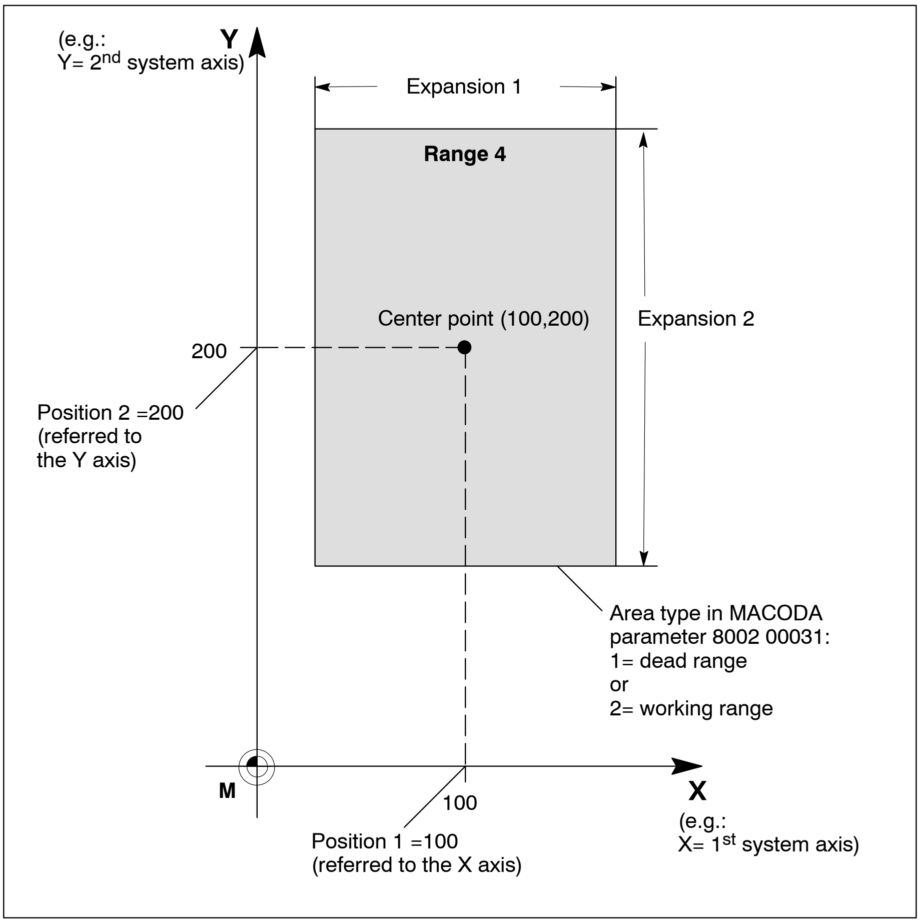
| | |
|---|---|
| i | i=1.. 10 |
| k | 1: modification of the i-th area, including ”activation”<br>0: modification of the i-th area, including ”deactivation” (without activation) |
| Type | 0: not defined<br>1: Dead range<br>2: Working range |
| Position1 | Determination of the center position of the 1st area dimension relating to the number of the 1st system axis set in MACODA parameter 8002 00001 to define the 1st dimension of the i-th area. |
| Position2 | Determination of the center position of the 2nd area dimension relating to the number of the 2nd system axis set in MACODA parameter 8002 00002 to define the 2nd dimension of the i-th area. |
| Expansion1 | Determination of the expansion of the 1st area dimension relating to the number of the 1st system axis set in MACODA parameter 8002 00001 to define the 1st dimension of the i-th area. |
| Expansion2 | Determination of the expansion of the 2nd area dimension relating to the number of the 2nd system axis set in MACODA parameter 8002 00002 to define the 2nd dimension of the i-th area. |

**Please note for AREADEF:**

- The syntax can be applied to a specific area only. If you want to modify more than one area, you must call up the syntax repeatedly.
- **No entries** are required for parameters that are to remain unchanged (provided that there are further parameters following), or these parameters may be **omitted** altogether (if at the very end). Also refer to the example given below.
- If no entries are made for the type, position1, position2, expansion1, or expansion2 parameters, the respective values are retained.
- If the parameter had never been programmed before, the default setting specified in MACODA will be used.

**Example**:

| | |
|---|---|
| `N10 AREADEF(4,0,,100,200)` | Type of range, expansion 1 and expansion 2 remain unchanged (MACODA or previous settings are applied). |



**Activating or deactivating the control area:**

The control areas can be activated and deactivated by programming the appropriate syntax. However, the position, expansion and type of area must be known.

Programming

**Activating or deactivating the control of the i-th area or of all areas on a channel:**

N.. AREAVALID( i , k)

where

| | | |
|---|---|---|
| i | i=1.. 10: | selecting i-th area |
| | i= –1: | selecting all areas on a channel for monitoring |
| k | k=1: | Activate monitoring |
| | k=0: | Deactivate monitoring |

### Settings in MACODA:

There is a box with 10 entries for each of the following MACODA area control parameters. This allows you to define 10 areas:

| | |
|---|---|
| 8002 00001 | System axis number of the 1st axis defining the 1st dimension of the area (e.g., 1 for the 1st system axis ...). |
| 8002 00002 | System axis number of the 2nd axis defining the 2nd dimension of the area. |
| 8002 00011 | 1st dimension center point of the area [mm], relative to the axis as defined by MP 8002 00001. |
| 8002 00012 | 2nd dimension center point of the area [mm], relative to the axis as defined by MP 8002 00002. |
| 8002 00021 | 1st dimension expansion of the area [mm], relative to the axis as defined by MP 8002 00001. |
| 8002 00022 | 2nd dimension expansion of the area [mm], relative to the axis as defined by MP 8002 00002. |
| 8002 00031 | Type of control area:<br>0: not defined<br>1: dead range<br>2: working range |
| 8002 00032 | Area can be modified by reprogramming:<br>0: no<br>1: yes |

## 3.89 Diameter programming (rotating function) DIA RAD

Effect

Workpieces processed on lathes usually have a rotation symmetrical cross-section. With the functionality “**diameter programming**”, it is possible to program the coordinates of the plain axis (mostly the X axis) as **diameter value** or **radius value**, optionally.
As a result, dimensions may be taken from the technical drawing into the part program directly and without requiring any conversions.

The following conditions apply to the plain axis with active diameter programming (operating modes “manual input” and “automatic”):

| **Programming of the plain axis with the NC functions** | **interpreted as** |
|---|---|
| Axis address under G90 (absolute) | Diameter value |
| Axis address under G91 (incremental) | Radius value |
| Coordinate for circle center point | Radius value |
| X = AC(...) | Diameter value |
| X = IC(...) | Radius value |
| Tool lengths | Radius value |
| Zero shift data | Radius value |

With active diameter programming, the following conditions apply in the operating mode “manual” under “jog mode” and “handwheel mode” for the plain axis:

- an incremental path specification may be interpreted optionally as radius or diameter difference (provided that the axis-specific interface signal ”incremental step as diameter” has been set).

With active diameter programming, several displays for the respective axis are represented as diameter values and sign ”∅”. Other displays are not affected by this.

| **Display as diameter values** | **Display as radius values** |
|---|---|
| – Workpiece position<br>– Distance to go<br>– End Position<br>– Program Value | – Machine Position<br>– Axis Position Values<br>– Lag |

Programming

DIA “Diameter programming” active:
The programmed coordinates of the plain axis are interpreted as diameter values.

RAD Radius programming” active:
The programmed coordinates of the plain axis are interpreted as radius values.

**Example:**

```
N10... DIA
...
```

**Please note for DIA, RAD:**

- Diameter programming is effective for the plain axis in operating modes manual and automatic only if it is programmed absolutely:
  Example:
  G90 X ..., G91 X=AC(...).
- The switch on behavior as well as the behavior upon Control Reset can be determined by the corresponding configuration of the init strings in MACODA parameters 7030 00010 and 7030 00020 (DIA for diameter programming).
- The diameter programming refers exclusively to the axis of a channel that is assigned the machining-technological meaning ”X axis” (MACODA parameter 7010 00030 axis classification = 1) in the default setting.
- In case of active axis transformation and the connected working range coordinate programming, diameter programming is not permitted and results in a runtime error.
- The program-flow related behavior is compatible with Typ1 osa T.

## 3.90 Programmable position-dependent HS output PDHSO(..)

Effect

Using the “Programmable position-dependent HS output” function, you can set a High-Speed Output in the part program.
High-Speed outputs are available for:

- PNC-P: on the expansion board “PNC Highspeed I/O”
- PNC-R: on modules “osa dc I/O” or “osa dc I/O ana”.

(refer to PNC-P Interface conditions and PNC-R Interface conditions manuals).

An HS output may be set with reference to the distance between the start and end point of an NC block for a programmable period of time.
Only one HS output may be assigned to a channel. The assignment is performed via presetting in MACODA parameter 4075 00102.
The programmed configuration data (release delay, time) have a modal effect.
The setting of the HS output acts block by block.

Programming

The HS output is programmed in two steps:

1. “Configuration”:
   **PDHSO**(<Set>,{<Release delay>},{<Time>})

2. Set HS output:
   **PDHSO**(<Set>)

where:

<Set> 0: configures the function in the program
1: activates the function for the active NC block

<Release delay> **Optional, default: 0 mm (at the block end)**
The release delay (value in mm or inch) defines the distance from the start or to the end of the block. Upon reaching this value, the signal is set.

| | |
|---|---|
| Release delay>0: | Distance from beginning of block |
| Release delay≦0: | Distance from end of block |

<Time> **Optional, default: 30 ms**
The time defines for how long the HS output is set.
Value range: 0.5 ms...10000 ms
The time value is internally rounded to the next larger integer multiple of the NC cycle time.

**Example:**

```
N05 G71
N10 PDHSO(0,-1.2,40)
```

Configuration: When calling up PDHSO(1), the HS output is set for approx. 40 msec at about 1.2 mm prior to reaching the endpoint.

```
. . .
N210 . . .
N220 G1 G91 F1000 X10 Y23
     PDHSO(1)
N230 . . .
```

In block N220, the HS signal is set with the values configured above.

**Please note for PDHSO(...):**

- The programmed configuration data (release delay, time) have a modal effect.
- The setting of the HS output acts block by block.
- An HS output which has already been activated is not affected by control reset. The time that has been set runs.
- The specified release delay refers to the axis setpoint position of the NC.
  The delays caused by the processing of the setpoint positions (fine interpolation) and the lag of the axes are not taken into account.
- If the HS output has been set (time starts running) and a new job arrives, the still active job is deleted and the new job is executed.

☞ **Since there is no change in the voltage level at the HS output, the external hardware is not capable of recognizing the new job.**

## 3.91 Limitation of the maximum number of prepared blocks PREPNUM

Effect

Using the PREPNUM function, the maximum number of blocks prepared by the block preparation function can be limited.

If more blocks have been prepared than specified by the PREPNUM function when PREPNUM is programmed, preparation of additional blocks will be stopped until the number of blocks prepared is identical with the number specified by the PREPNUM function.

For example, the PREPNUM function can be used to control further processing of runtime measuring results in the part program.

Programming

| | |
|---|---|
| PREPNUM 5 | Block preparation is limited to 5. |
| PREPNUM 0 | Block preparation will use the maximum number of blocks prepared available within the NC. |

Please note for the PREPNUM function:

- If the number of blocks programmed exceeds the maximum number of prepared blocks available in the NC, block preparation will have the same effect as if PREPNUM 0 had been programmed. The maximum number of prepared blocks available is defined in MACODA parameter 7060 00110.

## 3.92 NC synchronization functions: WAITA, WAITO WPV, WPVE SPV, SPVE ASTOPA, BSTOPA, WSTOPA, ASTOPO, BSTOPO, WSTOPO OFFSTOPA, OFFSTOPO

Effect

In the PNC, one program per channel can run. If the individual processing segments are divided into the different individual programs and if these programs run on **different channels**, the processing sequence of all individual programs can be controlled by sequence-related **NC synchronization functions**.

The following **NC synchronization functions are available:**

- “Waiting for interface signals”, at runtime
- “Waiting for the value of a permanent CPL variable”, at runtime
- “Writing of permanent CPL variables”, at runtime
- Channel synchronization by “movement stop”

### Waiting for statuses at the digital interface

Using the function WAITA / WAITO starts the waiting process, at runtime, until one or out of a maximum of 16 interface signals have reached a specified value.

**Conditions** for waiting when **several** interface signals have been specified:

- **WAITA**: “AND logic” of the individual signals.
  Waiting for several interface signals until **all** individual signals have reached the specified value.
- **WAITO**: “OR logic” of the individual signals.
  Waiting for several interface signals until **one** individual signal from the list of interface signals specified has reached the specified value.

**Example**:
Program 1 on channel 1 processes the end face of a turned part. Program 2 in channel 2 has to cut a groove into the end face and has to **wait** for program 1 to release the turned part for program 2. The release for program 2 is effected by **setting** certain interface signals. When the interface signal(s) has (have) reached the status, channel 1 transmits the release to channel 2. While program 2 is machining, program 1 is waiting for program 2 in order to continue its machining task.

### Waiting for the value of a permanent CPL variable

Using the function WPV / WPVE starts the waiting process, at runtime, until a permanent CPL variable has reached a certain comparison value.

**Conditions** for the evaluation of the comparison value:

- **WPV**: The comparison value is a CPL term which is compared with the value of the permanent variable **at active time**.

  The evaluation of the CPL term, at runtime, is restricted in that only a simple CPL term is permissible.

- **WPVE**: The comparison value is a CPL term which, although determined **at preparation time**, is compared with the value of the permanent variable only at runtime.

The comparison operators “equal to”, “not equal to”, “less than”, “less than or equal to”, “greater than” and “greater than or equal to” are admissible for WPV and WPVE.

### Writing of permanent CPL variables

Using the function SPV / SPVE, a value is assigned to a permanent CPL variable by **writing**, at runtime.

**Conditions**, under which a value is assigned to the permanent variable.

- **SPV**: the value to be assigned to the permanent variable is only defined at runtime.

  The evaluation of the CPL term, at runtime, is restricted in that only a simple CPL term is permissible.

- **SPVE**: the value to be assigned to the permanent variable is determined at preparation time (CPL interpretation time) but only assigned to the permanent CPL variable at runtime.

### Channel synchronization by movement stop

Using this synchronization function, it is possible to synchronize movements between channels. Depending on the **position** of one or several axes/coordinates on a channel, the synchronous movement on another channel is stopped and resumed.

Restrictions:

- The axes/coordinates used for synchronization have to belong to a channel other than the channel to be controlled, otherwise a modal interlock may occur.
- The channel to be controlled has to be in automatic or manual input mode.

- If AND conditions and OR condition(s) have been specified for the channel simultaneously, there will be a channel stop when the respective condition for at least one of the two functions has been fulfilled.

**Conditions** for a movement stop:

A single or several conditions may be specified which stop the specified channel.

- **ASTOPA, BSTOPA, WSTOPA**: as logical AND link (all conditions have to be fulfilled) or
- **ASTOPO, BSTOPO, WSTOPO**: as logical OR link (at least one of the conditions has to be fulfilled)

**Triggering** a movement stop:

Each condition is defined by the designation of an axis/coordinate and a corresponding threshold value (position) where the channel is supposed to stop.

Programming

WAITA / WAITO:
At runtime, the system will wait until **every** (WAITA) or **one** (WAITO) of the specified signals has reached the specified value.

Waiting for **each** of the specified signals:

**WAITA**(IC(<Parameter>) {= <Status>}, IC(<Parameter>) {= <Status>}, ....,{<Timeout>})

Waiting for **one** of the specified signals:

**WAITO**(IC(<Parameter>) {= <Status>}, IC(<Parameter>) {= <Status>}, ....,{<Timeout>})

where:

| | |
|---|---|
| IC(<Parameter>) | IC function for the digital interface between NC and PLC. Queries inputs and outputs of the PNC.<br>**optional**: 2 to 16 interface signals can be queried simultaneously. |
| Parameter | Transfer parameter of the IC function: Bit, group, index (refer to CPL programming manual) |
| Status | **optional, default: TRUE**<br>Boolean term used to compare the result of the IC function. If the condition is fulfilled, the block processing will start again. |
| Timeout | Time in ms.<br>**optional, default: 0**<br>If “timeout” elapses before the pertaining condition has been fulfilled, a warning is triggered and the system continues to wait.<br>If timeout has not been programmed or is equal to 0, no warning will be triggered. |

Example 1:

```
N10 WAITO(IC(10,1,1)=FALSE,
    IC(11,1,2) )
```

Waits actively until IC(10,1,1) reaches the value 0 or IC(11,1,2) reaches the value 1.

Example 2:

```
N10 WAITA(IC(10,1,1)=FALSE,
    IC(11,1,2))
```

Waits actively until IC(10,1,1) reaches the value 0 and IC(11,1,2) reaches the value 1.

**Please note for WAITA, WAITO:**

- If WAITA and WAITO are programmed in one NC block, block execution is stopped until both conditions have been fulfilled. The WAITO condition is evaluated first.

**The functions WAITA, WAITO, WVP, WVPE implicitly effect a downslope at the end of the block. Synchronization points which have been set incorrectly can lead to damage of the machine.**
**Test the program sequence for possible synchronization problems prior to the actual processing.**

Programming

WPV / WPVE:
The system waits until a permanent CPL variable has reached a certain comparison value.

The comparison value is determined at **runtime**:

**WPV**(<name of the perm. var.> <Comparison operator> <simple CPL term> {,<Timeout>})

The comparison value can be determined at **preparation time** already:

**WPVE**(<name of the perm. var.> <Comparison operator> <CPL term> {,<Timeout>})

where:

name of the perm. var. The permanent variable is marked by sign “@” followed by a variable name.

Comparison operator The following comparison operators may be selected:

= Permanent CPL variable is **equal to** value of CPL term. Only makes sense in case of integer or Boolean values.

< > Permanent CPL variable is **not equal to** value of CPL term. Only makes sense in case of integer or Boolean values.

< Permanent CPL variable is **less than** value of CPL term.

| | |
|---|---|
| ≦ | Permanent CPL variable is **less than or equal to** value of CPL term. |
| > | Permanent CPL variable is **greater than** value of CPL term. |
| ≧ | Permanent CPL variable is **greater than or equal to** value of CPL term. |

simple CPL term

To avoid impairment of movement generation, the evaluation of CPL terms at runtime is restricted, they are designated as **simple CPL terms**.
A simple CPL term is a mathematical term, consisting of permanent CPL variables, constants and the mathematical operations allowed in CPL (refer to CPL programming manual).

CPL term

Mathematical term in the programming language “CPL”.

Timeout

**optional, default: 0, unit in ms**
Timeout limits the time until when the related condition has to be fulfilled. If the time is exceeded, a warning is triggered and the system continues to wait.
If timeout has not been programmed or is equal to 0, no warning will be triggered.

**Example 1**:

```
N10 WPV(@9 = 10)
```

The program waits at the active point in time until the perm. variable @9 reaches the value 10.

**Example 2**:

```
N10 WPVE(@8 = (5 * #VAR2%))
```

The term ”5 * #VAR2%” is evaluated at preparation time. The value thus determined is compared to permanent variable @8 at runtime. As long as @8 does not correspond to the value determined, no new NC block will become active.

Programming

SPV / SPVE:
A value is assigned to the permanent CPL variable at runtime.

The value to be assigned is determined at **runtime**:
**SPV**(<name of the perm. var.> = <simple CPL term)

The value to be assigned is determined at **preparation time**:
**SPVE**(<name of the perm. var.> = <CPL term>)

where:

name of the perm. var. — The permanent variable is marked by sign “@” followed by a variable name.

simple CPL term — refer to function “WPV”.

CPL term — Mathematical term in the programming language “CPL”.

**Example 1**: SPV

```
N10 SPV(@6 = 1)
```

Value 1 is assigned to the perm. variable ”@6” at runtime.

```
N10 SPV(@5 = (7 * (@PERMVAR1% +5 )))
```

The value of the term (7*(@PERMVAR1% + 5)) is determined at runtime and then assigned @6.

**Example 2**: SPVE

```
N10  SPVE(@5 = (7 * #VAR1%))
```

The value of the term (7 * #VAR1%) is determined at preparation time and assigned @5 at runtime.

**Please note for WPV, WPVE, SPV, SPVE:**

- The permanent CPL variables used in the functions WPV, WPVE, SPV, SPVE are valid throughout the system. Therefore, the programmer has to make sure that incorrect use does not lead to unintended function interaction.
- In the framework of the NC functions offered, only the following simple types of permanent CPL variables are permitted:
  – INT
  – BOOL
  – REAL
  – DOUBLE
  In case of arrays, only individual elements may be addressed!

Programming

Channel synchronization with **AND condition(s)** for channel stop:

It is possible to specify several conditions simultaneously for each channel to be controlled in an NC function. As long as all conditions are fulfilled, the synchronous movement of the channel to be controlled will be stopped. If new AND conditions are specified for the channel, the AND conditions applicable for the channel until then will become ineffective.

**ASTOPA**(<Channel number >,<Cond.1>{,<Cond.2>{..{,<Cond.8>}}})
**BSTOPA**(<Channel number >,<Cond.1>{,<Cond.2>{..{,<Cond.8>}}})
**WSTOPA**(<Channel number >,<Cond.1>{,<Cond.2>{..{,<Cond.8>}}})

Programming

Channel synchronization with **OR condition(s)** for channel stop:

It is possible to specify several conditions simultaneously for each channel to be controlled in an NC function. As long as at least one condition is fulfilled, the synchronous movement of the channel to be controlled will be stopped.
If new OR conditions are specified for the channel, the OR conditions applicable for the channel until then will become ineffective.

