BECKHOFF New Automation Technology

Manual | EN

TF5200 | TwinCAT 3 CNC

Programming manual

12/18/2023 | Version: 1.238
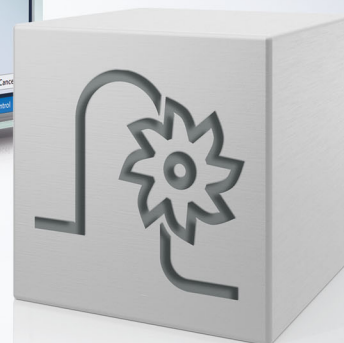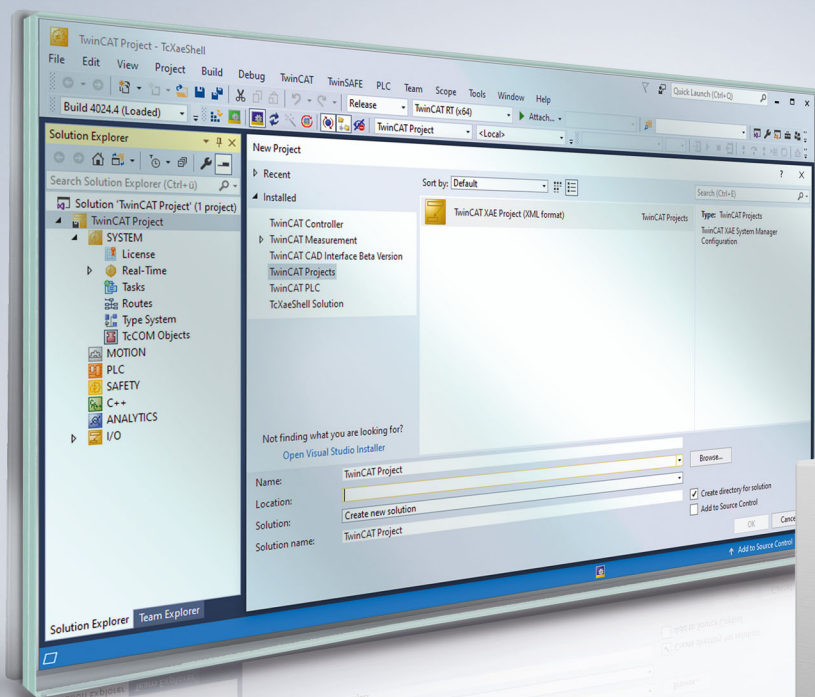
Preface

Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT technology is patent protected, in particular by the following applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

General and safety instructions

Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

Icons in explanatory text

1. Indicates an action.

⇨ Indicates an action statement.

⚠ DANGER
Acute danger to life!
If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.

⚠ CAUTION
Personal injury and damage to machines!
If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.

NOTICE
Restriction or error
This icon describes restrictions or warns of errors.

i **Tips and other notes**

This icon indicates information to assist in general understanding or to provide additional information.

General example

Example that clarifies the text.

NC programming example

Programming example (complete NC program or program sequence) of the described function or NC command.

i **Specific version information**

Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.

Table of contents

List of figures

1 Brief description

The controller processes syntax according to DIN 66025 or customary design and syntax elements according to extensions:

- Text-based program names

- Comprehensive parameter calculation (local and global parameters)

- Access to internal control data such as positions, measurement and tool data, zero offsets etc., by plain text designations (V.A.name, V.G.name)

- Definition of plain texts to designate free parameters in the NC program (V.L.*name*, V.S.*name*, V.P.*name*)

- Control block statements based on the "C" programming language, for example:
  - Conditional jumps: $IF, $ELSEIF, $ELSE, $ENDIF, $SWITCH, $CASE, $DEFAULT,$ENDSWITCH, $BREAK
  - Counting loops: $FOR, $ENDFOR, $CONTINUE, $BREAK
  - Loops with running condition: $WHILE, $ENDWHILE, $CONTINUE, $BREAK
  - Loops without running condition: $DO, $ENDDO, $CONTINUE, $BREAK
  - Jumps within the same NC program level: $GOTO

- Distinction between global (accessible by all main programs) and local subroutines (accessible only from the associated main program)

- Mathematical expressions, e.g.:
  - Basic standard arithmetic operations: + , - , * , / , ** , MOD
  - Numerical functions such as ABS,SQR,SQRT,EXP,LN,DEXP,LOG
  - Trigonometric functions such as SIN,COS,TAN,ASIN,ACOS,ATAN
  - Conversion functions such as INT,FRACT,ROUND

- Technology commands with configurable effect of each function:
  - Additional functions (MO....M65535)
  - Auxiliary functions (H0....H65535)
  - Spindle functions (S, M3, M4, M5, M19)
  - Tool functions (T, D)

- Processing of coordinate notation A, B, C, ..., U, V, W, if they are not used in any other way, and/or programming of strings (e.g. X_ACHSE, SPINDEL_1 ...).

> **ⓘ** The Appendix contains a <u>complete list of NC commands [▶ 787]</u>.

Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

2 Basic principles of programming

2.1 Typographical representation of syntax

The convention below applies to the typographical representation of NC command syntax described in this documentation:

Bold	Obligatory syntax elements of NC command.
[ ]	Optional non-recurrent occurrence
{ }	Optional multiple occurrence
|	Alternative between several symbols ("or")
=..	Mathematical expression [▶ 31] or string. The equals sign is optional.

2.2 Character and number formats

2.2.1 Character set and file format

The decoder used processes NC control data in ASCII format. NC control data can be generated using a programming system integrated in the operating system or in an editor.

The CNC processes the following characters:

Characters =	letters	and/or
	digits	and/or
	special characters	and/or
	control characters	

letters	{ A B C ... Z a b c ... z }
numerals	{ 1 2 3 ... 9 0 }
special characters	{ ! @ # $ % & * - + _ = ~ ( ) [ ] , ; : " < > / \ ? ' }
Control character = HT	Horizontal tabulator TAB
SP Space	SPACE
CR Carriage Return	CARRIAGE RETURN
LF Line Feed	LINE FEED

Restrictions with NC filenames

The following characters are not permitted for filenames:

special characters	("] # $; ,
control characters	TAB, CARRIAGE RETURN, LINE FEED

BECKHOFF

2.2.2   Numerical input

The input field for numerical values is limited by the internal numeric display.

This numeric display allows motion paths in the range of 200 m at a resolution of 0.1 µm.

By changing the internal conversion factors in a specification file, the input of positions, feeds, etc., may also be used in different units than mm or inch.

Numerical values as integers or decimal point inputs:

- Decimals are generally separated by a "." whereby leading zeroes may be omitted.
- Depending on the configuration and programming, inputs for lengths and position values are given in millimetres or inches; angles are entered in degrees [°] or gons [gon].

Input format:

Values: [µm]	Inputs: [mm]
0,1	0.0001 or .0001
1	0.001 or .001
10	0.01 or .01
100	0.1 or .1
1000	1.0 or .1
10000	10.0 or .10
100000	100.0 or .100
1000000	1000.0 or .1000

Numerical inputs as hexadecimal numbers:

- These numbers are enclosed in '...' and start with the character combination 16#, 0x or alternatively H. Following leading zeros or blanks are ignored.
- The characters A to F or a to f can be used for an additional six numerals.

Format:'16#<A...F, a..f, 0..9>**'** oder **'0x**<A...F, a..f, 0..9>**'**or **'H**<A...F, a..f, 0..9>**'**

Input format:

'16#FA1B'	is equivalent to decimal value 64027
'0x0ED2'	is equivalent to decimal value 3794
'H1869f'	is equivalent to decimal value 99999

| Version: 1.238 |

Numerical inputs as binary numbers (dual numbers):

- These numbers are enclosed in '...' and start with the character combination 2#, 02# or alternatively B. Following leading zeros or blanks are ignored.
- The two digits are displayed as zero (0) and one (1).

Format: '2#<0..1>**' or '02#**<0..1>**' or 'B**<0..1>**'**

Input format:

'2#1010011'	is equivalent to decimal value 83
'02#010011'	is equivalent to decimal value 19
'B11101010'	is equivalent to decimal value 234

Numerical inputs as octal numbers:

- These numbers are enclosed in '...' and start with the character combination 8# or 08#. Following leading zeros or blanks are ignored.
- The digits 0 to 7 are used for representation.

Format: '8#<0..7>**' or '08#**<0..7>**'**

Input format:

'8#12345'	is equivalent to decimal value 5349
'08#0107302'	is equivalent to decimal value 36546

BECKHOFF

2.3 Structure of NC control data: NC programs

Definitions for understanding:

NC programs are part of the NC control data in addition to tool data, zero offset data etc. NC programs describe the flow of machining processes.

Separating characters indicate the end of a sequence of characters (character string). The decoder evaluates the following separating characters:

CR	LF	ETX	TAB	\0	Blank	(	;	"	]	,	#	$

i These characters may not be used in file names or as part of % program names by the NC main program or subroutines (local, global); otherwise an error is output.

Comments contain non-decodable ASCII information which must be placed between the characters "(" and ")" or after the character ";". If an additional "(" is found within a comment, a matching ")" is expected. The block end character and the file end character end the comments.

Mathematical expressions [▶ 31] are composed of numerals, parameters, operators, functions, etc. They are evaluated by an integrated calculation function.

Examples: "100", "100+20", "100+P10", "100+PP10", "100*[2+P3]", "DEXP[2]", "100+SIN[P10+PP20]".

An NC program consists of:

- Comments
- Definitions of local subroutines, consisting of:
  - String "%L" followed by a subroutine name
  - A number of NC program blocks
  - A code identifying subroutine end (M17 or M29)
- Definition of the main program, consisting of:
  - String "%" followed by a main program name
  - A number of NC blocks
  - A code identifying main program end (M02 or M30).

i If a file outside the comments contains a first character that is neither a separating character nor a "%", the character is evaluated as the first character of an unnamed main program. It also means that no block numbers may be programmed in front of "%".

2.4 NC block structure

An NC block consists of a

- block number (optional)
- a number of words (NC commands)
- block end identifier

An NC block has a maximum length of 4000 characters.

The use of # commands (see Special functions [▶ 298]) excludes the programming of other words in the NC block (except for the block number).

As a rule, each block begins with a block number consisting of an N character followed by a mathematical expression. This expression is mapped in the display data and rounded off as an integer.

The use of block sequences (SEQUENCE) and program jumps ($GOTO) requires the unambiguous and ascending programming of block numbers.

Otherwise, the block number is of no significance for program flow. In this case the block number does not even need to be programmed in ascending order.

Example of a block structure:

```
N...  G...  X...  Y...  F...  S...  M...  <LF>
                                          │
                                          └── Block end ident. code
                                    │
                                    └── Machine function
                                        (e.g. spindle right)
                              │
                              └── Spindle speed
                        │
                        └── Feedrate
                  │
                  └── Position Y-axis
            │
            └── Position X-axis
      │
      └── Motion function
│
└── Block number
```
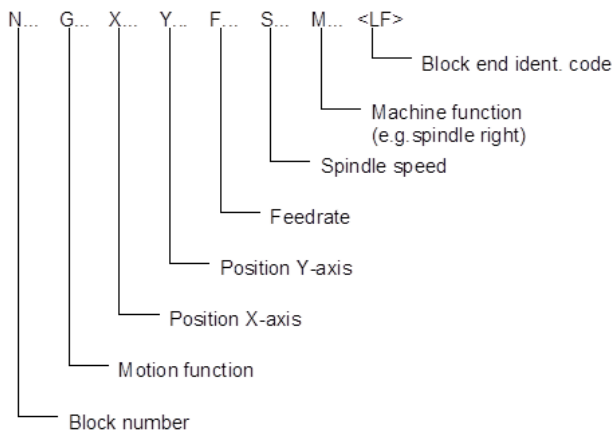
NC commands conforming to DIN 66025 need not be compulsorily separated by spaces or tabs. When programming text commands deviating from DIN (control block statements, special functions, etc.), the syntax requires separating characters which are also useful to structure an NC program.

Examples of an NC program structure:

without numbering	Partial numbering	Complete numbering
% 100	% 100	% 100
"Block 1"	N10 "Block 1"	N10 "Block 1"
"Block 1"	"Block 2"	N20 "Block 2"
"Block 1"	N20 "Block 3"	N30 "Block 3"
.	"Block 4"	N40 "Block 4"
.	.	.
.	.	.
M30	M30	N700 M30

Words must be distinguished according to their significance into:

- Geometrical information (e.g. positions),
- Technological information (e.g. spindle speed, feedrate, clockwise spindle rotation),
- Information on program flow control (called control blocks such as counting loops),
- Arithmetic information (e.g. calculation of a variable, parameter calculation).

Several words may be in a block (exception: special commands from Section Special commands [▶ 298]) whereby the processing sequence of control data within the block is defined by the controller. The programmer can then enter the individual words of an NC block in any order without this having any effect on processing. This programming manual contains special notes to point out exceptions.

The block end identifier normally consists of a combination of the control characters "CR" and "LF".

2.4.1   Skipping NC blocks '/'

Skipped blocks permit optional processing stages such as measuring loops, test blocks or dummy stages within an NC program.

2.4.1.1   Standard skipping

Specific NC blocks can be skipped by prefixing them with a "/" character. The controller ignores NC blocks if the function "Skip block" is activated before main program start by a BOOL command on the operating console (HMI) or by the PLC.

```
/ N3412 X100 …
```

In Builds up to V3.01.3020.01, any change in skip settings while an NC program is active only becomes effective at the next main program start. The extended skipping [▶ 27] function is then available in higher Builds.

```
/ N..
```

2.4.1.2   Extended skipping (skip levels)

ⓘ   This function is available as of Build V3.01.3021.00.

By combining a slash "/" and a numeral, you can set up to 10 different skip levels in the NC program (e.g. /5 N3412...). The skip levels are activated on the operating console (HMI) or by the PLC by a 32 bit command before main program start. Multiple skip levels can be activated at the same time.

In the Extended Skipping function, changes in skipping settings take effect immediately while the NC program is active. Defined break points can be implemented, e.g. by M functions followed by #FLUSH WAIT, to ensure that these skipping setting changes are safely accepted and become effective in the NC program.

For more information see [FCT-M6].

For compatibility reasons the skip level without a number "/" and "/1" have the same meaning.

```
/<1 - 10> N..
```

Extended skipping

```
%skip_levels
N05 G0 X0 Y0 Z0
/1 N10 G74 X1 Y1 Z1
N20 G01 F1000 X10 Y10
/2 N30 Z3
N40 X-1 Y-2
...
/10 N90 #CS ON [0,0,0,0,0,45]
N95 X30 Z50
/ N99 G92 X0
N999 M30
```

2.4.2   Block-specific comments

Explanatory comments can be inserted at almost any position in an NC program, even at the start of a program. Comments have no effect on the processing of an NC program.

Comments start with a "(". If the comment is extended up to block end, it is sufficient to separate it by "(". For comments within an NC block the corresponding ")" character is also required.

Alternatively, a comment can start with a semicolon ";". This comment type always extends up to block end.

Comments may be of any length. However, the maximum number of characters per block may not be exceeded.

It is possible to nest comments.

Comments in the NC code

```
% 100 (comment in full brackets)


N200 ... (comment only with open bracket
N300 (comment (nested comment))

N500 X10 (comment in block) Y20

N700 ... ;Comment after semicolon

N999 M30
```

> ● Comments on any number of program lines (see <u>Cross-block valid comments [▶ 326]</u>) may be made
> ⓘ using the special commands #COMMENT BEGIN and #COMMENT END.

2.4.3   Line break in NC block '\'

With some NC commands the syntax sequence of an NC block may be separated into two or several lines by inserting line breaks '\'' to improve clarity and readability. The line break character '\' may only be placed after complete syntax terms.

Separated blocks may only be used to continue an associated syntax sequence.

i | **EXCEPTION:**
At the beginning of a separated block, a new block numeral "N..." is permissible.

In general the line break '\' can be used in brackets for:

- Axis-specific programming X[...]
- Spindle-specific programming S[...]
- PLCopen commands S[MC_...]
- Cycle programming L CYCLE [...]
- Declaration of self-defined variable arrays

And in general for:

- DIN/ISO programming G01 X10 ...

NOTICE
NC commands from the group of additional functions (# commands) may **not** be split by a line break. A separate note is provided for exceptions!

BECKHOFF

Line break in NC block '\'

```
N10 X[INDP_ASYN G90 POS=100 G01 FEED=2500 \
N20 SLOPE_TYPE=STEP M23 M56 H78]
N…

N10 S[M4 REV5000 M11] S2[M3 REV5000 M34] S2[REV1000 M3 POS=45 M19 \
N20 M11 M12 H56]
N…

N10 S[MC_MoveAbsolute Position=133 Velocity=1000 Acceleration=500 \
N20 Deceleration=600 Jerk=200 Direction=2]
N…

N… L CYCLE [NAME=pocketmill.cyc @P1=100 @P2=80 @P3=5 @P5=15 \
@P6=80 @P7=60 @P8=10 @P9=65 @P10=50 @P11=3 @P12=1 \
@P13=2 @P14=1 @P15=-1 @P16=40 @P17=3]

N10 G60 G90 F1000 G01 \
N20 X100 Y150 Z250
N…

#VAR
V.P.ARRAY_1[3][6] = [10,11,12,13,14,15, \
                     20,21,22,23,24,25, \
                     30,31,32,33,34,35 ]
#ENDVAR
```

2.5 Word structure

A word consists of address letters, arithmetical expressions and texts <string>. The meaning of the individual address letters is described in the following sections. An overview is provided in "Assigned address characters [▶ 42]".

2.5.1 Mathematical expressions

Consisting of:

- Numerical values broken down into:
    - Integers <int> and
    - Decimal numbers <double>
- Arithmetic expressions <expr> are composed of:
    - numerical values
    - operators
    - functions
    - parameters
    - Variablen
    - macros

Example: [[sin["MAX_ANZ" * 30.00] + P2] / V.G.BLOCK_NR]

2.5.1.1 Integers <int>

Integers are specified without a decimal point. The internally permissible value range corresponds to "long integer" variables in the C programming language, i.e. the range from $-2.14*10^9$ to $+2.14*10^9$.

If internal calculation uses the unit 0.1 µm, the numerical range when values are entered in mm is between $-2.14*10^5$ and $+2.14*10^5$. This corresponds to a motion path of over 400 metres. Due to an internal limitation in the position controller, the motion path is limited to half, i.e. to just over 200 metres.

Where negative numerals are preceded by "-", positive numerals do not require the prefix "+".

2.5.1.2 Decimal numbers <double>

Decimal values are specified with a decimal point or alternatively, in exponential notation, with a lowercase 'e' (also referred to as scientific notation). The value range corresponds to integer numerals. If internal calculation uses the unit 0.1 µm, input values in mm are then provided with 4 digits after the decimal point.

Where negative numerals are preceded by "-", positive numerals do not require the prefix "+".

Examples of decimal numbers:

| 123.3456 | 0.6789 | .6789 | -345.56 | +78.987 |
| 12e5 | +2.5e6 | -4e7 | -523.6e-3 |

BECKHOFF

2.5.1.3   Arithmetic expressions <expr>

The usual calculation rules apply to handling arithmetical expressions:

- Order-of-operations rule
- the parenthesis rule, whereby square brackets "[ ]" must be used. Round parenthesis "(...)" are for comments.

Parameters are often used in arithmetical expressions. The notation of parameters is:

- P followed by an integer, e.g. P12.

Example of an arithmetic expression:

P5 = [[sin[R1*30.00] + P2] / P5]

Macros (strings) may be assigned to arithmetical expressions and parts of them.

A macro name leads to a macro content which is analysed. Recursive handling is also possible.

Macro names must be placed in quotation marks. When decoded, the notation is case-sensitive (uppercase/lowercase).

Nested strings are identified by a preceding '\' before the double quotation marks. Make sure that complete nesting levels are always grouped in a macro, i.e. adding ´[´ at the start and ´]´ at the end of macro content should have no effect on the result of the mathematical expression.

Nested macros

```
Correct:

N10 "STRING1" = "COS[\"STRING2\"]"
N20 "STRING2" = "5 * 12"
N30 "STRING3" = "SIN[89.5 + \"STRING1\"]"
N40 X[-2 * "STRING1" + "STRING2" + "STRING3"] (Move to X60)

M30

Wrong:

Only complete nesting levels may be compiled
in the string.
N10 "STRING1" = "COS["
N20 "STRING2" = "90]"
N30 "STRING3" = " \"STRING1\" \"STRING2\" "
```

Macros defined in the NC program are valid program global.

Section "Macroprogramming (# INIT MACRO TAB) [▶ 672] " describes how to program macros outside mathematical expressions.

Overview of all available calculation operations:

Basic types of calculation:

Addition	+	P1 = P2 + P3 + 0.357
Subtraction	-	P1 = P2 - 0.031
Multiplication	*	P1 = P2 * [P3 + 0.5]
Division	/	P1 = P2 * P3 / [P5 + P6]
Exponential calculation	**	P1 = 2**P3 (2 to the power P3)
Modulo calculation	MOD	P1 = 11 MOD 3 (-> 2)

Numerical functions:

Absolute value formation	ABS [..]	P1 = ABS [P2 - P4]
Squaring	SQR [..]	P1 = SQR [P2] + SQR [P3]
Square root	SQRT [..]	P1 = SQRT [SQR[P2]+SQR[P3]]

e function	EXP [..]	`P1 = EXP [P2 * P4]`
Natural logarithm	LN [..]	`P1 = LN [P2] + LN [P3]`
To the power of ten	DEXP [..]	`P1 = DEXP [P2]`
Common logarithm	LOG [..]	`P1 = LOG [P2]`

> **NOTICE**
>
> In the case of LN, LOG and SQRT the argument must always be greater than 0!

Bit operators:

AND operation	&	`P1 = P2 & P3`
OR operation	|	`P1 = P2 | P3`
Exclusive OR	^	`P1 = P2 ^ P3`
Complement	INV[..]	`P1 = INV[P2]`

> **NOTICE**
>
> The operands may be any positive mathematical expression or number within the range 0 ... 2^32-1 (UNS32). Negative expressions or numerals are not allowed. Floating point numbers are converted into integers.
>
> The result of a bit operation is always within the range of 0... 2^32-1 (UNS32).

Logic operators:

AND operation	&& / AND	`P1 = P2 && P3` `P1 = P2 AND P3`
OR operation	|| / OR	`P1 = P2 || P3` `P1 = P2 OR P3`
Exclusive OR operation	XOR	`P1 = P2 XOR P3`

NOT operation	NOT[..]	`P1 = NOT[P2]`
		`P1 = NOT[1] (P1 = 0)`
		`P1 = NOT[0.5] (P1 = 0)`
		`P1 = NOT[0.49] (P1 = 1)`
		`P1 = NOT[0] (P1 = 1)`

NOTICE
Operands may be any positive mathematical expression or numeral. Negative expressions or numerals are not allowed. A floating point numeral is evaluated as TRUE (1) if **its value is > or = 0.5.**

Comparison operators:

Loops (Section Statements for influencing NC program flow [▶ 225]) require comparison expressions. Verification can be conducted as follows:

Equality	==	`$IF P1 == 10`
Inequality	!=	`$IF P1 != 10`
Greater than or equal to	>=	`$IF P1 >= 10`
Less than or equal to	<=	`$IF P1 <= 10`
Less than	<	`$IF P1 < 10`
Greater than	>	`$IF P1 > 10`

Operator priorities:

The priorities of available operators are listed in descending order. 10 is the highest and 1 is the lowest priority.

Priority	Operator	Description
10	**	Power
9	*, /	Multiplication, division
8	+, -	Addition, subtraction
7	&	Bitwise AND
6	^	Bitwise exclusive OR
5	|	Bitwise OR
4	<=, >=, ==, <, >, !=	Comparison operators
3	&&, AND	Logic AND
2	XOR	Logic exclusive OR
1	||, OR	Logic OR

Possible truth values are:

True	TRUE	`$IF V.A.MERF.X == TRUE`
Not true	FALSE	`$WHILE V.G.WZ[2].OK == FALSE`

NOTICE
Processing truth values:
For TRUE, the value 1 is used in the controller.
For FALSE, the value 0 is used in the controller.

Trigonometric functions (angles specified in degrees):

Sine	SIN [..]	`P1 = SIN [P2]`
Cosine	COS [..]	`P1 = COS [P2]`
Tangent	TAN [..]	`P1 = TAN [P2]`
Cotangent	COT [..]	`P1 = COT [P2]`
Arcsine	ASIN [..]	`P1 = ASIN [P2]`
Arccosine	ACOS [..]	`P1 = ACOS [P2]`
Arctangent	ATAN [..]	`P1 = ATAN [P2]`
Arctangent with 2 arguments	ATAN2 [*y,x*]	`P1 = ATAN2[100,100]` `(-> result is 45°)`
Arc cotangent	ACOT [..]	`P1 = ACOT [P2]`

NOTICE
With the numerical functions ASIN and ACOS, the argument must always be between -1 and +1.
For the numerical function TAN, the argument should not assume the values -90, 90, 270 ... degrees.
For the numerical function COT, the argument should not assume the values... -180, 0, 180 ... degrees.
The numerical function ATAN2 results in x!=0 for the angle of a position relative to the X axis in the correct quadrant.
Special case: For ATAN2[0.0] (x = 0 and y = 0), the result is always 0.

Transformation functions:

Remove places after decimal point	INT [..]	P1 = INT [123.567] (P1 = 123)
Remove integer part	FRACT [..]	P1 = FRACT [123.567] (P1 = 0.567)
Round up to integer	ROUND [..]	P1 = ROUND [77.5] (P1 = 78) P1 = ROUND [45.4] (P1 = 45)
Round up	CEIL [..]	P1 = CEIL [8.3] (P1 = 9)
Round down	FLOOR [..]	P1 = FLOOR [8.7] (P1 = 8)

Constants:

3.141592654 (π)	PI	P2 = 2*PI (P2 = 6.283185307)

Special functions:

Check for existence of variables (V.P., V.L., V.S., V.E.) / parameters / M/H functions / macros	EXIST [<*Variable/ Parameter/ M function/ H function/ Macro>*]	$IF EXIST[V.P.MYVAR] == TRUE $IF EXIST[V.L.MYARRAY[0]] == TRUE * $IF EXIST[P1] != TRUE $IF EXIST[M55] == TRUE $IF EXIST[H20] == TRUE $IF EXIST["Macro1"] == TRUE *For arrays with valid indices!
Determine the size of an array Dimension of variables (V.P., V.L.,V.S., V.E.) / parameters	SIZEOF [<*Ar-ray_name>*, <*Dimension>*] or for 1. Dim. SIZEOF [<*Ar-ray_name>*]	$IF SIZEOF [V.P.MYARRAY,2] == 3 P1 = SIZEOF[P10,2] SIZEOF always results in -1 for non-existent array dimensions and for variable that are not arrays.
Determine smaller value	MIN [*x,y*]	P1 = MIN [P2, P3]
Determine greater value	MAX [*x,y*]	P1 = MAX [P2, P3]
larger value sign	SIGN [..]	P1 = SIGN [P2] results in positive values: 1 negative values: -1 Zero: 0

Encryption functions:

These functions are used to encrypt and decrypt strings. The related key is user-definable. Strings may contain important data that require protection by encryption.

Encrypted data can then be saved to file with #MSG SAVE, for example, or supplied to the PLC by V.E. variables.

Encrypt string	ENCRYPT ["Key", "String"]
Decrypt string	DECRYPT ["Key", "Decrypted string"]

The product of ENCRYPT or DECRYPT is assigned to a string type variable. In this case, note the following:

- With ENCRYPT, the string variable must at least be double the length of the string to be encrypted.
- With DECRYPT, the string variable must at least be half the length of the string to be decrypted.

Encrypt string and save to file

```
N10 V.E.encrypted = ENCRYPT[ "Key", "String to be encrypted" ]
N20 #MSG SAVE ["Encrypted text = %s", V.E.encrypted ]
..

Decrypt encrypted string and output as message:
N10 V.E.decrypted = DECRYPT[ "Key", V.E.encrypted ]
N20 #MSG ["%s", V.E.decrypted ]
..
M30
```

2.5.2   Operations for character strings

Overview of all available operations:

String operations:

Adding strings	The + character combines 2 strings. V.E.str = "Hello" + " world!" (-> Result is "Hello world!")
+	

Determine left substring	LEFT supplies the left starting string of a string. Get *anz* characters from string *str* based on the first character. V.E.str = LEFT["Hello world!", 5] (-> Result is "Hello")
LEFT[*str, anz*]	

Determine middle substring	MID supplies the substring of a string. Get *anz* characters from string *str* starting with character at position *pos*. V.E.str = MID["How are you?", 3, 5] (-> Result is "are")
MID[*str, anz, pos*]	

Determine right substring	RIGHT supplies the right final string of a string. Get *anz* characters from string *str* based on the final character. V.E.str = RIGHT["Hello world! How are you?", 12] (-> Result is "How are you?")
RIGHT[*str, anz*]	

Determining string length	LEN determines the length (number of characters) of a string.
LEN[*str*]	P1 = LEN["Hello world! How are you?"] (-> Result is 25)

Determining character value ORD[*str, pos*] or ORD[*str*]	ORD supplies the numerical value of a character in a string at position *pos*. P1 = ORD["Hello world!", 1] (-> Result is 72, character value of "H") P2 = ORD["Hello world!", 7] (-> Result is 119, character value of "w") If no position *pos* is specified, the value of the first character is returned. If a position *pos* greater than the length of the character string is specified, the error ID 21545 is output. For ASCII characters, the ORD function returns the exact ASCII value. The return value for empty character strings is 0. Available as of V3.1.3079.21

ℹ️ FIND[..] is case-sensitive (uppercase/lowercase).

Search for substring	FIND searches for a string str2 in a string str1 and gives the position of the first match of str2 in str1.

FIND[*str1, str2*]	```V.E.str1 = "Hello world! How are you?"```
	```V.E.str2 = "How"```
	```P1 = FIND[V.E.str1, V.E.str2]```
	```(-> Result is 14)```

	If string str2 in string str1 does not exist, FIND gives the result 0.

	```V.E.str1 = "Hello world! How are you?"```
	```V.E.str2 = "today"```
	```P1 = FIND[V.E.str1, V.E.str2]```
	```(-> Result is 0)```

Deleting a substring	DELETE deletes in string str a specific number of characters anz, starting with the character at position pos.
DELETE[*str, anz, pos*]	```V.E.str = DELETE["Hello world! How are you?", 5, 7]``` ```(-> Result is "Hello ! How are you?")```

Inserting a substring	INSERT inserts string str2 in string str1 starting after the character at position pos.

INSERT[*str1, str2, pos*]	```V.E.str1 = "Hello ! How are you?"```
	```V.E.str2 = "world"```
	```V.E.str = INSERT[V.E.str1, V.E.str2, 6]```
	```(-> Result is "Hello world! How are you?")```

Replacing a substring	REPLACE replaces a number of characters anz in string str1 by the substring str2, starting with the character at position pos.

REPLACE[*str1, str2, anz, pos*]	```V.E.str1 = "What is your name?"```
	```V.E.str2 = "age"```
	```V.E.str = REPLACE[V.E.str1, V.E.str2, 4, 14]```
	```(-> Result is "What is your age?")```

Combining strings [as of V3.1.3079.40]
 	
FSTRING["*str_stat*", *str_dyn*, {"*str_stat*", *str_dyn*,}]
	FSTRING generates strings with a dynamic content. In principle, the number of substrings in unlimited and consists of statis and dynamic elements.
	
	• Static elements are specified strings ("*str_stat*"). They are directly included in the resulting string.
	
	• Dynamic elements are all permitted operations with string and numerical values (*str_dyn*). The result of this calculation is then also included as a string in the resulting string.
	
	The result of an FSTRING operation is a string. It can also be used to continue the calculation with a plus (or other operators permitted for strings).
	
	• FSTRING[…] + FSTRING[…]
	
	• FSTRING[…] + "another string" ..
	
	This also permits the nesting of FSTRINGs:
	
	• FSTRING[.., FSTRING[..]]
	
	`V.L.Number = 123`
	`V.L.Float = 3.57`
	`V.E.str = FSTRING["V.L.Number: ", V.L.Number,`
	`                  " / V.L.Float: ", V.L.Float]]`
	`(-> Result is: "V.L.Number: 123 / V.L.Float: 3.57")`

Comparison operators:

ℹ️ Comparison operations are case-sensitive.

Equality
 	
==
	`V.E.str1 = "Peter"`
	`V.E.str2 = "Peter"`
	`$IF V.E.str1 == V.E.str2`
	`#MSG ["%s is equal to %s!", V.E.str1, V.E.str2]`
	`$ELSE`
	`#MSG ["Strings are not equal!"]`
	`$ENDIF`
	`(-> Result is "Peter is equal to Peter")`

Inequality
 	
!=
	`V.E.str1 = "Peter"`
	`V.E.str2 = "Steve"`
	`$IF V.E.str1 != V.E.str2`
	`#MSG ["%s is not equal to %s!", V.E.str1, V.E.str2]`
	`$ELSE`
	`#MSG ["Strings are equal!"]`
	`$ENDIF`
	`(-> Result is "Peter is not equal to Steve")`

Greater than
 	
or
 	
Greater than or equal to
	```V.E.str1 = "Peter"```
	```V.E.str2 = "Peter"```
	```$IF V.E.str1 > V.E.str2```
	```#MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2]```
	```$ELSEIF V.E.str1 >= V.E.str2```

>
 	
>=
	```#MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2]```
	```$ENDIF```
	```(-> Result is "Peter is greater than or equal to Peter!")```
	
	```V.E.str1 = "Peter"```
	```V.E.str2 = "Bob"```
	```$IF V.E.str1 > V.E.str2```
	```#MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2]```
	```$ELSEIF V.E.str1 >= V.E.str2```
	```#MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2]```
	```$ENDIF```
	```(-> Result is "Peter is greater than Bob!")```

Less than
 	
or
 	
Less than or equal to
	```V.E.str1 = "Peter"```
	```V.E.str2 = "Peter"```
	```$IF V.E.str1 < V.E.str2```
	```#MSG ["%s is less than %s!", V.E.str1, V.E.str2]```
	```$ELSEIF V.E.str1 <= V.E.str2```

<
 	
<=
	```#MSG ["%s is less than or equal to %s!", V.E.str1, V.E.str2]```
	```$ENDIF```
	```(-> Result is "Peter is less than or equal to Peter!")```
	
	```V.E.str1 = "Bob"```
	```V.E.str2 = "Tim"```
	```$IF V.E.str1 < V.E.str2```
	```#MSG ["%s is less than %s!", V.E.str1, V.E.str2]```
	```$ELSEIF V.E.str1 <= V.E.str2```
	```#MSG ["%s is less than or equal to %s!", V.E.str1, V.E.str2]```
	```$ENDIF```
	```(-> Result is "Bob is less than Tim!")```

Conversion functions:

Integer to String	INT_TO_STR[...]	```V.E.str = INT_TO_STR[123]```
Real to String	REAL_TO_STR[...]	```V.E.str = REAL_TO_STR[12.34]```
String to Integer	STR_TO_INT[...]	```V.E.sgn32 = STR_TO_INT["12"]```
String to Real	STR_TO_REAL[...]	```V.E.real64 = STR_TO_REAL["123.45"]```

2.5.3   Assigned address characters

Fixed meanings are assigned to the following address letters:

Address letters relating to technology:

D,d	*<int, double, expr>*	Tool compensation call
E,e	*<int, double, expr>*	Feed at block end
F,f	*<int, double, expr>*	Feed at block start
H,h	*<int, double, expr>*	Auxiliary function
M,m	*<int, double, expr>*	Switch function
S,s	*<int, double, expr>*	Spindle speed, synchronisation ratio etc.
T,t	*<int, double, expr>*	Tool position call

Address character relating to geometry:

G,g	*<int, double, expr>*	Preparatory function
I,i	*<int, double, expr>*	Interpolation parameter for 1st path axis
J,j	*<int, double, expr>*	Interpolation parameter for 2nd path axis
K,k	*<int, double, expr>*	Interpolation parameter for 3rd path axis
R,r	*<int, double, expr>*	Circle radius

Address letters relating to program flow:

L,l	*<string>*	Subroutine call, global
LL,ll	*<string>*	Subroutine call, local
N,n	*<int, double, expr>*	Block number
O,o	*<int, double, expr>*	Not assigned
$		Identifier for control block statements
#		Identifier for extended syntax elements

Address letters relating to arithmetic:

P,p	*<int, double, expr>*	Parameter

Variabel und per Kanalparametersatz zuweisbar sind die Adressbuchstaben zur Bezeichnung der numerischen Achsen. Normally the letters X, Y, Z are used to designate the 3 linear axes of a Cartesian spatial coordinate system. A, B, C are used to designate rotary axes (case-sensitive). According to DIN 66025, the second motion in parallel to the spatial coordinate system is specified by U, V, W.

Besides this simple option, axis designations may also consist of several characters (strings) (X_ACHSE, Y22, ZA3). To distinguish between an axis designation and a coordinate value, the "=" character is used, i.e. X1=120.345

The following sections in this programming manual mainly use the first case mentioned and often use address letters.

BECKHOFF

2.5.4   Programming examples

To clarify the subject mentioned above, a programming example is presented here in anticipation of the sections further below.

The following address characters are used:

Path preparatory functions:

G00 "Rapid traverse"

G01 "Linear interpolation"

Spindle:

S1000 "Spindle speed 1000 rpm."

Feedrate:

F5000 "Feedrate 5000 mm/min"

Numerical axes:

X, Y, Z "three Cartesian axes"

Machine functions:

M03 "Clockwise spindle rotation at programmed speed"

M05 "Spindle stop"

M30 "Program end"

Summary of previous commands described

```
% 100
N10 G00 X100 Y100          ;Move at rapid traverse to X=Y=100
N20 Z100;                   ;Move at rapid traverse to Z=100 G00
                           ;remains active until deselected by
                           ;another G function occurs
N30 G1 Z50 F5000 S1000 M3  ;Clockwise spindle rotation 1000 rpm and
                           ;Move linearly at feedrate
                           ;5000 mm/min) to Z=5
N40 Z100;                   ;Move at feedrate to Z=100
N50 G0 X200 Y200 Z200      ;Move at rapid traverse to X=Y=Z=200
N50 M5                     ;Spindle stop
N60 M30                    ;Program end
```

3   Path information

3.1   Axis commands

Axis designations are configurable and must be taken from the configuration-specific description [1] [▶ 786]-5. When decoded, the notation is case-sensitive (uppercase/lowercase).

The following axis designations are available:

- Individual address letters: {A, B, C, U, V, W, X, Y, Z, Q}

- After programming an axis designation consisting of only one address letter, a space is required between the position value and the next character in order not to cause any mix-up when the equals sign is assigned afterwards.

The axis designations "X" and "X50P1" exist in the NC channel and axis "X" should be moved to position "50".

X50P1=7	(ERROR)	X50P1 axis moves to position 7.
X50 P1=7	(RIGHT)	X axis moves to position 50.

- Strings (e.g. X_SCHLITTEN, X1, Y22, Z_ACHSE)

- The first character of the string must correspond to one of the reserved address letters (see above). Other characters may include the numerals 0 - 9. The string length of the axis designation may not exceed the maximum possible length (fixed), otherwise an error message is output.
  To avoid confusion, an equals sign must be placed in front of the position statement after all axis designations consisting of more than one character.
  This is necessary especially for axis designations ending with one of the numerals 0 - 9.

●
i
An equals sign must follow axis identifiers that contain more than one character.

X1 = <int, double, expr>   **Examples:**
                                                X1 = 100.0
                                                X22 = 0,001
                                                X_SCHLITTEN = SIN [30]
                                                Z_ACHSE = SQRT [2]/2

The following declarations also apply:

- All axis identifiers must be specified in the channel parameter block [1] [▶ 786]-5.
- An axis designation must always be followed by a numerical value or expression:

X <int, double, expr>

Examples:
X 100.0
Y 0.001
Z SIN [30]
A SQRT [2]/2
B 4 * R1/R2

Axis commands

```
;Axis identifiers used:
;Y, Y50, Y_ACHSE_SCHL_1, Z7

N010 G01 F1500
N020 Y50 = 51                   ;Axis Y50 to position 51
N030 Y52                        ;Axis Y to position 52
N040 Y50 Z7 = 54                ;Axis Y to position 50 and
                                ;Axis Z7 to position 54
N050 Y 70 Z7 = 55               ;Axis Y to position 70 and
                                ;Axis Z7 to position 55
N060 Y = 71 Z7 = 56             ;Axis Y to position 71 and
                                ;Axis Z7 to position 56
N070 Y[2+3]                     ;Axis Y to position 5
N080 Y50 = [4*3]                ;Axis Y50 to position 12
N090 Y_ACHSE_SCHL_1 = 23        ;Axis Y_ACHSE_SCHL_1 to
                                ;Position 23
N100 Y50 = P1                   ;Axis Y50 to position P1
N110 M30
```

This programming manual uses the common designations X, Y, Z for the three linear axes of a Cartesian system and A and B for two further path axes.

3.2   Measuring systems, input and precision ranges

The measuring systems used to specify positions, angles and velocities are follows:

Lengths and positions:	mm or inch
Linear velocities:	mm or inch per s or min
.. or also application-specific:	m per min
Angle:	deg or gon,
Angular velocity:	deg or gon per s or min.

In addition, the programmer has the freedom to apply self-defined measuring systems using parameter calculations.

All length specifications and linear velocities are calculated to a standard accuracy of 0.1 µm. The maximum path length that can be entered at this resolution is then -214 m... +214 m. Numerical inputs when programming may not exceed a range of -214 000.0000 mm to +214 000.0000 mm. This does not apply to individual elements (e.g. parameters) of an arithmetical expression if the result of the arithmetical expression is within the given range. It must be considered that the numerical range may not be exceeded even in combination with offsets and compensations.

Exception: The definition of a circle radius is possible up to maximum 10^9 mm. However, the target point of the arc must always be within the maximum path range (-2.14... +2.14) * 10^5 mm.

Enter the axis type in the channel parameter block. If the application involves a rotary axis (or spindle), the angles are handled at a resolution of 0.0001° by default. A range of n * 360° where n=1190 is then programmable.

At a resolution of 0.0001°, angles are possible within a range of

(-2.14 ... +2.14) * 10^5 / 360 = 2 * 594 revolutions.

3.3 Coordinate systems

After the homing, the controller is at the machine zero point or in the machine coordinate system. If this is followed by inputs from the NC program (e.g. X100), the programmed coordinates (*Index p*) then coincide with the absolute coordinates (*Index a*):

$$x_a = x_p$$

$$y_a = y_p$$

Offsets occur due to the definition of workpiece coordinate systems. The spatial positions in the workpiece coordinate system differ from the coordinate systems that are defined by the physical machine axes. Here a distinction must be made between programmed, constant, translatory offsets in a single axis and dynamic offsets that result from kinematic (e.g. cylindrical □ Cartesian) or geometric transformations (e.g. tool radius compensation, mirroring), which in general affect several axes.

For example, the zero point can be offset from machine zero point M to a freely selectable workpiece zero point W or a workpiece coordinate system by a zero offset (NPV-G54…G59). The absolute coordinates result from adding NPV and programmed coordinates:

$$x_a = x_{NPV} + x_p$$

$$y_a = y_{NPV} + y_p$$

Irrespective of these NPVs specified by the zero offset data block, additional offset types can be explicitly programmed in the subroutine e.g. with G92 X... Y ... Z ...

This reference point offset (BPV) is added to the preceding NPV. The absolute coordinates can then be determined as follows:

$$x_a = x_{NPV} + x_{BPV} + x_p$$

$$y_a = y_{NPV} + y_{BPV} + y_p$$
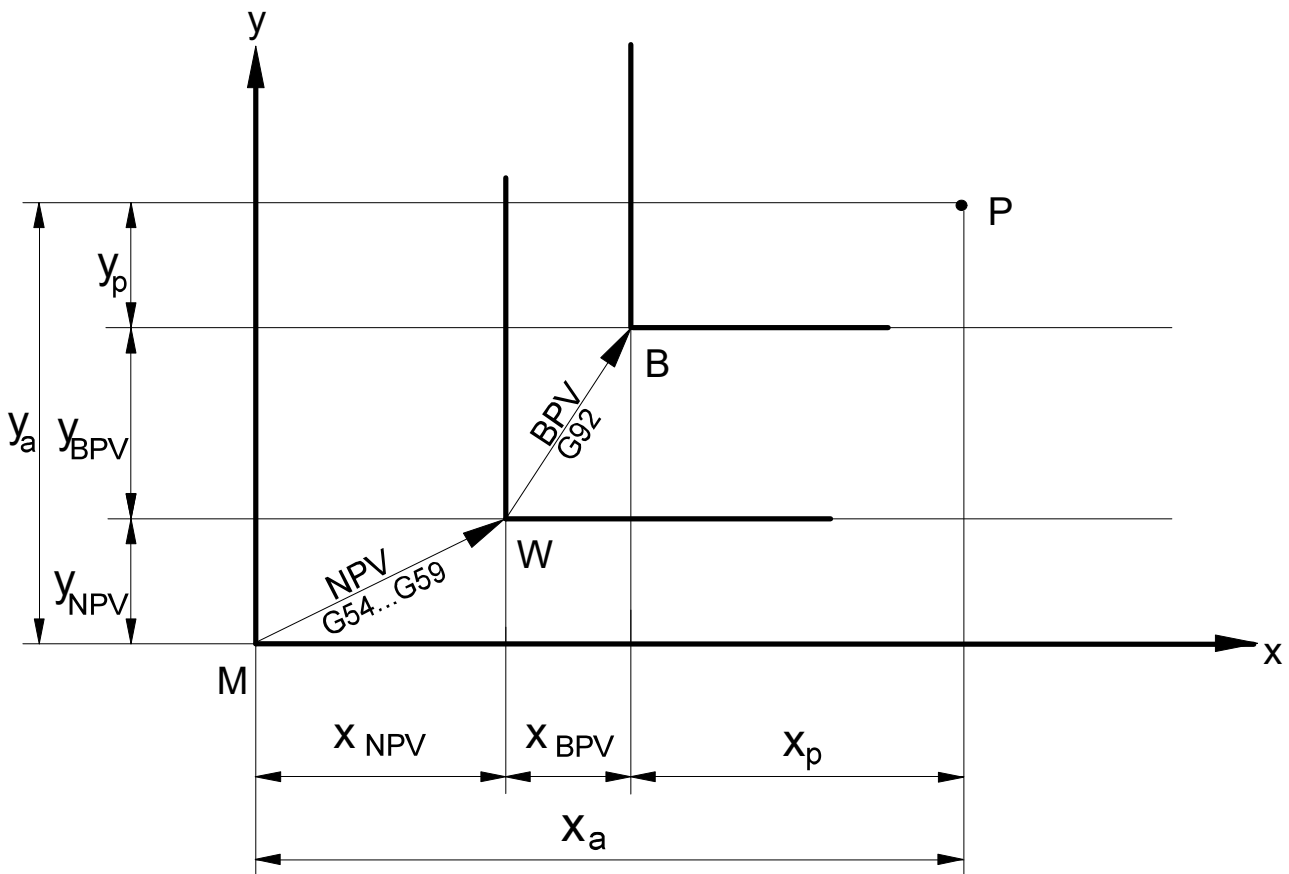
Fig. 1: Defining a workpiece coordinate system using NPV and BPV (for legend, see below)

xa, ya :	Absolute coordinates
xp, yp :	Programmed coordinates
xNPV, yNPV :	Zero offset
xBPV, yBPV :	Reference point offset
M :	Machine zero point
W :	Workpiece zero point
B :	Reference point for coordinates
P :	Position

The coordinate display on the user interface shows active offsets by the remaining difference between the coordinates of physical machine axes (ACS) and workpiece coordinates (PCS). However, some offsets also result from manipulating machine and workpiece coordinates (e.g. tool radius compensation, mirroring) and therefore do not result in a coordinate difference.

The tables below provide an overview of the additional offset types in anticipation of the sections further below. The following conditions for the parameters are active:

Activation and **deactivation** infer the time when the offset becomes visible on the user interface as a coordinate difference or change in coordinates. However in general, an offset only becomes physically active at the earliest with the first motion that follows any activation or deactivation. For example, deactivation at program end leads to a compensating motion in the first motion of the following program.

Programmable offsets (linear, constant)

No.	Description	Definition	ACS – PCS Difference when active	Activation	Deactivation	Temporary suppression

BECKHOFF

1	Reference point offset	NC program	yes	NC block "*G92 X.. Y..*" G90/91 Consider dependency	NC block "*G92 X0 Y0*" or NC program start	"#SUPPRESS OFFSETS" "#MCS ON"
2	Zero offset	List NC prg.	yes	NC block "*G54...G59*"	NC block "G53" or NC program start	"#SUPPRESS OFFSETS" "#MCS ON"
3	Clamp position offset	List	yes	Program paths Not changeable during program	NC program start Not changeable during program	"#SUPPRESS OFFSETS" "#MCS ON"
4	Tool offset	List NC prg. External	yes	NC block "*D..*"	NC block "D0" or NC program start	"#SUPPRESS OFFSETS" "#MCS ON"
5	Position preset	NC program	yes	NC block "#PSET.."	NC block "#PRESET..." or program start	"#SUPPRESS OFFSETS" "#MCS ON"
6	CS offset	NC program	yes	NC block "*#CS ON[vx,vy,vz,..*"	NC block "#CS OFF" or NC program end	"#MCS ON"
7	ACS offset	NC program	yes	NC block "*#ACS ON[vx,vy,vz,..*"	NC block "#ACS OFF" or NC program end	"#MCS ON"

Offsets caused by geometric transformation (linear, dynamic):

No.	Description	Definition	ACS – PCS Difference when active	Activation	Deactivation	Temporary suppression
8	CS	NC program	yes	NC block "#CS ON[..,..,..]"	NC block "#CS OFF" or NC program end	"#MCS ON"
9	ACS	NC program	yes	NC block "#ACS ON[..,..,..]"	NC block "#ACS OFF" or NC program end	"#MCS ON"
10	Contour rotation	NC program	No	NC block "#ROTATION ON[ANGLE..]"	NC block "#ROTATION OFF"	"#MCS ON"
11	Mirroring	NC program	No	NC motion block after G21/22/23	NC motion block after G20	Not suppressible
12	Tool radius compensation	NC program	No	NC block G41/42	NC block G40	Not suppressible
13	Kinematic transformation	NC program	No	NC block "#TRAFO ON Automatic program start	NC block "# TRAFO OFF	"#MCS ON"

Offsets caused by special functions:

No.	Description	Definition	ACS – PCS Difference when active	Activation	Deactivation	Temporary suppression
14	Offset due to manual mode with parallel interpolation	Handwheel NC prg.	yes	NC block "G201"	NC block "G202"	Not suppressible
15	Offset due to measurement run	NC program	yes	NC block "G101"	NC block "G102"	"#SUPPRESS OFFSETS" "#MCS ON"
16	Offset after ho- ming	NC program	No	NC block "G74 X.. Y.. Z.. "	Not possible	Not suppressible

BECKHOFF

The NC command #SUPPRESS OFFSETS only acts within an NC block

The NC command #MCS ON deactivates any offsets until command #MCS OFF is programmed.

Within every (A)CS the offset types 1, 2 and 5 are stored "locally".
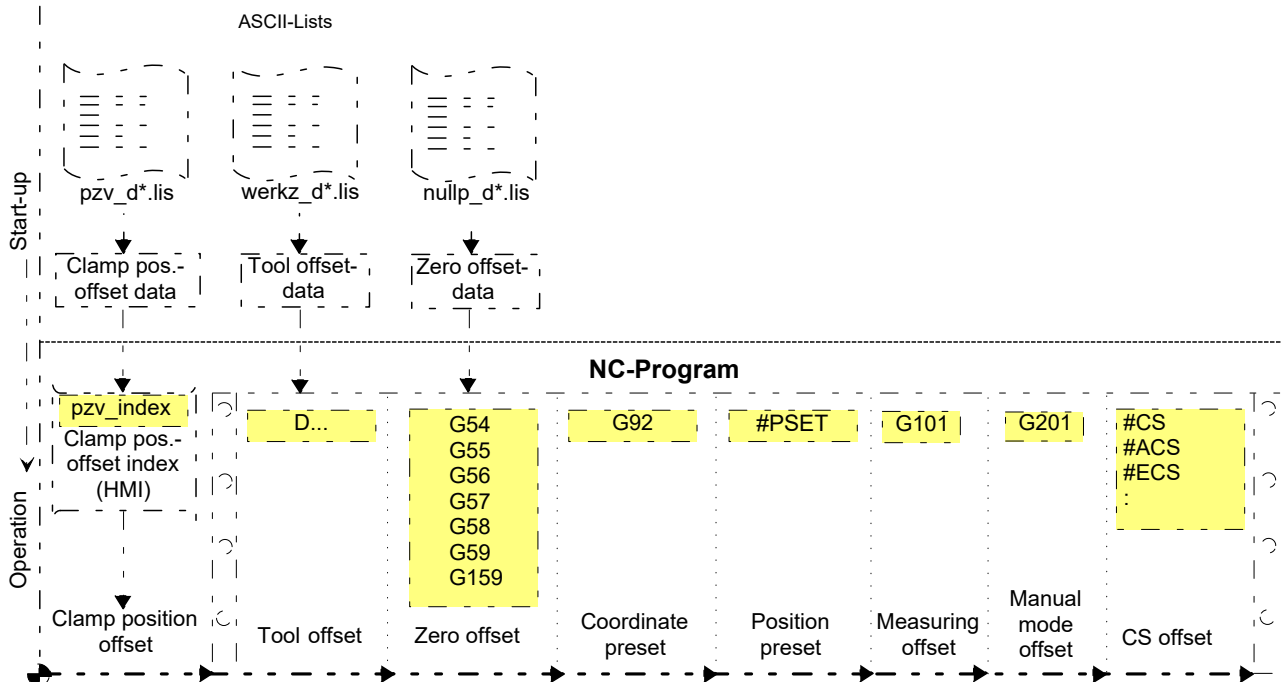
Fig. 2: Overview of additional offsets and coordinate systems

4   G functions

A complete list of G functions is contained in the overview of commands in the Appendix under G functions (G..) [▶ 787].

"G" functions describe the type of feed motion, interpolation type and measuring method. They switch time-related influences and activate certain operation states. The syntax consists of the letter G combined with an identifier. The identifier is assigned a fixed validity and can be optionally preceded by a 0:

```
Syntax:
```
G[0]*<id>*

Various distinguishing features must be considered with G functions:

Effectiveness:

There are G functions that only retain their validity for a specific block after programming (**non-modal**) and G functions which retain their validity the first time they are programmed until they are explicitly deselected (**modal**).

Exception:

Certain G functions mutually exclude each other. For example, G01 (linear interpolation) and G02 (circular interpolation) cannot be selected simultaneously. Therefore, the functions summarised in these groups may not be programmed in the same NC block.

A modal function is automatically deselected if a different function in the same group is selected in a following block.

G functions

```
:
N50 G01 X100 Y200               ; linear interpolation effective
N60 G41 X200 Y200               ; linear interpolation effective
N70 X300 Y250                   ; linear interpolation effective
N80 X100 Y50          ; linear interpolation effective
N90 G02 X100 Y50 I100           ; circular interpolation effective
:
```

Initial position:

During switching, after RESET or at program end, the controller is in initial position. In initial position, several G functions are active without being explicitly selected.

4.1 Path preparatory functions

4.1.1 Rapid traverse (G00)

```
Syntax:
```

| **G00** | Linear interpolation in rapid traverse | modal |

When G00 is selected, the rapid traverse velocity of the axes (specified in the machine parameters) is used for motion velocity. This results in an axis velocity where at least one axis is moved at its rapid traverse velocity.

Any number of straight lines can be programmed in the Cartesian spatial coordinate system (X, Y, Z). All programmed tracking axes move at linear velocity in such a way that the start and end of their motion take place simultaneously with the main axes.

Main axis:

Tracking axis:
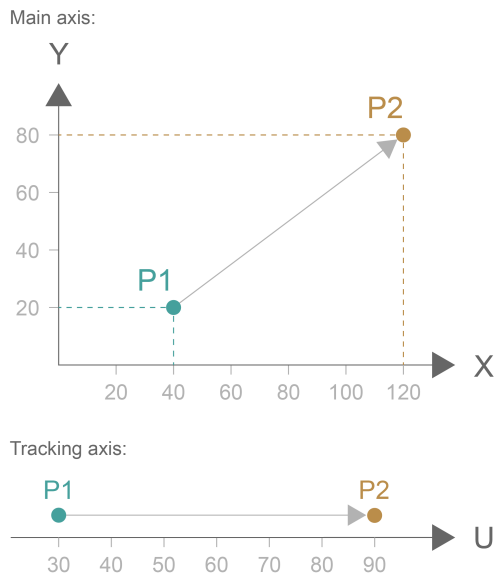
Fig. 3: Position in rapid traverse with the parameters

Rapid traverse G00

```
N05 G00 G90 X40 Y20 U30        ;move to starting point P1
;Absolute dimensional input:
N10 G00 G90 X120 Y80 U90       ;move from P1 to P2
;Incremental dimensional input:
N10 G00 G91 X80 Y60 U60        ;move from P1 to P2
```

Special case: Aligning a circular axis using G00 G90

If a rotary axis is programmed using G00 when G90 is active (G90: Absolute programming), the programmed target point is calculated in modulo, i.e. the rotary axis moves by a maximum of half a rotation.

4.1.2 Linear interpolation (G01)

G01	Linear interpolation at programmed feedrate	modal

If G01 is selected, the programmed path moves along a straight line to the target point at the feedrate specified in the F word (e.g. mm/min). Any number of straight lines can be programmed in the Cartesian spatial coordinate system (X, Y, Z). All programmed tracking axes move at linear velocity in such a way that the start and end of their motion take place simultaneously with the main axes.
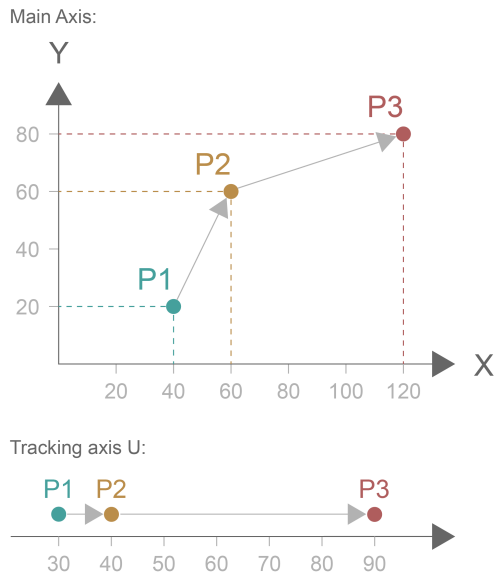
Fig. 4: Graphic display of linear interpolation (G01)

Linear interpolation G01

```
N05 G00 G90 X40 Y20 U30       ; move to starting point P1
; Absolute dimensional input:
N10 G01 G90 X60 Y60 U40 F1000 ;move from P1 to P2 feedrate 1000 mm/min)
N20 X120 Y80 U90 ;move from P2 to P3 feedrate 1000 mm/min)
; Incremental dimensional input:
N10 G01 G91 X20 Y40 U10 F1000 ;move from P1 to P2 feedrate 1000 mm/min)
N20 X60 Y20 U50 ;move from P2 to P3 feedrate 1000 mm/min)
```

4.1.3   Circular interpolation (G02/G03)

Syntax:

G02	Helical interpolation clockwise (CW)	modal
G03	Helical interpolation counter-clockwise (CCW)	modal

When G02 or G03 is selected, the programmed path is travelled to the target point in circular motion at the feedrate specified in the F word (e.g. mm/min) Circular motion can be travelled in the three main planes of the spatial coordinate system (X-Y, Z-X, Y-Z). The main plane is selected using the functions G17, G18, G19 (see Plane selection [▶ 110]).

All programmed tracking axes move at linear velocity in such a way that the start and end of their motion take place simultaneously with the main axes.
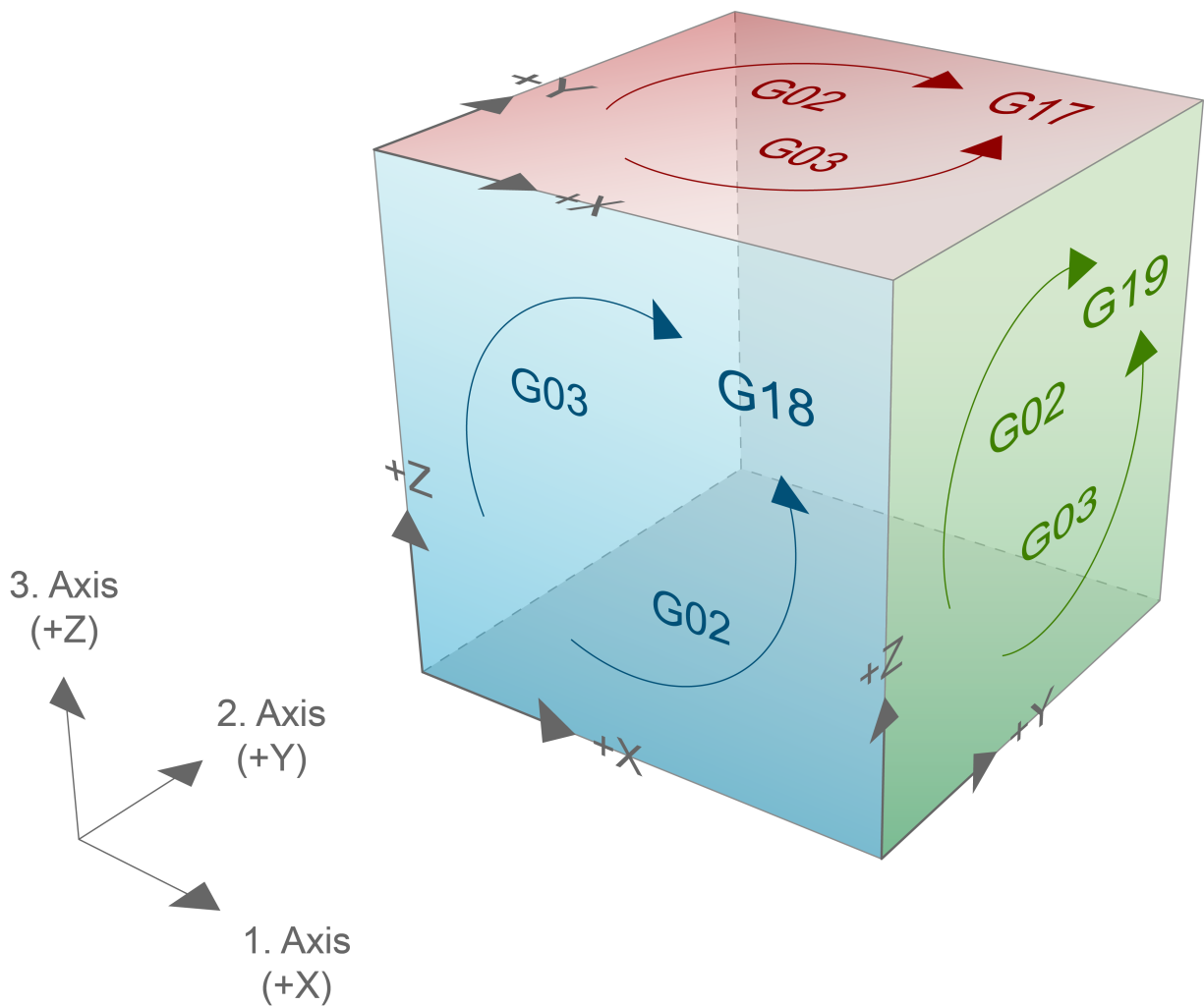
Fig. 5: Description of circle functions G02 and G03

The circle is defined by taking the starting point of the circle "Ka" (determined in the preceding block), the target point of the circle "Ke" and the centre point of the circle "Km". The centre point of the circle is specified by the interpolation parameters I, J, K relative to the starting point of the circle when G162 is active or absolute when G161 is active.

G162:	**(Basic settings)**
I	relative position of Km in the X direction
J	relative position of Km in the Y direction
K	relative position of Km in the Z direction
G161:	
I	absolute position of Km in the X direction
J	absolute position of Km in the Y direction
K	absolute position of Km in the Z direction

If the circle centre point is incorrectly defined, an error message is output if no centre point compensation is switched on (G165). When G165 is active, a centre point is defined so that a circle can be travelled. It also means that if the interpolation parameters are not programmed, circle centre point compensation originates at I, J, K = 0. In addition, the circle centre point coordinates are "non-modal".

The maximum permissible circle radius is 10^9 mm. The target point of the arc may not exceed the maximum motion path of the axes of +- 2.14×10^5 mm.

If G02/G03 is active and if the interpolation parameters I, J, K are programmed with no circle **end** point, a full circle is travelled. Alternatively to I, J, K, the circle radius may also be defined with R (see Radius programming (R, G163) [▶ 134]). However, it is not possible to program a full circle with R.

Syntax example for G17 plane:

G02 | G03 [**X**.. **Y**..] **I**.. **J**.. | **R**..

G02	G03	Circular interpolation CW / CCW
X.. Y..	Target point in XY plane in [mm, inch]
I..J..	Position of circle centre point of interpolation in XY plane (I in X, J in Y) in [mm, inch], according to G161/G162
R..	Radius of an interpolated partial circle (alternative to I, J) in connection with specification of a target point in [mm, inch].

Syntax according to selected interpolation plane:

Plane	Interpolation type	Target point in plane	Centre point/radius
G17	G02/G03	X..Y..	I..J../R
G18	G02/G03	Z..X..	K..I../R
G19	G02/G03	Y..Z..	J..K../R

Circular interpolation

```
N05 G0 X0 Y0
N10 G01 X10 Y10 F1000
N20 G02 X30 Y30 I10 J10           ;Semicircle about M1 circle end point X30 Y30
;Alternative N20:
N20 G02 X30 Y30 R[10*SQRT[2]]   ;Semicircle about M1 circle end point X30 Y30
N30 G02 I10 J10                   ;Full circle about M2
```
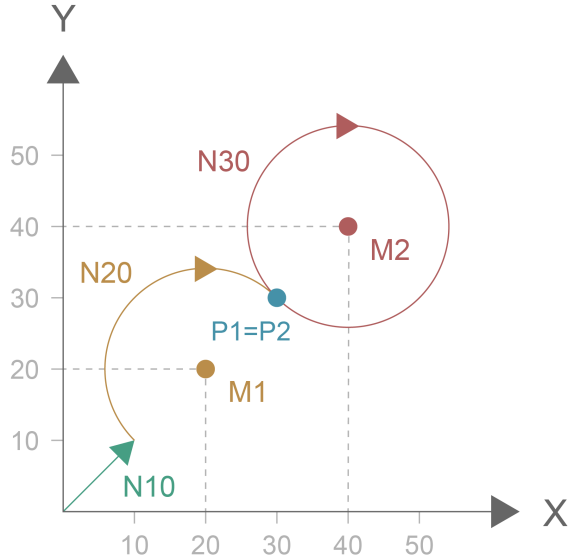

Alternatively, circles can also be programmed by specifying the radius. This is possible using G163="radius value" or using the address letter R="radius value". It is also possible to define using R1="radius value" (see sections Specifying centre point for circle definition (G161/G162) [▶ 133] - Controlling centre point offset in circle (G164/G165) [▶ 137]).

Circular interpolation

```
;Absolute dimensional input:
N05 G90 G00 X40 Y30 U40              ;Circle starting point (Ka), starting position
N10, G90 F1000                        ;Absolute dimension, feedrate
N20 G17                              ;Select X-Y plane
N30 G03 G161 X60 Y50 I60 J30 U90     ;Circle: Ka -> Ke and straight line: P1 -> P2
;Incremental dimensional input:
N05 G90 G00 X40 Y30 U40              ;Circle starting point (Ka), starting position
N10 G91 F1000                        ;Incremental dimension, feedrate
N20 G17                              ;Select X-Y plane
N30 G03 G162 X20 Y20 I20 U50         ;Circle: Ka -> Ke and straight line: P1 -> P2
```

Main axis:
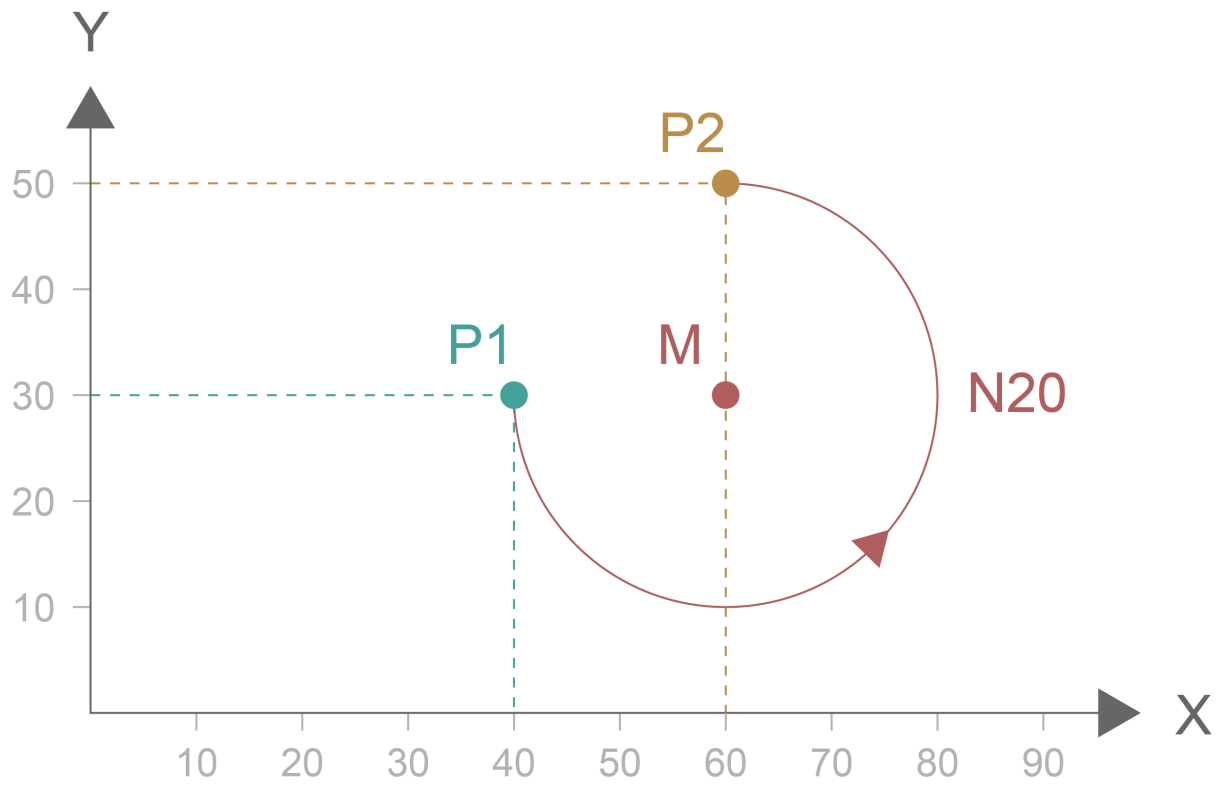
Tracking axis:
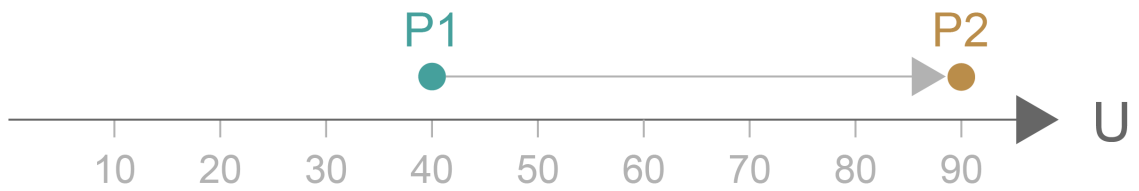
Fig. 6: Examples of circular interpolation

4.1.4 Helical interpolation (G02 Z.. K../G03 Z.. K..)

Helical interpolation is the superposition of a circular interpolation (plane of 1st and 2nd main axes) and a linear motion in the 3rd main axis. The resulting helical motion is executed at constant pitch. The pitch is programmed by the third parameter of the circular interpolation depending on the selected plane.
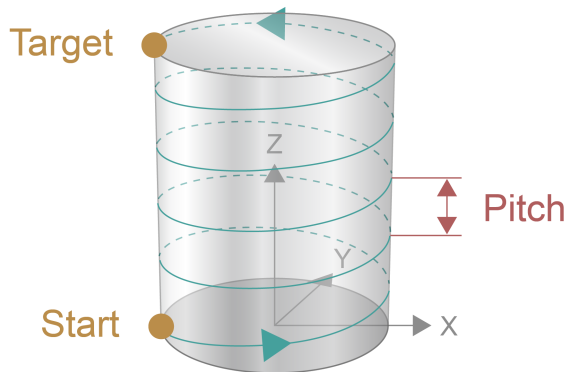
Fig. 7: Displaying helical interpolation at constant pitch