**ASTOPO**(<Channel number >,<Cond.1>{,<Cond.2>{..{,<Cond.8>}}})
**BSTOPO**(<Channel number >,<Cond.1>{,<Cond.2>{..{,<Cond.8>}}})
**WSTOPO**(<Channel number >,<Cond.1>{,<Cond.2>{..{,<Cond.8>}}})

where:

| | |
|---|---|
| ASTOPA, ASTOPO | Specification of the conditions with the axis positions related to the machine coordinate system (MCS or ACS). |
| BSTOPA, BSTOPO | Specification of the conditions with the axis positions related to the basis workpiece coordinate system (BCS ). |
| WSTOPA,<br>WSTOPO | Specification of the conditions with the axis positions related to the workpiece coordinate system (WCS ). |
| Channel number | Number of the channel to be controlled (1..n). Integer value or integer variable. |
| cond.1,<br>cond.2.. cond.8 | Specification of at least 1, optionally up to 8 conditions with a “greater than/less than comparison” each of an axis/coordinate with a threshold value. |

Each condition has the form:
<Axis/coordinate> <Comparison operator> <Comparison value>

where:

| | |
|---|---|
| Axis/coordinate | An **axis** can be described by its physical name or the system axis number.<br>A **coordinate** can be described by its logical name or the channel coordinate number. Axis and coordinate names have to be programmed as CPL string constant or as CPL string variable. |
| Comparison operator | permitted operators: < , <= , > , >= |
| Comparison value | Real value or real CPL term. The value is determined at preparation time and has a modal effect. |

☞ **From one channel, a maximum of 4 other channels can be stopped by AND/OR conditions.**

**Example 1**: Use of axis names and numbers

```
10 AXISNO% = 2
20 AXISNAME$ = ”X”
30 STOPCHAN% = 2
..
N40 ASTOPO(STOPCHAN%, AXISNO% < 10)
..
N90 ASTOPO(STOPCHAN%, ”Z” > 20.3)
..
N150 ASTOPO(STOPCHAN%, AXISNAME$<1.5)
```

Definition:
- axis number,
- axis name
- channel number

**Example 2**: Activating an AND condition for workpiece coordinates

```
N10 WSTOPA(3, ”z”<12.0, ”x”>15)
```

Channel 3 is stopped for as long as the following is applicable to the controlling channel:
Position of the workpiece coordinate (WCS) z of the channel is less than 12 mm and position of the workpiece coordinate (WCS) x is greater than 15 mm.

Programming

Canceling all OR stop conditions in the control channel:

**OFFSTOPO** Deletes OR condition(s) for channel stop.

Programming

Canceling all AND stop conditions in the control channel:

**OFFSTOPA** Deletes AND condition(s) for channel stop.

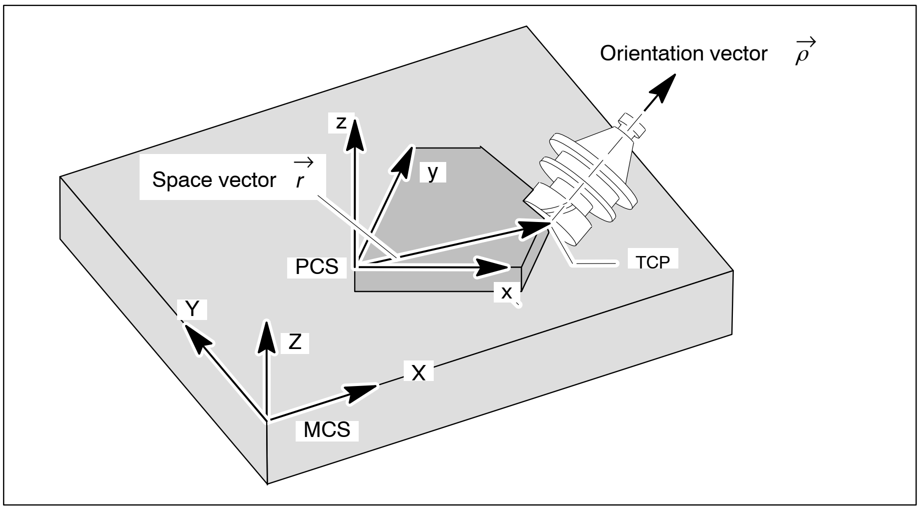
**Please note for the synchronization functions:**

- ASTOPO, BSTOPO, WSTOPO, OFFSTOPO act modally and cancel each other mutually.
- ASTOPA, BSTOPA, WSTOPA, OFFSTOPA act modally and cancel each other mutually.

## 3.93 Orientation programming phi, theta, psi, O(), ROTAX()

Using **orientation**, the main axes of a tool, laser, gripping device and similar are aligned in a specified direction in space.

Depending on machine kinematics, the orientation in space is conditional on whether one, two or three **orientation coordinates** are used. Orientation relates to the program coordinate system (PCS) and is determined by the angles $\varphi$ (phi), $\vartheta$ (theta), $\psi$ (psi) or Cartesian components (refer to tensor, vector orientation).



The PNC distinguishes between the following four **orientation movements**.

- **Linear orientation with axis programming**
  The orientation of the tool (cutter, laser, gripping tool) is programmed by specifying the angle of the rotation axes acting on the tool.
  The motion depends on the special axis kinematics. This type of programming only makes sense for axis kinematics whose rotary axis positions can be mapped one-to-one to the polar coordinate positions.

- **Linear orientation with coordinate programming**
  The orientation of the tool (cutter, laser, gripping tool) is programmed by specifying angles $\varphi$ and $\vartheta$ of the orientation vector.
  The motion is independent of the special axis kinematics.

- **Vector orientation**
  This function relates to rotation symmetrical tools (e.g. lasers, cutters). The motion is described by two orientation coordinates or a Cartesian orientation vector with the same meaning $\vec{\rho}$.
  The motion is independent of the special axis kinematics.

- **Tensor orientation**
  This function relates to non-rotation symmetrical tools (e.g. gripping tools). The motion is determined by the three Eulerian angles $\varphi$, $\vartheta$ and $\psi$, or by a 3x3 orientation tensor.
  The motion is independent of the special axis kinematics.

## 3.93.1 Linear orientation with axis programming

The orientation of the tool (cutter, laser, gripping tool) is programmed by specifying the angle of the rotation axes acting on the tool.
The orientation movement is performed as a linear interpolation of the rotary axes. Therefore, the motion depends on the special axis kinematics. This type of programming only makes sense for axis kinematics whose rotary axis positions can be mapped one-to-one to the polar coordinate positions.

Condition:

- For “Linear orientation with axis programming” it is necessary to activate an axis transformation with orientation identification 2. This is done with the aid of function Coord(<i>).
- All names of the programmed coordinates have been defined in MACODA parameter 7080 00010.
  [1] x
  [2] y
  [3] z
  [4] B
  [5] C

Programming

The **orientation movement of the rotary axes and the coordinates** can be programmed as follows:

N.. {NC fct} {x.. y.. z..} B.. C..

where

| | |
|---|---|
| NC fct | Function which expects the working range coordinates as local parameters (refer to table on page 3–260) |
| x, y, z | Working range coordinates if a TCP movement is to be generated in addition to the orientation movement |
| B.. C.. | Orientation with rotary axis names “B” and “C”.<br>Programming only possible absolutely and in degrees. |

**Example**: G1 x10 y50 z30 B90 C90

**Please note for linear orientation with axis programming:**

- The following angle intervals are applicable to the rotary axis positions:
  $0° \leq B < 360°$
  $0° \leq C \leq 360°$
  A special path logic ensures that no rotation by more than 180° is executed.

## 3.93.2 Linear orientation with coordinate programming

Linear orientation with coordinate programming uses coordinates phi ($\varphi$) and theta ($\vartheta$) to orientate a tool (cutter, laser, gripping tool) in space. In contrast to vector orientation, the movement is not carried out as a movement of the coordinate of rotation, but rather as a linear interpolation in $\varphi$ and $\vartheta$, i.e. as a straight line on an imaginary $\varphi - \vartheta$ plane.

☞ **For details, please refer to “PNC description of functions” manual, Orientation movement of the tool section.**

Condition:

- For “Linear orientation with coordinate programming” it is necessary to activate an axis transformation with orientation identification 2. This is done with the aid of function Coord(<i>).
  Afterwards, orientation coordinates $\varphi$ and $\vartheta$ can be programmed.
- All names of the programmed coordinates have been defined in MACODA parameter 7080 00010.
  [1] x
  [2] y
  [3] z
  [4] phi
  [5] theta

Programming

The **orientation movement of the orientation vector** can be programmed using one of the following three alternatives:

- N.. {NC fct} {x.. y.. z..} phi<$\varphi$> theta<$\vartheta$>
- N.. {NC fct} {x.. y.. z..} O(<$\varphi$>,<$\vartheta$>)
- N.. {NC fct} {x.. y.. z..} O(<$\rho_x$>,<$\rho_y$>,<$\rho_z$)

where

| | |
|---|---|
| NC fct | Function which expects the working range coordinates as local parameters (refer to table on page 3–260) |
| x, y, z | Working range coordinates if a TCP movement is to be generated in addition to the orientation movement |
| phi<$\varphi$> theta<$\vartheta$>: | Orientation using angle names “phi” and “theta” and angles $\varphi$, $\vartheta$. Programming possible absolutely and in degrees.<br>**Example:** G1 x10 y50 z30 phi90 theta90 |

O(<φ>,<ϑ>): Orientation with function O(...) and polar angles φ, ϑ of the orientation vector. Programming only possible absolutely and in degrees.
**Example**: G1 x10 y50 z30 O(90,90)

O(<$\rho_x$>,<$\rho_y$>,<$\rho_z$>): Orientation with function O(...) and the Cartesian components $\rho_{x,}$ $\rho_{y,}$ $\rho_z$ of the orientation vector. Components are automatically standardized to 1 within the NC. Programming is only possible absolutely.
**Example**: G1 x10 y50 z30 O(0,1,0)

☞ **The distinction between angle and vector programming is made by the number of parameters in function “O(..)”.**

**Please note for linear orientation with coordinate programming:**

- The following intervals are applicable to the angles:
  $0° \leq \varphi < 360°$
  $0° \leq \vartheta \leq 180°$

  A special path logic ensures that no rotation by more than 180° is executed.

G Instructions phi theta psi O() ROTAX()

### 3.93.3 Vector orientation

Effect

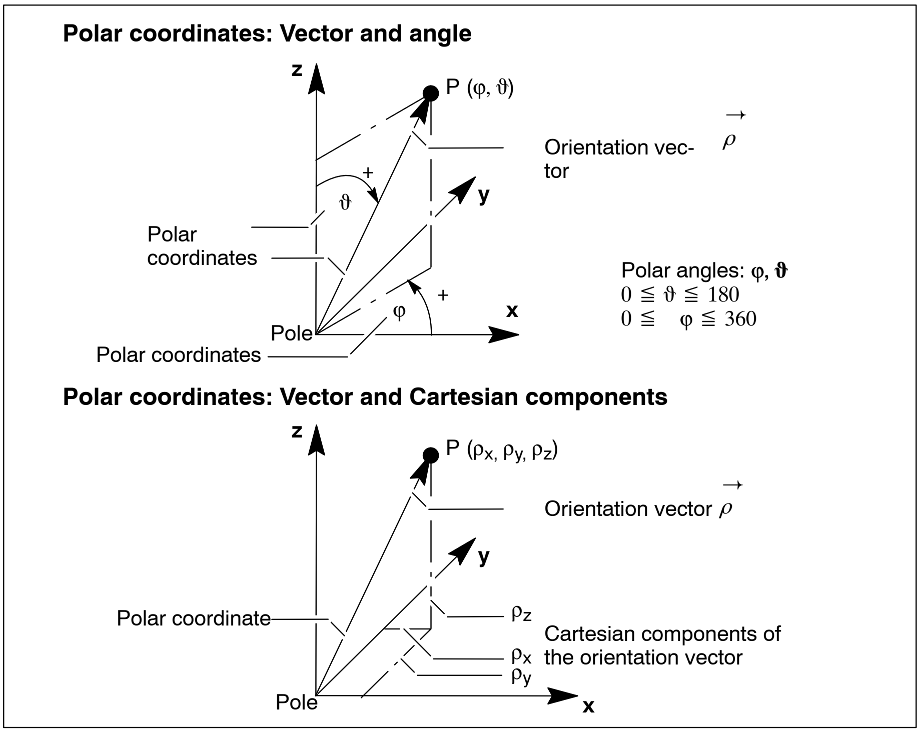
Vector orientation relates **to rotation symmetrical tools** (e.g. lasers, cutters). The motion is described by two orientation coordinates or a Cartesian orientation vector with the same meaning $\vec{\rho}$. The motion is carried out as a rotary movement of the orientation vector from the programmed initial orientation to the final orientation. The motion is independent of the special axis kinematics.

The tool orientation is represented by an orientation vector with a length of 1:

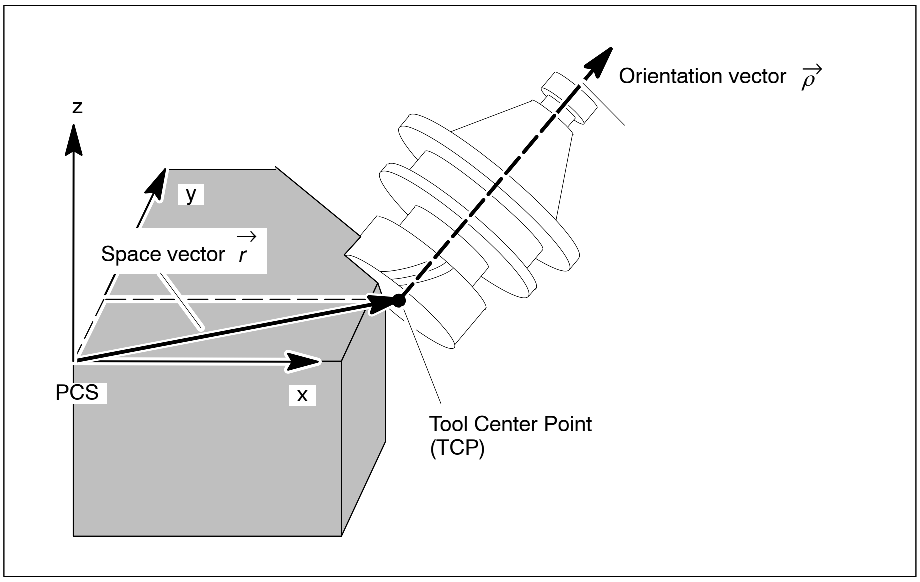
$$\vec{\rho} = \begin{bmatrix} \rho_x \\ \rho_y \\ \rho_z \end{bmatrix} \quad \text{where } |\vec{\rho}| = \sqrt{\rho_x^2 + \rho_y^2 + \rho_z^2} = 1$$

or by the two angles phi (φ) and theta (ϑ) (polar coordinates).
The following relationship applies to the vector components of $\vec{\rho}$and the polar angles:

$$\vec{\rho} = \begin{bmatrix} \cos\varphi \sin\vartheta \\ \sin\varphi \sin\vartheta \\ \cos\vartheta \end{bmatrix}$$


Polar coordinates: Vector and angle
z
P (φ, ϑ)
Orientation vector $\vec{\rho}$
+
ϑ
y
Polar coordinates
+
φ
x
Pole
Polar coordinates
Polar angles: φ, ϑ
0 ≦ ϑ ≦ 180
0 ≦ φ ≦ 360
Polar coordinates: Vector and Cartesian components
z
P (ρx, ρy, ρz)
Orientation vector $\vec{\rho}$
y
Polar coordinate
ρz
Cartesian components of the orientation vector
ρx
ρy
Pole
x

The orientation vector is located along the tool symmetry axis and points to the tool holder (refer to the figure).



A movement of the orientation vector corresponds to a movement of the tool longitudinal axis around its TCP.

☞ **For details, please refer to “PNC description of functions” manual, Tool orientation section.**

Condition:

- For vector orientation it is necessary to activate an axis transformation with orientation identification 2. This is done with the aid of function Coord(<i>).
  Afterwards, orientation coordinates φ and ϑ can be programmed.
- All names of the programmed coordinates have been defined in MACODA parameter 7080 00010.
  [1] x
  [2] y
  [3] z
  [4] phi
  [5] theta

Programming

The **vector orientation** can be programmed using one of the following six alternatives:

- N.. {NC fct} {x.. y.. z..} phi<$\varphi$> theta<$\vartheta$>
- N.. {NC fct} {x.. y.. z..} O(<$\beta$>)
- N.. {NC fct} {x.. y.. z..} O(<$\varphi$>,<$\vartheta$>)
- N.. {NC fct} {x.. y.. z..} O(<$\rho_x$>,<$\rho_y$>,<$\rho_z$>)
- N.. {NC fct.} {x.. y.. z..} ROTAX(<$\varphi_u$>,<$\vartheta_u$>) O(<$\beta$>)
- N.. {NC fct} {x.. y.. z..} ROTAX(<$u_x$>,<$u_y$>,<$u_z$>) O(<$\beta$>)

where

NC fct Function which expects the working range coordinates as local parameters (refer to table on page 3–260)

x, y, z Working range coordinates if a TCP movement is generated in addition to the orientation movement

phi<$\varphi$> theta<$\vartheta$>

Programming with coordinate names “phi” and “theta”. Programming possible absolutely and in degrees. Although $\vartheta$ has only been defined for the interval [0,180] (refer to Fig. on page 3–248), any desired values may be used for programming. Internally, the angles are converted to the defined interval, e.g. programming ”phi0 theta–10” will result in the internal angle values = 10 and =180.

**Example**: `G1 x10 y50 z30 phi90 theta90`
The TCP is moving towards the space position (10,50,30), the orientation vector assumes a position along the y direction. In this position, the orientation vector has its final value.

O(<$\rho_x$>,<$\rho_y$>,<$\rho_z$>)

Direct programming of the Cartesian components $\rho_x$, $\rho_y$, $\rho_z$ of the orientation vector. Components are automatically standardized to 1 within the NC. Programming is only possible absolutely.

**Example**: `G1 x10 y50 z30 O(0,1,0)`
This example is equivalent to the previous one. As a consequence of the automatic standardization, e.g. the specifications O(1,2,4), O(2,4,8) and O(0.5,1,2) are identical.

O(<φ>,<ϑ>)

Programming using polar angles φ, ϑ of the orientation vector. Programming possible absolutely and in degrees. The distinction between angle and vector programming is made by the number of parameters in function O(..). A difference to direct phi-theta programming is only to be observed for incremental programming

**Example**: `G1 x10 y50 z30 O(90,90)`
This example is equivalent to the previous one.

With the programming methods described above, the orientation vector turns by an **internal calculated axis of rotation** $\vec{u}$ which is standing perpendicularly on $\vec{\rho}_a$ and $\vec{\rho}_e$. This corresponds to a plane made up of its initial state $\vec{\rho}_a$ and its final state $\vec{\rho}_e$. The vector furthermore moves along the shortest path, i.e. it describes an angle $\beta$ of less than or equal to 180 degrees.

☞ **To be able to calculate an axis of rotation for the orientation internally, the start and end orientation of the orientation vector may not run parallel or anti-parallel.**

The rotation speed of the orientation vector depends on whether a TCP movement is taking place in addition to the orientation movement. Two situations are possible:

- TCP and orientation movement:
  The programmed feedrate refers exclusively to the TCP movement. The orientation movement ”follows” synchronously.
- Pure orientation movement:
  The programmed feedrate is the angle speed of the coordinate of rotation movement around $\vec{u}$. Movements of other axes without TCP component ”follow” synchronously.
  If OMEGA is programmed for feedrate programming in addition to F, the angle speed corresponds to the OMEGA value.

The following programming methods using ROTAX(..) may be used to generate more general movements by **programming the axis of rotation** $\vec{u}$.

ROTAX(<$u_x$>,<$u_y$>,<$u_z$>) O(<β>)

- ROTAX(..) defines the orientation of the axis of rotation around which the orientation vector rotates by the Cartesian components $u_{x,}$ $u_{y,}$ $u_{z.}$ Programming is only possible absolutely.
- O(..) defines angle β, around which the $\varphi_a$ vector rotates around the axis of rotation. Programming is possible incrementally and in degrees.
  β may assume any desired values, i.e. several rotations are also possible. The positive and negative angle β creates selected rotations with a different sense of rotation.

**Example**:

```
N.. G1 x10 y20 z30 O(1,0,0)
N.. ROTAX(1,0,1) O(90)
N.. O(180)
N.. O(-270)
```

Starting from the start orientation vector $\vec{\varphi}_a$ = (1,0,0) the orientation vector $\vec{\rho}$ describes a total rotation by 270° around the axis of rotation $\vec{u} = \left(1/\sqrt{2}, 0, 1/\sqrt{2}\right)$. In the case of a reversal, it rotates by 270° back to its original position.

ROTAX(<$\varphi_u$>,<$\vartheta_u$>) O(<β>)

- ROTAX(..) defines the orientation of the axis of rotation around which the orientation vector rotates by polar coordinates $\varphi_{u,}$ $\vartheta_u$. Programming is possible absolutely and in degrees.
- O(..) defines angle β, around which the $\varphi_a$ vector rotates around the axis of rotation. Programming is possible incrementally and in degrees.
  β may assume any desired values, i.e. several rotations are also possible. The positive and negative angle β creates selected rotations with a different sense of rotation.

**Example**: `ROTAX(0,45) O(720)`

**Please note for vector orientation:**

- The following intervals are applicable to the polar angles:
  $0° \leq \varphi < 360°$
  $0° \leq \vartheta \leq 180°$
  $\vartheta$ may be programmed using freely defined values (it is internally converted to definition interval [0,180]).

- The movement of the orientation vector is carried out as a rotation around a **programmed axis of rotation,** or if ROTAX(..) and O(..) have not been programmed, around an **internally calculated axis of rotation**.
- ROTAX(..) has to be programmed prior to or together with O(<β>), otherwise a runtime error will occur.
- With the exception of ROTAX programming, all the different methods of programming the vector orientation are absolute programming methods, i.e. they are independent of the changeover from G90/G91. Coordinate-specific incremental programming with the IC attribute will lead to a runtime error.
  **Example**: N10 phi=IC(30).

Syntax variants:

| **Function** | **Meaning** | **G90/G91 behavior** |
|---|---|---|
| phi<φ> theta<ϑ> | Polar coordinates of $\vec{\rho}$ | always absolute |
| O(<$\rho_x$>,<$\rho_y$>,<$\rho_z$>) | Cartesian components of $\vec{\rho}$ | always absolute |
| O(<φ>,<ϑ>) | Polar coordinates of $\vec{\rho}$ | always absolute |
| O(<β>) | Angle of coordinate of rotation with programmed axis of rotation $\vec{u}$ | always incremental |
| ROTAX(<$u_x$>,<$u_y$>,<$u_z$>) | Cartesian components of $\vec{u}$ | always absolute |
| ROTAX(<φ>,<ϑ>) | Polar coordinates of axis of rotation $\vec{u}$ | always absolute |

### 3.93.4 Tensor orientation for non-rotation symmetrical tools

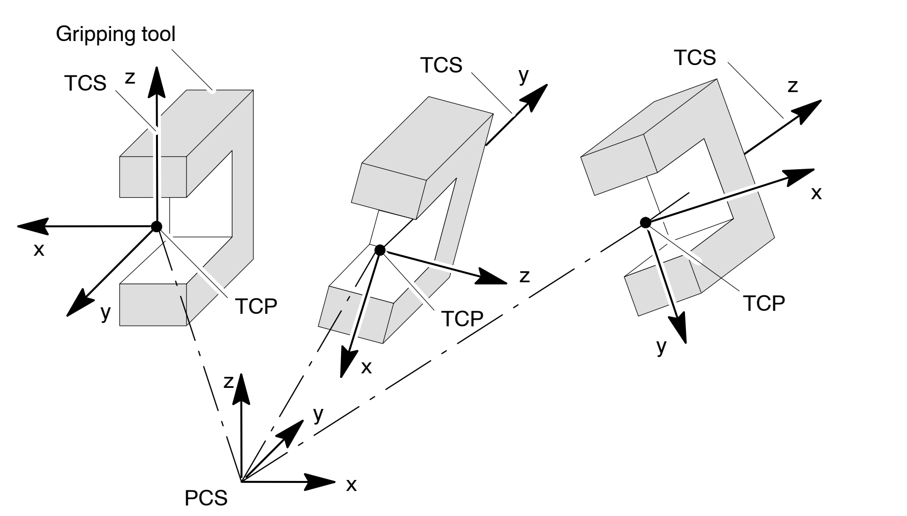
Effect

Tensor orientation relates to **non-rotation symmetrical** tools (e.g. gripping tools).
A tool coordinate system (TCS) which is **permanently** connected to a tool is oriented in space against a reference coordinate system (PCS, MCS,..).

- With a **3x3 orientation tensor** $\overset{\leftrightarrow}{O}$
  Rotation matrix which aligns the TCS into a different spatial position around the tool's TCP.

- With the **Eulerian angles $\varphi$ (phi), $\vartheta$ (theta) and $\psi$ (psi)**
  Three orientation coordinates are sufficient for a general orientation of the TCS. Three consecutive rotations by the Eulerian angles $\varphi$ (phi), $\vartheta$ (theta) and $\psi$ (psi) around the main coordinates of the PCS give the TCS its new orientation.

**Different orientations and reference positions of the TCS**



☞ **For details, please refer to “PNC description of functions” manual, Tool orientation section.**

Condition:

- For tensor orientation it is necessary to activate an axis transformation with orientation identification 3. This is done with the aid of function Coord(<i>).
  Afterwards, orientation coordinates $\varphi$ and $\vartheta$ can be programmed.
- All names of the programmed coordinates have been defined in MACODA parameter 7080 00010 [1..3] or [4..6]:
  [1] x
  [2] y
  [3] z
  [4] phi
  [5] theta
  [6] psi

## Programming

The **TCS orientation** can be programmed using one of the following five alternatives:

- N..{NC-Fkt} {x.. y.. z..} phi<$\varphi$> theta<$\vartheta$> psi<$\psi$>
- N..{NC fct} {x.. y.. z..} Ox(<$o_{11}$>,<$o_{21}$>,<$o_{31}$>) Oy(<$o_{12}$>,<$o_{22}$>,<$o_{32}$>)
  or
  N..{NC fct} {x.. y.. z..} Ox(<$o_{11}$>,<$o_{21}$>,<$o_{31}$>) Oz(<$o_{13}$>,<$o_{23}$>,<$o_{33}$>)
  or
  N..{NC fct} {x.. y.. z..} Oy(<$o_{12}$>,<$o_{22}$>,<$o_{32}$>) Oz(<$o_{13}$>,<$o_{23}$>,<$o_{33}$>)
- N.. {NC fct} {x.. y.. z..} Ox(<$\varphi_x$>,<$\vartheta_x$>) Oy(<$\varphi_y$>,<$\vartheta_y$>)
  or
  N.. {NC fct} {x.. y.. z..} Ox(<$\varphi_x$>,<$\vartheta_x$>) Oz(<$\varphi_z$>,<$\vartheta_z$>)
  or
  N.. {NC fct} {x.. y.. z..} Oy(<$\varphi_y$>,<$\vartheta_y$>) Oz(<$\varphi_z$>,<$\vartheta_z$>)
- N.. {NC function} {x.. y.. z..} ROTAX(<$u_x$>,<$u_y$>,<$u_z$>) O(<$\beta$>)
- N.. {NC function} {x.. y.. z..} ROTAX(<$\varphi_u$>,<$\vartheta_u$>) O(<$\beta$>)

where

NC fct — Function which expects the working range coordinates as local parameters (refer to table on page 3–260)

x, y, z — Working range coordinates if a TCP movement is performed in addition to the orientation movement

phi<$\varphi$> theta<$\vartheta$> psi<$\psi$>

Orientation of the TCS using Eulerian angles $\varphi$, $\vartheta$, $\psi$. Programming is possible absolutely/incrementally and in degrees. Any values may be used for the Eulerian angles, which are converted by the NC internally to their respective defined interval.

**Example**: `G1 x10 y50 z30 phi90 theta90 psi45`

Ox(<$o_{11}$>,<$o_{21}$>,<$o_{31}$>)
Oy(<$o_{12}$>,<$o_{22}$>,<$o_{32}$>)
Oz(<$o_{13}$>,<$o_{23}$>,<$o_{33}$>)

or

Ox(<$\varphi_x$>,<$\vartheta_x$>)
Oy(<$\varphi_y$>,<$\vartheta_y$>)
Oz(<$\varphi_z$>,<$\vartheta_z$>)

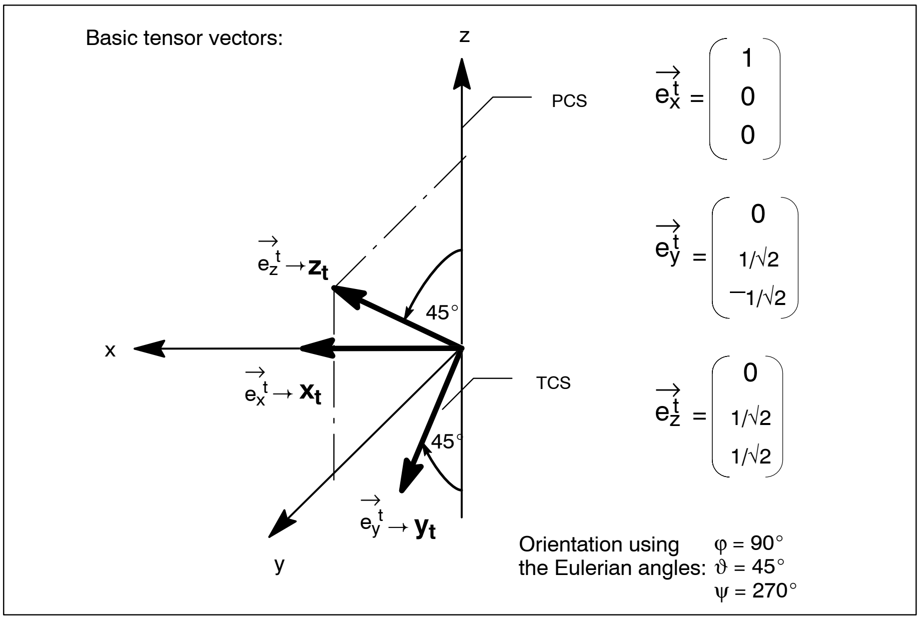
Ox(..) defines the direction of the x coordinate of the TCS in the reference coordinate system.
The direction may be specified in Cartesian vector components Ox(<$o_{11}$>,<$o_{21}$>,<$o_{31}$>) or in polar coordinates Ox(<$\varphi_x$>,<$\vartheta_x$>).

The definition applies analogously to the column vectors Oy(..) and Oz(..).
Programming is only possible absolutely, standardization to 1 is done internally by the NC. It is only permitted to program 2 of the three TCS coordinates. They need not stand perpendicular on each other because one of them is corrected internally to 90 degrees:

| Programmed combination | Corrected column vector |
|---|---|
| Ox(..) Oy(..) | Oy(..) |
| Ox(..) Oz(..) | Ox(..) |
| Oy(..) Oz(..) | Oz(..) |

**Example**:

```
G1 x10 y20 z30 Ox(1,0,0) Oy(0,0.707,-0.707) or
G1 x10 y20 z30 Ox(1,0,0) Oz(0,0.707,0.707) or
G1 x10 y20 z30 Oy(0,0.707,-0.707) Oz(0,0.707,0.707)
```


Basic tensor vectors:
z
PCS
x
y
TCS
$\vec{e}_z^{\,t} \rightarrow z_t$
$\vec{e}_x^{\,t} \rightarrow x_t$
$\vec{e}_y^{\,t} \rightarrow y_t$
45°
45°
$\vec{e}_x^{\,t} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$
$\vec{e}_y^{\,t} = \begin{pmatrix} 0 \\ 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$
$\vec{e}_z^{\,t} = \begin{pmatrix} 0 \\ 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$
Orientation using the Eulerian angles:
$\varphi = 90°$
$\vartheta = 45°$
$\psi = 270°$

With the programming methods described above, the start orientation tensor $\overleftrightarrow{O}_a$ rotates around an **internal calculated axis of rotation** $\vec{u}$ to the end orientation tensor $\overleftrightarrow{O}_e$. The internally calculated angle $\beta$ is less than or equal to 180 degrees. The orientation movement is always carried out on the shortest path. It is independent of the special axis kinematics.

The rotation speed of the orientation tensor depends on whether or not a TCP movement is taking place in addition to the orientation movement. Two situations are possible:

- TCP and orientation movement:
  The programmed feedrate refers exclusively to the TCP movement. The orientation movement ”follows” synchronously.
- Pure orientation movement:
  The programmed feedrate is the angle speed of the coordinate of rotation movement around $\vec{u}$. Movements of other axes without TCP component ”follow” synchronously.
  If OMEGA is programmed for feedrate programming in addition to F, the angle speed corresponds to the OMEGA value.

The following programming methods using ROTAX(..) may be used to generate more general movements by **programming the axis of rotation $\vec{u}$**:

ROTAX(<$u_x$>,<$u_y$>,<$u_z$>) O(<β>)
or
ROTAX(<$\varphi_u$>,<$\vartheta_u$>) O(<β>): ROTAX(..)

- ROTAX(..) defines the orientation of the axis of rotation around which theorientation vector rotates by the Cartesian components $u_{x,}$ $u_{y,}$ $u_z$ or the polar coordinates $\varphi_u$, $\vartheta_{u.}$ Programming possible absolutely and in degrees.
- O(..) defines angle β, by which the $O_a$ tensor rotates around the axis of rotation. Programming is possible incrementally and in degrees.
  β may assume any desired values, i.e. several rotations are also possible. The positive and negative angle β creates selected rotations with a different sense of rotation.

**Example**:

```
ROTAX(0,45) O(720)
ROTAX(1,0,1) O(720)
```

**Please note for tensor orientation:**

- The following intervals are applicable to the Eulerian angles
  $0° \le \varphi < 360°$
  $0° \le \vartheta \le 180°$
  $0° \le \psi \le 360°$.

  The orientation tensor can be clearly created with the Eulerian angles, however with the restriction that in case of $\vartheta = 0°$, the sum $\varphi+\psi$ and in case of $\vartheta = 180°$ the difference $\varphi-\psi$ is necessary to define an orientation.

  $\vartheta$ may be programmed with freely defined values, even though an interval [0,180] has been defined. Internally, the angles are converted to the definition interval.
- If programming of the tensor columns $\vec{e}_x$, $\vec{e}_y$, $\vec{e}_z$ results in two column vectors being parallel or anti-parallel, it is not possible to calculate the orientation tensor. A runtime error is then triggered.
- ROTAX(..) has to be programmed prior to O(<β>), otherwise a runtime error will occur.

Syntax variants:

| **Function** | **Meaning** | **G90/G91 behavior** |
|---|---|---|
| phi<$\varphi$> theta<$\vartheta$> psi<$\psi$> | Eulerian angles of the orientation | absolute/incremental AC/IC progr. possible |
| Ox(<$o_{11}$>,<$o_{21}$>,<$o_{31}$>)<br>Oy(<$o_{12}$>,<$o_{22}$>,<$o_{32}$>)<br>Oz(<$o_{13}$>,<$o_{23}$>,<$o_{33}$>) | Vector $\vec{e}_x^{\,t}$ in the PCS (1st O column)<br>Vector $\vec{e}_y^{\,t}$ in the PCS (2nd O column)<br>Vector $\vec{e}_z^{\,t}$ in the PCS (3rd O column) | always absolute |
| Ox(<$\varphi_x$>,<$\vartheta_x$>)<br>Oy(<$\varphi_y$>,<$\vartheta_y$>)<br>Oz(<$\varphi_z$>,<$\vartheta_z$>) | $\vec{e}_x^{\,t}$ in polar coordinates of the PCS<br>$\vec{e}_y^{\,t}$ in polar coordinates of the PCS<br>$\vec{e}_z^{\,t}$ in polar coordinates of the PCS | always absolute |
| O(<$\beta$>) | Angle of coordinate of rotation with programmed axis of rotation $\vec{u}$ | always incremental |
| ROTAX(<ux>,<uy>,<uz>) | Cartesian components of $\vec{u}$ | always absolute |
| ROTAX(<$\varphi_u$>,<$\vartheta_u$>) | Polar coordinates of axis of rotation $\vec{u}$ | always absolute |

## 3.94 Working range coordinate programming and axis transformations COORD(..)

Effect

In contrast to axis coordinate programming, working range coordinate programming is totally independent of the axis configuration of the machine and of the tool compensation of the tools used.

Working range coordinate programming is used to:

- program the position of the tool tip (TCP)
- program the tool orientation
- superimpose the positioning of the tool tip with an orientation movement of the tool.

As a precondition of working range coordinate programming, an “axis transformation” previously defined with COORD(..) has to be activated which determines the axis setpoint values of all necessary physical axes at the machine from the programmed working range coordinates.

Different types of axis transformation are stored in MACODA under separate numbers, by which they can be called up in the program. Furthermore, it is possible to realize customer-specific, freely defined transformations which require a maximum of six working range coordinates.

Programming

**COORD(<i>)** Working range coordinate programming with i-th axis transformation ON

where

| | | |
|---|---|---|
| <i> | 1..10: | refers to one of the ten axis transformation blocks in MACODA |

- If the activated axis transformation supports the vector or tensor orientation, the corresponding orientation NC function is activated.
- The programming of coordinate names is activated. Depending on the type of axis transformation, a certain subset of the maximum of six coordinates (x,y,z,phi,theta,psi) is then programmable.
- The axis positions are converted to coordinate values, corresponding to the forward transformation equations (the workpiece coordinate display jumps from axis positions to coordinate values).

☞ **Any programming has to be performed in “working range coordinates” with activated working range coordinate programming. It is not permitted to use transformed axes.**

Programming **COORD(0):** Working range coordinate programming OFF

- If an orientation NC function is active, it is deactivated.
- The programming of coordinate names is deactivated. The axis names can then be used again without restrictions.
- The coordinate values are converted to axis positions, corresponding to the backward transformation equations (the workpiece coordinate display jumps from coordinate values to axis positions).

☞ **Switching over between different axis transformations is possible without prior deactivation.**

## NC functions on the basis of working range coordinates

In addition to position programming in the part program, there is a series of functions which expect working range coordinates as local parameters in the part program.

The list below shows all functions expecting working range coordinates as parameters or referring to working range coordinates when working range coordinate programming (Coord(<i>)) is active.

| NC functions | Description | Effect in the program |
|---|---|---|
| G00, G01, G02, G03, G05, G10, G11, G12, G13, G32, G73, G200, G202, G203 | NC functions that generate a movement | The positions are programmed as working range coordinates. |
| G17, G18, G19, G20 | Plane selection and pole programming | The working range coordinates intended to define the machining plane are selected. In case of G20, the pole coordinates for polar coordinate programming are programmed at the same time. |
| G34, G134, G234 | Chamfers and transition arcs | The transition segments are calculated as working range coordinates. |
| G138, G352, G354, ..., G359 | Determination of the workpiece coordinates | Define the position of the WCS in relation to the BCS. The parameters of G352 are coordinates. |
| G37, G38, G60, G168, G268 | Determination of the program coordinates | Define the position of the PCS in relation to the WCS. The programmed working range coordinates always refer to the current PCS. |
| G40, G41 | Path compensation | Path compensation is performed within the selected machining plane. The machining plane is defined by 2 working range coordinates. |
| G78, G145, ..., G845, G147, ..., G847, H | Tool compensation | Tool compensations are normally accounted for by the PCS. By changing over the compensation, it may also be performed in the TCS (axis transformation). |
| G90, G91, G189, AC, IC | Type of programming | States whether the working range coordinates should be interpreted absolutely or incrementally. |

G Instructions COORD(..)

| NC functions | Description | Effect in the program |
|---|---|---|
| G92 | Program zero point | Defines the program coordinate zero point within the current PCS. |
| G75 | Special functions | The traversing movement is programmed in working range coordinates. Measurement is performed for the axes configured on the channel and released in MACODA. All of the measuring values are axis positions. The conversion from axis measuring values to coordinate values in connection with CPL access functions is being prepared. |
| G175, G275 | | The traversing movement is programmed in working range coordinates. At the same time, a physical axis is specified via its index for which a measurement is to be carried out. The measuring value provides an axis position for this physical axis. The conversion from axis measuring values to coordinate values in connection with CPL access functions is being prepared. |
| G105 | | The traversing movement is programmed in working range coordinates. The linear modulo axis must not be a member of the axis transformation (pseudo coordinate). |
| G301 | | The traversing movement is programmed in working range coordinates. The oscillating axis must not be a member of the axis transformation (pseudo coordinate). |

## NC functions on the basis of axes

With activated working range coordinate programming, the following functions will **continue** to be programmed **with axis coordinates**:

| NC functions | Description | Effect in the program |
|---|---|---|
| G06, G14, G608, G114, G177, G594, G595 | Functions to influence the axis dynamics | |
| G21 | Axis classification | The programmed axis classification has an effect at the earliest when the working range coordinate programming has been deactivated. |
| G54–G59, G154–G159, G254–G259, G160, G260, G360 | (Axis) zero shifts | The existing ZS are axis shift values. Zero shifts at the level of coordinates are realized by the respective coordinate functions – e.g. inclined plane. The ZS are taken into account in the interpolator behind the axis transformation. When an axis or coordinate transformation is activated or switched over, it is no longer necessary to switch off the ZS. |
| G374, G520, ..., G524 | Functions causing a movement | Drive-controlled movements are generated. |
| G151, ACP, ACN, DC | Type of programming | The positioning type for an endless axis is an axis property. |
| G510, ..., G513, G515, G516 | Axis transfer | Axes are transferred, not coordinates. |
| G581 | Axis coupling | Two axes are coupled to each other. |
| G131, G631 | Tool guidance | The tool axis is a parameter of these functions. Tool guidance with one working range coordinate is not possible. |
| G900 | SERCOS parameter | Has a direct effect on the SERCOS drive. |
| G610, G611, G612 | Punching | The stroke release times are axis properties. |
| AREADEF, AREAVALID | Control area | At present a function which mixes coordinates and axes. For this reason, it must not be used together with an axis transformation or coordinate transformation (e.g. ”inclined plane”).<br>The function needs to be extended to machine protection areas (axis function) and PCS protection areas (coordinate function). |
| G74, G76 | Special functions | Axis values or axis positions are programmed. The axis position is transformed into working range coordinates internally. The interpolation is performed in working range coordinates. |

## 3.94.1 5 axis transformation

Effect

The 5 axis transformation realized in the PNC includes:

- 3 linear coordinates (e.g. x, y, z)
- 2 orientation coordinates ϑ, φ (e.g. theta, phi)
- 3 linear axes (e.g. X, Y, Z)
- 2 rotary axes (e.g. B and C)

There are 3 types of 5 axis transformation:

- Linear orientation with axis programming (Type 3032101)
- Linear orientation with coordinate programming (Type 3232101)
- Vector orientation

Special properties

- Feed rate: having activated the 5 axis transformation, there is a switchover to working range coordinate programming. The programmed feedrate (F) refers to the programmable position coordinates only, i.e. the F word is used to program the path speed of the tool center point (TCP).
  Additional orientation and pseudo-coordinate movements do not change this path speed.
  The orientation and pseudo coordinate movement is guided along synchronously, i.e. the end position is reached simultaneously for all coordinates. The movement of the orientation and pseudo-coordinates carried along may, however, lead to an additional limitation of the path kinematics (maximum path speed and acceleration) because the limit values of all axes involved in the movement are monitored.
- Rotary axes B and C may be endless axes as well as rotary axes.

☞ **For details, please refer to “PNC description of functions” manual.**

Programming

### 5 axis transformation type 3032101

Each 5 axis transformation of the type 3032101 is programmed as follows:

1. Switching on the working range coordinate programming using Coord(<i>) (i-th transformation initialized in MACODA).
2. Type 3032101 supports **linear orientation with axis programming** (refer to page 3–245).
3. Programming the program coordinates in working range coordinates using the NC functions listed in the table on page 3–260.
4. Out of maximally 6 working range coordinates, a maximum of 3 linear position coordinates (x,y,z) is permitted with this type. Axis coordinates as an alternative to coordinate names are no longer permitted.In case of this type, orientations are performed as linear interpolation by the rotary axes names (B,C).
5. Switching off the selected axis transformation using COORD(0) or by selecting a different axis transformation.

All names can be set in MACODA.