```
Syntax example for G17 plane:
```
G02 | G03 X.. **Y**.. **Z**..**I**.. **J**.. | **R**.. **K**..

G02 | G03	Circular interpolation CW / CCW
X.. Y..	Target point in XY plane in [mm, inch]
Z..	Target point on helical axis perpendicular to XY plane in [mm, inch]
I.. J..	Position of circle centre point of interpolation in XY plane (I in X, J in Y) in [mm, inch], according to G161/G162
R..	Radius of interpolated circle (alternative to I,J) in [mm, inch]
K..	Pitch of helix in Z (value always <u>without</u> sign) in [mm, inch]

Syntax according to selected interpolation plane:

Plane	Interpolation type	Target point in plane	Target point on helical axis	Centre point /radius	Pitch
G17	G02/G03	X..Y..	Z..	I..J../R	K
G18	G02/G03	Z..X..	Y..	K..I../R	J
G19	G02/G03	Y..Z..	X..	J..K../R	I

It is not necessary to define the pitch so that the helix reaches the programmed target point exactly. In this case, the NC kernel calculates a "corrected" pitch taking into consideration the fixed points for start and target points. The corrected pitch approximates to the programmed pitch as closely as possible.

In this case, the helix target point is first calculated based on the programmed pitch. If the calculated target point differs from the programmed target point, correction is required. The criteria for correction is the distance between the programmed target point and the calculated target point viewed in the direction of rotation.

If the distance is less than or equal to π(180°), the target point of the helix is shifted in the <u>op**po**site</u> direction of rotation towards the programmed target point, i.e. the pitch is increased.

If it is greater than π(180°), the target point of the helix is shifted <u>in</u> the direction of rotation of the programmed target point, i.e. the pitch is reduced.
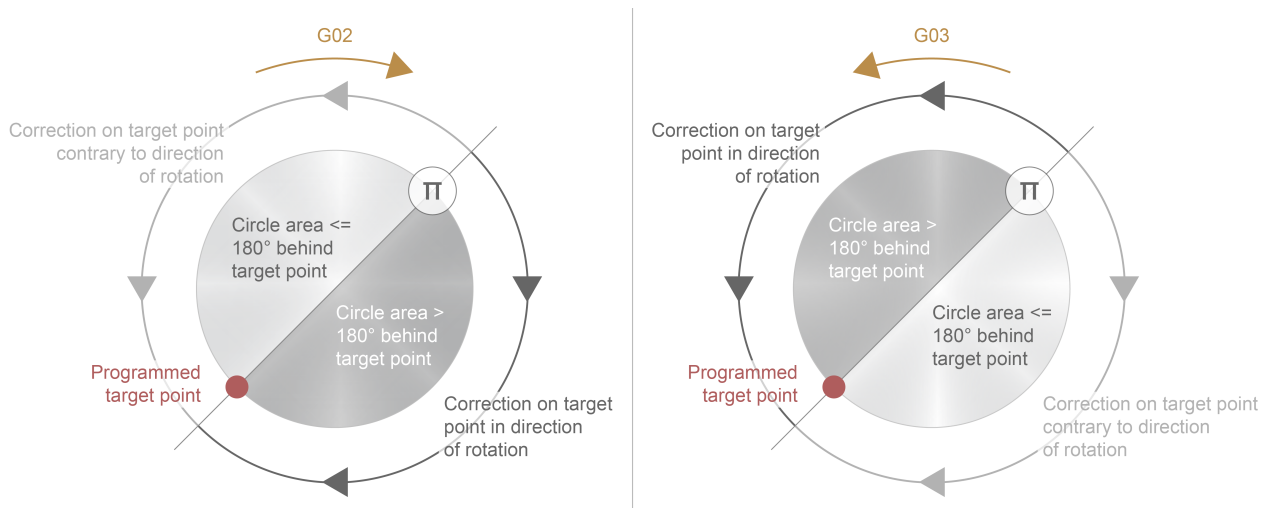
Fig. 8: Correcting the helix pitch depending on the direction of rotation

Principal correction of a helix clockwise (G02) (1st case)

The target point calculated using the programmed pitch P_{prog} is located within the range of 180° <u>behind</u> the programmed target point (viewed in the direction of rotation).

Correction takes place by increasing the pitch P_{corr}.
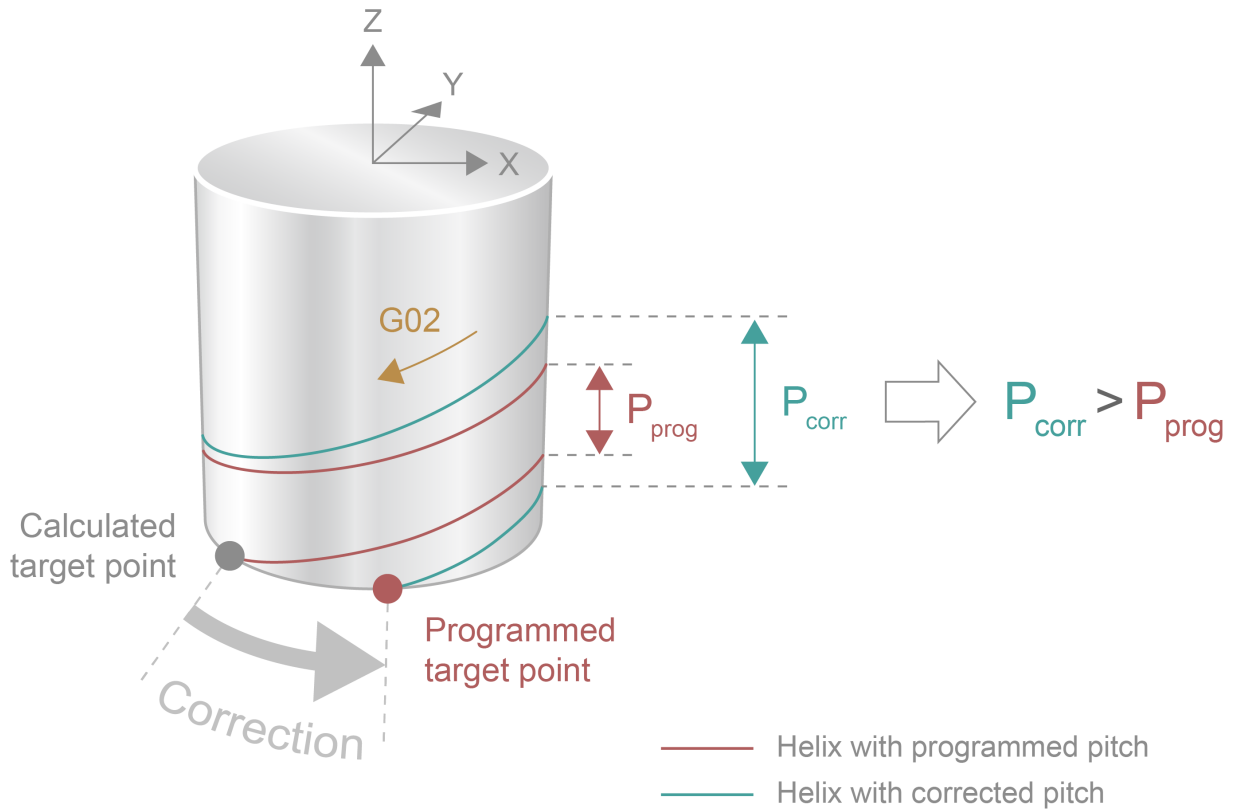
Fig. 9: Correcting a helix within the range of 180° behind the programmed target point

Principal correction of a helix clockwise (G02) (2nd case)

The target point calculated using the programmed pitch P_{prog} is located within the range of 180° <u>ahead</u> of the programmed target point (viewed in the direction of rotation).

Correction takes place by reducing the pitch P_{corr}.
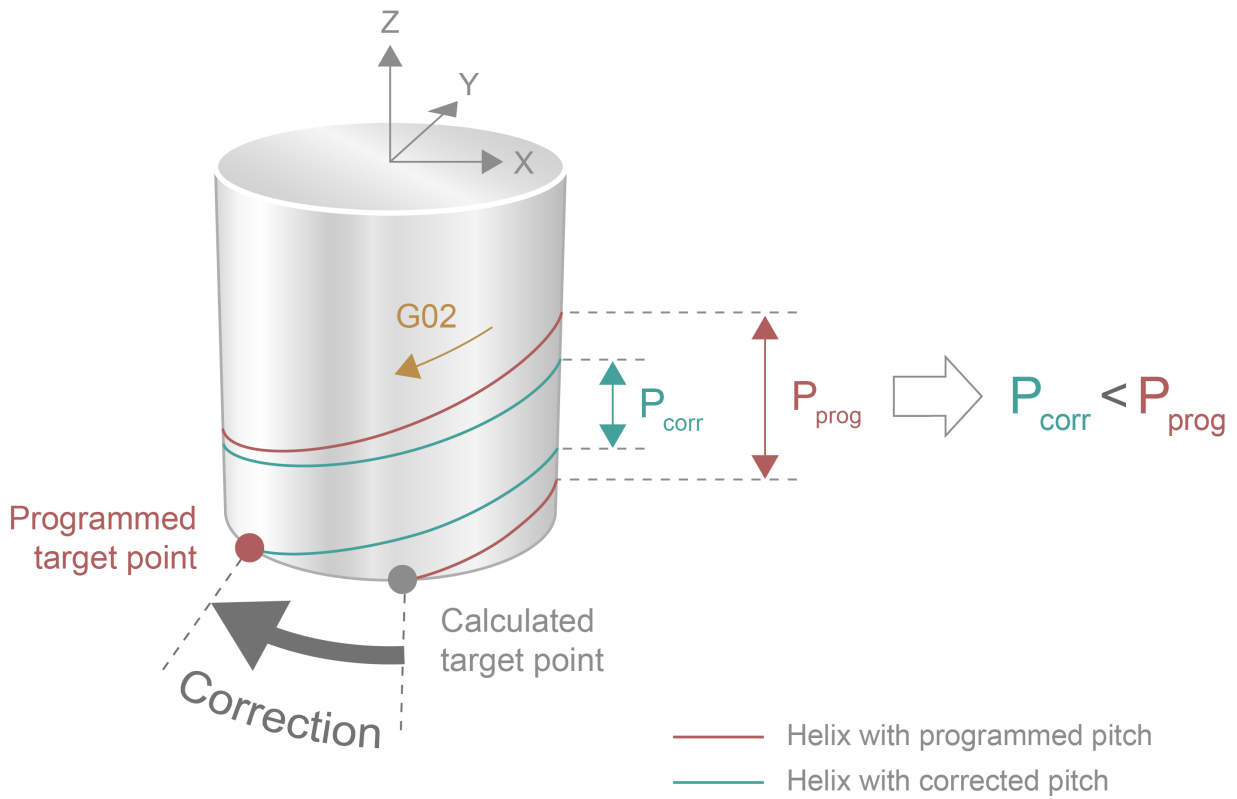
Fig. 10: Correcting a helix within the range of 180° ahead of the programmed target point

Helical interpolation in the XY plane clockwise

The following helix is travelled:

Starting point a: X-10 Y0 Z0

Target point b: X0   Y-10 Z-20

Helix centre point I, J: Zero point

Helix pitch K: variable

```
:
N10 G17 G90 X-10 Y0 Z0 F500 G161
N20 G02 X0  Y-10 Z-20  I0   J0   K..
:
```
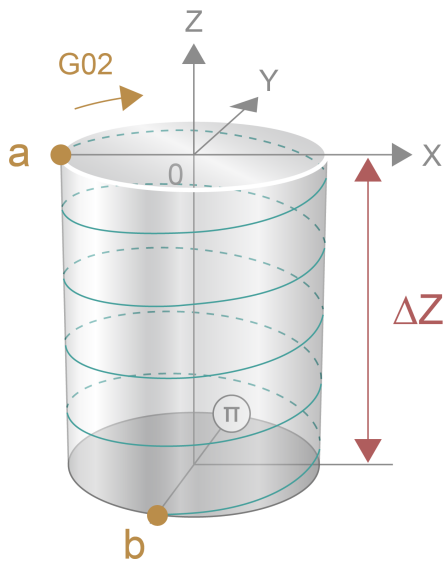

Fig. 11: Helical interpolation in the XY plane clockwise

Minimum rotation: ¾ → pitch K=26.66

Pitch K greater than or equal to 26.66:

The helix from a to b is generally executed in ¾ rotation because the correction is limited to the maximum possible pitch K = 26.66.

Pitch K < 26.666:

Programmed pitch K (in mm)	Helix rotations from a to b	Corrected pitch K (in mm)
17.5	¾	26.66
16	1¾	11.4
15	1¾	11.4
12.5	1¾	11.4
10	1¾	11.4
7.5	2¾	7.27
5	3¾	5.33
2.5	7¾	2.58
2	9¾	2.05
1	19¾	1.01

4.1.4.1   Simple helical interpolation

With simplified helical programming, no pitch is defined, only a target point. Depending on the target point the result is a helical motion with maximum one complete rotation.

```
Syntax example for G17 plane:
```
G02 | **G03 X**.. **Y**.. **Z**.. **I**.. **J**.. | **R**..

G02 | G03	Circular interpolation CW / CCW
X.. Y..	Target point in XY plane in [mm, inch]
Z..	Target point on helical axis perpendicular to XY plane in [mm, inch]
I.. J..	Position of circle centre point of interpolation in XY plane (I in X, J in Y) in [mm, inch], according to G161/G162
R..	Radius of interpolated circle (alternative to I,J) in [mm, inch]

```
Syntax according to selected interpolation plane:
```

Plane	Interpolation type	Target point in plane	Target point on helical axis	Centre point/radius
G17	G02/G03	X..Y..	Z..	I..J../R
G18	G02/G03	Z..X..	Y..	K..I../R
G19	G02/G03	Y..Z..	X..	J..K../R

Helical interpolation in the XY plane counter-clockwise

The following helix is travelled:

Starting point a: X-10 Y0 Z0

Target point b: Z20

Helix centre point I, J: Zero point

```
N10 G17 G90 X-10 Y0 Z0 F500 G161
N20 G03 I0 J0 Z20
```
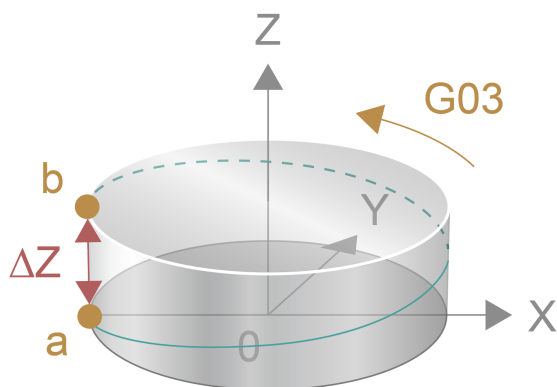

Fig. 12: Helical interpolation in the XY plane counter-clockwise

4.1.5 Arc in space (G303)

i **This function is available as of CNC Build V3.01.3061.0.**

Arcs can be programmed in space using G303. An arc is clearly described by 3 points:

- Starting point
- Interim point
- Target point

Coordinates are programmed for:

- the interim point by I, J, K and
- for the target point by X, Y, Z

Starting point, interim point and target point of an arc may not be located on a straight line and the distance between each of the 3 points must be greater than 0.

The 3 points define the arc plane. The motion direction is clearly fixed by the sequence starting point-interim point-target point.

The coordinates of the interim and target points refer to the currently active absolute or relative dimension specified (G90 or G91). With G91, the last point approached, i.e. the starting point, reference point.

```
Syntax example of G303:
```

G303 I.. **J**.. **K**.. **X**.. **Y**.. **Z**..   modal

G303   Arc in 3D space
I.. J.. K..   Coordinates of interim point in space in [mm, inch]
X.. Y.. Z..   Coordinates of target point in space in [mm, inch]

Restrictions:

- Full circles cannot be programmed using 3-point definition.
- Arc motion is only permitted if tool radius compensation (G40) is deselected.

Arc in space (G303)

```
%Circle_G303
;Start at X0 Y0 Z0
N10 G17 G90 G01 F2000 X0 Y0 Z0
N20 G01 X0 Y0 Z30 ;Circle start point (1)
;Move arc via interim point I,J,K (2) to target point X,Y,Z (3)
N30 G303 I30 J75 K15 X60 Y0 Z0

N100 M30
```
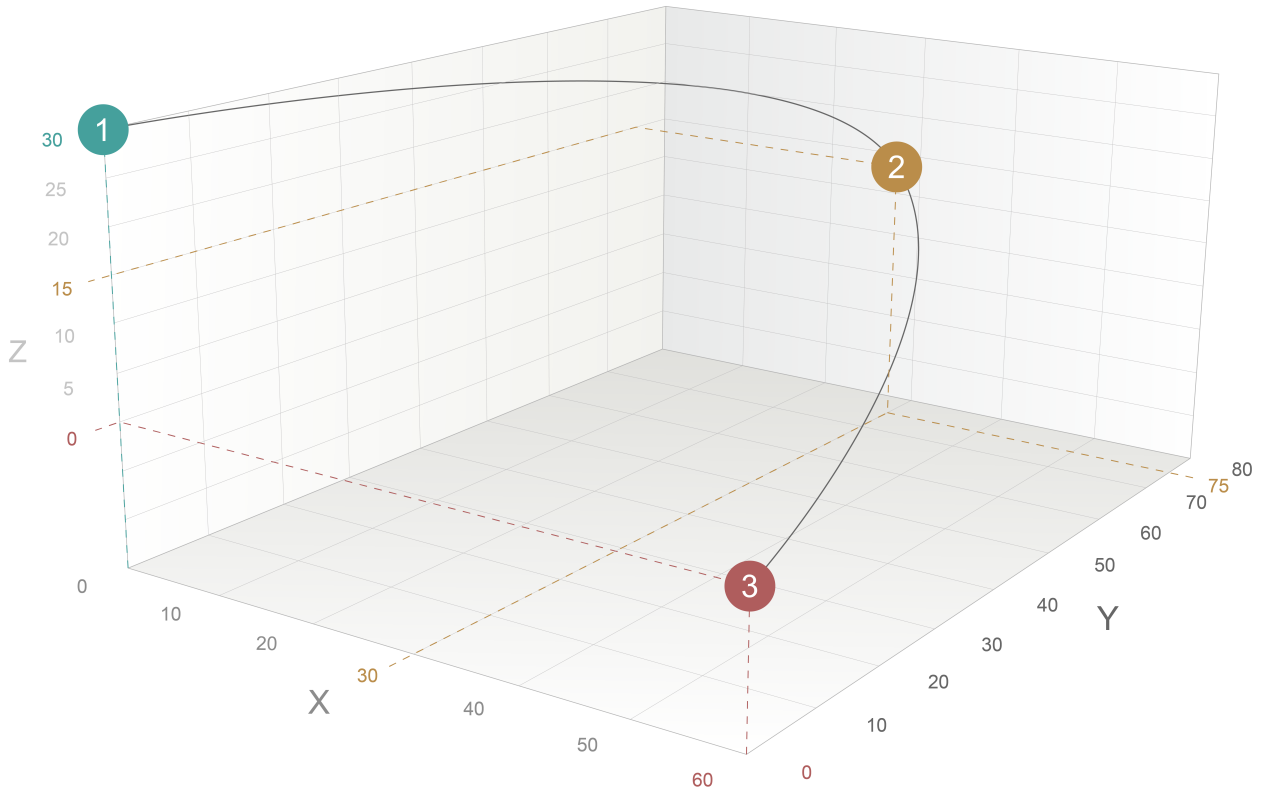

Fig. 13: Arc in space defined by starting point (1), interim point (2) and target point (3)

4.1.6 Contour line programming (#ANG)

In technical drawings, simple contours (e.g. turned parts) are often described by specifying angles and individual positions. This measurement is quick and easy to enter into an NC program using contour line programming.

Contour lines are located in a plane (G17, G18, G19) and describe a different type of linear block programming in the form of straight lines.

Contour line programming with active circular interpolation will result in an error message output.

Contour line consisting of a straight line

Starting with a starting point (SP), a contour line describes the linear block by:

- Specifying an angle (ANG)
- and a coordinate (POS) of the target point (ZP).

The controller calculates the unknown second target point coordinate from the angle and the programmed coordinate. It is irrelevant which of the two target point coordinates is specified. Normally, this depends on the measurement in the actual drawing.

```
Syntax example for G17 plane:
```

#ANG=.. X.. | Y..   modal

#ANG=..   Angle relative to the first main axis of the active plane in [°]
X..   Coordinate of target point in the first main axis in [mm, inch]
Y..   Coordinate of target point in the second main axis in [mm, inch]

Contour line with coordinate in the first main axis
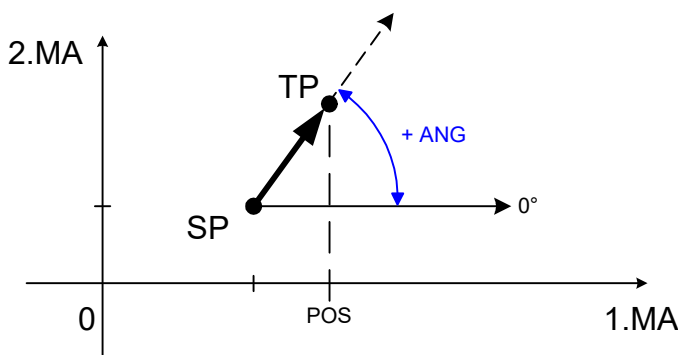
Fig. 14: Contour line with coordinate in the first main axis

Contour line in G17 with target coordinate in X

```
N10 G17 G90 G0 X10 Y10          ;Approach start position
N20 G01 F2000 #ANG=60 X20       ;Contour line to target point: X20, Y27.3205
N30 ...
```

Contour line with coordinate in the second main axis
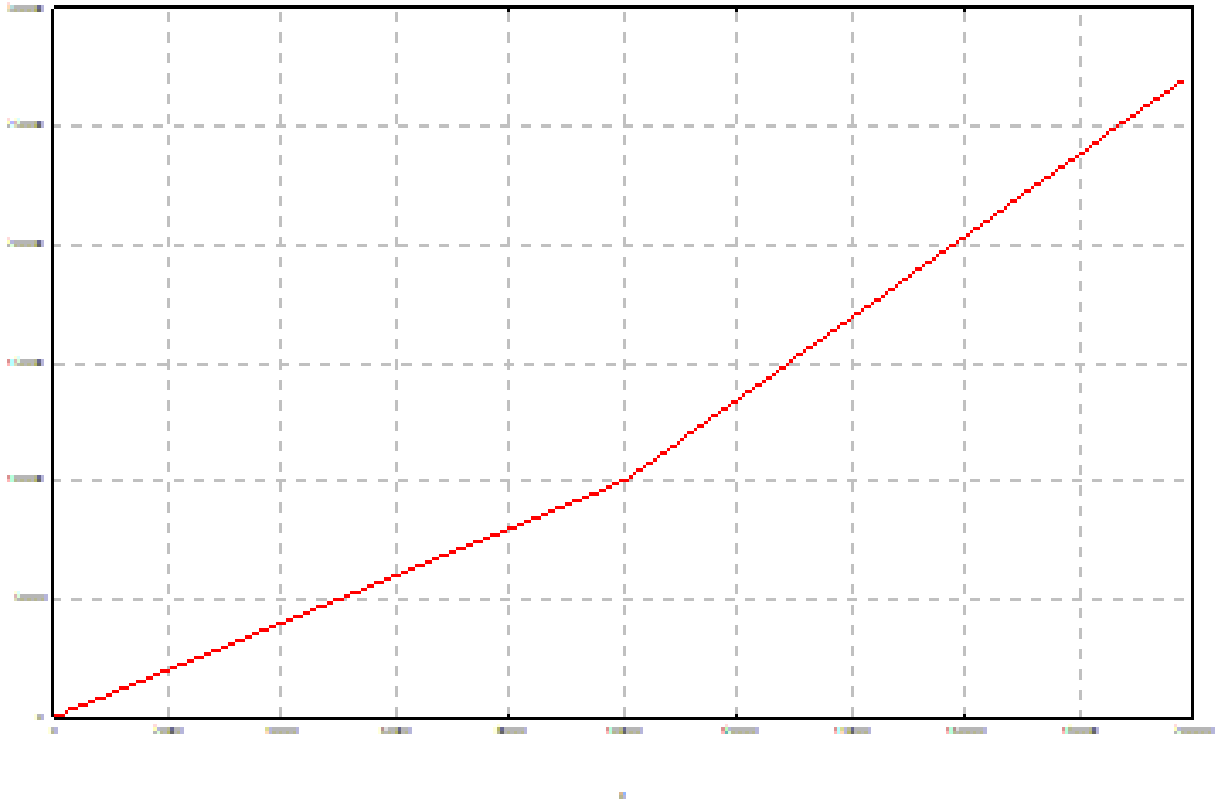
Fig. 15: Contour line with coordinate in the second main axis

Contour line in G17 with target coordinate in Y

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=35 Y20 ;Contour line to target point: X24.2812, Y20
N30 ...
```
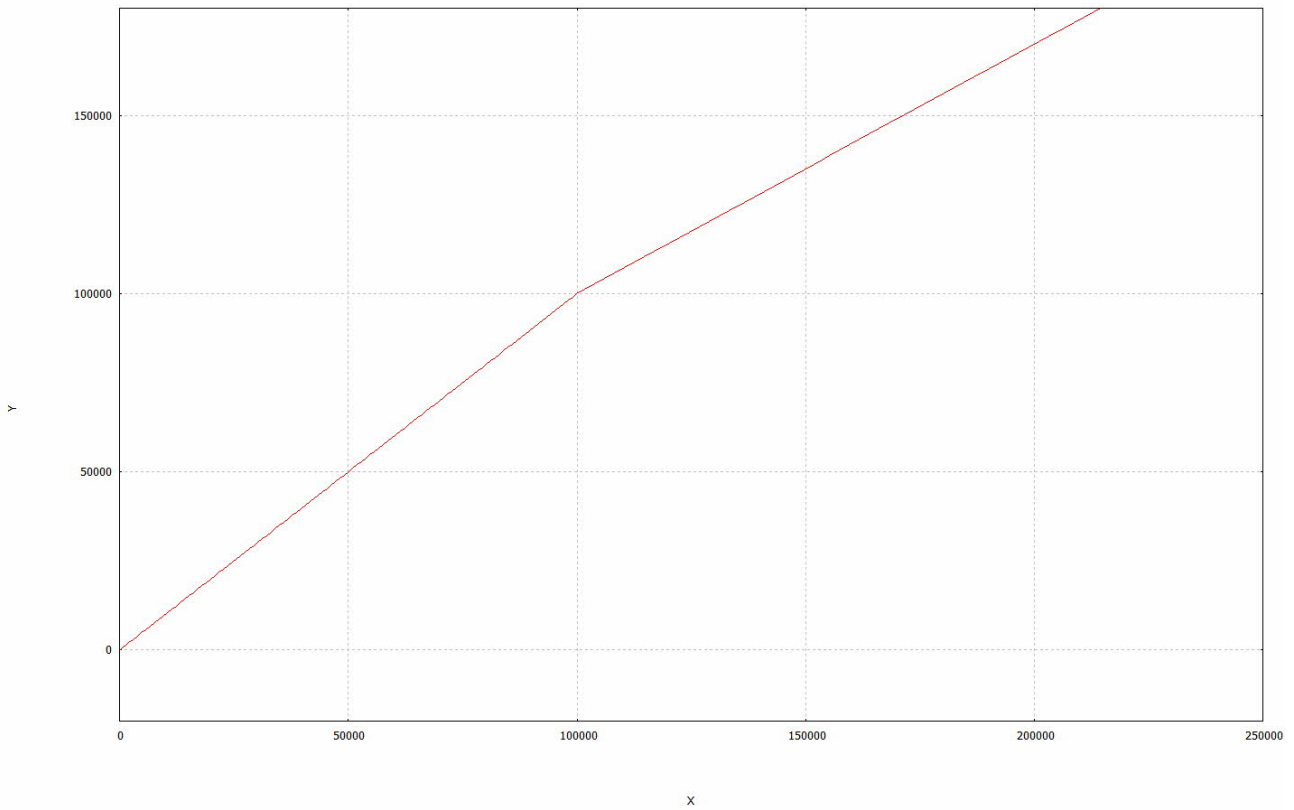
Page break

Validity check of target point:

When the complete target point is determined, a check is made whether the programmed target point coordinate (POS) can be reached with the specified angle. If the target point cannot be reached with the specified angle, an error message is output.
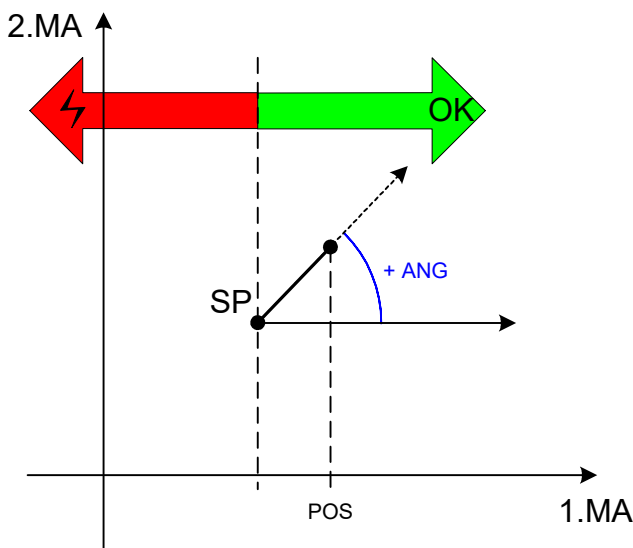
Fig. 16: Validity range of target point

Contour line consisting of 2 straight lines

A contour line consisting of 2 straight lines can be programmed in various combinations of angles and target coordinates. The associated rules are explained in the permitted cases listed below.

Page break

Case 1: Combination of two angles and two target coordinates

The target point ZP1 of the first straight line results from an angle ANG1 and a target coordinate. Based on this, the target point ZP2 of the second straight line also results from an angle ANG2 and a target coordinate. The target coordinates of ZP1 and ZP2 can be programmed as absolute (G90) or relative (G91).
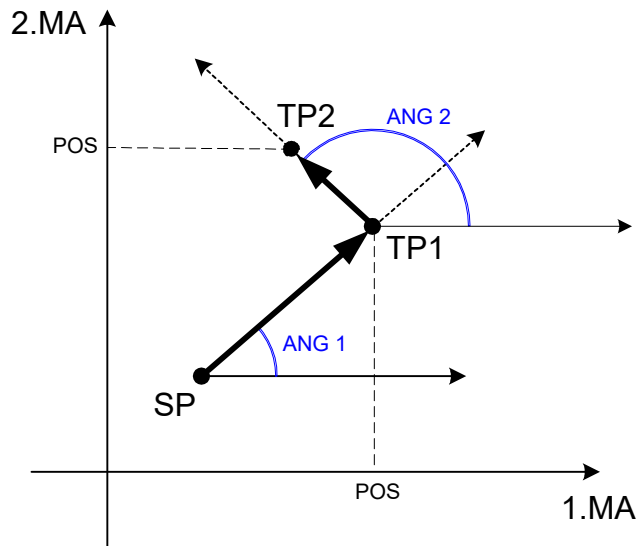
Fig. 17: Contour line with two straight lines (2 angles each with one target coordinate)

Contour line with 2 straight lines in G17 and 2 angles with target coordinates

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=25 X30 ;Straight line 1, target point: X30, Y19.3258
N30 #ANG=120 Y50 ;Straight line 2, target point: X12.2904, Y50
N40 ...
```
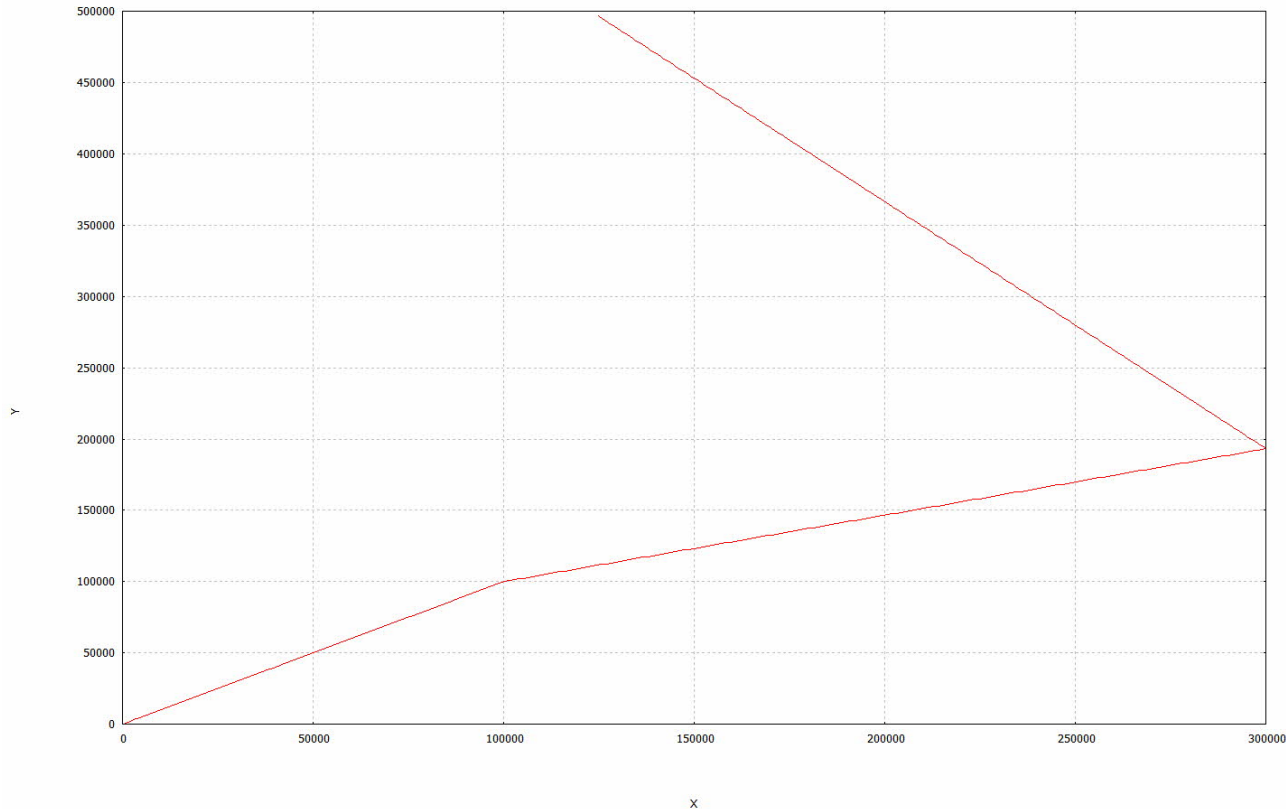
Case 2: Combination of two angles and target coordinate 2

The angles ANG1 and ANG2 are each programmed completely (Cartesian) for the two straight lines and for the second straight line of the target point ZP2. The target point ZP2 must always be specified as absolute (G90). The target point of the first straight line ZP1 can then be determined as the intersecting point of the straight lines.
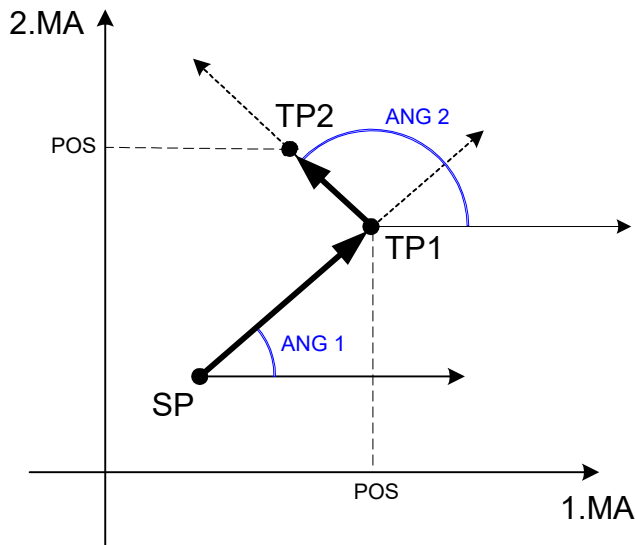
Fig. 18: Contour line with 2 straight lines, 2 angles, complete target point 2

Contour line with 2 straight lines in G17 and 2 angles and complete target point 2

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=15 ;Straight line 1
N30 #ANG=100 Y60 ;Straight line 2, target point 2
N40 ...
```
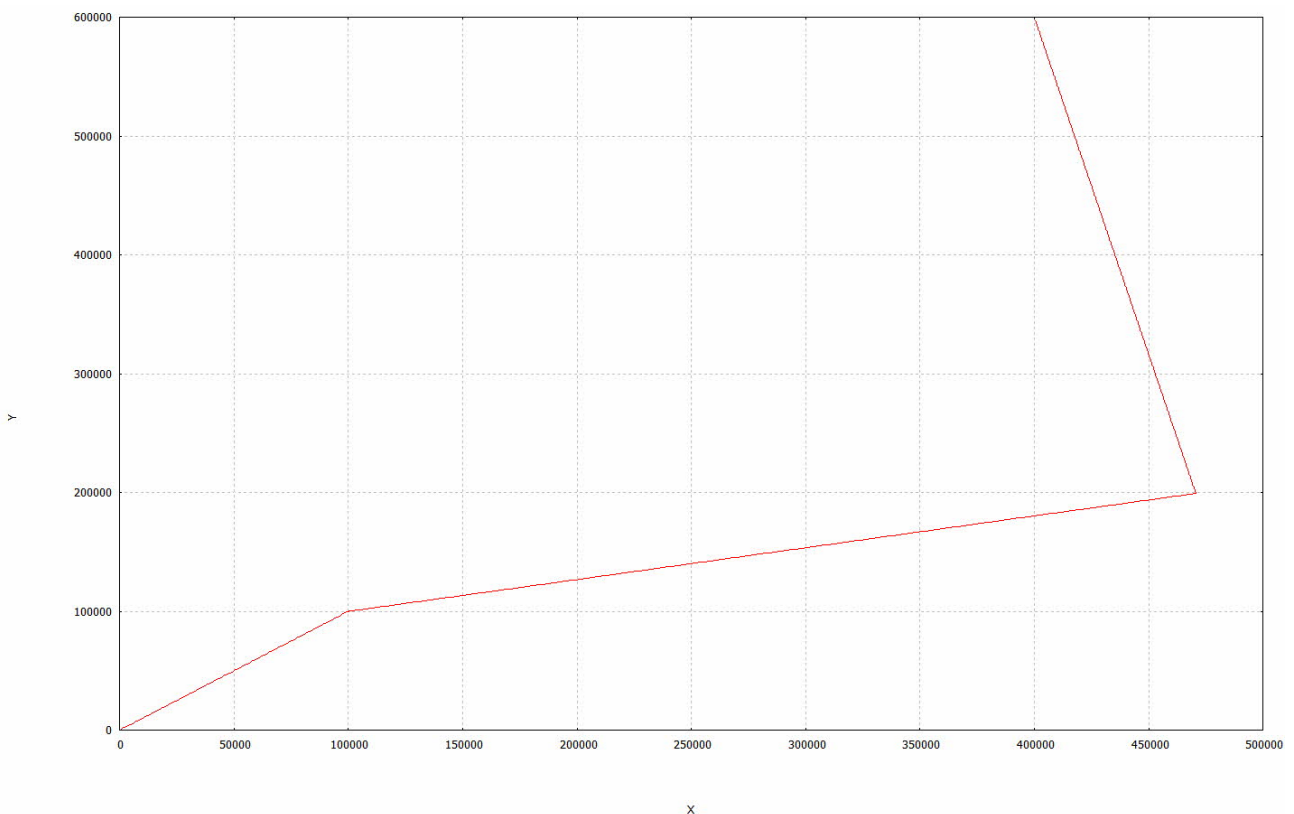
Special case 2-1: Combination of two angles and one target coordinate 2

The angles ANG1 and ANG2 are each programmed for the two straight lines and only one target coordinate of ZP2 for the second straight line. The other coordinate of target point ZP2 results from the associated components of starting point SP. The target coordinate ZP2 must always be specified as absolute (G90). The target point of the first straight line ZP1 can then be determined as the intersecting point of the straight lines.
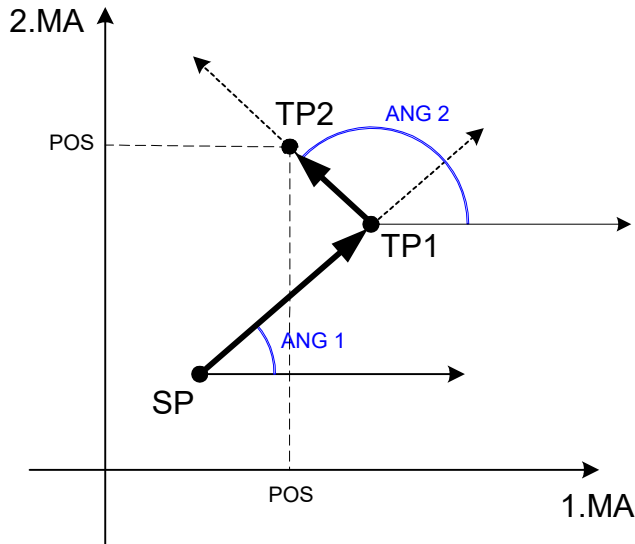
Fig. 19: Contour line with 2 straight lines, 2 angles, incomplete target point 2

Contour line with 2 straight lines in G17, 2 angles and incomplete target point 2

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=75 ;Straight line 1
N30 #ANG=100 Y60 ;Straight line 2, one target coordinate
N40 ...
```
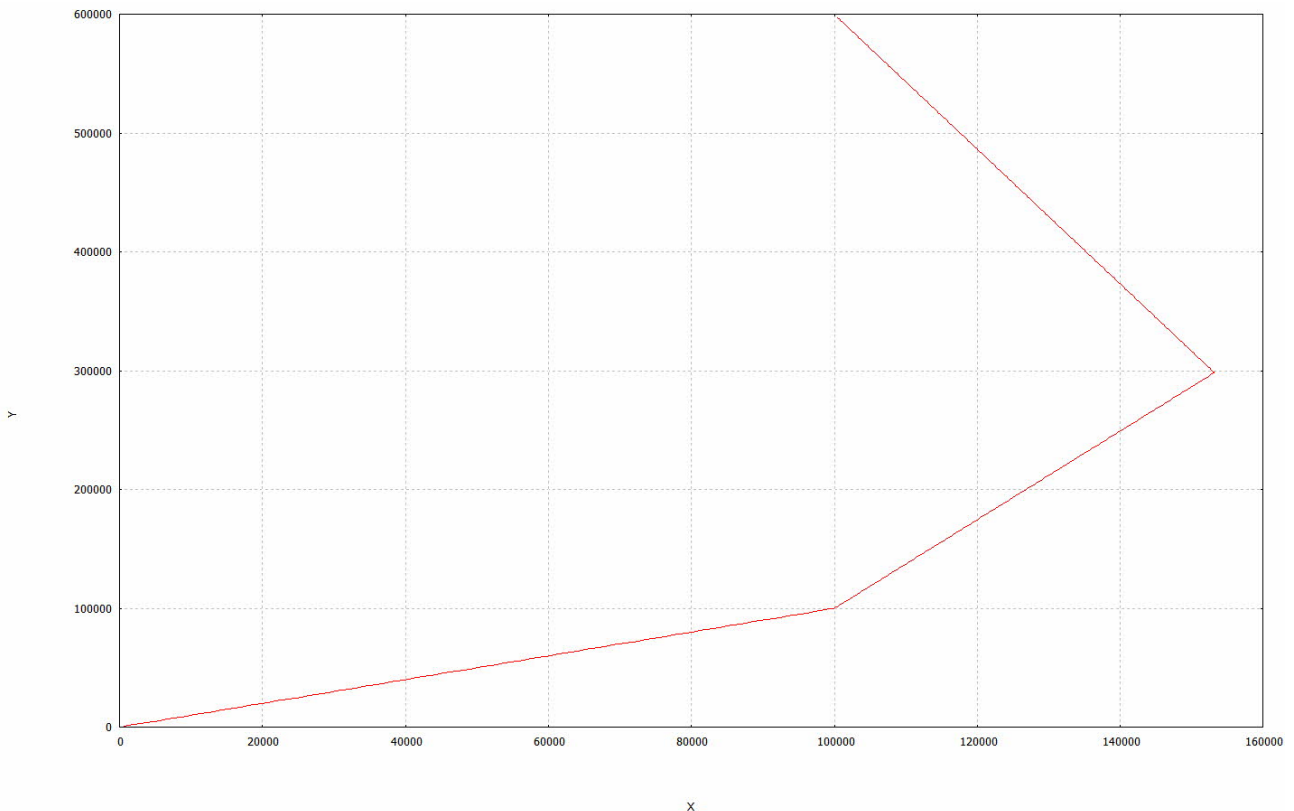
Special case 2-2: Combination of two angles, no target coordinates

If only angles and no target coordinates are programmed, the target points ZP1 and ZP2 are identical to the starting point SP. Only two motions are then possible perpendicular to the current plane.

Validity check of target points:

This checks whether the programmed target points can be reached with the programmed angle starting from the starting point. The orientations resulting from the programmed angles define the valid range for the target points.
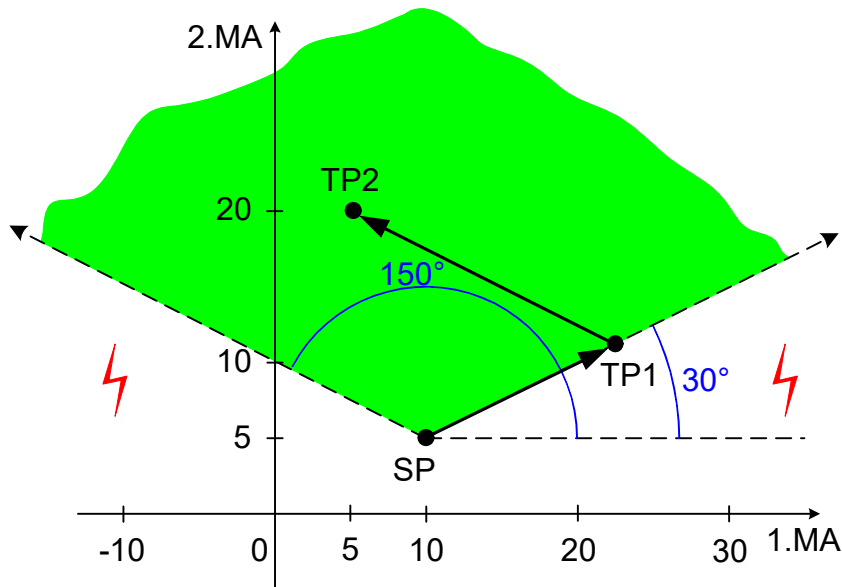
Fig. 20: Validity range of target points with 2 straight lines

Contour line consisting of several straight lines

Any number of straight lines can be connected together to describe a contour. The target points of the straight lines must then be clearly determinable geometrically. The programming rules for a contour line with 2 straight lines must also be complied with for linked contours.

Contour line with several straight lines in G17

```
N020  G17 G90 G01 F2000
N030  X10 Y10 ; Move to start position
N040  #ANG=0 X20
N050  #ANG=90 Y20
N060  #ANG=45 X30
N070  #ANG=135 X20
N080  #ANG=90 Y50
N090  #ANG=180 X15
N100  #ANG=135 Y55
N110  #ANG=225 Y50
N120  #ANG=180 X0
N130  #ANG=270 Y40
N140  #ANG=225 Y30
N150  #ANG=315 Y20
N160  #ANG=270 Y10
N170  #ANG=0 X10
N180 M30
```
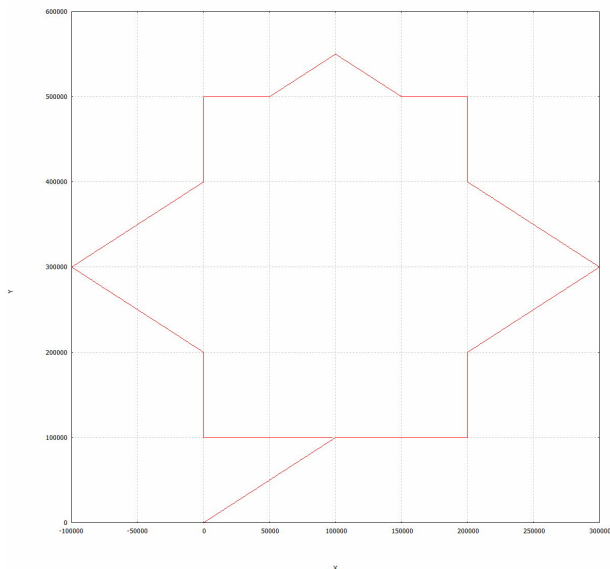
Contour lines in combination with chamfers and roundings

Contour lines can be combined with a complete scope of functions for programming chamfers and roundings (see section Chamfers and roundings [▶ 150]). This is illustrated by the programming example for a turned part below.

Contour line of a turned part with chamfers and roundings

```
N030 G18 G90 G00 X0 Z150
N040 X5 G01 F2000
N050 #ANG=100 #CHR=5 #FRC=1000
N060 #ANG=130 X25 Z140 #RND=5 #FRC=1500
N070 #ANG=90 X40 #CHR=4 #FRC=1000
N080 Z120 #RND=5 #FRC=1500
N090 #ANG=140 X50 #CHR=2 #FRC=1000
N100 Z100
N110 ...
```
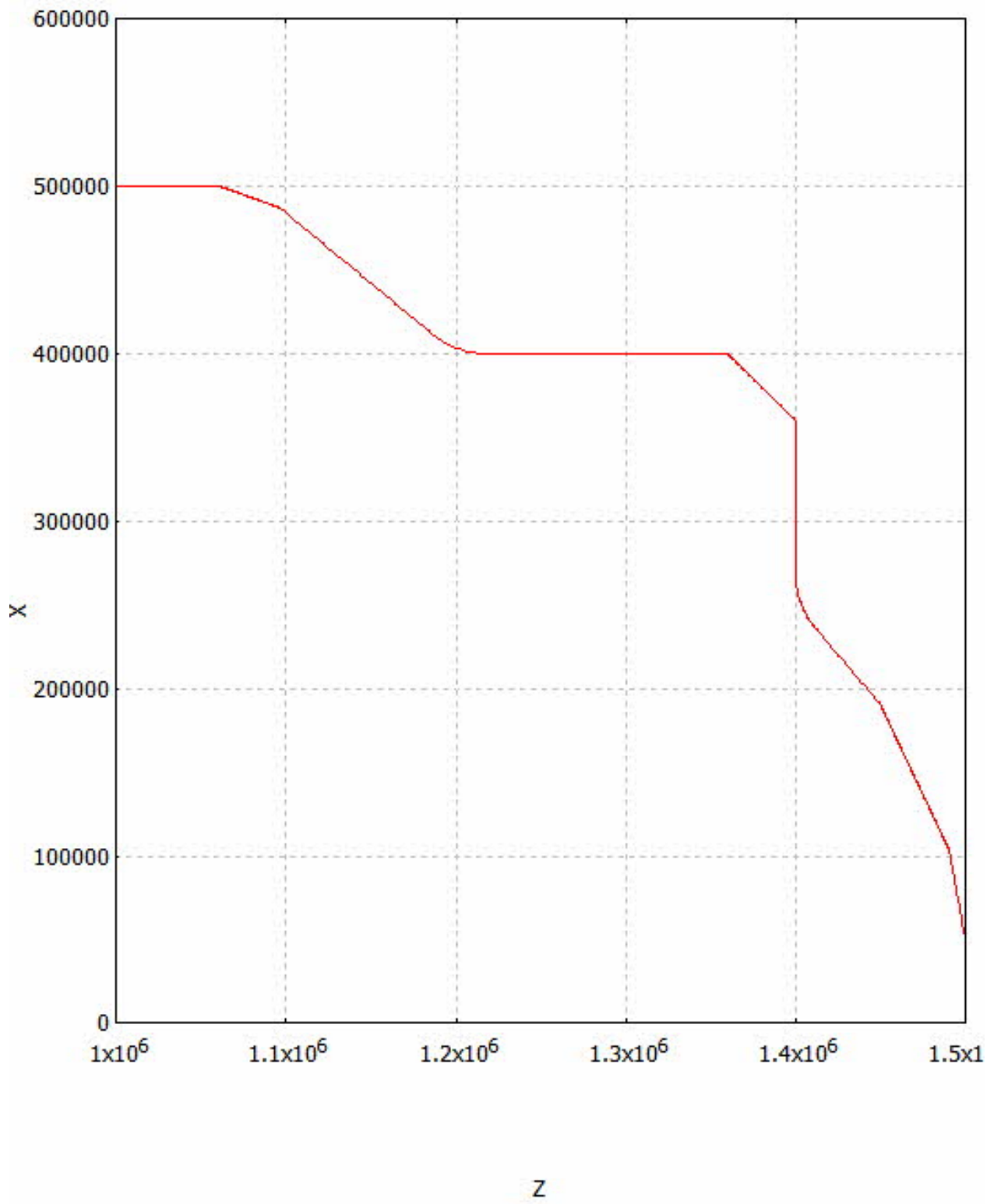
4.1.7 Dwell time (G04), (#TIME)

`Syntax:`

G04 *<1st_main_axis>.. | <time> | <main_spindle>..*   non-modal

G04	Dwell time
<1st_main_axis>..	Dwell time is specified by the name of the <u>1st main axis</u> in [s] or, alternatively...
<time>	… as following direct or parameterised specification [as of Build V2.11.2026.02] of the dwell time in [s] or …
<main_spindle>.	By the name of the <u>main spindle</u> and by specifying a number of revolutions [U] [as of V2.11.2023.02].

Dwell times are required for relief cutting or other machine functions, for example.

Dwell time may only be programmed in the NC block alone (exception: block no.).

Dwell time (G04)

```
N10 G04 X4.5          (wait 4.5 seconds)
N20 G04 3.0           (wait 3.0 seconds)
N30 P1=2
N40 G04 P1         (wait 2.0 seconds)
N50 V.L.TIME=3.5
N60 G04 V.L.TIME       (wait 3.5 seconds)
N70 M3 S200
N80 G04 S10          (wait 10 revolutions (wait 3 seconds))
```

Another possibility to specify dwell time is to use the function #TIME.

`Syntax:`

#TIME *<time>*   non-modal

#TIME	Dwell time plain text command
<time>	Dwell time value direct or parameterised in [s]

Dwell time (#TIME)

```
N10 #TIME 2.5          (wait 2.5 seconds)
N20 P1=2
N30 #TIME P1           (wait 2.0 seconds)
N40 V.L.TIME=3.5
N50 #TIME V.L.TIME      (wait 3.5 seconds)
```

4.1.8   Programmable homing (G74)

```
Syntax:
```

G74	Approach reference point	non-modal

G74 allows the NC program-controlled execution of a homing motion (RPF) which must contain the statements on the axes to be referenced and the sequence in which the axes are to execute the homing motion. The values programmed with axis names define the homing sequence.

For axes with the same value, homing is triggered simultaneously.

Further information on referencing is contained in the functional description of "Homing" [FCT-M1].

Programmable homing (G74)

```
;Sequential commanding:
N10 G74 X2 Y3 Z1      ;Homing sequence: Z-> X -> Y

;Parallel commanding:
N10 G74 X1 Y1 Z1      ;Homing sequence: X,Y,Z simultaneously
```

4.1.9 Reference point offset (G92)

G92   Reference point offset   non-modal

G92 allows a programmable reference point offset in the given axes by a freely programmable value in [mm, inch] (additive reference point offset). Depending on the G90/G91 setting, the currently programmed reference point offset is set absolutely or added to the existing one.

> ℹ **"Non-modal"**
>
> …only applies to G92; of course, reference point offset itself only applies up to the new G92 programming

Reference point offset (G92)

```
N10 G90                 (Absolute dimensional specification)
N20 G92 X10 Z30         (Displaces the programmed and the absolute)
.                       (coordinates by 10 in X, 30 in Z.)
.
.
Nnn G92 X0 Z0           (Reset of reference point offset)
```

> ℹ In addition, the following applies to selecting a reference point offset in G91 mode:
>
> The programming of...
>
> N10 G92 X10 Y20
>
> N20 G0 X0 G91
>
> ... may not cause any movement of the X axis (corresponds to relative movement about 0). The reference point offset is then only effective for an axis when the next motion information is programmed in absolute mode (G90).

4.1.10 Set negative software limit switch (G98)

`Syntax:`

| **G98** | Set negative software limit switch | non-modal |

G98 sets the negative limit switch positions in [mm, inch] in all programmed axes. Depending on the G90/ G91 setting, this may be absolute or additive to the previous software limit switch position.

The positions for negative limit switches are saved in the axis-specific variables:

V.A.-SWE.X,   V.A.-SWE.Y,   V.A.-SWE.Z, etc.

(see also <u>Axis-specific variables [▶ 547]</u>).

● **"Non-modal"**
ℹ …only applies to the command G98; the software limits switches themselves are effective in modal
   mode.

After machine start-up, the default value of axis parameter P-AXIS-00177 is valid first.

The following applies concerning the validity of limit switch positions for all builds of V2.11.20xx and V2.11.28xx:

- The limit value can be further restricted in the NC program by programming but cannot be increased. In other words, the limit value defined in the axis parameter list cannot be increased by G98.

- In static axis constellations (without axis exchange) the limit value changed in the NC program first remains valid at program end and is also effective in the next NC program activated. Only after CNC reset followed by a program restart does the original default value become valid again.

As of Build V3.1.3077.0 the following applies:

- See description in Section "Supplements to G98 and G99"

In dynamic axis constellations (with axis exchange) reset to the original default value is executed when the axis is adopted in the channel.

A G98 change acts on the axis motion path range, on independent axes and on single axes. Relative motion path ranges in manual mode are not affected; they are influenced by the NC command <u>#MANUAL LIMITS [...]</u> <u>[▶ 166]</u>.

Set negative software limit switch (G98)

```
N10 G90
...
N100 G98 X-1000 Y-2000          Sets negative software limit switch
                                in X to -1000 and in Y to -2000
```
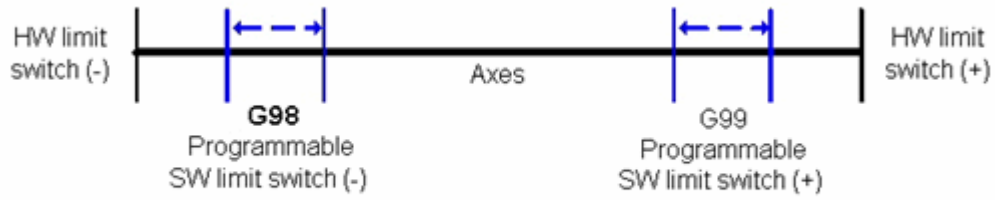
4.1.11   Set the positive software limit switch (G99)

`Syntax:`

G99   Set the positive software limit switch   non-modal

G99 sets the positive limit switch positions in [mm, inch] in all programmed axes. Depending on the G90/G91 setting, this can be absolute or additive to the previous software limit switch position.

The positions for positive limit switches are in the axis-specific variables

V.A.+SWE.X,   V.A.+SWE.Y,   V.A.+SWE.Z, etc.

(see also Axis-specific variables [▶ 547]).

● **"Non-modal"**

i   …only applies to command G99; the software limit switches themselves are effective in modal mode.

After machine start-up, the default value of axis parameter P-AXIS-00178 is first valid.

The following applies concerning the validity of limit switch positions for all builds of V2.11.20xx and V2.11.28xx:

- The limit value can be further restricted in the NC program by programming but cannot be increased. In other words, the limit value defined in the axis parameter list cannot be increased by G98.

- In static axis constellations (without axis exchange) the limit value changed in the NC program first remains valid at program end and is also effective in the next NC program activated. Only after CNC reset followed by a program restart does the original default value become valid again.

As of Build V3.1.3077.0 the following applies:

- See description in Section "Supplements to G98 and G99"

A G99 change acts on the axis motion path range, on independent axes and on single axes. Relative motion path ranges in manual mode are not affected; they are influenced by the NC command #MANUAL LIMITS […] [▶ 166].

BECKHOFF

Set the positive software limit switch (G99)

```
N10 G90
...
N100 G99 X+1000 Y+2000          Sets positive software limit switch
                                in X to +1000 and in Y to +2000
```
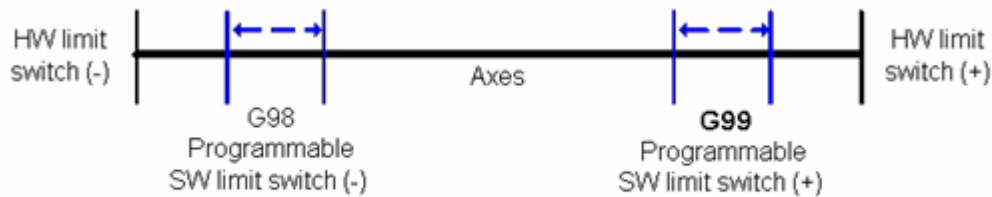

4.1.12   Extensions to G98 and G99

As of Build V3.1.3077.0, limit switch positions referenced to the limits specified in the axis parameter list can also be **extended** by G98 and. This permits a temporary change to an extended section within an NC program and back. The positive limits must continue to be greater than the negative limit. At the next NC program started or after a CNC reset, the configured default values will again a0pp0ly.

The following applies in

- automatic mode: When G98/G99 are programmed before the motion movement, the limited section compared to the configured setting can also be enlarged.

- Manual mode: When G98/G99 are programmed before manual mode is activated,, the limited section compared to the configured setting can be extended:
   ◦ Relative offset limits P-AXIS-00137 and P-AXIS-00138. The new limit acts immediately when manual mode is activated.
   ◦ Absolute offset limits P-AXIS-00492 and P-AXIS-00493: These values are effective if they are entered in the axis parameter list !=0. They can then be set to the maximum configured limit switch positions (P-AXIS-00177, P-AXIS-00178). In this way, the axis cannot move beyond these limits in manual mode, even if the limits are extended by G98 and G99. However, these absolute manual mode limits can be adapted towards the new limit switch positions using CNC objects.

The parameter P-CHAN-00498 (as of Build V3.1.3080.4) sets the limit behaviour. When P-CHAN-00498 is set, it is prohibited to extend the limit.

Increase software limit switch range using G98 and G99

```
;Assuming: Software limit switches are configured to +- 200 in X, Y
N10 G01 G90 X199
...
N100 G98 X-500 Y-500  ;neg. Software limit switches X and Y -> -500
N200 G99 X500 Y500    ;pos. Software limit switches X and Y -> +500
N300 G01 X450 Y450    ;Move within extended section
...
N400 G01 X100 Y100    ;Back to limited section
N500 G98 X-200 Y-200  ;neg. Software limit switches X and Y -> -200
N600 G99 X200 Y200    ;pos. Software limit switches X and Y -> 200
...
```

4.1.13   Measuring functions

After controller start-up, the measurement type specified in the channel parameter P-CHAN-00057 is specified. In the NC program, #MEAS MODE [▶ 332] or #MEAS [TYPE..] [▶ 333] can be used at any time to select a new measurement type.

In the NC program, the variable V.G.MEAS_TYPE [▶ 553] supplies the currently valid measurement type. The following 7 measurement types are available:

Measurement type	Meaning
1*	Measurement run with at least one axis, measuring feed programmable via F word.
2*	Measurement run with exactly one axis, measuring feed is defined in P-AXIS-00215. An error message is output if the probe signal is missing.
3	Measurement run with at least one axis, measuring feed programmable via F word, optionally with continued motion up to the target point.
4	Measurement run only with maximum 3 main axes, measuring feed programmable via F word.
5	Interruptible measurement run with at least one axis, measuring feed programmable via F word.
6	Interruptible measurement run with at least one SERCOS axis, measuring feed programmable via F word.
7*	Measurement run with motion to a fixed stop with at least one axis, measuring feed programmable via F word.

* with these measurement types, a measurement run is also possible using independent axes [▶ 754].

The following variables related to the measurement run are available in the NC program (see also Section Axis-specific variables (V.A) [▶ 547]).

ℹ️   Only the variable values of the axes programmed during the measurement run are updated. All other axis-specific measuring variables retain the old values or 0.

V.A.MERF.*<axis>*	Measurement run completed?
V.A.MESS.*<axis>*	Supplies the axis-specific measuring value in [mm, inch] after a measurement run in which the measurement took place. The value always includes all offsets in the calculation. With 2.5D: ACS values or with CS / TRAFO: PCS values
V.A.MOFFS.*<axis>*	Measuring offset in [mm, inch]
V.A.MEIN.*<axis>*	Measuring offset calculated in [mm, inch]

> **ℹ️ As of Build V2.11.2020.07**

V.A.MEAS.ACS.VALUE.*<axis>*	Measurement value in the axis coordinate system in [mm, inch] including all offsets.
V.A.MEAS.PCS.VALUE.*<axis>*	Measured value in the coordinate programming system in [mm, inch] without offsets.

Restrictions:

A measurement run cannot be programmed if:

- Polynomial contouring (G261, G61) is active
- Spline interpolation (AKIMA, B-Spline) is active
- HSC functions are active

4.1.13.1 Measure with multiple axes (G100) (Type 1)

```
Syntax:
```
G100 *<axis_name>*.. { *<axis_name>*.. } [**F..**]   non-modal

G100	Select measurement run
<axis_name>..	Target point of measuring axis in [mm, inch]
F..	Measuring feed in [mm/min, m/min, inch/min]

Any axis may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 1) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. At least one axis must participate in a measurement run. The measuring feed is specified by the F word. The motion path in the measuring block must be greater than 0.

Measure with multiple axes (G100) (Type 1)

General representation of a measurement run.

```
%G100_Type_1

N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 ; X10/Y20 target point of the measurement run
...
```
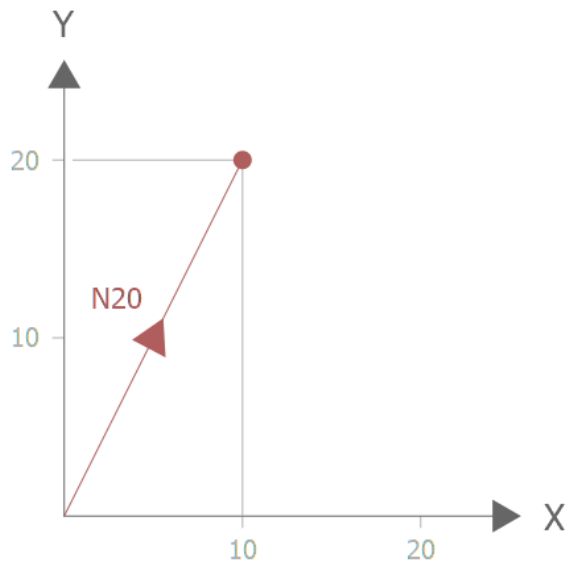
Fig. 21: Programmed measurement run in N20 with measuring function type 1

The program stops after the probe signal is detected. The remaining motion path of the measuring block is no longer output.

BECKHOFF

Measure with multiple axes (G100) (Type 1)

Successful measurement run followed by continuation along programmed path.

```
%Meas_run

N10 G90 G00 X0 Y0 Z0
N20 X5
N30 G100 X10 Y10 F500
N40 G01 X7
N50 M30
```
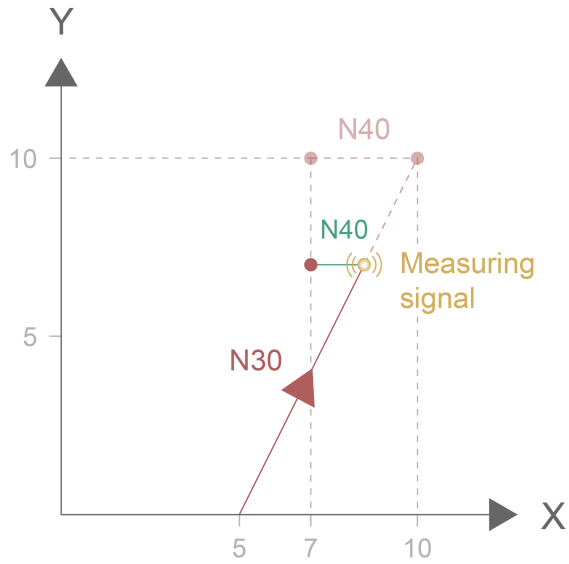
Fig. 22: Programmed path with measuring function Type 1

4.1.13.2 Measure with a single axis (G100) (Type 2)

`Syntax:`

G100 *<axis_name>..*   non-modal

G100	Select measurement run
<axis_name>..	Target point of measuring axis in [mm, inch]

Only one axis may participate in the path motion of the measuring block. The axis programmed in the measuring block must be identified as the measuring axis (P-AXIS-00118). The measuring method (Type 2) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. Exactly one axis must participate in a measurement run. The measuring feed is specified in P-AXIS-00215. The motion path in the measuring block must be greater than zero.

Measure with a single axis (G100) (Type 2)

General representation of a measurement run

```
%G100_Type_2

N10 G90 G00 X0 Y0
X10 Y20 F200 ;X10 target point of the measurement run
...
```


Fig. 23: Program the measuring function Type 2

The program stops after the probe signal is detected. The remaining motion path of the measuring block is no longer output. An error message is output if no probe signal is adopted in the measuring block.

Measure with a single axis (G100) (Type 2)

Successful measurement run followed by continuation along programmed path.

```
%Meas_run
N10 G90 G00 X0 Y0 Z0
N20 Y5
N30 G100 X10
N40 G01 X7
N50 M30
```
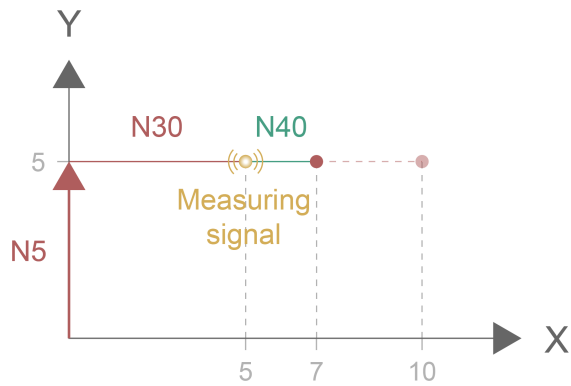

Fig. 24: Programmed path with measuring function Type 2

4.1.13.3 Measure with motion through to target point (G100/G106) (Type 3)

Syntax:

G100 <*axis_name*>.. { <*axis_name*>.. } [**G106**] [**F..**]   non-modal

G100	Select measurement run
<*axis_name*>..	Target point of measuring axis in [mm, inch]
G106	Move up to target point
F..	Measuring feed in [mm/min, m/min, inch/min]

Any axis may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 3) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. At least one axis must participate in a measurement run. The measuring feed is specified by the F word. The motion path in the measuring block must be greater than 0.

Measure with motion up to target point (G100/G106) (Type 3)

General representation of a measurement run.

```
%G100_Type_3

N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 G106 ;X10/Y20 target point of the measurement run
...
```
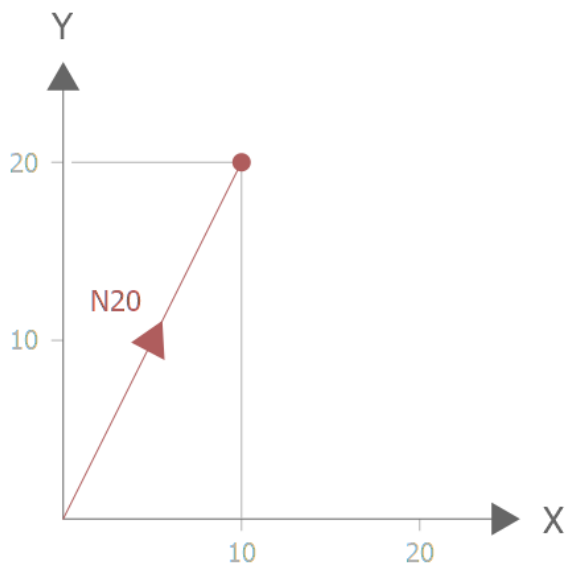

Fig. 25: Programmed measurement run in N20 with measuring function type 3

After a probe signal is detected, the system continues up to the target point of the measuring block if G106 is programmed. If G106 is not programmed, the system decelerates after the probe signal and the remaining motion path is no longer output (same reaction as with measurement type 1).

Measure with motion through to target point (G100/G106) (Type 3)

Successful measurement run followed by continuation along programmed path.

```
%Meas_run

N10 G90 G00 X0 Y0 Z0
N20 G01 X5 F500
N30 G100 G106 X10 Y10 ;After probe signal, move up to target point
```

```
N40 G01 X7
N50 M30
```
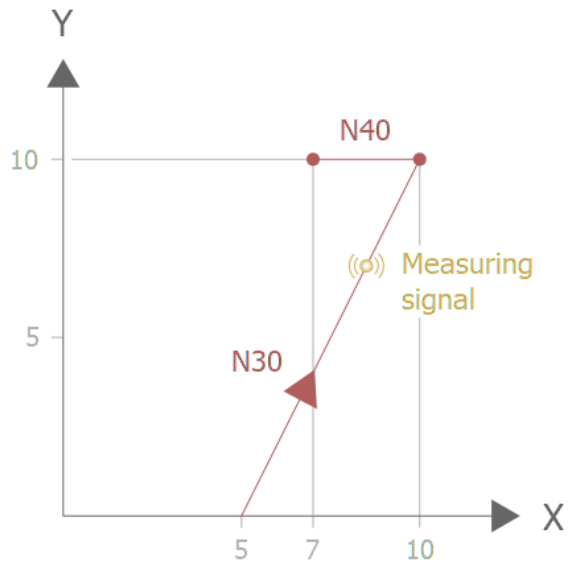

Fig. 26: Programmed path with measuring function Type 3

4.1.13.4 Measure with main axes (G100) (Type 4)

```
Syntax:
```

G100 *<axis_name>*.. { *<axis_name>*.. } [**F..**]   non-modal

G100	Select measurement run
<axis_name>..	Target point of measuring axis in [mm, inch]
F..	Measuring feed in [mm/min, m/min, inch/min]

The three main axes may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 4) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. A maximum of three main axes may participate in a measurement run. The measuring feed is specified by the F word. The motion path in the measuring block must be greater than 0.

Measure with main axes (G100) (Type 4)

General representation of a measurement run.