☞ **With active axis transformation, the axis positions are converted to coordinate values. The display of the workpiece coordinates changes from axis coordinate values to working range coordinate values.**

**Example**:

| | |
|---|---|
| `N10 G1 X0 Y0 Z0 B0 C0` | Programming of the logical or physical axis names |
| `N20` **`COORD(1)`** | The configuration of the 5 axis transformation type 3032101 is located in the MACODA parameter block 1:<br>Linear coordinates: **x,y,z**<br>Orientation coordinates: **B,C** |
| `N30` **`x100 y200 z300 B20 C60`**<br>`..` | Linear coordinate interpolation with additional orientation movement |
| `N40` **`G2 x.. y.. z.. I.. J..`**<br>**`B70 C80`**<br>`..` | Helical movement (x,y,z) of the TCP with additional orientation movement. |
| `N50` **`G1 B20 C10`**<br>`..` | Pure orientation movement. The TCP remains constant. |
| `N60` **`COORD(0)`**<br>`..` | Switching off the 5 axis transformation. |

Programming

## 5 axis transformation type 3232101

Each 5 axis transformation of the type 3232101 is programmed as follows:

1. Switching on the working range coordinate programming using COORD(<i>) (i<th  transformation initialized in MACODA)
2. Type 3232101 supports **linear orientation with coordinate programming** (refer to page 3–246).
3. Programming the program coordinates in **working range coordinates** using the NC functions listed in the table on page 3–260. Out of maximally 6 working range coordinates, a maximum of 3 linear position coordinates (x,y,z) 2 rotary orientation coordinates (φ, ϑ) is permitted for programming with this type. Axis coordinates as an alternative to coordinate names are no longer permitted.
4. The syntax ROTAX(..) O(..) is not possible.
5. Switching off the selected axis transformation using COORD(0) or by selecting a different axis transformation.

All names can be set in MACODA.

☞ **For a given axis kinematics it is recommended to configure all three types of axis transformation (3032101, 3232201, and 3232101) in MACODA. Afterwards, you can change over between the different orientation movements in the NC program by programming COORD(1), COORD(2), and COORD(3).**

**Example**:

| | |
|---|---|
| `N10 G1 X0 Y0 Z0 B0 C0` | Programming of the logical or physical axis names |
| `N20 COORD(2)` | The configuration of the 5 axis transformation type 3232101 is located in the MACODA parameter block 2:<br>Linear coordinates: **x,y,z**<br>Orientation coordinates: **phi, theta** |
| `N30 x100 y200 z300 phi5 theta5`<br>`..` | Linear coordinate interpolation in angles φ (phi) and ϑ (theta). |
| `N40 G2 x.. y.. z.. I.. J.. phi20 theta60`<br>`..` | Helical movement (x,y,z) of the TCP with additional linear orientation of the orientation vector. |
| `N50 G1 phi0 theta45` | Pure linear orientation movement.<br>The TCP remains constant. |
| `N60 COORD(0)`<br>`..` | Switching off the 5 axis transformation. |

Programming

### 5 axis transformation type 3232201

Each 5 axis transformation of the type 3232201 is programmed as follows:

1. Switching on the working range coordinate programming using Coord(<i>) (i-th transformation initialized in MACODA).
2. Type 3232201 supports **vector orientation** (refer to page 3–248).
3. Programming the program coordinates in working range coordinates using the NC functions listed in the table on page 3–260.
   Out of maximally 6 working range coordinates, a maximum of 3 linear position coordinates (x,y,z) 2 rotary orientation coordinates ($\varphi$, $\vartheta$) is permitted for programming with this type. Axis coordinates as an alternative to coordinate names are no longer permitted.
4. Switching off the selected axis transformation using COORD(0) or by selecting a different axis transformation.

All names can be set in MACODA.

☞ **With active axis transformation, the axis positions are converted to coordinate values. The display of the workpiece coordinates changes from axis coordinate values to working range coordinate values.**

**Example**:

| | |
|---|---|
| `N10 G1 X0 Y0 Z0 B0 C0` | Programming of the logical or physical axis names |
| `N20` **`COORD(3)`** | The configuration of the 5 axis transformation type 3232201 is located in the MACODA parameter block 3:<br>Linear coordinates: **x,y,z**<br>Orientation coordinates: **phi, theta** |
| `N30` **`x100 y200 z300 phi5`**<br>**`theta5`**<br>`..` | Linear coordinate interpolation with additional coordinate of rotation movement of the orientation vector. |
| `N40` **`G2 x.. y.. z.. I.. J..`**<br>**`phi20 theta60`**<br>`..` | Helical movement (x,y,z) of the TCP with additional coordinate of rotation movement of the orientation vector. |
| `N50` **`G1 phi0 theta45`**<br>`..` | Pure vector orientation movement.<br>The TCP remains constant. |
| `N60` **`COORD(0)`**<br>`..` | Switching off the 5 axis transformation. |

## 3.94.2 6 axis transformation

Effect

The 6 axis transformation realized in the PNC includes:

- 3 linear coordinates (e.g. x, y, z)
- 3 orientation coordinates ϑ, φ, ψ (e.g. theta, phi, psi),
- 3 linear axes (e.g. X, Y and Z)
- 3 rotary axes (e.g. A, B and C)

There are 2 types of 6 axis transformation:

- **Type 3333301** allows TCP programming via three linear coordinates and the tool orientation (TCS) by programming the Eulerian angles φ (phi), ϑ (theta) and ψ (psi).
  The orientation movement is performed as TCS rotation around an axis of rotation fixed in space.

- **Type 3033101** supports TCP programming via three linear coordinates and tool orientation by programming the three rotary axes.
  The orientation movement is executed linearly in the rotary axis positions.

☞ **For details, please refer to “PNC description of functions” manual.**

Programming

**6 axis transformation type 3033101**

Each 6 axis transformation of the type 3033101 is programmed as follows:

1. Switching on the working range coordinate programming using COORD(<i>) (i<th transformation initialized in MACODA)
2. Type 3033101 does **not** support tensor orientation.
3. Programming the program coordinates in **working range coordinates** using the NC functions listed in the table on page 3–260.
   Out of maximally 6 working range coordinates, a maximum of 3 linear position coordinates (x,y,z) is permitted with this type. Axis coordinates as an alternative to coordinate names are no longer permitted.
   In case of this type, tool axis orientations are performed as linear interpolation of the rotary axes (A,B,C).
4. Switching off the selected axis transformation using COORD(0) or by selecting a different axis transformation.

All names can be set in MACODA.

☞ **With active axis transformation, the axis positions are converted to coordinate values. The display of the workpiece coordinates changes from axis coordinate values to working range coordinate values.**

**Example**:

| | |
|---|---|
| `N10 G1 X0 Y0 Z0 B0 C0` | Programming of the logical or physical axis names |
| `N20` **`COORD(1)`** | The configuration of the 6 axis transformation type 3033101 is located in the MACODA parameter block 1:<br>Linear coordinates: **x,y,z**<br>Orientation coordinates: **A,B,C** |
| `N30` **`x100 y200 z300 A10 B20 C60`** | Linear coordinate interpolation with additional orientation movement |
| `..` | |
| `N40` **`G2 x.. y.. z.. I.. J.. A30 B70 C80`** | Helical movement (x,y,z) of the TCP with additional orientation movement. |
| `..` | |
| `N50` **`G1 A45 B20 C10`** | Pure orientation movement. The TCP remains constant. |
| `..` | |
| `N60` **`COORD(0)`** | Switching off the 6 axis transformation. |
| `..` | |

Programming

### 6 axis transformation type 3333301

Each 6 axis transformation of the type 3333301 is programmed as follows:

1. Switching on the working range coordinate programming using COORD(<i>) (i<th  transformation initialized in MACODA)
2. Type 3333301 supports **tensor orientation** (refer to page 3–254).
3. Programming the program coordinates in working range coordinates using the NC functions listed in the table on page 3–260.
   Out of maximally 6 working range coordinates, a maximum of 3 linear position coordinates (x,y,z) 3 rotary orientation coordinates ($\varphi$, $\vartheta$, $\psi$) is permitted for programming with this type. Axis coordinates as an alternative to coordinate names are no longer permitted.
4. Switching off the selected axis transformation using COORD(0) or by selecting a different axis transformation.

Orientation movements are generated by programming the Eulerian angles $\varphi$, $\vartheta$, $\psi$ or the related alternative syntax options.

All names can be set in MACODA.

☞ **With active axis transformation, the axis positions are converted to coordinate values. The display of the workpiece coordinates changes from axis coordinate values to working range coordinate values.**

**Example**:

| | |
|---|---|
| `N10 G1 X0 Y0 Z0 B0 C0` | Programming of the logical or physical axis names |
| `N20 COORD(2)` | The configuration of the 6 axis transformation type 3333301 is stored in MACODA parameter block 2. |
| `N40 G2 x.. y.. z.. I.. J..`<br>`    phi20 theta60 psi230`<br>`..` | Helical movement (x,y,z) of the TCP with additional coordinate of rotation movement of the orientation tensor. |
| `N50 G1 phi0 theta45 psi90`<br>`..` | Pure tensor orientation movement. The TCP remains constant. |
| `N60 COORD(0)`<br>`..` | Switching off the 6 axis transformation. |

☞ **For a given axis kinematics it is recommended to configure both types of axis transformation (3033101 and 3333301) in MACODA. Afterwards, you can change over between linear and tensor orientation in the NC program by programming COORD(1) and COORD(2).**

## 3.95 Axis distance control for digitizing DistCtrl

Effect

The ”Axis distance control for digitizing” function keeps the distance between the surface scanned and the measuring device (e.g. laser) constant during digitizing. This is to ensure that the available working range of the measuring device is not exceeded.
For a detailed description of the function, please refer to the ”PNC description of functions” manual.

Programming

**DistCtrlOn** Starts axis distance control. Furthermore, the current distance between the measuring device and the surface is taken over as reference value.
Programming ”DistCtrlOn” by itself will activate the configuration data for axis distance control defined in MACODA.

As an option, various additional instructions may be programmed to override some of the configuration data from MACODA.

☞ **MACODA configuration data that has been overriden will not be active again before program deselection, channel or system reset.**

**DCAXIS(<axis>,<corr>)**
Overrides MP 7050 00702.

<axis> Name or number of the channel axis for which axis distance control is to be activated.

<corr> +1 or 1: Account for correction values in positive direction of movement
–1: Account for correction in negative direction of movement

**DCFILTER(<time>)**
Overrides MP 7050 00730.

<time> 0: Filter off
>0: Filter on, values in ms

**DCLIMIT([<speed>],[<accel>])**
Overrides MP 7050 00740 and 7050 00741.

<speed> Overrides MP 7050 00740.
Value input, depending on active unit of measurement (G71, G70), in the unit mm/min or inch/min.

<accel> Overrides MP 7050 00741.
Value input, depending on active unit of measurement (G71, G70), in the unit m/s$^2$ or 1000 inch/s$^2$.

**DCMON([<collision>],[<hole>])**
Overrides MP 7050 00750 and 7050 00752.

<collision> Overrides MP 7050 00740.
Value input, depending on active unit of measurement (G71, G70), in the unit mm/min or inch/min.

<hole> Overrides MP 7050 00741.
Value input, depending on active unit of measurement (G71, G70), in the unit m/s$^2$ or 1000 inch/s$^2$.

**DistCtrlBreak** Interrupts axis distance control. The current correction value remains active.

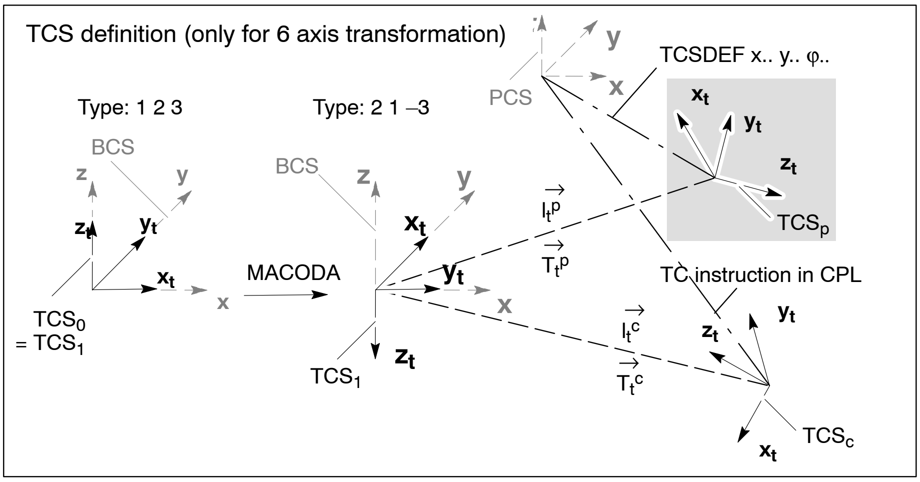
**DistCtrlContinue** Resumes axis distance control after an interruption. The NC controls the deviation from the reference value as fast as possible.

**DistCtrlOff** Deactivates axis distance control, stores the current correction value, and stops axis movement.
If DistCtrlOff is programmed in the same block as a traversing movement, the NC will not turn off axis distance control before the movement has been completed.

## 3.96 TCS definition in program coordinates **TCSDEF**

Effect

Function ”TCS definition in program coordinates ” generates a tool coordinate system **TCS$_p$** that may be displaced and/or rotated in relation to a current TCS$_c$ or TCS$_1$.
The coordinate values specified in the TCSDEF instruction for TCS$_p$ will be converted to quantities $\vec{I}_t^p$ and $\vec{T}_t^p$ by the NC and stored in the tool correction memory.
Switching the function off with TCSUNDEF will reactivate the TCS$_c$ or – if no explicit tool compensation is active – the TCS$_1$ (refer to figure below).



Programming

Defining the position of the tool coordinate system TCS$_p$:

**TCSDEF**[<Linear coordinates>][<Orientation coordinates>]
where

| | |
|---|---|
| <Linear coordinates> | Coordinates that refer to the current PCS |
| <Orientation coordinates> | Coordinates that refer to the current PCS or all alternative syntax options of tensor orientation (refer to following examples) |

The following programming methods are possible:
(coordinate names x, y, z, phi, theta and psi declared in MACODA)

| | |
|---|---|
| TCSDEF x.. y.. z.. phi.. theta.. psi.. | Orientation of TCS$_p$ in Eulerian angles |
| TCSDEF x.. y.. z.. O(<φ >,<ϑ >,<ψ >) | Orientation of TCS$_p$ in Eulerian angles |
| TCSDEF x.. y.. z.. Ox(..) Oy(..) Oz(..) | Orientation of TCS$_p$ as a tensor |
| TCSDEF x.. y.. z.. ROTAX(...) O(<β>) | Rotating the TCS1 to the new TCS$_p$ |

**Please note for TCSDEF:**

- TCSDEF may only be used in connection with an active 6 axis transformation.
- The values programmed by coordinate names x, y, z, phi, theta and psi are always interpreted as absolute values by the PCS (are **not** subject to G90/G91 changeover).
  However, individual programming of IC() and AC() is supported.
- The following applies to the alternative syntax options for orientation programming:
  The values programmed by O(<φ >,<ϑ >,<ψ >), Ox(..), Oy(..) and Oz(..) are absolute values within the PCS.
- Rotary axis programming ”ROTAX(...) O(<β>)” is an incremental rotation of $TCS_1$ around the angle β.
- Programming TCSDEF without parameters has no effect.

Programming

Resetting active tool coordinate system $TCS_p$:

**TCSUNDEF** Reset to previously active tool coordinate system (e.g. $TCS_c$ or if no tool compensation had been active: $TCS_1$)

**Please note for TCSUNDEF:**

- An automatic reset takes place with coordinate changeover ”N.. COORD(<i>)”.

## 3.97 Path velocity-dependent laser power control LFPON LFPOFF

Effect

This function controls the power of a laser depending on the actual feedrate value ($V_{path}$). For this purpose, a suitable voltage value is output to an analog output in the way defined in MP 4075 00104.

The rms path velocity $V_{path}$ is calculated from the velocities of the selected coordinates:

- by selecting the active plane (APL) or the active space (ASP). An active axis transformation or coordinate transformation (inclined plane), if any, is also accounted for.
- by a direct selection of coordinates in the part program:
  - No active axis transformation:
    All pseudo-coordinates (axes) of a channel may be selected.
  - An axis transformation is active:
    Working range or pseudo-coordinates may be selected. The selected working range coordinates are bound to the transformation active at that moment.

**Restrictions**:

- The available analog outputs limit the number of channels that may use this function.
- When using the ”inclined plane” function, only the ”active working range” can be used for coordinate selection with PL(ASP) in order to generate the velocity $V_{path}$.
- The function is supported with axes and coordinates if 5 axis transformation is active.
- In the event of an error (runtime error, diagnostics class 1 error), when ”drive under control” of a drive involved in the path (no enable signal, drive off) is canceled, and in the event of “Feed Hold”, no laser voltage will be output.

Programming

LFPON **Starts** the path velocity-dependent laser power control. Programming this function by itself will activate the configuration data defined in MACODA. Additional parameters may be programmed as an option.

LFP **Sets the parameters for** the active laser power control in the part program.

Parameters for LFPON and LFP:

LL([<%voltage>], [<VMin>])
Lower power limit: The voltage value entered in this parameter will be output below the specified path velocity.

LL([<%voltage>], [<VMax>])
Upper power limit: The voltage value entered in this parameter will be output above the specified path velocity.

| | |
|---|---|
| <%voltage> | 0% .. 100%: corresponds to 0 .. 10 volts |
| <VMin> | Lower key value of path velocity in mm/min or inch/min |
| <VMax> | Upper key value of path velocity in mm/min or inch/min |

PL(<plane name>)
Coordinate selection for calculating the path velocity through a plane selection

| | | |
|---|---|---|
| <Plane name> | “APL”: | Current plane (G17, G18, G20) |
| | “ASP”: | Current space |
| | “MCD”: | MACODA values |

CD(<Coordinate 1>, [<Coordinate 2>], ... , [<Coordinate n>])
Coordinate selection for calculating the path velocity directly using the logic name

| | |
|---|---|
| <Coordinate x> | x = 1..n<br>Logic names of the working range coordinates or pseudo-coordinates (axes) involved |

LFPOFF **Terminates** the path velocity-dependent laser power control.

LPCOFF alternatively to LFPOFF

Please note:

- Functions LFPON, LPCOFF (LFPOFF) act modally.
- When the control unit is booted, the laser power control is deactivated, and the settings from the MACODA parameters are active.
- Data stored in MACODA parameter 7060 00010, ”Default state upon booting”, and MACODA parameter 7060 00020, ”Default state upon a control reset” will supersede this presetting.
- After control reset and M2/M30, the laser power control is deactivated, and the settings from the MACODA parameters are reactivated.

**Examples**:

| | |
|---|---|
| `LFPON LL(10,100)` | Activation of laser power control with the default value for the lower voltage limit of 10% (=1 V) at 100 mm/min |
| `LFP LL(10,100)` | Programming the lower voltage limit (10% (=1 V) at 100 mm/min) in the NC program |
| `LFP UL(90,500)` | Programming the upper voltage limit (90% (=9 V) at 500 mm/min) in the NC program |

The coordinate selection is determined by MP 7050 00820
(=2, corresponds to the “active plane”).

## 3.98 Online correction in workpiece coordinates HWOCON HWOCOFF

Effect

Programming the “Online correction” will mean that while the part program is active or inactive, an online

- correction of the **position** or **orientation** is carried out in the workpiece coordinate system (WCS) using the handwheel.
- traversing movement of the position of the **longitudinal tool axis** in TCS Z direction of the tool will take place in the tool coordinate system (TCS) (no tool compensation!).

Online correction is controlled:

- directly by the PLC via NC block input
- via machine functions, or
- by a part program.
  Within a part program, the online correction may be controlled for its own or for another channel.

☞ **For details, please refer to “PNC description of functions” manual.**

Programming

HWOCON OCONCH<Channel no.> OCCOORD<Coordinate no..> {OCSTEP<Increments>}

**Activates** the online correction via the PLC or machine functions or from any desired channel.

HWOCON OCCOORD<Coordinate no..> {OCSTEP<Increments>}

**Activates** the online correction in **the own** channel.

where

| | |
|---|---|
| OCONCH<Channel no.> | Number of the channel for which online correction is activated. |
| OCCOORD<Coordinate no.> | 1..8: Number of the coordinate<br>9: TCS Z direction (only with active 5 axis transformation, the correction in TCS Z direction is converted into a movement of the linear working range coordinates (x,y,z)). |
| {OCSTEP<Increments>} | **Optional**: Step size of a handwheel increment (from axis interface).<br>If not specified, the step size will be taken over from the axis interface (I 1.0...1.3, ”Manual feed/incremental step”) defined in MP 7050 00926. |

Programming

HWOCOFF OCOFFCH<Channel no.>

Deactivates the online correction via the PLC or machine functions or from any desired channel.

HWOCOFF

**Deactivates** the online correction in **the own** channel.

where

OCOFFCH<Channel no.>

Number of the channel for which online correction is deactivated.

Programming

HWOCDEL

**Deactivates** the online correction **and deletes** the correction values

**Please note for online correction:**

- Online correction is not possible in ”Manual” operating mode (jogging mode) or in ”Manual approaching the reference point”.
- Machine-oriented absolute position:
  G76 traverses to a displaced position, i.e. online correction is not calculated back.
- CPL functions PPOS and CPROBE do not take the correction value of the online correction into account.
- Probe:
  G75 measures the correct position. Use CPL function PROBE to read the measured value.
- Measuring at the fixed stop:
  G375 measures the correct position. Use CPL function PROBE to read the measured value.
- Limit switches:
  A coordinate position generated by online correction is not analyzed by the NC for a possible travel beyond the software limit switches. For this purpose, you should activate “Limit switch control” in the SERCOS drive.

## 3.99 Jogging in workpiece coordinates JogWCSSelect

Effect

The function can jog coordinates/pseudo-coordinates of a channel in workpiece coordinates (WCS) and in Z direction of tool coordinates (TCS). The ”Set-up, jogging in workpiece coordinates” operating mode is available for this purpose.

The following **coordinates** may be jogged:

- all pseudo-coordinates (axes) if axis transformation has been deactivated
- with active 5 axis transformation: all linear and orientation coordinates, the TCS Z direction and the pseudo-coordinates (axes).

The coordinate/pseudo-coordinate to be jogged is selected by the PLC via the NC block input (program module -B04SATZV) using the JogWCSSelect.. NC function.
Alternatively, the coordinate may also be selected in any part program, e.g., a CPL program.

Before jogging, the ”Set-up, jogging in workpiece coordinates” **operating mode** must have been selected:

- directly by the PLC (mode 14), if mode selection by the PLC is active.
- at the MMI using the ”Jog” softkey, if initialized for jogging workpiece coordinates in MACODA 6001 00030.

Programming

Selecting a coordinate:

JogWCSSelect JWSCHAN<Channel no.> JWSCOORD<Coordinate no.> {JWSFEED<F value>} {JWSSTEP<Increments>}
where

JWSCHAN<Channel no.> Number of the channel for which a coordinate is selected.