```
%G100_Type_4

N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 ;X10/Y20 target point of the measurement run
...
```
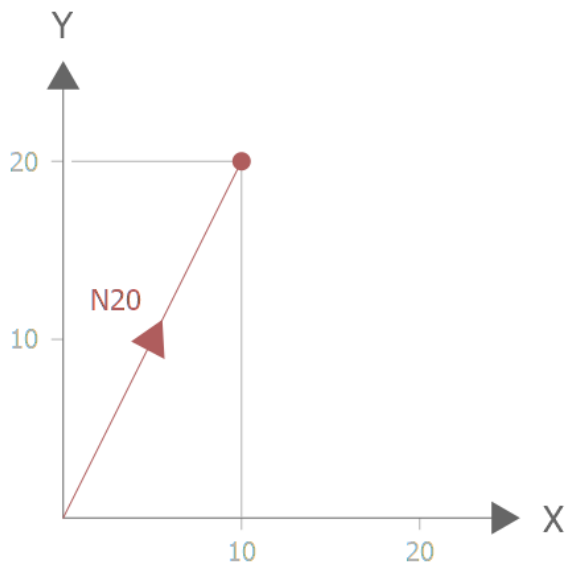

Fig. 27: Programmed measurement run in N20 with measuring function type 4

The program stops after the probe signal is detected. The remaining motion path of the measuring block is no longer output.

Measure with main axes (G100) (Type 4)

Successful measurement run followed by continuation along programmed path.

```
%Meas_run

N10 G90 G00 X0 Y0 Z0
N20 X5
N30 G100 X10 Y10 F500
N40 G01 X7
```

```
N50 M30
```
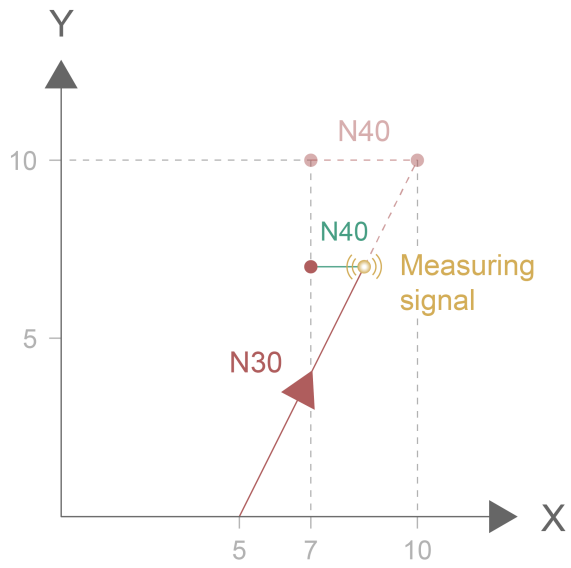

Fig. 28: Programmed path with measuring function Type 4

4.1.13.5   Measure with interruption and jump (G310) (Types 5, 6)

Syntax:

G310 [**G00** | **G01 F..**] *<axis_name>.. {<axis_name>..}* [**$GOTO**<*Label*>]   non-modal

G310	Interruptible block
G00	G01	Interruptible interpolation modes
F..	Measuring feed in [mm/min, m/min, inch/min]
<axis_name>..	Measuring axes with target points in [mm, inch]
$GOTO<*Label*>	Jump target after interrupted measurement run

Any axis may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 5.6) must be parameterised (P-CHAN-00057).

This measuring method offers the option to abort a motion by a probe signal. The path motion must be explicitly programmed in the same block. When the path motion is aborted by the measuring signal, the program branches to the jump target (label) specified in the G310 block. If the probe signal does not occur during the motion block, the NC program is continued with the next NC block.

Measure with interruption and jump (G310) (Types 5,6)

```
N10 G00 X0 Y0
N20 G310 G01 F100 X100 Y200 $GOTO[N_LABEL]
;If interrupted, jump to N_LABEL
N30 G01 X200
N40 $GOTO[ENDE]
N50 [N_LABEL] X0 Y0
N60 [ENDE] M30
```

After the motion is interrupted by a probe signal, the coordinates of the programmed target point are replaced by the actual positions of all the measuring axes in the channel. Next, the logic jumps to the specified block

If no signal is received, the system moves to the programmed target point. So, then there is no jump and the next block is executed instead.

The next block is always executed if a jump target was not programmed.

The current axis positions after an interruption in path motion can be read in the NC program using the V.A.MESS... variable.

If G310 is programmed when TRC (G41/G42) is active, the program is interrupted and an error message is output.

4.1.13.6 Measure with motion to a fixed stop (G100) (Type 7)

When a measurement run is executed with motion to a fixed stop, torque limitation must be activated in all drives involved and any drive-based following error monitor must be disabled.

The measurement run ends as soon as the fixed stop is detected in one of the axes involved in the measurement run.

A programming example and the settings required for a measurement run with motion to a fixed stop are described in greater detail in the functional description "Measurement (C4)" ([FCT-C4]).

4.1.13.7 Calculate measuring offsets (G101/G102)

The measuring offset is the offset depicted in the figure below between the recorded measuring position and the target point. It is calculated as follows:

$$\text{Measurement offset} = \text{measuring point} - \text{target point}$$
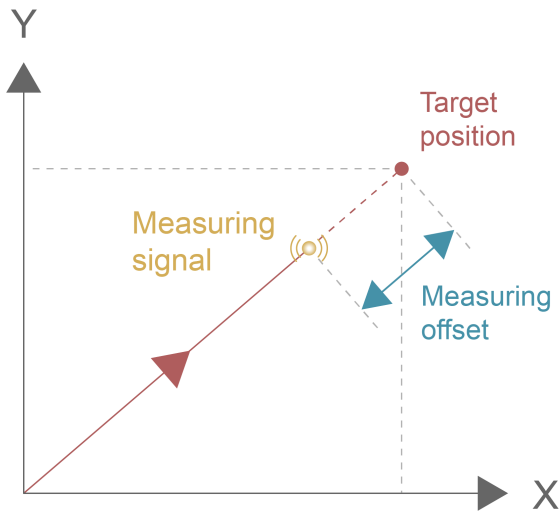
Fig. 29: Measurement offset between probe position and programmed target position

Syntax:	
G101 *<axis_name>*.. { *<axis_name>*.. }	non-modal

G101	Include measuring offset calculation in offset
<axis_name>..	Axis-specific factor to include measuring offset in calculation

> **ⓘ** Several axes can also be specified for measurement type 2 if a separate measurement run was conducted for each of these axes beforehand.

For the programmed coordinates, the measuring offset determined from the measured values is included in the calculation of an additional offset between programmed and absolute coordinates. An error message is output if no measured values were detected beforehand. The numeral after the axis designation represents the inclusion factor.

The offset caused by the measurement offset is valid until it is deselected by G102.

Calculate measuring offsets (G101/G102)

Includes the measuring offset for X in the calculation with factor 1 and for Y with factor 7 in the offset between programmed and absolute coordinates.

```
N10 G101 X1 Y7
```

Syntax:	
G102 { *<axis_name>*.. }	non-modal

G102	Extract measuring offset from offset
<axis_name>..	Axis for which the measuring offset is extracted. Compared to G101, the value behind the axis specified is meaningless but required due to syntactical reasons.

The measured values adopted with G100 and included in the calculation as further offsets with G101 are extracted according to the following rule:

If one or more axes are programmed, only these axes are calculated. If no axis was programmed, all offsets are extracted.

An error message is generated if an axis was programmed for which no measuring offset is included in the calculation.

The offset included in the calculation with G101 is always extracted.

Calculate measuring offsets (G101/G102)

Extract an axis/all axes

```
N10 G102 X1 ;Only extract offset of X-axis
N20 G102    ;Extract offsets of all axes
```

4.1.13.8 Edge banding (G108)

The wood machining and furniture industry requires the edge banding function to perform the exact glueing of veneer strips. When a veneer strip is glued on, the position of the edge start may differ on various workpieces by several millimetres. So, the veneer strip must be cut at a position that is different from the NC block limit. This is a two-dimensional problem in the XY plane and linear and circular motion is allowed.

The start of the veneer strip is measured by a leading probe (see Fig. below). The measured values are used to determine the exact position at which the veneer strip must be cut.

The following response must be achieved:

When the measuring probe is crossed, the interpolator outputs a specified distance to go. The system then waits until all axes are inside the control window. An M function of synchronisation mode MNE_SNS (P-CHAN-00027) is then transferred to the PLC interface to cut the veneer strip. After acknowledgement, interpolation is continued up to the programmed target point. Only one MNE_SNS function may be active, i.e. multiple programming in one NC block is not allowed. But the MNE_SNS function may be programmed in a block using the M functions of other synchronisation modes.
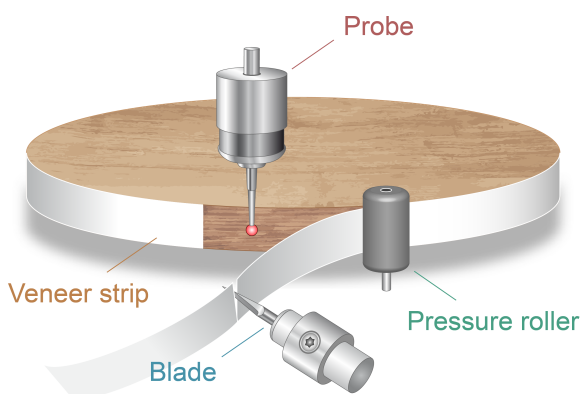
Fig. 30: Glue on a veneer strip

Syntax:		
G108	Edge banding	modal

Edge banding is selected using the G108 function which is modal. Depending on the parameterisation in the channel parameter block P-CHAN-00029, the following 2 methods are possible:

> ℹ️ If only one CNC axis is moved, G108 is also allowed for the special case without main axis motion. In this case the measurement may take place in one of the tracking axes.

4.1.13.8.1 Glue in one motion block (Method 1)

Selecting the measurement using G108 has a modal effect until an M function of synchronisation mode MNE_SNS (P-CHAN-00027) is programmed. The measurement run is started with this M function. The probe signal must occur in the motion block that was programmed together with this M function.

After the probe signal is detected, the system continues moving along the distance to go P-CHAN-00030 and G108 is deactivated implicitly.

Glue in one motion block (Method 1)

```
N05 X0 Y0
N10 G108                    (Activate edge banding)
N20 G01 X90 Y90 F20
N30 M97 G01 X150 Y150 F8    (M97 is of mode MNE_SNS, start measurement run)
                            (Continue with distance to go after probe signal)
M30
```

NOTICE
If no probe signal is detected during the measurement run, an error message is output at block end.

4.1.13.8.2 Edge banding across several motion blocks (G107)(Method 2)

The problem often involves a contour which is described by several short NC blocks (e.g. generated by a CAD system). If edge banding is to occur on this type of contour, the resulting problem is to hit the edge exactly on a contour element.

The edge banding function is also activated by G108 [▶ 101] and has a modal effect.

Special parameterisation in the channel parameter block (P-CHAN-00029) offers the extended function to only execute edge banding n blocks after the M function. The M function itself is also output immediately but the trigger event is only output later. After the probe signal is detected, the system continues moving along the distance to go for a further m blocks. The NC command G107 indicates explicitly the latest point at which the measurement run must end.

Syntax:

| **G107** | Deselect cross-block edge banding | modal |

Glue across several motion blocks (Method 2)

```
N05 X0 Y0
N10 G108                   (Activate edge banding)
N20 G01 X90 Y90 F20
N30 M97 G01 X100 Y100 F8    (M97 is of mode MNE_SNS, start measurement run)
N40 X110 Y110
N50 X120 Y120
N60 X130 Y130
N70 X140 Y140
N80 X150 Y150              (<- last measurement run block!)
N90 G107                   (End of measuring edge banding)
N80 G00 X200 Y200
M30
```

ⓘ The measuring process is also possible in the same motion block. However, method 1 offers simpler programming for this case.

NOTICE
If no probe signal is detected during the measurement run, an error message is output with G107.

4.1.13.8.3 Program distance to go

V.G.RW   Distance to go with edge banding   Read and write access

The axis group-specific "V.G.RW" variable can be used to redefine the distance to go for edge banding in [mm, inch]. The distance to go is the distance remaining after the probe signal is received. In initial state, the corresponding working data point for the distance to go is adopted from the channel parameter block (P-CHAN-00030).

Program distance to go

```
N010 G91 G00 X0 Y0 Z10      (Linear interpolation)
N020 V.G.RW = 5             (Define a new distance to go)
N030 G108                   (Activate edge banding)
N040 G01 X10 Y10 F300       (Linear interpolation)
N050 M97 G01 X30 Y10 F200   (M97 is of mode MNE_SNS,)
                            (Meas. run with lin. interpolation,)
                            (Continue with distance to go after probe signal)
```

NOTICE
The M function for edge banding must be defined as an MNE_SNS mode (P-CHAN-00027).

4.2 Determining acceleration/deceleration (G08/G09/G900/G901)

ℹ️ The acceleration definition operations on only effective in connection with slope types STEP, TRAPEZ and SIN2. The definition operation has no effect with slope type HSC.

`Syntax:`

G08	Acceleration at block start	modal, initial state
G09	Deceleration at block end	non-modal, effect dependent on G901/G900
G901	Deceleration <u>after</u> block end; can be cancelled by G09 blockwise	modal
G900	Deceleration <u>towards</u> block end independent of G09 modal active	modal

If two consecutive NC blocks are programmed at different feedrates, a "soft" adaptation occurs at the block limit. As specified by G08, an acceleration takes place only at block start. G09 specifies that a deceleration to the feedrate of the next block should already occur at the end of the current block.

G08 and G09 are the default settings at program start. The G901 function defines the default setting for deceleration only after block end. The inverse G function G900 is a change-back function and is equivalent to the channel default setting.

<table>
<tr><td align="center">NOTICE</td></tr>
<tr><td>During transition from G00 to G01, G02 or G03, G09 is always active, i.e. deceleration at block end down to the velocity of the following block.

If the path velocity limit is already achieved while G901 is active, the velocity of the following block is reached at block end, i.e. then G901 is not effective.</td></tr>
</table>

Define acceleration (G08/G09/G900/G901)

Acceleration at block transition in initial state (equiv. to G08).

```
N10 G01 X500 F400
N20     X900 F1000
```
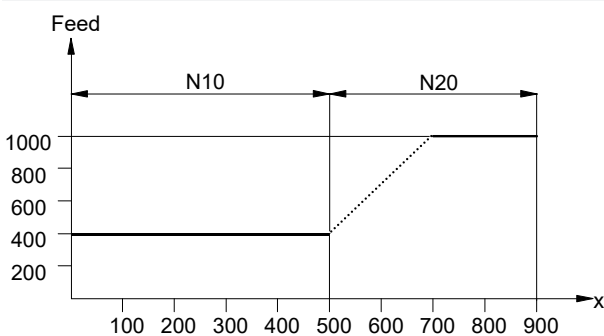

Fig. 31: Acceleration at block transition in the default state (corresp. to G08)

Define acceleration (G900/G901)

Deceleration at block transition with G901 and G900

```
N10 G01 G901 X500 F1000
N20          X900 F400
```
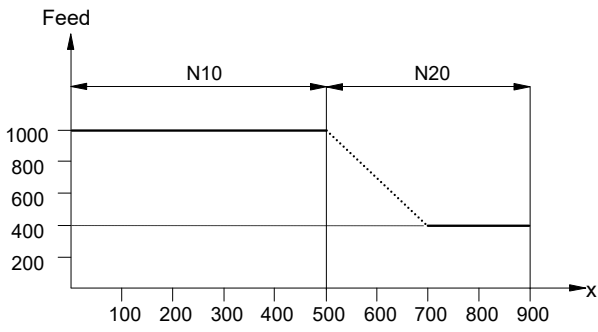
Fig. 32: Deceleration at block transition with G901 and G900

```
N10 G01 G900 X500 F1000
N20          X900 F400
```
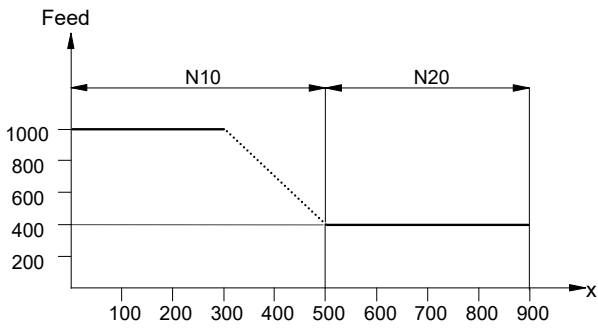

Fig. 33: Deceleration at block transition with G901 and G900

Define acceleration (G900/G901)

Combination of G09 with G901 and G900.

```
N10 G01 G901 X200 F2000
N20     G09  X400 F1600
N30          X600 F1200
N40     G900 X800 F800
N50          X1000 F400
:
```
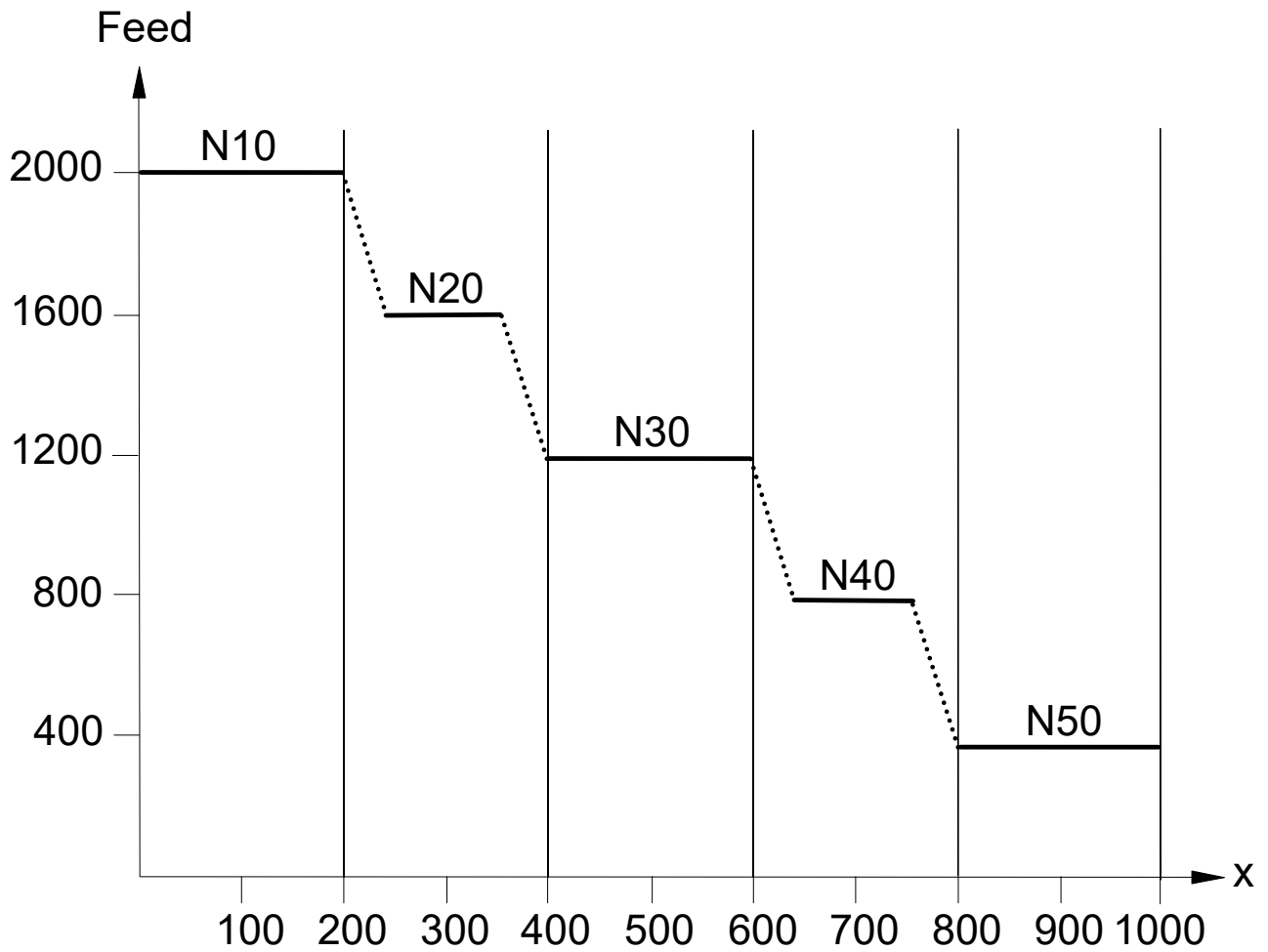

Fig. 34: Combination of G09 with G901 and G900

4.3 Path/time-related feed interpolation (G193/G293)

ℹ️ Path-related feed interpolation is only effective in combination with linear slope (#SLOPE [TYPE=STEP]) or HSC slope (#SLOPE [TYPE=HSC]).

Time-related feed interpolation is only effective in combination with linear slope (#SLOPE [TYPE=STEP]).

Syntax:

| **G193** | Path-related feed interpolation | non-modal |
| **G293** | Time-related feed interpolation | non-modal |

When G193/G293 is selected, the feedrate between the initial and the programmed end velocity is linearly interpolated.

ℹ️ It is not permitted to program G193 with G293, G08 or G09 in the same NC block.

Path-related feed interpolation with (G193)

```
N10 G01   X500 F1000
N20 G193  X900 F400
N30 X1000 F400
```

At block transition N10/N20, path-related feed interpolation is activated and decelerated linearly to end velocity across the path programmed in N20.
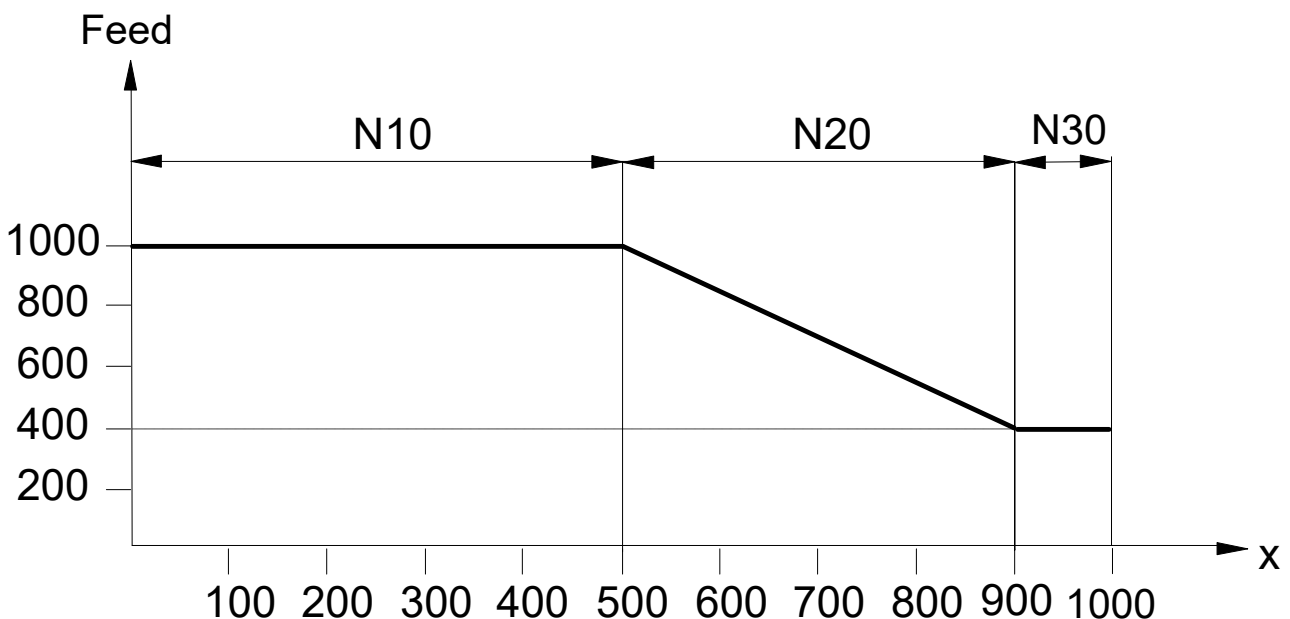
Fig. 35: Path-related feed interpolation with G193

Override changes are superimposed taking account of the permissible axis dynamics.

Path-related feed interpolation with (G293)

```
N10 G01   X500  F1000
N20 G293  X900  F400
N30       X1000 F400
```

Time-related feed interpolation is activated at block transition N10/N20 and linearly decelerated over time to end velocity.
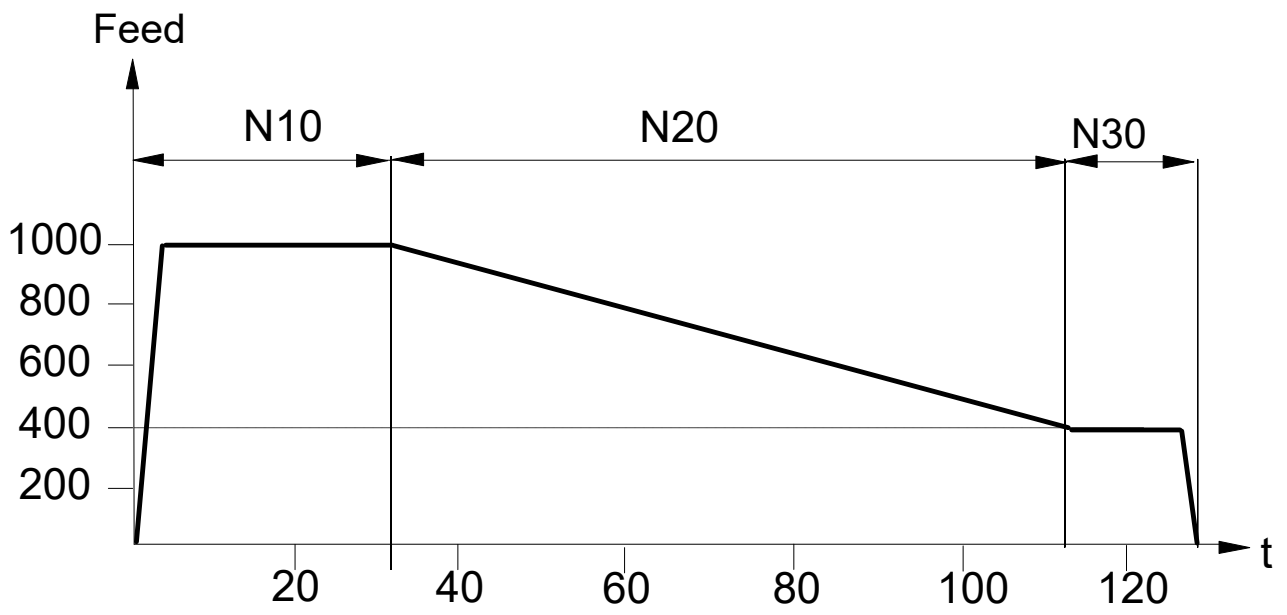
Fig. 36: Time-related feed interpolation with G293

4.4 Selection of planes (G17/G18/G19)

Syntax:

G17	X-Y plane	modal, initial state
G18	Z-X plane	modal
G19	Y-Z plane	modal

Programming G17, G18 or G19 defines the plane in which tool radius compensation and circular interpolation (see Circular interpolation (G02/G03) [▶ 56]) are to act:
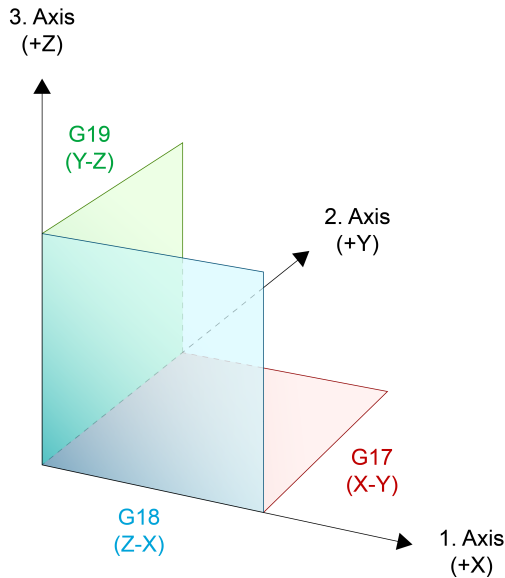
Fig. 37: Display of plane selection (G17/G18/G19)

Tool radius compensation:

Please note that tool radius compensation is always active on the first two main axes.

Circular interpolation:

After a plane change with G17, G18 or G19, assignment again applies as per DIN: X, Y, Z are then assigned to I, J, K. The table below illustrates the syntax according to the selected interpolation plane:

Plane	Interpolation type	Target point in plane	Centre point/radius
G17	G02/G03	X..Y..	I..J../R
G18	G02/G03	Z..X..	K..I../R
G19	G02/G03	Y..Z..	J..K../R

4.5 Mirroring in the plane (G21/G22/G23/G20)

The term "virtual coordinates" is used to mirror in the plane. When mirroring is executed, the virtual coordinates (x_m) are the values entered in the NC program. By contrast, the real coordinates (x_r) are mirrored and are executed in reality. Mirror functions always act on the first and second main axes of the current plane (G17, G18, G19). The third main axis of the current plane is not mirrored.

`The following mirroring functions are available (illustrated example for G17, X-Y plane):`

G21   Mirroring programmed paths on the X axis   modal

$x_m = -x_r$

$y_m = y_r$

G22   Mirroring programmed paths on the Y axis   modal

$x_m = -x_r$

$y_m = y_r$

G23   Superimpose G21 and G22   modal

$x_m = -x_r$

$y_m = y_r$

G20   Deselect the mirroring function   modal
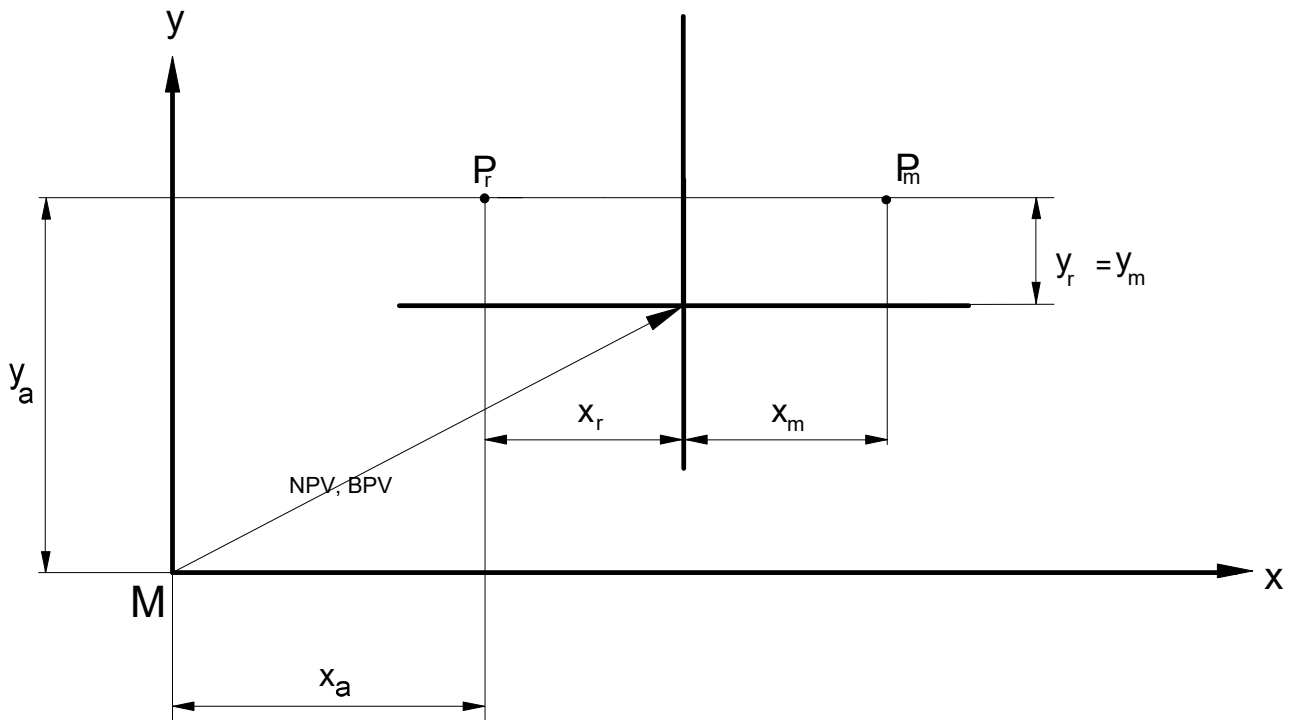
Fig. 38: Virtual and mirrored (real) coordinates with G21

Contours to be mirrored should be specified in a subroutine. This subroutine is then called after the mirroring function. Mirroring is modal.

> **ℹ** If mirroring is programmed when tool radius compensation is selected, the side of the compensation also changes. This means that, when G41 is active, the equidistance to the programmed path after mirroring is calculated on the right of the contour (G42: TRC right of contour). This also happens even if the path direction is changed.
>
> Tool offsets and shifts (e.g. G54, G92, #PSET...) are not mirrored with G21, G22 and G23.
>
> By contrast, the reference point offset (G92) is mirrored with the mirror function G351 [▶ 115].

Mirroring in the plane (G21)

```
%L DREIECK            (Subroutine)
N10 G90 X10 Y20       (Define the mental)
N20 G91 X10 Y-10      (Coordinates to be mirrored)
N30 X-10 Y-10
N40 Y20
N50 M29

%SPIEGELUNG           (Main program)
N10 G92 G90 X60 Y40   (Reference point offset)
N20 G01 F500          (Straight line at feed 500)
N30 G21               (Mirroring in the X axis)
N40 LL DREIECK        (Call subroutine)
N60 G92 G90 X0 Y0     (Reset the reference point offset)
N70 M30
```
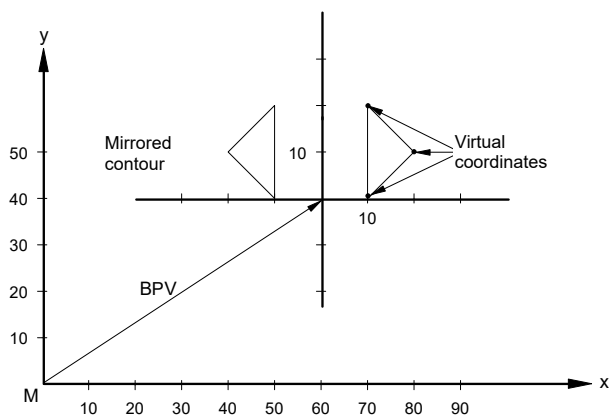

Fig. 39: Example of mirroring

NOTICE
Note the following when circles are mirrored:
In general, only coordinates set in the NC program are mirrored (i.e. only the centre point coordinates for a full circle). After selecting the mirroring function, the motion path runs directly to the target point. The starting point of the motion path is not mirrored.
Consequence: In the motion block only the target point is mirrored but not the complete motion path (see Fig.: * Mirroring the target point in motion block")
Effect: If a full circle is programmed as a motion block and the starting/target points are not in 0/0, a new circle radius results from mirroring the centre point coordinates and the contour is changed (see Fig.: "Changing the contour when mirroring a full circle").

Effect:
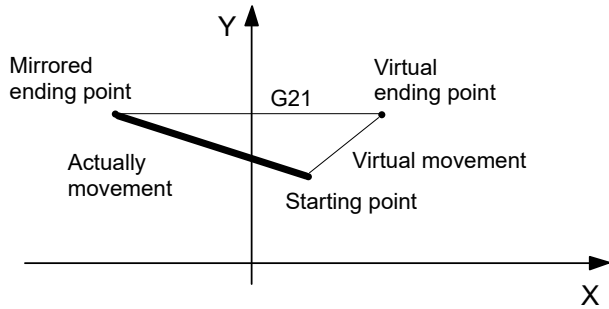
Fig. 40: Mirroring the target point in the motion block.
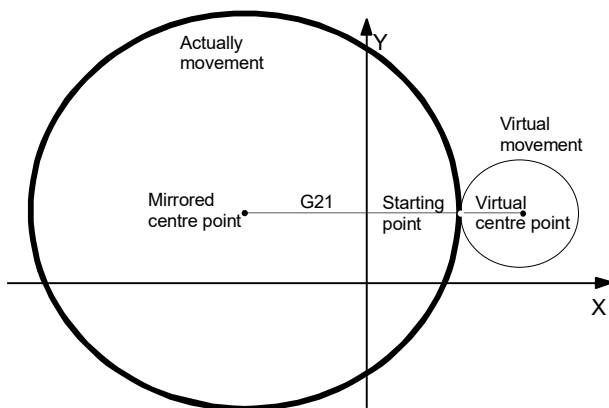
Fig. 41: Changing the contour when mirroring a full circle.

3D considerations:

If the mirroring functions are selected in a different plane than the XY plane (G18, G19), note that the direction of rotation for circular interpolation may change depending on the selected plane: In addition, mirroring has no impact on the Z coordinate, i.e. the virtual and mirrored values in the Z direction are always identical.

If one views the ZX-Plane (view direction parallel to the Y-Axis), then the movements of the Modi G21 and G23 as well as G22 and G20 are one over the other.

If one views the YZ-Plane (view direction parallel to the X-Axis), then the movements of the Modi G21 and G20as well as G22 and G23 are one over the other.
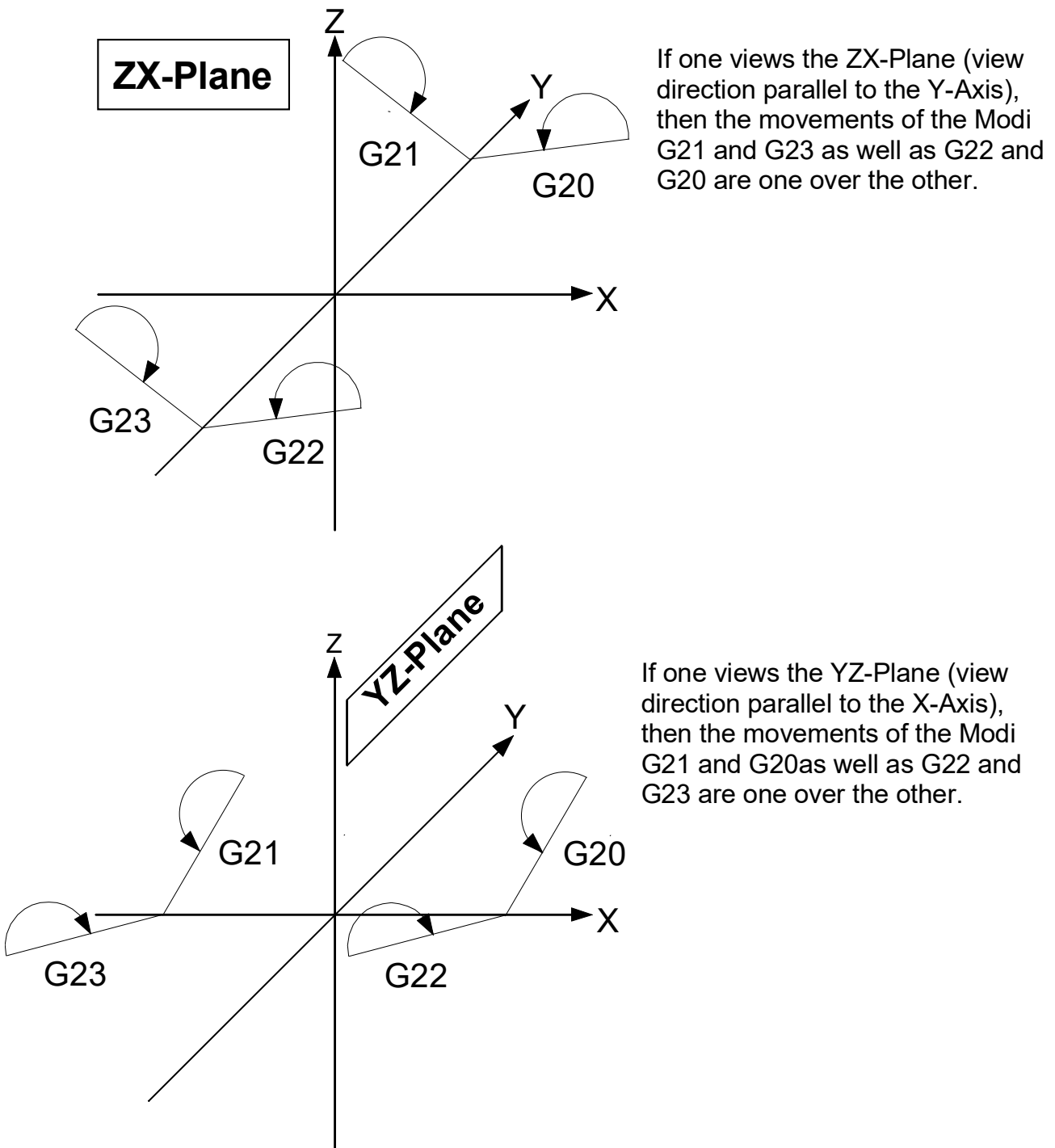
Fig. 42: Effects of mirroring functions on the direction of circular rotation in different planes

4.6 Mirroring with axis specification (G351)

G21 to G23 only select mirroring only for the first two main axes. The syntax with G351 described below permits free programming for axis mirroring.

Syntax:

G351 *<axis_name>* [[**+**] | **-**] **1** { *<axis_name>* [[**+**] | **-**] **1** }   non-modal

G351	Axis-specific selection of mirroring. The G function G351 is only valid as non-modal. However, mirroring is modal for an axis programmed with this function (modal).
<axis_name>..	The axis coordinate defines whether to select or deselect mirroring in the axis.

Coordinate value -1:	Selects mirroring
Coordinate value 1 or +1:	Deselect the mirroring function

Mirror with axis specification (G351)

```
G351 X-1 Y1 Z+1    (Select mirroring in the X axis and deselect mirroring)
                   (in the Y and Z axes)
```

- The axes mirrored can be programmed at any point in the NC block.

- At least one axis coordinate must be programmed together with G351.

- The G351 function must be programmed alone in the NC block. An exception is block number N.

- Repetitive selection or deselection of axis mirroring is permitted. However, an error message is output if repeated programming takes place in the same NC block.

- If mirroring is selected in synchronous mode for the lead axis (master axis), mirroring is not automatically selected for the tracking axis (slave axis). However, the slave axis is always tracked according to the path motions of the master axis. Therefore, an additional path motion resulting from mirroring the master axis always influences the slave axis.

- Mirroring is deselected for all axes at program start and reset. When axes are changed, mirroring of the changed axis is deselected.

- Mirroring the first or second main axis influences the path direction during circular interpolation and tool radius compensation.

- When mirroring is programmed when tool radius compensation is active, the selected side (G41/G42) is swapped automatically. This is only allowed for linear blocks.
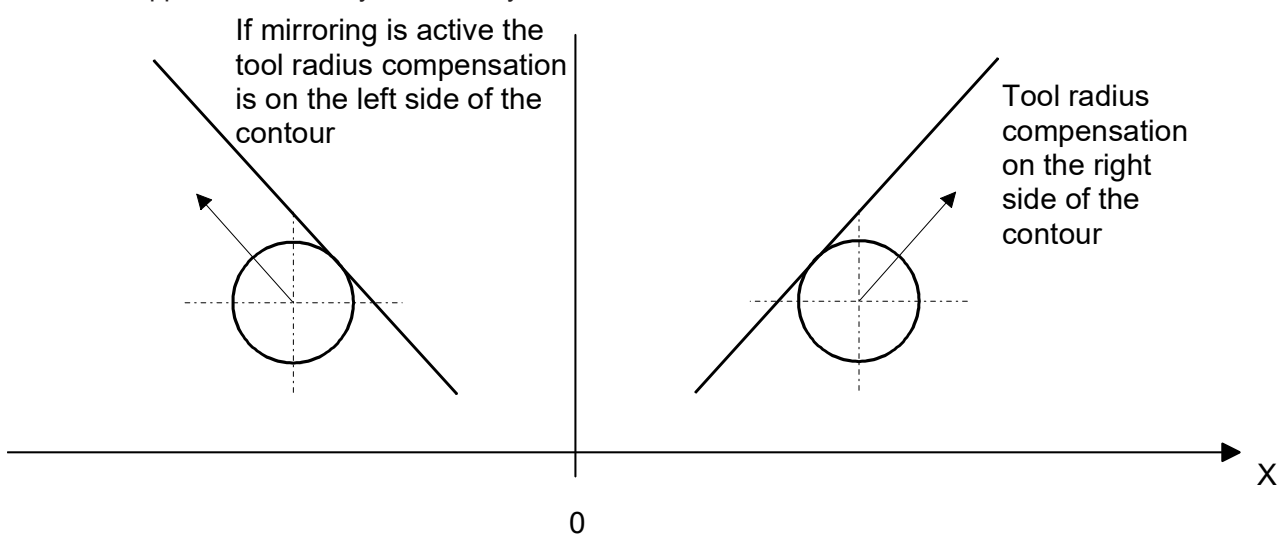
Fig. 43: Mirroring the selected side with active tool radius compensation

- If a reference point offset G92 is active in an axis mirrored with G351, the coordinates of the reference point offset are also mirrored.
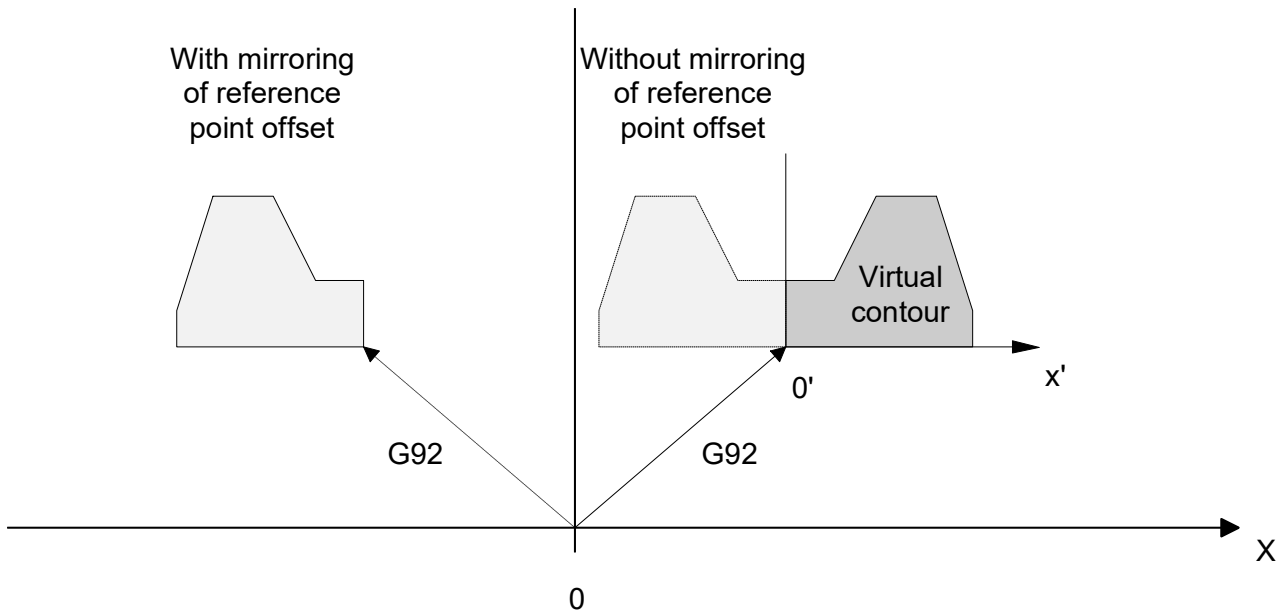
Fig. 44: Mirroring a reference point offset G92

- The coordinates of the circle centre point I, J, K are also mirrored (see Section Plane selection (G17/ G18/G19) [▶ 110])
- When chamfers and roundings (G301/G302) are inserted, the I word is read as chamfer length or as radius. Therefore there is no need to consider mirroring here.

Mirror with axis specification (G351)

The examples below show how to use the G351 function. Assuming that the axes X, Y and Z are the 1st, 2nd and 3rd main axes.

```
N10 G351 X-1         (Select mirroring in the X axis (G21))

N20 G351 Y-1         (Select mirroring in the Y axis (G22))

N30 G351 X-1 Y-1     (Select mirroring in the X and Y axis (G23))

N40 G351 X1 Y+1      (Deselect mirroring in the X and Y axes (G20))

N50 X1 G351 Y-1 Z1   (Select mirroring in the Y axis and deselect)         (mirroring
in the X and Z axis)
```

4.7   Units (G70/G71)

`Syntax:`

G70	Inputs in inches (mm, inch)	modal
G71	Inputs in metric units	modal, initial state

The statement G70 or G71 acts on all path and coordinate values.

Exception: Reading/writing tool parameters (V.G.WZxx.Pxx.) and kinematic parameters (V.G.WZxx.KIN_PARAMxx) always act directly regardless of the current unit.

4.8   Implicit subroutine calls (G80–G89/G800..)

`Syntax:`

G80 – **G89** [[<Val1>,<Val2>, - ,<Val50>]]	Subroutine call	non-modal
or in addition		
G800 -G8xx [[<Val1>,<Val2>, - ,<Val50>]]	Subroutine call	non-modal

G80-G8x / G800–G8xx	When programming G80–G89, G800– G80–G89 and G800-G839**, an assigned global subroutine is implicitly called and executed. The default names of these subroutines can be configured either in the channel parameters P-CHAN-00160 - P-CHAN-00169 and P-CHAN-00187 or defined during program runtime using the command #FILE NAME [▶ 413].
	If no program name is saved when G80–G89, G800–G819 and G800-G839** are programmed, the error message ID 20131 "Unknown G function" is generated. The global subroutine is called only once; this means G80–G89, G800–G819 and G800-G839** have no modal effect.
<Val1>, - ,*<Val50>*	Optionally, a maximum of 50 transfer parameters (mathematical expressions in REAL format) can be bracketed in **a fixed sequence** to supply a subroutine (cycle). The parameters are separated by commas. Gaps in the sequence must be marked by consecutive commas ", ,".
	By specifying transfer parameters, the subroutine call is handled as a cycle call according to the rules for cycles.
	In analogy to cycle programming, the parameters can be read out in the subroutine using @Px accesses. There is a fixed assignment between the parameter and the @Px read access (e.g. @P1 reads parameter value 1, @P2 reads parameter value 2 and so on). Subroutines called in this way can also use the additionally extended cycle syntax with the @ character. The variable V.G.@P[i].VALID [▶ 553] in the subroutine (cycle) determines a parameter is programmed (valid).

**Extended to 40 calls (G800 – G839) as of V3.1.3079.23

A G80–G89 and G800.. is always executed as the last action at block end. This means that axis motions are programmed in the same NC block. They are always executed before the global subroutine is called.

Implicit subroutine calls (G80–G89/G8xx)

Assume the global subroutine g80_cycle.nc is called for G80:

```
N10 #FILE NAME[ G80="g80_up_test.nc" ]
Nx ..
N30 G80        Call g80_up_test.nc as global subroutine
:
Assume the global subroutine g80_up_test.nc is called for G815:
..
N10 #FILE NAME[ G815="g815_up_test.nc" ]
Nx ..
N30 G815        Call g815_up_test.nc as global subroutine
:
G85 calls the global subroutine cycle_test.nc with parameters:
N10 #FILE NAME[ G85="cycle_test.nc" ]
Nx ..
N30 G85 [10,2, ,15,-3, ,5]     Call cycle_test.nc as global subroutine
```

Example 2:

```
G803[5, @P1, @P2, @P3]
```

The meaning of the line above:

```
G803[@P1=5, @P2=@P1, @P3=@P2, @P4=@P3]
```

The result: all transferred parameters have the same value: 5

4.9 Dimension systems (absolute dimension/incremental dimension) (G90/G91)

```
Syntax:
```

G90	Absolute dimension	modal, initial state
G91	Incremental dimension	modal

With an absolute dimensional input (G90), all coordinate specifications are based on the coordinate origin, i.e. the coordinates in a motion block specify the target point in the coordinate system.

With incremental programming (G91), the coordinate values are based on the target point of the preceding motion block, i.e. the coordinates in a motion block specify the path to be travelled.

4.9.1 Exclusive programming

In the default setting, only one measuring system may be selected in the NC block. It is not permitted to use redundant programming or programming of G90 and G91 in the same NC block. The position of G90/G91 within the NC block then has no meaning.

Exclusive programming

```
:
N10 X10 Y10 (Absolute measuring system G90 is selected, basic setting)
N20 G91 X20 Y20 (Deselect absolute and select relative programming)
N30 X30 G90 Y30 (Deselect relative and select absolute programming)
:
N100 G90 Z30 G90    (Error message: redundant programming of G90/G91)
N110 G91 X10 G90 Z30 (Error message: redundant programming of G90/G91)
```

4.9.2 Combined programming

Channel parameter P-CHAN-00116 can be used to deselect the exclusive measuring system programming for <u>path axis</u> coordinates. Then it is possible to program absolute and relative measurements in the same NC block. It is also permitted to use the repeated programming of G90 and G91 in the same NC block.

The position of G90/G91 within the NC block then has a meaning. The measuring system last programmed is valid for all following path axis positions in the NC block and all other NC blocks up to the next G90/G91.

Combined programming

```
:
N10 X10 Y10 (Absolute dimensioning G90 is selected, default)
N20 G91 X20 G90 Y20 (relative for X axis, absolute for Y axis)
N30 X30 G91 Y30 Z20 (absolute for X axis, relative for Y/Z axis)
N30 G90 X30 G91 Y30 G90 Z20 (absolute for X/Z axis, relative for Y axis)
:
N100 G90 G91 Z30 (relative for Z axis)
N110 G91 X10 G90 Z30 (relative for X axis, absolute for Z axis)
```

> ℹ️ G90/G91 has no influence on the auxiliary coordinates I, J, K of circular or helical interpolation. Its measuring system is defined exclusively by G161/G162.

4.10   Exact stop (G60/G360/G359)

`Syntax:`

G60	Exact stop	non-modal

... or for exact stop across several blocks:

G360	Selecting exact stop	modal
G359	Deselecting exact stop	modal

G60/G360 allow the exact approach to a target point within the exact stop limits. The feedrate is decreased to zero up to block end and decreases the following error.

The exact stop can be used if edges must be precision-machined or if the target point must be precisely reached in case of direction reversal. The motion is continued to the next position when a parameterisable control window is reached (P-AXIS-00236).

Examples of exact stop

```
N10 G1 F1000 X0 Y0
N20 G1 X10
N30 G60          ( Exact stop for this block end )
N40 G1 Y20
N50 G60          ( Exact stop for this block end )
N60 G1 X30
;...
```

By analogy, implementation with G360 and G359

```
N10 G1 F1000 X0 Y0
N20 G1 X10
N30 G360         ( Select exact stop )
N40 G1 Y20

N60 G1 X30
N70 G359         ( Deselect exact stop )
;..
```

4.11 Polynomial contouring (G61/G261/G260)

```
Syntax:
```

| **G61** | Polynomial contouring (at block end) | non-modal |

... or for polynomial contouring across several blocks:

| **G261** | Selecting polynomial contouring (at block end) | modal |
| **G260** | Deselecting polynomial contouring | modal |

Polynomial contouring is the constant curvature and constant direction connection between two motion blocks. For this purpose, the originally programmed motion blocks are shortened. A contouring curve is added between the blocks. This process permits contouring between the transitions straight line - straight line, straight line - circle and circle – circle (see Figure below). It is not restricted to a particular plane but allows contouring between any number of curves located in space.

Motion blocks without motion path are not considered here. (See Relevant block length [▶ 256]).
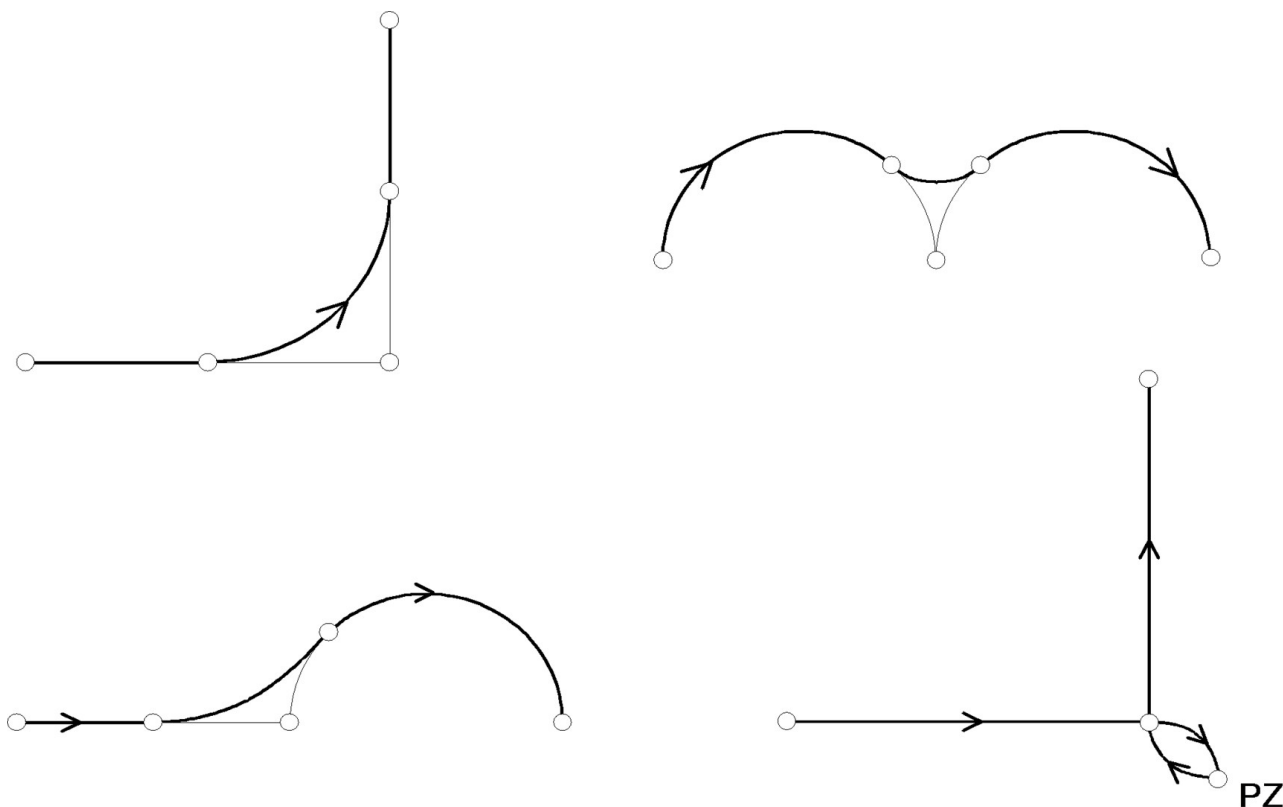
Fig. 45: Examples of polynomial contouring

The following contouring types are available:

- Contouring with corner deviation
- Dynamic optimised contouring
- Dynamic optimised contouring with master axis
- Contouring with interim point
- Dynamically optimised contouring of the contour.

Depending on the contouring type, different conditions may be specified. The parameters remain valid until the contouring process is fully executed. If the contouring parameters between the pre-block and post-block are changed, the change becomes effective with the next contouring operation.

For more information, see section Polynomial contouring for long blocks (G61/G261/G260) [▶ 252].

4.12   Corner deceleration

In the milling process, the removal volume V_z and the associated spindle performance required in the inside corners increases as a function of block transition geometry since the tool already removes material from the subsequent contour (see figure below).

If machining is always performed at the spindle's performance limit, the feedrate in the corner section must be reduced so that there is also sufficient spindle performance in the inside corners. In order to stay within the spindle's performance limits, it must be possible in the NC program to define a point on the path after which the velocity can be reduced. There are three NC commands available for this corner deceleration: one for parameterisation, one for activation and one for deactivation.
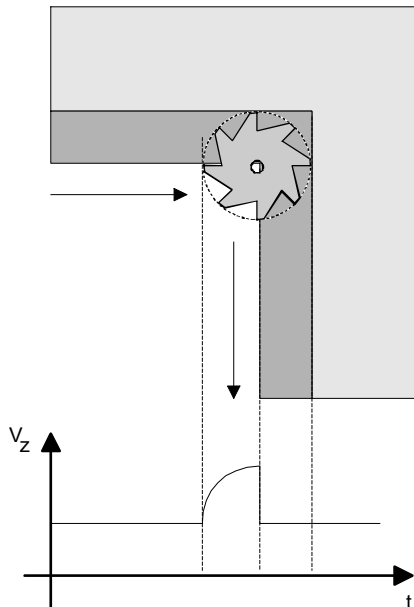
Fig. 46: Changing the removal volume Vz over time at an inside corner of 90° and at constant feedrate

4.12.1 Parameterising corner deceleration (#CORNER PARAM)

ℹ️ As of Build **V2.11.2010.02** the command **#CORNER PARAM [...]** replaces the command **#SET CORNER PARAM [...].** For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#CORNER PARAM [DIST=.. **UNIT=**<ident> **FEED**=.. **]**

DIST=..	Corner distance in [mm, inch]
UNIT=<ident>	Corner feed unit FEED=... Permitted identifiers:
	FWORD Unit according to current F word
	PERCENT Unit in [%]
FEED=..	Corner feed rate, [according to UNIT<ident>]

The point in space after which the path feedrate is to be reduced linearly along the path can be calculated by specifying a corner distance. The corner distance to be programmed refers here to the corrected path, not to the originally programmed target points.

ℹ️ Corner deceleration is only effective in combination with the linear slope P-CHAN-00071.
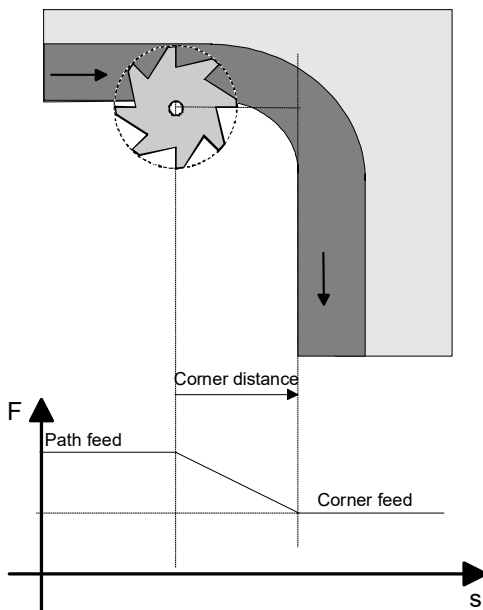
Fig. 47: Representation of feed at a circular inside contour

4.12.2 Selecting/deselecting corner deceleration (G12/G13)

Syntax:

G12	Deselecting corner deceleration	modal, initial state
G13	Selecting corner deceleration	modal

An error message is output if G13 is programmed without previously defining the corner deceleration parameter.

An error message is output if G12 and G13 are programmed at the same time in an NC block.

These G functions are modal. The initial state after system start-up is "Corner deceleration inactive" (G12).

No error message is output if corner deceleration (G13) and end of program (M30) are selected. The initial position G12 is selected before the program is started.

An error message is output at a corner distance of 0.

Selecting and deselecting corner deceleration (G12/G13)

```
:
N10 G01 G90 X10 Y10 F1000
N15 #CORNER PARAM[DIST=10 UNIT=PERCENT FEED=50] ;Corner distance 10mm
                                               ;Corner feed 50%
N20 G13      ;Activate
N30 X50      ;Corner deceleration active
N40 Y50      ;Corner deceleration active
N50 G12      ;Deselect
N60 X30 Y30
N70 M30
```

Please note that corner deceleration is not activated until block N030. The block transition N010/N030 is not affected by corner deceleration.

Similarly, corner deceleration is still active in block N040 because it was not deselected.

Deselecting corner deceleration must be programmed before or in the NC block in which corner deceleration should not longer be active.

4.13 Zero offsets (G53/G54/...G59)

In this section zero offset is abbreviated to 'NPV'.

Syntax:

G53	Deselect zero offset *	modal, initial state
G54	Activate 1st NPV	modal
G55	Activate 2nd NPV	modal
G56	Activate 3rd NPV	modal
G57	Activate 4th NPV	modal
G58	Activate 5th NPV	modal
G59	Activate 6th NPV	modal

The significance of the G53 data block can be controlled by P-ZERO-00001. Depending on parameterisation, G53 means either that NPV is deselected or the G53 data block is used as an addition NPV.

The default setting of the NPV which is automatically active in initial state is also parameterisable P-ZERO-00002.

G54... G59 provide the corresponding zero offsets from the zero offsets table. The zero offset is already valid in the block in which G53, G54, etc. is programmed. However, no path motion can take place without specifying any coordinates.

In initial state (G53) no zero offsets are active.

> **i** In addition, the following applies when selecting a zero offset in G91 mode:
>
> The programming of...
>
> N10 G54
>
> N20 G0 X0 G91
>
> ... does not call motion on the X axis since this is a motion relative to zero.
>
> As a result, a zero offset is only effective when the next motion datum is programmed in absolute coordinates (G90).

4.13.1   Enhanced zero offset variables

Variable	Meaning	Read	Write
V.G.NP[j].ALL	Address all axes of a zero offset	yes	yes
V.G.NP_AKT.V[i]	Current (currently active) zero offset of an axis	yes	yes
V.G.NP_AKT.ALL	Address currently active zero offsets of all axes	yes	yes
V.G.NP_DEFAULT	Index of effective zero offset in initial state	yes	yes

Write access via the V.G.NP[j] and V.G.NP_DEFAULT variables remains effective until the next list interpretation (explicit request or start-up). By contrast, write accesses via V.G.NP_AKT are only effective on zero offsets currently stored in the decoder. When the NPV previously written via V.G.NP_AKT is re-selected, the original values from the list again become effective. The current data record must be saved to the list by assignment via V.G.NP[j].ALL to retain write access to V.G.NP_AKT after deselection.

Enhanced zero offset variables

Saving the currently active zero offset values:

```
V.G.NP[12].ALL=V.G.NP_AKT.ALL
```

4.13.2 Adding and subtracting offsets

The V.G.NP[j].ALL variable re-assigns complete NPV data records by the additive assignment of existing NPVs. The offsets of all axes are included in the various calculations. The notations ` += `, ` -= `, and `=` are permitted in NPV assignment.

G54 (NPV1) is assigned the combination of G55 (NPV2) and G57 (NPV4):

```
N10 V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL
```

The same operation but with the inclusion of G54:

```
N10 V.G.NP[1].ALL = V.G.NP[1].ALL + V.G.NP[2].ALL + V.G.NP[4].ALL
```

or

```
N10 V.G.NP[1].ALL += V.G.NP[2].ALL + V.G.NP[4].ALL
```

NOTICE
It is not possible to interconnect V.G.NP[j].ALL variables with axis-specific V.G.NP[j].V[i] variables or constants within an assignment.

G54 (NPV1) is defined by the combination of G55 (NPV2), the X offset of G57 (NPV4) and a compensation value:

WRONG:

```
N10 V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].V.X + 100
```

RIGHT:

A assignment must be made in two steps:

```
N10 V.G.NP[1].ALL = V.G.NP[2].ALL
N20 V.G.NP[1].V.X = V.G.NP[4].V.X + 100
```

4.13.3   Access to the current zero offset

The currently active NPV in the decoder is accessed via the V.G.NP_AKT.V[i] variable. The operator need not know which NPV (i.e. which index) is currently selected.

The current NPV of the X axis should be 200.

```
N10 V.G.NP_AKT.V.X = 200
```

The current NPV in all axes should be expanded by the offset values from G55 (NPV2).

```
N10 V.G.NP_AKT.ALL = V.G.NP_AKT.ALL + V.G.NP[2].ALL
```

or

```
N10 V.G.NP_AKT.ALL += V.G.NP[2].ALL
```

Changing the current NPV is *not* effective across programs. However, by adding an assignment, the operator can can also use the current NPV in other programs.

The current NPV should be saved under G54 (NPV1) for future utilisation in other programs.

```
N10 V.G.NP[1].ALL = V.G.NP_AKT.ALL
```

4.13.4   Default zero offset

The active NPV after program start is parameterisable (P-ZERO-00002).

In the NC program, the operator can use the V.G.NP_DEFAULT variable to access the default index. The change is valid across all programs.

Starting with this program line, G55 (NPV2) will be the new default NPV. This means that G55 is automatically active at next program start.

```
N100 V.G.NP_DEFAULT = 2
```

4.13.5   Creating zero offset groups

A zero offset group (NPVG) defines the rules to create an NPV, i.e. it describes the components contained in a specific NPV data block. An NPVG is defined in the NC program or by pre-assignment in the channel parameter list [1] [▸ 786]-1 in conjunction with a symbolic string. The use of symbolic strings is described in the section Macros [▸ 672]. Three steps are basically required to activate an NPVG:

Creating zero offset groups

1st step:

Define an NPVG in the NC program, e.g. under the macro name VERSCH_1:

```
N10 " VERSCH_1 " = " V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL "
```

or in the channel parameter list [1] [▸ 786]-1:

```
makro_def[0].symbol VERSCH_1
makro_def[0].nc_code V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL
```

2nd step:

Creating the NPV (new assignment of the NPV1 data block):

```
N20 " VERSCH_1 "
```

3rd step:

Select the NPV (in this example: NPV1, i.e. G54)

```
N30 G54
```

4.13.6 Extended zero offset (G159)

Syntax:

G159=..   Extended zero offset with specification of list index of the zero   modal
data block

Access to additional NPV data blocks [3] [▶ 786]. It is also possible to access the G53…G58 data blocks.
The maximum number of extended NPVs is parameterisable [6] [▶ 786]-6.12.

Extended zero offset (G159)

```
G159 = 0     ;corresponds to G53
G159 = 1     ;corresponds to G54
G159 = 2     ;corresponds to G55
G159 = 3     ;corresponds to G56
G159 = 4     ;corresponds to G57
G159 = 5     ;corresponds to G58
G159 = 6     ;corresponds to G59
G159 = 7     ;Data block with list index 7
:
G159 = 10    ;Data block with list index 10
:
```

4.13.7   Enable/disable zero offsets axis-specific (G160)

Syntax:

G160= .. *<axis_name>..* { *<axis_name>..* }   modal

G160=..   Axis-specific zero offset with specification of list index *<i>* of the zero data block

<axis_name>..   Axis with validity flag of its zero offset

                         0: Zero offset of the axis is included in the calculation.

                         1: Zero offset of the axis is not included in the calculation.

You can define *<i>* in each zero offset data block *in advance* by setting the parameters with P-ZERO-00004 for the axes *<axis_name>* for which the offset is or is not to be included in the calculation. This means, individual axis offsets can be disabled or enabled.

This axis-specific validity of a zero offset can be changed in the NC program using G160.

Before selecting G55 (zero offset data record with index 2) the offsets of the X and Z axes are disabled and the offsets of the Y axis are enabled.

```
:
N10 G160 = 2 X1 Y0 Z1
N20 G55
:
```

As a result, only the axis offsets of G55 which are not disabled (Y axis) are included in the motion in the next motion block.

4.14 Specifying centre point for circle definition (G161/ G162)

`Syntax:`

G161 Circle centre point absolute modal

G162 Circle centre point relative modal, initial state

When G162 is active (initial state), the centre point is defined by I, J and K relative to the circle starting point.

When G161 is active, I, J and K are specified absolute in the programmer's coordinate system.

4.15   Radius programming (R/G163)

Syntax:

R..   Specify circle radius value in [mm, inch]   modal

Radius programming is expected if no circle centre point (with I, J and/or K) is specified. The value following the character R is interpreted as the radius in [mm, inch]. It is automatically used in lines with circular programming. As a result, the radius statement is modal and, in case of multiple circular path motions with the same radius, it is unnecessary to repeat the radius specification.

> **i** The maximum permissible circle radius is 10^9 mm. However, the target point of the arc may not exceed the maximum permissible motion path of the axes of +- $2,14*10^5$ mm.

> **i** If the radius value is possible, the shortest possible arc is defined; if the radius value is negative, the largest possible circle is defined (see figure below).
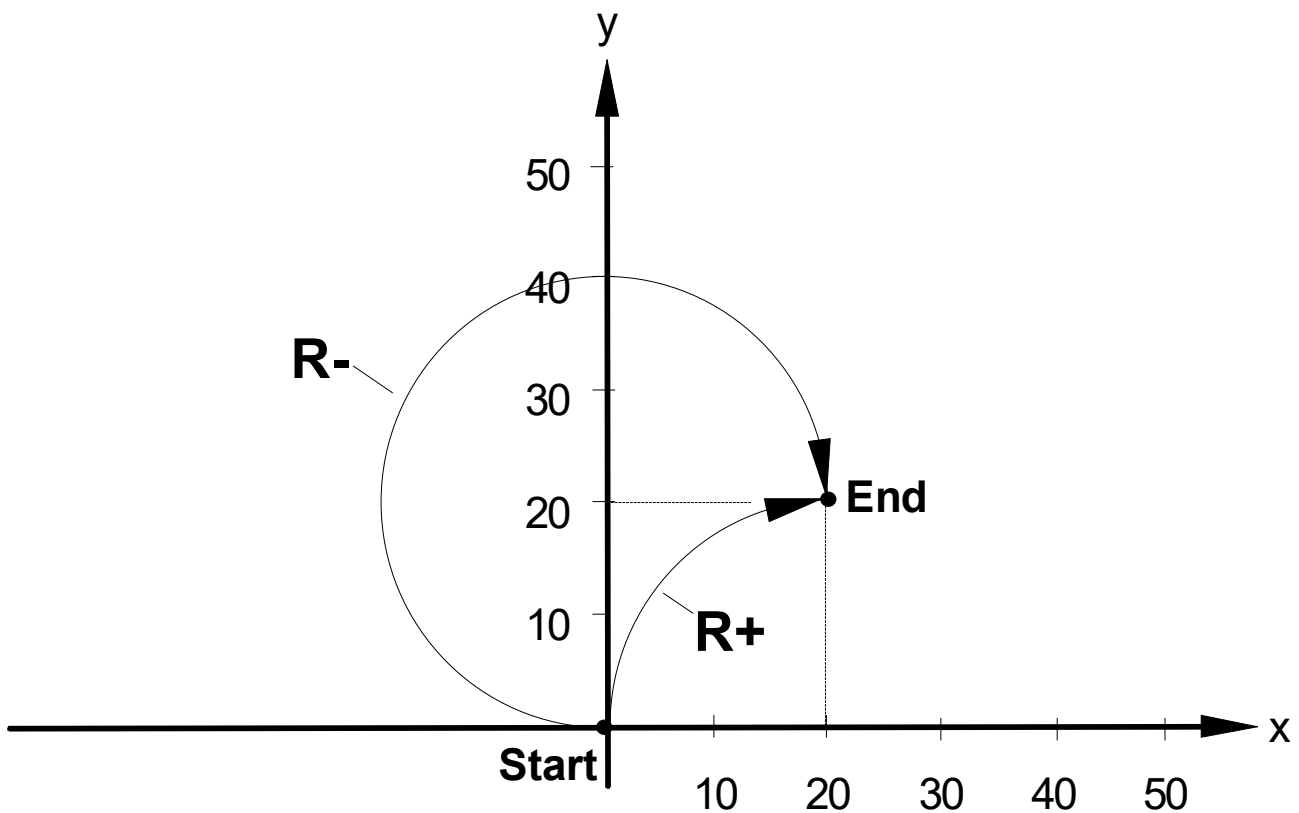
With indexed radius programming, the radius can be specified as "R1". In this case, the index may only have the value 1. If "R1" is used on the right-hand assignment side, the index value 1 may not be programmed as a mathematical expression.

Radius programming (R-, G163)

```
1.)
N10 P2 = 1
N20 RP2 = 5     ;permitted

2.)
N10 R1 = 5
N20 P2 = R1     ;permitted

3.)
N10 R1 = 5 P2 = 1
N20 P3 = RP2    ;not permitted
```

These examples produce semicircles of radius 50.

```
1.)
N10  G90 G01 X0 Y0 F500
N20 G02 X100 R50 ;clockwise semicircle
N30  G03 X200  R50 ;counter-clockwise semicircle


2.)
N10  G90 G01 X0 Y0 F500
N20  G02 R=50 ;no motion as yet here
N30  X100 ;clockwise semicircle


3.)
N10  G90 G01 X0 Y0 F500
N20  R1=50
N30  G02 X100 ;clockwise semicircle
N40  G03 X200 ;counter-clockwise semicircle


4.)
N10  G90 G01 X0 Y0 F500
N20  G02  X100  R1=50 ;clockwise semicircle
N30  G03 X200 ;counter-clockwise semicircle
```

The programming below results in an error message since R1 is interpreted as a radius of value 1.
```
N10 G90 G01 X0 Y0 F500
N20 R1=50
N30 G02 X100 R1
```

As an alternative to circle definition with R or R1, the circle radius can be specified using the following G code:

```
Syntax:
```

| **G163=**.. | Specify circle radius value in [mm, inch] | modal |

The circle definition using radius is valid when circular interpolation is selected until it is redefined or until it is deselected by specifying an I and/or J and/or K.

Circle radius programming with G163

```
(N10: Motion to origin)
(N20: Clockwise semicircle with target value X100 and under preset)
(of the circle radius by G163 (radius specified is modal))
(N30: Counter-clockwise semicircle with target value X200 and radius,)
(which was defined in N20 and is modal)

%Radiusprogramming_G163
N10 G90 G01 X0 Y0 F1000
N20 G02 G163=50 X100 ;clockwise semicircle
N30 G03 X200 ;counter-clockwise semicircle
N40 M30
```
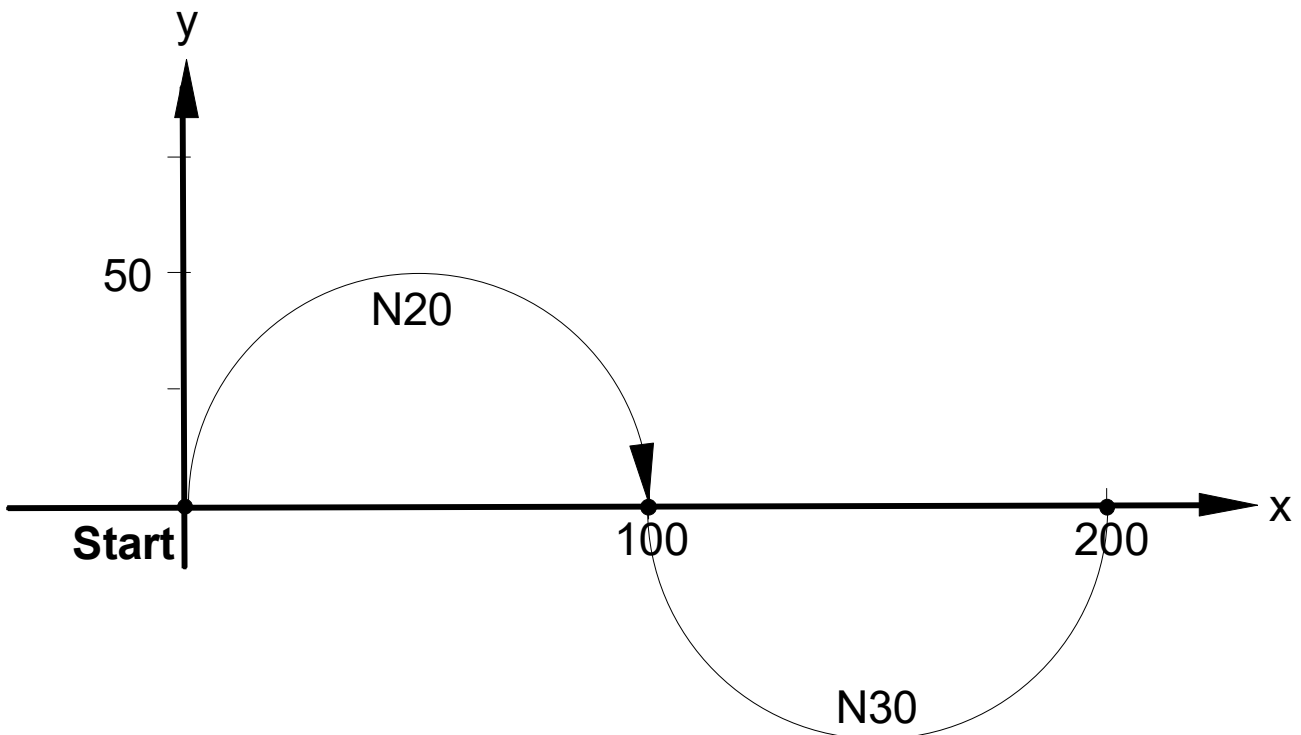
NOTICE

If the starting and end points of the circle programmed with "R", "R1" or "G163" are identical, an error message is output. If a full circle is travelled, it must be programmed with I/J/K.

4.16 Controlling centre point offset in circle (G164/G165)

`Syntax:`

G164	Centre point compensation OFF	modal
G165	Centre point compensation ON	modal, initial state

When G165 is active, a circle programmed by an I, J and K statement is compensated in such a way that an arc can be interpolated if the circular direction (G02/G03), start position (end point of the previous block) and end point (coordinates in the circular block). The centre point programmed with I, J and K may then be offset. The more precise the centre point is specified, the less the centre point offset will be.

> ℹ️ If the circle is programmed by specifying a radius R, no circle centre point compensation is effective since the centre point is then always calculated exactly here.

For the deviation from programmed to compensated centre point, two limit values P-CHAN-00059 and P-CHAN-00060 are monitored. If they are exceeded, an error message is output:

mittelpkt_diff: Permissible deviation in 10^{-4} mm

mittelpkt_faktor: Percentage deviation in 0.1%

A check is made whether the centre point offset Δm is greater than the absolute value "mittelpkt_diff"

$$\Delta m > \text{mittelpkt\_diff ?}$$

and whether the centre point offset Δm is greater than the product of "mittelpkt_faktor/1000" and the corrected radius "radius".

$$\Delta m > \text{mittelpkt\_faktor/1000 * radius ?}$$

Therefore, the upper limit for Δm is linearly dependent on the calculated radius. This results in the relationship contained in between centre point offset Δm and the calculated radius "radius".

Example:

"mittelpkt_faktor" = 5 signifies that the distance between the programmed centre point coordinates and the compensated centre point coordinates may be maximum 0.5% of the compensated radius of the circle.
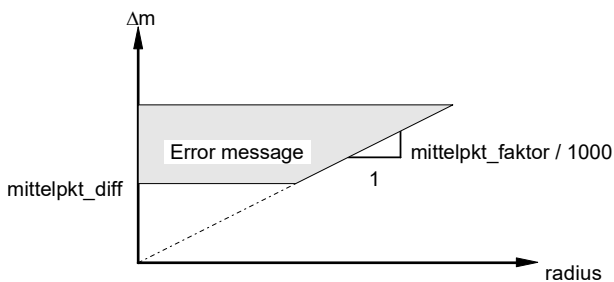
Fig. 48: Relationship between centre point offset Δm and the calculated radius "radius"

The programmed centre point coordinates must then lie in a circumcircle about the compensated centre point of the circle. The radius of this circumcircle corresponds to the permissible centre point offset Δm which can be set using the two parameters 'mittelpkt_diff' and 'mittelpkt_faktor':
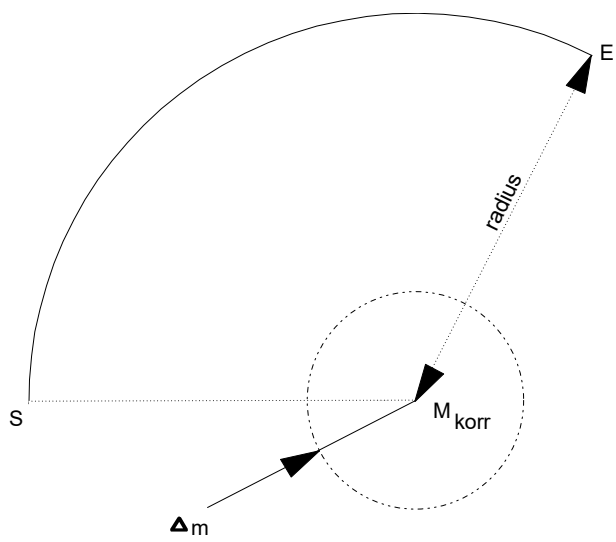
Fig. 49: Area of permissible programmed centre points

4.16.1 Special function: circle radius compensation in combination with G164

In certain situations, circle centre point compensation (G165) can lead to an unfavourable offset in the programmed circle centre point and therefore the circle's position. Such unfavourable situations may occur when the circle starting and target points are close together and the circle is almost equivalent to a programmed full circle.

This type of circle with specified target point with and without circle centre point compensation (G165/G164) is programmed in the example below. To simulate the resolution error in the post-processor and when circle centre point compensation is active, the circle's target point is shifted by 0.1µm in each case in the x and y directions relative to the starting point. The circle with G165 is rotated about the starting point and the position of the compensated circle centre point M_K shifts very considerably relative to the programmed centre point M.

Special function: circle radius compensation in combination with G164

```
N10 G00 G90 X0 Y0 Z0
N20 G01 X-50 Y0 F20000
N30 G01 X0
N40 G165 G02 X0.0001 Y0.0001 J200
N50 G01 Y50
M30

N10 G00 X0 Y0 Z0
N20 G01 X-50 Y0 F20000
N30 G01 X0
N40 G164 G02 X0.0 Y0.0 J200
N50 G01 Y50
N60 M30
```

BECKHOFF
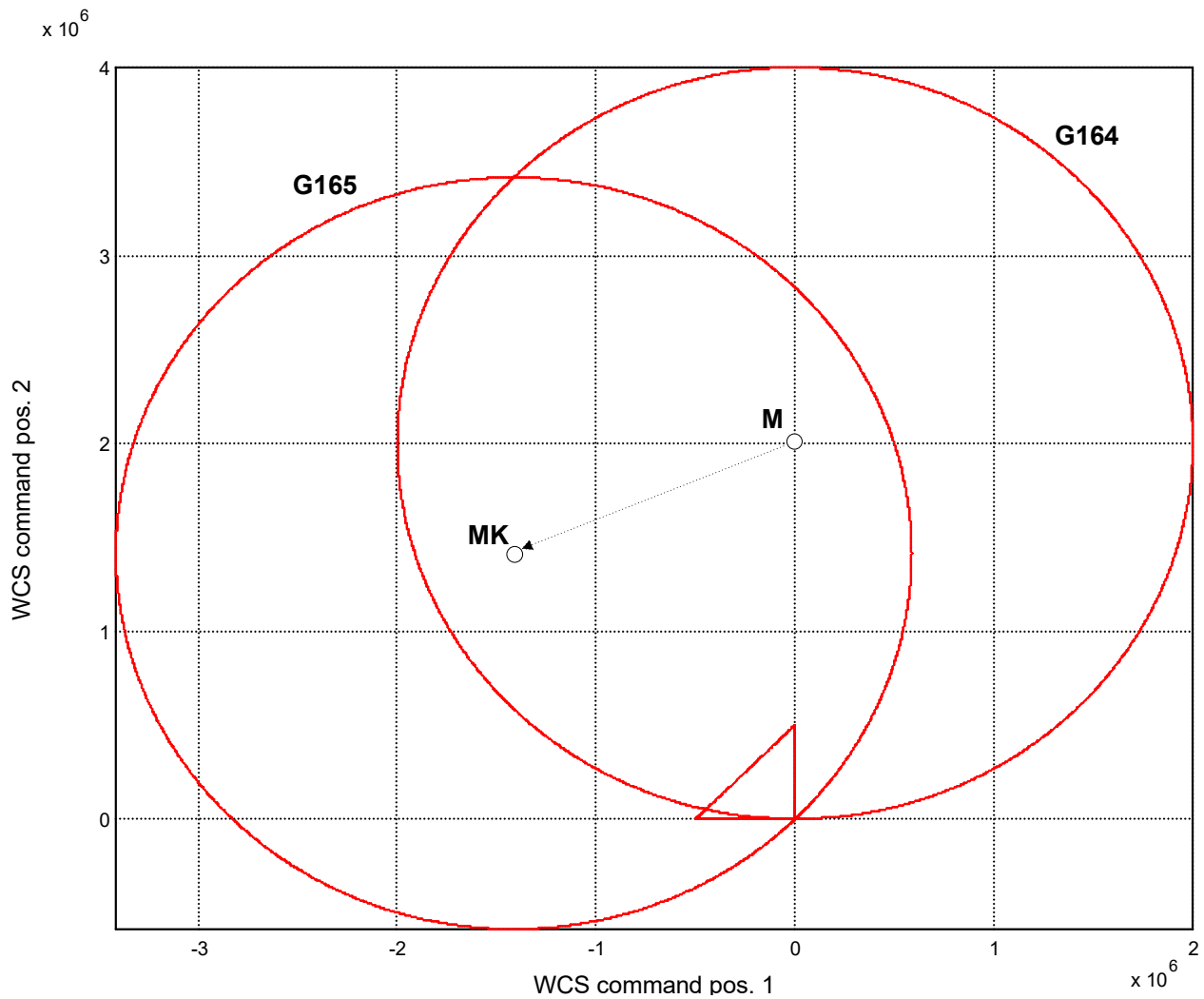
Fig. 50: Circle centre point shift in the case of G165

In such cases a better result can generally be obtained when circle centre point compensation (G164) is disabled and when max_radius_diff_circle > 0 (P-CHAN-00171) and max_proz_radius_diff_circle (P-CHAN-00172) are set. The programmed circle centre point is not changed by the function and the circle radius difference is transferred linearly by the circle's angle from the starting radius to the target radius.

If the circle radius deviations lie in the order of magnitude of the resolution accuracy, the circle distortions and dynamic effects are generally negligible.

ℹ️ For a full circle, the circle starting and target points must be identical in this case.

Version: 1.238

4.17 Feedforward control (G135/G136/G137)

`Syntax:`

G135	Selecting feedforward control	modal
G136 *<axis_name>.. { <axis_name>.. }*	Specify weighting	modal
G137	Deselecting feedforward control	modal

Path distortions can be reduced by using velocity and acceleration feedforward control.

Axis group.specific activation is programmed with G135. An axis-specific percentage weighting of the calculated feedforward control variables in [%] takes place with G136. It is limited to 100% for all axes.

G137 deactivates axis group-specific feedforward control. In case of axes for which no feedforward control is to be implemented after global selection with G135, a percentage weighting of 0% must be specified with G136. It is also possible to enter the selection and weighting of feedforward control in a single block.

At every program start, feedforward control is explicitly disabled in the interpolator and the weighting factors are set to 100%.

If feedforward control is disabled or enabled during the NC program, the weighting factors remain at the values set by G136 or, if no G136 is programmed, to 100%.

NOTICE
After an axis exchange, the G136 weighting factors are reset to 100% for all axes involved.

Feedforward control (G135/G136/G137)

```
G135    (Select feedforward control: weighting for all axes 100%)
G136 X80 Y95 Z0(Weighting;  Z axis has no feedforward control here)
G137    (Deselect feedforward control: weighting for all axes 100%)
```

4.18 Weighting of maximum velocity (G127/ G128)

```
Syntax:
```

G127 *<axis_name>..* { *<axis_name>.. }*	Axis-specific weighting of maximum velocity: weighting for specific axes in [%] **[as of V2.11.2808.14]**	modal
G128=..	Axis group-specific weighting of maximum velocity. Weighting for all axes in [%].	modal

Using the G127/G128 function it is possible to change the maximum velocity.

The maximum velocity can be influenced by a percentage change in the associated velocity characteristic value.

If programming takes place with G127/G128, all axes which are not programmed or not yet programmed are set to 100%. Each further selection of this function, irrespective of the previous programs, signifies 100%. This means, the geometrical data processing always weights the default value P-AXIS-00212 with the percentage value.

Therefore, 50% programmed twice in succession means the setting is made to 50% and not to 25%.

> **i** Feed weighting at maximum velocity does not act on single axis motions e.g. homing, manual mode or independent axes.

NOTICE
If G127/G128 > 100% the maximum velocity is limited to the MAX value P-AXIS-00212. If G127/ G128 = 0, the maximum velocity is limited to the minimum value 1µm/s.

Weighting of maximum velocity (G127/ G128)

```
N10 G127 X70 Y60          ;Axis-specific weighting of vb_max
                          ;vb_max of X axis is restricted to 70%
                          ;vb_max of Y axis is restricted to 60%
N20 G128 = 100            ;Axis group-specific weighting of vb_max
                          ;vb_max of all axes to 100%

Special feature:
N20 G128 = 100 X10 Y20    ;Using G128, axis positions
                          ; can also be programmed within the same block!
```

4.19   Weighting of rapid traverse velocity (G129)

Syntax:

| **G129**=.. | Axis group-specific weighting of rapid traverse weighting for **all** axes in [%] | modal |

The G129 functions can be used to change the rapid traverse velocity G00.

The feedrate can be influenced by a percentage change in the associated velocity characteristic value.

If programming takes place with G129, all axes which are not programmed or not yet programmed are set to 100%. Each further selection of these functions, irrespective of the previous programs, signifies 100%. This means, the geometrical data processing always weights the default value P-AXIS-00209 with the percentage value.

Therefore, 50% programmed twice in succession means the setting is made to 50% and not to 25%.

> **i** Feed weighting only acts on rapid traverse NC blocks (G00). It has no effect on single axis motions e.g. homing, manual mode or independent axes.

NOTICE

If G129 is >100%, rapid traverse velocity is limited to the MAX value P-AXIS-00212.

If G129 = 0, rapid traverse velocity is limited to the minimum value 1μm/s.

Weighting of rapid traverse velocity (G129)

N10 G129 = 70	(Axis group-specific rapid traverse weighting) (Rapid traverse velocity of <u>all</u> axes to 70%)
N20 G00 X100 Y150	(Linear interpolation, rapid traverse motion with 70%)
Special feature:	
N50 G129 = 70 X10 Y20	(With G129 in the same block, axis positions) (can also be programmed!

4.20 Parameterising the acceleration profile

4.20.1 Acceleration weighting (G130/G131/G231/G333/G334)

```
Syntax of weightings for accelerations:
```

G130 <axis_name>. { <axis_name>..}	Axis-specific acceleration weighting, weighting for specific axes in [%], only acts on feed blocks G1/G2/G3.	modal
G131=..	Axis group-specific acceleration weighting with G01, G02, G03, weighting for all axes in [%]	modal
G231=..	Axis group-specific acceleration weighting with G00, weighting for all axes in [%]	modal

Weightings for accelerations in feedhold:

G333 <axis_name>. { <axis_name>..}	Axis-specific acceleration weighting with feedhold, weighting for specific axes in [%] **[as of V3.1.3079.16]**	modal
G334=..	Axis group-specific acceleration weighting with feedhold, weighting for all axes in [%] **[as of V3.1.3079.16]**	modal

The G130/G131/G231 functions can change acceleration ramps.

This acceleration can be influenced by a percentage change in the corresponding default assignment of acceleration characteristic values. With a jerk-limited profile, these values are the axis parameters P-AXIS-00001 and P-AXIS-00002.

When P-CHAN-00097 is set, the feedhold parameter list is used for the deceleration process. In this case, the deceleration of the feedhold ramp P-AXIS-00024 by a percentage change using G333/G334. Feedhold weighting is only effective if the resulting deceleration is equal to or greater than the active values of the weighted G01/G00 accelerations.

When the function is programmed with G130/G131/G231/G333/G334, all axes which are not programmed or not yet programmed are set to 100%. Every additional selection of these functions, irrespective of previous programming, refers to 100%. This means that geometrical data processing always weights the default values [2] [▶ 786]-1 and/or [2] [▶ 786]-2 with the percentage value.

Therefore, 50% programmed twice in succession means the setting is made to 50% and not to 25%. A weighting of over 100% is possible up to maximum axis acceleration P-AXIS-00008.

Alternatively, a reduction in acceleration can take place using the "Reducing path acceleration" control unit.

NOTICE
After an axis exchange, the G130/G333 weighting factors are reset for all axes involved to 100%.

i Acceleration weighting does **not** act on single axis motions, e.g. homing, manual mode or independent axes.

Acceleration weighting (G130/G131/G231/G333/G334)

N10 G130 X70	;Axis-specific weighting of acceleration ;Acceleration of X axis is restricted to 70%
N20 G01 F1000 X100	;Linear interpolation
N30 G130 Y60	;Acceleration of Y axis is restricted to 60% ;Acceleration of X axis remains at 70%
N40 Y100	;Linear interpolation
N50 G131 = 100	;Axis group-specific acceleration weighting ;G01,G02,G03 acceleration of <u>all</u> axes to 100%
N60 G231 = 80	;Axis group-specific acceleration weighting ;G0 acceleration of <u>all</u> axes to 80%
N70 G00 X200	;Rapid traverse
N80 G333 X150	;Axis-specific weighting of acceleration ;with feedhold. Deceleration of X axis is increased to ;150%.
N90 G334 = 200	;Axis group-specific acceleration weighting ;with feedhold. Deceleration of <u>all</u> axes is increased to ;200%.
Special feature:	
N50 G131 = 100 X10 Y20	;With G131/G231/G334 axis positions can ;also be programmed in the same block!

4.20.2   Ramp time weighting (G132/G133/G134/G233/G338/G339)

```
Syntax of weightings for ramp time:
```

G132 *<axis_name>.* { *<axis_name>*..}	Axis-specific ramp time weighting, weighting for specific axes in [%], only acts on feed blocks G1/G2/G3.	modal
G133=..	Axis group-specific ramp time weighting with G01, G02, G03, weighting for all axes in [%]	modal
G134=..	Axis group-specific weighting of geometrical ramp time, weighting for all axes in [%]	modal
G233=..	Axis group-specific ramp time weighting with G00, weighting for all axes in [%]	modal

Weightings for ramp times in feedhold:

G338 *<axis_name>.* { *<axis_name>*..}	Axis-specific ramp time weighting with feedhold, weighting for specific axes in [%] **[as of V3.1.3079.16]**	modal
G339=..	Axis group-specific ramp-time weighting with feedhold, weighting for all axes in [%] **[as of V3.1.3079.16]**	modal

These ramp times can be influenced by a percentage change in the corresponding default ramp times. The G132/G133/G233 functions can change the ramp time of axis acceleration with a non-linear slope [2] [▶ 786]-1. (If the slope is linear, the acceleration curve is step-shaped (see figure in Section Jerk-limiting slope [▶ 356])

The G134 function can change the geometrical ramp time of a non-linear slope (P-AXIS-00199).

The parameters for acceleration ramp times for the up-gradation and down-gradation of acceleration are P-AXIS-00196 and P-AXIS-00195 respectively, and for the up-gradation and down-gradation of deceleration, the parameters are P-AXIS-00198 and P-AXIS-00197 respectively.

When P-CHAN-00097 is set, the feedhold parameter list is used for the deceleration process. In this case, the deceleration of the feedhold ramp P-AXIS-00081 by a percentage change using G338/G339.

When the function is programmed with G132/G133/G134/G233/G338/G339, all axes which are not programmed or not yet programmed are set to 100%. Every additional selection of these functions, irrespective of previous programming, refers to 100%. This means that geometrical data processing always weights default values with the percentage value. Therefore, 50% programmed twice in succession means the setting is made to 50% and not to 25%.

NOTICE
After an axis exchange, the G132/G333 weighting factors are reset for all axes involved to 100%.

i   Ramp time weighting does **not** act on single axis motions e.g. homing, manual more an independent axes.

Ramp time weighting (G132/G133/G134/G233/G338/G339)

`N10 G132 X200`	`;Axis-specific weighting of ramp time` `;Ramp time of X axis is increased by 200%`
`N20 G01 F1000 X100`	`;Linear interpolation`
`N30 G132 Y50`	`;Ramp time of Y axis is decreased by 50%` `;Ramp time of X axis remains at 200%`
`N40 Y100`	`;Linear interpolation`
`N50 G133 = 100`	`;Axis group-specific weighting of ramp time` `;G01,G02,G03 ramp times of all axes to 100%`
`N60 G134 = 50`	`;Axis group-specific weighting of ramp time` `;Geometrical ramp time of all axes to 50%`
`N70 G233 = 80`	`;Axis group-specific weighting of ramp time` `;G00 ramp times of all all axes to 80%`
`N80 G00 X200`	`;Rapid traverse`

`N90 G338 X150`	`;Axis-specific weighting of ramp time`
	`;with feedhold. Ramp time of X axis is increased to`
	`;150%.`

`N100 G339 = 200`	`;Axis group-specific weighting of ramp time`
	`;with feedhold. Ramp time of all axes is increased to`
	`;200%.`

`Special feature:`	
`N50 G133 = 100 X10 Y20`	`;With G133/ G134/G233/G339 axis positions can` `;also be programmed in the same block!`
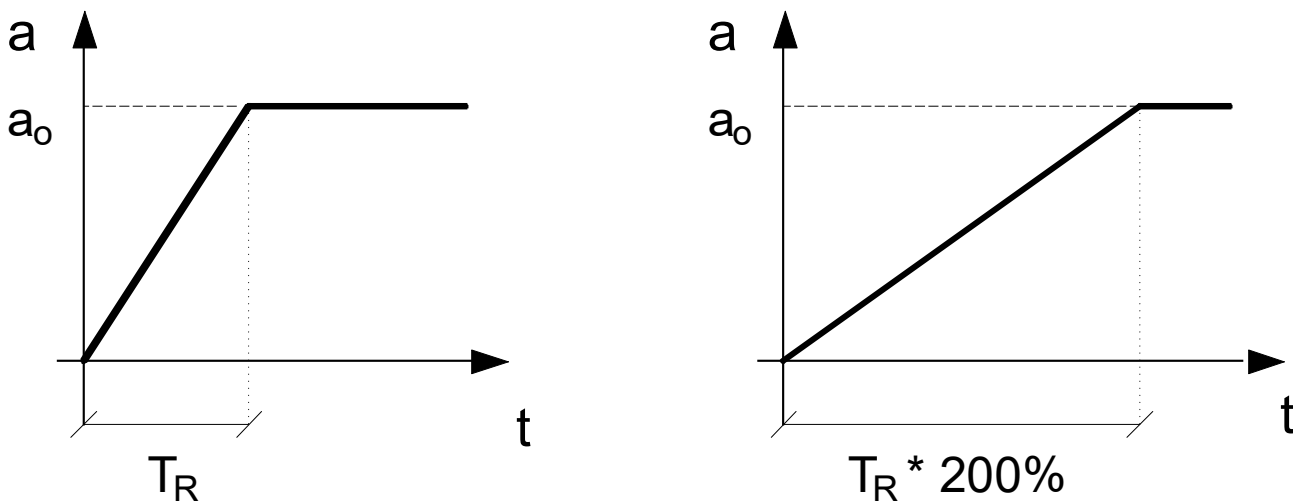
Fig. 51: Example of ramp time weighting with G132/G133/G233

The figure below shows the influence of G134 on acceleration perpendicular to the path (centrifugal acceleration a1--> a2) with increasing feed (v1 --> v2) during circular motion (P1 --> P2).

If the change in centrifugal acceleration is reduced by increasing the G134 ramp time, acceleration is smoother and target acceleration a2 is not reached until point P3.
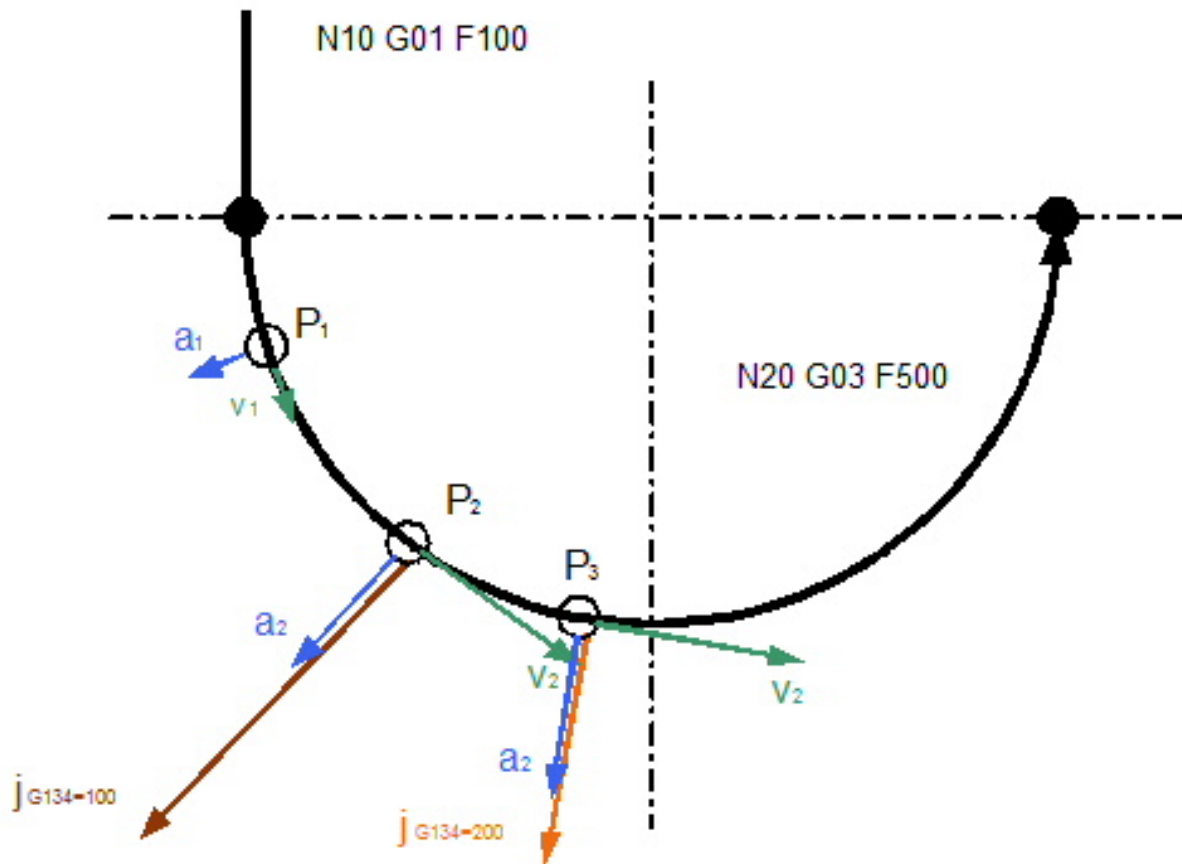
Fig. 52: Ramp time weighting with G134 and with circular interpolation

4.21   Machining time or feedrate (G93/G94/G95/G194)

Syntax:

G93	Specify machining time in seconds	modal
G94	Feed per minute	modal, initial state
G95	Feedrate per revolution	modal
G194	Feedrate calculation based on maximum weighted axis feeds	modal

The action of the F word can be optionally switched over by using the G functions G93, G94, G95 and G194.

G93 combined with the F word defines a machining time in [s].

G94 combined with the F word:

- feed in [mm/min, m/min, inch/min] for linear axes
- feed in [°/min] for rotary axes

G95 combined with the F word defines a feedrate per revolution in [mm/U, inch/U]. The function is described in greater detail in Section Feedrate per revolution (G95) [▸ 602].

G194 combined with the F word defines a weighting factor in [%] for maximum feedrates. The maximum permissible feedrate on the path then results from the axis-specific values P-AXIS-00212. At least one axis then moves at its maximum weighted velocity. Only weighting factors less than 100% are permitted.

Machining time or feedrate (G93/G94/G95/G194)

```
N10 G90 F1000 X100     (Feedrate 1000 mm/min (G94 default))
N20 G194 F90           (Weighting 90% to max. axis feedrate)
Nxx X200               (Feedrate e.g. 9000 mm/min at vb_max=10000 mm/min)
N80 G94 X50            (Feedrate 1000 mm/min valid from N10)
Nxx X.. Y.. Z..        (Interpolation)
N120 G93 F20           (Machining time 20 s)
Nxx X.. Y.. Z..        (Interpolation)
N160 G94 F1500 X150    (Feedrate 1500 mm/min)
Nxx X.. Y.. Z..        (Interpolation)
N200 M30
```

4.22 Inserting chamfers and roundings (G301/G302) (#FRC/ #CHR/#CHF/#RND)

```
Syntax:
```

| **G301** | Inserting chamfers | Both functions are effective between |
| **G302** | Inserting roundings | two motion blocks |

G301 inserts a straight line at the identical angle of inclination to the adjacent contour elements (chamfers).

G302 inserts an arc with a tangential transition to the two adjacent contour elements (roundings).

These functions are non-modal and generate precisely one insertion segment (arc or straight line). G301/ G302 blocks may be only written between blocks with active G functions of the group "G00, G01, G02/G03 except for G05".

The I word programmed in the same NC block defines the rounding size or the chamfer width of the insertion segments in [mm, inch]. The I word remains stored and active. This means that I word need no longer be programmed in the following G301/G302 at the same radius or same chamfer width.

In the initial programming of G301/G302, an I word unequal to zero must be programmed in the NC block, otherwise an error message is output (error which leads to abortion of decoding).

Effectiveness of path feedrate in an inserted chamfer or rounding segment:

- If the previous feedrate is G00 (rapid traverse), the segment is also travelled at maximum possible velocity.
- If the previous feedrate is G01/G02/G03, the programmed feedrate is also valid in the segment.
- A feedrate may also be specified in a block containing G301/G302. This feedrate is also valid in all following G01/G02/G03 blocks.
- The feedrate is also adapted when G11 and G41/G42 are active.

Specific chamfer or rounding feedrate:

> **As of Build V3.1.3057.04, an active feedrate can only be programmed in the inserted chamfer or rounding segment together with a specified chamfer or rounding.**

```
Syntax:
```

| **#FRC**=.. | Feedrate in inserted chamfer or rounding segment using the F word unit (e.g. mm/min) | non-modal |

Inserting chamfers and roundings (G301/G302)

Chamfer: 90° corner with 2 straight lines (I=20 is specified for a chamfer 20x45°)

```
N100  G00  G91  X100  Y0
N110  G01            Y100 F200
N120  G301 I20
N130            X-60
N140  G00  G90  X0   Y0
```
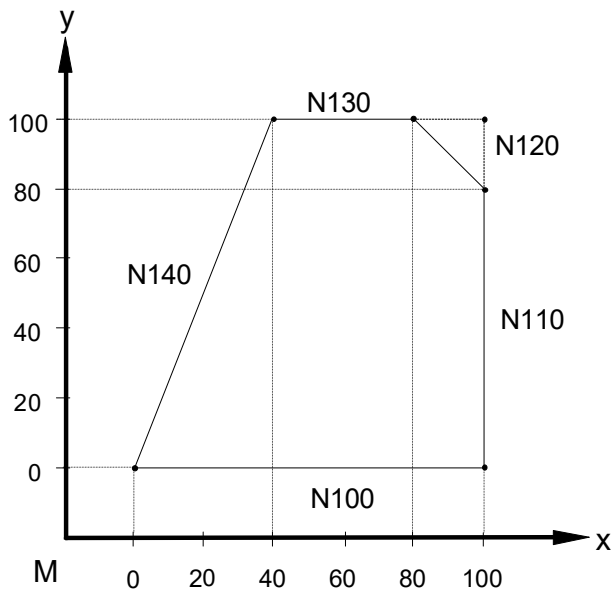

Fig. 53:

Rounding: rectangular pocket with corner radius 20 mm, length 200 mm, width 100 mm, segment-specific feedrates

Default:					Alternatively:			
N100	G00	X0	Y0		N100	G00	X0	Y0
N110	G01	X200	F200		N110	G01	X200	F200
N120	G302	I20	F150		N120	G302	I20	#FRC=150
N130		Y100	F200		N130		Y100	
N140	G302		F150		N140	G302		#FRC=150
N150		X0	F200		N150		X0	
N160	G302		F150		N160	G302		#FRC=150
N170		Y0	F200		N170		Y0	
N180	G302		F150		N180	G302		#FRC=150
N190		X40	F200		N190		X40	
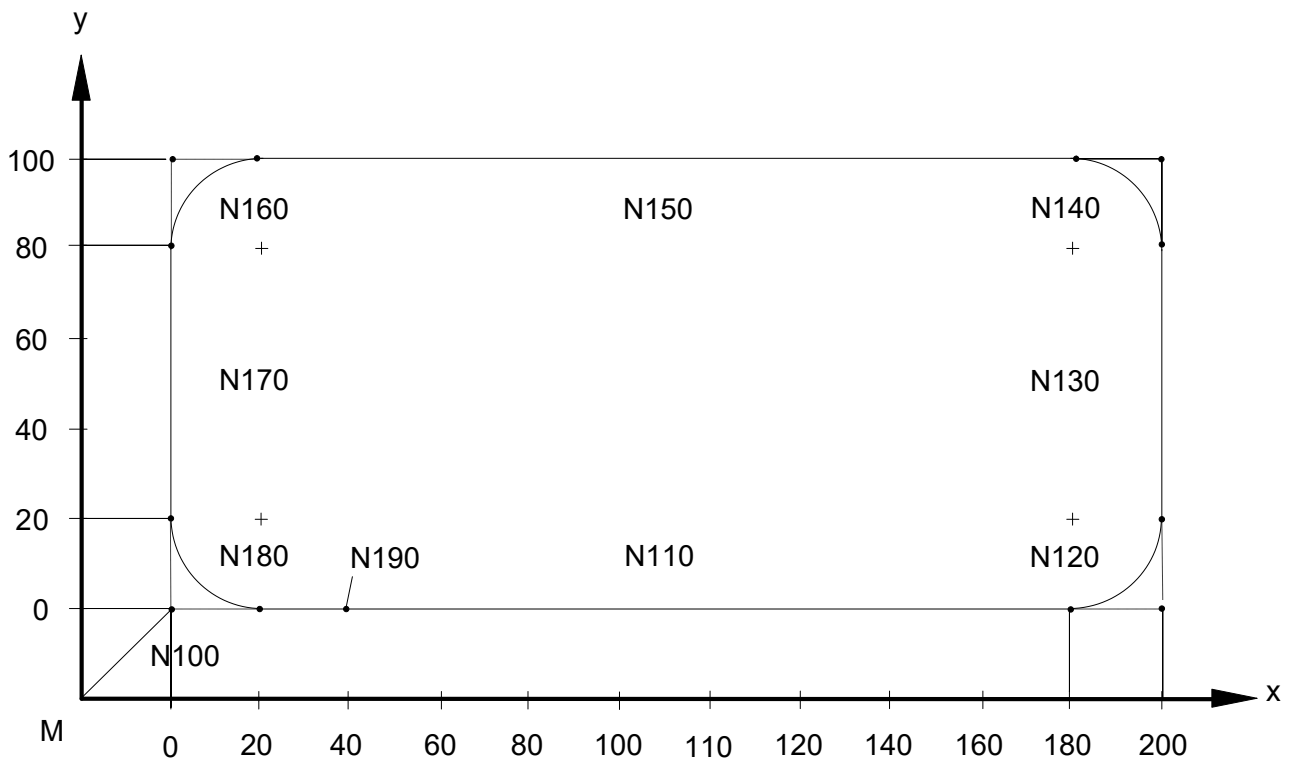
The chamfer or rounding is always added in the plane in which the second motion block is programmed.

Example A:

```
N100  G18  X20
N110  G19
N120  G301 I5
N130  Y20 Z20
```

Example B:

```
N100  G18  X20
N110  G301 I5
N120  G19
N130  Y20 Z20
```

Example A and B both yield the same result, contouring in the Y-Z plane.

> **i** As of Build V3.1.3057.04 there are additional options to program chamfers and roundings:

Extended G functions G301 and G302:

Chamfers and roundings are programmed as additional values with G301 and G302. Chamfers and rounding values must always be specified in [mm, inch]; they are non-modal.

G301 and G302 can be programmed directly in the first motion block. A separate NC block is not required.

```
Syntax:
```

| **G301=..** | Insert chamfers specifying the chamfer in [mm, inch] | non-modal |
| **G302=**.. | Insert radii specifying the radius in [mm, inch] | non-modal |

Inserting chamfers and roundings (G301/G302)

Rectangular pocket with 2 chamfers (G201) and 2 roundings (G302) with specific feedrates

```
N05  G17 G00 G90 X0 Y0
N10  G01 F2000 X100     G301=20 #FRC=500
N20             Y100    G302=20 #FRC=1000
N30         X0          G301=20 #FRC=500
N40             Y0      G302=20 #FRC=1000
N50  X40
N50  M30
```
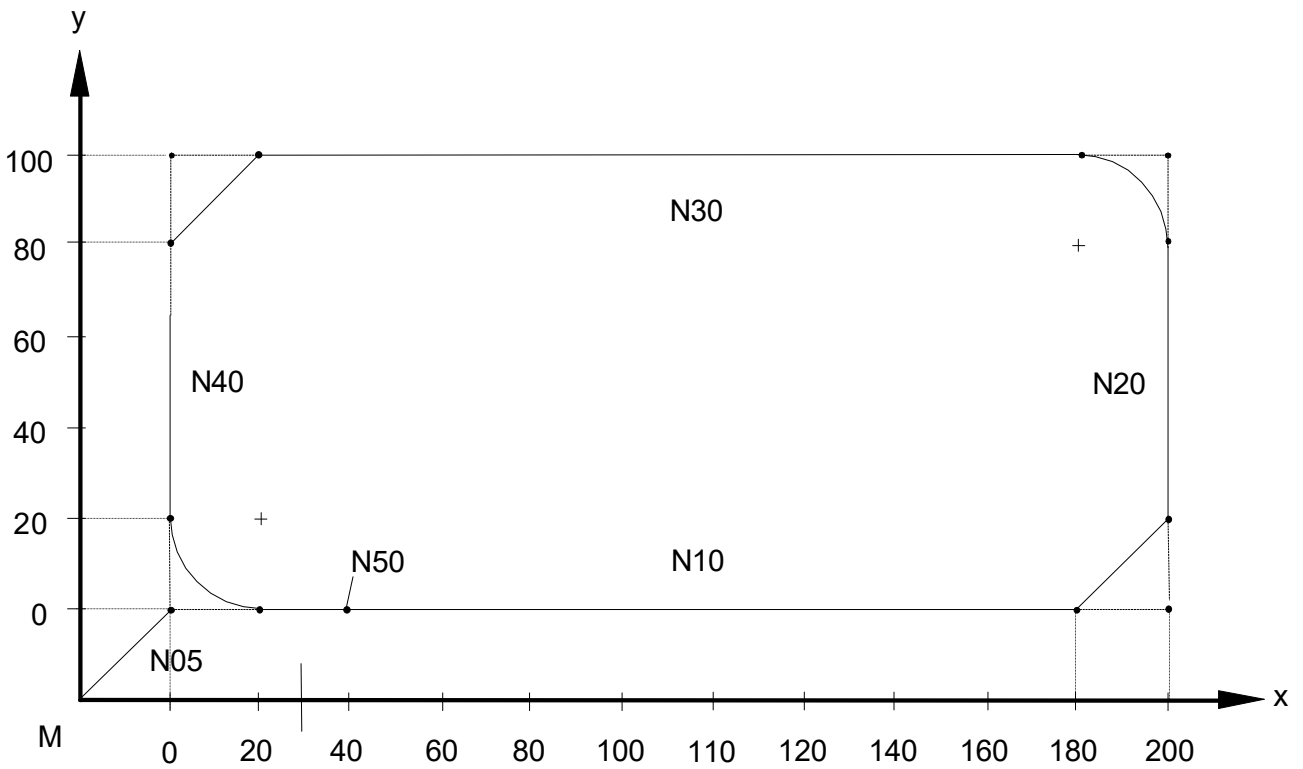
Programming chamfers and roundings with # commands:

Chamfers and roundings are programmed as additional values with# commands. Chamfers and rounding values must always be specified in [mm, inch]; they are non-modal.

commands can be programmed directly in the first motion block. A separate NC block is not required.

A chamfer can be programmed in two ways: either by specifying

- chamfer width (analogous to G301) or
- chamfer length

Syntax:

#CHR=..	Specify chamfer width in [mm, inch]	non-modal
#CHF=..	Specify chamfer length in [mm, inch]	non-modal
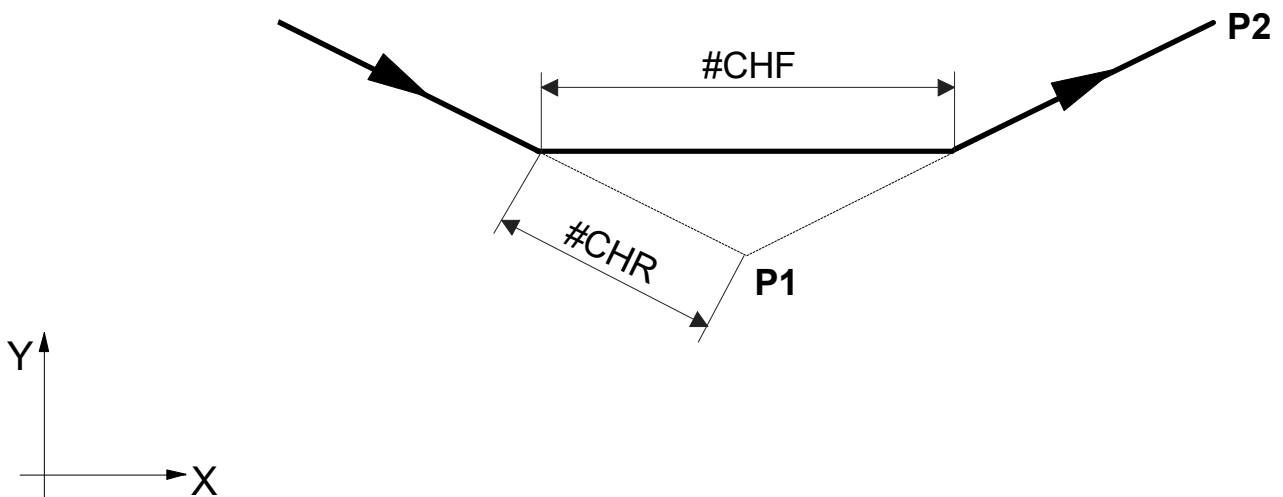
Rounding is programmed by:

`Syntax:`

#RND=..   Specify radius in [mm, inch]   non-modal
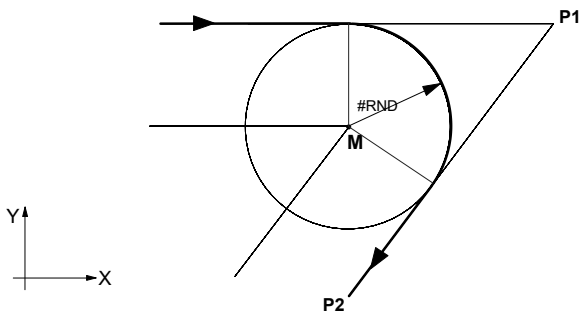
Rectangular pocket with 2 chamfers (#CHR, #CHF) and 2 roundings (# RND) with specific feedrates

```
N05  G17 G00 G90 X0 Y0
N10  G01 F2000 X200        #CHR=20 #FRC=500
N20              Y100  #RND=20 #FRC=1000
N30          X0        #CHF=35 #FRC=500
N40              Y0    #RND=20 #FRC=1000
N50          X40
N60  M30
```
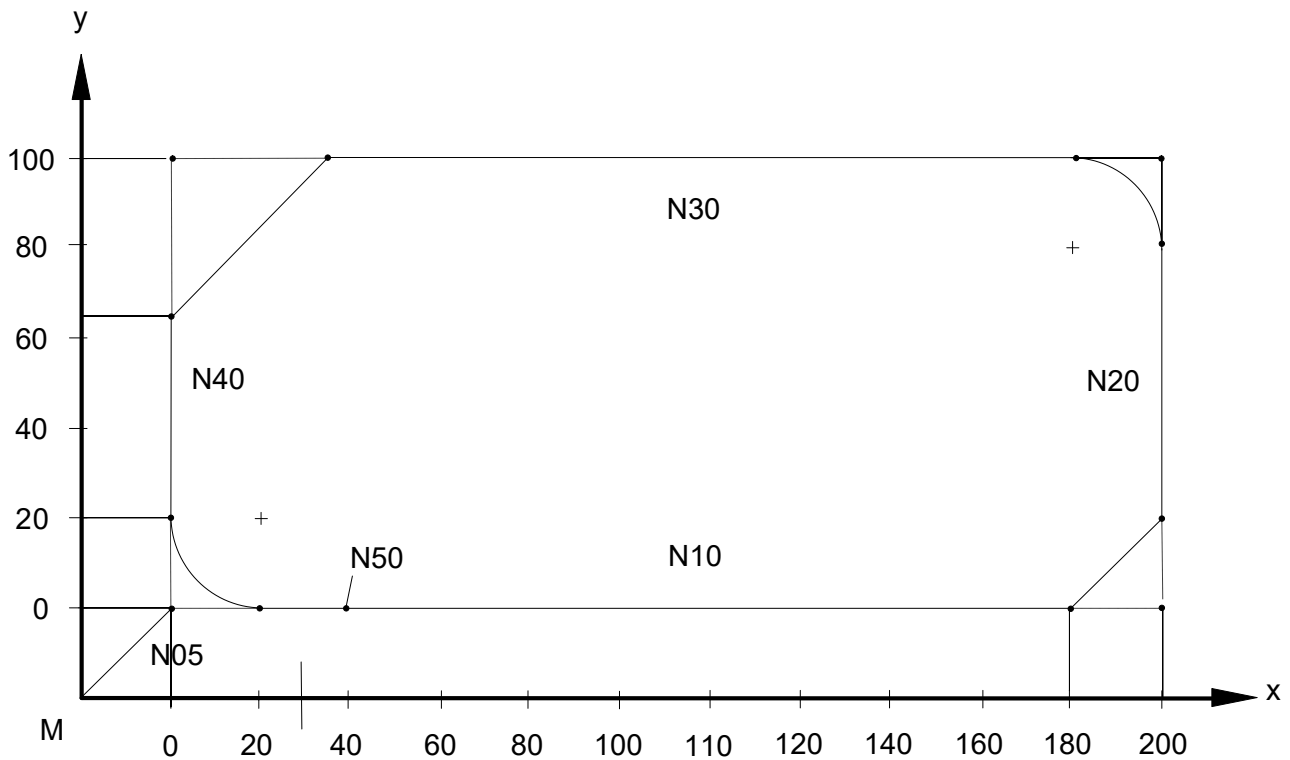
4.22.1 Insert chamfers using G301 as example

With G301 the I word defines the distance between the corner point of the programmed contour and each intersecting point between the inserted straight lines and the contour elements. If one or both the contour elements are arcs, this distance is considered as chord length.

If a given size of the insertion radius or the chamfer results in a direction reversal of one or both contour elements, an error message is output.

```
:
N10 G91 G01 X80 Y-40 F100          ;P1
N20 G301 I40
N30 G01 X80 Y40                    ;P2
:
```
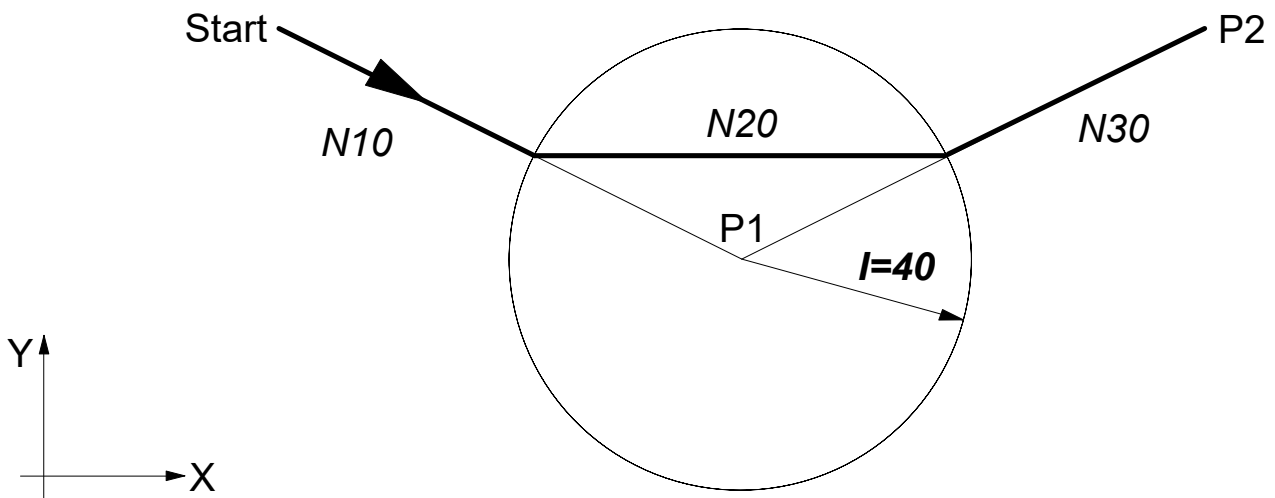
Fig. 54: Insert a chamfer between two straight lines
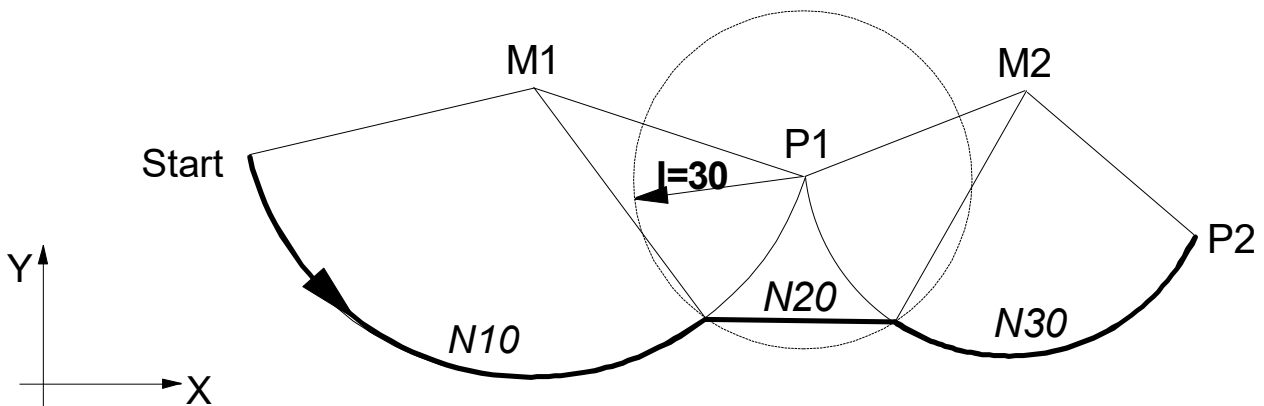
Fig. 55: Insert a chamfer between two arcs

```
:
N10 G01 X20 Y-10 F100              ;P1
N20 X20                           ;P2
N30 G301 I30
N40 X60 Y50                       ;P3
:
```
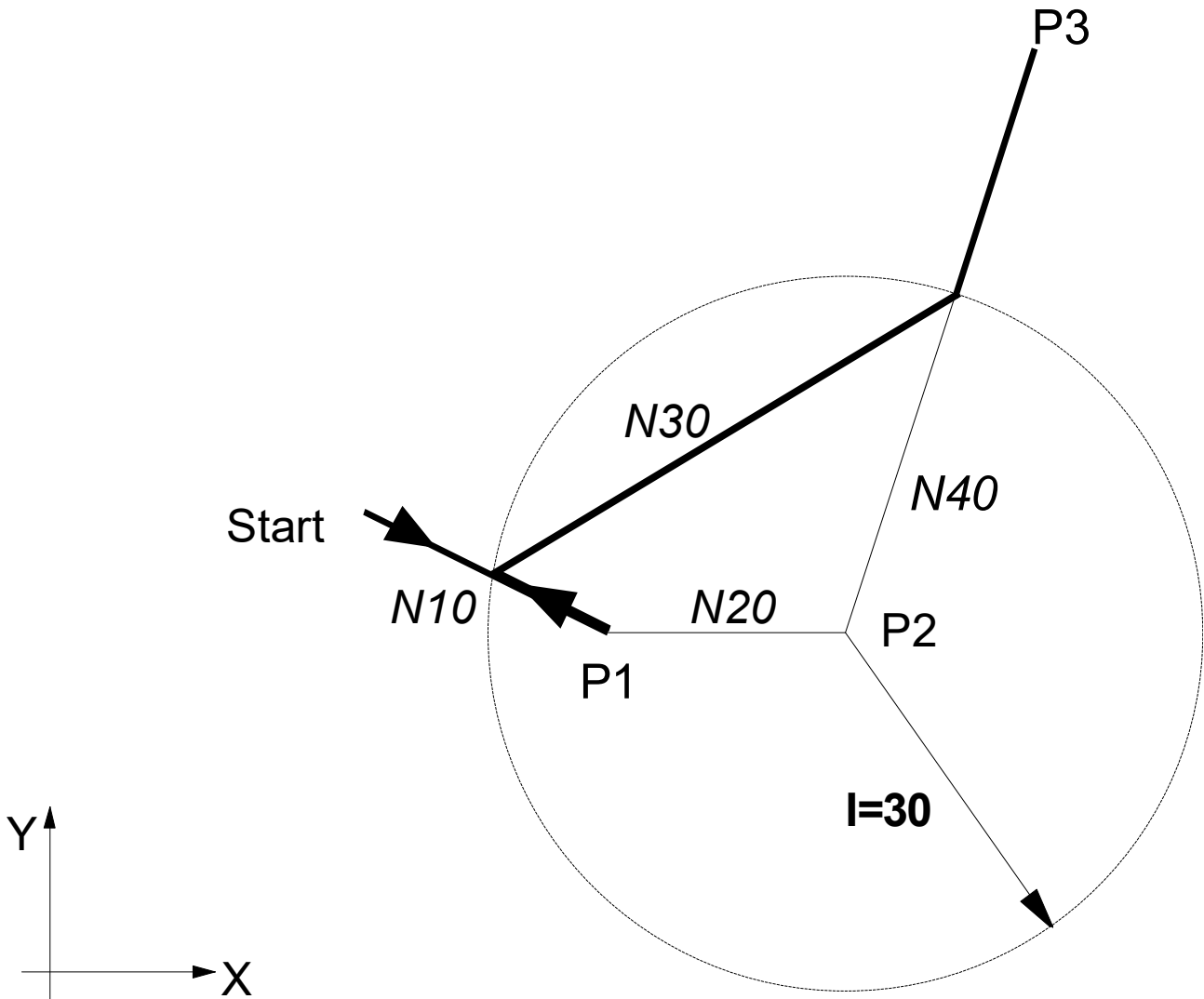
Fig. 56: Error due to direction reversal

```
:
N10 G01 X20 Y-10 F100              ;P1
N20 X20                           ;P2
N30 G301 I30
N40 X60 Y50                       ;P3
:
```

4.22.2   Inserting roundings using G302 as example

With G302 the I word defines the radius of the inserted arc. Its centre point is the intersecting point of the two equidistances with an interval I to the programmed path. The positions of the equidistances are selected so that the programmed contour is retained as far as possible during the contouring operation.

ℹ️ No contouring radius can be inserted in tangential block transitions.

```
:
N10 G91 G01 X60 F100                          ;P1
N20 G302 I30
N30 X-40 Y-55                                 ;P2
:
```
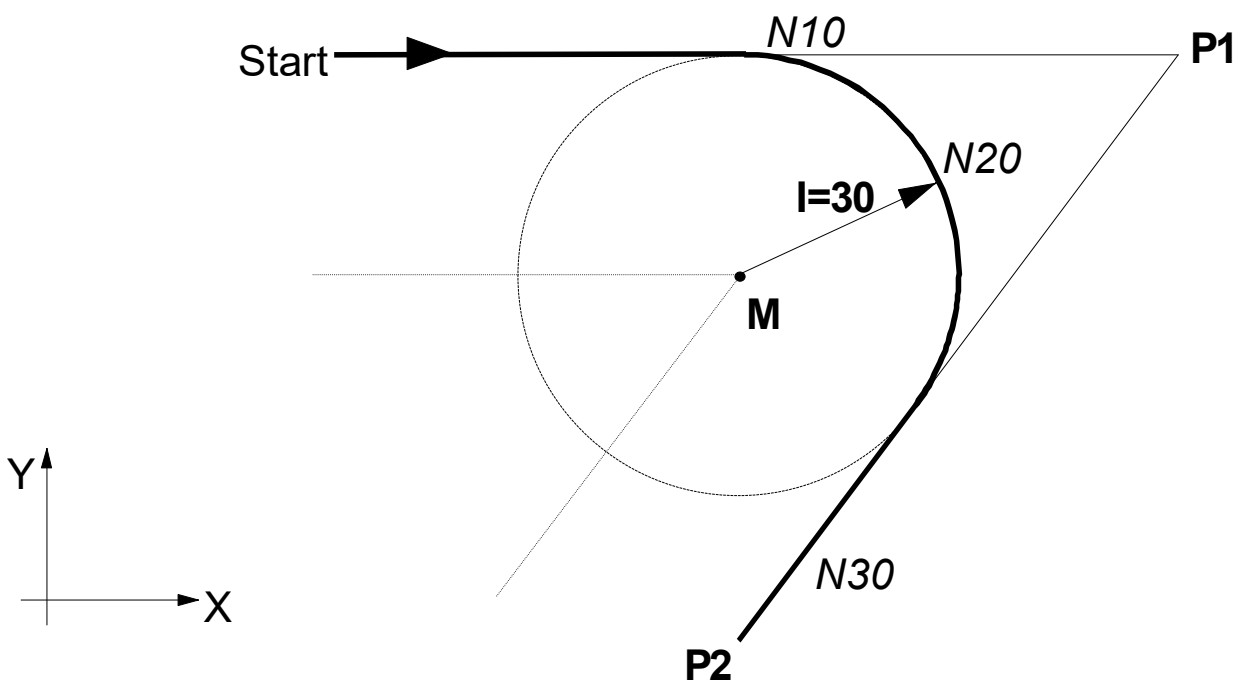

Fig. 57: Inserting an arc between two straight lines
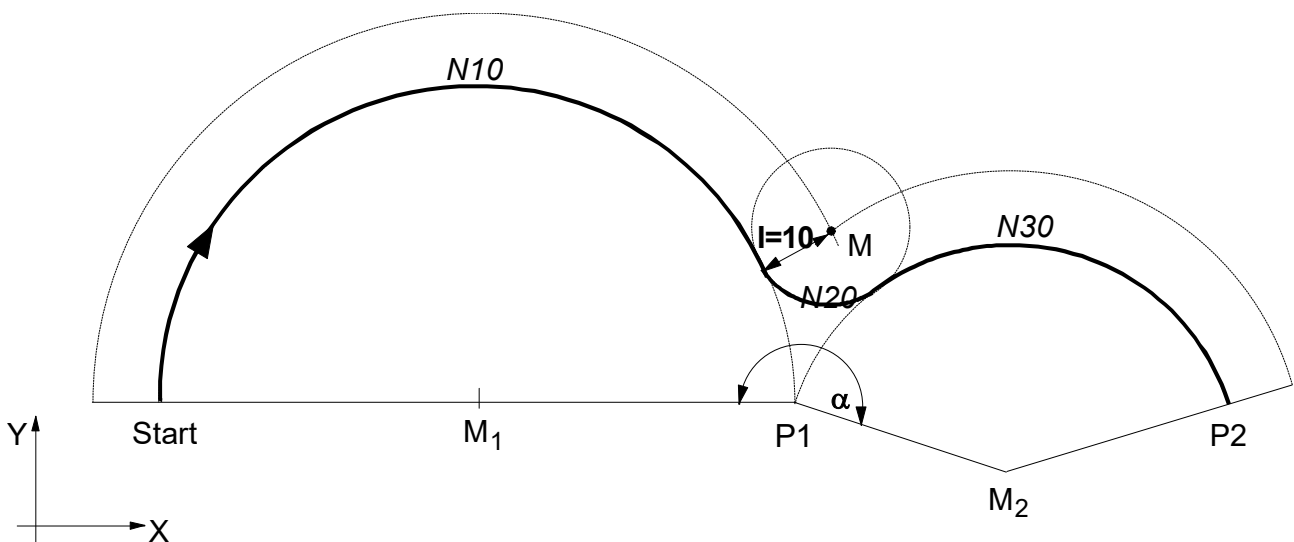
Fig. 58: Inserting an arc between two circles (angle α ≥ 180°)
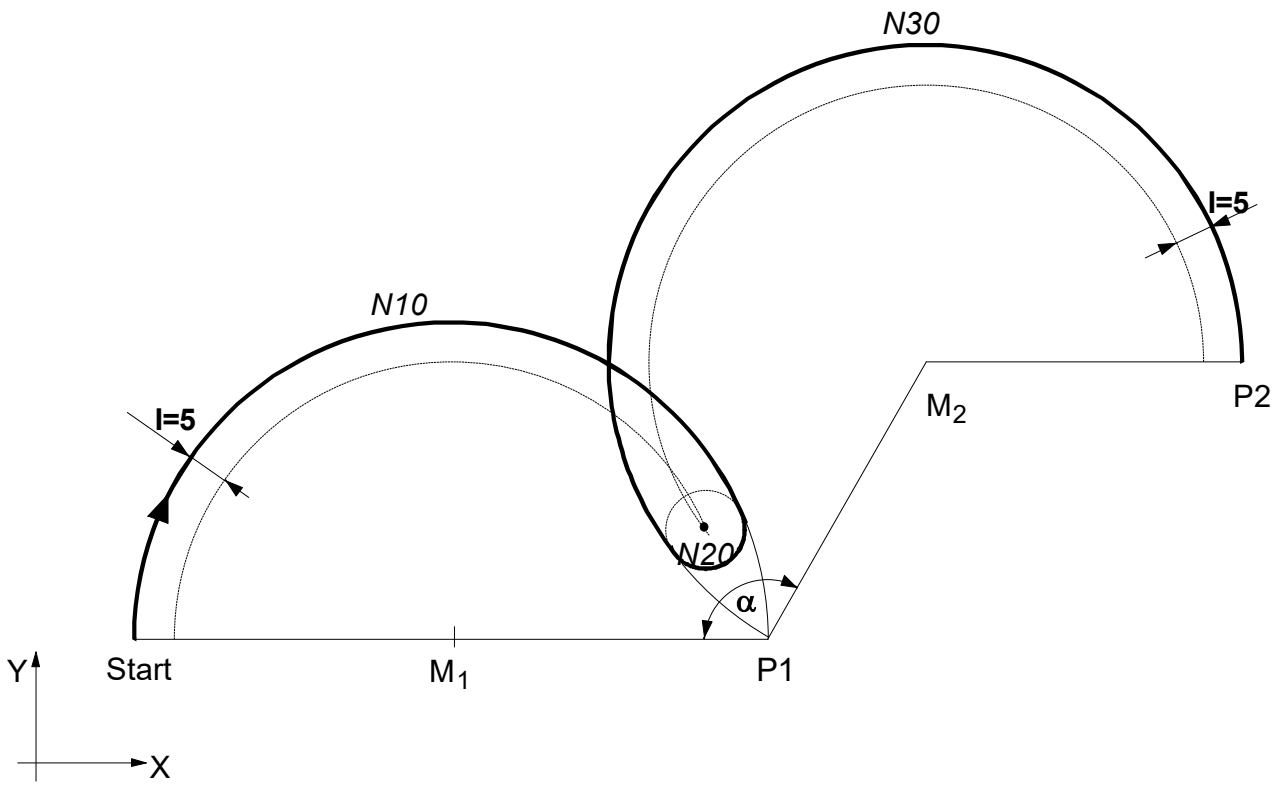
Fig. 59: Inserting an arc between two circles (angle α < 180°)

4.23   Manual mode

Manual mode (HB) permits the external control of single axes with physical elements of manual mode (handwheel, inching keys, joystick). While an interpolation is running, i.e. during the execution of the NC program in HB mode, the operator can add additional set values to the path. The following manual modes are available.

Handwheel function: Any desired path at any desired velocity.

Continuous jog mode: Any desired path at parameterisable velocity.

Incremental jog mode: Pre-specified path at parameterisable velocity

Interruptible jog mode: same as incremental jog mode but with the option of motion interruption.

These manual modes can be activated from the user interface. The corresponding parameters, e.g. resolution stages, velocities, jogging distance etc., are programmed by using appropriate NC commands ("#" commands, "G" commands).

The current manual mode and the axis controlled by a physical manual mode element can be changed at any time. A physical manual mode element can add set values to several axes in several NC channels at the same time. One axis can only be controlled by a single manual mode element and with a single operation element.
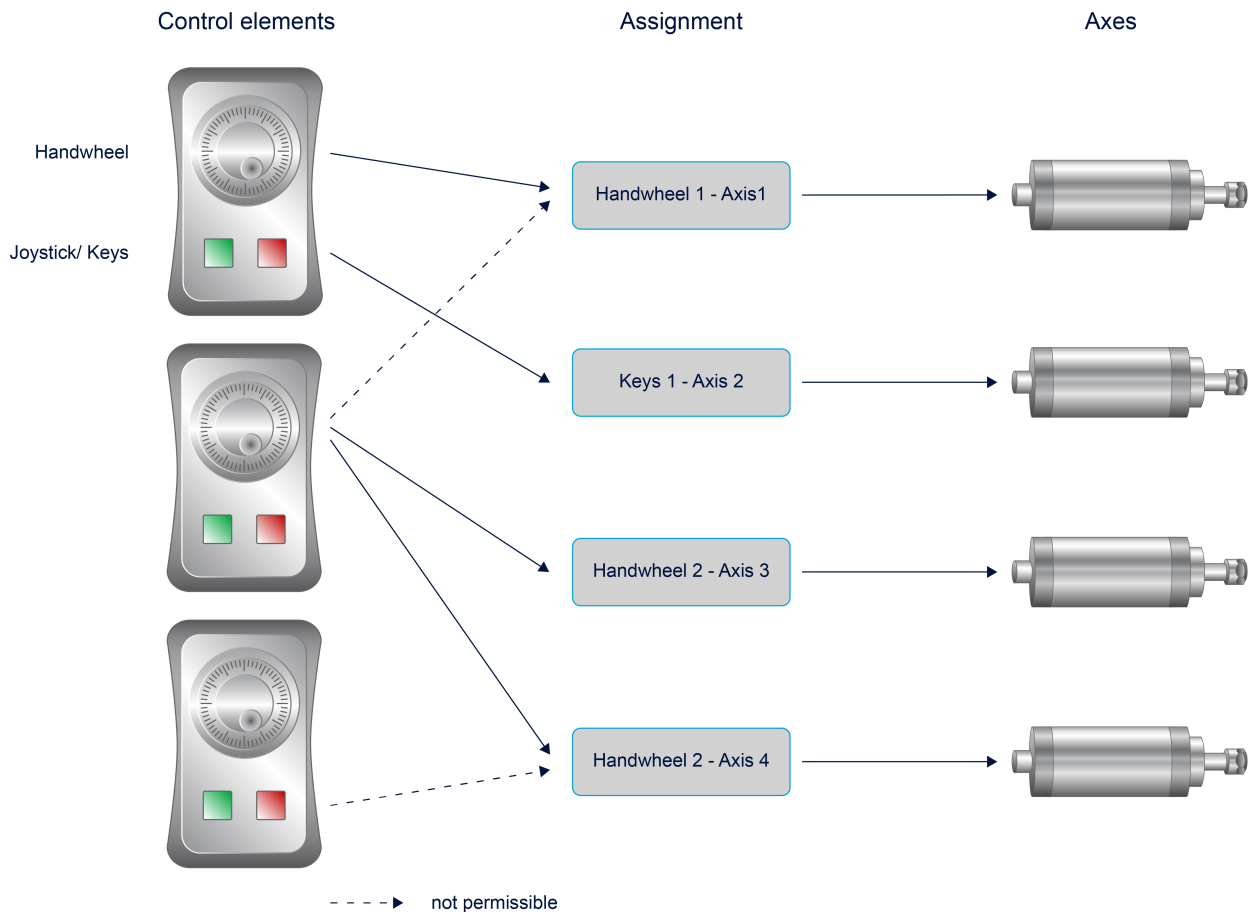
Fig. 60: Manual mode and its options

4.23.1 Selecting/deselecting manual mode with parallel interpolation (G201/G202).

The internal interface between interpolator and manual mode is activated for the programmed path axes. Spindles cannot be programmed.

> **i** As of Build **V2.11.2010.02** the specification of axes **X.. Y..** Replaces the command **#ACHSE [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

G201 selects manual mode for specific axes. After that, manual mode with parallel interpolation is active for these axes until it is deselected by G202.