JWSCOORD<Coordinate no.> 1...8: The number of the coordinate
9: TCS Z direction
- the TCS Z direction only exists with active 5 axis transformation.
- a correction in TCS Z direction is converted into a movement of the linear working range coordinates (x,y,z).

{JWSFEED<F value>} **Optional,**
**default:** corresponds to feedrate setting at the axis interface (refer to MP 7050 01020).
Unit: mm/min or degrees/min (G71), or
inch/min or degrees/min (G70)

| | |
|---|---|
| JWSCHAN<Channel no.> | Number of the channel for which a coordinate is selected. |
| {JWSSTEP<Increments>} | **Optional,**<br>selects incremental jogging and simultaneous specification of the step size in increments.<br>**Default:** incremental and continuous jogging mode setting at the axis interface<br>(refer to MP 7050 01020). |

☞ **JWSSTEP may only be programmed together with JWSFEED.**

Notes:

# 4 Spindles

Spindles may be operated:

- as individual spindles
- in spindle groups.

For operation, spindles are implicitly

- assigned to channels
- transferred from one channel to another.

A spindle may be operated:

- in speed mode
- in position mode, or
- in synchronism with other spindles.

Spindles can be programmed

- in the part program
- by manual data input, or
- via machine functions,

and can be

- set via the interface.

Spindles are programmed using

- M functions
- S functions
- G functions
- special functions for “position mode“.

## 4.1 Individual spindles, spindle groups and channels

Individual spindles and spindle groups are not assigned to a specific channel in the PNC until a channel makes an implicit “reservation” for the spindles required.

Each individual spindle or spindle group has the following **standard functions**:

- Clockwise rotation, with/without coolant
- Counterclockwise rotation, with/without coolant
- Stop
- Positioning (spindle orientation)
- Automatic gear selection
- Manual gear selection
- Spindle speed programming
- Tapping without compensation chuck (G32)

☞ **The functionality of spindle groups is not to be confounded with a group of two or more coupled spindles with position control, which are operated in positional synchronism (refer to sect. 4.4).**

The following contains explanations of spindle groups, channel reservation and all functions that may occur in the context of spindle programming.

## 4.1.1 Assigning individual spindles to spindle groups

Effect

All 8 spindles can be grouped together to a maximum of 4 spindle groups. The spindles of a spindle group are programmed together, so that a less parametric programming is required as opposed to the individual programming of the spindles. The spindles of a spindle group are also designated as parallel spindles.

Programming (refer to sect. 4.2, "Spindle functions") a whole spindle group requires less parametric programming than entering the parameters of each individual spindle assigned to a group. When programming a spindle group function, the auxiliary functions of both the spindle group function and of the functions of the individual spindles assigned to this group are displayed on the interface, provided that the corresponding bit-coded auxiliary functions have been programmed in MACODA.

Each spindle assigned to a spindle group can be activated both via spindle group functions and via individual spindle functions. If an NC block contains concurrent instructions for spindle groups and for individual spindles, the control unit sends a runtime error message.

**Modal assignment of spindles to spindle groups:**

Spindle groups are specified by programming them in the part program / via manual data input for each channel individually, i.e. one and the same spindle may be assigned to different spindle groups on different channels. In order to modify a spindle group, the spindles to be newly assigned to this spindle group must be specified, e.g.

Programming

| | |
|---|---|
| SPG1(1,2,3) | means that from now on spindles 1, 2 and 3 are assigned to spindle group 1 on this channel.<br>(SPG = spindle group) |

At the time a spindle group is being programmed, the spindles required to form this group may still be included in some other groups. However, every spindle may be assigned to any spindle group at any time. When a spindle function is programmed for a spindle group, prior to its activation the respective spindles are checked as to whether they are released by the so-called spindle data management and are thus available for being assigned. This means that if a spindle was previously activated by another channel, this spindle must now be switched to speed mode (no C axis mode) and be in STOP state (M5).

The spindle assignment entered in MACODA for a spindle group can be reset to its original configuration on a channel by programming as follows:

| | |
|---|---|
| SPGn(0) | n = spindle group index 1 ... 4 |

A spindle group can be disbanded on a channel by entering the value ”–1”:

| | |
|---|---|
| SPGn(–1) | No spindles are assigned to the respective spindle group any more on this channel. |

**Examples**:

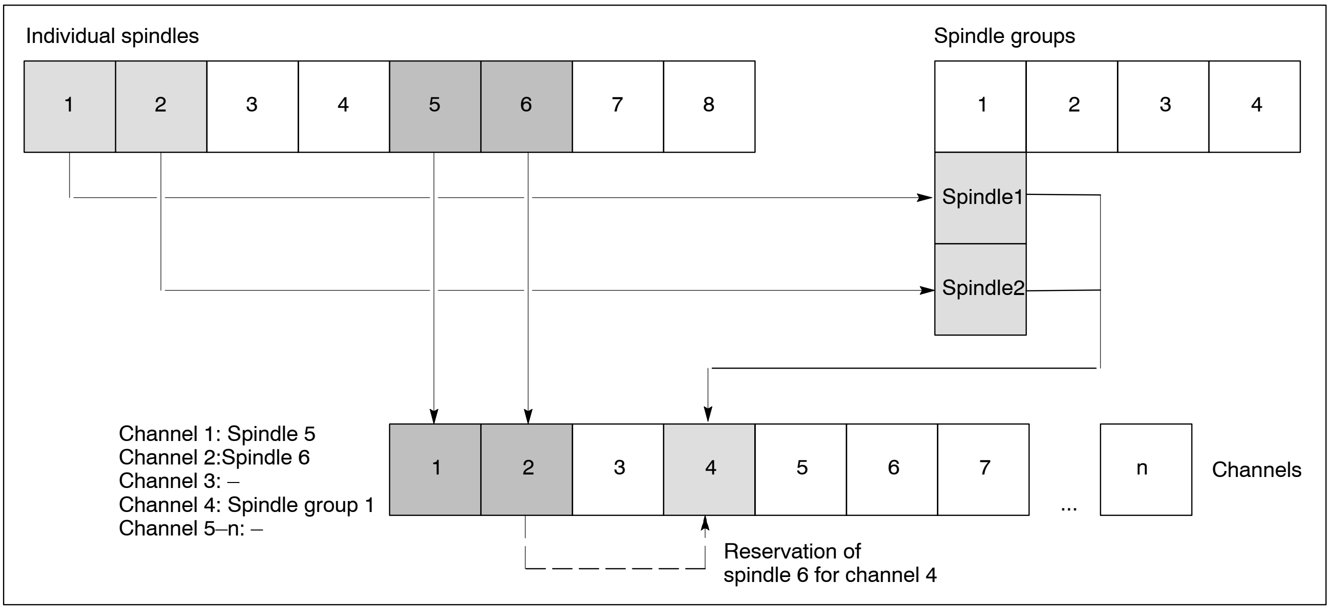
| | |
|---|---|
| `N... SPG1(1,2,3)` | Spindle group 1 is created of spindles 1, 2 and 3. |
| `N... M19` | Spindles 1, 2 and 3 traverse to their respective reference point (M19 = default for SPG1) |
| `N... SPG2(2,4,5)` | Spindle group 2 is created of spindles 2, 4 and 5, i.e. spindle 2 is removed from SPG1 |
| `N... M19` | Spindles 1 and 3 traverse to their respective reference point (M19 = default for SPG1) |

**Restoring spindle group default settings:**

| | |
|---|---|
| SPGALL(0) | By programming SPGALL(0), the default setting according to MACODA is restored for all spindle groups of the channel. |

The assignment of the individual spindles to a spindle group is preset in MACODA parameter 1040 00002. It may be changed by programming in the part program and by manual input.

## 4.1.2 Reserving spindles and spindle groups for specific channels



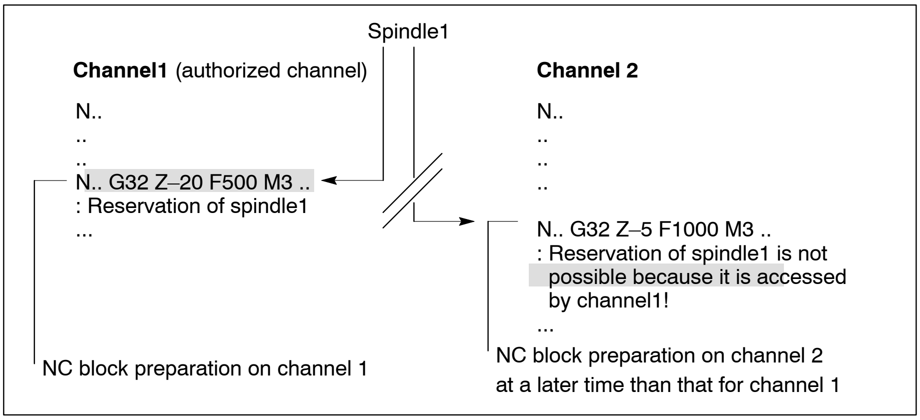
Effect

As a rule, no spindle (spindle group) is permanently assigned to any specific channel. Therefore, all spindles (spindle groups) can be activated from **any** channel.

Whenever you enter a traversing motion for a spindle on a channel via the part program or manual data input, the spindle concerned is reserved implicitly for that respective channel. This applies irrespective of whether the input of the traversing motion is made through an individual spindle function or a spindle group function.

Access to reserved spindles from another channel is blocked, i.e. any attempt to select them from another channel will result in a runtime error (exception: conditional spindle release, see below, Programming with SADM).

Until a reserved spindle is released, it can be activated only on the channel (authorized channel) where the reservation originated. Therefore, a spindle is not released and thus does not become available to any other channel until it is stopped by its authorized channel.


Spindle1
Channel1 (authorized channel)
N..
..
..
N.. G32 Z–20 F500 M3 ..
: Reservation of spindle1
...
NC block preparation on channel 1
Channel 2
N..
..
..
..
N.. G32 Z–5 F1000 M3 ..
: Reservation of spindle1 is not possible because it is accessed by channel1!
...
NC block preparation on channel 2 at a later time than that for channel 1

### Making a spindle reservation on a channel:

Programming

The following spindle functions produce an **implicit** spindle reservation for the calling channel:

- M3, M13
- M4, M14
- M19
- G32 (tapping without compensation chuck)
- G96 (constant cutting speed)

Any spindle functions available may be programmed on the channel from where the spindle was reserved:

- Spindle speed programming S, S1 – S8, SSPG1 – SSPG4
- M3, M13, M4, M14, M5, M19
- M40, M41–M44, M48
- G32
- G192, G292
- G96

It is forbidden for any other than the authorized channel to activate a reserved spindle. The attempt will produce runtime error 2001: “Spindle is used by another channel!”

The following functions are subject to this restriction:

- Spindle speed programming S, S1 – S8, SSPG1 – SSPG4
- M3/M13, M4/M14, M19, M5
- M40, M41–M44, M48
- G32
- G192, G292
- G96

If called from the init string of an unauthorized channel, the following functions are suppressed and, therefore, no runtime error will occur:

- M5
- M40, M41–M44, M48
- G96, G97

### Releasing a spindle reserved for a channel:

Programming

A reserved spindle can be released on the authorized channel:

- by programming M5, or
- at the end of G32 if the spindle was reserved using G32.

Termination of a part program with M30 or a control reset on the authorized channel has the following effects:

- deselection of M40 (if active), current gear range remains selected,
- deselection of G96, spindle speed programming is activated, and
- functions entered in the init string (MACODA parameter 7060 00020) are activated:

| | |
|---|---|
| M5 | Spindle is stopped if M3/M13, M4/M14 were active or if the spindle was positioned with M19. Subsequently, the spindle is released. |
| M40 | Selection/Repeat selection of the ”Automatic gear selection” function |
| M41–M48 | Automatic gear selection may have already been deselected (see above). Manual gear selection in the init string does not produce a gear switch. |

### Transferring a reserved spindle to another channel:

Effect

In exceptional cases, it may be necessary to activate a spindle reserved with M3, M4 or M19 from an adjacent channel (part program or manual data input).

In those cases, NC function SADM makes it possible for the authorized channel currently “owning” the spindle to transfer the spindle reservation **to another channel**.

Programming

### Conditional spindle release:

SADM Si=0 ... Sn=0

A channel currently holding a reservation for one or more spindles, which was made by programming M3, M4 or M19, is caused by this command to grant **any other channel** the right to access the spindle:

- With the SADM command, the spindle can now be accessed by any other channel.
- The spindle can be activated via machine functions any time.

Programming

### Activating a conditionally released spindle from another channel:

| | |
|---|---|
| SADM Si=1 ... Sn=1 | This command allows you to activate (a) conditionally released spindle(s) from another channel via a part program or by manual data input. |
| i | Index of the i-th spindle (i=1..n). |
| n | number of available spindles (currently: $n_{max.}$ .=8) |

## 4.2 Functions for individual spindles and spindle groups (M functions) in speed mode

### 4.2.1 Spindle functions

Effect

**Spindle functions** can be programmed in the part program or by manual data input for individual spindles or for spindles assigned to a spindle group.
Each of the 8 spindles may be assigned optionally to one of 4 spindle groups.

Any number of individual spindles and/or spindle groups may be programmed in one NC block.

Syntax

The syntax for every spindle function of every spindle/spindle group is determined in MACODA. Apart from common M functions, the various functions may be assigned freely defined names with up to 8 digits.

☞ **No distinction is made between individual spindles and spindle groups in the following description of the various spindle functions because they show the same behavior. Basically, programming a spindle group just means less programming effort.**

☞ **The M functions described below are suggestions of MACODA settings.  For better transparency, the previous (fixed) M codes are used as default parameters.**

Declaration applying to the documentation below:

| Syntax | Assignment | Example |
|---|---|---|
| 1 | 1st Spindle group | M3 |
| 2 | 1st Spindle | M103 |
| 3 | 2nd Spindle | M203 |

Programming

**Spindle, clockwise rotation:**

M3
M103
M203

The spindle(s) start(s) rotating clockwise (viewed when facing the spindle working range).
The spindle speed can be set via the pertinent S address. The spindle speed may be programmed together with M3 in one and the same block.

The function remains **modally active** until it is canceled by another command for the spindle(s) concerned.
This means that after a gear change, e.g., the type of motion is restored that was active previously.

Programming **Spindle, clockwise rotation and coolant ON:**

M13<br>M113<br>M213 Same as with M3, M103, M203, plus activation of coolant.

Programming **Spindle, counterclockwise rotation:**

M4<br>M104<br>M204 The spindle starts rotating counterclockwise (viewed when facing the spindle working range). Otherwise the same as ”Spindle, clockwise rotation”.

Programming **Spindle, counterclockwise rotation and coolant ON:**

M14<br>M114<br>M214 Same as with M4, M114, M214,  plus activation of coolant.

Programming **Spindle stop:**

M5<br>M105<br>M205 The spindle is stopped. This command remains active until it is canceled by another spindle command.

Programming **Programmable spindle orientation (spindle orientation):**

M19
M119
M219

The spindle positions itself at a specific angle. This function may be executed while the spindle is at a standstill or rotating. When at a standstill, the spindle will orient itself along the shortest possible path. When rotating, the spindle will maintain its last direction of rotation.

By activating this function, the drive will automatically change to the internal position control mode if the speed is below the positioning speed (SERCOS parameter S-0-0222). As soon as another spindle command (M3, M4, M5) is activated, drive operation mode ”Position control” is deactivated. This function may be programmed alone or together with other M or G instructions. However, there must not be any other function programmed for one and the same spindle that would have to run concurrently (e.g. M3, M4, M5).

The spindle orientation function may be programmed in a block with or without the corresponding S word:

- Programming **without** the S word:
  The spindle positions itself relative to its reference point (refer to SERCOS interface).
- Programming **with** the S word (= positioning angle in degrees)
  - The spindle positions itself at the angle specified by the S word, which is relative (i.e. additive) to its reference point. Angles programmed outside the interval [ 0° ≤ positioning angle < 360° ] will be translated by the control unit to match the permitted interval. This has the effect that the spindle will never have to traverse for more than one rotation.
  - If the spindle is already in the correct position, no motion is executed.

**Examples**: programmable spindle orientation

| | |
|---|---|
| `N... M19` | The spindles of the 1st spindle group position themselves relative to their respective reference angle. |
| `N... M119` | The 1st spindle positions itself to its reference angle. |
| `N... M219` | The 2nd spindle positions itself to its reference angle. |
| `N... M19 S180` | The spindles of the 1st spindle group position themselves at 180°. |
| `N... M119 S1= –180` | The 1st spindle positions itself at 180°. |
| `N... M219 S2=370` | The 2nd spindle positions itself at 10°. |
| `N... PTEST10 M119` | The 1st spindle positions itself to its reference angle. Subsequently, subprogram “TEST10” is executed. |

## 4.2.2 Gear functions

Effect

The total speed range available on a machine is divided by **switched gears** into several smaller speed ranges, for which various gear ranges may be defined.

The number of gear ranges (max. 4), their speed limits (min./max speed) and other specific spindle parameters are defined in group 1040 of MACODA.

☞ **Gear selection functions have no impact whatsoever on analog spindles.**

Programming

**Automatic gear selection**

M40
M140
M240

By means of the M function, gear selection is programmed once at the beginning of the part program.

On the basis of the programmed speed, the control unit selects the appropriate gear from the maximum of 4 gear ranges that may be programmed in MACODA.

Please note:

- In the event of overlapping speed ranges of the various gears, the control unit will always select the lower gear (with the higher motor speed).
- Programming ”0” for the speed will have the effect that **no** gear changes are carried out.

In MACODA blocks 7060 00010 and 7060 00020, you can configure the automatic gear selection function to be the default status.

**Examples**:

| | |
|---|---|
| `N... M40` | Automatic gear selection for 1st spindle group |
| `N... M140` | Automatic gear selection for 1st spindle |
| `N... M240` | Automatic gear selection for 2nd spindle |

Programming

## Manual gear selection

If desired, the gear range (1–4) for every spindle/spindle group may be entered **manually** in the part program. In this case, the control unit deselects the automatic gear selection function.

If a speed outside the speed range of a gear is entered in the case of manual gear selection, the PNC will display the minimum or maximum speed for the respective gear.

| | |
|---|---|
| M41–M44<br>M141–M144<br>M241–M244 | Manual gear selection is programmed in the part program, **if required**. |

**Examples**:

| | |
|---|---|
| `N... M42` | Manual selection of 2nd gear for 1st spindle group |
| `N... M141` | Manual selection of 1st gear for 1st spindle |
| `N... M244` | Manual selection of 4th gear for 2nd spindle |

Programming

## Neutral gear

| | |
|---|---|
| M48<br>M148<br>M248 | Changes the gear of the spindle/spindle group to neutral. Afterwards, the gear of the spindle/spindle group will be in neutral. |

## 4.2.3 Specifying the spindle speed

Effect

The spindle speed refers to individual spindles, or – in the presence of several spindles – to all spindles of a spindle group.

Programming

**Spindle speed input:**
Si=,... Sn=
SSPGj=,.., SSPGm=
S

where

| | |
|---|---|
| Si= | Speed specified for the ith spindle(s). |
| i | Index of the ith spindle (i=1..n). |
| n | number of available spindles ($n_{max.}$=8) |
| SSPGj= | Speed specified for the jth spindle group. |
| j | Index of the jth spindle group (j=1..m). |
| m | number of available spindle groups ($m_{max.}$=4) |

S Abridged programming format for the

- speed of the whole spindle group containing the 1st spindle according to default setting.
- speed of the 1st spindle only: S $\equiv$ S1, provided that the 1st spindle is not assigned to any spindle group according to MP 1040 00002.

**Please note for spindle speed specifications:**

- The speed values programmed are interpreted by default as revolutions per minute.
- When G96 is active, the programmed speed is interpreted as the cutting speed in m • rpm.
- A programmed speed value can be modified with the spindle-specific override. An override setting of 100% is equivalent to the programmed speed value.
- The control unit will always reduce outputs of the speed setpoint to comply with the limits entered in MACODA. Please note that these limits depend on the selected gear.
- You can set an additional speed limit by programming G192 or G292.
- The set speed value is applied until it is overwritten by a new ”S word” (acting modally).
- In test mode (general inhibit, please refer to the operating manual), there is no speed output to spindles.
- Programmed speed values of auxiliary spindles are ouput only in the form of auxiliary functions (e.g. 32 bit aux. functions)

**Examples**:

| | |
|---|---|
| `N.. G97` | Speed programming active. |
| `N.. G.. X.. Y.. Z.. F..`<br>`SSPG1=1000` | The spindles of the 1st spindle group are to rotate at 1000 rpm. |
| `N.. G.. X.. Y.. Z.. F..`<br>`S1=2000` | 1st spindle speed: 2000 rpm |
| `N.. G.. X.. Y.. Z.. F..`<br>`S3=2000` | 3rd spindle speed: 2000 rpm |
| `N.. G.. X.. Y.. Z.. F..`<br>`S1500` | The 1st spindle or the spindle group to which the 1st spindle is assigned by default is to rotate at 1500 rpm. |

**CAUTION**
**Incorrect programming may cause machine damage!**
**In combination with the spindle positioning function (M19, ...), the control unit will interpret the S word not as speed but instead as the positioning angle! The meaning of the S word (spindle speed/ cutting speed) is defined by G97/G96!**

### 4.2.4 Activating spindles via machine functions

To activate a spindle via the machine functions, the following conditions apply:

- The spindle was stopped by M5.
- The spindle was started by a part program, which has been stopped by a feed-hold command (spindle is reserved for this channel).
  In this case, the spindle must not be reserved by G32 (tapping without compensation chuck).
- The spindle was started via a machine function.
- The spindle was started in set-up mode (spindle manual or spindle jog).

Spindle specification via machine functions does not lead to a reservation of the spindle. Therefore, it can be activated at any time by a part program, by an external NC block input or by jogging.

Programming

**The following spindle functions may be selected:**

- Speed S or SSPG
- Spindle functions M3/M13, M4/M14, M5 or M19
- gear ranges M41–M44 or M48

☞ **If a spindle in a part program that has been stopped by a feed-hold command is switched to constant cutting speed (G96), no S word may be specified (error 2575).**

## 4.2.5 Activating the spindle via the interface

To activate a spindle via interface (spindle manual or spindle jog), the following conditions apply:

- The spindle was stopped by M5.
- The spindle was started by a part program, which has been stopped by a feed-hold command (spindle is reserved for this channel).
  In this case, the spindle must not be reserved by G32 (tapping without compensation chuck).
- The spindle was started in set-up mode (spindle manual or spindle jog).