```
Syntax:
```
G201 *<axis_name>***..** {*<axis_name>***..**}   modal

| *<axis_name>***..** | Select manual mode for some specific axes. The coordinate value is only required for syntax reasons; otherwise it is irrelevant. When selected, axes must always be specified. |

G202 deselects manual mode for all or specific axes.

G202   Deselect manual mode for all axes   modal, initial state

… or for specific axes

G202 <axis_name>.. { <axis_name>..}

| *<axis_name>***..** | Deselect manual mode for specific axes. The coordinate value is only required for syntax reasons; otherwise it is irrelevant. |

After G202, manual mode offsets accumulated in the interpolator while G201 was active are note deleted. This either occurs

- During the next implicit position adjustment in the channel (e.g. initiated by an axis exchange, transformer selection, etc.)\or
- by an explicit set point request with #CHANNEL INIT[CMDPOS].

Selecting/deselecting manual mode with parallel interpolation (G201/G202).

```
.....
G00 X100 Y100
;Status transition of X/Y axes to manual mode
G201 X1 Y1
P1 = 0
$WHILE P1 == 0
;Set up X/Y axes in manual mode
;Program continues by setting P1 to 1 on operating console
$ENDWHILE
;Status transition of all axes to normal mode
G202
....
;Optional: Request command positions, delete manual mode offsets
#CHANNEL INIT[CMDPOS]

G01 Y200 F500
.....
```

4.23.2 Selecting manual mode without parallel interpolation (G200)

The internal interface between interpolator and manual mode is activated for the programmed path axes. Spindles cannot be programmed.

> **i** As of Build **V2.11.2010.02** the specification of axes **X.. Y..** Replaces the command **#ACHSE [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Manual mode without parallel interpolation is selected with G200.

Syntax:

G200	Manual mode for all axes	non-modal

… or for specific axes

G200 <axis_name>**. {**<axis_name>**..}**		non-modal

*<axis_name>***..**	Select manual mode for some specific axes. The coordinate value is only required for syntax reasons; otherwise it is irrelevant.

When G200 is programmed, processing of the current NC program is interrupted in the interpolator. Manual modes can be activated, switched over and deactivated. After axes are traversed in manual mode, a request to continue the NC program can be sent to the interpolator by the control statement "Continue motion".

In this operating mode, the offset limits (maximum motion path in manual mode) are automatically set to the software limit switch positions so that the entire range between software limit switches can be traversed in manual mode. After manual mode is deselected, the previous relative offset limits are again valid.

A parallel interpolation of axes during manual mode is not possible with G200. After the "Continue motion" statement, all axes and operating modes are deactivated so that manual mode is no longer possible.

After G200 is deactivated, the decoder requests all manual mode offsets and command positions from the interpolator and sends the current command positions to all participants in the NC channel. Manual mode offsets are stored in the decoder in the variables V.A.MANUAL_OFFSETS (see also section Axis-specific variables (V.A.) [▶ 550] and can be addressed in this way in the NC program. Manual mode offsets in the interpolator are deleted.

If G200 is programmed without specifying an axis, this command acts on all existing path axes.

Selecting without parallel interpolation (G200)

```
.....
G00 X100 Y100
G200 X1 (Manual mode for X axis)
G01 X200 Y200 F600
G01 Y200 F500
G200 (Manual mode for all axes)
G01 Y500 Z500
.....
```

4.23.3 Reaction at program end (M02, M30)

At the end of the NC program, the axes may not be traversed in manual mode. This means that the command "M30"or "M02" are treated as if an additional G202 was programmed.

4.23.4 Parameterising operating modes

commands for parameterisation may only be programmed when manual mode (G202) is **deselected** for the assigned axes.

4.23.4.1 Handwheel mode (#HANDWHEEL)

> **i** As of Build **V2.11.2010.02** the command **#HANDWHEEL [...]** replaces the command **#SET HR [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#HANDWHEEL [AX=*<axis_name>* | **AXNR**=.. **RES1**=.. **[RES2**=.. **RES3**=..] **]**

AX=*<axis_name>*	Name of manual mode axis
AXNR=..	Logical number of manual mode axis, positive integer.
RES1=..,	Resolution stages (maximum 3),
RES2=..,	[mm/handwheel rev., inch/handwheel rev., °/handwheel rev.]
RES3=..	

Handwheel operating mode

```
When the axis name is specified:
...
G202 X1
#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
...
G201 X1
...

..or when the logical axis number is specified:
...
G202 X1
#HANDWHEEL [AXNR=1 RES1=0.1 RES2=0.2 RES3=0.5]
...
G201 X1
...
```

4.23.4.2   Continuous jog mode (#JOG CONT)

ℹ️ As of Build **V2.11.2010.02**, the command **#JOG CONT [...]** replaces the command **#SET TIP [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#JOG CONT [AX=<*axis_name*> **| AXNR=.. FEED1=.. [FEED2=.. FEED3=..]]**

AX=<*axis_name*>	Name of manual mode axis
AXNR=..	Logical number of manual mode axis, positive integer.
FEED1=..,	
FEED2=..,	Velocity stages (maximum 3) in [mm/min, m/min, inch/min, °/min]
FEED3=..	

Continuous jog mode

```
When the axis name is specified:
G202 X1
...
#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
...
G201 X1
...

..or when the logical axis number is specified:
G202 X1
...
#JOG CONT [AXNR=1 FEED1=1.0 FEED2=1.5 FEED3=2.0]
...
G201 X1
...
```

4.23.4.3 Incremental jog or interruptible jog mode (# JOG INCR)

ℹ️ As of Build **V2.11.2010.02**, the command **#JOG INCR [...]** replaces the command **#SET JOG [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#JOG INCR [AX=<*axis_name*>**| AXNR**=.. **DIST1**=.. **FEED1**=.. **[DIST2**=.. **FEED2**=.. **DIST3**=.. **FEED3**=..] **]**

AX=<*axis_name*>	Name of manual mode axis
AXNR=..	Logical number of manual mode axis, positive integer.
DIST1=..	
FEED1=..,	Pairs of (incremental width; velocity) (maximum 3) in
DIST2=..	[mm;mm/min, mm;m/min, inch;inch/min, °;°/min]
FEED2=..,	
DIST3=..	
FEED3=..	

Incremental jog or interruptible jog mode

```
When the axis name is specified:
...
G202 X1
#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]
...
G201 X1
...

..or when the logical axis number is specified:
...
G202 X1
#JOG INCR [AXNR=1 DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]
...
G201 X1
...
```

4.23.5 Specify offset limits (#MANUAL LIMITS)

i As of Build V2.11.2010.02, the command #MANUAL LIMITS [...] replaces the command #SET OFF-SET [...]. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#MANUAL LIMITS [AX=<*axis_name*> | AXNR=.. NEGATIVE=.. POSITIVE=..]

AX=<*axis_name*>	Name of axis for which the offset limits are valid.
AXNR=..	Logical number of axis for which the offset limit is to be valid,
	Positive integer
NEGATIVE=..	Negative relative offset value. Must be programmed as <0 in [mm, inch]
POSITIVE=..	Positive relative offset value. Must be programmed as <0 in [mm, inch]

This command defines the positive and negative limits for the permissible relative path motion in G201/G202 manual mode for each path axis. The relative negative and positive offset limits refer here to the starting point when manual mode was selected. Offset limits are also considered in G200 by setting the parameter P-CHAN-00114.

i Relative offset limits can be overwritten at any time in the NC program. A sign check is made. Relative offset limits apply to each axis in the programming coordinate system (PCS).

Preset offset limits

```
N090 G01 F1000 X300
N100 #MANUAL LIMITS [AX=X NEGATIVE=-80 POSITIVE=50]
N110 G201 X1
:
```
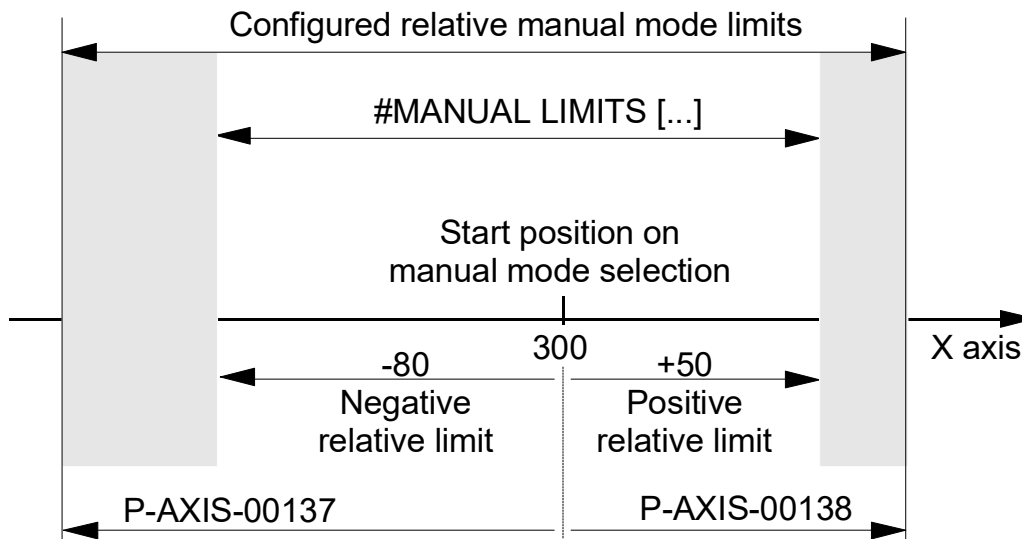
Axes that are operated as rotary modulo axis or as spindles (M3, M4, M5, M19, S...) are not considered in this command. These offset limits apply to all manual modes. They can also be overwritten when manual mode is active (G201). Only a sign check of the programmed values is executed. No other checks are made, e.g. checking for range extensions. Relative offset limits can also be specified using the parameter data record P-AXIS-00138 and P-AXIS-00137 of the axis.

4.23.6   Example of parameterising an axis in manual mode

Example of parameterising an axis in manual mode

```
%main
#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]
#MANUAL LIMITS [AX=X NEGATIVE=-5 POSITIVE=5]
...
G201 X1
...
G202 X1
...
```

The command

#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]

specifies the handwheel resolution stages of the X axis. The resolution stages are 0.1, 0.2 and 0.5 mm/ handwheel revolution.

The command

#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]

specifies the motion velocities for continuous jog mode of the X axis. The velocity stages are 1.0, 1.5 and 2.0 mm/min.

The command

#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]

parameterises incremental jog and interruptible jog mode for the X axis. Incremental widths and velocities are programmed in pairs.

Stage 1:	Incremental width 0.1 mm	Velocity 1.0 mm/min
Stage 2:	Incremental width 0.2 mm	Velocity 1.5 mm/min
Stage 3:	Incremental width 0.5 mm	Velocity 2.0 mm/min

The command

#MANUAL LIMITS [AX=X NEGATIVE=-5 POSITIVE=5]

sets negative offset limit for manual mode to – 5 mm and the positive offset limit to +5 mm.

The following commands select or deselect manual mode for the X axis. If G202 is programmed along, manual mode is deactivated for all axes.

G201 X1

G202 X1

G202

4.23.7   #ECS in connection with manual mode

The initial situation is the use of a rotated PCS. In this situation, the tool is to be moved in the tool direction in manual mode.

By default, manual mode is activated via the PLC with the activation string "G200".
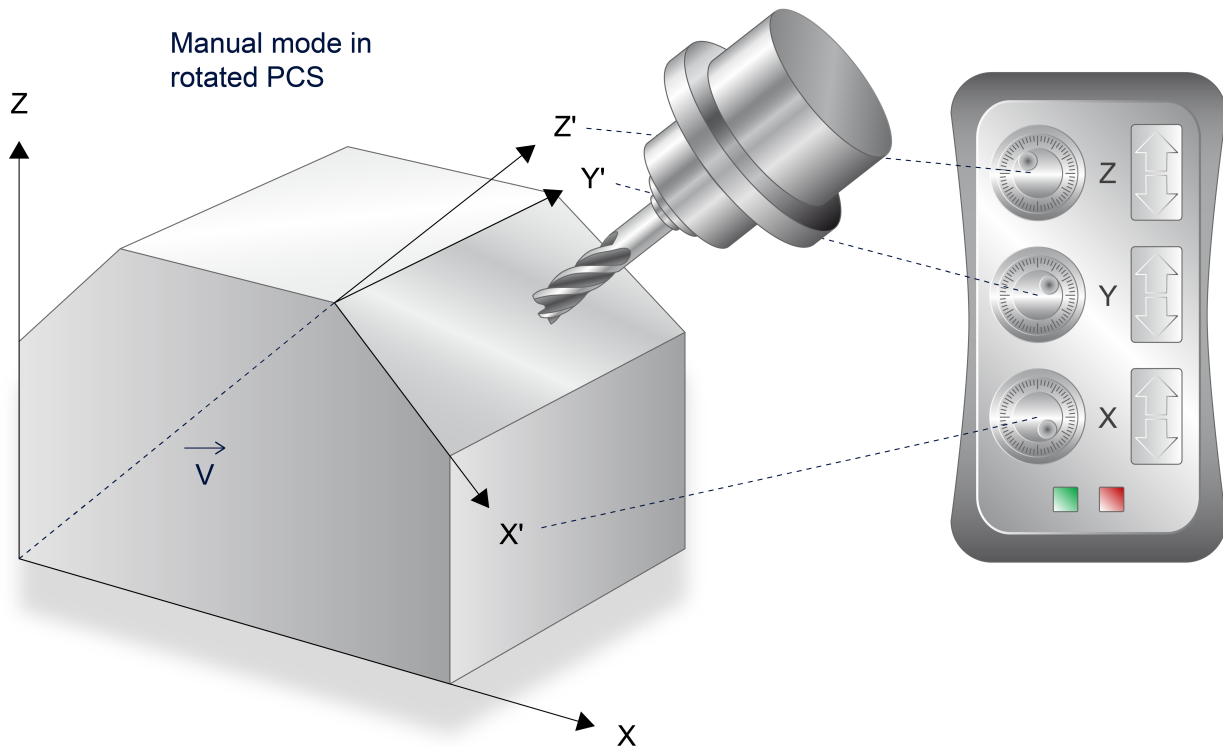
Fig. 61: Manual mode with rotated PCS

An extended sequence in the PLC is required to activate the function in the present case. The sequence must contain the following:

- Activate the kinematic used ID(#KIN ID[<ID>]); alternatively, save the kinematic ID in the tool.
- Activate the effector coordinate system (#ECS ON)
- Activate manual mode (G200)
- Activate the control elements of the 3 axes involved
- Move the Z axis (3rd main axis) in incremental or continuous jog mode

Extended string sequence

```
#KIN ID[< kinematic ID >]$R$N (or D <tool ID>$R$N)
#ECS ON $R$N
G200
```

> ● **It is imperative to terminate the sequence with G200.**
> ℹ **$R§N : String denoting line break in IEC61131-3.**

4.24   Requesting offset, command and actual values

The NC commands described below trigger jobs to include manual mode offset, command and actual values of the interpolator in the operating data of the NC program interpreter. This data can then be accessed by variables in the NC program.

Offset, command and actual values of path axes can only be requested. Axes that are operated as rotary modulo axis or spindles (M3, M4, M5, M19, S...) are not considered in the following commands.

Synchronisation between the NC program interpreter and the interpolator is identical for all the NC commands described. The job is only complete after all the data is adopted in the working data of the NC program interpreter and decoding of the NC program is continued.

Overview of commands and properties:

Command	Channel ini-tialisation	Manual mode offsets in IPO	Updating of …		
			V.A.MANUAL_OFFSETS	V.A.ABS	P parameter, V.S./ V.P./ V.L.
#GET MANUAL OFF-SETS	no	are retained	yes	no	no
#CHANNEL INIT [CMD-POS]	yes	are deleted	no	yes	no
#CHANNEL INIT [ACT-POS]	yes	are deleted	no	yes	no
#GET CMDPOS	no	are retained	no	no	yes
#GET ACTPOS	no	are retained	no	no	yes

4.24.1 Request current manual mode offsets and file to "V.A.MANUAL_OFFSETS[ ]" (#GET MANUAL OFFSETS)

> **i** As of Build **V2.11.2010.0**2 the command **#GET MANUAL OFFSETS** replaces the command **#GET IPO OFFSET**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#GET MANUAL OFFSETS

This command is useful when combined with G201/G202 (manual mode with parallel interpolation). The NC program interpreter requests the current manual mode offsets of all path axes assigned to it by the interpolator. After the values are supplied, they are stored in the NC program interpreter in the variables V.A.MANUAL_OFFSETS (see also section Axis-specific variables (V.A.) [▶ 550]) and can be addressed in the NC program. There is **no** implicit position initialisation of the NC channel. Manual mode offsets are **not** deleted in the interpolator. The variables V.A.ABS (see also section Axis-specific variables (V.A.) [▶ 550]) are not updated.

#GET MANUAL OFFSETS

```
X100
G201
..... Moving in manual mode
G202
#GET MANUAL OFFSETS
G01 X[100 + V.A.MANUAL_OFFSETS.X] F500 (Position initialisation of the)
                                      (NC channel after)
                                      (deselecting manual mode)
```

4.24.2 Request current command positions and file to "V.A.ABS[ ]" (#CHANNEL INIT)

ℹ️ As of Build **V2.10.1504** the command **#CHANNEL INIT [CMDPOS]** replaces the command **#SET DEC LR SOLL**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#CHANNEL INIT [CMDPOS { AX=<axis_name> **| AXNR=.. }]**

AX=<axis_name>	Name of the axis for which the command value is requested.
AXNR=..	Logical number of the axis for which the command value is requested,
	Positive integer.

The NC program interpreter requests the interpolator for the current command positions of all or specific path axes, files them to the working data and initialises the NC channel with these positions. The current command positions in the working data can then be accessed by programming the variable V.A.ABS (see also Section Axis-specific variables (V.A.) [▶ 550]). Any stored manual mode offsets in the interpolator are then deleted automatically. The axis positions correspond to those for zero offset. The values stored in the variable V.A.MANUAL_OFFSETS (see also Section Axis-specific variables (V.A.) [▶ 550]) are not updated.

ℹ️ ***The axis-specific request for command positions is available as of Build V2.11.2038.03.**

ℹ️ If **no** axes are programmed, the command positions for **all** path axes in the channel are requested.

If axes are programmed, only the command positions for them are requested.

#CHANNEL INIT [CMDPOS …]

```
%channel_init_cmd
G01 F1000 X100 Y200
G201
..... Moving in manual mode
G202
#CHANNEL INIT [CMDPOS AX=X AX=Y ]        ;Request for command value by
                                         ;axis specified by name..
#CHANNEL INIT [CMDPOS AXNR=1 AXNR=2 ]  ;..or logical axis number..
#CHANNEL INIT [CMDPOS]            ;....or for all path axes
#MSG ["Cmdpos X:%F, Y:%F ",V.A.ABS.X, V.A.ABS.Y]
M30
```

4.24.3 Request current actual positions and file to "V.A.ABS[ ]" (#CHANNEL INIT)

Syntax:

#CHANNEL INIT [ACTPOS { **AX**=*<axis_name>* | **AXNR**=.. } **]**

AX=*<axis_name>*	Name of the axis for which the actual value is requested.
AXNR=..	Logical number of the axis for which the actual value is requested,
	Positive integer.

The NC program interpreter requests the interpolator for the current actual positions of all or specific path axes, files them to the working data and initialises the NC channel with these positions. Access can then be made to the current actual positions in the working data by programming the variable V.A.ABS (see also Section Axis-specific variables (V.A.) [▶ 550]). Any stored manual mode offsets in the interpolator are then deleted automatically. The axis positions correspond to those for zero offset. The values stored in the variable V.A.MANUAL_OFFSETS (see also Section Axis-specific variables (V.A.) [▶ 550]) are not updated.

ⓘ If **no** axes are programmed, the actual positions for **all** path axes in the channel are requested.

If axes are programmed, only the actual positions for them are requested. Command positions are requested for axes which are not programmed.

NOTICE

The axis can be moved by adopting the actual position as command position. If an axis is adopted in an NC program loop, it may cause the drive to drift.

#CHANNEL INIT [ACTPOS …]

```
%channel_init_act
G01 F1000 X100 Y200

#CHANNEL INIT [ACTPOS AX=X AX=Y ]       ;Request for actual value by
                                        ;axis specified by name..
#CHANNEL INIT [ACTPOS AXNR=1 AXNR=2 ]  ;..or logical axis number..
#CHANNEL INIT [ACTPOS]             ;....or for all path axes
#MSG ["Actpos X:%F, Y:%F ",V.A.ABS.X, V.A.ABS.Y]
M30
```

4.24.4 Request current command positions of axes and file to variables or parameters (#GET CMDPOS)

ℹ️ As of **Build V2.11.2010.02** the command **#GET CMDPOS [...]** replaces the command **#SET IPO SOLLPOS [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#GET CMDPOS [V.S.*<string>* **| V.P** *<string>* **| V.L***<string>* **| P..=***<axis name>*

{ **,V.S.** *<string>* **| V.P.***<string>* **| V.L.***<string>* **| P..=** *<axis_name>* } **]**

The NC program interpreter requests the interpolator for the current actual positions of the specified path axes and files them to the specified self-defined variables (V.S./ V.P./ V.L.) or parameters (P..). There is **no** implicit position initialisation of the NC channel. Any manual mode offsets of the selected axes in the interpolator are not deleted. The variables V.A.ABS and V.A.MANUAL_OFFSETS are not updated.

#GET CMDPOS […]

```
.....
#GET CMDPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1 CV.P.POS2
.....
```

4.24.5   Request current actual positions of axes and file to variables or parameters (#GET ACTPOS)

> **i** **This command is available as of CNC Build V2.11.2022.05 onwards.**

Syntax:

#GET ACTPOS [(**V.S.**<*string*> | **V.P.** <*string*> | **V.L.**<*string*> | **P..**)**=** <*axis_name*>

{ (,**V.S.** <*string*> | **V.P.** <*string*> | **V.L.**<*string*> | **P..**)**=** <*axis_name*> } **]**

The NC program interpreter requests the interpolator for the current actual positions of the specified path axes and files them to the specified self-defined variables (V.S./ V.P./ V.L.) or parameters (P..). There is **no** implicit position initialisation of the NC channel. Any manual mode offsets of the selected axes in the interpolator are not deleted. The variables V.A.ABS and V.A.MANUAL_OFFSETS are not updated.

#GET ACTPOS […]

```
.....
#GET ACTPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1 CV.P.POS2
.....
```

BECKHOFF

4.25   Gear change (G112)

Syntax:

G112 *<axis_name>.. { <axis_name>.. }*   Gear change   non-modal

The function G112 can change gear stages for individual path axes. G112 is also available for spindle axes. However, the special characteristics described in Section <u>Gear change (G112) [▶ 630]</u> must be taken into consideration.

With axes which have no real gear, this functionality can be used to change various axis parameters. For example, controller gain and position lag monitoring can be matched to one another in a separate parameter block and can then be selected by the "gear number" defined in the NC program.

<table><tr><td>*NOTICE*</td></tr><tr><td>G112 may not be programmed with a path condition in the same NC block.</td></tr></table>

Gear change (G112)

```
N50 G112 X4 Y8 (X axis: gear stage 4, Y axis: gear stage 8)
```

4.26   Influence on the look-ahead functionality (G115/G116/G117)

Syntax:		
G115=..	General influencing of the look-ahead functionality	modal
G116 *<axis_name>..* { *<axis_name>.. }*	Influencing the calculation of block transition velocity	modal
G117	Resetting the look-ahead functionality	modal, initial state

Overview of the functionality of the look-ahead function

Various single functions are implemented in the look-ahead function. They limit motion velocities to maximum values in order to always maintain the permissible axis velocities and axis accelerations.

The following single functions influence the velocity profile:

- (1) Calculation of maximum path velocities based on the maximum axis velocities of the axes participating in the motion as specified in the channel parameter list P-CHAN-00071. The velocity on the programmed path is limited so that each of the axes participating in the motion does not exceed its maximum axis velocity.

- (2) Calculation of maximum path velocities on curved paths based on maximum chord error. In this case, a curve radius can be determined on each curve at the point of greatest curvature. A circular section can then be defined on this circle and its starting and end points correspond to a point generated on the path. Provided that the real mechanical system is moved along the associated chord, the height of the associated circular section can be included as an error criterion.

- (3) Calculation of maximum path velocities on curved paths based on the maximum axis acceleration specified in the channel parameter list P-CHAN-00071. The maximum path velocity calculated based on the secant error can still be too high, particularly with a short tracing time and smaller circular movements. The resultant centripetal accelerations would then exceed the permissible axis accelerations.

- (4) Calculation of path transition velocity based on the block transition geometry, taking into account the axes participating in the motion and the limit values specified in the channel parameter list P-CHAN-00071. As a rule, the relevant portions of the single axes alternate in the total motion with a block transition. However, this change in axis portions only occurs in the first tracing cycle on the new path (for linear motion). If it is assumed that the transition should take place without the generated acceleration (e.g. by the slope function), the resultant axis accelerations can be estimated. The maximum path velocity at block transition can then be calculated so that the maximum axis accelerations are not exceeded at the transition.

General case:

Normal mode with look-ahead

After program start all functionalities of the look-ahead function are switched on (G117). This ensures that the parameters specified in the channel parameter list P-CHAN-00071, e.g. maximum axis velocity and maximum axis acceleration, are always maintained.

Special case:

Switch off single functions of the look-ahead function by G115

Influence the look-ahead functionality (G115/G116/G117)

```
Nnn G115 = 0            (Switch off the single functions (2),(3) and (4))
                        (of the look-ahead function)
```

The table below explains the possible combinations of the single functions.

> ℹ The corresponding selected or deselected functions always refer to all axes.

Overview of permissible identification numbers (ID) related to G115:

Single function ID	(1)	(2)	(3)	(4)
0	*			
2	*			*
4	*	*		
6	*	*		*
8	*		*	
10	*		*	*
12	*	*	*	
14	*	*	*	*

*	Single function active		Single function not active

```
Nnn G115 = 2   (Switch off single functions (2) and (3))
Nnn G115 = 12  (Switch off single functions and (4))
Nnn G115 = 14  (Switch off single functions after)
               (program start, cf. G117)
```

Special case:

Influence the look-ahead functionality with G116, G117

The G116 function switches off the calculation of the block transition velocity (single function (4)) for single programmable axes. No reduction in block transition velocity (path velocity) takes place because of "corners" in the path.

> ℹ In this case, the dynamic axis data (axis accelerations) are not normally maintained.

```
N10 G116 X1 Y2          (No reduction in block transition)
                        (rate of different axis)
                        (rates of the axes X or Y)
N20 G01 G91 X100 Y-100 F1000
```

```
N30 X-100
```

In this example, block transition N20/N30 is traversed without any drop in set velocity. The command values change at the block transition in jumps according to the axis portions on the path.

> **_NOTICE_**
> The coordinate value is only required for syntax reasons; otherwise it is irrelevant.

The G117 function switches **all** single function of the look-ahead function back to active (default setting after program start).

```
Nnn G117                    (Enable all single functions (1),(2),(3),(4))
                            (of the look-ahead function)
```

4.27   Override (G166)

```
Syntax:
```

G166	Set override to 100%	non-modal

The G166 function switches off the external influence on path axis overrides **blockwise** and suppresses the effect of programmed override values on <u>the path [▶ 436]</u> or <u>an axis [▶ 769]</u>.

Override (G166)

```
%override_G166_extern
;Assuming: External override 50%
N10 G00 G90 X0 Y0 Z0     ;Rapid traverse 50%
N20 G01 X10 F2000        ;Feed F1000
N30 X20 Y20              ;Feed F1000
N40 G166 Z30             ;Feed F2000, override 100%
N50 X30                  ;Feed F1000
N60 G166 Y30             ;Feed F2000, override 100%
N70 G166 G00 X0 Y0 Z0    ;Rapid traverse 100%
N80 G01 F3000
N90 X10 Y20 Z30          ;Feed F1500

M30
MN10 G00
...
o%override_G166_path

;Path override: G01 120%, G00 75%
N10 #OVERRIDE [FEED_FACT=120 RAPID_FACT=75]

override_G166_path

;Path override: G01 120%, G00 75%
N10 #OVERRIDE [FEED_FACT=120 RAPID_FACT=75]

N20 G01 X10 Y10 Z10 F1000   ;Feed F1200
N30 G166 X20 Y20 Z20        ;Feed F1000, override 100%
...
N50 G00 X50                 ;Rapid traverse 75%
N60 G166 Y50 Z50            ;Rapid traverse 100%
...
M30

%override_G166_ax

;Axis override for X: G01 20%, G00 60%
N10 X[OVERRIDE FEED_FACT=20 RAPID_FACT=60]

N20 G00 X10                 ;Rapid traverse in X with override 60%
N30 Y10 Z10                 ;Rapid traverse in Y/Z with 100%
N40 G166 X20                 Rapid traverse in X with override 100%

N50 G01 X30 F2000           ;Feed in X with F400, override 20%
N60 Y20 Z20                 ;Rapid traverse in Y/Z with F2000 (100%)
```

```
N70 G166 X40            ;Feed in X with F2000, override 100%
...
M30
```

4.28 Cycle synchronisation at block end (G66)

Syntax:

G66	Synchronise motion block end with IPO cycle	non-modal

The G66 function adapts the velocity in the motion block so that each interpolation point is always reached in the interpolator (IPO) cycle.

A motion block programmed in combination with G66 can be executed reproducibly in program loops without interference or undesirable velocity fluctuations.

4.29   Rotate the coordinate system in the plane (G68/G69)

ℹ **This function is available as of CNC Build V3.1.3079.33.**

This function rotates a coordinate system in the current plane (G17/G18/G19). Contours programmed in the machine coordinate system can be adapted quickly and easily to workpieces in offset positions.

Contour rotation acts directly on the programmed axis coordinates (contour) **before** all other contour-influencing functions, i.e. all offsets and mirroring operations are not influenced by the rotation and can be used as before (*).

Rotation may also be applied within an already rotated coordinate system (#(A)CS).

A change of plane with G17/ G18/ G19 automatically deselects an active contour rotation and a warning is output.

As a alternative to G68/ G69, contour rotation can be programmed using #ROTATION [▶ 380].

```
Syntax (example in G17):
```

| **G68 R.. X.. Y..** | Select contour rotation | modal |
| **G69** | Deselect contour rotation | modal, initial state |

R..	Rotation angle in degrees [°], absolute. If no angle is specified, the value 0° is set. The rotation angle has no influence on already programmed circle radii.
X.. Y..	Absolute coordinates of the centre of rotation in [mm, inch] in the main axes of the current plane.
	The following applies: G17 - X and Y, G18 - Z and X, G19 - Y and Z
	With coordinates that are not programmed, the current actual position is set as the centre of rotation.
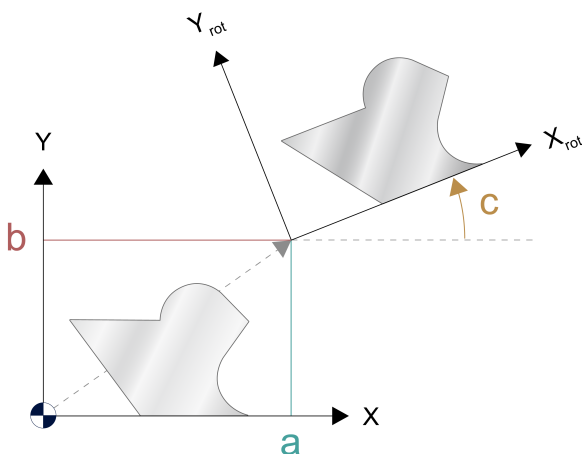
Fig. 62: Significance of rotation parameters in the main plane (example G17):

| a: X.. | b: Y.. | c: R.. |

The programmed rotation parameters can be read with the following variables:

V.G.ROT_ACTIVE	Contains the value 1 if a rotation is active
V.G.ROT_ANGLE	Rotation angle
V.G.ROT_CENTER1	Offset of the first main axis relative to the centre of rotation
V.G.ROT_CENTER2	Offset of the second main axis relative to the centre of rotation

---

ⅰ (*) It makes no difference whether the offsets (e.g. G54, G92 etc.) were programmed before or after G68; they always act in the axis directions of the basic coordinate system of the machine (MCS).

In addition, tool offsets always act independently of P-TOOL-00010 in the axis directions of the MCS.

Rotation in a plane (contour rotation)

```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

%ang1.nc
N100 G53 G17
N110 LL part
N130 G68 R-45 X10 Y100
N140 LL part
N150 G21 (mirroring of X coordinates)
N160 LL part
N170 G69
M30
```
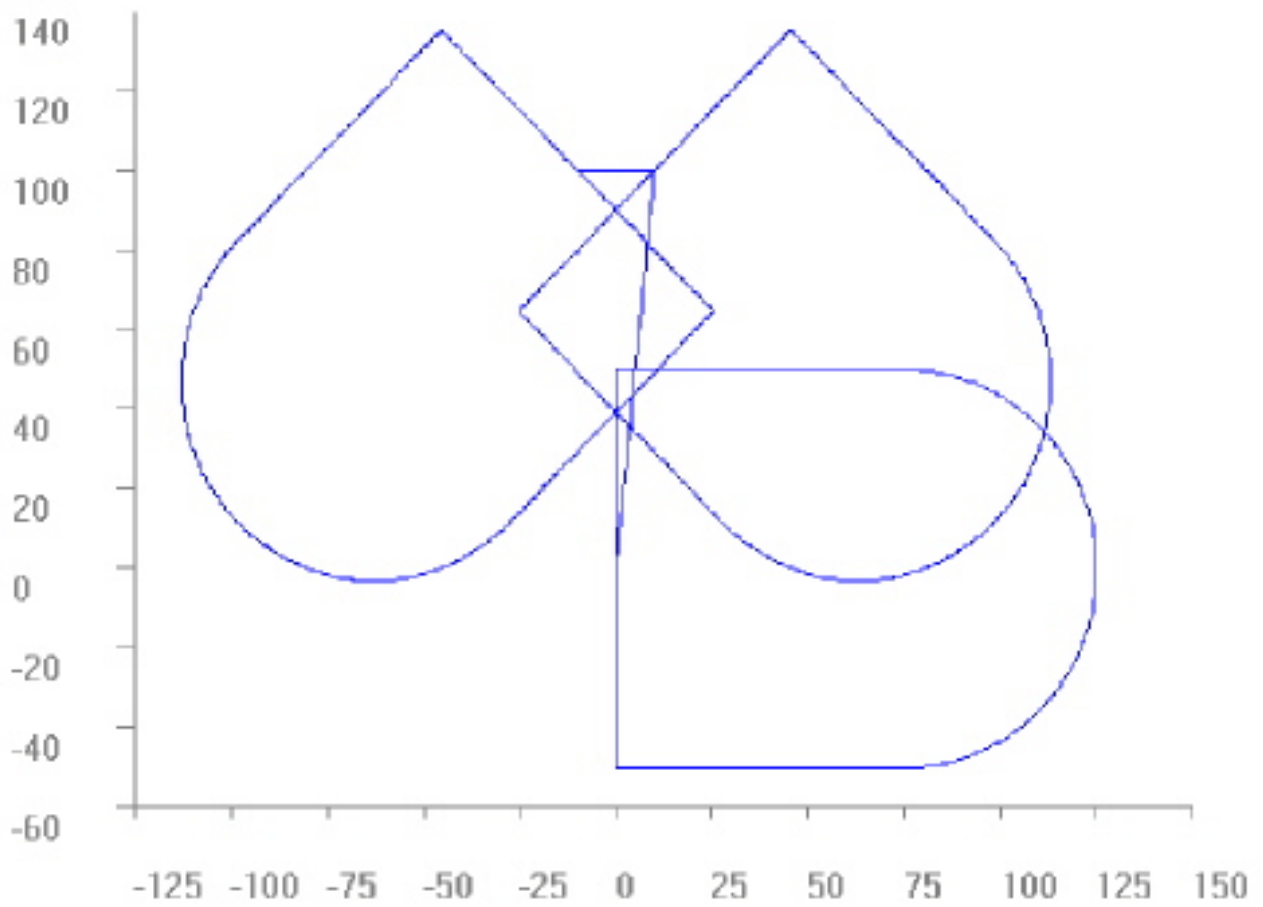
5 Switching and supplementary functions (M/H/T)

A complete list of G functions is contained in the overview of commands in the Appendix under M functions (M..) [▶ 791].

5.1 User-specific M/H functions

M/H functions are mostly determined by programming the "Programmable Logic Controller" (PLC). This controller contains an interface ("Window to PLC") information on the programmed M/H functions in the current NC block. In addition, it provides synchronisation conditions which control synchronisation between the NC and the PLC.

Syntax:

M..

or

H..

M/H functions can be programmed with general mathematical expressions. In particular, this allows the assignment of values by means of parameters, e.g. MP10. Before access to the M/H function, the calculated numerical value is rounded off and converted into an integer. The smallest permissible value is 0 (zero). Negative values generate an error message.

The maximum number of M/H functions per NC channel [6] [▶ 786]-8.1/-8.2 and the maximum number of M/H functions per NC block [6] [▶ 786]-8.3 are fixed.

The EXIST function (see Section Arithmetical expressions <expr> [▶ 32]) checks whether a variable exists at all.

User-specific M/H functions

The EXIST request first checks whether a user-specific M function is at all defined in the channel parameter list P-CHAN-00027 before the M function is executed.

```
...
N10 G90 Y0
N20 $IF EXIST[M80] == TRUE
N30 X0 Y0 Z0 M80 (M80 is available)
N40 $ELSE
: …
N60 $ENDIF
...
M30
```

The following M functions have a predefined meaning in the NC program, i.e. they are not freely available. The user can only assign their synchronisation mode P-CHAN-00027.

Syntax:

M00	Programmed stop
M01	Optional stop
M02	Program end, stop the machine
M10, M11	Latch axes
M17	Subroutine end
M29	Subroutine end
M30	Program end, stop the machine

5.1.1 Programmed stop (M00)

M00 causes an interrupt in a running NC program, e.g. to execute a measurement or to remove chips. Machining continues when the "CONTINUE MOTION" button is pressed. If M00 is present in a block with motion statements, the interruption occurs after the motion statement is executed.

5.1.2 Optional stop (M01)

M01 acts in the same way as M00. However, the effectiveness of M01 must be activated on the operating console of the controller.

5.1.3 Program end (M02/M30)

M02 and M30 are identical. In the main program and subroutine, they cause a reset to the start of the main program and reset the NC to initial state.

In addition, a signal can be sent to the PLC by configuration. For example, the PLC uses this signal to synchronize a workpiece change.

5.1.4 Subroutine end (M17/M29)

M17 and M29 terminate a subroutine properly. See also Section <u>Subroutine techniques</u> [▶ 196]

A signal can also be forwarded to the PLC by configuration.

5.1.5   Call a tool change program (M06)

When configured as required (P-CHAN-00118), the M function M06 is not interpreted as a technology function. Instead, it calls a global subroutine. In this subroutine the user can program all the actions required for a physical tool change.

ℹ Depending on the version, the processing lines of the tool change program are masked or visible in the display when M6 is executed in the default setting while the NC program is running or in single block mode cycle. When active display is off, only the M6 call is displayed during this time.

This feature is switchable by the channel parameter P-CHAN-00211.

ℹ **In TwinCat systems, all cycle lines are visible in the display by default.**

5.2   Axis-specific M/H functions

In general, user-specific M/H functions programmed in conventional DIN syntax are handled and executed for specific channels.

If the user wants to force axis-specific handling for each user-specific M/H function, he can also preassign them in the channel parameters P-CHAN-00039 and P-CHAN-00025.

```
Syntax:
```
m_default_outp_ax_name[*<m_nr>*]   *<axis_name>*

or

h_default_outp_ax_name[*<h_nr>*]   *<axis_name>*

<m_nr>	User-specific M function with axis-specific effect.
<h_nr>	User-specific H function with axis-specific effect.
<axis_name>	Axis name of the axis on which the M/H function is to act. Path axes and spindle axes are permitted in this case.

Axis-specific M/H functions

Channel parameter list [1] [▶ 786]:

```
:
# Definition of axis-specific M functions
# ========================================
m_default_outp_ax_name[20] S2
m_default_outp_ax_name[21] S3
m_default_outp_ax_name[22] Z
:
# Definition of axis-specific H functions
# ========================================
h_default_outp_ax_name[10] X
h_default_outp_ax_name[11] Y
h_default_outp_ax_name[12] Z
:

N.. S1000 M3 M20 M21 M22 H10  ;S and M3 act on the main spindle axis
                             ;M20 acts on the S2 spindle axis
                             ;M21 acts on the S2 spindle axis
                             ;M22 acts on the Z axis
                             ;H10 acts on the X axis
:
```

i If an axis-specific M/H function is programmed in a spindle-specific bracket expression, the default setting is ignored and the M/H function acts on the corresponding spindle axis.

Moreover, for path axes, the NC program can also force an axis-specific output of M/H functions by the use of bracketed expressions:

```
Syntax:
```
<axis_name> **[** [**M**..] [**H**..] { [**M**..] [**H**..] } **]** { *<axis_name>* [...] }

<axis_name>	Axis name of the axis on which the M/H function is to act. Only path axes are permitted in this case.
M..	User-specific M function with axis-specific action.
H..	User-specific H function with axis-specific action.

```
:

N10 S1000 M3 X100 X[M20 H12] Y[H10]        (S, M3 act on the main spindle axis)
                                           (M20, H12 act on the X axis)
                                           (H10 acts on the Y axis)

:
```

This gives the user the option of executing his specific M/H functions very flexibly by programming or configuring for specific axes or specific channels.
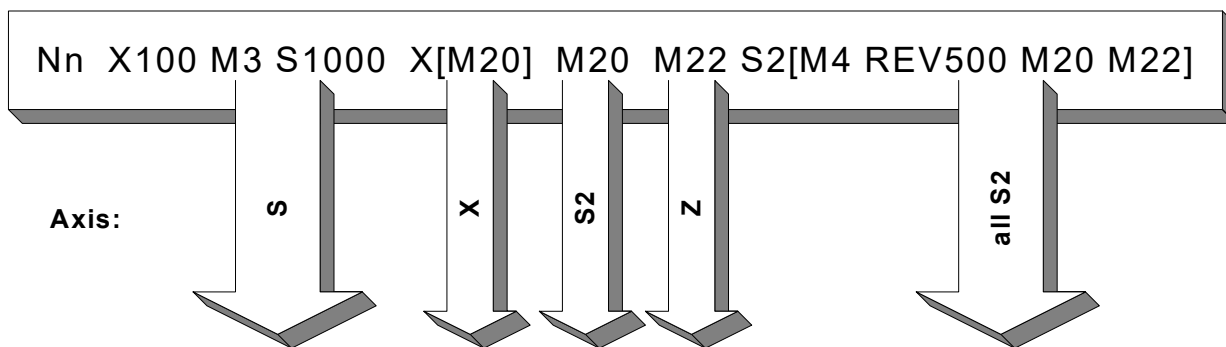
5.3   M/H functions with optional additional information

With M/H functions, an additive value can be optionally programmed in the NC program and made available to the PLC on the technology interface in combination with the M/H function.

The additive value can be programmed in all user-specific M/H functions and in the internal M functions M00, M01, M02, M17, M29 and M30. The M/H functions can then be used without restriction in both channel-specific and axis-specific programming syntax.

The following M functions may only be programmed without an additive value:

- spindle functions M03, M04, M05 and M19
- gear change functions M40-M45 and
- the subroutine function M6 (if configured as such in P-CHAN-00118)

```
Syntax:
```
M.. [**=** *<additive_value>*]

or

H.. [**=** *<additive_value>*]

M..	User-specific M function
H..	User-specific H function
= <additive_value>	Optional additive value. It can be programmed directly as a negative or positive integer or as a general mathematical expression.

M/H functions with optional additive information

```
%m_h_add_fct

(M functions with additive value)
N10 M52=-345
N20 M12=123      ;with channel parameter m_default_outp_ax_name[12] Z
N30 M10=321      ;with channel parameter m_default_outp_ax_name[10] S
N35 P1 = 567 P2 = 345
N40 X[M54=P1]
N50 S[REV 1000 M03 M63=-789]
N60 M12=123 M10=321 M52=-345 X[M54=567] S[REV 1000 M03 M63=-789]
N70 M63=-789 M52=-P2 M54=567
N80 X[M52=-345 M54=567] Y[M63=-789] S[M05 M63=789 M54=-567] M54 M63


(H functions with additive value)
N110 H5=-345
N120 H6=123      ;with channel parameter h_default_outp_ax_name[6] Z
N130 H9=321      ;with channel parameter h_default_outp_ax_name[9] S
N135 P3 = 567 P4 = -345
N140 X[H7=P3]
N150 S[REV 1500 M04 H8=-789]
N160 H6=123 H9=321 H5=-345 X[H7=567] S[REV 1500 M04 H8=-789]
N170 H8=-789 H5=P4 H7=567
N180 X[H8=-789 H4 H5=-345] Y[H7=567] S[M05 H5=345 H7=567] H3 H8
```

```
(Mixed M/H functions with additive value)
N200 X[M52=-345 H4 H8=-789 M54=567 H5=345] H3=333 M54=444 H7=567 M63

(M/H functions with additive value in axis-specific function (INDP))
N05 X[INDP G90 G01 FEED=2000 POS=555 M54=151 H8=-181]

N999 M30=111
```

This additive information can be read in the PLC via the data of the M/H function.

See also [HLI// Data of the M/H function]

5.4 Tool position selection (T)

Syntax:

T..	Select tool position	modal

The tool command determines the tool required for the machining step. The tool number is forwarded externally to the PLC which loads the selected tool to a tool magazine for a tool change. It should be noted that the T word itself has no influence on the internal controller calculation of tool geometry (tool data). The D word [▶ 470] is used for this purpose. However, it is activated automatically by the T word if P-CHAN-00014 is programmed accordingly.

The T word initiates no tool change. Normally, this takes place with machine function M06.

Tool data [5] [▶ 786], [6] [▶ 786]-9.7 is stored ion the tool data table of the decoder [6] [▶ 786]-9.1. In addition, there is also the alternative option of communicating with a decentralised (external) tool data management system provided by the user and requesting tool data via an additional interface. The internal or external access to tool data is parameterised P-CHAN-00016.

In order to avoid waiting periods between the time that tool data is loaded and then transferred, the tool to be externally loaded normally requests one or more motion blocks before the tool data is actually included in the calculation P-CHAN-00087. If several possibly different T numbers are programmed before the D number of the first corresponding T number appears, all T numbers before the last one programmed are ignored. This means that the data supplied by the external tool management system is deleted if the tool D number does not coincide with the T number programmed.

6   Velocities (F/E)

```
Syntax:
```

| **F..** | Feedrate in the block | modal |
| **E..** | Feedrate at block end | non-modal |

With interpolation modes G1 [▶ 55], G2, G3 [▶ 56] or for G100 [▶ 95], the programmed paths are traversed at the path velocity agreed in the F word. The F and E words are specified

- in [mm/min, m/min, inch/min] for translatory axes
- in [°/min] for rotary axes.

The F word is modal. Programming in mm/min or m/min can be configured with the channel parameter P-CHAN-00108.

The command G93 [▶ 149] can also specify a machining time by the F word instead of a path velocity. The description in contained in the Section "G functions".

The path velocity at block end is programmed by the E word. The value is block-specific, i.e. non-modal. If the E word is not specified or the programmed value is greater than the F word, the E word is assigned the value of the F word

> **ℹ** The E word is only effective in combination with G94 [▶ 149].
>
> Due to active contour-changing functions such as G301, G302 [▶ 150], G41, G42 [▶ 476], G261 [▶ 122], #HSC [▶ 244] [OPMODE 1…], inserted contour elements adopt the E feedrate of the main blocks as the new F feedrate.
>
> For all other spline and HSC functions, it is recommended to avoid the use of the E feedrate.

Values can be assigned to the F and E words either directly or for each parameter. In this case, decimal numbers are also permitted (REAL format).

> **ℹ** The PLC can also specify path feed externally and also weight it with the NC command #FF [▶ 459].

Velocities (F-, E-)

```
N10 X200 G01 F1000 G90  ;F feedrate 1000 mm/min
N20 X300
N30 X350 F800           ;Lower F feedrate from 1000 to 800 mm/min
```
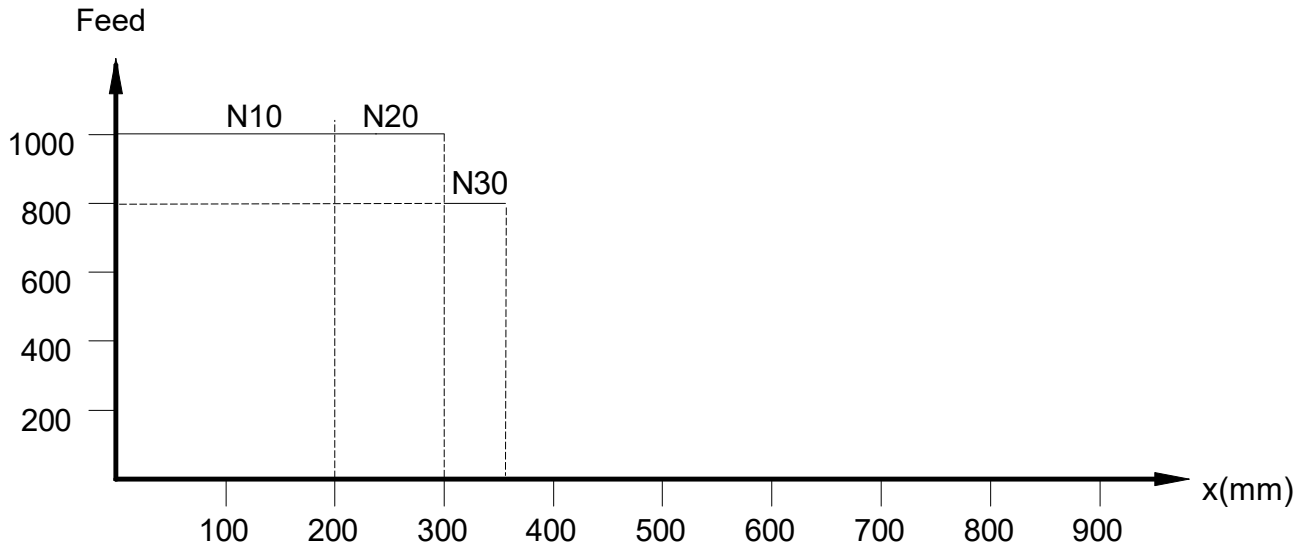

Fig. 63: Program feedrate using F word

```
N05 G01 G94 G90
N10 X200 F1000 E500     ;F feedrate 1000 mm/min, E feedrate 500 mm/min
N20 X250 E400           ;F feedrate 1000 mm/min, E feedrate 400 mm/min
N30 X350 F800 E300      ;F feedrate 800 mm/min, E feedrate 300mm/min
X400 E500               ;F feedrate 800 mm/min, E feedrate 800mm/min
```
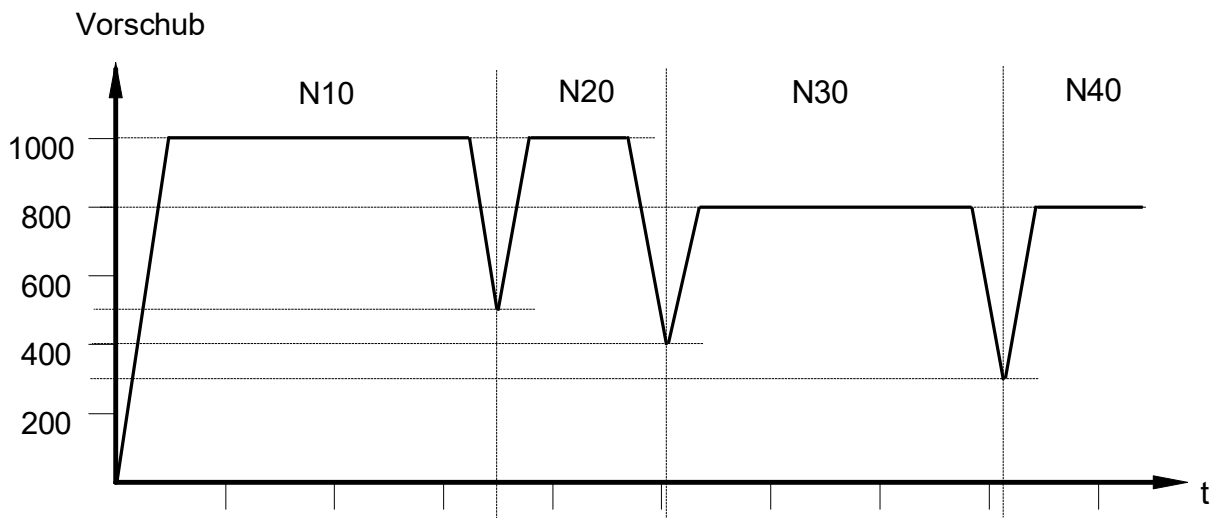

Fig. 64: Program feedrate using F and E words

```
N10 G01 G94 G90 G261
N20 X100 F1000 E200      ;F feedrate 1000 mm/min, E feedrate 200 mm/min
N30 Y100 E200             ;F feedrate 1000 mm/min, E feedrate 200 mm/min
N40 X0                    ;F feedrate 1000 mm/min
N50 Y0 E200             ;F feedrate 1000 mm/min, E feedrate 200mm/min
N60 X50
N70 G260
```
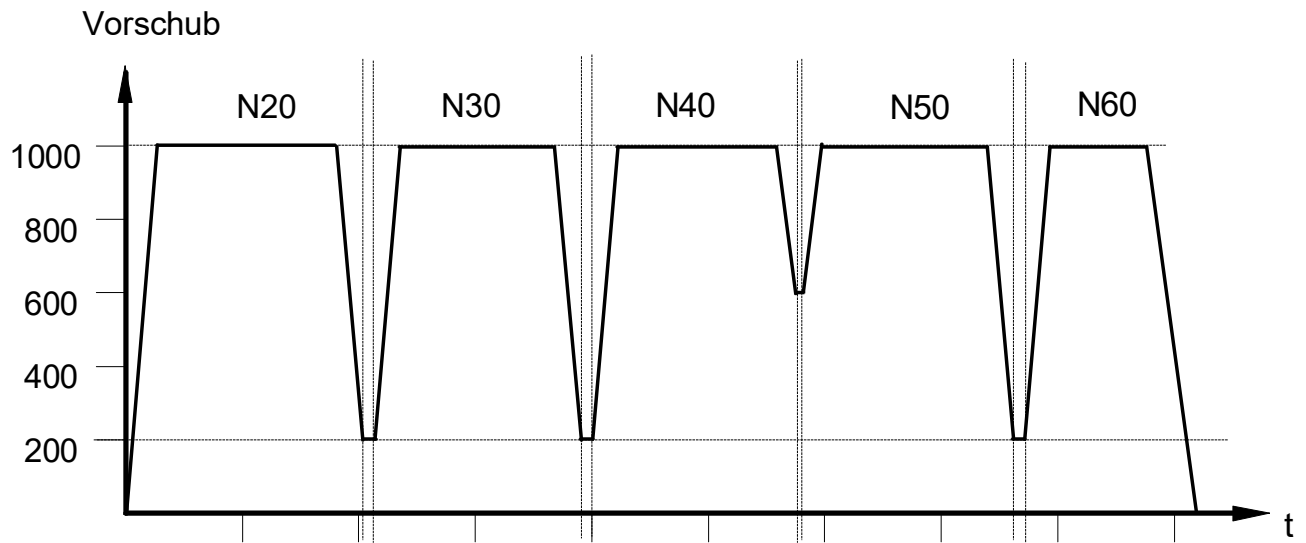

Fig. 65: Effect of E word on inserted contour elements (here G261)

7   NC block numbers (N)

The NC block number can be programmed with the address letter "N". If configured, this number is entered both in the display data as well as in the error messages (alternatively the offset value in the file can also be used for this purpose).

N..

The block number can be programmed using general mathematical expressions. In particular, this permits value assignment by parameters e.g. the run parameter in program loops. The calculated numerical value is automatically rounded off internally and converted into an integer.

NC block numbers (N function)

```
N10 G01 X200 F500
N20 $FOR P1=1, 10, 1
N[P1*100] XP1
.
.
N30 $ENDFOR
```

For program flow, the block number has no significance. Therefore, it need not be used in ascending form.

8 Subroutine techniques

Identical motion sequences and function flows that are repeated several times can be implemented as subroutines. There are two types of subroutines:

1. Local subroutines
2. Global subroutines or cycles

Several subroutines (local or global) nested in one another can be called. The number of subroutines used and the nesting depth are fixed [6] [▶ 786]-24.

Variables are transferred from the main program to the subroutine indirectly using P parameters or directly in cycles by calling specific transfer parameters (@P...).

Overview of program definition and program call:

Main program

Definition:	%PROG_NAME or % PROG_NAME	The main program definition is only required when there are local subroutines.
Call:	From the operating console by specifying the complete file name (including file extension)	

Local subroutine

Definition:	%L UP_NAME	Separator between "L" and program definition, otherwise it is assumed that a main program is defined
Call in NC program:	LL UP_NAME	Separator between "LL" and program call, otherwise it is assumed that a global subroutine is called

Global subroutine

Definition:	%PROG_NAME or % PROG_NAME	A subroutine definition is only required if there are local subroutines
Call in NC program:	L FILE_NAME	Specify the complete file name (including file extension). Separator between "L" and program call

---

ℹ If the first character that is neither a separator nor a "%" is found in the file outside the comments, this character is evaluated as the first character of an unnamed main program. It also means that no block numbers may be programmed in front of "%".

8.1 Local subroutines (Call LL <string>) <string>)

A local subroutines is called by

LL <string> (Caution: Blank character between "LL" and <string> is mandatory).

<string>   Name of local subroutine without quotation marks, maximum 83 characters

Local subroutines (LSUB) together with the main program are located in a common data file which must state all the subroutines before the actual main program.

It should be noted that local subroutines can only be called from the main program in the same data file.

Local subroutines begin with "%L" and a program designation. Between "L" and the program designation, there must be a minimum of one separator.

The end of the subroutine is marked by the functions M17 or M29. A program abort is also possible with M02 or M30. A warning is then issued.

If M17 or M29 is missing, the subroutine is ended by "%" (first character of the following main program of a further subroutine).

Local subroutines (Call LL <string>)

```
Structure of a data file consisting of the NC main program and local subroutines:
%L UP1          (1st local subroutine)
N1 .....
N2 .....
N9 M17          (M17 can also be left out)


%L UP2          (2nd local subroutine)
N11 .....
N12 .....
N19 M29          (M29 can also be left out)


%MAIN           (main program)
N100 .....
N105 .....
N200 LL UP1     (Call 1st LSUB)
.
N250 LL UP2     (Call 2nd LSUB)
.
N300 M30        (Main program end)
```

No NC commands are permitted before the definition of a program. This is an exception to the rule that the sequence in the NC block has no significance for program execution (see section <u>NC-Satzaufbau [▶ 25]</u>).

8.2 Global sub-routines (Call L <string>)

A global subroutine is called by

L *<string>*

<string>	File name of the global subroutine without quotation marks, max 83 characters (including file extension and possibly an absolute or relative path name)

Global subroutines (GSUB) are independent program units in a separate file. A global subroutine is called by its complete file name (including file extension). A global subroutine can also consist of local subroutines and a main program part.

It is only necessary to specify the name of the global subroutine by % in the file to indicate the start of the main program part after defining the local subroutines. It can be skipped if the file contains no local subroutines.

The calling main program is also stored as an independent program unit in another file. Global subroutines can be called form all main programs.

The end of a global subroutine is marked by the functions M17 or M29. A program abort is also possible with M02 or M30. A warning is then issued.

Global subroutines (Call L <string>)

```
;all global subroutines
%MAIN (main program)
N100 .....
N105 .....
N200 L gup_1.nc  ;Direct call of a global subroutine by
                 ;file name if program path is configured
N250 L D:\prog\ini_1.nc  ;Direct call of a global subroutine by
                         ;specifying absolute program path
N300 M30
```

8.3 Parametric subroutine call (LL / L V.E. or macro)

Instead of using fixed names, local and global subroutines can be also be called by external variables or macros. This permits a parametrisable flow of the NC program.

External variables of be of the string or string array type (see also section External variables (V.E.) [▶ 581]). The maximum string length of local subroutines is 83 characters. It is also 83 characters for global subroutines but this includes an absolute or relative path name.

Macros must be defined by L or LL before they are used. The macro content has a maximum string length of 80 characters.

A **local** subroutine is called from the main program by LL :

LL V.E. ... (Caution: Blank character between LL and V.E. …. is mandatory).
or
LL "<string> "(Caution: Blank character between LL and macro name is mandatory).

V.E. ...	Name of local subroutine parameterised by external variable
"*<macro_name>*"	Name of local subroutine parameterisable by macro. If the macro is not defined, the <macro name> is treated as a normal local subroutine name.

A **global** subroutine is called by L :

LV.E. … or **L V.E**. …
or
L"<macro_name>" or **L** "<macro_name>"

V.E. ...	Name of the file in which this global subroutine is stored when the file is parametrised by an external variable.
"*<macro_name>*"	Name of the file in which this global subroutine is stored when the file is parametrised by an external variable. If the macro is not defined, the <macro name> is treated as a normal global subroutine name.

Parametric subroutine call (LL / L V.E. or macro)

Call of subroutines by "string"-type external variables

```
;local subroutine
%L TASCHE
N10 .....
.
N99 M17

;Main program
%MAIN
N100 .....
N105 .....
;Call of local subroutine by ext. variable V.E.LUP,
;which contains the string TASCHE
N110 LL V.E.LUP
.

;Call of global subroutine by ext. variable V.E.GUP,
;which contains the string of a file name
N200 L V.E.GUP


N300 M30
```

Parametric subroutine call (LL / L V.E. or macro)

Call of subroutines by macros

```
;local subroutine 1
%L TASCHE_1
N10 .....
.
N99 M17

;local subroutine 2
%L TASCHE_2
N10 .....
.
N99 M17

;local subroutine 3
%L TASCHE_3
N10 .....
.
N99 M17

;Main program
%MAIN
;Macro definitions
N10 "LUP_1" = "TASCHE_1"
N20 "LUP_2" = "TASCHE_2"
N30 "LUP_3" = "TASCHE_3"

N40 "GUP_1" = "gup_1.nc"
N50 "GUP_2" = "D:\prog\ini_1.nc"

N100 .....
;Call of local subroutines by macros,
;which contain the strings TASCHE_1, TASCHE_2,
;TASCHE_3
N110 LL "LUP_1"
N120 LL "LUP_2"
N130 LL "LUP_3"

N200 .....
;Call of global subroutines by macros
;which contain file name strings
N210 L "GUP_1"
N220 L "GUP_2"

N300 M30
```

8.4 Implicit global subroutine call at program start

The same recurring initialisations of data must often be executed at the start of different NC main programs. If these initialisations are summarised in a global subroutine, this program can be executed implicitly as the first action at every NC program start with the channel parameter P-CHAN-00119.

The full scope of NC programming functions can be used in this global subroutine. This means that it behaves in the same way as a subroutines call with the L word in the main NC program.

> **i** This mechanism is primarily intended for initialisation, definition and presetting of offsets, G functions, parameter values, variables etc. Concrete machining processes should not be programmed here.

8.5 Implicit global subroutine call at program end

The same recurring actions (e.g. deselecting transformations, offsets, functions etc.) must often be executed at the end (M02, M30) of different NC main programs. If these initialisations are summarised in a global subroutine, this program can be executed implicitly with the channel parameter P-CHAN-00252 before M02 or M30. **before** M02 or M30.

The full scope of NC programming functions can be used in this global subroutine. This means that it behaves in the same way as a subroutines call with the L word in the main NC program.

> **i** This mechanism is primarily intended for initialisation, definition and presetting of offsets, G functions, parameter values, variables etc. Specific machining processes should not be programmed here.

8.6 Cycles as global or local subroutines (Call L | LL CYCLE)

Cycles are available in the NC kernel in the form of global or local subroutines and permit special machining operations such as deep hole drilling or pocket milling. The machining task defined in the cycle is described in general form. When the cycle is invoked, the data is supplied when the transfer parameters are assigned.

A cycle is programmed independently of the currently valid plane (G17, G18, G19) and independently of the axis names configured in the NC channel. Only the direction from which machining should be executed in the current plane must be specified at cycle call. In the cycle, access can be made to an encapsulated group of parameters of its own. They are assigned values at cycle call.

A special syntax characterised by the "@" character is available for this purpose. This character is used in cycle programming in combination with:

@Pxx	Transfer parameter in the cycle call and the cycle
@X, @Y, @Z	Main axes in the cycle
@I, @J, @K	Centre point coordinates in the cycle
@S	Main spindle in the cycle

The **cycle call** must be programmed in its own NC block without any further NC commands. The syntax consists of a global or local subroutine call with additional specification of cycle-dependent transfer parameters.

```
Syntax:
```
L | LL CYCLE [NAME=<*cycle*> [**MODAL_MOVE** / **MODAL_BLOCK**] **@P1**=.. **@P200**=.. { \ } **]**

NAME=<*cycle*>	Name of cycle (file name)
MODAL_MOVE (MODAL, old syntax)	Modal cycle call. The cycle is again executed implicitly after every further NC block in the main program containing **motion commands**.
MODAL_BLOCK	Modal cycle call. The cycle is again executed implicitly after every further NC block in the active NC program. With the following NC commands, the implicit block-modal cycle call is suppressed: • Blank lines, comment lines • Subroutine calls (L, LL, M6, G8xx). • $ commands ($GOTO, $IF, $FOR etc...) • Program end M functions (M2, M30, M17, M29) The modal effect of MODAL_MOVE / _BLOCK is deselected by the NC command #DISABLE MODAL CYCLE.
@Pxx=..	List of transfer parameters. A maximum of 200 @Pxx parameters of type REAL or STRING (as of V3.3079.25) can be transferred. Write and read accesses are only allowed within a cycle. The @Pxx parameters can be assigned direct values, any variables, P parameters and mathematical expressions.
\	Separator ("Backslash") for programming the command over multiple lines

Transfer parameters - @P parameter

- Inside the brackets, **no** order is required to specify the key words and supply parameters. When programming, users only need to know which @P parameter must be assigned for the cycle.

- @P parameters that are not needed can be omitted.

- With read access to a @P parameter that was not transferred to the cycle, it is implicitly created (default) and initialised with 0 (zero). This then increases the memory requirement. The user can switch off this behaviour using P-CHAN-00463 as of CNC Build V3.1.3079.20. When a read access is executed to @P parameters that are not transferred, the error ID 20394 is output.
  In CNC Builds up to V3.1.3079.19, non-programmed @P parameters are initialised with 0 (zero) on read access.

- The variable V.G.@P[i].VALID determines whether a @P parameter is used or is valid in the cycle.

- The functions IS_STRING and IS_NUMBER can be used to check whether @P parameters in the cycle are a string or a number, see Example 4 [▶ 209]. (as of V3.3079.25)

- In CNC Builds up to V3.1.3079.19, the transfer parameters are retained until the programmed call of another cycle (L CYCLE.. or G80.. [▶ 118]). As of CNC Build V3.1.3079.20, the transfer parameters are deleted when the cycle is closed (M17 or M29).

- @P parameters that are programmed in the cycle call but are not used in the cycle itself are ignored.

- The variable V.G.CYCLE_ACTIVE [▶ 553] determines whether the current subroutine or the current program level is a cycle.

NOTICE
A cycle is a self-contained NC program unit with a defined machining task. It is advisable to avoid nested calls of cycles because there is a danger of assigning transfer parameters several times.

ℹ Depending on the version, the processing lines of the cycle are masked or visible in the default setting in the running NC program or in single-block mode during the execution of a cycle in the display. When the display is off, only the cycle call is displayed during this time.

This feature is switchable by the channel parameter P-CHAN-00211.

ℹ **In TwinCat systems, all cycle lines are visible in the display by default.**

Definitions required before cycle call:

- Modal G functions, circle geometry data and the currently active feedrate (F word) active before cycle call are retained beyond the cycle. This feature is switchable by the channel parameter P-CHAN-00210.

- Modal G functions that are programmed in combination with axis names (e.g. G92, G98, G99, G100, G112, G130, etc.) are not restored if they were programmed in the cycle itself.

- The machining plane (G17, G18, G19) should be defined in the higher-level NC program before cycle call. The axis perpendicular to this plane in drilling cycles is the axis in which the drilling operation is executed and in milling cycles it is the feed axis for depth.

- Any tool geometry compensation (e.g. length compensation) must also be selected before the cycle is invoked.

- The values required for feedrate, spindle speed and spindle rotation direction must be defined in the higher-level NC program, unless there are corresponding transfer parameters in the cycle.

- Spindle commands programmed in the cycles always refer to the active main spindle of the NC channel. Make sure that this main spindle is defined before cycle call.

- The start position for a corresponding drilling or milling operation and the tool's orientation must always be approached before the cycle is invoked in the higher-level NC program.

Deselecting a modal cycle

A modal acting cycle (keyword MODAL_MOVE or MODAL_BLOCK in the cycle call) is deselected with the following NC command. The command must be programmed on its own in the NC block.

`Syntax:`

#DISABLE MODAL CYCLE

Available cycles:

The following cycles are available:

- Machining cycles
- Calibration and measurement cycles
- and Cycles for kinematic optimisation

Available cycles

The example below of a cycle call for drilling (drill.cyc) presents various parameter assignment parameter.

The drill.cyc drilling cycle requires the following transfer parameters:

@P1	Position of the retraction plane (absolute)
@P2	Position of the machining plane (absolute)
@P3	Safety distance (unsigned)
@P4	Final drilling depth (absolute) or
@P5	Final drilling depth relative to the machining plane (unsigned)

Cycle call with constant values:

```
..
Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P4=40]
..
or by specifying a relative drilling depth @P5:
..
Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P5=60]
..
```

Cycle call with variables:

```
Variables must be defined and assigned values before cycle call.
..
#VAR
V.L.RPL = 110
V.L.WPL = 100
V.L.SDST = 4
V.L.DEP = 50
#ENDVAR
Nxx L CYCLE [NAME=drilling.cyc @P1= V.L.RPL @P2=V.L.WPL @P3=V.L.SDST @P4=V.P.DEP]
..
```

Cycle call with P parameters:

```
The parameters must be defined and assigned values before cycle call.
..
Nxx P10 = 110
Nxx P11 = 100
Nxx P15 = 4
Nxx P17 = 50

Nxx L CYCLE [NAME=drilling.cyc @P1= P10 @P2=P11 @P3=P15 @P4=P17]
..
```

Cycle call with mathematical expressions:

```
..
Nxx P20 = 100

Nxx L CYCLE [NAME=drilling.cyc @P1= 10+P20 @P2=2*50 @P3=5-1 @P4=P20/2]
..
```

Cycle call with constant; any sequence of parameters in brackets:

```
..
Nxx L CYCLE [@P4=40 NAME=drilling.cyc @P2=100 @P3=4 @P1=110]
..
```

Cycle call with constant values; cycle should have a modal effect:

```
..
Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P4=40 MODAL_MOVE]
..
[Example:]
%drill_main
N05 T1 D1
N10 M06
N15 G53 G17 G90 M3 S300 F200 S300
N16 G0 X0 Y0 Z0
N20 Z110
N30 X40 Y40 (drill position 1)
N40 L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=2 @P4=55 MODAL_MOVE]
N50 X60 Y60 (drill position 2 and implicit cycle call because it is modal)
N60 X100 Y60 (drill position 3 and implicit cycle call because it is modal)
N70 X100 Y20 (drill position 4 and implicit cycle call because it is modal)
#DISABLE MODAL CYCLE
N80 X0 Y0 M5
N100 M30
```

	Notes on creating cycles
i	As far as possible, cycles must be programmed as generally valid and independently of the axis names currently used in the NC channel and the definition of planes. For this purpose, the cycle has the option of using plane-independent "neutral axis names " @X, @Y and @Z for the first three main axes. Meanings:

@X always the first main axis

@Y always the second main axis

@Z always the third main axis

Example 1: Axes in the cycle

```
Nxx G91 @X=@P1 @Y=@P2 @Z=@P3 F1000 G01
```

By analogy, so-called "neutral centre point coordinates" are available for programming circles. Meanings:

@I always the centre point coordinate in the first main axis

@J always the centre point coordinate in the second main axis

@K always the centre point coordinate in the second main axis

Example 2: Circle in the cycle

```
Nxx G91 G02 @X=@P1 @Y=@P2 @I=@P4 @J=@P5 F1000
```

To remain independent from the spindle name configured during spindle programming, the main spindle can always be addressed in the cycle by the neutral spindle name @S.

@S always the main spindle

Example 3: Spindle in the cycle

```
Nxx @S=1000 M3 (main spindle cw at 1000 rpm)
```

Example 4: Check @P parameter

Transferred @P parameters can be checked by the functions IS_STRING and IS_NUMBER.

```
%L cycle
( Check variables)
$IF IS_STRING[@P1] == TRUE
#MSG["Text: %s",@P1]
$ELSE
#MSG["Error no String"]
$ENDIF

$IF IS_NUMBER[@P2] == TRUE
#MSG["Number: %f",@P2]
$ELSE
#MSG["Error not a number"]
$ENDIF

M17

% Main
LL CYCLE [NAME=cycle @P1 ="String1" @P2=12.34]
M30
```

8.7   Calling block sequences (L SEQUENCE)

Block sequences are contiguous program parts (sequences) or single NC blocks in the current NC program or a global subroutine which can be executed once or several times with L SEQUENCE.

A bock sequence is defined by specifying the start and end identifications by:

- Block numbers N.. or
- Jump labels ([Stringlabel] analogous to the definition for $GOTO)

i Every call of a block sequence is identical to a subroutine call. The same rules on nesting depth apply as for global subroutines.

NOTICE

Context evaluation:

The program context in the subroutine is not set up until the first NC line of the block sequence is executed. All previous NC lines passed through are not evaluated. Previously defined variables/coordinate systems, parameters, modal statements, etc. are neither created nor initialised. Therefore, they are unknown or not available in the block sequence.

In particular when executing block sequences with control block statements ($IF-$ELSE-$ENDIF, $SWITCH,..), users must ensure themselves that they pass through the entry and return points without conflict.

```
Syntax of L SEQUENCE when using block numbers:
```
L SEQUENCE [[**NAME**=<*string*>] **N**.. [**N**..] [**REPEAT**=..] [**ENDTAG**] **]**

NAME=<*string*>	Name of the current subroutine or a global subroutine in which the block sequence makes a pass. Optional: If no name is programmed, the block sequence passes through the current NC program.
N..	Number of the first block to be executed (start number, start of block sequence)
N..	Number of the last block to be executed (return number, end of block sequence) Optionally, if both block numbers are identical, only this block is executed.
REPEAT=..	Number of repetitions of a block sequence, positive integer ≥ 1. Optionally, if REPEAT is not specified, the block sequence makes a single pass.
ENDTAG	Marks the call L SEQUENCE itself as an additional valid end of the block sequence. Optionally, if both ENDTAG and a return number N.. are programmed at the same time, the sequence end found first is the valid one.

The controller searches for the programmed N (block) numbers in the specified NC program (which can also be the same program that calls up the command). The two N numbers mark the first and last NC blocks to be executed in the block sequence; NC blocks outside this block sequence are not executed.

It is recommended to use a unique ascending numbering format for the NC blocks.

The start and return numbers can also be swapped in the command when programmed. In the NC program, however, the block sequence always passes through from the lower N number to the higher N number.

An error message is issued if the start or return number is not found.

If the block sequence is to be executed multiple times (REPEAT > 1), the program starts at the start number again at the end of the block sequence. Once all passes have been executed, the program returns from the block sequence and the rest of the program sequence is continued.

If only an N number was specified in the command, only this line is passed. This corresponds to a call with two identical N numbers.

The L SEQUENCE call itself may be located in the block sequence defined by the N numbers. When the same call is read again, the following two reactions are possible:

Without ENDTAG the recall is ignored and the block sequence is executed until the return number.

With ENDTAG the L SEQUENCE call is marked as the valid sequence end and the block sequence is terminated.

Calling block sequences with block numbers (L SEQUENCE)

Repeat block sequence between 2 block numbers once:

```
...
N20 ...                          ;Start number
...
N50 ...                          ;Return number
...
N80 L SEQUENCE [N20 N50]         ; or..

N80 L SEQUENCE [N20 N50 REPEAT=1]
...
```

Repeat block sequence between 2 block numbers several times:

```
...
N20 ...                          ;Start number
...
N50 ...                          ;Return number
...
N80 L SEQUENCE [N20 N50 REPEAT=4]
...
```

BECKHOFF

Block sequence between 2 block numbers. Bracketed sequence call is ignored when the block sequence is executed since ENDTAG is not set:

```
...
N20 ...                          ;Start number
...
N40 L SEQUENCE [N20 N80]
...
N80 ...                          ;Return number
...
```

Block sequence between 2 block numbers. Bracketed sequence call is the first block sequence end found since ENDTAG is set:

```
...
N20 ...                          ;Start number
...
N40 L SEQUENCE [N20 N80 ENDTAG]
...
N80 ...                          ;Return number
...
```

Block sequence between 2 block numbers with ENDTAG. ENDTAG is not relevant since the return number is before the sequence call:

```
...
N20 ...                          ;Start number
...
N50 ...                          ;Return number
...
N80 L SEQUENCE [N20 N50 REPEAT=4 ENDTAG]
...
```

Repeat a single NC block several times:

```
...
N20 ...                          ;Start number
...
N80 L SEQUENCE [N20 REPEAT=4]     ; or..

N80 L SEQUENCE [N20 N20 REPEAT=4]
...
```

Block sequence between 1 block number and ENDTAG:

```
...
N20 ...                          ;Start number
...
N80 L SEQUENCE [N20 ENDTAG]
...
```

Repeat block sequence between 2 block numbers several times. Sequence call is before the block sequence:

```
...
N80 L SEQUENCE [N100 N150 REPEAT=4]
...
N100 ...                          ;Start number
...
N150 ...                          ;Return number
```

Nested multiple call of block sequences between block numbers:

```
...
N40 L SEQUENCE [N60 N150 REPEAT=2] ;Sequence call 1
...
N60 ...                           ;Start number 1
...
N90 ...                           ;Start number 2
...
N120 ...                           ;Return number 2
...
N130 L SEQUENCE [N90 N120 REPEAT=4] ;Sequence call 2
...
N150 ...                           ;Return number 1
...
```

Repeat block sequence between 2 block numbers in a global subroutine several times:

```
...
N20 ...
...
N80 L SEQUENCE [NAME="glob_1.nc" N50 N150 REPEAT=4]
...
```

Nested multiple calls of block sequences in the current program and a global subroutine between block numbers:

```
...
N20 L SEQUENCE [N60 N150 REPEAT=2] ;Sequence call 1
...
N60 ...                           ;Start number 1
...
N80 L SEQUENCE [NAME="glob_1.nc" N50 N150 REPEAT=3]   ;Sequence call 2
...
N150 ...                           ;Return number 1
...
```

Alternatively, a block sequence can also be programmed by jump labels.

```
Syntax of L SEQUENCE when using jump labels (string labels):
```
L SEQUENCE [[**NAME**=*<string>*] **[***<START>***]** [**[***<END>***]**] [**REPEAT**=..] [**ENDTAG**] **]**

NAME=*<string>*	Name of the current subroutine or a global subroutine in which the block sequence makes a pass. Optional: If no name is programmed, the block sequence passes through the current NC program.
[*<START>*]	Start label of first block to be executed (start of block sequence)
[*<END>*]	End label of last block to be executed (return, end of block sequence) Optionally, if both block numbers are identical or only the start label is specified, only this block is executed.
REPEAT=..	Number of repetitions of a block sequence, positive integer $\geq$ 1. Optionally, if REPEAT is not specified, the block sequence makes a single pass.
ENDTAG	Marks the call L SEQUENCE itself as an additional valid end of the block sequence. Optionally, if both ENDTAG and an end label are programmed at the same time, the sequence end found first is the valid one.

The controller searches for the programmed jump labels in the specified NC program (which can also be the same program that calls the command). The two jump labels mark the first and last NC blocks to be executed in the block sequence; NC blocks outside this block sequence are not executed.

Jump labels are set at block start or directly after the block number. An error message is issued if the start or return label is not found.

If the block sequence is to be executed multiple times (REPEAT > 1), the program starts at the start label again when it reaches the end of the block sequence. Once all passes have been executed, the program returns from the block sequence and the rest of the program sequence is continued.

If only a single start label was specified in the command, only this line is passed. This corresponds to a call with two identical N numbers.

The L SEQUENCE call itself may be located in the block sequence defined by the jump labels. When the same call is read again, the following two reactions are possible:

Without ENDTAG the recall is ignored and the block sequence is executed up to the return label.

With ENDTAG the L SEQUENCE call is marked as the valid sequence end and the block sequence is terminated.

Calling block sequences with jump labels (L SEQUENCE)

Repeat block sequence between 2 jump labels once:

```
...
N20 [STARTLBL] ...                      ;Start label
...
N50 [ENDLBL] ...                        ;Return label
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL]] ;or..

N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=1]
...
```

Repeat block sequence between 2 jump labels several times:

```
...
N20 [STARTLBL] ...                              ;Start label
...
N50 [ENDLBL] ...                                ;Return label
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4]

...
```

Block sequence between 2 jump labels. Bracketed sequence call is ignored when the block sequence is executed since ENDTAG is not set:

```
...
N20 [STARTLBL] ...                              ;Start label
...
N40 L SEQUENCE [[STARTLBL] [ENDLBL]]
...
N80 [ENDLBL]...                                 ;Return label
...
```

Block sequence between 2 jump labels. Bracketed sequence call is the first block sequence end found since ENDTAG is set:

```
...
N20 [STARTLBL]...                               ;Start label
...
N40 L SEQUENCE [[STARTLBL] [ENDLBL] ENDTAG]
...
N80 [ENDLBL] ...                                ;Return label
...
```

Block sequence between 2 jump labels with ENDTAG. ENDTAG is not relevant since the return label is before the sequence call:

```
...
N20 [STARTLBL]...                               ;Start label
...
N50 [ENDLBL] ...                                ;Return label
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4 ENDTAG]
...
```

Repeat a single NC block several times:

```
...
N20 [STARTLBL] ...                        ;Start label
...
N80 L SEQUENCE [[STARTLBL] REPEAT=4]         ;or..

N80 L SEQUENCE [[STARTLBL] [STARTLBL] REPEAT=4]
...
```

Block sequence between 1 jump label and ENDTAG:

```
...
N20 [STARTLBL] ...                        ;Start label
...
N80 L SEQUENCE [[STARTLBL] ENDTAG]
...
```

Repeat block sequence between 2 jump labels several times. Sequence call is before the block sequence:

```
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4]
...
N100 [STARTLBL] ...                       ;Start label
...
N150 [ENDLBL] ...                         ;Return label
```

Nested multiple call of block sequences between jump labels:

```
...
N40 L SEQUENCE [[STARTLBL1] [ENDLBL1] REPEAT=2] ;Sequence call 1
...
N60 [STARTLBL1] ...                        ;Start label 1
...
N90 [STARTLBL2] ...                        ;Start label 2
...
N120 [ENDLBL2] ...                         ;Return label 2
...
N130 L SEQUENCE [[STARTLBL2] [ENDLBL2] REPEAT=4]  ;Sequence call 2
...
N150 [ENDLBL1]...                          ;Return label 1
...
```

Repeat block sequence between 2 jump labels in a global subroutine several times:

```
...
N20 ...
...
N80 L SEQUENCE [NAME="glob_1.nc" [SUP1] [EUP1] REPEAT=4]
...
```

Nested multiple calls of block sequences in the current program and a global subroutine between jump labels:

```
...
N20 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=2]      ;Sequence call 1
...
N60 [STARTLBL] ...                                 ;Start label 1
...
N80 L SEQUENCE [NAME="glob_1.nc" [SUP1] [EUP1] REPEAT=3] ;Sequence call 2
...
N150 [ENDLBL]...                                   ;Return label 1
...
```

When programming with block numbers, a block sequence is terminated by specifying the return number; when programming with jump labels a block sequence is terminated by specifying the end label or by specifying the keyword ENDTAG in the call L SEQUENCE itself.

Alternatively, a block sequence can also be terminated by specifying the NC command #SEQUENCE END.

`Syntax of default end label:`

#SEQUENCE END

---

i   **No other NC commands may be programmed in the same NC block together with #SEQUENCE END, with the exception of a block number.**

Then it is no longer necessary to specify an end label in the L SEQUENCE call. Only the start label must be set together with the additional information that this start label does not address a single NC block but the start of a block sequence. The program name, number of repetitions and ENDTAG are also optional.

`Syntax of L SEQUENCE with block number or jump label in combination with #SEQUENCE END:`

L SEQUENCE [[**NAME**=*<string>*] **N..** | **[**<*START*>**] BEGIN** [**REPEAT**=..] [**ENDTAG**] **]**

NAME=*<string>*	Name of the current subroutine or a global subroutine in which the block sequence makes a pass. Optional: If no name is programmed, the block sequence passes through the current NC program.
N..	Number of the first block to be executed (start of block sequence)
[*<START>*]	Start label of first block to be executed (start of block sequence)
BEGIN	Start number or start label addresses the start of a block sequence. #SEQUENCE END or ENDTAG mark the sequence end.
REPEAT=..	Number of repetitions of a block sequence, positive integer > 1. Optionally, if REPEAT is not specified, the block sequence makes a single pass.
ENDTAG	Marks the call L SEQUENCE itself as an additional valid end of the block sequence. Optionally, if both ENDTAG and #SEQUENCE END are specified at the same time, the sequence end found first is the valid one.

Calling block sequences (L SEQUENCE) with end label #SEQUENCE END

Execute block sequence between start number and end label once.

```
...
N20 ...                         ;Start number
...
N50 #SEQUENCE END               ;End label
...
N80 L SEQUENCE [N20 BEGIN]
...
```

Repeat block sequence between start label and end label multiple times:

```
...
N20 [STARTLBL]...               ;Start label
...
N50 #SEQUENCE END               ;End label
...
N80 L SEQUENCE [[STARTLBL] BEGIN REPEAT=4]
...
```

Execute block sequence between start label and end label once. Sequence call is before the block sequence:

```
...
N80 L SEQUENCE [[STARTLBL] BEGIN]
...
N100 [STARTLBL] ...             ;Start label
...
N150 #SEQUENCE END              ;End label
```

Repeat block sequence between start number and end label in a global subroutine multiple times:

```
...
N80 L SEQUENCE [NAME="glob_1.nc" N50 BEGIN REPEAT=4]
...
```

Block sequence between start label and end label. Bracketed sequence call is the first block sequence end found since ENDTAG is set:

```
...
N20 [STARTLBL]...                       ;Start label
...
N40 L SEQUENCE [[STARTLBL] BEGIN ENDTAG]
...
N80 #SEQUENCE END                       ;End label
...
```

9 Parameters and parameter calculation (P)

In NC programs, parameters can be used as placeholders for numerical values. The advantage of parameters is that the value of a parameter may be changed during the program flow. This allows the production of flexible NC programs.

A parameter is designated by "P" followed by a number without blank.

Parameters and parameter calculation

In a sub-routine, e.g. a drilling cycle, instead of coordinate values (drill depth, drill feed, dwell etc.) parameters are used. The parameters are then assigned the final values in each calling main program:

For the global sub-routine

%4712(drilling, face countersinking)

the following parameters are to be defined:

P10 -Reference plane=withdrawal plane
P11 -Drilling depth
P12 -Dwell period

Starting plane

Reference plane (P10)

Drilling depth (P11)

Dwell period at
drilled depth (P12)
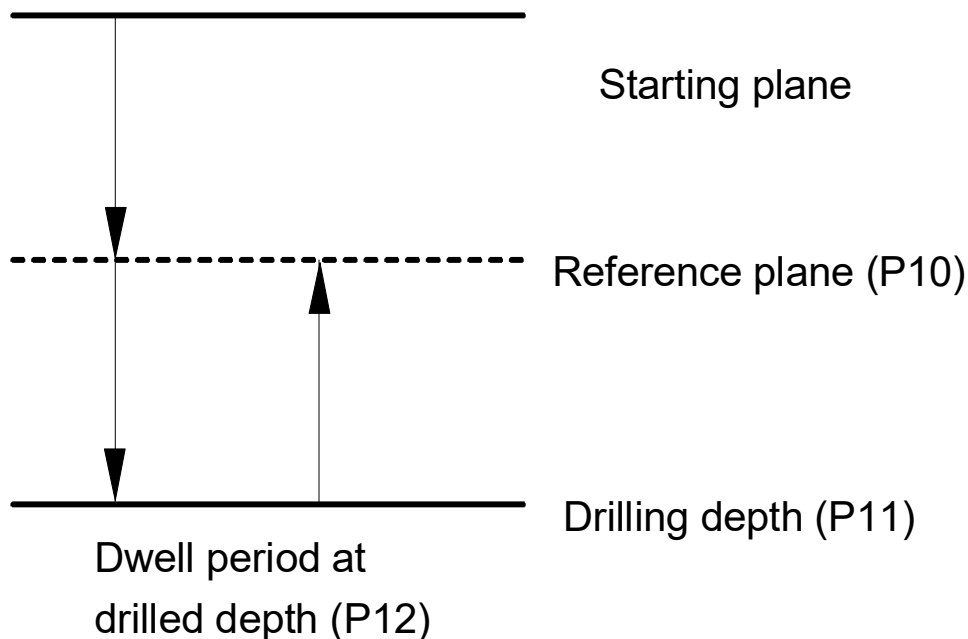
Fig. 66: Application example of parameter calculation

The call in the main program then looks like this:

```
:
N100 P10=20.5 P11=12.6 P12=1.2
N110 L4712
:
```

Syntax:

P..	Standard parameter

P..	The parameter index must always be greater than zero. However, it can assume any desired value. The maximum number of parameters used in the channel is fixed [6] [▶ 786]-6.19.

> Parameter arrays (e.g. P100[50]) are also allowed in addition to plain parameters. The dimension of the arrays is fixed [6] [▶ 786]-6.20.

```
Syntax:
```
P..[*<idx>*] {[*<idx>*]}   Parameter arrays

The channel parameter P-CHAN-00067 specifies whether the P parameters are active program global.

Parameters can be created (and initialised as required) in an NC program either within a declaration block which starts with #VAR and ends with #ENDVAR or implicitly with the first write access. However, parameter arrays must always be created in a declaration block.

For a better overview, the initialisation of a parameter array can be written over several NC blocks by using the "\" character.

```
Syntax:
```
#VAR   Start of declaration block

:

:

:

#ENDVAR   End of declaration block

#VAR and #ENDVAR

```
#VAR
P10[3][6] = [10,11,12,13,14,15, \
             20,21,22,23,24,25, \
             30,31,32,33,34,35 ]
P20[3][4] = [40,41,42,43, 50,51,52,53, 60,61,62,63]
P100
#ENDVAR

P200 = 10 P201=11
:
```

> **i** Access to parameter arrays starts at index 0. Based on the example above, access P10[0][5] gives the value 15.

Parameters and parameter arrays can also be deleted in the NC program. The #DELETE command is provided for this:

`Syntax:`
#DELETE P.. {, P..}

#DELETE P..

```
#DELETE P10, P20, P100, P200, P201
```

In addition, the SIZEOF and EXIST functions are provided (see Section <u>Arithmetical expressions <expr></u> [▶ <u>32</u>]) to define the dimension size of parameter arrays and to check the existence of parameters.

Parameters receive their values assigned by the NC program, e.g. P12=0.12. They also allow the processing of control-dependent or process-dependent values of the control system, e.g.:

- current spindle rpm,
- torque in the drives,
- values of external measuring devices
- values of heat or force sensors
- keyboard inputs via operating menu

etc.

Linked arithmetical expressions can also be used instead of the direct assignment of numerals (see Section <u>Mathematical expressions</u> [▶ <u>31</u>]).The known mathematical rules apply for inputs, e.g.:

- point-before-slash calculation,
- The parentheses rule; however, the square parentheses "[ ]" must be used.

9.1 Programming of coordinates by parameters

The syntax when programming coordinates for axis designations is:

<axis_name> **P**..

<axis_name>	Designation of the axis
P..	Assigned parameter

P.. can also be formed by a mathematical expression.

Assigning parameters to address letters

```
XP1*SIN[P2*30]
```

9.2 Indirect parameters

In arithmetical expressions and assignments, indirect parameters are used in the same way as direct parameters.

Both direct (Pnn) and indirect programming (PPnn) is performed using the P word. When indirect parameters are used, the following applies:

PPnn points to the parameters Pnn.

When a PPnn is initialised, the address of a Pnn is assigned. The use of PPP... is also possible.

Indirect parameters

If P120=10, the value 10 is loaded to the parameter 120. However, the statement PP120=123.456 assigns this value to the parameter whose address exists in P120, i.e. P10. Accordingly, PP121=SQRT[2,0] produces the following result:

P120	10		P10	123.456
P121	13		P11	
P122			P12	
P123			P13	1.4142
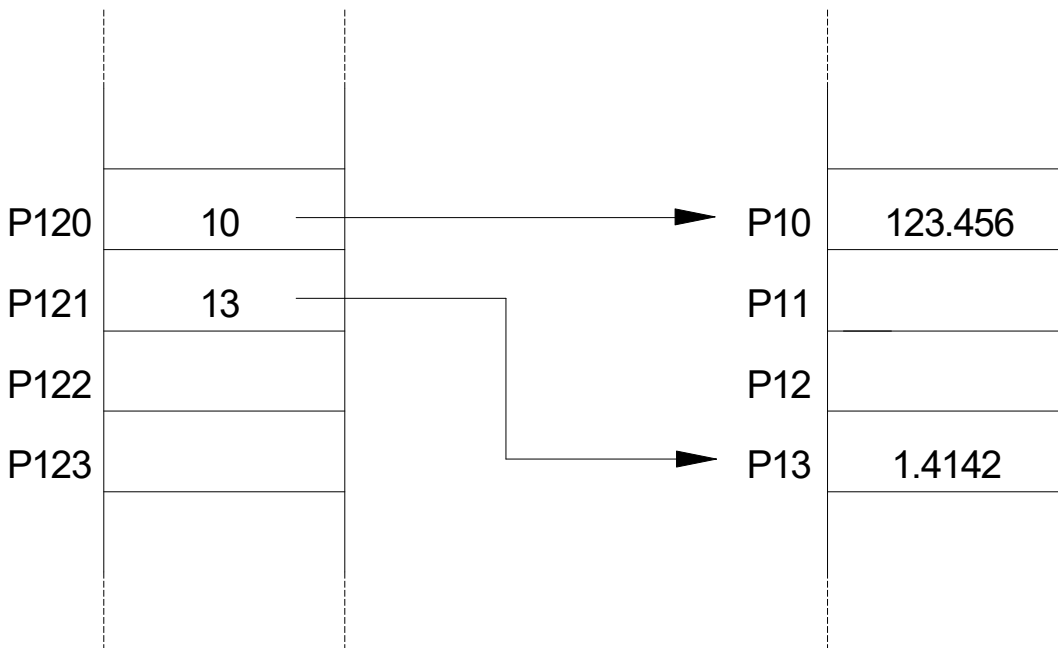
Fig. 67: Illustration of the effect of indirect P parameters

The use of indirect parameters permits the assignment of entire fields of parameters:

Indirect parameters

```
Assignment of P parameters P20 and P40 with 50
:
N110 P1 = 20  P2 = 40
N120 PP1 = 50
N130 PP2 = PP1
Assignment of P parameters P15 to P25 with 0.0
N110 $FOR P1 = 10,20,1
N120 P[P1 + 5] = 0.0
N130 $ENDFOR
:
```

10 Statements for influencing NC program flow

A complete list of G functions is contained in the overview of commands in the Appendix under <u>Control block statements ($..)</u> [▶ 793].

```
The syntax for control block statements is:
```
$<*statement*>

<*statement*>	Control block strings as described in Section <u>Conditional jumps</u> [▶ 227]. Note that between $ and <*statement*> no blank characters are permissible.

Statements for influencing program flow (control blocks) permit the implementation of:

- Conditional jumps, e.g. to trigger optional machining steps depending on a measured value
- Incremental digital loops to simplify the programming of several repetitive machining steps, e.g. for line milling or for drilling hole circles
- Loops with running condition to allow the repetition of several machining steps until the abort condition is fulfilled. For example, if the infeed of the tool and a machining operation are to be carried out until a definite coordinate value is reached. Loops may be programmed as endless loops if a running condition is missing or not fulfilled.

The following rules apply for the use of control blocks:

- Only one control block may be present in one NC block.
- Control block statements may be nested. The nesting depth is fixed.
- Only the block number and "/" may be programmed in front of the control block.
- In the invalid branch of a control block statement, a syntax check is performed for block numbers and other (nested) control block statements (see examples of IF-ELSE branching).

Syntax check in an invalid branch:

```
N10 $IF 0
N20 XY            (Here no syntax check takes place)
N30 $ENDIF
N10 $IF 0
N20 ...
N30 $IF XY        (Syntax check due to nested control block statement;)
                  (error message due to unknown term)
N40 ...
N50 $ENDIF
N60 $ENDIF
N10 $IF 0
NXY               (Syntax check of block number;)
                  (error message due to unknown term)
N30 $ENDIF
```

Due to inaccuracies in the calculation and the internal representation of parameters, comparative operations (see Section <u>Arithmetical expressions <expr></u> [▶ 32]) in control block statements may lead to an erroneous result. Therefore in cases of doubt, check for a tolerance range instead of for precise values.

```
WRONG:
N10 $FOR P1 = 0, 10, 1
N20 P2 = P2 + 0.01
N30 $ENDFOR
N40 $IF P2 == 0.1           (Due to inaccuracies of calculation P2)
N50 ...                     (may be unequal to 0.1 so that the $ELSE)
N60 $ELSE                   (branch is executed)
N70 G04 X20
N80 $ENDIF

RIGHT:
N10 $FOR P1 = 0, 10, 1
N20 P2 = P2 + 0.01
N30 $ENDFOR
```

```
N40 $IF ABS[P1 - 0.1] <= .000001 (Check a tolerance range for)
N50 G04 X5                       (unproblematic NC machining)
N60 $ELSE                        ($IF branch is)
N70 ...                          (executed)
N80 $ENDIF
```

10.1   Conditional jumps

10.1.1   The IF - ELSE branch

The following control statements are used for IF-ELSE branches:

$IF, $ELSE, $ELSEIF, $ENDIF.

```
Syntax:
```

Branching always starts with

$IF=..

and always ends with

$ENDIF

Control statements

$ELSE
and
$ELSEIF

are optional and serve to set up multiple branches.

NOTICE
The condition in the $IF control block is checked by verifying the mathematical expression for "true" or "not true" (TRUE and FALSE). To be able to also use decimal variables, the jump condition is regarded as fulfilled (TRUE) if...
...the absolute value of the mathematical expression is > or = 0.5.

BECKHOFF

The IF - ELSE branch

```
N10 ...
N20 $IF P1       Only if |P1| is greater or equal to 0.5 are the
                 statements N30 to N50 executed.
N30 ...
N40
N50
N60 $ENDIF
```

However, the following is also possible:

```
N10 ...
N20 $IF P1 >= 0.5    Only if P1 is greater or equal to 0.5 are the
                     statements N30 to N50 executed.
N30 ...
N40
N50
N60 $ENDIF
```

or:

```
N10 ...
N20 $IF P1 > P2     Only if P1 is greater than P2 are the statements
                    N30 to N50 executed, otherwise N70 to N90
N30 ...
N40 ...
N50 ...
N60 $ELSE
N70 ...
N80 ...
N90 ...
N100 $ENDIF
```

These use of ELSEIF permits:

```
N10 ...
N20 $IF P1 == 0     Only if P1 is equal to 0 are the statements N30
                    to N50 executed, otherwise a check is made in the
                    $ELSEIF condition whether P2 is >= 0.5 and
                    accordingly N70 to N90 or N110 to N130 are executed.
N30 ...
N40
N50
N60 $ELSEIF P2>=0.5  The $ELSEIF conditions is used to form
                     nested branches.
N70 ...
N80
N90
N100 $ELSE
N110 ...
N120
N130
N140 $ENDIF
```

NOTICE
The C programming language also makes a distinction in syntax between
Assignment: P5 = 3
and
Comparison: $IF P5 == 3
The following applies to Version 2.3 and earlier: As mathematical expressions expect always the sequence…
Operator -> Term -> Operator -> Term -> etc.
…expressions preceded by a minus sign must be bracketed in comparison operation ("-" is interpreted as operator).

```
$IF P1 >= -5     incorrect since term->operator->operator->term
```

Version: 1.238

```
$IF P1 >= [-5]    correct since term->operator->term->operator->term
```

```
$IF P1 >= [-5]    correct since term->operator->term->operator->term
```

10.1.2 Switch branching ($SWITCH)

SWITCH branching permits the processing of various NC program variants as a function of an arithmetic expression.

The control statements $SWITCH, $CASE, $DEFAULT, $ENDSWITCH are used for branching..

`Syntax:`

Branching always starts with:

$SWITCH *<expr1>*

followed by several

$CASE *<expr2>*

...

$BREAK

optionally followed by:

$DEFAULT

and always ends with

$ENDSWITCH

SWITCH branching

```
N100 $SWITCH P1=INT [P1*P2/P3] If the result of the arithmetical
                               expression is equal to 1, the
N110 $CASE 1                   blocks after $CASE 1 are executed (N120-140)
N120 ...
N130
N140 $BREAK
N150 $CASE P2                  If the result is equal to P2, the blocks
                               N160 ..N170 are executed.
N160 ...
N170 $BREAK
N300 $CASE n
N320 ...
N330 $BREAK
N350 $DEFAULT                  The $DEFAULT block is optional and is used to
                               execute the NC blocks N360-N380
N360 ...                       if the result of the $SWITCH block
N370                           does not match any of the $CASE cases.
N380
N390 $ENDSWITCH
```

> **ℹ** The expressions <expr1> and <expr2> are compared using the internal REAL mode. Here, both expressions are evaluated as equal if the value difference is < 0.001.
>
> The expressions <expr1> and <expr2> can also assume negative values.

10.1.3 The $GOTO statement

In addition to the subroutine technique or the use of control block statements ($IF, $FOR...), this functionality offers another option for branching to other program parts. The GOTO command can be called at any point in the NC program by setting jump labels in the NC program.

`Syntax:`

There are two options to use jump statements:

Expression label:

N<*block_no*> **:**   Definition jump target, block number with colon

$GOTO N<*block_no*>   Jump call

String label:

[<*name*>**]**   Definition jump target, name in square brackets
$GOTO [<*name*>**]**   Jump call

Characteristics:

- The $GOTO call can be placed in the NC program before or after the label definition. A label search is carried out in the program in the upwards and downwards directions.

- The label must always be called and defined at the same program level (locally within the program). Program jumps between the main program and a subroutine as well as jumps between subroutines are not permitted (see figure below).

- Identical labels may be defined in main programs and subroutines.

- It is possible to jump to the same label from several points in the NC program.

- A $IF statement may be combined with a $GOTO in the same NC line. In this case, no associated $ELSE/$ENDIF statement may be programmed.

- Other NC commands may be programmed before and after a $GOTO command in the same NC line. However, the jump is the last action in the NC line.

- External jumps are possible to any levels of a $IF-$ELSE-$ENDIF control block and within and between these levels. Then however, this jump-in level is the active level (condition is assumed as true; see Programming example).

- Jumps within $WHILE, $FOR, $DO, $REPEAT are not permitted.

- Complete exit from any control block statement by a $GOTO from any level is always permitted.

- Labels in comments (#COMMENT BEGIN, #COMMENT END) are not recognised.

- In string labels, no distinction is made between uppercase and lowercase.

- All labels ignored during decoding are stored. The maximum number of storable expression labels [6] [▶ 786]-6.41, string label [6] [▶ 786]-6.42 and string label length [6] [▶ 786]-6.43 are specified.

- At each jump call, a check is made whether the jump label already is known, i.e. stored. If the check is positive, the jump is executed immediately. If the jump label is unknown, the search is started in the active program level from the current NC line in the program level through to program end (M29/M30). If the jump label is not found, error message P-ERR-20840 is output.

- After reaching the maximum number of storable labels and other new jump labels are decoded, these labels are no longer saved. This is displayed by the warnings P-ERR-20829 or P-ERR-20831. At every new jump label call with an unknown label, the search starts again at the start of the current program level. In this case, the jump process may require more time with very large NC programs.

The $GOTO statement

```
%goto
N05  P1=1
N06  P2=1
N10  G74 X1 Y2 Z3
N11     X0 Y0 Z0

N15  $IF P1==1 $GOTO N40: ;   Jump from outside to N40 in a
                          ;    control block
N20  X10
N25  Y10
N30  $IF P1==2
N35    X20
N40:   $IF P2==1
N45    X30
N50:     Y30 $GOTO N65: ;   Jump to N65 between control block levels
                        ;    IF-ELSE
N51    $ENDIF
N55  $ELSE
N60    Y40
N65:   X40
N70  $ENDIF
N80    Z99
```

```
N999   M30
```

P R O G R A M   L E V E L S

Main	UP 1(local)	UP 2 (local)	GUP 1(global)	UP 3 (local)

LUP1

LUP3

M29

LUP2

M29

M29

M30

M29

Examples for:

possible jumps
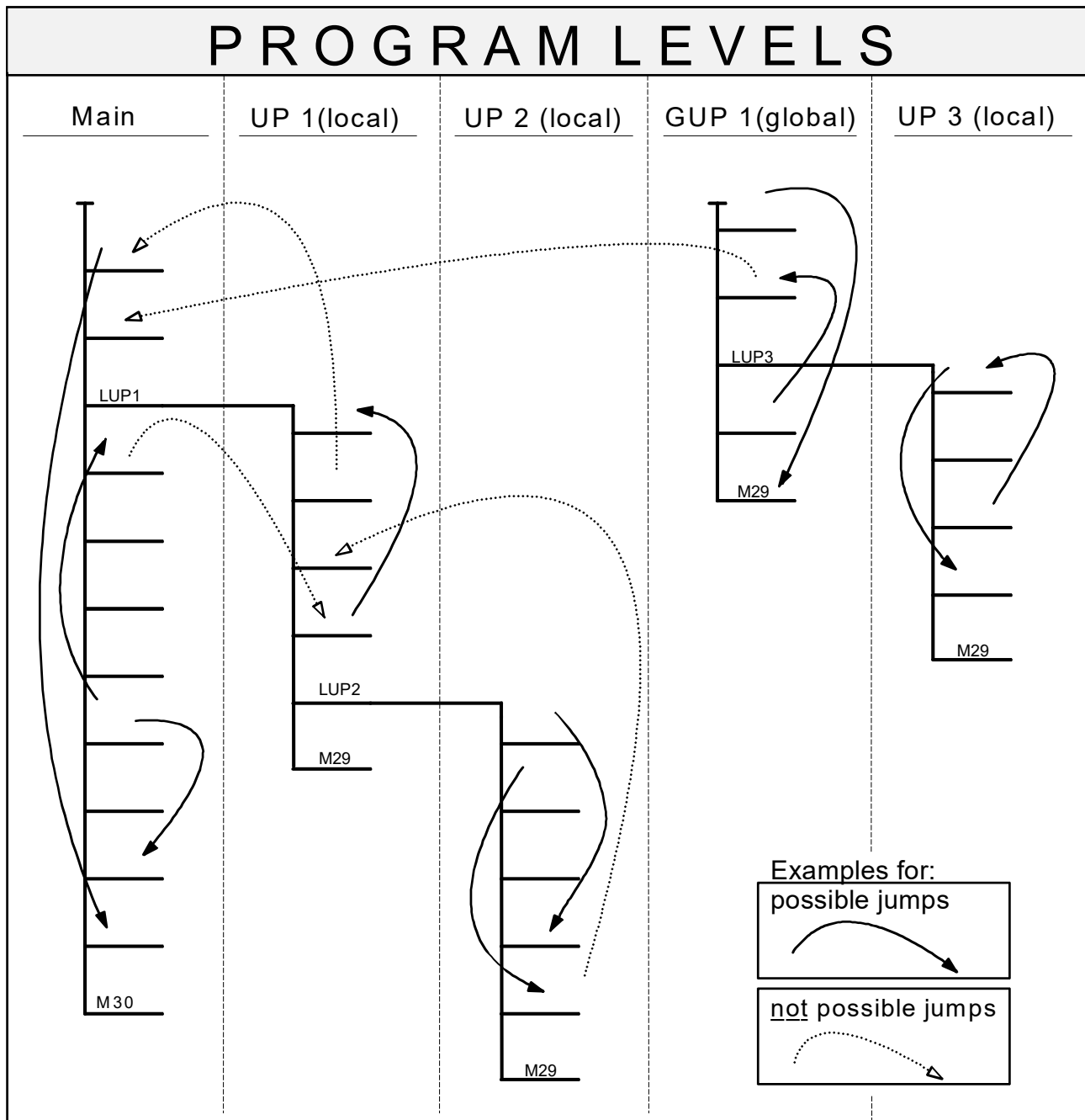
not possible jumps

Fig. 68: Permitted and impermissible jumps in the $GOTO command

```
N10 G1 XY
N20: X100                              ;Label definition N20:
$IF V.L.dummy_1 <100 $GOTO N20         ;Jump to label N20
$IF V.L.dummy_1 >200
$GOTO [LABEL_1]                        ;Jump to label [LABEL_1]
Y20
$ENDIF
[LABEL_1] X0                           ;Label definition [LABEL_1]
N30 A0
$FOR V.P.my-var = 0, 4, 1
$IF V.L.dummy_2 <200 $GOTO [CONTINUE]   ;Jump to label [CONTINUE]
$SWITCH V.P.my-var
$CASE 0
V.P.AXE-X=V.P.GROUP[1].position[V.P.my-var]
$BREAK
$CASE 1
V.P.AXE-Y=V.P.GROUP[1].position[V.P.my-var]
$BREAK
$CASE 2
V.P.AXE-Z=V.P.GROUP[1].position[V.P.my-var]
$BREAK
$CASE 3
V.P.AXE-A=V.P.GROUP[1].position[V.P.my-var]
$DEFAULT
$ENDSWITCH
$ENDFOR
[CONTINUE]                             ;Label definition [CONTINUE]
N1000 ...
...
```

10.1.3.1 Parametric jump call

The $GOTO command can also program the jump label destinations in parametric form. This permits the external control of an NC program flow (e.g. from the PLC).

In the case of a jump call for expression labels, all the mathematical expressions provided *<expr>* in the syntax scope to display block numbers are permissible, e.g. parameters, local and global variable and external variables..

```
Syntax:
```

$GOTO N..   Jump call

The jump call of string labels can be parameterised by external variables of the string or string array type (see also section External variables (V.E.) [▶ 581]). The name of the jump label is then stored in the external variable.

$GOTO V.E. …   Jump call

Parametric jump call

```
N10 ...
:
N50 $GOTO NV.E.JUMP_EXPR   ;Jump e.g. to N200 via ext. variable
                          ;V.E.JUMP_EXPR containing the value 200
:
:
N100 $GOTO V.E.JUMP_STR    ;Jump e.g. to [CONTINUE) via ext. variable
                          ;V.E.JUMP_STR containing CONTINUE string
:
:
N200:...
:
N500:…
:
:
:
[CONTINUE]…
:
N…
```

10.2   Counting loop ($FOR)

Counting loops permit the processing of statements n times. The number of loop passages is checked by a counting variable.

```
Syntax:
```

The syntax of a counting loop starts with:

$FOR P..**=** *<expr1>* **,** *<expr2>* **,** *<expr3>*

and always ends with:

$ENDFOR

Here, P.. is the counting variable. Its start value is specified by <expr1>, its end value by <expr2> and the counting increment by <expr3>.

> **ℹ** **Only integer values may be used as counting variables.**
>
> If decimal numbers are used, it is not possible to precisely represent the increment exactly (exception: powers of two) since a rounding error accumulates when added. This may lead to a loop that passes through one loop too few.

Instead of the P parameters, it is also possible to use variables ("V.") with write access.

If the counting increment is negative, the loop is aborted if the end value is undershot; if the counting increment is positive, the loop is aborted if the end value is exceeded. Programming the counting increment 0 leads to an endless loop and to the output of a warning.

Counting loops

```
N100 $FOR P1= 10, 100, 2  P1 is pre-assigned the value 10 at loop start.
                          The counting loop is passed until P1
                          exceeds the value 100; then P1 is
                          incremented by 2 at the end of every loop
                          pass.
N110 X SIN [P1 * 5]       Within the counting loop, the NC blocks N110
                          to N130 are executed.
N120 Y COS [P1 * 5]
N130 ...
N150 $ENDFOR
```

```
Negative step width:
N100 $FOR P1= 100, 10, -2  P1 is pre-assigned the value 100 at loop start.
                              The counting loop is passed until
N110 X SIN [P1 * 5]    P1 undershoots the value 10; then P1 is
                              decremented by 2 at the end of every loop
                              pass. In the counting loopsNC blocks N110
                               to N130 are executed..
N120 Y COS [P1 * 5]
N130 ...
N150 $ENDFOR


Loops not executed:
N100 $FOR P1= 100, 10, 1  P1 is pre-assigned the value 100 at loop start.
                              The counting loop is passed until
                              P1 exceeds the value 10.
N110 X SIN [P1 * 5]       But here no loop since P1 is pre-assigned the value 100.
N120 Y COS [P1 * 5]
N130 ...
N150 $ENDFOR


Endless loop:
N100 P2=20
N110 $FOR P1= 100, 10, 0 Endless loop
N120 $IF P2 == 50
N130 $BREAK
N140 $ENDIF
N150 $ENDFOR
```

10.3 Loops with running condition

10.3.1 Verification of running condition at loop start ($WHILE)

`Syntax:`

The WHILE loop starts with:

$WHILE *<expr>*

and always ends with

$ENDWHILE

At the start of every loop pass, the stated parameters are verified. The loop is aborted if the expression <expr> assumes the value range FALSE (-0.5 < expr < 0.5).

Verification of running condition at loop start

```
N90 P1 = 100.0
N100 $WHILE P1 > 0.5      P1 > 0.5 is checked for FALSE
N110 P1 = P1 - 1.5 YP1. The loop is passed
N120 $ENDWHILE            until P1 fulfils the abort condition.
N130 ...
```

10.3.2 Verification of running condition at loop end ($DO), ($REPEAT)

There are two kinds of loops available.

`Syntax:`

The syntax of the DO loop starts with:

$DO

and always ends with

$ENDDO *<expr>*

The stated parameters are checked at the end of every loop pass. The loop is aborted if the expression <expr> assumes the value range FALSE (expr < 0.5).

`Syntax:`

The syntax of the REPEAT loop starts with:

$REPEAT

and always ends with

$UNTIL *<expr>*

The stated parameters are checked at the end of every loop pass. The loop is aborted if the <expr> assumes the value range TRUE (expr < 0.5).

> ● As opposed to the $WHILE and $FOR loops, the $DO and $REPEAT loops are always passed at
> **i** least once.

Verification of running condition at loop end

```
N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 $DO
N40 P1=P1+1
N50 XP1
N60 $ENDDO P1 <= P2      P1 is checked for TRUE at loop end.
                         The loop is passed until P1 no longer
                         fulfils the condition.
N99 M30

N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 $REPEAT
N40 P1=P1+1
N50 XP1
N60 $UNTIL P1 > P2       P1 is checked for TRUE at loop end.
                         The loop is passed until P1 no longer
                         fulfils the condition.
N99 M30
```

10.4 Influencing loop flow sequences

10.4.1 The $BREAK statement

`Syntax:`

$BREAK

It is not always useful to exit a loop with the abort criterion. The keyword $BREAK can also abruptly terminate the execution of a loop in addition to program execution with single $CASE labels of the $SWITCH statement (see section <u>Switch branching [▶ 230]</u>).

For example, this is useful with extremely nested loops if execution of the innermost loop should be interrupted.

The $BREAK statement

```
N10 $WHILE <expr1>       The loop is terminated if
N20 ...                  expr1 is "not valid" or
N30                      expr2 is "valid".
N40 $IF <expr2>
N50 $BREAK
N60 $ENDIF
N70 ...
N80
N90 $ENDWHILE
N100 ...
```

10.4.2 The $CONTINUE statement

Syntax:

$CONTINUE

As opposed to $BREAK, the $CONTINUE statement does not abort the loop but branches it to the loop start. All statements after $CONTINUE are then not executed.

The $CONTINUE statement

```
N10 $FOR <expr1>        The statements in lines N70 and N80
N20 ...                 are only executed if
N30                     expr2 is "not valid".
N40 $IF <expr2>
N50 $CONTINUE
N60 $ENDIF
N70 ...
N80
N90 $ENDFOR
N100 ...
```

11   Smoothing methods

Introduction

A programmed curve must be rounded and smoothed within specific tolerances to allow it to move even over corners without stopping as quickly and uniformly as possible. This is referred to as smoothing and there are various methods provided.

Simple contours with few long linear and circular blocks are ideal for polynomial contouring. Select and this function with **G261** and deselect with **G260** . This method is described in Section G functions [▶ 122].

On the other hand, it is preferable to use **#HSC**methods for many short linear blocks. The methods include the highly rugged SURFACE method [▶ 247] that is particularly suited to free-form surface machining. It achieves the best results in the event of disruptions in the programmed contour and blocks which have very different lengths. By contrast, this places greater requirements on the hardware. The B spline method [▶ 244] can also be used to trim a contour.. It requires less high-performance hardware but may lead to drops in path velocity on less properly programmed contours.

If these HSC programs also contain circular blocks, the transitions can be smoothed by **#CONTOUR MODE**functions. This requires the option CIR_MODE [▶ 247] and the channel parameter P-CHAN-00239 which are described in the sections mentioned above. HSC programs may also include circular blocks. If NC programs contains many short blocks, it is advisable to use the HSC profile generator #SLOPE[TYPE=HSC] [▶ 357].

Besides smoothing a programmed contour, a frequent function is to filter axis command values symmetrically. These functions are described in Section Filter programming.

Besides these recommended standard methods, there are a number of other methods such as interpolation with the Akima spline [▶ 284], the direct programming of B spline control points [▶ 290] and older HSC functions [▶ 293].

Name of function	Its suitability	Advantages	Disadvantages
#CONTOUR MODE	For simple contours with few long blocks	Greater path velocities at contour knee angles	Not for short blocks
SURFACE methods	For complex contours with several short blocks	Very rugged	Increased hardware requirements
B spline method	Trim a contour	Slow motion sections with unfavourable programming	Not so high hardware requirements
Filter programming	To filter axis command values symmetrically		
Akima spline	Interpolate specified interpolation points	Runs precisely through the programmed points	Generally requires a denser and exactly calculated specification of points
PSC functions with OP1 and OP2	Rigid machines	Low hardware requirements	Relatively strong excitation of machine structure

11.1 Programs with several short blocks

i The use of this feature requires a license for the "HSC" extension package. It is not included in the scope of the standard license.

NOTICE

B splines for the programmed control points are generated using extended HSC programming. For this kind of HSC programming it is recommended to first select the HSC profile type (slope 3) using the command #SLOPE [TYPE...] [▶ 357].

Depending on the machining task the following 2 methods are available for selecting/deselecting HSC programming and parametrisation:

Method 1 is especially suited to a single move around a contour (trimming). In this case, the contour consists of very many short blocks which are to be moved at a high feedrate.
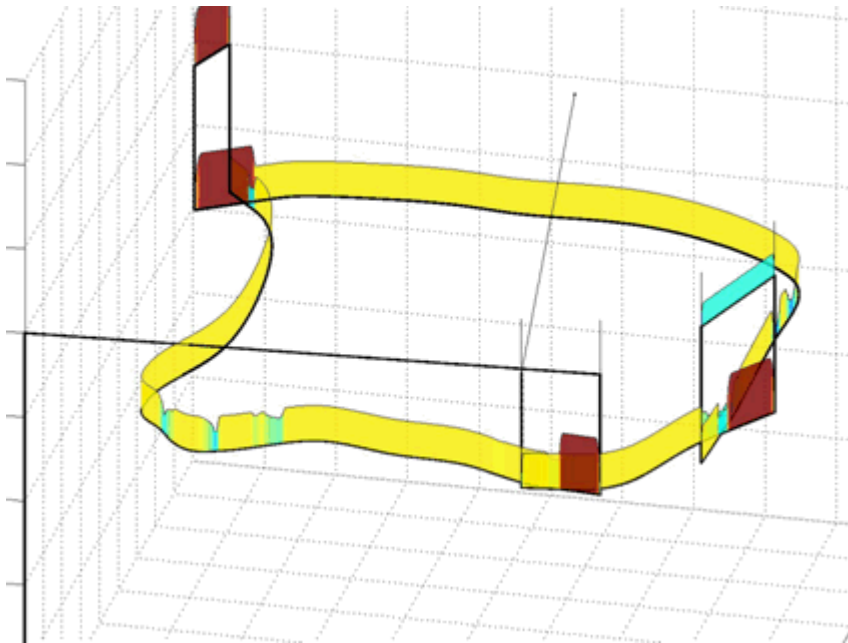
Fig. 69: Trim a contour

Method 2 is especially suited to machining free-form surfaces. For NC programs generated by CAD systems, the workpiece is usually machined in several paths (line-by-line or helical). Special algorithms are used (surface optimiser) to achieve a high surface quality within the shortest possible machining time.
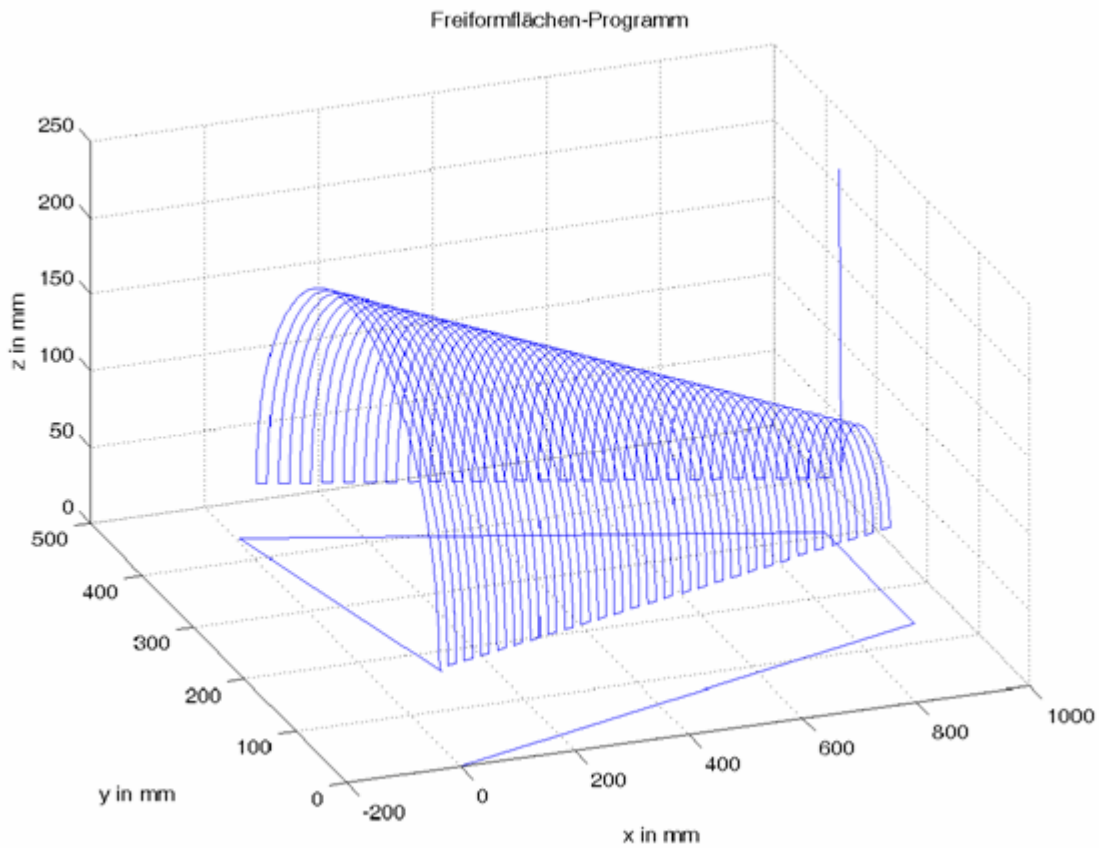
Fig. 70: Line-by-line surface machining

11.1.1 Trim a contour (#HSC ON/OFF)

`Syntax:`

#HSC [**ON** | **OFF**] [**BSPLINE** [**PATH_DEV=..**] [**TRACK_DEV=..**] [**MERGE=..**] [**AUTO_OFF_PATH=..**]
                 [**AUTO_OFF_TRACK=..**] [**AUTO_OFF_G00=..**] [**AUTO_OFF_G60=..**]
                 [**MAX_PATH_LENGTH=..**] [**MAX_ANGLE=..**] **]**]

ON	Enable HSC programming.
OFF	Disable HSC programming.
BSPLINE	Keyword for HSC programming with BSPLINE. Must always be programmed as **first** keyword.
PATH_DEV=..	Maximum deviation of B spline from programmed path contour in [mm, inch *]. The spline is deselected automatically if this deviation is exceeded. If the maximum deviation is defined as 0, path deviation is not monitored.
	Default value: 0.2 mm
	*when P-CHAN-00439 is active
TRACK_DEV=..	Maximum deviation of tracking axes in [°]. If the maximum deviation is defined as 0, tracking axes is not monitored.
	Default value: 5 °
MERGE=..	Merge blocks. The maximum deviation is determined depending on the values taken from PATH_DEV and TRACK_DEV.
	0: No block merging (default)
	1: Merge blocks
AUTO_OFF_PATH=..	Automatic block separation if the programmed B spline deviation of the main axes is exceeded (PATH_DEV).
	0: No deselection if deviation is too large (default), block is separated
	1: Deselect if deviation is too large
AUTO_OFF_TRACK=..	Automatic block separation if the programmed B spline deviation of the tracking axes is exceeded (TRACK_DEV).
	0: No deselection if deviation is too large (default), block is separated
	1: Deselect if deviation is too large
AUTO_OFF_G00=..	Automatic deselection of B spline interpolation for G00 blocks.
	0: No implicit deselection due to rapid traverse block (default)
	1: Implicit deselection due to a rapid traverse block
AUTO_OFF_G60=..	Automatic deselection of B Spline interpolation for programmed exact stop G60 or G360.
	0: No implicit deselection due to exact stop (default)
	1: Implicit deselection due to exact stop
MAX_PATH_LENGTH=..	Minimum path length of relevant blocks in [mm, inch *]. If blocks are longer than the specified length, the B Spline is deselected implicitly.
	Default value: 0 mm (implicit deselection due to block length does not take place)
	*when P-CHAN-00439 is active

MAX_ANGLE=.. Maximum contour knee angle in [°] for transitions between two linear blocks up to which a B spline is inserted. The B Spline is deselected internally if the angle between two linear blocks is greater.

Default value: 160 °
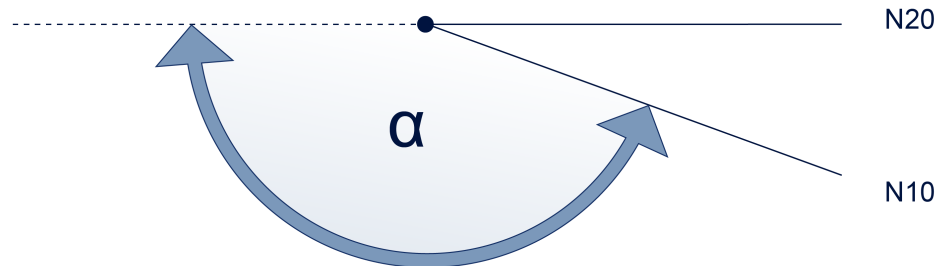
Control points are programmed with linear blocks (G00 and G01). Their target points are used as control points. It must be considered that only the start and end of the curve runs straight through the control points.

ℹ️ The parameters may also be specified in several steps. For example, this means that it is possible to first define the maximum contour deviation ("PATH_DEV "). Then in a second command, the maximum path length ("MAX_PATH_LENGTH ") and selection of B spline interpolation ("ON") are defined.

NOTICE
Parameterisation cannot be changed while B spline interpolation is active.

Trimming a contour

The spline curve is based on the control points N40 – N155 but in this case, the spline curve only runs straight through them at N20 and N150.

```
N20 G00 X0 Y0 Z0 F10000
N30 #HSC ON [BSPLINE PATH_DEV=0.2 MERGE=1 ...] Parametrisation + selection
N40   X3     Y25
N50   X15    Y15
N60   X23    Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #HSC OFF
N170 M30
```

... or also

```
N20  G00 X0 Y0 Z0 F10000
N25  #HSC [BSPLINE PATH_DEV=0.2 MERGE=1 ...] Parameterisation
N30  #HSC ON Selection
N40   X3     Y25
N50   X15    Y15
N60   X23    Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #HSC OFF Deselect
N170 M30
```

11.1.2 Surface machining with Surface Optimiser (method 3)

The HSC Surface Optimiser was developed to achieve consistent machining results regardless of the point distribution by the CAM system. In particular, the density of interpolation points on neighbouring machining paths can fluctuate with some CAM systems, which would lead to an uneven machining result. The figure below shows a machining result of this kind. The two points marked in red are missing on one of the neighbouring machining paths. This results in a different tool path (blue) in contrast to the neighbouring paths if the smoothing process is not optimised.
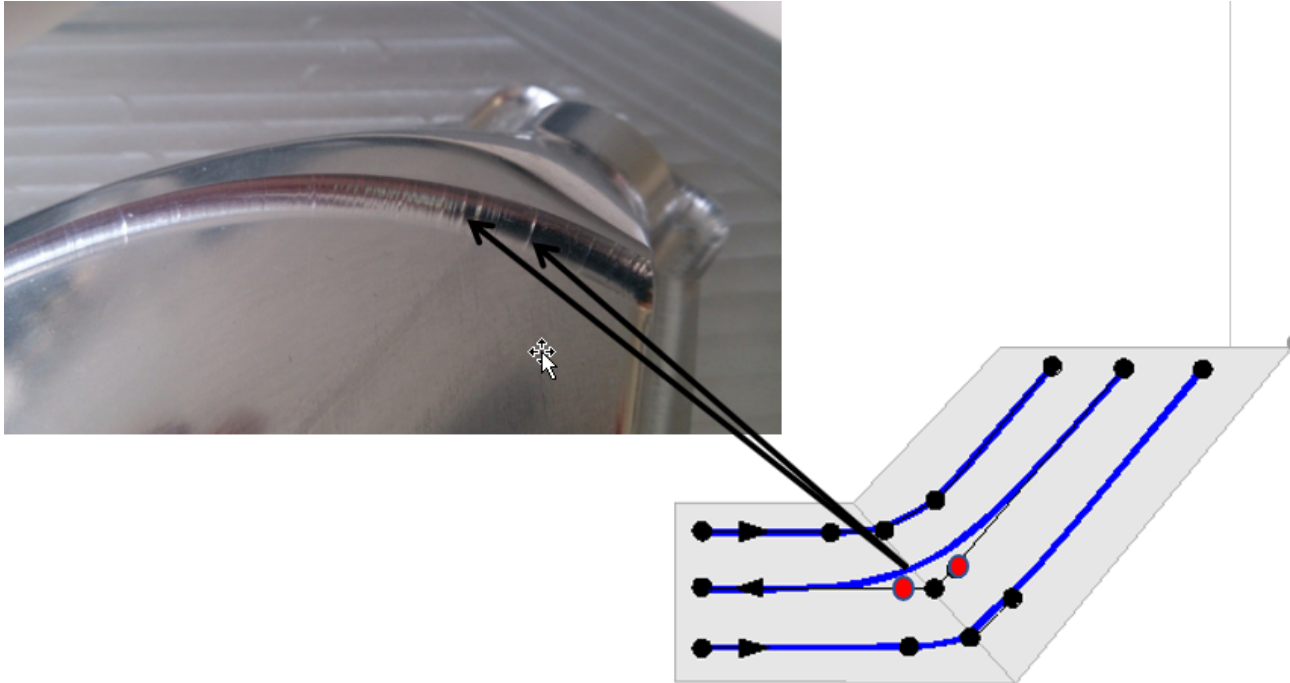
Fig. 71: Problems in workpiece quality due to uneven distribution caused by the CAM system.

In addition to uniform machining paths, the HSC Surface Optimiser ensures a high feedrate that is as constant as possible. Due to the necessary calculations, the use of the Surface Optimiser requires high-performance control hardware.

Programming

```
Syntax:
```
#HSC [**ON** | **OFF**] [[**SURFACE** [**PATH_DEV**=..] [**PATH_DEV_G00**=..] [**TRACK_DEV**=..] [**TRACK_DEV_G00**=..]

[**MAX_ANGLE**=..] [**CHECK_JERK**=..] [**AUTO_OFF_G00**=..] [**CIR_MODE**=..]

[**CIR_MIN_ANGLE**=..] [**CIR_MIN_RADIUS**=..] [**MERGE**=..] [**LENGTH_LONG_CIR**=..] **]**]

ON	Enable HSC programming.
OFF	Disable HSC programming.
SURFACE	Keyword for HSC machining with surface optimiser. Must always be programmed as **first** keyword.
PATH_DEV=..	Define maximum contour error.
	> 0.0: Maximum path deviation in [mm, inch *]
	Default value: 0.2 mm

ℹ Empirically, the contour error should be set 2 or 3 times larger than the secant error which is defined when the NC program is generated in the CAM system.

The tool is not in contact with the workpiece in G0 motions. As a result, the tolerance can be set significantly larger than PATH_DEF without changing the precision of the workpiece.

PATH_DEV_G00=..	Define maximum contour error for G0-G0 transitions.
	> 0.0: Maximum path deviation in [mm, inch *]
	Default value: The value of PATH_DEV applies
TRACK_DEV=..	Define the maximum orientation error.
	>= 0.0: Maximum path deviation in [°]
	Default value: 2 °
TRACK_DEV_G00=..	Define the maximum orientation error for G0-G0 transitions.
	>= 0.0: Maximum path deviation in [°]
	Default value: The value of TRACK_DEV applies

---

i If a ball milling cutter is used, the value can be set significantly larger than PATH_DEV (e.g. 10 times).

The tool is not in contact with the workpiece in G0 motions. As a result, the tolerance can be set significantly larger than TRACK_DEV without influencing the precision of the workpiece.

MAX_ANGLE=..	Define the maximum contour knee angle in degrees for transitions between two linear blocks up to which this method can be applied. If the angle between the two linear blocks exceeds this limit, the mode is deselected internally.
	>= 0.0: Maximum knee angle in [°]
	Default value: 160 °

CHECK_JERK=..	Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110). This parameter overwrites the initial state defined in the channel parameter list by P-CHAN-00110 (check_jerk_on_poly_path).
	0: No jerk monitoring
	1: Jerk monitoring based on the geometric ramp time P-AXIS-00199. This may reduce path velocity.
	2: Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non-linear velocity profile.
AUTO_OFF_G00=..	Automatic deselection of optimisation with G00 blocks
	0: No implicit deselection due to rapid traverse block (default)
	1: Implicit deselection due to a rapid traverse block
CIR_MODE=..	Define contouring of circular motions:
	0 : No contouring of circular motions G02/G03
	1 : Circular block contouring. (default)
	2 : Contour circular blocks and optimise long circular blocks Available as of V3.1.3075.01
CIR_MIN_ANGLE=..	Define minimum circle angle
	Valid values: >= 0.0 : Minimum circle angle in degrees

	The minimum circle angle at which circular motions can be traversed by the method using exact interpolation. Circular blocks with small swept angles are approximated by a spline curve for faster processing. (Default value = 30°)
CIR_MIN_RADIUS=..	Define minimum circle radius
	Valid values: >= 0.0 : Maximum circle radius in [mm, inch *]
	The minimum circle radius defines the circle radius from which circular motions can be traversed by the method using exact interpolation. Circular blocks with a small radius or with the magnitude of PATH_DEV are approximated by a spline curve.
	Available as of V3.1.3075.01
MERGE=..	Merge blocks. The maximum deviation is determined depending on the values taken from PATH_DEV and TRACK_DEV.
	0: No block merging (default)
	1: Merge blocks
LENGTH_LONG_CIR= ..	Minimum length of segments for long circular blocks when CIR_MODE= 2 in [mm, inch *] is used (default value= 2)
	Available as of V3.1.3075.01
	*with active P-CHAN-00439

Default values of free-form surface machining

PATH_DEV	0.2 mm (default value of PATH_DEV)
TRACK_DEV	2° (default value of TRACK_DEV)
PATH_DEV_G00	PATH_DEV
TRACK_DEV_G00	TRACK_DEV
CIR_MODE	1
MAX_ANGLE	160 °
CHECK_JERK	The valid channel parameter is P-CHAN-00110 (check_jerk_on_poly_path, default value = 1)
AUTO_OFF_G00	0
CIR_MIN_ANGLE	30 °
CIR_MIN_RADIUS	0.0
LENGTH_LONG_CIR	2 mm

> **i** The parameters may also be specified in several steps. For example, this means it is possible to first define the maximum contour deviation ("PATH_DEV "). Then in a second command, jerk monitoring ("CHECK_JERK") and the selection of HSC surface interpolation ("ON") are defined.

> **i** **When #HSC[SURFACE] is used, you are advised to use #SLOPE[TYPE=HSC] at the same time for path velocity planning.**

NOTICE

Parameterisation cannot be changed while smoothing is active.

One condition to use this function is that it should be parameterised in the start-up list for each channel in which the function is to be used.

Surface machining with Surface Optimiser

Example of a setting in the start-up list:configuration.channel[].path_preparation.function FCT_DEFAULT| FCT_SURFACE

```
N20 G00 X0 Y0 Z0 F10000
```

```
;Parametrisation + selection
N30 #HSC ON [SURFACE PATH_DEV=0.02 CHECK_JERK=0]
N40    X3      Y25
N50    15      Y15
N60    23      Y12
N70    X25     Y25
N80    X30     Y35
N90    X50     Y37.5
N100   X55     Y32.5
N110   X58     Y12
N120   X70     Y12
N130   X77.5   Y10
N140   X90     Y35
N150   X100    Y37.5
N160 #HSC OFF
N170 M30
```

Alternative programming:

```
N20 G00   X0      Y0      Z0 F10000
N25 #HSC [SURFACE PATH_DEV=0.02 CHECK_JERK=0] ;Parameterisation
N30 #HSC ON ;Select
N40    X3      Y25
N50    15      Y15
N60    23      Y12
N70    X25     Y25
N80    X30     Y35
N90    X50     Y37.5
N100   X55     Y32.5
N110   X58     Y12
N120   X70     Y12
N130   X77.5   Y10
N140   X90     Y35
N150   X100    Y37.5
N160 #HSC OFF
N170 M30
```

11.1.3 FIR filter (#FILTER)

> **ⓘ** **The availability of this function depends on the configuration and on the version scope.**

In order to achieve a high surface finish in free-form surface machining, any excitation of machine oscillations must be avoided as far as possible.

FIR axis filters (Finite Impulse Response filters) provide the user with the option of smoothing the axis setpoints for the drives to minimise excitations in the machine.

The precondition for using a FIR filter using the #FILTER command is a configured filter type (P-AXIS-00586) of the corresponding axis.

> **ⓘ** **This function is an additional option requiring a license.**

Syntax:

#FILTER [ON | OFF] [ORDER=.. ORDER_TIME=.. SHARE=.. AX_DEV=.. FCUT=.. ACC_FACT=.. QUALITY=..]

ON	Enable FIR filter.
OFF	Disable FIR filter.
ORDER=..	Specify filter order
ORDER_TIME=..	Specify filter order over time in [µs]
SHARE=..	Define the degree of effectiveness (analogous to P-AXIS-00590) of the filter in [%] value range 0 – 100 default value = 100
AX_DEV=..	Specify the tolerance for tolerance monitoring in [mm, inch *]. Default value = 0 (no tolerance monitoring). *with active P-CHAN-00439
FCUT=..	Specify the cut-off frequency (analogous to (P-AXIS-00585) of the filter in [Hz] default value = 30
ACC_FACT=..	Increase the path velocity at block transitions with FIR filter enabled. The greater the value setting, the less the velocity is reduced at the block transition. This requires a valid setting of P-AXIS-00013 (a_trans_weight) for the axes. Value range = 1.0 – 10.0 Default value =: 1.0
QUALITY=..	Specify the filter quality of the filter core curve value range: 0 < QUALITY <= 1 default value = 1.0 Parameter available as of V3.1.3075.04

> **ⓘ** **The #FILTER ON/OFF command enables or disables all the FIR filters of the axes in the channel.**

It is possible to use FIR filters on all axes. It is also possible to use different filters for each axis by axis-specific configuration in the axis lists.

FIR filters can be globally enabled or disabled and reparameterised across all axes in the NC program during machining (see Programming example).

> ℹ️ **Tolerance monitoring can only be configured and activated in the NC program.**

Tolerance monitoring is programmed by the parameter AX_DEV. It ensures that every axis remains within the specified tolerance [mm, inch].

Tolerance monitoring always monitors all axes and therefore can only be controlled globally in the NC program.

Tolerance monitoring in only active if AX_DEF was specified with a corresponding tolerance.

For further information see [FCT-C37//Description]

This command replaces the previously available #FILTER ON [HSC] command.

11.2 Polynomial contouring for long blocks (G61/G261/G260)

```
Syntax:
```

G61	Polynomial contouring (at block end)	non-modal
... or for polynomial contouring across several blocks:		
G261	Selecting polynomial contouring (at block end)	modal
G260	Deselecting polynomial contouring	modal

11.2.1 Definition of terms

The following terms are briefly explained:

Polynomial contouring:	Curvature and direction-continuous connection of two motion blocks.
Contouring curve:	Curve composed of two 4th order polynomials per axis.
Block length:	The path length of the curve corresponding to the motion block.
Corner distance:	Distance from the start/end of the contouring curve to the programmed target point/ starting point of a motion block (see figure below). The corner distance is always limited to half the block length. In a circular block, the corner distance is the arc length from the starting point of the contouring curve up to the programmed target point of the arc.
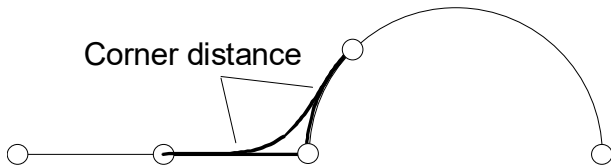
Fig. 72: Definition of corner distance

Pre-block:	Motion block before the contouring curve
Post-block:	Motion block after the contouring curve
Pre-distance:	Corner distance of the pre-block
Post-distance:	Corner distance of the post-block
Interim point:	Point at which the two partial curves of the contouring curve meet.
Corner deviation:	The distance between the programmed corner point and the interim point of the contouring curve (see figure below).
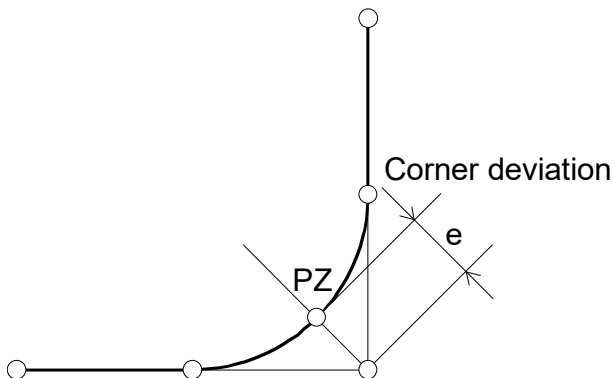
Fig. 73: Definition of corner deviation

Comparing the programming of G61 – G261/G260

The three NC programs all generate the identical contour shown in the figure below.