Spindle specification via the interface does not lead to a reservation of the spindle. Therefore, it can be activated at any time by a part program, by an external NC block input or by jogging.
No spindle-specific activities are initiated when exiting set-up mode.

Programming

**The following spindle functions may be selected:**

- ManualM3, ManualM4, ManualM5, ManualM19
- JogM3, JogM4

☞ **If a spindle in a part program that has been stopped by a feed-hold command is switched to constant cutting speed (G96), no S word may be specified (error 2575).**

## 4.3 G functions involving spindle programming

The following G functions involve spindles:

| | |
|---|---|
| G32 | Tapping without compensation chuck |
| G33 | Tapping |
| G95 | Feedrate programming in mm/rev. |
| G97 | Direct speed programming |
| G96 | Constant cutting speed |
| G104 | Dwell time in spindle revolutions |
| G192 | Speed limitation, minimum speed |
| G292 | Speed limitation, maximum speed |
| G517 | C axis OFF |
| G518 | C axis ON |
| G533 | Additional tapping functions |

For detailed descriptions, please refer to section “G instructions”.

## 4.4 Special spindle functions

### 4.4.1 Spindle functions in position mode

Effect

In normal operation, spindles are always operated in spindle speed mode.
In **special cases** (e.g. ”Spindle operation in positional synchronism”), spindles must be activated **in position mode** (position control).

The following functions are available for **position mode**:

| | |
|---|---|
| ● Spindle, clockwise rotation | (refer to sect. 4.2.1) |
| ● Spindle, counterclockwise rotation | (refer to sect. 4.2.1) |
| ● Spindle stop | (refer to sect. 4.2.1) |
| ● Spindle orientation | (refer to sect. 4.4.2) |
| ● Spindle operation in positional synchronism | (refer to sect. 4.4.5 and 4.4.2) |

### 4.4.2 Spindle referencing

When position mode is active, the spindle reference point must be known for the following **functions**:

- Spindle orientation
- Spindle operation in positional synchronism

The reference point is determined by means of the spindle positioning function (M19) while **spindle speed mode** is **active**.

Programming

**Spindle referencing**

M<19> Si ...Sn

Spindles are referenced with the spindle positioning function (M19 or application-specific M function) while spindle speed mode is active.

### 4.4.3 Switching to position-controlled spindle operation

Effect

For spindle operation with spindle position control, the spindle must be switched from speed mode to position mode.

Programming

**Manual drive interface switching: SDOM**

SDOM Si=0|1 ... Sn=0|1
or
SpDriveOpMode Si=0|1 ... Sn=0|1

where:

Si Switches the drive interface of the i-th spindle(s) to:
Si=0: Speed mode or
Si=1: Position mode

n number of available spindles (currently: $n_{max.}$=8)

i Index of the i-th spindle (i=1..n)

☞ **Function SDOM should be called only after the spindle was stopped with M5 because otherwise a spindle-stop command is triggered internally.**

☞ **The operating mode of a main spindle (for main spindle, refer to page 4–17) can also be switched over in connection with G533 (G533 SPC).**

## 4.4.4 Main spindle switchover

Effect

Switches over the main spindle.

Functions G33, G95 and G104 act upon the main spindle within a channel. The desired main spindle can therefore be defined

- statically by MACODA parameter 7020 00010, or
- dynamically in the part program using the MAINSP(..) function.

Programming

**MAINSP**<Num>
or
**MAINSP**(<NumNam>)

where

<Num> number of the spindle: Any number between 1 and 8.
–1: Reset to MACODA setting.

<NumNam> number of the spindle: Number between 1 and 8
**-or-**
name of the spindle in inverted commas (e.g. ”S1”)
**-or-**
CPL integer variable containing the spindle number
**-or-**
CPL string variable containing the spindle name
**-or-**
–1: Reset to MACODA setting.

## 4.4.5 Spindle operation in positional synchronism

Spindle operation in positional synchronism is required mainly for lathes (e.g. for clamping a workpiece/tool onto 2 spindles facing each other, for transferring workpieces, etc.).
These processes require several spindles running simultaneously and in positional synchronism (spindle coupling).

### Definition of spindle coupling

Up to **4 different groups of coupled spindles** can be activated simultaneously by the PNC.

- A group of coupled spindles consists of one leadscrew (master) and up to seven slave spindles.
- The slaves can be coupled with the master at any offset angle between 0° and 359.9999° (coupling distance).
- When coupling is active, every slave spindle can be turned by up to ±3600° (absolute) relative to its coupling distance.
- Slave spindles can be added or removed while coupling is active.

The following **boundary conditions** must be fulfilled for spindle coupling:

- The spindle drives must be configured as endless rotary axes (modulo axes) (refer to “PNC functional description” manual).
- All spindles of a group of coupled spindles must have a common speed range.
- All spindles of a group of coupled spindles must have similar dynamics.
- All spindles of a group of coupled spindles must be equipped with an encoder system which, when M19 (Spindle positioning) is used in speed mode, determines simultaneously the reference points of
  - the spindle (spindle speed mode)
  - the ”C axis ” (position mode).

### Restrictions applying to spindle operation in positional synchronism:

- The maximum spindle speed of a group of coupled spindles depends on the NC cycle time (MACODA parameter 9030 00001):

  **$S_{max}$[min$^{-1}$] = 14400 / NC cycle time [msec]**

  Example: NC cycle time = 4 min$^{-1}$
  $S_{max}$ = 14400/4= 3600 min$^{-1}$

## 4.4.6 Configuring slave spindles

**Coupling distance: SCD**

Effect

The coupling distance defines the positional difference in setpoint values between the master spindle and its slave spindle(s) from the time coupling takes effect.

Programming

**Setting the coupling distance for one or several slave spindles:**

SCD Si=<distance i>... Sn=<distance n>
or
SpCoupleDistance Si=<distance i>... Sn=<distance n>

where:

| | |
|---|---|
| Si | i-th slave spindle(s) |
| distance i | Positional difference in setpoint values of the i-th slave spindle(s) from the time coupling takes effect |
| n | number of available spindles (currently: $n_{max.}$ =8) |
| i | Index of the i-th spindle (i=1..n) |

| | |
|---|---|
| range of values (distance): | –359.9999° .. + 359.9999° |
| Default: | 0° |
| Validity: | spindle-specific date |

**Synchronous mode window: SCSW**

Effect

At the start of spindle synchronization, the NC waits until the deviation of the actual position values from the setpoint values of the respective slave spindles lies within the interval defined by the window [–value,+value]. When synchronous spindle mode is active, this window is monitored.
If an error occurs, the IF signal ”positional synchronism 1” on the spindle-specific output interface is reset.

Programming

**Defining the synchronous mode window:**

SCSW Si=<window i>... Sn=<window n>
or
SpCoupleSyncWindow Si=<window i>... Sn=<window n>

where:

| | |
|---|---|
| Si | i-th slave spindle(s) |
| window i | specification of the synchronous mode window for the i-th slave spindle(s) |
| n | number of available spindles (currently: $n_{max.}$=8) |
| i | index of the i-th spindle (i=1..n) |

| | |
|---|---|
| range of values (window): | 0° .. 20° |
| Default: | 1° |
| Validity: | spindle-specific date |

### Synchronous mode error window: SCEW

Effect

When synchronous spindle mode is active, this window is monitored. If an error occurs, the IF signal ”positional synchronism 2” on the spindle-specific output interface is reset.

Programming

**Defining the synchronous mode error window:**

SCEW Si=<window i>... Sn=<window n>
or
SpCoupleSyncErrorWindow Si=<window i>... Sn=<window n>

where:

| | |
|---|---|
| Si | i-th slave spindle(s) |
| Window i | specification of the synchronous mode error window for the i-th slave spindle(s) |
| n | number of available spindles (currently: $n_{max.}$=8) |
| i | Index of the i-th spindle (i=1..n) |

| | |
|---|---|
| Range of values (window): | 0° .. 359.9999° |
| Default: | 10° |
| Validity: | spindle-specific date |

## 4.4.7 Defining groups of coupled spindles

**Creating, modifying or disbanding groups of coupled spindles: SCC**

Effect

With the SCC command, you can define, modify (add or remove slave spindles) or delete groups of coupled spindles.

When groups of coupled spindles are defined or new slave spindles are added to them, they are subsequently switched automatically to position mode, if required.

Conversely, when slave spindles are removed or a group of coupled spindles is disbanded, the spindles are automatically switched back to speed mode if this mode was active prior to coupling.

Please note for the following **functions**:

| | |
|---|---|
| CP, couple | group of coupled spindles |
| group j of coupled spindles | number of the j-th group of coupled spindles: 1 ... 4 |
| MA, master | master spindle |
| S<number i> | i-th slave spindle(s) |
| number | physical spindle index of the master spindle: i= 1 ... n |
| number i ..n | physical spindle index of the i-th slave spindle |
| n | number of available spindles (currently: $n_{max.}$=8) |
| i | Index of the i-th spindle (i=1..n) |
| j | j = 1...4 |

Programming

**Definition of a group of coupled spindles:**

SCC CP=<group j> MA=<number> S<number i>=1 ...
S<number n>=1>
or
SpCoupleConfig Couple=<group j> Master=<number>
S<number i>=1 ... S<number n> = 1

Programming

**Adding slave spindles to or removing slave spindles from a group of coupled spindles:**

SCC CP=<group j> S<number i>=0|1 ... S<number n>=0|1
or
SpCoupleConfig Couple=<group> S<number i>=0|1 ...
S<number n>=0|1

where

S<number i> i-th slave spindle(s):
S<number i>=0: remove spindle from group, or
S<number i>=1: add spindle

When modifying a group of coupled spindles, there is no need to program the number of the master spindle because the group of coupled spindles is already clearly identified by its number.

Programming

**Disbanding a group of coupled spindles**:

SCC CP=<group j> MA=0
or
SpCoupleConfig Couple=<group j> Master= 0

**Waiting for synchronous mode: SCWAIT**

Effect

The part program waits until the programmed group of coupled spindles is successfully created, reconfigured or disbanded. The effect of this function is equivalent to that of a conditional WAIT.

**Waiting for synchronous mode:**

Programming

SCWAIT CP=<group j>
oder
SpCoupleWaitSync Couple=<group j>

## 4.4.8 Programming while coupling is active

### Entering an angular offset while coupling is active: SCPO

Effect

When coupling is active, the stated angular offset is approached. This angle acts additively on the existing coupling offset. Thus, you can twist the master and slave spindles with reference to each other while coupling is active; the absolute offset angle (coupling distance SCD + angular offset SCPO) between master and slave spindles can be redefined any time.

The torsion may be carried out while spindle rotation is active.

While a torsion is active, the "positional synchronism 1" signal is reset on the spindle-specific output interface.

Programming

**Entering an angular offset while coupling is active:**

SCPO S<number i>=<offset i> ... S<number n>=<offset n> {POSVEL<speed>}
or
SpCouplePosOffset S<number i>=<offset i> ... S<number n>= <offset n>{POSVEL<speed >}

where

| | |
|---|---|
| S<number i> | i-th slave spindle(s) |
| offset i ... offset n | Torsion angle of the i-th slave spindle(s). The torsion angle is entered as an absolute value: ±3600° |
| Speed | Speed ratio between master and slave spindle at which the stated offset is activated. This parameter is optional and acts modally. The default value is the respective standard spindle speed specified via SERCOS ident number S-0-0222. |
| n | number of available spindles (currently: $n_{max.}$=8) |
| i | Index of the i-th spindle (i=1..n) |

### Waiting for angular offset: SCPOWAIT

Effect

The part program is stopped until the angular offset programmed with SCPO is completed. The effect of this function is equivalent to a conditional WAIT.

Programming

**Waiting for angular offset:**

SCPOWAIT CP<group j>
or
SpCouplePosOffsetWait Couple=<group j>

where

| | |
|---|---|
| group j | number of the j-th group of coupled spindles: 1...4 |

## 4.4.9 Spindle coupling process

### 1. Creating a coupling

The following conditions must be fulfilled for creating a spindle coupling:

- The **coupling parameters** (distance, synchronous mode window, ...) have been configured.
- The **reference points** of the spindles involved have been determined with M19 in speed mode.
- **Position mode** is activated for all spindles involved:
  Failing position mode activation for a spindle involved, the spindle concerned is stopped, switched to position mode, and restarted.

Following programming of SpCoupleConfig (SCC):

- The **limits applying to the group of coupled spindles** (spindle speed, acceleration) are determined and relayed to the future master spindle.
- The ”**coupling number**” is output on the interfaces of all spindles involved (master and slave(s)).
- ”**Spindle is master**” is output on the interface of the master spindle.

Varying with the state of motion of the spindles involved, **various sequences of motions** may occur in the coupling process:

| **Creating a coupling** | **Sequence of motions in the coupling process** | **Note** |
|---|---|---|
| Master and slave spindles are at rest **(M5 or M19)**: | Slave spindle approaches its coupling point on the **shortest possible path**. Upon reaching the synchronous mode window, ”Positional synchronism 1” and ”Positional synchronism 2” are output on the interface. | The PLC must authorize a slave spindle motion. This authorization can be generated by evaluating the IF signals ”Coupling number” + ”Spindle command”. |
| Master spindle at rest **(M5 or M19),** slave spindle rotating **(M3 or M4)**: | Slave spindle approaches its coupling point **directly**. Upon reaching the synchronous mode window, ”Positional synchronism 1” and ”Positional synchronism 2” are output on the interface. | |

| **Creating a coupling** | **Sequence of motions in the coupling process** | **Note** |
|---|---|---|
| Master spindle rotating **(M3 or M4),** slave spindle at rest **(M5 or M19)** | Slave spindle speed is **accelerated** to match the speed of the master spindle. When their speeds match, the slave spindle approaches its coupling point (SpCoupleDistance) on the **shortest path**. Upon reaching the synchronous mode window, "Positional synchronism 1" and "Positional synchronism 2" are output on the interface. | The PLC must authorize a slave spindle motion. This authorization can be generated by evaluating the IF signals "Coupling number" + "Spindle command". |
| Master and slave spindles rotating **(M3 or M4)**: | Slave spindle speed is **accelerated** or **slowed down** to match the speed of the master spindle. When their speeds match, the slave spindle approaches its coupling point on the **shortest path**. Upon reaching the synchronous mode window, "Positional synchronism 1" and "Positional synchronism 2" are output on the interface. | |

### 2. Coupling is active

The slave spindles follow the master spindle.
If the limits of the programmed synchronous mode window and/or the synchronous mode error window are exceed, this is signaled by the NC by

- resetting the IF signal "Positional synchronism 1" (synchronous mode window),
- resetting the IF signal "Positional synchronism 2" (synchronous mode error window).

While an offset angle is activated with SpCouplePos Offset (SCPO), the NC resets the IF signal "Positional synchronism 1".

### 3. Uncoupling

When a slave spindle is uncoupled, it takes over the active motion functions (spindle speed and direction of rotation) of the master spindle.

At the interface, the following signals are reset:

- Coupling number
- Spindle is master
- Positional synchronism 1, and
- Positional synchronism 2.

If the "Spindle orientation" function is active when spindles are uncoupled, the slave spindles are switched to M5 (spindle stop).

## 4.4.10 Test mode with spindle coupling active

**DANGER**
**By switching to test mode all groups of coupled spindles are stopped and uncoupled.**

## 4.4.11 Effects of spindle-specific interface signals on spindle couplings

### IF signal “Drive off”

The IF signal ”Drive OFF” causes the **NC to slow down** the group of coupled spindles **to a stop**. The state of motion of the master spindle is set to spindle stop (M5). When the group of coupled spindles has come to a standstill, the “Drive OFF” signal of the spindle concerned is **relayed** to the drive.

This will produce the following effects:

- The NC blocks any further programming of the group of coupled spindles.
- The IF signals ”Positional synchronism 1” and ”Positional synchronism 2” of all slave spindles are reset.
- The IF signal ”Coupling error” is set for the master spindle.
- The state of motion of all spindles involved is set to spindle stop (M5).

### IF signal “Drive inhibit”

When the IF signal ”Drive inhibit” is set, the NC must relay this signal to the drive. This will immediately open the control loop of the drive!

Therefore, the NC **cannot** intervene actively, instead, it can only respond:

- The NC slows down the remaining group of coupled spindles to a stop.
- The NC blocks any further programming of the group of coupled spindles.
- The IF signals ”Positional synchronism 1” and ”Positional synchronism 2” of all slave spindles are reset.
- The IF signal ”Coupling error” is set for the master spindle.
- The state of motion of all spindles involved is set to spindle stop (M5).

☞ **Fault conditions caused by IF signals “Drive OFF” and “Drive inhibit” can be overcome only by a master spindle control reset (IF signal) or an overall (system) control reset (PLC or operator input).**

## 4.4.12 Effects of drive-specific messages on spindle couplings

**Resetting ”Drive under control”:**

The NC responds to this signal by

- Slowing down the remaining group of couple spindles to a stop.
- The NC blocks any further programming of the group of coupled spindles.
- The IF signals ”Positional synchronism 1” and ”Positional synchronism 2” of all slave spindles are reset.
- The IF signal ”Coupling error” is set for the master spindle.
- The state of motion of all spindles involved is set to spindle stop (M5).

☞ **This fault condition can be overcome only by a master spindle control reset (IF signal) or an overall (system) control reset (PLC or operator input).**

**Diagnostics class 1 error**

A ”Diagnostics class 1 error” will immediately **open** the control loop of the drive. The NC will respond as follows:

- Slowing down the remaining group of couple spindles to a stop.
- The NC blocks any further programming of the group of coupled spindles.
- The IF signals ”Positional synchronism 1” and ”Positional synchronism 2” of all slave spindles are reset.
- The IF signal ”Coupling error” is set for the master spindle.
- The state of motion of all spindles involved is set to spindle stop (M5).

☞ **This fault condition (reset of ”Drive under control” or diagnostics class 1 error) can be overcome only by a master spindle control reset (IF signal) or an overall (system) control reset (PLC or operator input).**

Notes:

# 5 Auxiliary and special functions

In addition to path information, auxiliary and special functions are required to provide **technological** information.

Auxiliary functions are sent to the PLC. The transfer sequence is defined as follows:

- bit-coded auxiliary functions  
  They are collected in the sequence in which they have been programmed and sent in packages of 13 auxiliary functions or upon receipt of the last one.  
  An acknowledgement is compulsory only with the last package.
- bcd-coded auxiliary functions  
  They are sent individually in the sequence in which they have been programmed.  
  An exception are auxiliary functions that act internally (e.g. ”S”). They are sent last.
- combined programming  
  Bit-coded auxiliary functions are sent after each 13th auxiliary function or, resp. after the last one.  
  With the exception of auxiliary functions acting internally (”S”), bcd-coded auxiliary functions programmed previously are sent previously. Auxiliary functions acting internally are sent last.

**CAUTION**  
**The functions described below may have different effects on your machine!**  
**Many auxiliary and special functions may be implemented manufacturer-specifically. Therefore, the documentation provided by the respective machine tool manufacturer always takes priority.**  
**If you are not sure whether the functions described herein actually apply to your machine, contact your system administrator!**

## 5.1 F address (feedrate)

Effect

F addresses are used to determine the **feedrate of the tool** during machining.

However, the PNC may interpret F addresses in different ways.

Depending on the G instruction currently active, programmed F words will act as:

- interpolation time in seconds for G1, G2, G3 and G5 (see G93, page 3–111) or
- feedrate in mm/min or inch/min (see G94, page 3–112) or
- feedrate in mm/rev (see G95, page 3–116).

**DANGER**
**Failure to observe the feedrate preset in the machine parameters may pose a hazard to the machine and personnel!**

**”Power off”, ”Control reset” or ”Reset” will activate the F word set in machine parameters 7060 00020 or 7060 00010 (default value = F0)!**

**This parameter also contains the information whether G93, G94 or G95 will be active after the events mentioned above (default value = G94)!**

Programming

**Example:** Time programming with G93

| | |
|---|---|
| `N10 G93 G1 X300 Z400 A50 B120 F60` | The programmed linear interpolation will last 60 seconds. |

**Example:** Feedrate programming in mm/min with G94

| | |
|---|---|
| `N10 G1 G94 X200 Z300 F200` | programmed feedrate 200 mm/min |
| `N11 G4 F40` | dwell time 40 seconds |
| `N12 X300 Z400` | the 200 mm/min feedrate is active again |

**Example:** Feedrate programming in mm/rev with G95

| | |
|---|---|
| `N9  S2000 M4` | spindle speed 2000 rev/min, ccw run |
| `N10 G1 G95 X200 Z300 F0.2` | programmed feedrate 0.2 mm/rev |
| `N..` | |
| `N12 X300 Z400` | the 0.2 mm/rev feedrate is active again |

☞ **You may also program a dwell time using an F word in connection with G4 (cf. G4, page 3–12).**

## 5.2 FA address (feedrate of asynchronous axes)

Effect

Normally, asynchronous axes are traversed in rapid mode. If this behaviour is not desired under certain circumstances, you may use the FA address to influence the traversing speed of all asynchronous axes programmed in the same block.

**CAUTION**
**Incorrect programming may cause machine damage!**

**This feedrate will only be active in the block it is programmed, and it will only be effective for the asynchronous axes programmed in the same block as the FA word!**

**Programming asynchronous axes without the FA word in a subsequent block will let the axes traverse in rapid mode again.**

Programming

**Example:** Programming the feedrate of asynchronous axes in mm/min

| | |
|---|---|
| `N10 G1 G94 X200 Z300 F200` | programmed feedrate of synchronous axes of 200 mm/min |
| `N11 UA400 VA140 FA250` | the asynchronous axes UA and VA are traversed to the programmed positions at 250 mm/min |
| `N12 UA0 WA10` | The asynchronous axes UA and WA traverse to the programmed positions in rapid mode |

## 5.3 S address (spindle speed)

Refer to section 4.2.3 “Specifying the spindle speed“

## 5.4 M functions

M functions (which are sometimes also referred to as M codes) consist of the address letter M and a key number. A leading ”0” in the key number need not be written in the program.