```
%poly_G61
N10   X0    Y0 G01 F1000
N20  X20 Y100
N30  G61   X40 Y100
N40  G61   X60 Y20
N50  G61   X80 Y20
N60  G61   X100 Y100
N70   X120 Y100
N80   X140 Y20
N90   X160 Y20
N100 M30
```

```
%poly_G261_1
N10   X0    Y0 G01 F1000
N20   X20   Y100
N30   G261 X40 Y100
N40   X60   Y20
N50   X80   Y20
N60   X100 Y100
N70   G260 X120 Y100
N80   X140 Y20
N90   X160 Y20
N100 M30
```

```
%poly_G261_2
N10 X0     Y0 G01 F1000
N20 X20   Y100
N25 G261
N30 X40   Y100
N40 X60   Y20
N50 X80   Y20
N60 X100 Y100
N70 X120 Y100
N75 G260
N80 X140 Y20
```
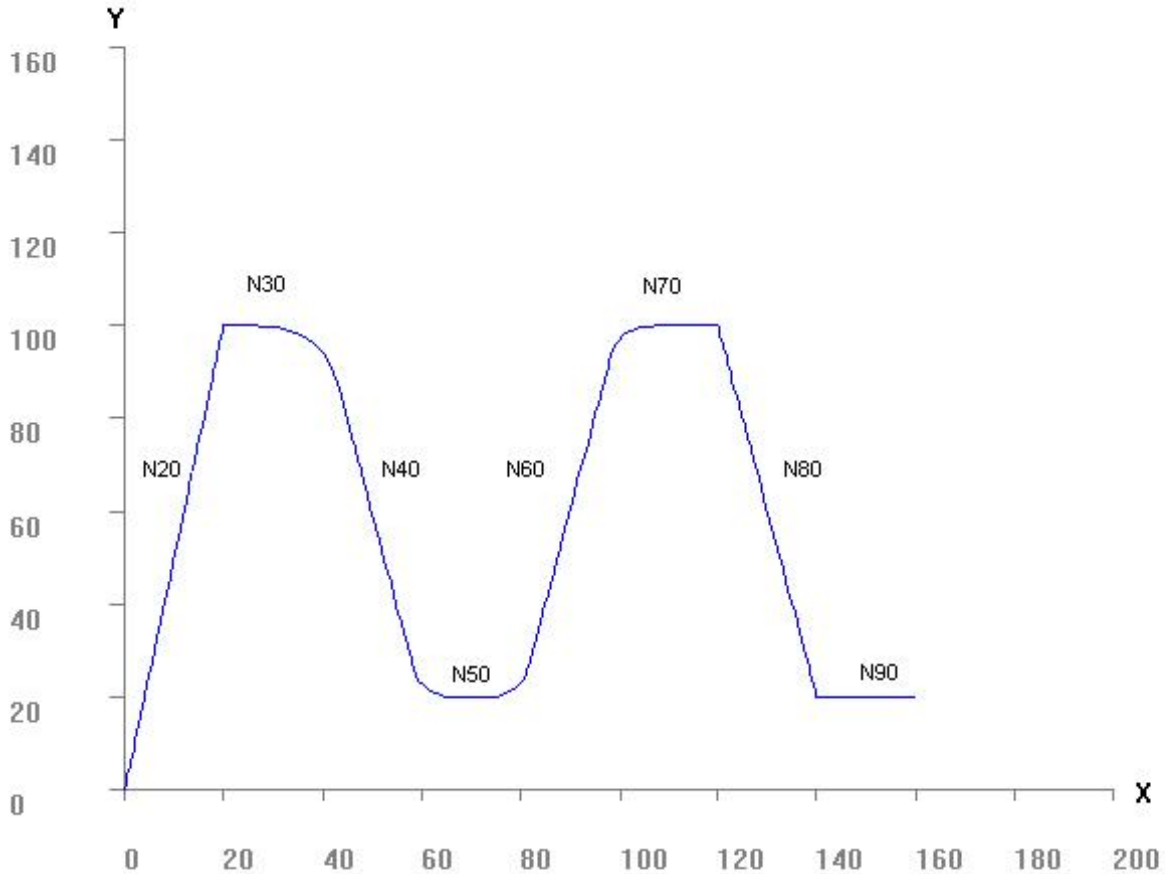

Fig. 74: Contour for programming G61 – G261/G260

11.2.2 General properties

The process of polynomial contouring is calculated from the geometrical path contour of the main axes in space. The given conditions, e.g. corner deviation or percentage path velocity, result in a position on the original contour from which the contour can be changed or replaced by a contouring curve (polynomial). This means that the starting or target point of the contouring curve which is known on the original path curve.

Using the determined starting and target points of the polynomial of the main axes calculated using the conditions, it is also possible to specify the position of the tracking axes at which their original contour can be replaced by a polynomial.

With tracking axes, as with main axes, a curvature and direction-continuous polynomial is inserted between the corner distances of the pre-block and post-block, taking into consideration the max. acceleration of these axes. However, the originally specified corner deviation refers only to the deviation of the main axis in space so that, if required, an additional limit value can be specified for the maximum deviation of the tracking axes. Any theoretical excess of this deviation by the tracking axis causes a reduction in the contouring curve (reduction in corner distance).

Polynomial contouring is automatically suppressed depending on the transition between the pre-block and post-block if:

- The transition of all axes is mirrored tangentially or directly.

- The transition of the main axes is tangential and no maximum deviation (value = 0) was specified for the tracking axes.

- After programming G61, program end is reached without post-block. In addition, a warning message is output.

11.2.2.1 Maximum corner distance, minimum residual block length

To avoid any "degeneration" of the polynomial, the following limitations apply additionally:

- The corner distance may assume a maximum of 50% of the original block length. If the corner distance selected is greater, the distance of the pre-block and post-block is limited accordingly. If the corner distance at block start and block end is 50% of the original block length, the block is skipped completely.

- When parameterising the contouring function, the minimum residual block length can be set between 0% and 100%. This corresponds to a variable maximum corner distance from 50% to 0%. At every program start, the minimal residual block length is first set to 0% (block can be completely contoured). If the minimum residual block length is specified as 10%, for example, the corner distances of this block can be maximum (100% - 10%) / 2 = 45% of the original block length.

- In circular blocks the maximum corner distance (distance travelled on circle) is limited so that the angle covered does not exceed 90°.

11.2.2.2   Relevant block length

If the programmed relevant block length, which is defined via RELEVANT_PATH, is less than the defined minimum length of 32µm, the block is limited to this minimum length.

In addition, the contour can include very short compensation blocks which are inserted by a programming system (CAD/CAM) or by tool radius compensation. After compensation the block retains a continuous path.

To avoid abortion of contouring by these short blocks, a minimum block length can be defined. From this point onwards, polynomial contouring is then relevant for the post-block. Shorter blocks are skipped during active contouring, i.e. contouring is considered in the following block.

Here, a limit for the motion path of the main axes in space as well as a limit can be specified for the motion path of the tracking axes. The block is skipped completely only when both the motion path of the main axes and the motion path of the individual tracking axes are below the specified limit. Polynomial contouring combines the pre-block and the post-block in direction and curvature-continuous function. The initial blocks need not be adjacent (contour need not be continuous).

If a block is skipped, the maximum corner deviation of the main axes and tracking axes can only be approximated. This means, it is assumed that the deviation of the contouring can be ignored in the skipped blocks.

Relevant block length

```
#CONTOUR MODE [DEV, PATH_DEV 5, RELEVANT_PATH 2]
N03 G01 X0 Y0 Z0 C0 F4
N907090 G04 X0.1
N04 X5 G261
N05 Y1
N09 X10 Y3 G260
N907091 Y0
```
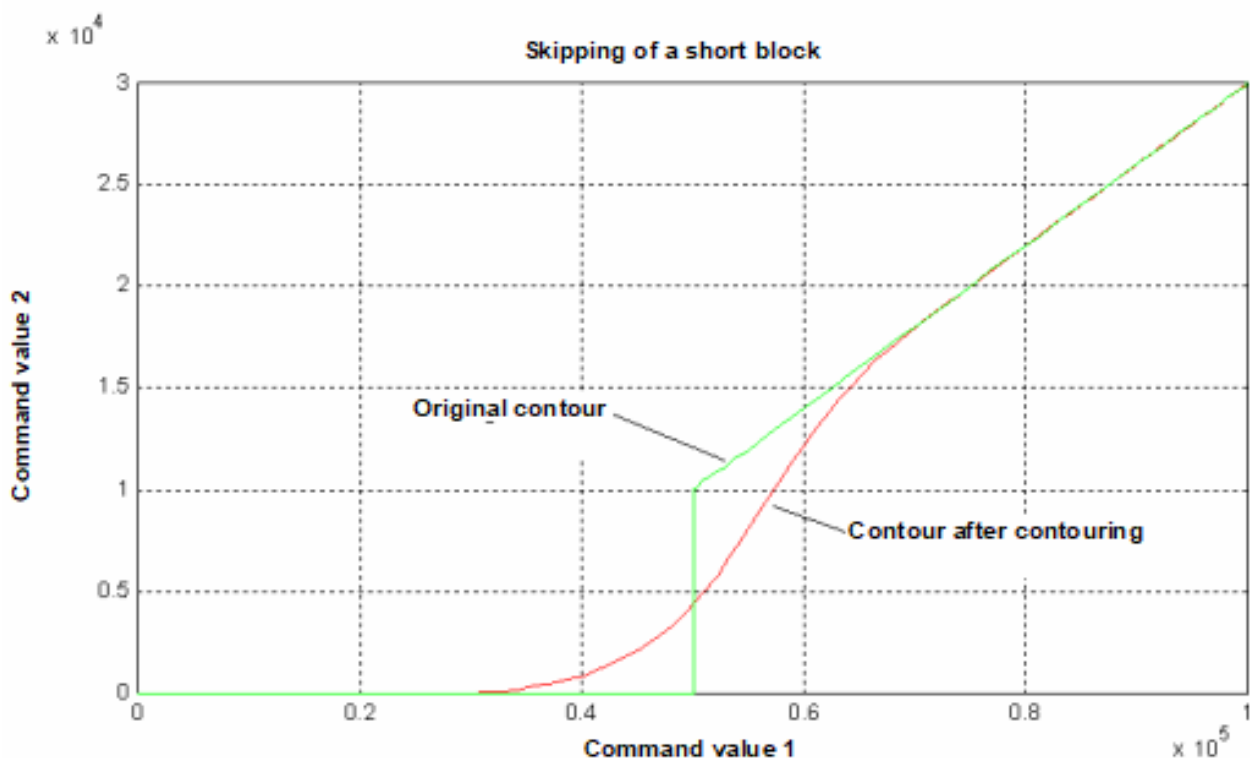

Fig. 75: Example of skipping a short block N05 when contouring

Special case 1: Sequence of multiple short blocks behind the block transition

If multiple sequential blocks (N20, N30, N40) are shorter than the minimum motion path specified, the blocks are skipped provided the distance to the target point from the last relevant end point (N10) is shorter than the specified minimum motion path. If the target point of the skipped block is outside this envelope curve, the block (N40) is used to calculate the contouring curve even if it is shorter than the specified minimum length. This method permits a slight deviation from the original contour even if multiple sequential blocks are skipped.
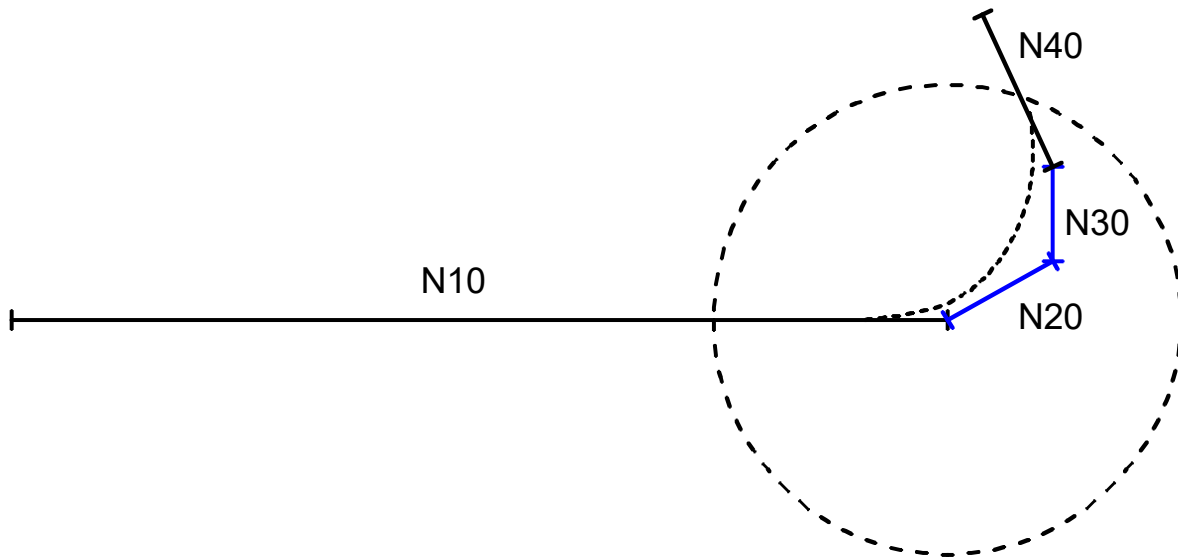
Fig. 76: Some single blocks (N20, N30 and N40) are too short but the target point is outside the minimum block length.

Special case 2: Sequence of multiple short blocks behind the block transition, last block is extremely short

As an exception, the block N40 itself may be shorter than the minimum system-specific length (about 16 µm) required for contouring. In this case, the last end point and the new target point are connected by a linear block. This new linear block N20´ is then used to calculate the contouring curve.
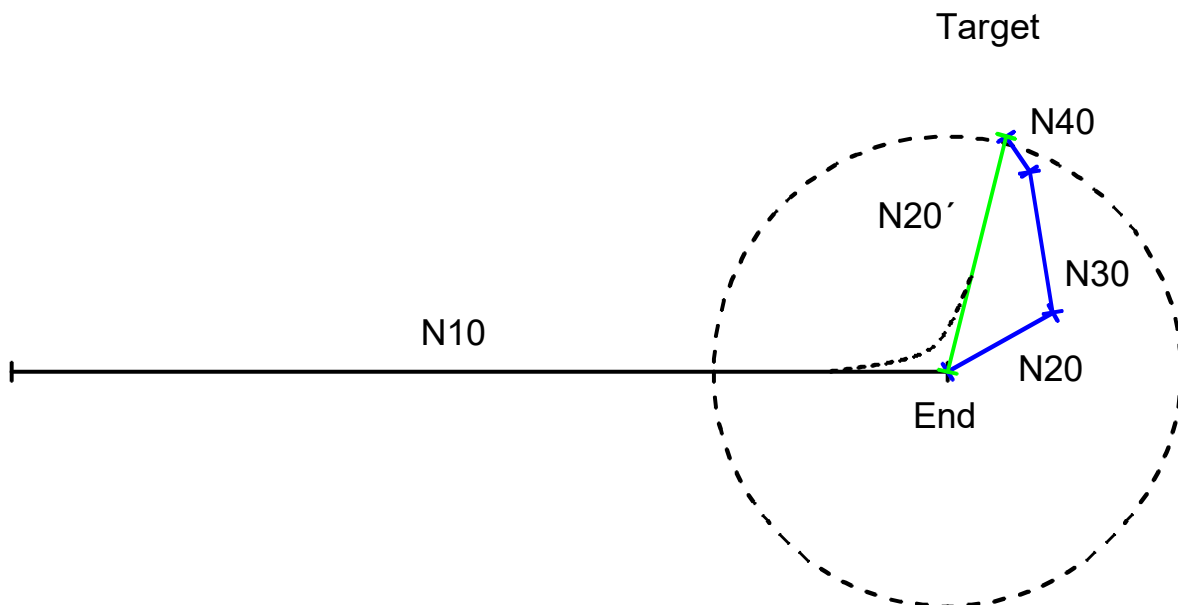
Fig. 77: Single blocks (N20, N30 and N40) are too short but the sum of all blocks exceeds the minimum system-specific block length.

Special case 3: Short blocks before the block transition

If the blocks at the beginning of contouring (before block transition) are already shorter than the minimum system-specific length, the blocks are skipped. The blocks are skipped until the distance between the last valid point and the current target point exceeds the minimum block length. If this is the case, the last end point and the current target point is connected by a linear block N10'. This linear block then is used as the start block for contouring.
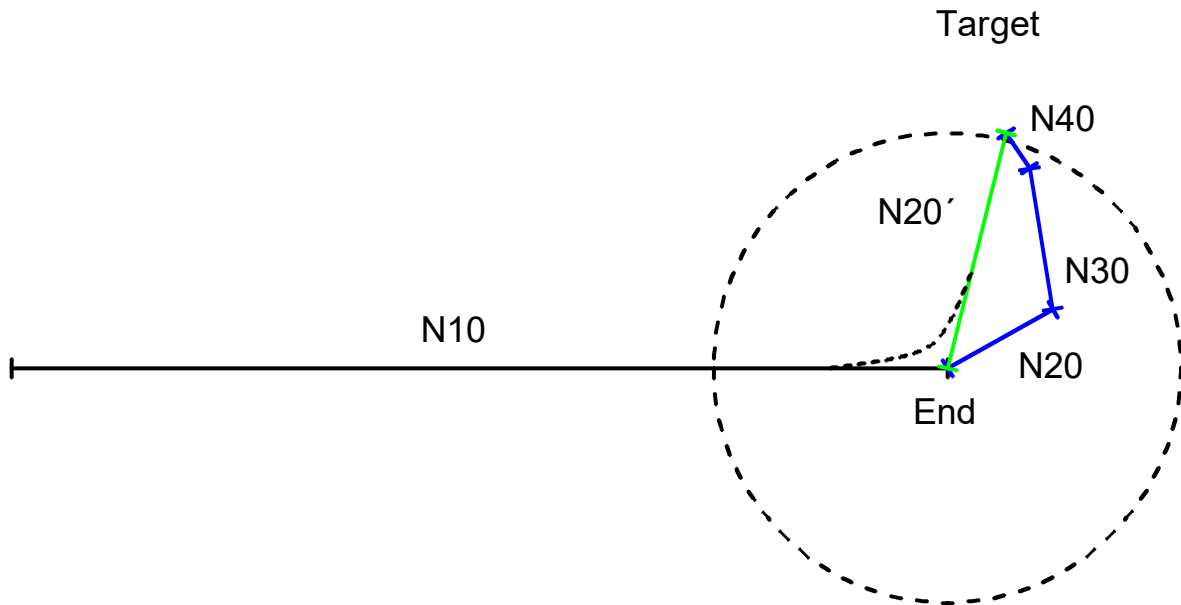
Fig. 78: Multiple blocks (N10, N20 and N30) are too short but the sum of all blocks exceeds the minimum system-specific block length.

Special case 4: Deselect contouring or changing parameterisation

If contouring is deselected when blocks are skipped or the basic conditions for contouring are changed, the current contouring may only be continued up until they are deselected or parameters are changed. After this function, contouring may be continued using the new parameters.

```
#CONTOUR MODE [ DEV, PATH_DEV 5, RELEVANT_PATH 2]
N10 G91 G01 F1000 X10      G61
N20 X2 Y1
N30 Y1.5
N40 X-1 Y2…
```
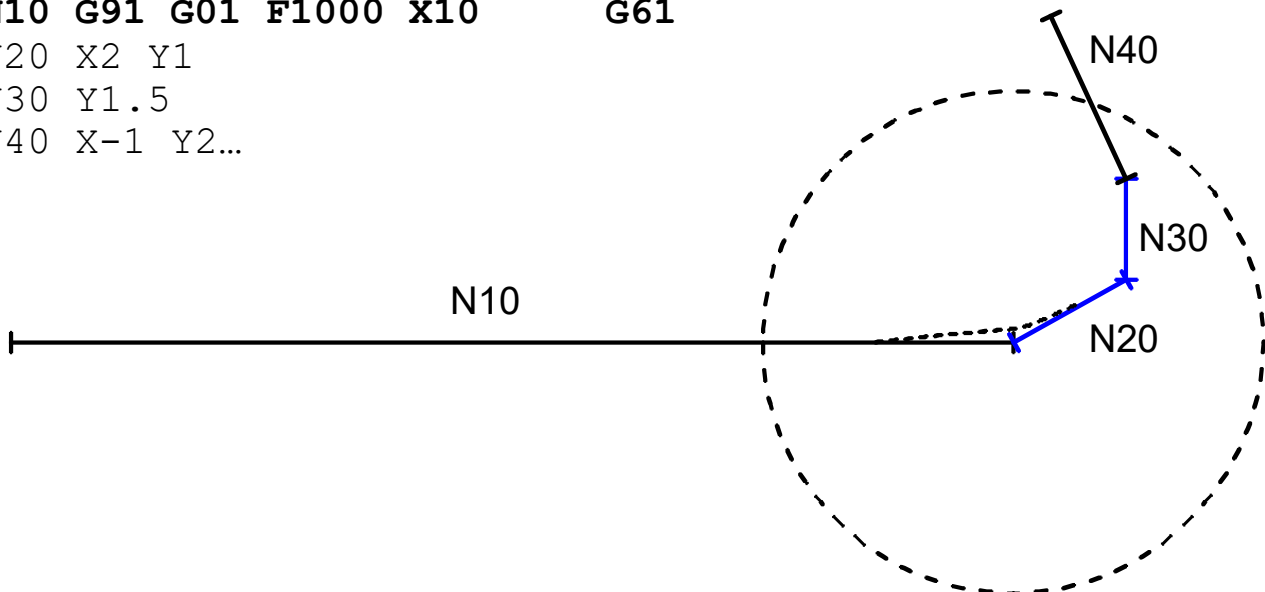

Fig. 79: Some single blocks (N20, N30 and N40) are too short but contouring is deselected as of block N20.

11.2.2.3   Executing additional blocks

If a command without contour information is programmed in addition to the motion blocks at block end (N10 – N20) (e.g. M function requiring acknowledgement with pre-block output and post-block synchronisation, MVS_SNS), the command may be executed before, during or after the contouring curve.

Executing additional blocks

```
N10 X100 G61 M25
N20 Y100
```
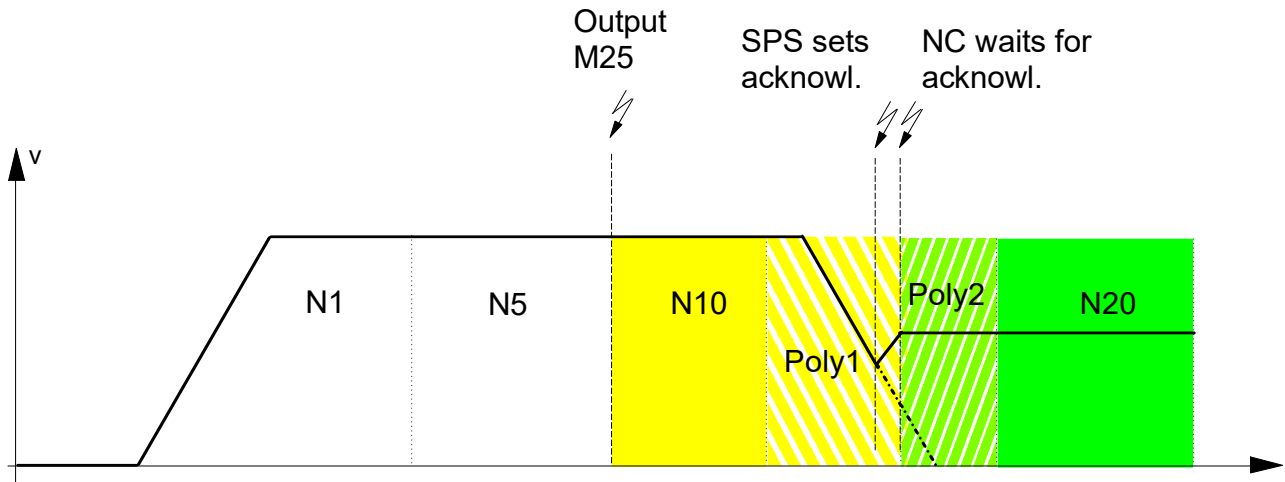

Fig. 80: Synchronisation without contour-relevant actions during contouring
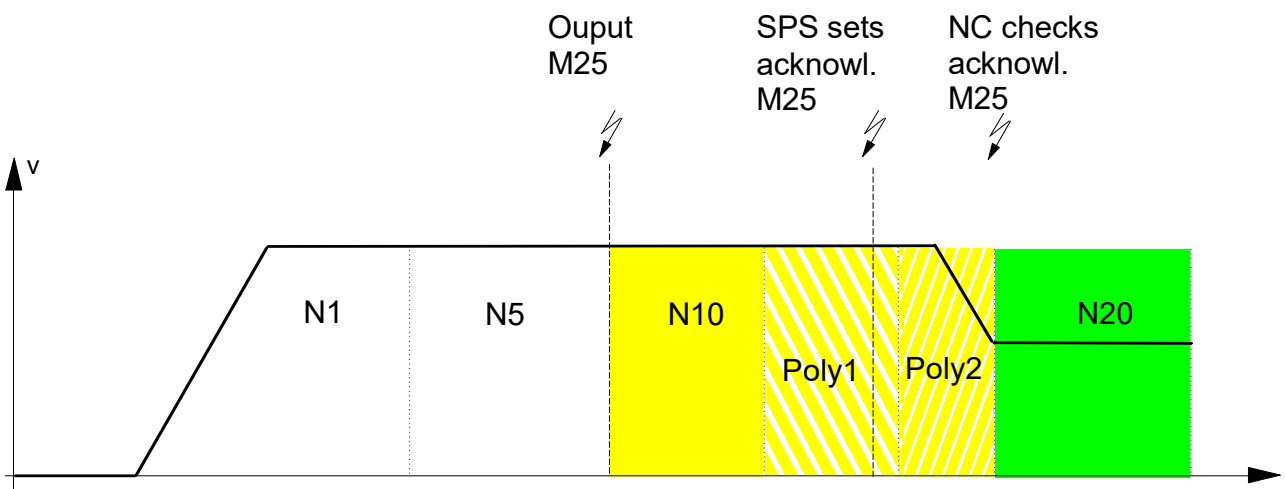
Fig. 81: Synchronisation without contour-relevant actions after contouring

There are 3 options to execute these commands:

1. Directly after pre-block (N10) and before the first contouring polynomial
2. Between the first and the second contouring polynomial
3. After the second contouring polynomial and before the post-block (N20)

11.2.2.4 Jerk within the polynomial

The curvature of the polynomial results in a jerk for the axes running across the path trajectory. This jerk is normally checked with the maximum dynamic parameters of the axes (P-AXIS-00199). If the jerk is too strong, path velocity is reduced accordingly. In some user-specific applications, this reduction in velocity is undesirable because of the maximum jerk. This can be defined specifically by control commands in the NC command #CONTOUR MODE. The control commands overwrite the pre-definition in the channel parameter list P-CHAN-00110 and are valid modal up to program end.

In the example below, the block transition from N6 to N7 is contoured by polynomials and this is considered by the jerk. The transition from N7 to N8 is also contoured but with no consideration for jerk on the path contour.

Jerk within the polynomial

```
%poly_jerk.nc
(default setting in the channel parameter list: (check_jerk_on_poly_path)

#SLOPE[TYPE=TRAPEZ]
#CONTOUR MODE [ DEV, PATH_DEV 4, RELEVANT_PATH 51]
N0003 G1 X0 Y100 Z0 F4

N0004 G261
N0005 G1 G91 X100
N0006 Y-50
N0007 #CONTOUR MODE [CHECK_JERK=1]
N0008 X100
N0009 #CONTOUR MODE [CHECK_JERK=0]
N0010 Y-50
N0009 G260
N0055 M30
```

11.2.2.5   Velocity curve in the contouring section

Depending on axis parameterisation and the application, it may be necessary to influence the velocity curve in the contouring section. In the default definition, the contouring section is travelled at maximum permissible path velocity. If the axes have strongly different dynamics, this could lead to an unacceptable excitation of vibrations in the machine because path velocity is adjusted in the contouring section.

The characteristic in the contouring section can be adjusted by specific control commands in the NC command #CONTOUR MODE.

In the example below, the block transition from N6 to N7 is contoured by polynomials which are moved in the contouring section at maximum velocity, i.e. the velocity is adjusted here by different axis dynamics. The transition from N9 to N10 is also contoured but without any velocity adjustment. This leads to a constant path velocity in the contouring section.

Velocity curve in the contouring section

```
%poly_const_speed
N0003 #SLOPE[TYPE=TRAPEZ]
N0004 G1 X0 Y0 Z0 F8000
N0005 #CONTOUR MODE [CONST_VEL=0]
N0006 X100 G61
N0007 Y100
N0008 #CONTOUR MODE [CONST_VEL=1]
N0009 X0 G61
N0010 Y0
N0020 M30
```
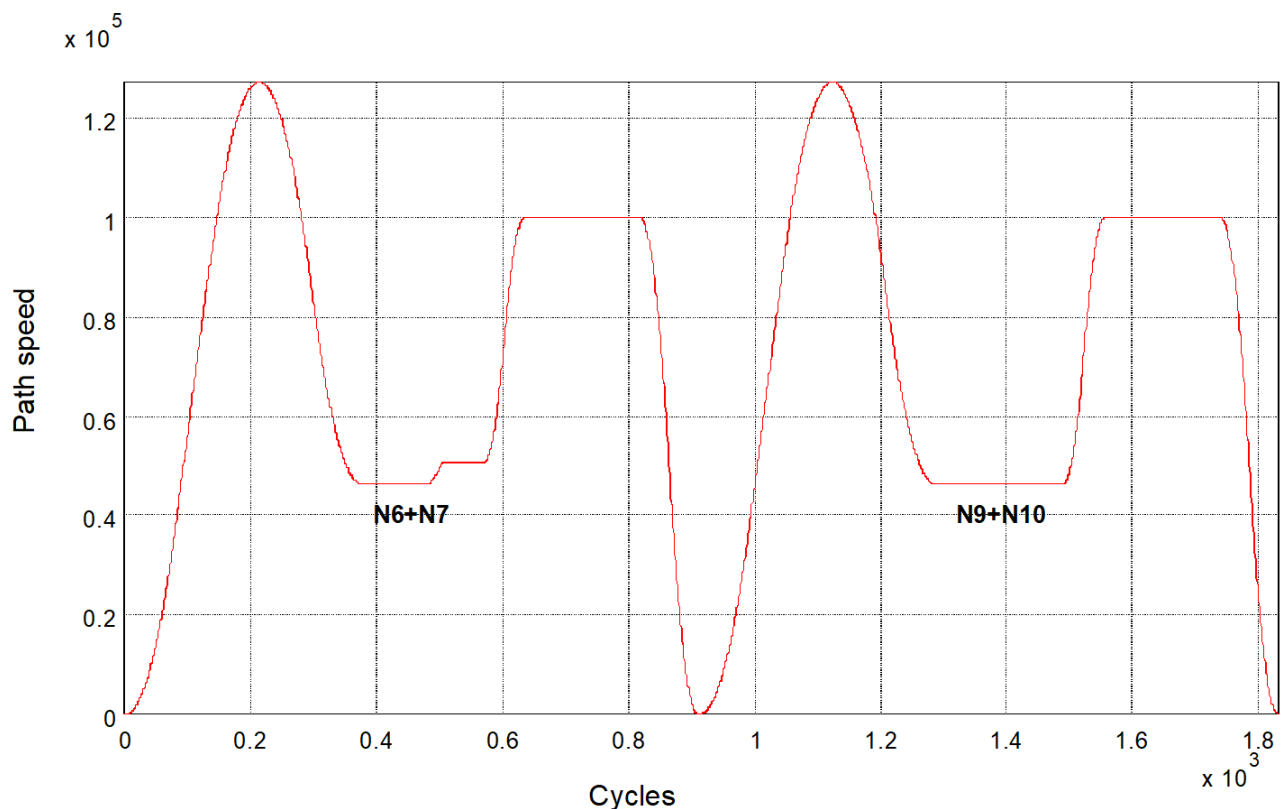

Fig. 82: Characteristic in the transition section

11.2.3 Parameterising contouring modes in the NC program (#CONTOUR MODE)

Before the actual activation of polynomial contouring (G61/G261), the individual options are parameterised by the NC command **#CONTOUR MODE**.

Depending on the contour mode, specific keywords are provided for parameterisation. The command has the following syntax structure:

#CONTOUR MODE [<*contour_mode*> <*parameter*> <*action*>]

<*contour_mode*>	DEV	Contouring with corner deviation (default)
	DIST	Contouring with corner distance
	DIST_SOFT	Dynamic optimised contouring
	DIST_MASTER	Dynamic optimised contouring with master axis
	POS	Contouring with interim point
	PTP	Dynamically optimised contouring of the contour.

<*Parameter*>	PATH_DEV TRACK_DEV ...	**Caution:** The parameters for deviations and tolerances must always be specified in [mm, inch] or [°]. When specifying in [inch], please refer to the note in P-CHAN-00439.

<*action*>	PRE_ACTION	Execute M/H actions related to the contouring curve
	INTER_ACTION	
	POST_ACTION	

11.2.4   Activating contouring modes in the NC program

Contouring is activated by the G functions G61 (blockwise) or G261 (modal) after parameterising the corresponding contouring mode.

i	**Starting at Build V2.11.2022.13 and higher** ... alternatively, contouring may be selected or deselected by additionally specifying ON/OFF in the command #CONTOUR MODE. Programming G261/G260 is then no longer necessary.

Activating contouring modes in the NC program

```
%Contour_on_off
N10 G90 G01 X0 Y0 Z0 A0 C0 F60
N20 #CONTOUR MODE ON [DEV PATH_DEV=1.0]    ;Parameterisation and
                                           ;activation (= G261)
N30 X100
N40 Y100
N50 X0
N60 Y0
N70 #CONTOUR MODE OFF                      ;Deactivation (= G260)
N80 M30
```

11.2.4.1   Contouring with corner deviation

i   Default parameterisation of this contouring type becomes effective after program start.

Corner distances used to shorten motion blocks are automatically determined after purely geometric considerations to prevent a user-specified corner deviation from being exceeded.

Corner distances are limited depending on the specified minimum residual block length. However, both distances are limited symmetrically. In this case, the programmed path velocity has no influence on the contouring curve.

It is better to use the parameter RELEVANT_PATH to obtain optimised contouring. It is recommended to adopt the maximum corner deviation value PATH_DEV.

```
Syntax of parameterisation:
```
#CONTOUR MODE [DEV [**PATH_DEV**=..] [**RELEVANT_PATH**=..] [**TRACK_DEV**=..] [**RELEVANT_TRACK**=..]

[**REMAIN_PART**=..] [<*action*>] [**CHECK_JERK**=..] [**MAX_ANGLE**=..] [**CONST_VEL**=..] **]**

DEV	Contour with maximum corner deviation
PATH_DEV=..	Maximum deviation of programmed contour in [mm, inch]*
	Default value: 1 mm
	* when P-CHAN-00439 is active
RELEVANT_PATH=..	Minimum path length of relevant post-blocks in [mm, inch *].
	Default value: 0 mm
	* when P-CHAN-00439 is active
TRACK_DEV=..	Maximum deviation of tracking axes in [°]
	Default value: 0 °
RELEVANT_TRACK=..	Minimum path of tracking axis for relevant post-blocks in [°]
	Default value: 0 °
REMAIN_PART=..	Distance to go in [0%-100%] of original block
	Default value: 0%
<*action*>	Identifier for time of execution of additional actions (M/H):
	PRE_ACTION: Actions **before** contouring curve.
	INTER_ACTION: Actions **in** contouring curve (default).
	POST_ACTION: Actions **after** contouring curve.
CHECK_JERK=..	Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110) with:
	0: Without jerk monitoring (default).
	1: Jerk monitoring based on the geometric ramp time (P-AXIS-00199). This may reduce path velocity.
	2: Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non-linear velocity profile.

MAX_ANGLE=..   Maximum contour knee angle in [°] for transitions between two linear blocks up to
                      which contouring is active.

                      Default value: 178° (i.e. the entire contour is contoured)

CONST_VEL=..   Constant path velocity in the contouring section with:

                      0: Without constant path velocity (default).

                      1: At constant path velocity.

Contouring with corner deviation

```
...
N100 #CONTOUR MODE [DEV PATH_DEV=5]
N110 G01 X100 G61
N120 G01 Y100
...
```
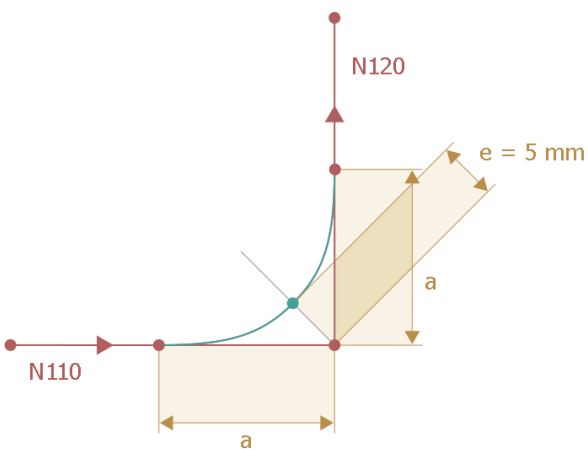

Fig. 83: Contouring with corner deviation

11.2.4.2 Corner distance contouring

If the point from which the original contour may be left is known, the user can explicitly specify the corner distances of the pre-blocks and post-blocks by which the adjacent motion blocks are to be shortened.

The corner distances are limited if they do not exceed the minimum residual block length .

If the corner distances a and b are specified as <u>identical</u>, the other corner distance is symmetrically limited if one corner distance is limited to the minimum remaining block length.

If the corner distances a and b are specified as <u>different</u>, only the distance which is too long is reduced if limited. With asymmetrical path lengths, this can lead to a "degenerated" contour but this may sometimes be desirable.

```
Syntax of parameterisation:
```
#CONTOUR MODE [DIST [**PRE_DIST**=..] [**POST_DIST**=..] [**RELEVANT_PATH**=..]

[**RELEVANT_TRACK**=..] [**TRACK_DEV**=..] [**REMAIN_PART**=..]

[<*action*>] [**CHECK_JERK**=..] [**MAX_ANGLE**=..] [**CONST_VEL**=..] **]**

DIST	Contouring with corner distance specified
PRE_DIST=..	Corner distance in [mm, inch*] after which there is a deviation from the original contour.
	Default value: 1 mm
	*when P-CHAN-00439 is active
POST_DIST=..	Corner distance in [mm, inch*] after which there is a return to the original contour.
	Default value: 1 mm
	*when P-CHAN-00439 is active
RELEVANT_PATH=..	Minimum path length of relevant post-blocks in [mm, inch *]
	Default value: 0 mm
RELEVANT_TRACK=..	Minimum path length of tracking axis for relevant post-blocks in [°].
	Default value: 0 °
TRACK_DEV=..	Maximum deviation of tracking axes in [°]
	Default value: 0 °
REMAIN_PART=..	Distance to go in [0%-100%] of original block
	Default value: 0%
<action>	Identifier for time of execution of additional actions (M/H):
	PRE_ACTION: Actions **before** contouring curve.
	INTER_ACTION: Actions **in** contouring curve (default).
	POST_ACTION: Actions **after** contouring curve.
CHECK_JERK=..	Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110) with:
	0: Without jerk monitoring (default).
	1: Jerk monitoring based on the geometric ramp time (P-AXIS-00199). This may reduce path velocity.
	2: Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non-linear velocity profile
MAX_ANGLE=..	Maximum contour knee angle in [°] for transitions between two linear blocks up to which contouring is active.
	Default value: 178° (i.e. the entire contour is contoured)

CONST_VEL=..   Constant path velocity in the contouring section with:

0: Without constant path velocity (default).

1: At constant path velocity.

Contouring with corner distance

```
...
N100 #CONTOUR MODE [DIST PRE_DIST=10 POST_DIST=5]
N110 G01 X100 G61
N120 G01 Y100
...
```
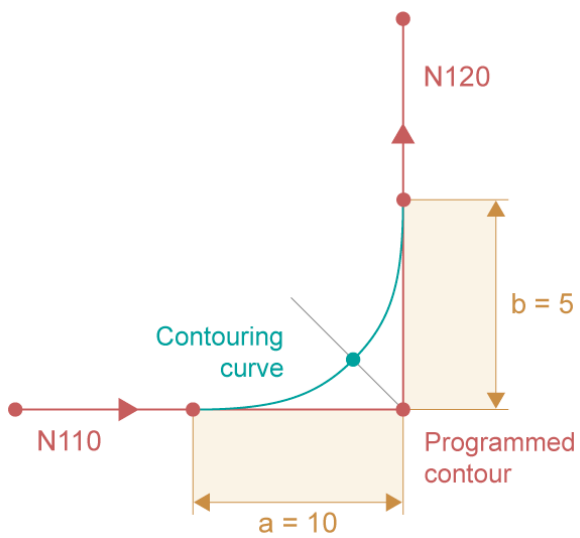

Fig. 84: Corner distance contouring

11.2.4.3   Dynamic optimised contouring

Contouring types with corner deviation and interim point define the contouring curve by a direction- and curvature-continuous connection between two motion blocks. This contouring curve referred to the axes may result in a fluctuation in acceleration.

When the possible dynamic data is considered with reference to the axes (acceleration, jerk), the contouring curve is defined at **uniform acceleration** (minimum jerk) of the two axes. By utilising maximum axis acceleration, the **duration** of the contouring curve is reduced.

```
Syntax of parameterisation:
```

#CONTOUR MODE [DIST_SOFT [**PATH_DIST**=..] [**TRACK_DIST**=..] [**ACC_MAX**=..] [**ACC_MIN**=..]

[**RAMP_TIME**=..] [**DIST_WEIGHT**=..] **]**

DIST_SOFT	Dynamic optimised contouring
PATH_DIST=..	Corner distance of pre-block and post-block (symmetrical) in [mm, inch *] after which a deviation from the original contour is allowed. The definition refers to the motion path of the feed axes.
	Default value: 1 mm
	Monitoring off: -1 mm
	*when P-CHAN-00439 is active
TRACK_DIST=..	Corner distance to pre-block and post-block in [°] after which non-feed axes (tracking axes) may deviate from the original contour.
	Default value: Value is adopted automatically from PATH_DIST provided this value was not explicitly specified (since program start).
	Monitoring off: -1 °
ACC_MAX=..	Percentage in [0%-100%] of maximum axis acceleration (machine data) which may be used by the contouring curve.
	Default value: 100%
ACC_MIN=..	Percentage in [0%-100%] of maximum axis acceleration (machine data) which should be used by the contouring curve. If the specified corner distance (see PATH_DIST) is not maintained, the acceleration is increased to maximum value (ACC_MAX).
	Default value: 50%
RAMP_TIME=..	Percentage weighting of the ramp time in [0%-10000%].
	Default value: 100%
DIST_WEIGHT=..	Influences the split of contoured linear blocks in [0%-100%]: In the 0% pre-definition, all blocks are halved; at 100%, the split ratio corresponds to the lengths of adjacent blocks. This value can be used to combine the two methods by percentage.
	Default value: 0%

Restrictions:

- If a circular block is used for contouring, the contouring curve is calculated with corner distance without dynamic optimisation.
- The calculation uses only one ramp time (maximum value of the four individual ramp times).
- No processing of kinematic transformations. In this case, calculation is performed with corner distance without dynamic optimisation.
- In many cases, weighting the corner distances by the parameter DIST_WEIGHT depending on the pre-/post-blocks results in an optimised utilisation of the available block length.

When axis-specific contouring is executed, the corner distances of the pre-block and post-block are always identical (symmetrical). If the maximum corner distances are also limited on the half block motion path, a shorter contouring section and therefore a lower contouring velocity results for longer motion paths due to the shorter preceding/following motion path.
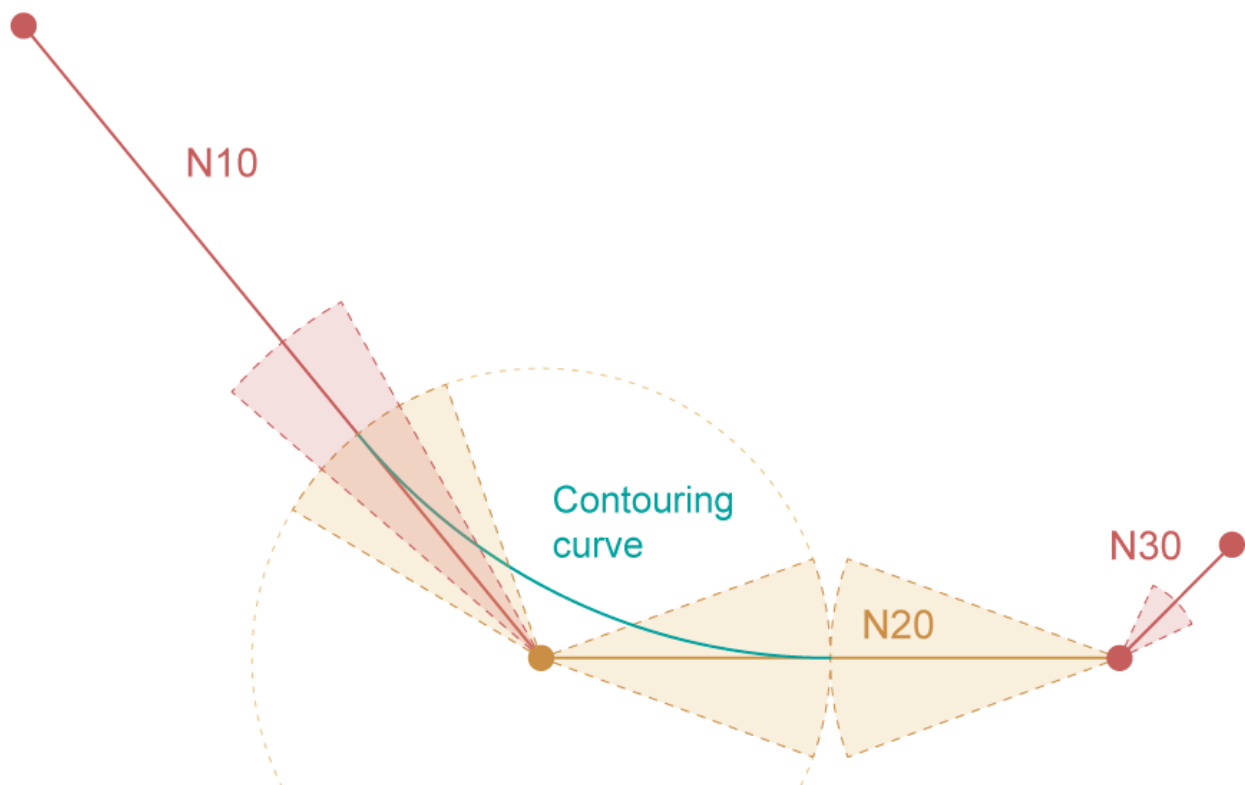
Fig. 85: Maximum corner distance of block N20 independent of the block lengths of N10 and N20 (DIST_WEIGHT = 0%)

If the length of the pre- and post-blocks are considered in the calculation of the maximum corner distances, the contouring zone can be increased.

BECKHOFF
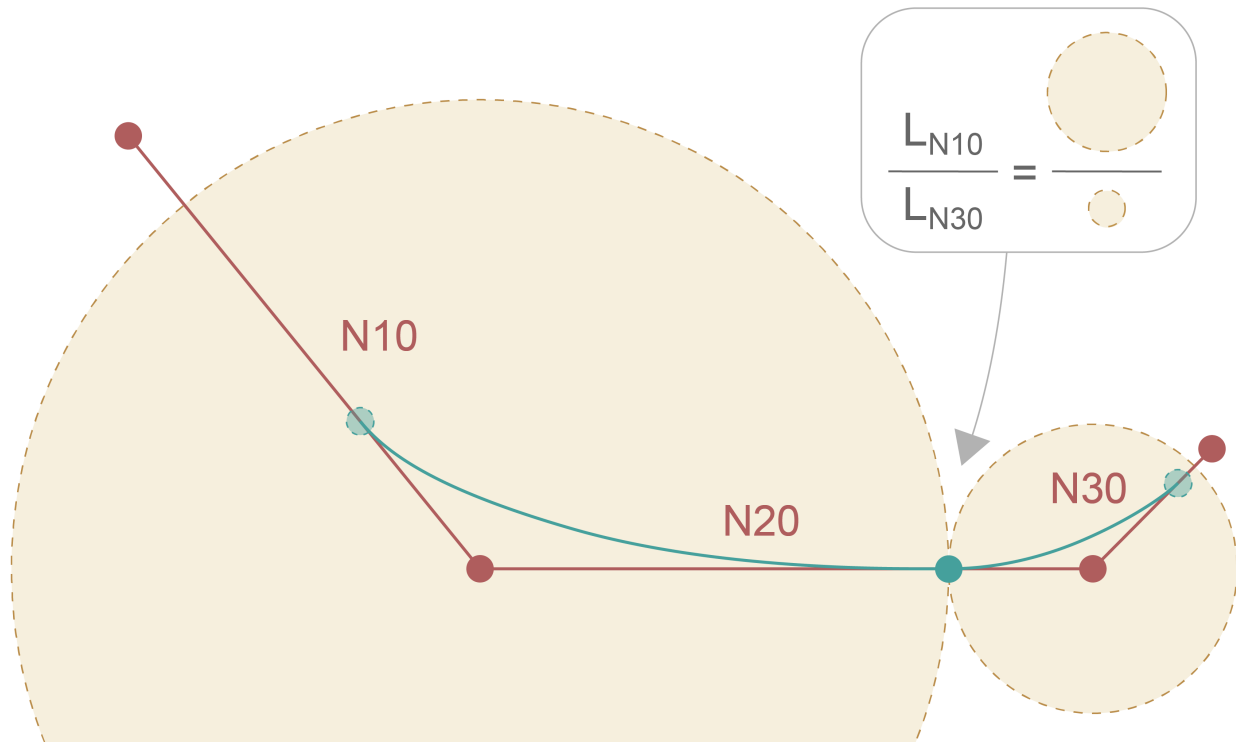
Fig. 86: Maximum corner distance of block N20 subdivided relative to the block lengths of N10 and N30 (DIST_WEIGHT = 100%)

Dynamic optimised contouring

Comparison of contouring of a 90° corner with the methods:

- Dynamically optimised contouring (DIST_SOFT):

```
N010 #CONTOUR MODE [DIST_SOFT PATH_DIST=12]
N020 G0 X0 Y80
N030 G261
N040 G01 X40 Y40 F2.5
N050 G01 X80 Y80
N060 G260
N070 M30
```

- Contouring with corner deviation (DEV):

```
N010 #CONTOUR MODE [DEV PATH_DEV=0.2]
N020 G0 X0 Y80
N030 G261
N040 G01 X40 Y40 F2.5
N050 G01 X80 Y80
N060 G260
N070 M30
```

Comparison of contouring curves:
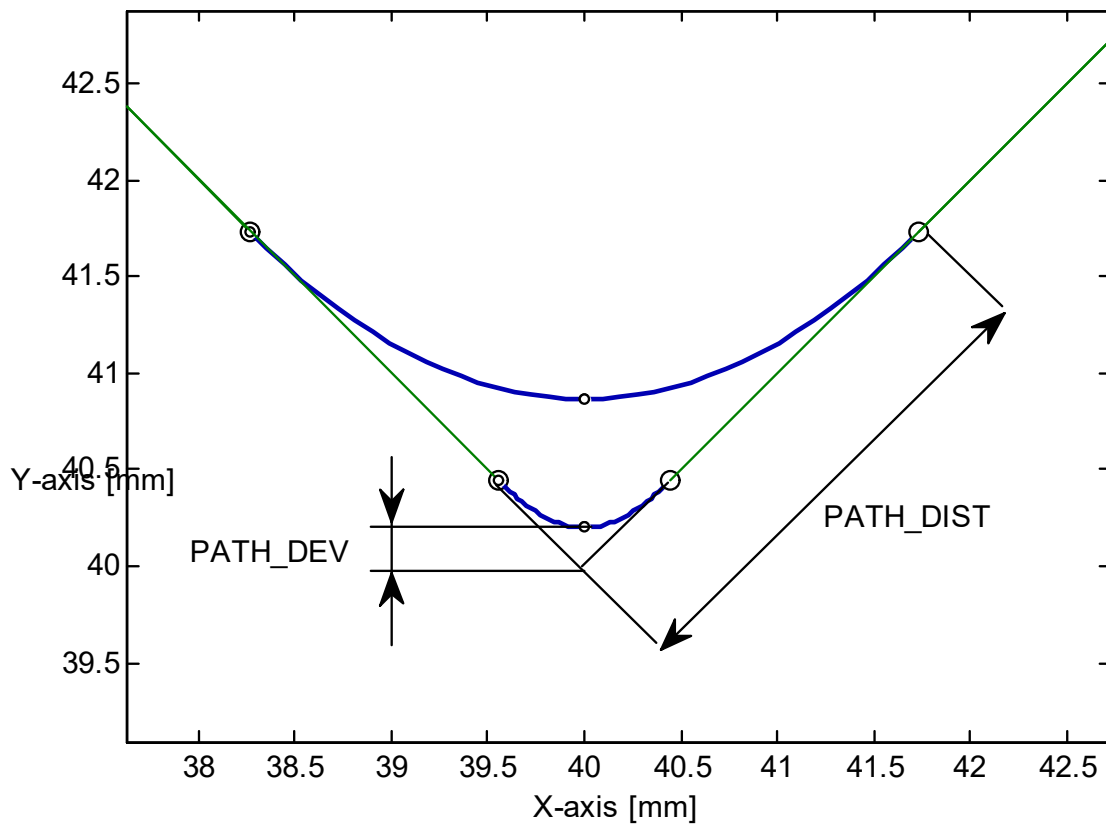
11.2.4.4   Dynamic optimised contouring with master axis

A feed master axis is used in this variant of the dynamically optimised contouring curve. This generally results in a more favourable velocity profile.

The feed master axis is identified in the axis parameter list by an entry in P-AXIS-00015 and marked as the only feed axis in the channel parameter list (P-CHAN-00011).

Further properties and limitations correspond to the dynamically optimised contouring mode.

`Syntax of parameterisation:`

#CONTOUR MODE [DIST_MASTER [**SYM_DIST**=..] [**ACC_MAX**=..] [**ACC_MIN**=..]

[**RAMP_TIME**=..] [**DIST_WEIGHT**=..] **]**

DIST_MASTER	Dynamically optimised contouring with feed master axis
SYM_DIST=..	Corner distance of pre-block and post-block (symmetrical) in [mm, inch *] after which a deviation from the original contour is allowed.
	Default value: 1 mm
	Monitoring off: -1 mm
	*when P-CHAN-00439 is active
ACC_MAX=..	Percentage in [0%-100%] of maximum axis acceleration (machine data) which may be used by the contouring curve.
	Default value: 100%
ACC_MIN=..	Percentage in [0%-100%] of maximum axis acceleration (machine data) which should be used by the contouring curve. If this specified corner distance (see SYM_DIST) is not maintained here, the acceleration is increased up to maximum value (ACC_MAX).
	Default value: 50%
RAMP_TIME=..	Percentage weighting of the ramp time in [0%-10000%].
	Default value: 100%
DIST_WEIGHT=..	Percentage weighting of corner distances relative to the pre-/post-block in [0%-100%].
	Default value: 0%

11.2.4.5 Contour with interim point

Here, the user specifies not only the corner distances but also an interim point P at which the two polynomial curves are adjacent to each other (expert mode). This mode permits the retention of the programmed contour and fully utilises the dynamics by specifying the corner distance zero. In other words, the corner distances need not be symmetrical here.

Syntax of parameterisation:

#CONTOUR MODE [POS [**PRE_DIST**=..] [**POST_DIST**=..] [**X**..] [**Y**..] [**Z**..] [<*action*>]

[**CHECK_JERK**=..] [**CONST_VEL**=..] **]**

POS	Contour by specifying the interim point
PRE_DIST=..	Corner distance in [mm, inch*] after which there is a deviation from the original contour. The value 0 mm is possible here.
	Default value: 1 mm
	*when P-CHAN-00439 is active
POST_DIST=..	Corner distance in [mm, inch*] after which there is a return to the original contour. The value 0 mm is possible here.
	Default value: 1 mm
	*when P-CHAN-00439 is active
X..	Position of interim point in the first main axis in [mm, inch]
Y..	Position of interim point in the second main axis in [mm, inch]
Z..	Position of interim point in the third main axis in [mm, inch]
<*action*>	Identifier for time of execution of additional actions (M/H):
	PRE_ACTION: Actions **before** contouring curve.
	INTER_ACTION: Actions **during** the contouring curve (default).
	POST_ACTION: Actions **after** contouring curve.
CHECK_JERK=..	Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110) with:
	0: Without jerk monitoring (default).
	1: Jerk monitoring based on the geometric ramp time (P-AXIS-00199). This may reduce path velocity
	2: Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non-linear velocity profile
CONST_VEL=..	Constant path velocity in the contouring section with:
	0: Without constant path velocity (default).
	1: At constant path velocity.

Contour with interim point

```
...
N100 #CONTOUR MODE [POS PRE_DIST=2 POST_DIST=3 X110 Y-10 Z0]
N110 G01 X100 G61
N120 G01 Y100
...
```
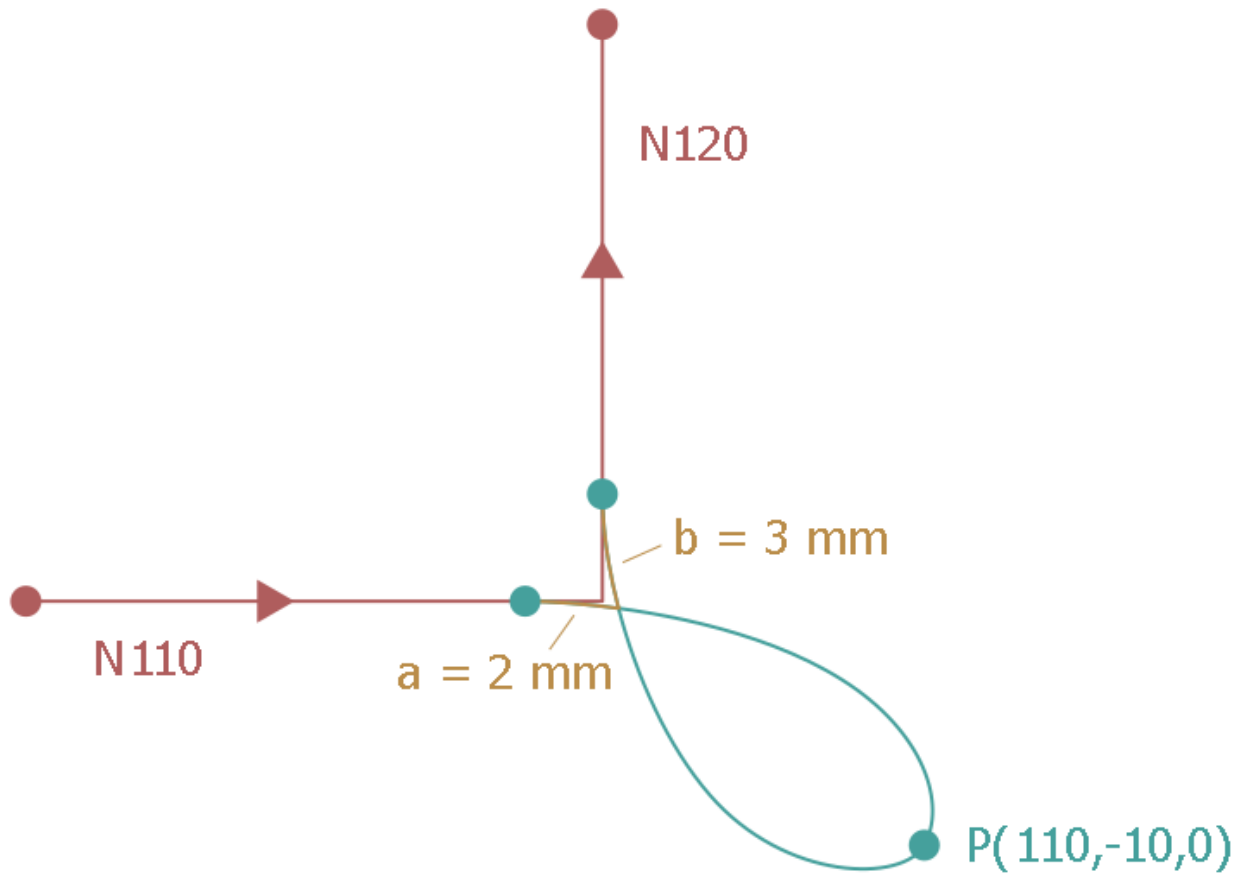

Fig. 87: Contour with interim point

11.2.4.6 Dynamically optimised contouring of the complete contour

This mode is suitable for handling tasks where the feedrate need not be constant in the rounded contour. The contouring curve is selected so that at least one axis involved utilises the dynamics available. As opposed to dynamically optimised contouring (DIST_SOFT), this mode involves the entire contour. The figure below shows a typical application:
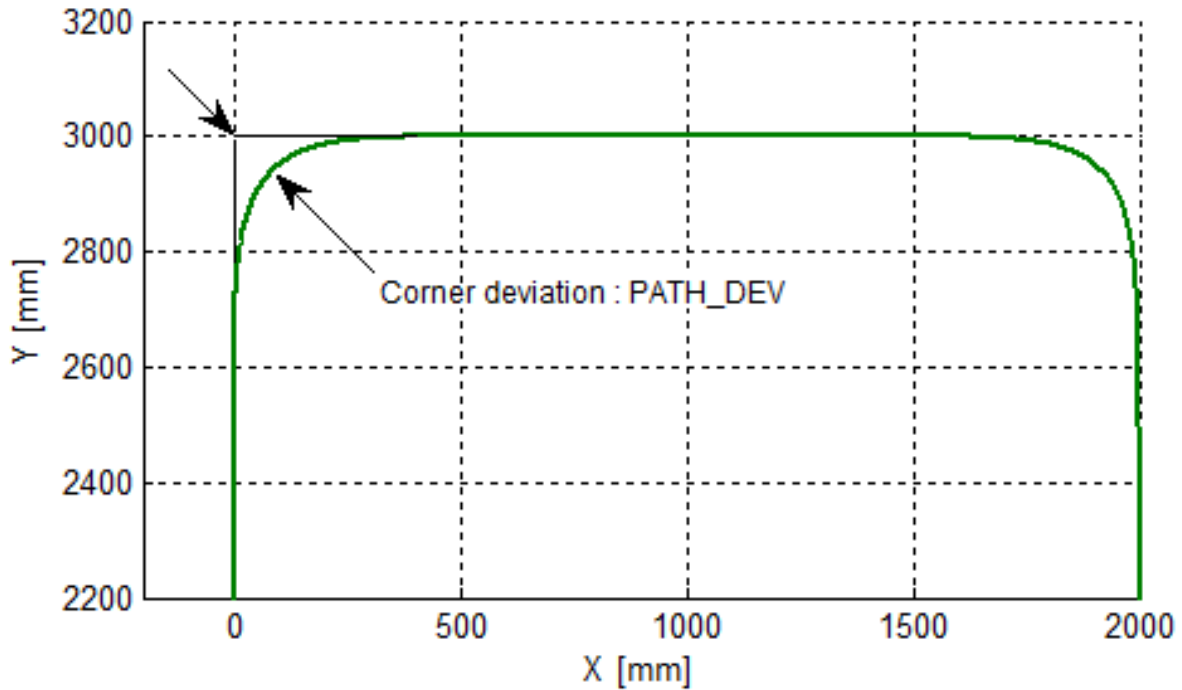
Comprehensive planning avoids unnecessary acceleration zeroes at block limits and calculates uniform velocity profiles as shown in the figure below.
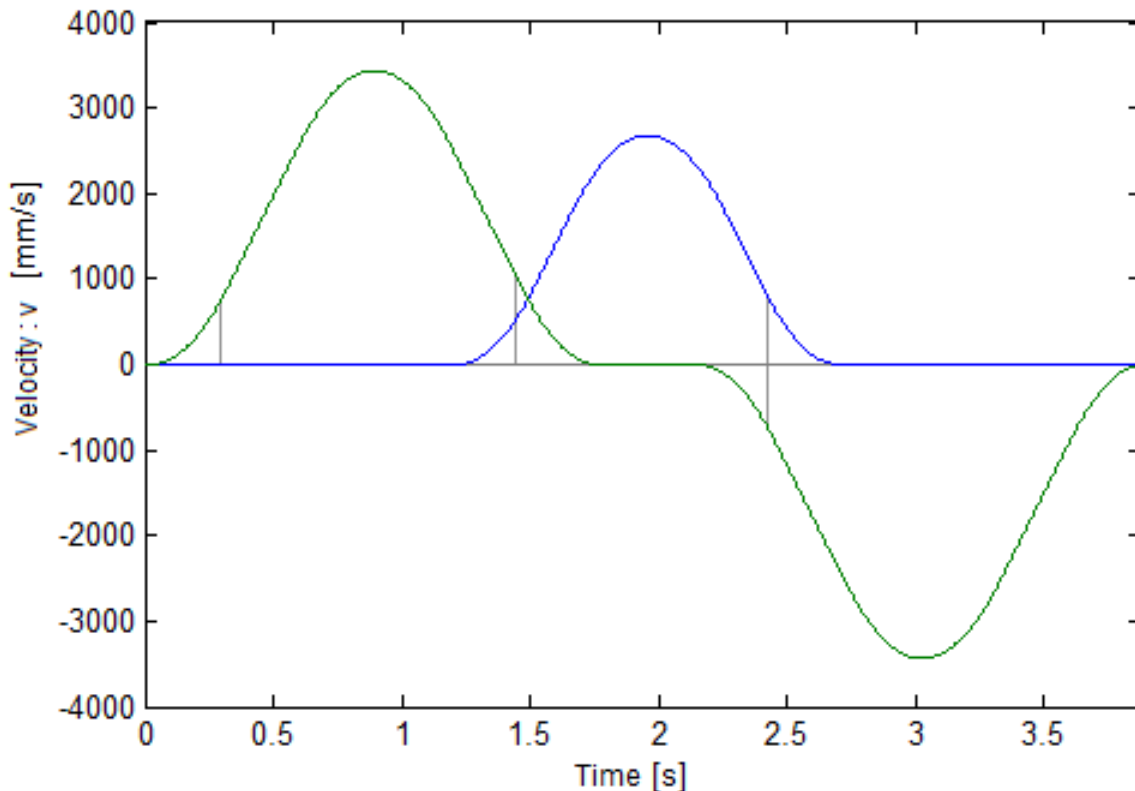
Accelerations with constant jerk change to reduce further excitations. The acceleration phases are then placed in the straight sections before and after the rounded contour:
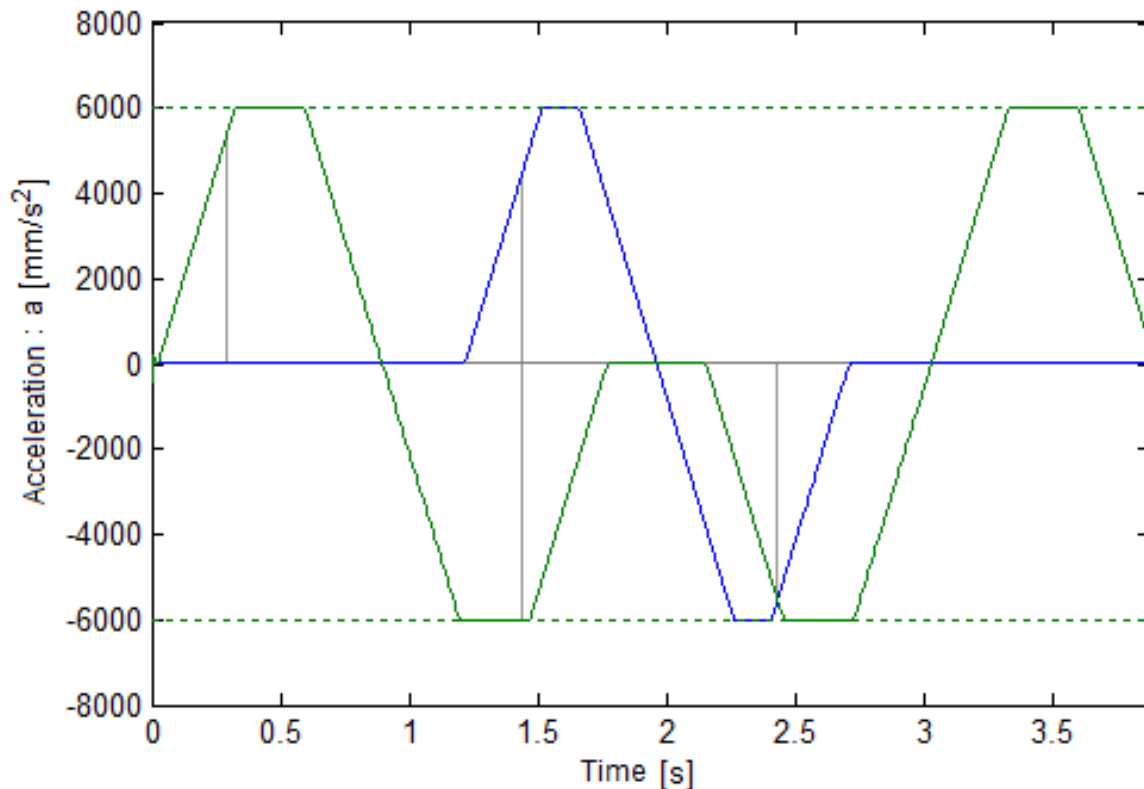
The corner deviation defines the distance of the rounded contour to the program corner point.

If the position is known at which a deviation from the original contour is permitted, the user can explicitly specify the amount of pre-block and post-block corner distances by which the adjacent motion blocks are shortened. The corner distances are limited if they do not exceed the minimum residual block length .

```
Syntax of parameterisation:
```
#CONTOUR MODE [PTP [PATH_DEV=..] [PATH_DIST=..] [MERGE=..] [<action>]]

PTP	Axis-specific contouring with specification of corner distance [as of Build V3.1.3052.01]
PATH_DEV=..	Maximum corner deviation from the programmed contour in [mm, inch *].
	Default value: 1 mm
	*when P-CHAN-00439 is active
PATH_DIST=..	Corner distance of pre-block and post-block (symmetrical) in [mm, inch *] after which a deviation from the original contour is allowed. The definition refers to the motion path of the feed axes **[as of Build V3.1.3079.16].**
	Default value: 1 mm
	*when P-CHAN-00439 is active
MERGE=..	Merge tangential blocks **[as of V3.1.3079.16]** where:
	0: Do not merge
	1: Merge (default)
action	Identifier for time of execution of additional actions (M/H) where:
	PRE_ACTION: Actions **before** contouring curve.
	INTER_ACTION: Actions **in** contouring curve (default).
	POST_ACTION: Actions **after** contouring curve.

NOTICE

This mode is not suitable for:

- Programs containing many short motion paths (see also HSC).
- Programs with **circular blocks** since this results in the automatic deselection of the mode.

NOTICE

This functionality can only be used if the start-up parameter is parameterised for each channel in which the function is to be used.

Example of a setting in the start-up list:

```
configuration.channel[].path_preparation.function FCT_DEFAULT|FCT_PTP
```

Dynamically optimised contouring of the complete contour

```
...
N100 #CONTOUR MODE [PTP PATH_DEV=5]
N110 G01 X100 G61
N120 G01 Y100
...
```
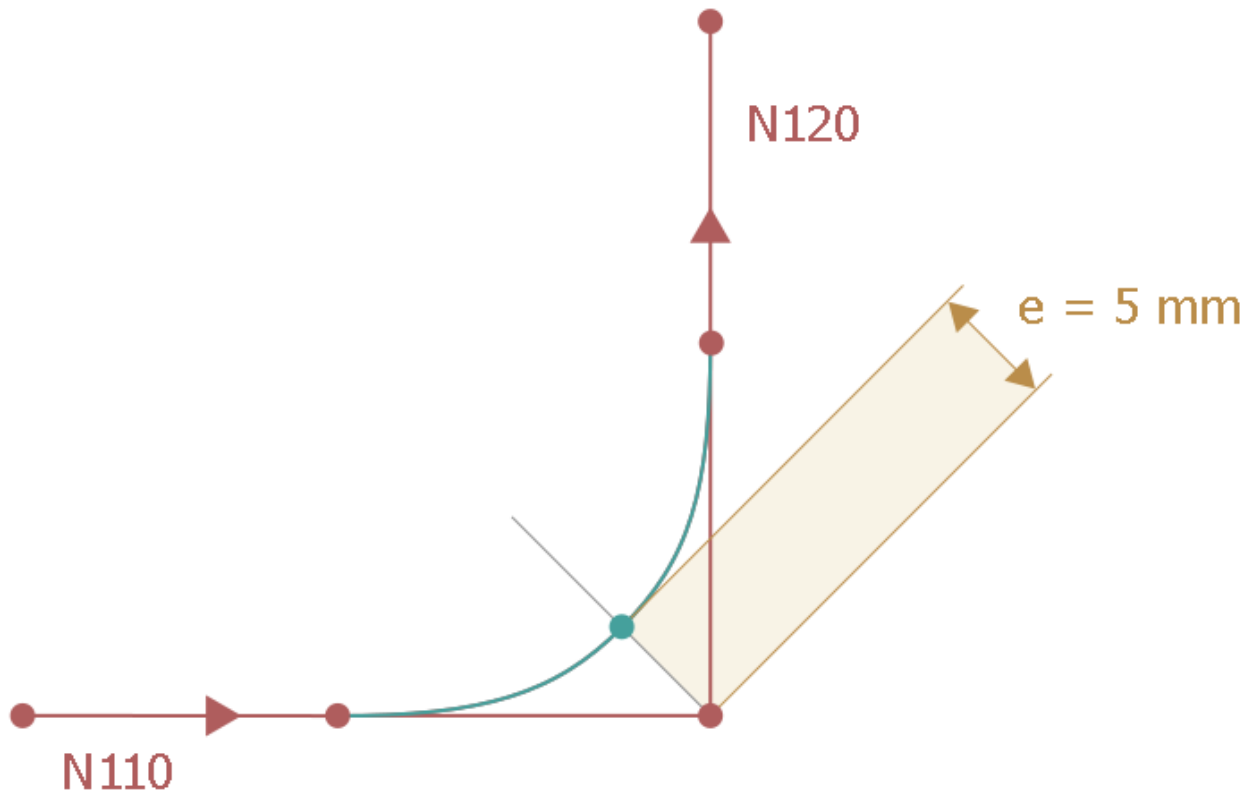

Fig. 88: Dyn. optimised contouring of the entire contour specifying corner deviation

---

11.2.5   Example

The examples below show the influence of the different output of M functions during contouring.