**Example:** M function M03 (spindle ON – clockwise)

```
N ... G01  X200  Y145  Z–67.678  F250  S1000  T16  M03
             Coordinate data       Special functions  | |
                                                      |  Key code
                                              Address letter M
```

M functions can be used to form separate program blocks, or can be combined with other words (G, S, F, T) in one block.

☞ **The standard M functions of the PNC are shown in the “Annex” under the ”Overview of M functions”.**

☞ **If 2 mutually exclusive internally effective M functions are programmed in the same block, the last M function programmed will be active.**

**This refers to M functions within the following groups:**

- **M03–M05, M13–M14, M19, M103–M105, M113–M114, M119**
- **M203–M205, M213–M214, M219**
- **M40, M41–M44, M140, M141–M144, M48, M148**
- **M240, M241–M244, M248**

## 5.4.1 Subprogram calls

In addition to different G addresses and the P address (cf. page 2–8), subprograms can also be called by **8 non-modal M functions**.

You may define the actual M functions as well as the programs called by these M functions in MACODA.

The subprogram called will be executed once.

☞ **The allocation of the M instruction to the program name is machine tool manufacturer–specific and can be defined in MACODA parameters 3090 00003 and 3090 00004.**
**Please contact your systems administrator for the M functions defined as subprogram calls for your specific machine.**

Programming

As a rule, only **one** subprogram call by P, G or M may be included in one block.

If an address letter (e.g. G or M) occurs repeatedly in a block, the address calling the subprogram must be programmed at the end of the line.

**Example:** Calling a subprogram with M6

| | |
|---|---|
| `N500 M3 S500 M6` | **Correct!** |
| `N500 M6 M3 S500` | **Wrong!** (will produce runtime error) |

If both a traversing motion and a subprogram call are programmed in one block, the subprogram is called after completion of the traversing motion.

## 5.4.2 Stop processing M00, M01, M02/M30

**Program stop M00**

Effect

The NC program is interrupted and the machine movements are stopped when the block has been executed.

Program execution is restarted by ”Cycle start”. Current statuses will not be changed.

Programming

You may program ”Program stop” together with other NC functions. When all programmed functions have been executed, ”Program stop” will become effective.

**Conditional program stop M01**

Effect

The NC program will stop if the ”optional stop” interface signal is **additionally** present.

Program execution is restarted by ”Cycle start”. Current statuses will not be changed.

Programming

You may program ”Conditional program stop” together with other NC functions. When all programmed functions have been executed, ”Conditional program stop” will become effective.

**End of program M2, M02, M30**

Effect

M2, M02 or M30 will end a program.

If the program is a subprogram,

- the NC will output the corresponding auxiliary function and
- return to the calling program.

☞ **Modal statuses that have been changed in the subprogram will not be reset!**

If the program is a main program,

- the NC will reset the interface signal ”end of program” and cancel ”program running”,
- activate all statuses defined in MACODA parameter 7060 00020, ”Default status”, for an ”M30” event,
- return to the top of the main program, and
- wait for the next ”Cycle start”.

**CAUTION**
**Undefined default statuses may cause damage to the machine!**
**If certain conditions or functions must be present or performed after the end of a main program, you have to ensure that the init string for the ”M30” event has been given the proper parameters in MP 7060 00020. It must contain all functions that set the NC to the required/desired condition after an ”M30” event.**
**In this context, you should note that modal functions of a group for which no other function has been entered in the init string will remain active even after the end of program!**

☞ **For more information on ”groups”, please refer to ”Instructions and special functions”, sect. 2.1.1.**

☞ **For an overview of the G functions and their assignment to groups, please refer to ”Overview of G instructions” in the annex.**

☞ **The M functions which mutually influence each other are described in the annex section “Overview of M functions”.**

☞ **For more information on the default statuses, please refer to the ”Power–up state” section of the ”Description of functions” manual.**

Programming

You may program M2, M02 or M30 as the only instruction in a program block.

## 5.4.3 Spindle instructions

- 1st spindle group: M03, M04, M05, M13, M14, M19
- 1st spindle: M103, M104, M113, M114, M105, M119
- 2nd spindle: M203, M204, M213, M214, M205, M219

(refer to sect. 4.2.1, “Spindle functions”.

## 5.4.4 Gear ranges

**Selection of the gear range**

- 1st spindle group: M40, M41-M44
- 1st spindle: M140, M141-M144
- 2nd spindle: M240, M241-M244

**Neutral gear**

- 1st spindle group: M48
- 1st spindle: M148
- 2nd spindle: M248

(refer to sect. 4.2.2, “Gear functions”.

## 5.4.5 Tool changeM6

The M6 function initiates a tool change.

It calls a subprogram with the name freely defined in MACODA parameters 3090 00003 and 3090 00004.

For tool change, please also refer to section 5.5.

## 5.5 T address (tool selection)

Effect

With this function, you request the tool to be used in the next machining process.

The related tool number identifies the tool. It is also used for storing and calling the tool dimensions during part program execution.

The PNC can output the tool number in BCD or binary format to an automatic tool changer to initiate the magazine search run.

For machine tools with manual tool change, the programmed T word is used as a job instruction for the operating personnel or for checking coincidences between the required tool and the tool available in the spindle.

The actual tool change is initiated by M06.

The appropriate signal settings and the maximum word length for programming the tool number are defined in MACODA.

Please refer to your machine tool builder’s manual for the structure and length of the tool number.

Programming

Depends on tool management system.

**Example:**Programmed tool selection

```
N100 T123 M06        Select tool 123. Then initiate tool change with
                     M06.
N110 G0 X100 Y200    Start machining with tool 123.
N120 G1 X150 Y230
N...
N500 T234 M06        Select tool 234. Then initiate tool change with
                     M06.
```

Notes:

# A Annex

## A.1 Abbreviations

| Abbreviation | Meaning |
|---|---|
| Aux.fct | Auxiliary function |
| C: | Drive name, in this case drive C (hard disk drive) |
| ESD | electrostatic discharge<br>Abbreviation of all references to electrostatic discharge, e.g. ESD protection, ESD hazard |
| Fx | Function key with number x |
| GUI | Graphical user interface |
| HP | Main program |
| LSEC | Lead Screw Error Compensation |
| MDI | ”Manual data input” mode |
| MP | MACODA parameter |
| MSD | Machine Status Display |
| MTB | Machine tool builder (manufacturer) |
| NC, CNC | Numerical Control |
| PE | Protective Earth |
| PLC | Programmable Logic Controller |
| SK | Softkey |
| SP | Subprogram, subroutine |

## A.2 Instructions (Overview)

For more information on ”groups” (”group” column), please refer to ”Instructions and special functions”, sect. 2.1.1.

By selecting a new G function, the modal effect of a previously active function with the **same group number** is deselected and replaced by the modal effect of the new G function.

G instructions of group ”0” are **no** ”modal” functions. Therefore, they will not deselect each other mutually!

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G00 | | Linear interpolation at rapid travel (see G200) | 2 | 3–1 |
| G01 | | Linear interpolation at feedrate (see G73) | 2 | 3–3 |
| G02 | | Circular interpolation / helical interpolation clockwise | 2 | 3–4 |
| G03 | | Circular interpolation / helical interpolation counter-clockwise | 2 | 3–4 |
| G04 | | Dwell time | 0 | 3–12 |
| G05 | | Circular interpolation / helical interpolation with tangential entry | 2 | 3–13 |
| G06 | | Acceleration programming ON (see G206) | 11 | 3–15 |
| G07 | | Acceleration programming OFF (see G206) | 11 | 3–15 |
| G08 | | Path slope ON | 3 | 3–19 |
| G09 | | Path slope OFF | 3 | 3–19 |
| G10 | | Polar-coordinate programming (like G0) | 2 | 3–27 |
| G11 | | Polar-coordinate programming (like G1) | 2 | 3–27 |
| G12 | | Polar-coordinate programming (like G2) | 2 | 3–27 |
| G13 | | Polar-coordinate programming (like G3) | 2 | 3–27 |
| G14 | | KV programming ON | 9 | 3–28 |
| G15 | | KV programming OFF | 9 | 3–28 |
| G16 | 5.1 | No plane | 5 | 3–29 |
| G17 | | X/Y plane selection | 5 | 3–30 |
| G18 | | Z/X plane selection | 5 | 3–30 |
| G19 | | Y/Z plane selection | 5 | 3–30 |
| G20 | | Plane selection 2 out of 6 axes and Pole programming for polar-coordinate programming | 5 | 3–32 |
| G21 | 5.1 | Programming of axis classifications | 0 | 3–34 |
| G22 | | Table activation | 0 | 3–35 |
| G23 | | Conditional jump | 0 | 2–11 |
| G24 | | Unconditional jump | 0 | 2–11 |
| G32 | | Tapping without compensation chuck (see G532) | 0 | 3–38 |
| G33 | 7.1 | Tapping (see G533) | 2 | 3–43 |
| G34 | | Rounding of corners (with max. admissible deviation) ON (see G36) | 12 | 3–51 |
| G35 | 6.2 | Chamfer programming or rounding of corners (see G34) OFF (see G36) | | 3–52 |
| G36 | | Delete max. admissible deviation programmed for G34 | 0 | 3–51 |
| G37 | | Determination of mirror or rotating point | 22 | 3–55 |
| G38 | | Mirroring, scaling, rotating ON | 22 | 3–55 |
| G39 | | Mirroring, scaling, rotating OFF | 22 | 3–55 |

Annex

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G40 | | Cutter path compensation OFF | 41 | 3–67 |
| G41 | | Cutter path compensation to the left of the workpiece ON | 41 | 3–67 |
| G42 | | Cutter path compensation to the right of the workpiece ON | 47 | 3–67 |
| G53 | | Axis zero shift (ZS) OFF (see G153; G253) | 17 | 3–73 |
| G54..G59 | | Axis zero shift (ZS) OFF (see G154; G254) | 17 | 3–73 |
| G60 | | Programmed contour shift ON (see G67) | 20 | 3–75 |
| G61 | | In-position logic ON (see G163 and G164..166) | 13 | 3–76 |
| G62 | | In-position logic OFF (see G163) | 13 | 3–76 |
| G63 | | Feedrate 100% ON (see G66) | 7 | 3–77 |
| G64 | | Feed compensation: Cutter contact point | 42 | 3–78 |
| G65 | | Feed compensation: Cutter center point | 42 | 3–78 |
| G66 | | Feedrate 100% OFF (see G63) | 7 | 3–77 |
| G67 | | Programmed contour shift OFF (see G60) | 20 | 3–75 |
| G68 | | Outside angle compensation: Arc | 43 | 3–80 |
| G69 | | Outside angle compens.: Intersection of the equidistants | 43 | 3–80 |
| G70 | | Inch programming | 8 | 3–82 |
| G71 | | Metric programming | 8 | 3–82 |
| G73 | | Linear interpolation at feedrate with in-position logic (see G1) | 2 | 3–82 |
| G74 | | Approach reference point coordinates | 0 | 3–83 |
| G75 | | Probe input | 0 | 3–85 |
| G76 | | Traverse to machine-oriented absolute axis position | 0 | 3–95 |
| G78 | 6.2 | Compensation switchover (drill-axis switching) ON | 36 | 3–96 |
| G79 | 6.2 | Activate presetting for compensation directions | 36 | 3–96 |
| G80 | | Deactivate boring cycles G81–G86 and G184 | 1 | 3–99 |
| G81 | | Boring cycle: Drilling; retraction at rapid | 1 | 3–99 |
| G82 | | Boring cycle: Drilling; retraction at feedrate | 1 | 3–99 |
| G83 | | Boring cycle: Deep-hole drilling; retraction at rapid | 1 | 3–99 |
| G84 | | Boring cycle: Tapping with compensation chuck | 1 | 3–99 |
| G85 | | Boring cycle: Boring; retraction at rapid | 1 | 3–99 |
| G86 | | Boring cycle: Boring; retraction at feedrate | 1 | 3–99 |
| G90 | | Absolute data input 1 (see. G189) | 4 | 3–108 |
| G91 | | Incremental data input: | 4 | 3–108 |
| G92 | | Set actual value | 0 | 3–110 |
| G93 | | Time ogramming: | 6 | 3–111 |
| G94 | | Feedrate programming in mm/min | 6 | 3–112 |
| G95 | | Feedrate programming in mm/rev | 6 | 3–116 |
| G96 | 6.2 | Constant cutting speed | 35 | 3–117 |
| G97 | | Direct speed programming (see G196) | 35 | 3–117 |
| G99 | 7.3 | Spline programming | | 3–119 |
| G104 | | Dwell time in spindle revolutions | | 3–12 |
| G105 | | Zerosetting of modulo axis (linear endless axis) | 0 | 3–129 |
| G106 | | Programmable path acceleration (setting) | | 3–17 |
| G107 | | Programmable path acceleration (resetting) | | 3–17 |
| G108 | | Limited-jerk velocity control with path slope | 3 | 3–20 |

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G112 | | Consideration of the existing braking distance with active path slope OFF | 38 | 3–131 |
| G113 | | Consideration of the existing braking distance with active path slope ON | 38 | 3–131 |
| G114 | | Feed forward control ON | 10 | 3–132 |
| G115 | | Feed forward control OFF | 10 | 3–132 |
| G130 | 5.1 | Tangential tool guidance OFF | 45 | 3–134 |
| G131 | 5.1 | Tangential tool guidance ON | 45 | 3–134 |
| G134 | | Rounding of corners (with specification of rounding radius) ON | 12 | 3–51 |
| G138 | | Workpiece position compensation ON | 23 | 3–138 |
| G139 | | Workpiece position compensation OFF | 23 | 3–138 |
| G145..G845 | | External tool compensation ON (1st to 8th) | 25 | 3–140 |
| G146 | | External tool compensation OFF | 25 | 3–140 |
| G147..G847 | 4.4.1<br>(7.1) | General tool compensation ON (1st to 8th)<br>(addition of Eulerian angles and cutter position) | 52 | 3–142 |
| G148 | 4.4.1 | General tool compensation OFF | 52 | 3–142 |
| G150 | | Changing the positioning type for endless axes (MP) | 27 | 3–146 |
| G151 | | Changing the positioning type for endless axes ON | 27 | 3–146 |
| G153 | | 1st additive axis ZS OFF | 18 | 3–73 |
| G154..G159 | | 1st additive axis ZS ON | 18 | 3–73 |
| G160 | | External axis zero shift ON (no.1) (see G167) | 24 | 3–149 |
| G161 | | In-position logic at rapid travel ON | 14 | 3–150 |
| G162 | | In position logic at rapid travel OFF | 14 | 3–150 |
| G163 | | In-position logic ON at feedrate *and* rapid travel (see G61/62 and G161/162) | 13 | 3–76 |
| G164 | | In-position logic: v=0 and check for ”positioning window” | 15 | 3–151 |
| G165 | | In-position logic: v=0 and check for ”positioning window rough” | 15 | 3–151 |
| G166 | | In-position logic: v=0 without check for “positioning window” | 15 | 3–151 |
| G167 | | External axis zero shift OFF (see G160) | 24 | 3–149 |
| G168 | 5.1 | program coordinate shift ON | 46 | 3–154 |
| G169 | 5.1 | Program coordinate shift OFF | 46 | 3–154 |
| G175 | | On-the-fly measurement (initialization) (see G275) | 0 | 3–87 |
| G177 | | Torque reduction | 0 | 3–157 |
| G184 | | Boring cycle: Tapping without compensation chuck | 1 | 3–99 |
| G189 | | Absolute data input 2 (see. G90) | 4 | 3–108 |
| G192 | | Speed limitation minimum speed (see G292) | 29 | 3–158 |
| G194 | | Incremental speed programming with acceleration adaptation | 0 | 3–115 |
| G196 | | Constant cutting speed (see G97) | 35 | 3–117 |
| G200 | | Linear interpolation at rapid feed without decelerating to v=0 (see G0) | 2 | 3–2 |
| G202 | | circular movement, turning clockwise | 2 | 3–9 |
| G203 | | circular movement, turning counter-clockwise | 2 | 3–9 |
| G206 | | Acceleration programming: Storing of currently valid axis acceleration (see G6) | 0 | 3–15 |
| G228 | | Block transition without deceleration | 0 | 3–21 |
| G234 | 6.2 | Chamfer programming | | 3–52 |

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G253 | | 2nd additive axis ZS OFF | 19 | 3–73 |
| G254..G259 | | 2nd additive axis ZS ON | 19 | 3–73 |
| G260 | | External axis zero shift ON (no.2) (see G167) | 24 | 3–149 |
| G268 | 5.1 | Additive program coordinate shifts ON | 47 | 3–154 |
| G269 | 5.1 | Additive program coordinate shift OFF | 47 | 3–154 |
| G275 | | On-the-fly measurement (see G175) | 0 | 3–87 |
| G292 | | Speed limitation maximum speed (see G192) | 30 | 3–158 |
| G301 | | Oscillating axis (see G350) | 2 | 3–159 |
| G310 | 7.3 | Ramp functions: Constant-speed interpolator on | 3 | 3–163 |
| G311 | 7.3 | Ramp functions: activate acceleration interpolator with linear velocity rise | 3 | 3–163 |
| G312 | 7.3 | Ramp functions: activate deceleration interpolator with linear deceleration | 3 | 3–163 |
| G313 | 7.3 | Ramp functions: activate acceleration interpolator with sine-shaped velocity rise | 3 | 3–163 |
| G314 | 7.3 | Ramp functions: activate deceleration interpolator with sine-shaped deceleration | 3 | 3–163 |
| G315 | 7.3 | Ramp functions: activate acceleration interpolator with $\sin^2$-shaped velocity rise | 3 | 3–163 |
| G316 | 7.3 | Ramp functions: activate deceleration interpolator with $\sin^2$-shaped deceleration | 3 | 3–163 |
| G328 | | Precision programming ON | 2 | 3–166 |
| G329 | | Precision programming OFF | 2 | 3–166 |
| G350 | | Oscillating axis (initialization) (see G301) | 0 | 3–159 |
| G352 | | Inclined plane (direct programming) | 26 | 3–169 |
| G353 | | Inclined plane OFF | 26 | 3–169 |
| G354..G359 | | Call of an inclined plane table | 26 | 3–169 |
| G360 | | External axis zero shift ON (no.3) (see G167) | 24 | 3–149 |
| G374 | | Traverse to reference point | 0 | 3–84 |
| G375 | 7.1 | Measuring fixed stop | 0 | 3–172 |
| G408 | | Point-to-point movement using SHAPE | 3 | 3–22 |
| G475 | 7.1 | Move to fixed stop | 0 | 3–173 |
| G476 | 7.1 | Cancel fixed stop | 0 | 3–175 |
| G477 | 7.1 | Torque reduction fixed stop | 0 | 3–176 |
| G500 | | Look-ahead for collision monitoring | 0 | 3–184 |
| G510 | 5.1 | Integrating axes in axis groups with opt. error message | 0 | 3–177 |
| G511 | 5.1 | Integrating axes in axis groups with WAIT | 0 | 3–177 |
| G512 | 5.1 | Removing an axis from an axis group | 0 | 3–177 |
| G513 | 5.1 | Accepting the axis configuration from MACODA | 0 | 3–177 |
| G515 | 5.1 | Assigning ”Logical axis name” | 0 | 3–177 |
| G516 | 5.1 | Removing ”Logical axis name” | 0 | 3–177 |
| G517 | 6.2 | C axis off | 0 | 3–177 |
| G518 | 6.2 | C axis on | 0 | 3–177 |
| G520 | | Drive-controlled interpolation 1 | 0 | 3–182 |
| G521/G522 | | Drive-controlled interpolation 2 | 37 | 3–182 |
| G523/G524 | | Drive-controlled interpolation 3 | 0 | 3–182 |

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G532 | | Activation of tapping without compensation chuck for several spindles (see G32) | 0 | 3–39 |
| G533 | 7.1 | Special functions for tapping (see G33) | 0 | 3–49 |
| G543 | | Collision monitoring ON | 44 | 3–184 |
| G544 | | Collision monitoring OFF | 44 | 3–184 |
| G575 | | Switching NC blocks via high-speed signal | 0 | 3–89 |
| G581,G580 | 5.1 | Axis coupling | 48 | 3–188 |
| G580 | 5.1 | Disbanding a group of axes | 48 | 3–194 |
| G581 | 5.1 | Forming a group of axes | 48 | 3–191 |
| G582 | 5.1 | Creating a spline table | 0 | 3–200 |
| G594 | 6.2 | Explicit suppression of axes for feedrate computing OFF | 53 | 3–205 |
| G595 | 6.2 | Explicit suppression of axes for feedrate computing OFF | 53 | 3–205 |
| G608 | | Axis-by-axis programmable SHAPE | 3 | 3–25 |
| G610 | 5.1 | Stroke release time (default values) | 49 | 3–212 |
| G611 | 5.1 | Stroke release time (inpos window reached) | 49 | 3–212 |
| G612 | 5.1 | Stroke release time (interpolation has reached the end point) | 49 | 3–212 |
| G630 | 5.1 | Tangential tool orientation OFF | 50 | 3–215 |
| G631 | 5.1 | Tangential tool orientation ON | 50 | 3–215 |
| G660 | 5.1 | Punching/Nibbling OFF | 51 | 3–218 |
| G661 | 5.1 | Punching ON | 51 | 3–218 |
| G661 | 5.1 | Nibbling ON | 51 | 3–218 |
| G900 | | Programming SERCOS ID numbers while in a part program | 0 | 3–224 |
| G9321 | | Retraction from tapped hole (switching of spindle to position mode) | 0 | 3–40 |
| G9322 | | Retraction from tapped hole | 0 | 3–40 |

| NC functions | as of version | Title | Group | Page |
|---|---|---|---|---|
| AC(..) | | Local absolute data input (see G90/G91/G189) | | 3–108 |
| AREADEF | 5.1 | Area definition | 0 | 3–226 |
| AREAVALID | 5.1 | Activating or deactivating the control area | 0 | 3–226 |
| ASTOPA(..)<br>ASTOPO(..) | 6.2 | NC synchronization function: Channel synchronization by movement stop | | 3–236 |
| BSTOPA(..)<br>BSTOPO(..) | 6.2 | NC synchronization function: Channel synchronization by movement stop | | 3–236 |
| Coord(..) | 6.2 | 5 axis transformation | | 3–263 |
| Coord(..) | 6.2 | 6 axis transformation | | 3–267 |
| Coord(..) | 6.2 | Working range coordinate programming and axis transformation | | 3–259 |
| DIA | 6.2 | Diameter programming | | 3–231 |
| DistCtrl | 7.3 | Axis distance control for digitizing | | 3–270 |
| DC(..) | | Local setting of the positioning type for endless axes ACP(..), ACN(..) | | 3–146 |
| GOTOB | | Jump instruction with beginning of the program as the destination | | 2–11 |

Annex

| NC functions | as of version | Title | Group | Page |
|---|---|---|---|---|
| GOTOL | | Jump instruction with end of the program as the destination | | 2–11 |
| HWOCON<br>HWOCOFF | | Online correction in workpiece coordinates | | 3–276 |
| IC(..) | | Local incremental data input (see G90/G91/G189) | | 3–108 |
| JogWCSSelect | | Jogging in workpiece coordinates | | 3–278 |
| LABEL | | Destination labelling within a program | | 2–11 |
| LFPON<br>LFPOFF | | Path velocity-dependent laser power control | | 3–274 |
| MAINSP | 7.1 | Main spindle switchover | | 4–17 |
| O(..) | 6.2 | Orientation programming with angle of rotation | | 3–244 |
| OFFSTOPA<br>OFFSTOPO | 6.2 | NC synchronization function: Cancel stop conditions | | 3–236 |
| PDHSO(..) | 6.2 | Programmable position-dependent HS output | | 3–233 |
| phi, theta<br>O(..)<br>ROTAX(..) | 7.1 | Orientation programming: Vector orientation | | 3–248 |
| phi, theta<br>O(..) | 7.1 | Orientation programming: Linear orientation | | 3–246 |
| phi, theta,<br>psi<br>Ox(..), Oy(..),<br>Oz(..) O(..)<br>ROTAX(..) | 7.1 | Orientation programming: Tensor orientation for non-rotation symmetrical tools | | 3–254 |
| PREPNUM | | Limitation of the maximum number of prepared blocks | 0 | 3–235 |
| RAD | 6.2 | Radius programming | | 3–231 |
| ROTAX(..) | 6.2 | Orientation movement with programming the axis of rotation | | 3–244 |
| SPLINEDEF | 7.3 | Spline programming | | 3–119 |
| SPV(..) | 6.2 | NC synchronization function: Writing of permanent CPL variables | | 3–236 |
| SPVE(..) | 6.2 | NC synchronization function: Writing of permanent CPL variables | | 3–236 |
| TCSDEF | 7.1 | TCS definition in program coordinates | | 3–272 |
| TTOFF | | Tangential tool orientation OFF | none | 3–215 |
| TTON | | Tangential tool orientation ON | none | 3–215 |
| WAITA(..) | 6.2 | NC synchronization function: Waiting for several interface signals | | 3–236 |
| WAITO(..) | 6.2 | NC synchronization function: Waiting for one interface signal | | 3–236 |
| WPV(..) | 6.2 | NC synchronization function: Waiting for the value of a permanent CPL variable | | 3–236 |
| WPVE(..) | 6.2 | NC synchronization function: Waiting for the value of a permanent CPL variable | | 3–236 |
| WSTOPA(..)<br>WSTOPO(..) | 6.2 | NC synchronization function: Channel synchronization by movement stop | | 3–236 |

## A.3 Overview of M functions

| Function | as of version | Title | Page |
|---|---|---|---|
| M00 | | Program stop; restart execution after ”Cycle start” | 5–6 |
| M01 | | Optional stop; restart execution after ”Cycle start” | 5–6 |
| M02 | | End of program; for main and subprograms | 5–6 |
| M03 | | Spindle ON, clockwise (1st spindle/spindle group) | 4–7 |
| M04 | | Spindle ON, counterclockwise (1st spindle/spindle group) | 4–7 |
| M05 | | Spindle, Stop (1st spindle/spindle group) | 4–7 |
| M06 | | Tool change | 5–8 |
| M13 | | Spindle ON, clockwise and cooling ON (1st spindle/spindle group) | 4–7 |
| M14 | | Spindle ON, counterclockwise and cooling ON (1st spindle/spindle group) | 4–7 |
| M19 | | Spindle ”orientation” (1st spindle/spindle group) | 4–9 |
| M30 | | as M02 | 5–6 |
| M40 | | Automatic gear selection (1st spindle/spindle group) | 4–10 |
| M41–M44 | | Manual gear selection (1st spindle/spindle group) | 4–10 |
| M48 | | Neutral gear (1st spindle/spindle group) | 4–10 |
| M103 | | Spindle ON, clockwise (as M03) | 4–7 |
| M104 | | Spindle ON, counterclockwise (as M04) | 4–7 |
| M105 | | Spindle, Stop (as M05) | 4–7 |
| M113 | | Spindle ON, clockwise and cooling ON (1st spindle/spindle group) | 4–7 |
| M114 | | Spindle ON, counterclockwise and cooling ON (1st spindle/spindle group) | 4–7 |
| M119 | | Spindle ”orientation” (as M19) | 4–7 |
| M140 | | as M40 | 4–10 |
| M141–M144 | | as M41–M44 | 4–10 |
| M148 | | Neutral gear (1st spindle/spindle group) | 4–10 |
| M203 | | Spindle ON, clockwise (2nd spindle/spindle group) | 4–7 |
| M204 | | Spindle ON, counterclockwise (2nd spindle/spindle group) | 4–7 |
| M205 | | Spindle, Stop (2nd spindle/spindle group) | 4–7 |
| M213 | | Spindle ON, clockwise and cooling ON (2nd spindle/spindle group) | 4–7 |
| M214 | | Spindle ON, counterclockwise and cooling ON (2nd spindle/spindle group) | 4–7 |
| M219 | | Spindle ”orientation” (2nd spindle/spindle group) | 4–7 |
| M240 | | Automatic gear selection (2nd spindle/spindle group) | 4–10 |
| M241–M244 | | Manual gear selection (2nd spindle/spindle group) | 4–10 |
| M248 | | Neutral gear (2nd spindle/spindle group) | 4–10 |

## A.4 Overview of spindle functions

| Function | as of version | Title | Page |
|---|---|---|---|
| S.. | | Specifying the spindle speed for individual spindles or for all spindles of a spindle group | 4–12 |
| SADM | | Transferring a spindle to another channel | 4–4 |
| SCC | | Creating, modifying or disbanding groups of coupled spindles | 4–21 |
| SCD | | Setting the coupling distance for one or several slave spindles | 4–19 |
| SCSW | | Defining the synchronous mode window | 4–19 |
| SCEW | | Defining the synchronous mode error window | 4–19 |
| SCPO | | Entering an angular offset while coupling is active | 4–23 |
| SCPOWAIT | | Waiting for angular offset | 4–23 |
| SCWAIT | | Waiting for synchronous mode | 4–21 |
| SDOM | | Switching to position-controlled spindle operation | 4–16 |
| SPF, MAINSP | 7.1 | Main spindle switchover | 4–17 |
| SPGALL(0) | | Restoring spindle group default settings | 4–2 |
| SPG | | Modal assignment of spindles to spindle groups: | 4–2 |
| SPL | | Local override of spindle assignments to spindle groups | 4–2 |
| SSPG.. | | Specifying the spindle speed for a spindle group | 4–12 |

## A.5 G functions (sorted by groups)

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G04 | | Dwell time | 0 | 3–12 |
| G21 | 5.1 | Programming of axis classifications | 0 | 3–34 |
| G22 | | Table activation | 0 | 3–35 |
| G23 | | Conditional jump | 0 | 2–11 |
| G24 | | Unconditional jump | 0 | 2–11 |
| G32 | | Tapping without compensation chuck (see G532) | 0 | 3–38 |
| G36 | | Delete max. admissible deviation programmed for G34 | 0 | 3–51 |
| G74 | | Approach reference point coordinates | 0 | 3–83 |
| G75 | | Probe input | 0 | 3–85 |
| G76 | | Traverse to machine-oriented absolute axis position | 0 | 3–95 |
| G92 | | Set actual value | 0 | 3–110 |
| G105 | | Zerosetting of modulo axis (linear endless axis) | 0 | 3–129 |
| G175 | | On-the-fly measurement (initialization) (see G275) | 0 | 3–87 |
| G177 | | Torque reduction | 0 | 3–157 |
| G194 | | Incremental speed programming with acceleration adaptation | 0 | 3–115 |
| G206 | | Acceleration programming: Storing of currently valid axis acceleration (see G6) | 0 | 3–15 |
| G228 | | Block transition without deceleration | 0 | 3–21 |
| G275 | | On-the-fly measurement (see G175) | 0 | 3–87 |
| G350 | | Oscillating axis (initialization) (see G301) | 0 | 3–159 |
| G374 | | Traverse to reference point | 0 | 3–84 |
| G375 | 7.1 | Measuring fixed stop | 0 | 3–172 |
| G475 | 7.1 | Move to fixed stop | 0 | 3–173 |
| G476 | 7.1 | Cancel fixed stop | 0 | 3–175 |
| G477 | 7.1 | Torque reduction fixed stop | 0 | 3–176 |
| G500 | | Look-ahead for collision monitoring | 0 | 3–184 |
| G510 | 5.1 | Integrating axes in axis groups with opt. error message | 0 | 3–177 |
| G511 | 5.1 | Integrating axes in axis groups with WAIT | 0 | 3–177 |
| G512 | 5.1 | Removing an axis from an axis group | 0 | 3–177 |
| G513 | 5.1 | Accepting the axis configuration from MACODA | 0 | 3–177 |
| G515 | 5.1 | Assigning ”Logical axis name” | 0 | 3–177 |
| G516 | 5.1 | Removing ”Logical axis name” | 0 | 3–177 |
| G517 | 6.2 | C axis off | 0 | 3–177 |
| G518 | 6.2 | C axis on | 0 | 3–177 |
| G520 | | Drive-controlled interpolation 1 | 0 | 3–182 |
| G523/G524 | | Drive-controlled interpolation 3 | 0 | 3–182 |
| G532 | | Activation of tapping without compensation chuck for several spindles (see G32) | 0 | 3–39 |
| G533 | 7.1 | Special functions for tapping (see G33) | 0 | 3–49 |
| G575 | | Switching NC blocks via high-speed signal | 0 | 3–89 |
| G582 | 5.1 | Creating a spline table | 0 | 3–200 |
| G900 | | Programming SERCOS ID numbers while in a part program | 0 | 3–224 |

Annex

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G9321 | | Retraction from tapped hole (switching of spindle to position mode) | 0 | 3–40 |
| G9322 | | Retraction from tapped hole | 0 | 3–40 |
| G80 | | Deactivate boring cycles G81–G86 and G184 | 1 | 3–99 |
| G81 | | Boring cycle: Drilling; retraction at rapid | 1 | 3–99 |
| G82 | | Boring cycle: Drilling; retraction at feedrate | 1 | 3–99 |
| G83 | | Boring cycle: Deep-hole drilling; retraction at rapid | 1 | 3–99 |
| G84 | | Boring cycle: Tapping with compensation chuck | 1 | 3–99 |
| G85 | | Boring cycle: Boring; retraction at rapid | 1 | 3–99 |
| G86 | | Boring cycle: Boring; retraction at feedrate | 1 | 3–99 |
| G184 | | Boring cycle: Tapping without compensation chuck | 1 | 3–99 |
| G00 | | Linear interpolation at rapid travel (see G200) | 2 | 3–1 |
| G01 | | Linear interpolation at feedrate (see G73) | 2 | 3–3 |
| G02 | | Circular interpolation / helical interpolation clockwise | 2 | 3–4 |
| G03 | | Circular interpolation / helical interpolation counter-clockwise | 2 | 3–4 |
| G05 | | Circular interpolation / helical interpolation with tangential entry | 2 | 3–13 |
| G10 | | Polar-coordinate programming (like G0) | 2 | 3–27 |
| G11 | | Polar-coordinate programming (like G1) | 2 | 3–27 |
| G12 | | Polar-coordinate programming (like G2) | 2 | 3–27 |
| G13 | | Polar-coordinate programming (like G3) | 2 | 3–27 |
| G33 | 7.1 | Tapping (see G533) | 2 | 3–43 |
| G73 | | Linear interpolation at feedrate with in-position logic (see G1) | 2 | 3–82 |
| G200 | | Linear interpolation at rapid feed without decelerating to v=0 (see G0) | 2 | 3–2 |
| G202 | | circular movement, turning clockwise | 2 | 3–9 |
| G203 | | circular movement, turning counter-clockwise | 2 | 3–9 |
| G301 | | Oscillating axis (see G350) | 2 | 3–159 |
| G328 | | Precision programming ON | 2 | 3–166 |
| G329 | | Precision programming OFF | 2 | 3–166 |
| G08 | | Path slope ON | 3 | 3–19 |
| G09 | | Path slope OFF | 3 | 3–19 |
| G108 | | Limited-jerk velocity control with path slope | 3 | 3–20 |
| G310 | 7.3 | Ramp functions: Constant-speed interpolator on | 3 | 3–163 |
| G311 | 7.3 | Ramp functions: activate acceleration interpolator with linear velocity rise | 3 | 3–163 |
| G312 | 7.3 | Ramp functions: activate deceleration interpolator with linear deceleration | 3 | 3–163 |
| G313 | 7.3 | Ramp functions: activate acceleration interpolator with sine-shaped velocity rise | 3 | 3–163 |
| G314 | 7.3 | Ramp functions: activate deceleration interpolator with sine-shaped deceleration | 3 | 3–163 |
| G315 | 7.3 | Ramp functions: activate acceleration interpolator with $\sin^2$-shaped velocity rise | 3 | 3–163 |
| G316 | 7.3 | Ramp functions: activate deceleration interpolator with $\sin^2$-shaped velocity decrease | 3 | 3–163 |
| G408 | | Point-to-point movement using SHAPE | 3 | 3–22 |
| G608 | | Axis-by-axis programmable SHAPE | 3 | 3–25 |

Annex

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G90 | | Absolute data input 1 (see. G189) | 4 | 3–108 |
| G91 | | Incremental data input: | 4 | 3–108 |
| G189 | | Absolute data input 2 (see. G90) | 4 | 3–108 |
| G16 | 5.1 | No plane | 5 | 3–29 |
| G17 | | X/Y plane selection | 5 | 3–30 |
| G18 | | Z/X plane selection | 5 | 3–30 |
| G19 | | Y/Z plane selection | 5 | 3–30 |
| G20 | | Plane selection 2 out of 6 axes and Pole programming for polar-coordinate programming | 5 | 3–32 |
| G93 | | Time ogramming: | 6 | 3–111 |
| G94 | | Feedrate programming in mm/min | 6 | 3–112 |
| G95 | | Feedrate programming in mm/rev | 6 | 3–116 |
| G63 | | Feedrate 100% ON (see G66) | 7 | 3–77 |
| G66 | | Feedrate 100% OFF (see G63) | 7 | 3–77 |
| G70 | | Inch programming | 8 | 3–82 |
| G71 | | metric programming | 8 | 3–82 |
| G14 | | KV programming OFF | 9 | 3–28 |
| G15 | | KV programming OFF | 9 | 3–28 |
| G114 | | Feed forward control ON | 10 | 3–132 |
| G115 | | Feed forward control OFF | 10 | 3–132 |
| G06 | | Acceleration programming ON (see G206) | 11 | 3–15 |
| G07 | | Acceleration programming OFF (see G206) | 11 | 3–15 |
| G34 | | Rounding of corners (with max. admissible deviation) ON (see G36) | 12 | 3–51 |
| G134 | | Rounding of corners (with specification of rounding radius) ON | 12 | 3–51 |
| G61 | | In-position logic ON (see G163 and G164..166) | 13 | 3–76 |
| G62 | | In-position logic OFF (see G163) | 13 | 3–76 |
| G163 | | In-position logic ON at feedrate *and* rapid travel (see G61/62 and G161/162) | 13 | 3–76 |
| G161 | | In-position logic at rapid travel ON | 14 | 3–150 |
| G162 | | In position logic at rapid travel OFF | 14 | 3–150 |
| G164 | | In-position logic: v=0 and check for ”positioning window” | 15 | 3–151 |
| G165 | | In-position logic: v=0 and check for ”positioning window rough” | 15 | 3–151 |
| G166 | | In-position logic: v=0 without check for “positioning window” | 15 | 3–151 |
| G53 | | Axis zero shift (ZS) OFF (see G153; G253) | 17 | 3–73 |
| G54..G59 | | Axis zero shift (ZS) OFF (see G154; G254) | 17 | 3–73 |
| G153 | | 1st additive axis ZS OFF | 18 | 3–73 |
| G154..G159 | | 1st additive axis ZS ON | 18 | 3–73 |
| G253 | | 2nd additive axis ZS OFF | 19 | 3–73 |
| G254..G259 | | 2nd additive axis ZS ON | 19 | 3–73 |
| G60 | | Programmed contour shift ON (see G67) | 20 | 3–75 |
| G67 | | Programmed contour shift OFF (see G60) | 20 | 3–75 |
| G37 | | Determination of mirror or rotating point | 22 | 3–55 |
| G38 | | Mirroring, scaling, rotating ON | 22 | 3–55 |
| G39 | | Mirroring, scaling, rotating OFF | 22 | 3–55 |

Annex

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G138 | | Workpiece position compensation ON | 23 | 3–138 |
| G139 | | Workpiece position compensation OFF | 23 | 3–138 |
| G160 | | External axis zero shift ON (no.1) (see G167) | 24 | 3–149 |
| G167 | | External axis zero shift OFF (see G160) | 24 | 3–149 |
| G260 | | External axis zero shift ON (no.2) (see G167) | 24 | 3–149 |
| G360 | | External axis zero shift ON (no.3) (see G167) | 24 | 3–149 |
| G145..G845 | | External tool compensation ON (1st to 8th) | 25 | 3–140 |
| G146 | | External tool compensation OFF | 25 | 3–140 |
| G352 | | Inclined plane (direct programming) | 26 | 3–169 |
| G353 | | Inclined plane OFF | 26 | 3–169 |
| G354..G359 | | Call of an inclined plane table | 26 | 3–169 |
| G150 | | Changing the positioning type for endless axes (MP) | 27 | 3–146 |
| G151 | | Changing the positioning type for endless axes ON | 27 | 3–146 |
| G192 | | Speed limitation minimum speed (see G292) | 29 | 3–158 |
| G292 | | Speed limitation maximum speed (see G192) | 30 | 3–158 |
| G96 | 6.2 | Constant cutting speed | 35 | 3–117 |
| G97 | | Direct speed programming (see G196) | 35 | 3–117 |
| G196 | | Constant cutting speed (see G97) | 35 | 3–117 |
| G78 | 6.2 | Compensation switchover (drill-axis switching) ON | 36 | 3–96 |
| G79 | 6.2 | Activate presetting for compensation directions | 36 | 3–96 |
| G521/G522 | | Drive-controlled interpolation 2 | 37 | 3–182 |
| G112 | | Consideration of the existing braking distance with active path slope OFF | 38 | 3–131 |
| G113 | | Consideration of the existing braking distance with active path slope ON | 38 | 3–131 |
| G40 | | Cutter path compensation OFF | 41 | 3–67 |
| G41 | | Cutter path compensation to the left of the workpiece ON | 41 | 3–67 |
| G64 | | Feed compensation: Cutter contact point | 42 | 3–78 |
| G65 | | Feed compensation: Cutter center point | 42 | 3–78 |
| G68 | | Outside angle compensation: Arc | 43 | 3–80 |
| G69 | | Outside angle compens.: Intersection of the equidistants | 43 | 3–80 |
| G543 | | Collision monitoring ON | 44 | 3–184 |
| G544 | | Collision monitoring OFF | 44 | 3–184 |
| G130 | 5.1 | Tangential tool guidance OFF | 45 | 3–134 |
| G131 | 5.1 | Tangential tool guidance ON | 45 | 3–134 |
| G168 | 5.1 | program coordinate shift ON | 46 | 3–154 |
| G169 | 5.1 | Program coordinate shift OFF | 46 | 3–154 |
| G42 | | Cutter path compensation to the right of the workpiece ON | 47 | 3–67 |
| G268 | 5.1 | Additive program coordinate shifts ON | 47 | 3–154 |
| G269 | 5.1 | Additive program coordinate shift OFF | 47 | 3–154 |
| G581,G580 | 5.1 | Axis coupling | 48 | 3–188 |
| G580 | 5.1 | Disbanding a group of axes | 48 | 3–194 |
| G581 | 5.1 | Forming a group of axes | 48 | 3–191 |
| G610 | 5.1 | Stroke release time (default values) | 49 | 3–212 |
| G611 | 5.1 | Stroke release time (inpos window reached) | 49 | 3–212 |

Annex

| G function | as of version | Title | Group | Page |
|---|---|---|---|---|
| G612 | 5.1 | Stroke release time (interpolation has reached the end point) | 49 | 3–212 |
| G630 | 5.1 | Tangential tool orientation OFF | 50 | 3–215 |
| G631 | 5.1 | Tangential tool orientation ON | 50 | 3–215 |
| G660 | 5.1 | Punching/Nibbling OFF | 51 | 3–218 |
| G661 | 5.1 | Punching ON | 51 | 3–218 |
| G661 | 5.1 | Nibbling ON | 51 | 3–218 |
| G147..G847 | 4.4.1 (7.1) | General tool compensation ON (1st to 8th) (addition of Eulerian angles and cutter position) | 52 | 3–142 |
| G148 | 4.4.1 | General tool compensation OFF | 52 | 3–142 |
| G594 | 6.2 | Explicit suppression of axes for feedrate computing OFF | 53 | 3–205 |
| G595 | 6.2 | Explicit suppression of axes for feedrate computing OFF | 53 | 3–205 |
| G35 | 6.2 | Chamfer programming or rounding of corners (see G34) OFF (see G36) |  | 3–52 |
| G99 | 7.3 | Spline programming |  | 3–119 |
| G104 |  | Dwell time in spindle revolutions |  | 3–12 |
| G106 |  | Programmable path acceleration (setting) |  | 3–17 |
| G107 |  | Programmable path acceleration (resetting) |  | 3–17 |
| G234 | 6.2 | Chamfer programming |  | 3–52 |

## A.6 Index

Annex

Notes:

Rexroth
Bosch Group

Bosch Rexroth AG
Electric Drives and Controls
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr a. Main, Germany
info@boschrexroth.de
www.boschrexroth.de

Printed in Germany
DOK-PNC***-DIN*PROG***-AW11-EN-P