```
N907090              X0 Y0
G91 G01 F6000
N01 #CONTOUR MODE [DEV PATH_DEV=10 POST_ACTION]

N10 X100 G61 M25     (MVS_SNS)
N20 Y100 F3000
N30 X100 G61 F6000
N40 G04 X2
N50 Y100
N00 X0 Y0

N60 X100 G61
N70 Y100 M26         (MVS_SVS)

N907091 G04 X1
```

Output before the contouring curve:

```
N01 #CONTOUR MODE [DEV PATH_DEV=10.0 PRE_ACTION]
```
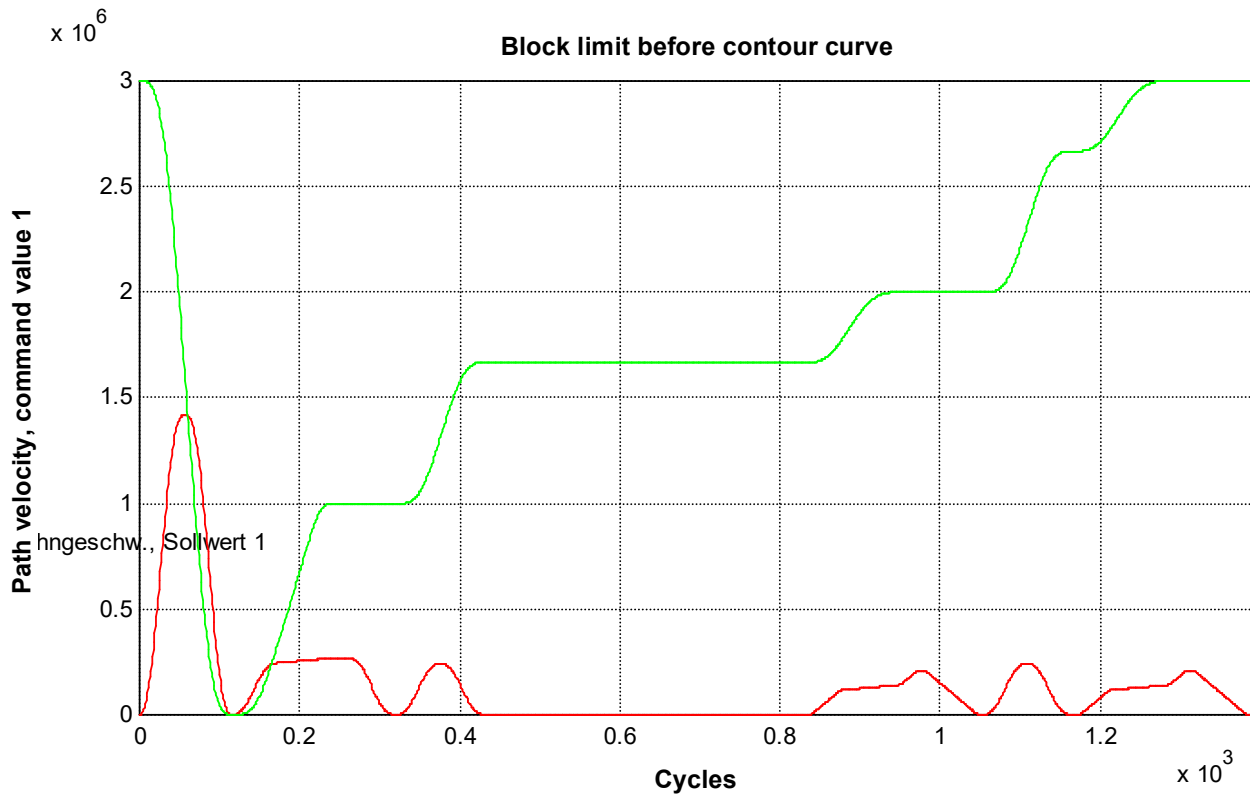

Fig. 89: Block limit before contouring curve

Output within contouring curve:

```
N01 #CONTOUR MODE [DEV PATH_DEV=10.0 INTER_ACTION]
```
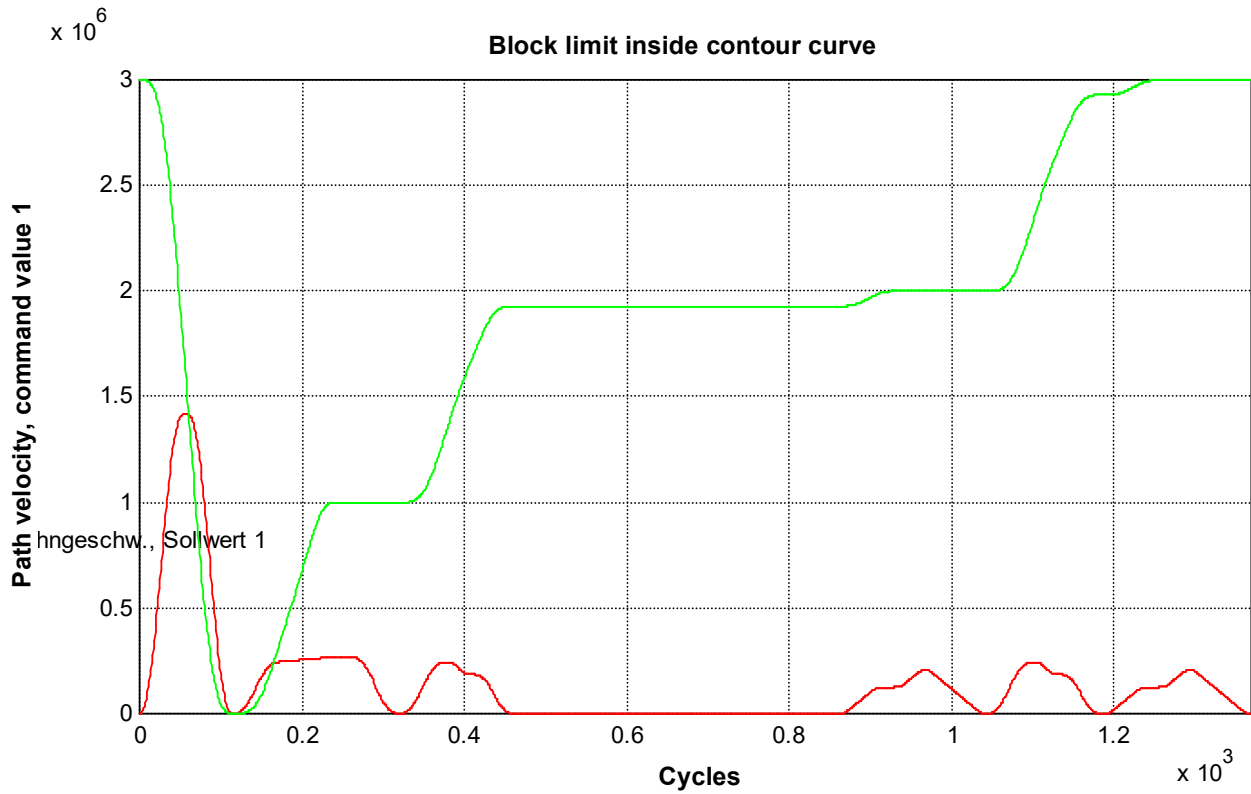
Fig. 90: Block limit within contouring curve

Output after contouring curve:

```
N01 #CONTOUR MODE [DEV PATH_DEV=10.0 POST_ACTION]
```
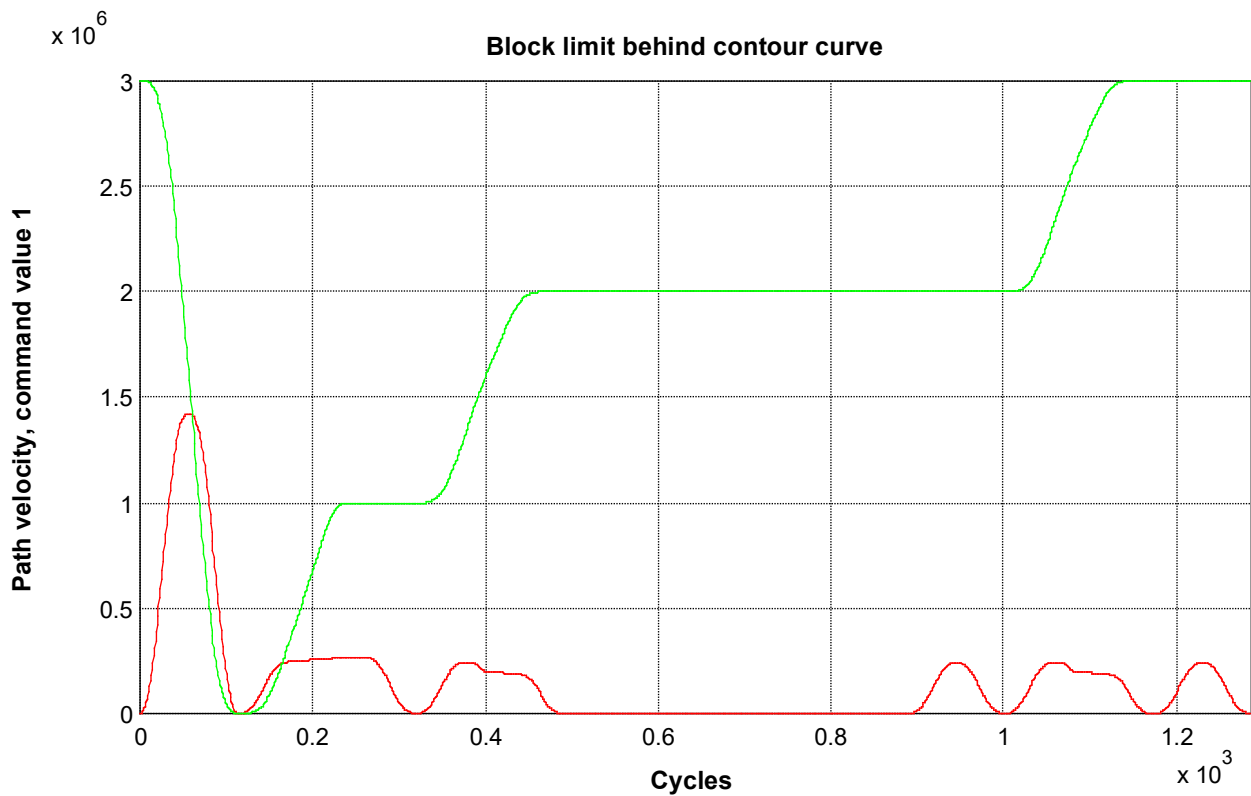
Fig. 91: Block limit after contouring curve

If the acknowledgement of M25 is delayed by block N10, the motion is stopped after the contouring curve and the program waits for PLC acknowledgement.
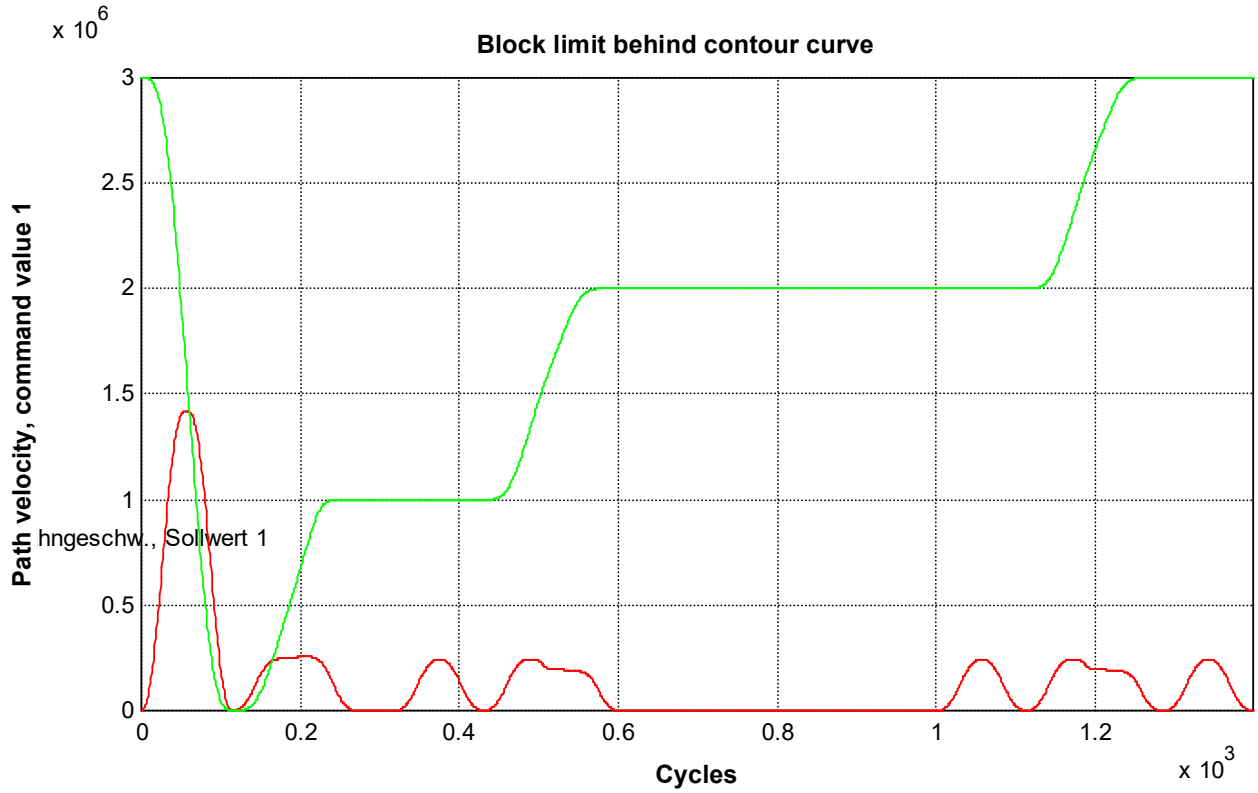
Fig. 92: Block limit after contouring curve

Change the limit angle during contouring:

```
#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELEVANT_TRACK=0.2]
F10000

G261
N5 #CONTOUR MODE [MAX_ANGLE=3]
N10 G01 X0 Y0 Z0 G61
N15 #CONTOUR MODE [MAX_ANGLE=4]
N20 G01 X100 Y0 Z0
N25 #CONTOUR MODE [MAX_ANGLE=5]
N30 G01 X100 Y100 Z0
N35 #CONTOUR MODE [MAX_ANGLE=6]
N40 G01 X0 Y0 Z0 G61
G260
```

Result:

Contour the N block always takes place at the limit angle of the previous N<i-5> block.

Variation of the contour angle with constant limit angle:

```
#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELEVANT_TRACK=0.2]
#CONTOUR MODE [RELEVANT_TRACK=0.3]
P100 = 50
F10000

#CONTOUR MODE [MAX_ANGLE=73]
N10 G01 X-P100 Y0 Z0 C0 A0

$FOR P123 = 0, 90, 7.5
  N2 G01 X0 Y0 Z0 C0 A0 G61
  P1 = COS[P123]*P100
  P2 = SIN[P123]*P100
  NP123 XP1 YP2
  N100 G01 X-P100 Y0 Z0 C0 A0
$ENDFOR

$FOR P123 = 270, 370, 7.5
  N120 G01 X0 Y0 Z0 C0 A0 G61
  P1 = COS[P123]*P100
  P2 = SIN[P123]*P100
  NP123 XP1 YP2
  N400 G01 X-P100 Y0 Z0 C0 A0
$ENDFOR

M30
```

red alpha=43°, blue alpha=73°

red alpha=43°, blue alpha=73°
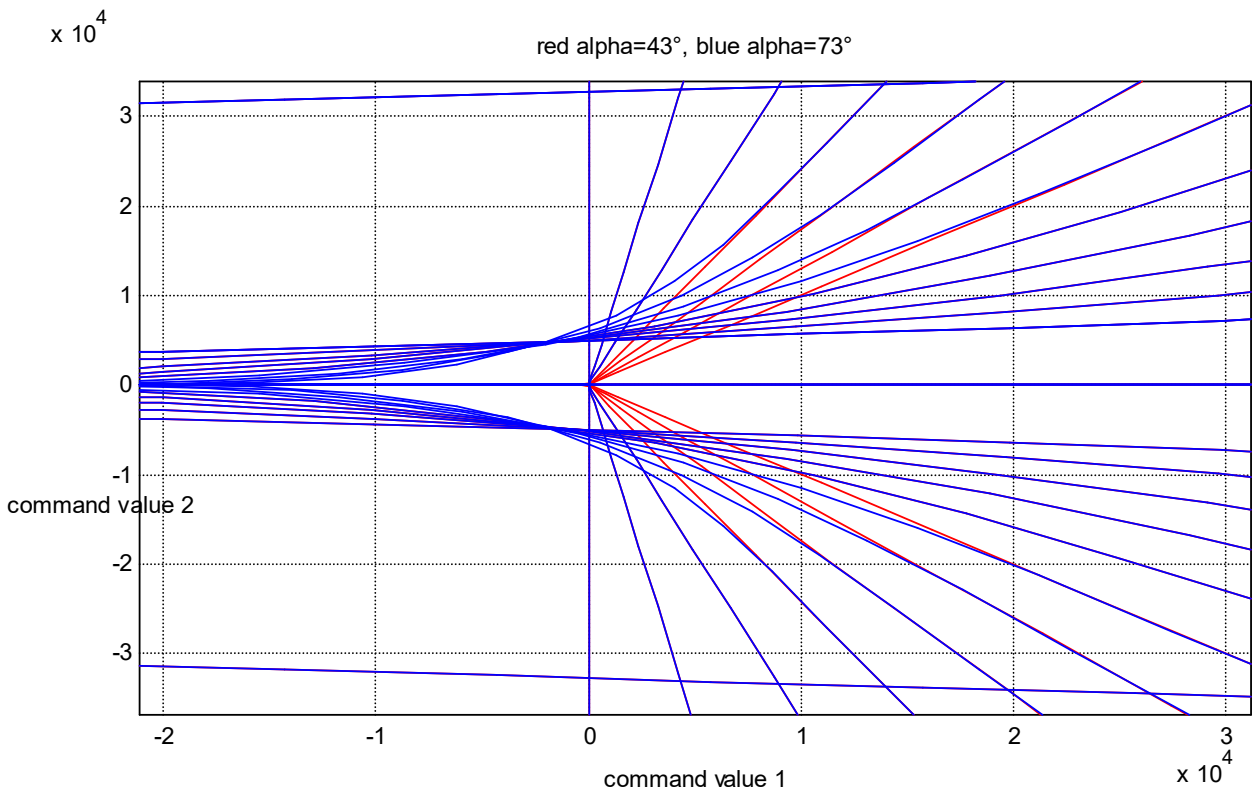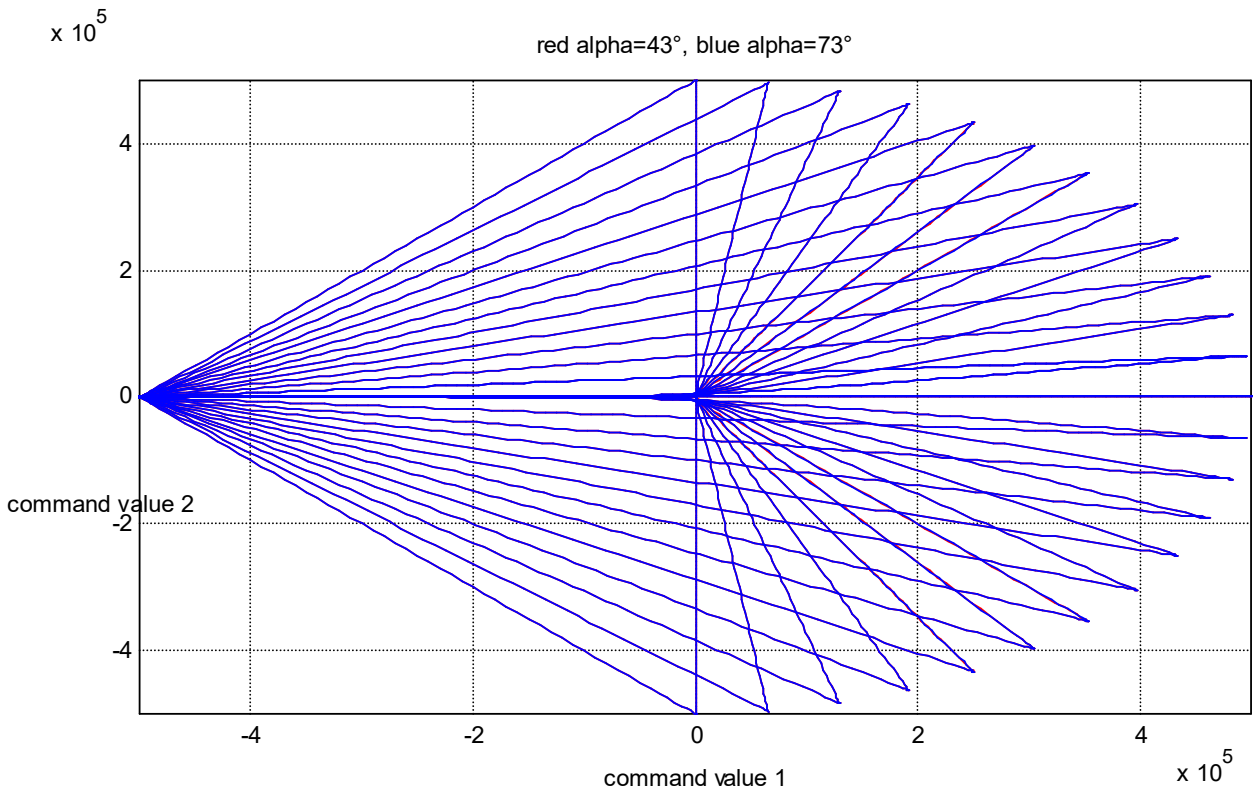
11.2.6   Remarks

If axes are released or fetched after programming G61 or G261/G260 (contouring at block end), the contouring mode cannot be executed.

Contouring mode is not executed

```
N10 G01  X100  Y0  Z0  F1000
N20 G01  X50   Y50  G61
N30 #PUT AX [Z](Contouring mode is not executed)
N40 G01  X100
N50 M30
```

With interim point contouring, the shape of the curve depends on the choice of interim point. The following curve shape is also possible:
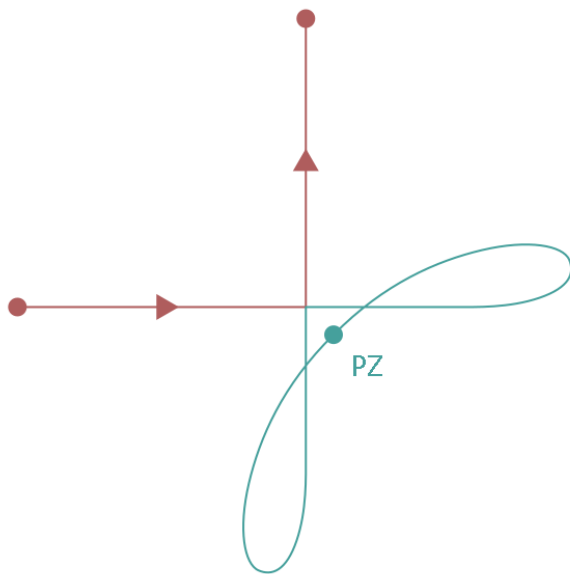
11.3 Other processes

11.3.1 Akima spline interpolation

> **i** The use of this feature requires a license for the "Spline" extension package. It is not included in the scope of the standard license.

The target points of the programmed linear blocks (G00 and G01) are the vertices through which the Akima splines are fitted.

11.3.1.1 Selecting AKIMA spline type (#SPLINE TYPE AKIMA)

> **i** As of Build V2.11.2010.02 the command **#SPLINE TYPE AKIMA**replaces the command **#SET SPLINETYPE AKIMA**.. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:
#SPLINE TYPE AKIMA

If provided in the system, the Akima spline is the default spline type. Nevertheless, it is recommended to specify the spline type explicitly.

Immediately after selecting the spline type, no tangential transition is possible from the preceding motion block to the spline curve.

11.3.1.2 Selecting Akima spline interpolation (#SPLINE ON)

> **i** As of Build **V2.11.2010.02** the command **#SPLINE ON** replaces the command **#SET SPLINE ON**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:
#SPLINE ON

Akima spline interpolation is selected in the last mode selected. The Akima spline curve starts at the last target point programmed. This point is the first vertex of the Akima spline curve.

The command may be programmed in the same statement as the second vertex or in the preceding statement.

Alternatively, Akima spline interpolation can be selected using the G151 command.

11.3.1.3 Deselecting Akima spline interpolation (#SPLINE OFF)

> **i** As of Build **V2.11.2010.02** the command **#SPLINE OFF** replaces the command **#SET SPLINE OFF**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:
#SPLINE OFF

Deselecting Akima spline interpolation is only permitted if at least 3 vertices were programmed or after 2 vertices if all tangents were specified (tangential transition at both ends of the curve or explicit programming of all tangents).

If the command is programmed in one statement together with a position, the respective point is not part of the Akima spline curve.

Alternatively, Akima spline interpolation can be selected using the G150 command.

11.3.1.4 Specifying transition type (spline curve) (#AKIMA TRANS)

> **i** As of Build **V2.11.2010.02** the command **#AKIMA TRANS [...]** replaces the command **#SET AS-PLINE MODE [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#AKIMA TRANS [[**START**=*<ident>* **END**=*<ident>*]]

START=*<ident>* END=*<ident>*	Types of tangential transition between the spline curve and the adjacent (linear or circular) motion blocks. Specification of the transition types is optional. If no specification is made, the AUTO presetting is used. Permitted identifiers:

AUTO	Tangent of the spline curve at the transition is automatically calculated
TANGENTIAL	Tangential transition to the preceding or succeeding block
USER	Tangent of the spline curve at the transition is explicitly specified by the user
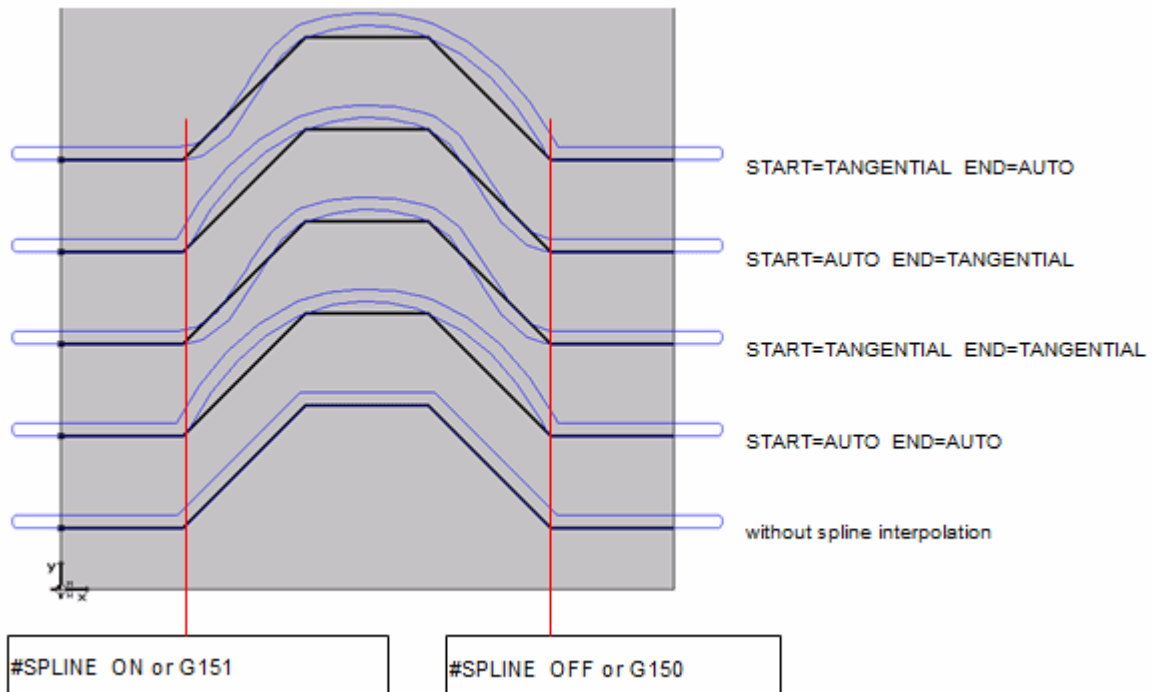
Fig. 93: Examples of combining transition types 1 and 2

If an explicit tangent specification is selected for a certain transition (transition type 3) and if no tangent is programmed for a certain axis, this tangent is automatically determined (transition type 1).

11.3.1.5 Defining the start tangent (#AKIMA STARTVECTOR)

> **i** As of Build **V2.11.2010.02** the command **#AKIMA STARTVECTOR ...** replaces the command **#SET ASPLINE STARTTANG ...**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#AKIMA STARTVECTOR {*<axis_name>*..}

<axis_name>..	Components of the tangent vector, real value

Definition of start tangent Using axis designations, a vector is specified whose direction corresponds to the direction of the tangent - the vector need not therefore be normalised: However, all the components of the main axes must always be specified.

With tracking axes, the vector component is calculated from the ratio of the motion path of the tracking axis to the motion path of the contour:

$$\text{Vector components}_{tracked} = S_{tracked} / S_{path} \text{ where } S_{path} = \sqrt{S_x{}^2 + S_y{}^2 + S_z{}^2}$$

11.3.1.6 Defining the end tangent (#AKIMA ENDVECTOR)

> **i** As of Build **V2.11.2010.02** the command **#AKIMA ENDVECTOR ...** replaces the command **#SET ASPLINE ZIELTANG ...**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

```
Syntax:
```
#AKIMA ENDVECTOR {<*axis_name*>..}

<*axis_name*>**..**   Components of the tangent vector, real value

Definition of target tangent; analogous to the definition of the start tangent.

Defining the end tangent

```
N10 G01 X20 Y0 F1000                    (due to following G151 becomes the first)
                                        (interpolation point of the splice curve)
N20 #AKIMA TRANS[START=USER END=AUTO] (Transition type with spec. of)
                                        (start tangent and auto. determination)
                                        (of end tangent)
N30 # AKIMA STARTVECTOR X1 Y1 Z0        (Pre-set start tangent)
N40 G151                                (Select spline interpolation)
N50 G01 X40 Y20
N60 X60
N70 Y0
N80 X80
N90 Y10                                 (due to following G150 becomes the last)
                                        (interpolation point of the spline-curve)
N100 G150                               (Deselect spline interpolation)
N110 X70
N120 M30
```

The following NC program supplies the same result but uses the second variant to select and deselect the spline interpolation.

```
N10 G01 X20 Y0 F1000
N20 #AKIMA TRANS[START=USER END=AUTO]   (Transition type with spec. of)
                                        (start tangent and auto. determination)
                                        (of end tangent)
N30 # AKIMA STARTVECTOR X1 Y1 Z0        (Pre-set start tangent)
N40 G151 G01 X40 Y20                    (Select spline interpolation)
N50 X60
N60 Y0
N70 X80
N80 Y10
N90 G150 X70                            (Deselect spline interpolation)
N100 M30
```

The following NC program supplies the same result but uses the second variant to select and deselect the spline interpolation.

```
N10 G01 X20 Y0 F1000
N20 #AKIMA TRANS[START=USER END=AUTO]   (Transition type with spec. of)
                                         (start tangent and auto. determination)
                                         (of end tangent)
N30 # AKIMA STARTVECTOR X1 Y1 Z0        (Pre-set start tangent)
N40 G151 G01 X40 Y20                    (Select spline interpolation)
N50 X60
N60 Y0
N70 X80
N80 Y10
N90 G150 X70                            (Deselect spline interpolation)
N100 M30
```

Caution: Block No. 80 contains the target point of the spline.

The program generates the following contour:
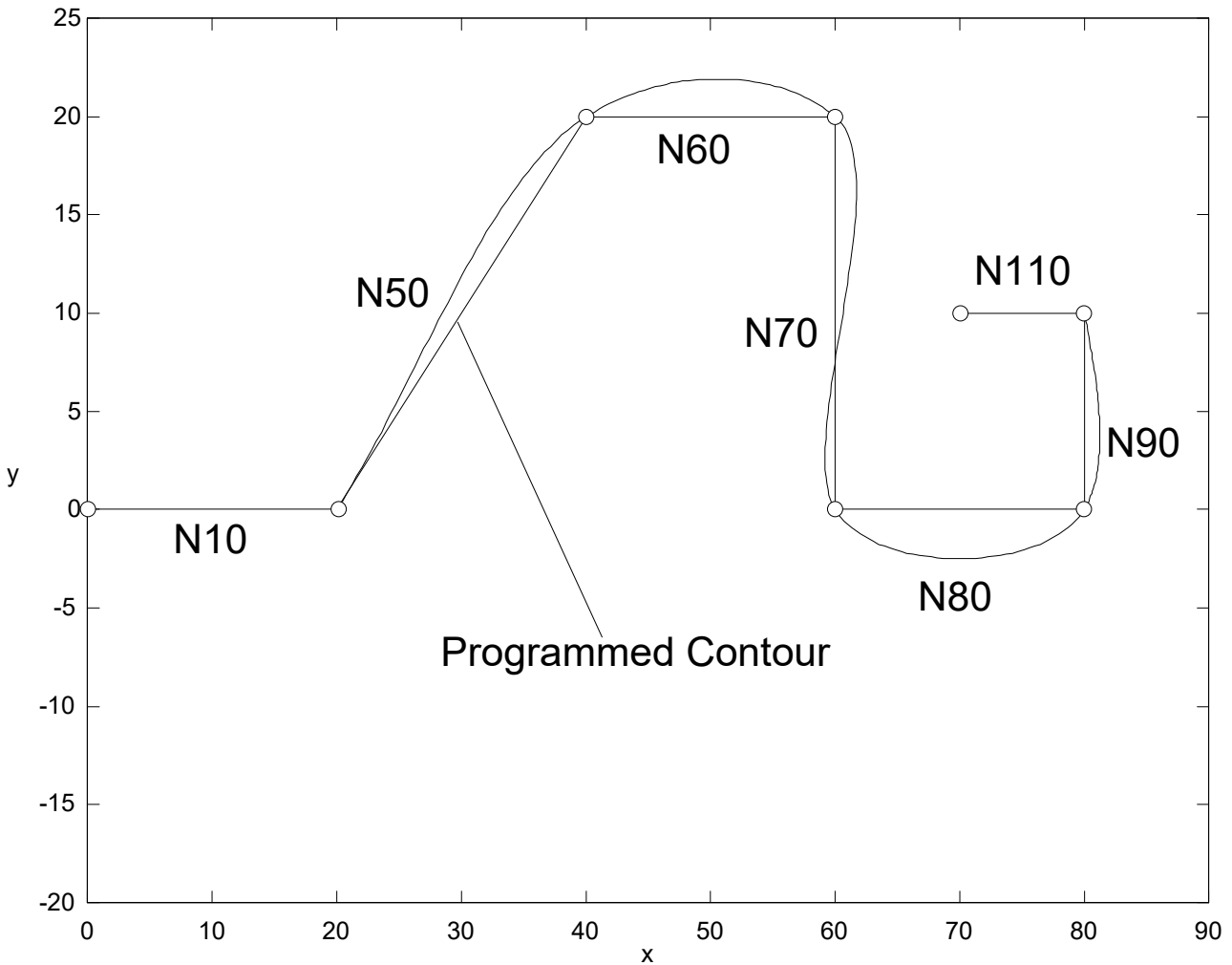
Fig. 94: Contour in the programming example (no. refers to 1st programming example)

It is clearly evident that the curve section corresponding to block N50 has the programmed slope 1 at its start (corresponding to the start of the spline curve). The slope at the spline end (end of block N90) is calculated automatically.

ⓘ   If circular blocks (G02 or G03) are inserted, the spline curve is interrupted before the circular block and when the next linear block arrives, a new spline curve is started automatically. The transitions to and from the circular block are tangential.

The spline curve is also interrupted if a linear block is programmed with stationary main axes and moving tracking axes. Spline interpolation is deselected by the automatic determination of the tangent at curve end. Tracking axes are interpolated linearly until a linear block with moving main axes is programmed. If this is the case, spline interpolation is automatically reselected. The transition to the spline curve occurs tangentially for both the main and the tracking axes.

Other functions (e.g. M functions) may also be programmed between the linear blocks which serve as vertices. However, the count of these functions which were programmed between totally five consecutive vertices is restricted depending on the configuration.

11.3.2 B spline interpolation

i The use of this feature requires a license for the "Spline" extension package. It is not included in the scope of the standard license.

The target points of the programmed linear blocks (G00 and G01) are the control point which are used to generate the B spline curve. It must be ensured that the B spline curve only runs through the control points only at the start and end.

11.3.2.1 Selecting B spline type (#SPLINE TYPE BSPLINE)

i As of Build **V2.11.2010.02** the command **#SPLINE TYPE BSPLINE** replaces the command **#SET SPLINETYPE BSPLINE**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:
#SPLINE TYPE BSPLINE

If the control system only contains the B spline type, this spline type is selected automatically. Nevertheless, it is recommended to specify the spline type explicitly.

11.3.2.2 Selecting B spline interpolation (#SPLINE ON)

i As of Build **V2.11.2010.02** the command **#SPLINE ON** replaces the command **#SET SPLINE ON**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:
#SPLINE ON

The B spline curve starts at the target point last programmed. The command may be programmed in the same statement as the second vertex or in the preceding statement.

Alternatively, B spline interpolation can be selected using the G151 command.

11.3.2.3 Deselecting B spline interpolation (#SPLINE OFF)

> ℹ️ As of Build **V2.11.2010.02** the command **#SPLINE OFF** replaces the command **#SET SPLINE OF**F. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

`Syntax:`

#SPLINE OFF

Deselecting B spline interpolation is only permitted if after least 4 vertices are programmed.

If the command is programmed in one statement together with a position, the respective point is no longer part of the B spline curve.

Alternatively, B spline interpolation can be deselected using the G150 command.

Deselecting spline interpolation

```
N10  #SPLINE TYPE BSPLINE
N20  G01 X0 Y50 Z0 F10000
N30  #SPLINE ON
N40   X3     Y25
N50   X15    Y15
N60   X23    Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #SPLINE OFF
N170 M30
```

The figure below shows the contour resulting from the programming example:
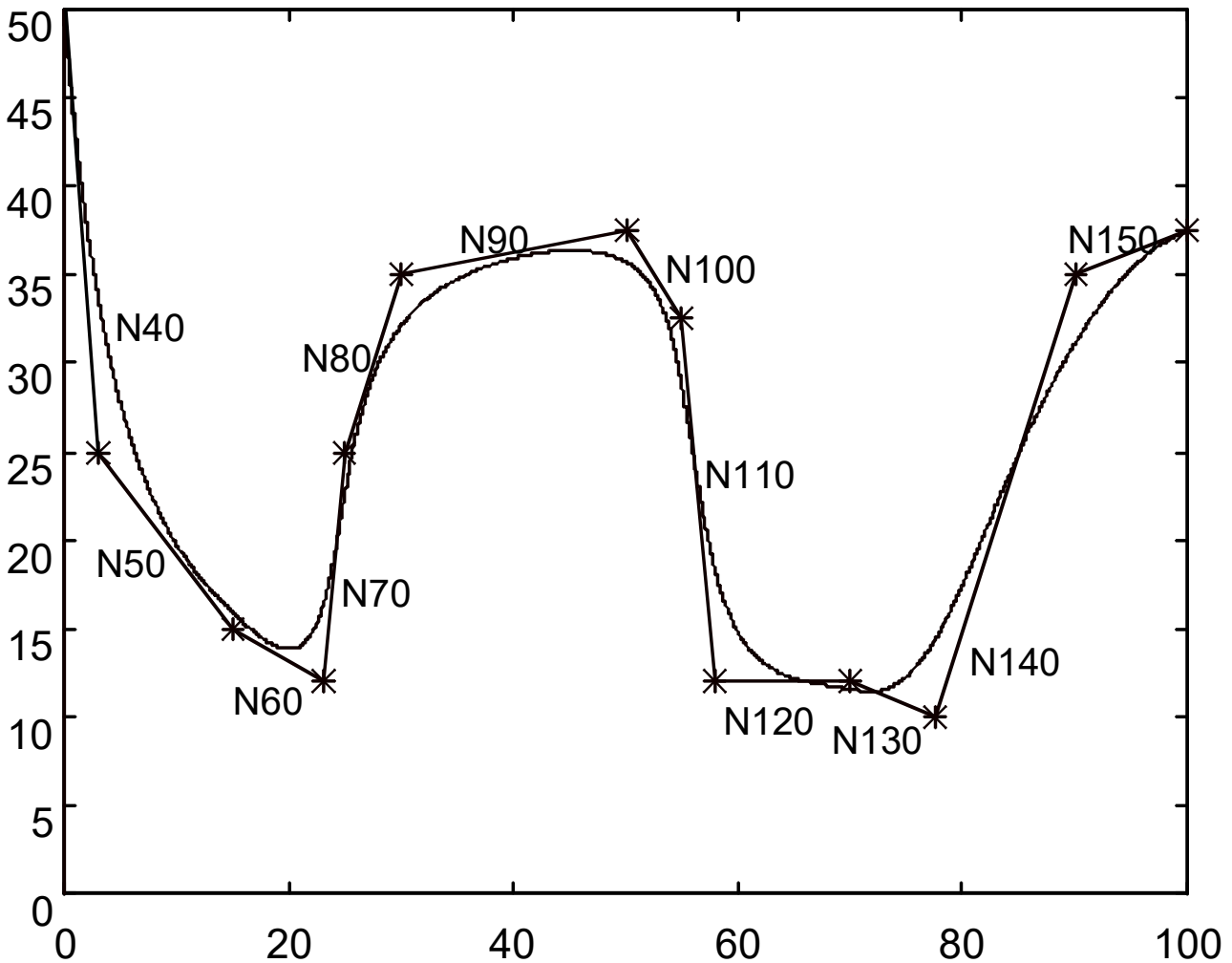
Fig. 95: Contour resulting from programming example

The figure shows the smoothing property of a B spline curve, especially for blocks N120/N130. Moreover, it shows that the curve does not pass through the control points. However, the polygon resulting from connecting the control points provides an estimation of the actual shape of the B spline curve.

> **i** With a B spline curve, it is not possible to specify the tangents at curve ends directly. However, because the tangents of the B spline curve has the same tangents of corresponding motion blocks at the curve end, the tangents may be specified by programming the first and last motion blocks.

11.3.3   PSC programming with OP1 and OP2

`Syntax:`

HSC [**ON** | **OFF**] **[** [**OPMODE**=..] [**CONTERROR**=..] **]**

ON	Enable HSC programming
OFF	Disable HSC programming
OPMODE=..	Set operation mode with:
	1: Insert transition polynomials.
	2: Generate interpolating spline curves.
CONTERROR=..	Define maximum contour error in [mm, inch] with Values >= 0.0: Maximum contour error "ε"

> Is only valid in combination with active OPMODE 1. If
> the parameter is set in OPMODE 2, it only is effective after changing
> to
> OPMODE 1.

If no parameters are programmed when HSC programming is enabled, the following default values are valid:

- OPMODE1
- CONTERROR=0.1 mm

ⓘ The parameters may also be specified in several steps. This means that it is possible to first enable the operation mode ("OPMODE") and HSC processing ("ON") in the first command and then change the maximum contour error ("CONTERROR") in a second command.

An error message is generated if the operation mode is changed while HSC processing is enabled.

11.3.3.1 Available operation modes

OPMODE 1: Insert transition polynomials

In operation mode 1, the blocks are shortened at motion block transitions and transition polynomials are inserted.
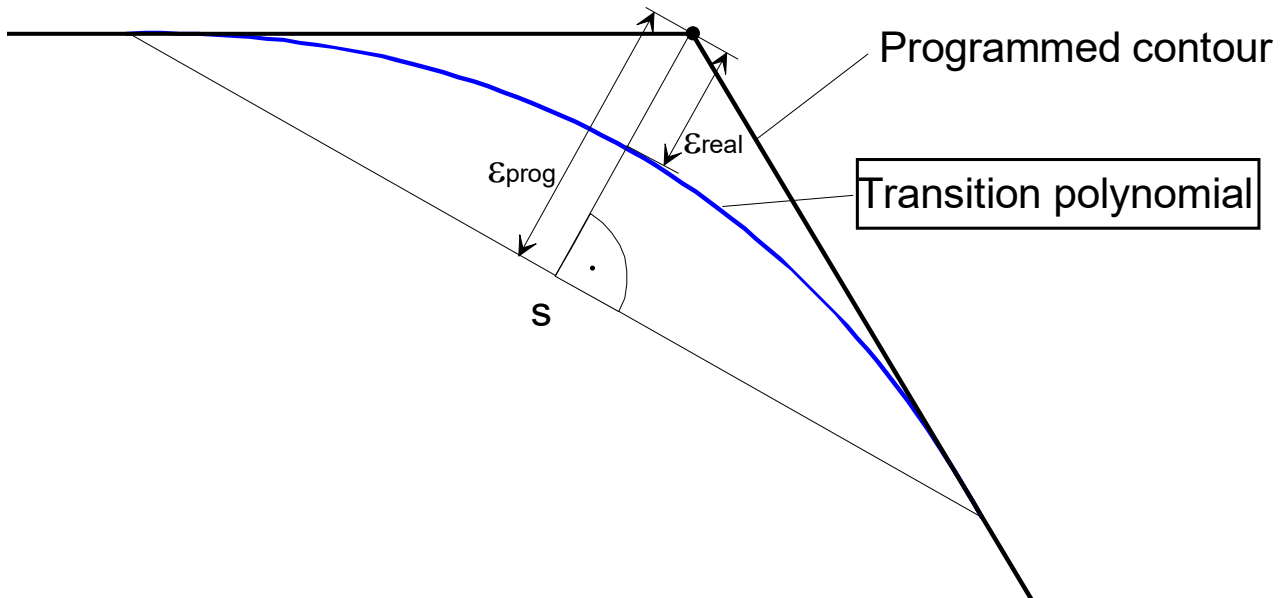
Fig. 96: Insert transition polynomials

The maximum contour error CONTERROR ("ε") is required to generate the polynomial transition. The inserted polynomial reduces the actual contour error (i.e. εreal ≤ εprog).

> **i** No polynomial is inserted in the current block transition if

- the block length of the first block is < 1.1µm or
- the block length of the target block is < 2.2µm or
- the knee angle is > 178°.

OPMODE 2: Generating spline curves for HSC programming

In operation mode 2, spline curves are generated by the specified edge points. At the start of prismatic parts, spline generation is automatically deselected and a tangential transition is executed.

With block transitions in the prismatic range which are detected based on the additional parameters as described in Section PSC programming with OP1 and OP2 [▶ 293] an automatic change takes place to operation mode 1 (OPMODE 1), i.e. transition polynomials are inserted. Contour deviation between vertices is not monitored for inserted spline curves.
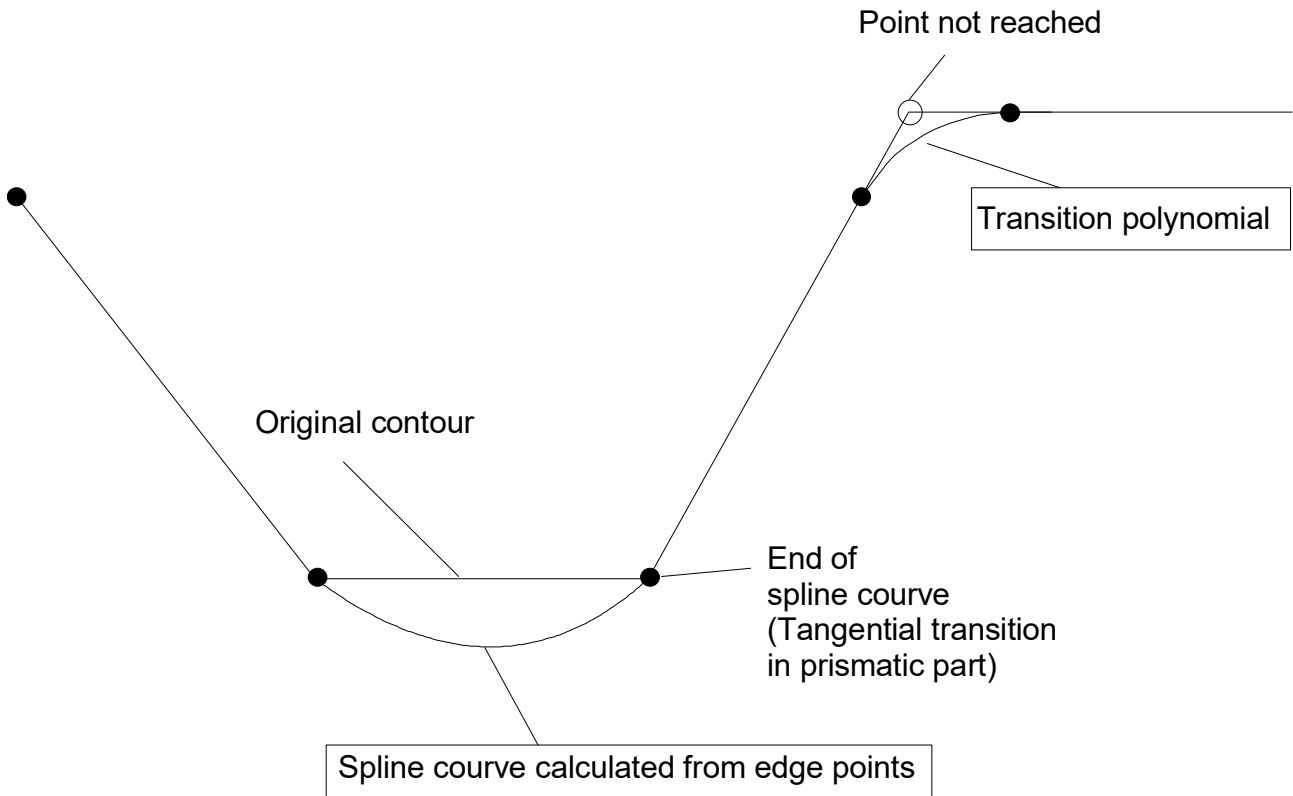
Fig. 97: Generating spline curves for HSC programming

OPMODE 1: Insert transition polynomials

```
...
#HSC[OPMODE 1 CONTERROR 0.01]            (Operation mode 1)
                                         (Max. contour error <= 0.01 mm)
...
#HSC ON               (Enable HSC programming)
...
#HSC OFF              (Disable HSC programming)
...
#HSC ON               (Enable HSC programming)
                      (Previously selected parameters are active)
                      (Operation mode1 (Max. contour error <= 0.01 mm)
...
#HSC OFF              (Disable HSC programming)
...
#HSC ON [OPMODE 2 CONTERROR 0.002]  (Enable free-form programming)
                                    (With operation mode 2)
                                    (Max. contour error <= 0,.02 mm)
...
#HSC[CONTERROR 0.005]  (Change ma. cont. error <= 0.005 mm)
...
#HSC OFF                 (Disable HSC programming)
```

OPMODE 2: Generating spline curves for HSC programming
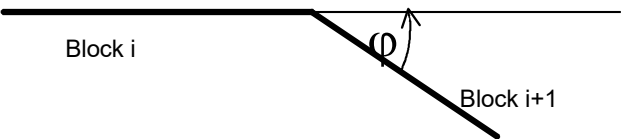
```
...
#HSC[OPMODE 1 CONTERROR 0.01]      (Select operation mode 1)
                                   (Max. contour error <= 0.01 mm)
#HSC ON                            (Enable HSC programming)
...
#HSC[OPMODE 2]                     (Error message!)
                                   (Changing operation mode is)
                                   (not allowed while HSC programming)
                                   (is active)
```

```
...
#HSC[OPMODE 1 CONTERROR 0.01]      (Select operation mode 1)
                                   (Max. contour error <= 0.01 mm)
```

11.3.3.2   Additional parameters

Normally, only the previously described parameters are required. For access to internal settings for operation mode 2, the following additional parameters are available:

`Syntax:`
#HSC [[**COS_PHI_MIN**=..] [**FACT_BLOCK_LEN**=..] **]**

HSC parameters	Type	Valid range (default)	Description
COS_PHI_MIN=..	Real	-1.0...1.0 (0.7)	Block i Block i+1 φ Minimum value for cos(φ). Smaller values lead to deselection of spline curve generation.
FACT_BLOCK_LEN=..	Real	> 0.0 (3.0)	Maximum factor by which the block length may exceed the median block length *. Larger values lead to deselection of spline curve generation. * The median block length results from the block lengths of the previous 5 motion blocks.

12   Additional functions

A complete list of additional functions is contained in the overview of commands in the Appendix under (#..).

Additional functions are a separate group of NC text commands. The permit the programming of specific extensions and technological processes which are covered by DIN/ISO programming. The syntax for additional functions is:

#<*string*> <*spezifische Zusatzsyntax*>

#<*string*>	Plaintext command. Between # and <*string*> no blanks are permitted.
<*specific additional syntax*>	Subsequent command-specific syntax elements which are programmed as additional strings directly or within brackets.

NOTICE
Each # command must be configured alone in a separate NC line. Any exceptions are specifically pointed out.

> **i** If not otherwise displayed explicitly, commas "," and equals signs "=" are **optional** in the specific additional syntax and are only used to improve the legibility of the NC program.
>
> **Example:**
> #STRING [A_VALUE 10 B_VALUE 20] ←→ #STRING [A_VALUE =10, B_VALUE=20]

12.1   Restoring axis configurations and axis couplings

Complex multi-channel machines which use axis exchange operations and synchronous axes (synchronous mode) for machining are aborted by an NC reset if an error occurs. The NC channels are then reinitialised and the channel configurations valid at the time of the abort are lost. To permit clearance of tool axes, it is necessary to restore the axis configuration which was active at the time of abort and comprising the axes found in the channel and the selected axis couplings. This is implemented by the NC commands described below.

12.1.1   Saving a current configuration (#SAVE CONFIG)

```
Syntax:
```
#SAVE CONFIG [[**AX**] [**AXLINK**] **]**

AX	Save the current axis configuration in the NC channel.
AXLINK	Save the current axis couplings selected in the NC channel. Couplings that were previously saved are deleted when AXLINK is programmed although no couplings are active.

The command #SAVE CONFIG is usually always used in the NC program after operations which cause a change in axis configuration or in the activation of axis couplings. For example, after an axis exchange command or after a command to select synchronous mode.

During the save operation, configuration data is sent from the decoder over the NC channel to the interpolator. The interpolator then sends this data directly back to the decoder where it is saved internally. This operation ensures that the current configuration of axes and axis couplings is stored in the decoder synchronously with the current processing state.

Saving a current configuration
```
%main
N10 X0 Y0 Z0
N15 #AX REQUEST [C,4,5] [B,5,6]
```

```
N20 #AX LINK [1, C=X, B=Y]
N25 #AX LINK [2, B=X]
N30 #AX LINK ON [1]
N35 #SAVE CONFIG [AX AXLINK]  ;Save axis configuration X,Y,Z,B,C
                             ;and the active coupling group 1
N.. X.. Y.. Z..
N200 #AX LINK OFF ALL
N210 #AX RELEASE [C]
N220 #SAVE CONFIG [AXLINK] ;Save axis configuration X,Y,Z,B.
                      ;The previously saved axis configuration
                      ;from block N35 is overwritten!
N.. X.. Y.. Z..
N99 M30
```

> **i** A saved configuration remains stored program global. It can only be overwritten by subsequent #SAVE CONFIG commands or deleted by #CLEAR CONFIG.

12.1.2   Loading or restoring a saved configuration (#LOAD CONFIG)

Syntax:

#LOAD CONFIG [[**AX**] [**AXLINK**] **]**

AX	Loading the last axis configuration saved. If no saved configuration is found, an error message is output. Axes which are not found in the NC channel are requested without axis offsets.
AXLINK	Loading the last active axis couplings saved. All axis couplings are summarised, restored and activated in <u>coupling group 1</u>. If no saved axis couplings are found, an error message is output.

The command #LOAD CONFIG is best used after an NC RESET in the clearance program to restore the last configuration saved. The machine operator is responsible for saving the required configuration in the main program by using #SAVE CONFIG and for correctly restoring the required configuration. If both keywords are programmed, the axis configuration is always first restored completely in the NC channel (without "FAST") irrespective of sequence in which the keywords are programmed. Then the last synchronous mode saved is re-activated.

Loading or restoring a saved configuration

Start clearance program after abort of processing and NC RESET:

```
%Clearance
N10 G53
N35 #LOAD CONFIG [AX AXLINK]   ;Restore the saved axis
                             ;configuration and axis couplings
                             ;under coupling group 1
N40# ECS ON          ;Define an effector CS to execute
                  ;withdrawal strategy
N.. X.. Y.. Z..
:
N200 #AX LINK OFF ALL
:
N.. X.. Y.. Z..
N99 M30
```

> **i** When #SAVE CONFIG and #LOAD CONFIG are used in the same NC program, an implicit FLUSH is always executed at the start of #LOAD CONFIG to ensure consistency of the saved configurations in the channel.

```
Nxx #SAVE CONFIG [AX AXLINK]
N..
Nxx #LOAD CONFIG [AX AXLINK]        ;First, implicit FLUSH then
                                    ;restore axis configuration,
                                    ;then restore axis couplings

N.. X.. Y.. Z..
N99 M30
```

12.1.3   Clearing a current configuration (#CLEAR CONFIG)

```
Syntax:
```
#CLEAR CONFIG

The command #CLEAR CONFIG completely clears the last configuration saved. A #LOAD CONFIG programmed directly after causes the error message "No restorable configuration found". This means that #SAVE CONFIG must first be used to save a configuration again before it can be restored with #LOAD CONFIG.

For the machine operator, a #CLEAR CONFIG is always helpful if he wants to prevent access to a configuration that may have been incorrectly saved to different NC programs.

Deleting a saved configuration

Execute clearance program, then delete the saved configuration:

```
%Clearance
N10 G53
N35 #LOAD CONFIG [AX AXLINK]    ;Restore the saved axis
                                ;configuration and axis couplings
                                ;under coupling group 1
N40# ECS ON          ;Define an effector CS to execute
                     ;withdrawal strategy
N.. X.. Y.. Z..
:
N200 #AX LINK OFF ALL
:
N.. X.. Y.. Z..
N.. #CLEAR CONFIG; Clear saved configuration after
                  ;the withdrawal motion ends
N99 M30
```

12.2   Axis exchange commands

This section describes NC commands to

- Request axes
- Release axes and to
- Define an axis configuration

At every program start, the axis configuration specified in the channel parameter list [1] [▶ 786]-5 is restored. Axis exchange commands are active in the currently selected axis group. An NC block may contain several axis requests and/or returns. These operations are executed in semi-parallel state.

---

i   It is not permitted to replace axes for which synchronous or manual mode is active.

When TRC is active, the first two main axes may not be replaced.

Axis exchange commands must be programmed alone in a separate NC line.

Axis exchange commands are only valid for path and tracking axes. Spindle axes are ignored. There are special commands [▶ 641] provided to release and request spindle axes.

As of Version V2.6, the previously available axis exchange commands were revised and extended in functionality and mode of operation. The Section "Standard axis exchange commands" describes the standard syntax available until Version V2.6. This syntax may continue to be used in the future.

The Section "Extended axis exchange commands" describes the new syntax. The syntax is fully downwards compatible with the previous scope of functions. However, the logic switches and additional functions to define axis exchange sequences offer more flexible programming options.

12.2.1 Standard syntax

12.2.1.1 Requesting axes (#CALL AX)

ℹ️ If an axis which is already present in the axis group of the NC channel is requested, no request is triggered for this axis.

The following NC command requests axes from the axis management.

```
Syntax:
```
#CALL AX [<*mode*>] **[**<*axis_name*>,<*axis_number*>,<*axis_index*> {,<*options*>} **]**

{ **[**<*axis_name*>,<*axis_number*>,<*axis_index*> {,<*options*>} **]** }

<mode>	With/without request for axis positions from the interpolator and a position initialisation of the NC channel when axes are replaced.
	---: With request for setpoint values from the interpolator and a position initialisation of the NC channel (default).
	FAST: Without request for setpoint values from the interpolator. Position initialisation of the NC channel.
<axis_name>	The permissible strings for axis designation start with the letters A, B, C, Q, U, V, W, X, Y and Z. The multiple assignment of the same designation for several axes (identification by logical axis number) generates an error message and the NC program is aborted.
<axis_number>	The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known in the axis management. If an unknown logical axis number or several identical logical axis numbers are requested, it results in an error message and the NC program is aborted.
<axis_index>	The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1]. The axis index may not yet be assigned an axis. If a request is made for an index that is already assigned a different axis, it results in an error message and the NC program is aborted.
	0: 1st main axis in the machining plane.
	1: 2nd main axis in the machining plane.
	2: 3rd main axis generally perpendicular to the machining plane.
	3: 1st tracking axis.
	...n_ (n-2) tracking axis.

NOTICE
In order to simplify programming, it is possible to leave the specification of an axis index empty for **tracking axes** (only with #CALL AX...). In this case, the next free axis index after index 3 is then assigned automatically to this tracking axis.
With **main axes** the index must **always** be specified explicitly.
However, it must be noted that the index of a tracking axis is important with regard to various functionalities. For example, all transformation axes must be arranged with no gaps after the main axes in the case of kinematic transformation (RTCP). In such cases, it is then necessary to program the axis index explicitly for the tracking axes.

<options>	Offsets are kept axis-specific. The adoption of different offsets can be controlled for requesting axes using the following keywords:
	---: No adoption of offsets (default)

ALL: Adopt all offsets *

BPV: Adopt reference point offset

PZV: Adopt clamping offset

WZV: Adopt tool offset *

NPV: Adopt zero offset

MOFFS: Adopt measuring offset

SOFFS: Adopt command value/manual mode offset

PSET: Adopt actual value offset

---

i With exception of tool offsets, all other offsets are always attached to the logical axis number an axis release or request (see programming example "Zero_offset" below).

NOTICE

* With #CALL AX, adoption of tool offsets only makes sense when the tool is deselected. As soon as a tool is active in the channel or if it is selected with #CALL AX after replacement, the adopted tool offsets are replaced with the offsets of the current tool.

It is therefore recommended to execute #CALL AX when a tool is deselected.

When a tool is selected, please note that the offsets are always included in the axis calculations depending on the sequence indexed in the tool data.

Requesting axes

```
%Zero_Offset
N010 X200
N015 G54
N015 V.G.NP_AKT.V.X = 11
N016 X0                    (Machine X axis end position is 11)
N020 #PUT AX [X]           (Release axis X with log. axis no. 1)
.....
N130 #CALL AX [X1,1.0,NPV] (Log. axis no. 1 under new name X1)
N140 X1=100                (End position of machine X1 axis is 111)
M30
```

Example:

Assign axis names, logical axis numbers and axis indices at program start:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

%Achstausch1

```
N10 #CALL AX FAST [X1,7,4]   (X1 axis without request for command values)
                             (and output of init. function block)
```

Assign axis names, logical axis numbers and axis indices after the axis request:

Axis identifier	Logical axis number	Axis index
X	**1**	**0**
Y	**2**	**1**
Z	**3**	**2**
		3
X1	**7**	**4**

Programming example continued:

```
:
N100 #CALL AX [Y1,8,6] [C,9, ]  ;Request Y1 and C axis, axis index
                                ;of C axis is determined automatically
```

Assign axis names, logical axis numbers and axis indices after second axis request:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
		5
Y1	8	6

Programming example continued:

```
:
N1000 #CALL AX FAST [Z1,13,5,ALL]  (Adopt all offsets)
N1010 #CALL AX [C1,11,7,NPV MOFFS] (Adopt zero)
                                   (and measuring offsets)
```

Assign axis names, logical axis numbers and axis indices after third axis request:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
Z1	13	5
Y1	8	6
C1	11	7

12.2.1.2   Releasing axes (#PUT AX, #PUT AX ALL)

This NC command returns axes of the axis group of the NC channel to the axis management. It is permitted to return axes which are not or no longer present and this generates no error message.

Syntax:

#PUT AX [*<axis_name>* {**,***<axis_name>* } **]**

<axis_name>	Axis designations may consist of strings with the starting characters A, B, C, Q, U, V, W, X, Y and Z.

This NC command returns all the axes in the axis group of the NC channel to the axis management.

Syntax:

#PUT AX ALL

Assign axis names, logical axis numbers and axis indices at program start:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

Releasing axes

```
N10 #PUT AX [ X, B]    (Release X axis; B axis not present)
                       (No error message is output)
```

Assign axis names, logical axis numbers and axis indices after axis release:

Axis identifier	Logical axis number	Axis index
		0
Y	2	1
Z	3	2

Programming example continued:

```
...
N100 #PUT AX ALL (Release all axes of this group.)
```

Assign axis names, logical axis numbers and axis indices after second axis release:

Axis identifier	Logical axis number	Axis index
		0
		1
		2

12.2.1.3 Definition of an axis configuration (#SET AX)

This NC command defines a new axis configuration to replace the existing axis configuration. Only exactly the axes which are programmed in NC command form the new axis configuration of the NC channel.

`Syntax:`

#SET AX [*<mode>*] **[***<axis_name>,<axis_number>,<axis_index>* {*,<options>*} **]**

{ **[***<axis_name>,<axis_number>,<axis_index>* {*,<options>*} **]** }

<mode>	With/without request for axis positions from the interpolator and a position initialisation of the NC channel when axes are replaced.
	---: With request for setpoint values from the interpolator and a position initialisation of the NC channel (default).
	FAST: Without request for setpoint values from the interpolator. Position initialisation of the NC channel.
<axis_name>	The permissible strings for axis designation start with the letters A, B, C, Q, U, V, W, X, Y and Z. The multiple assignment of the same designation for several axes (identification by logical axis number) generates an error message and the NC program is aborted.
<axis_number>	The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known by the axis management. If an request is made for an unknown logical axis number or several identical logical axis numbers, an error message is output and the NC program is aborted.
<axis_index>	The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1]. The axis index may not yet be assigned an axis. If a request is made for an index that is already assigned a different axis, it results in an error message and the NC program is aborted.
	0: 1st main axis in the machining plane.
	1: 2nd main axis in the machining plane.
	2: 3rd main axis generally perpendicular to the machining plane.
	3: 1st tracking axis.
	...n_ (n-2) tracking axis.

<options>	Offsets are kept axis-specific. The adoption of different offsets can be controlled for requesting axes using the following keywords:

---: No adoption of offsets (default)

ALL: Adopt all offsets *

BPV: Adopt reference point offset

PZV: Adopt clamping offset

WZV: Adopt tool offset *

NPV: Adopt zero offset

MOFFS: Adopt measuring offset

SOFFS: Adopt command value/manual mode offset

PSET: Adopt actual value offset

NOTICE

* When **the tool is selected**, pay attention to the following when adopting tool offsets with #SET AX:

- If axes are only swapped (internal axis replacement) by #SET AX and otherwise no additional axes are specified or requested, all offsets (including the tool offsets) are also replaced and continue to remain active. Specifying keywords to adopt offsets has no effect.
  If a new tool is then selected, the replaced offsets are replaced by the new offsets of the tool.

- As soon as an axis release or an axis request is triggered by #SET AX (external axis exchange), tool offsets are again included in the calculation of the sequence of the axes indexed in the tool data. Therefore, any adopted tool offsets are replaced by the current tool offsets. If the original tool offsets in the corresponding axes should continue to apply, a new tool must be selected with offsets which have been adapted to the new axis arrangement.

You are therefore advised to run #SET AX when a tool is deselected to ensure the correct assignment of tool offsets by the appropriate parameterisation in the data record of a new tool selected.

Example:

	[0]	[1]	[2]	[3]
Index of tool offsets in tool data				
Parameterised tool offsets e.g. for T1	50	0	70	20
Axis configuration at program start	X	Y	Z	---
Included tool offsets after T1 selection	50	0	70	---
"Internal" #SET AX {Z, X, Y}:	**Z**	**X**	**Y**	---
Tool offsets are also swapped or	*70*	*50*	*0*	*---*
"External" #SET AX {Z, X, Y, B}:	**Z**	**X**	**Y**	**B**
Tool offsets are recalculated corresponding to tool T1	*50*	*0*	*70*	*20*

Example:

Assigning axis names, logical axis numbers and axis indices at program start:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

Set axis configuration:

```
(X axis remains in place;)
(Y axis is released;)
(Z axis is re-sorted acc. to Index 4;)
(Y1 and Z1 axis are requested)

%ACHSTAUSCH1
N10 #SET AX [X,1,0][Y1,4,2][Z1,5,3][Z,3,4]
```

Assign axis name, logical axis numbers and axis indices after N10:

Axis identifier	Logical axis number	Axis index
X	1	0
		1
Y1	4	2
Z1	5	3
Z	3	4

12.2.2   Extended syntax

> ℹ **The availability of this function depends on the configuration and concrete scope of version.**

Extended syntax permits the programming of axis exchange sequences with macro definitions (Section macros [▶ 672]) or external variables of the string type (V.E….). This is especially useful for multi-channel machines and systems when static axis groups should be replaced between channels. For example, these axis groups can be defined in macros and used in axis exchange commands.

In addition, the extended syntax permits the internal processing of conflicts by setting so-called logic switches without the output of an error message or a warning. Logic switches are optional and can be programmed additionally in the command. If no logic switches are programmed, standard evaluation applies as before. This means that, if there are plausibility conflicts, the evaluation of axis exchange sequences is aborted and error messages are output.

The logic evaluation is identical for all axis exchange commands and also checks for plausibility **in the programmed axis exchange sequence** and plausibility **to existing axes in the NC channel.**

Logic switches are especially useful for axis exchange sequences which are defined by macros or string variables because any overlaps occurring here and redundant programmings can be resolved internally.

12.2.2.1   Requesting axes (#AX REQUEST)

The following NC command requests axes from the axis management.

```
Syntax:
```
#AX REQUEST [**NAM**, **NBR**, **IDX**] **[**<*axis_exchange_sequence*> {,<*options*>} **]**

{ **[**<*axis exchange sequence*> {,<*options*>} **]** }

NAM, NBR, IDX   Logic switches for handling conflicts:

NAM: Handling redundant axis names

NBR: Handling redundant axis numbers

IDX: Handling redundant axis indices

> ℹ Logic switches can be programmed individually or in combination.

<*axis_exchange_sequence*> consisting of:

<*axis_name*>   The permissible strings for axis designation start with the letters A, B, C, Q, U, V, W, X, Y and Z.

i In case of conflicts within the programmed axis exchange sequence:

Redundant axis names → ERROR, program abort

In case of conflicts with existing axes in the NC channel:

Identical axis names, different axis numbers → ERROR, program abort

If logic switch **NAM** is set, a conflict is resolved as follows:

The axis receives the default axis name from its axis parameter list P-AXIS-00297. The user must ensure a clear definition of the default axis name in the list.

<axis_number>   The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known in the axis management.

i In case of conflicts within the programmed axis exchange sequence:

Redundant axis numbers → ERROR, program abort

In case of conflicts with existing axes in the NC channel:

Axis number already exists in NC channel → WARNING

If the logic switch **NBR** is set, a conflict is resolved as follows:

The axis request is ignored, i.e. it is not executed.

<axis_index>   The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1]. The axis index may not yet be assigned an axis.

0: 1st main axis in the machining plane.

1: 2nd main axis in the machining plane.

2: 3rd main axis generally perpendicular to the machining plane.

3: 1st tracking axis.

...n_ (n-2) tracking axis.

NOTICE
To make programming easier, the axis index input can be left empty for **tacking axes**. In this case, the next free axis index after index 3 is then assigned automatically to this tracking axis.
With **main axes** the index must **always** be specified explicitly.
However, it must be noted that the index of a tracking axis is important with regard to various functionalities. For example, all transformational axes must be arranged after the main axes with no gaps in the case of kinematic transformation (RTCP). In such cases, it is then necessary to program the axis index explicitly for the tracking axes.

i In case of conflicts within the programmed axis exchange sequence:

Redundant axis numbers → ERROR, program abort

In case of conflicts with existing axes in the NC channel:

Axis index is already assigned in the NC channel, different axis → ERROR, program abort.

If the logic switch **IDX** is set, a conflict is resolved as follows:

The next free index in the axis configuration of the NC channel is automatically determined for the axis.

<options>   Offsets are kept axis-specific. The adoption of different offsets can be controlled for requesting axes using the following keywords:

---: No adoption of offsets (default)

ALL: Adopt all offsets *

BPV: Adopt reference point offset

PZV: Adopt clamping offset

WZV: Adopt tool offset *

NPV: Adopt zero offset

MOFFS: Adopt measuring offset

SOFFS: Adopt command value/manual mode offset

PSET: Adopt actual value offset

---

i With the exception of tool offsets, all other offsets are always attached to the logical axis number on axis release or request.

NOTICE

* In the case of #AX REQUEST, it only makes sense to adopt tool offsets when the tool is deselected. As soon as a tool is active in the channel or if it is selected with #AX REQUEST after replacement, the adopted tool offsets are replaced with the offsets of the current tool.

It is therefore recommended to execute #AX REQUEST when a tool is deselected.

When a tool is selected, please note that the offsets are always included in the axis calculations depending on the sequence indexed in the tool data.

Using standard functionality:

Assigning axis names, logical axis numbers and axis indices at program start:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

Requesting axes

```
N10 #AX REQUEST [X1,7,4]    ;Request X1 axis
```

Assign axis names, logical axis numbers and axis indices after an axis request:

Axis identifier	Logical axis number	Axis index
X	**1**	**0**
Y	**2**	**1**
Z	**3**	**2**
		3
X1	**7**	**4**

Programming example continued:

```
N100 #AX REQUEST  [Y1,8,6] [C,9, ] ;Request Y1 and C axis,
                                   ; C axis is set automatically
                                   ; to index 3
```

Assigning axis names, logical axis numbers and axis indices after second axis request:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
		5
Y1	8	6

Programming example continued:

```
N1000 #AX REQUEST FAST [Z1,13,5, ALL]  ;Adopt all offsets
N1010 #AX REQUEST [C1,11,7, NPV MOFFS] ;Adopt zero
                                 ;offset and measuring offset
```

Assign axis names, logical axis numbers and axis indices after third axis request:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
Z1	13	5
Y1	8	6
C1	11	7

Example 2:

Use the additional syntax (logic switches):

Assigning axis names, logical axis numbers and axis indices at program start in channel 1:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

Assigning axis names, logical axis numbers and axis indices at program start in channel 2:

Axis identifier	Logical axis number	Axis index
X	7	0
Y	8	1
Z	9	2

In channel 1: Request X and Y axes from channel 2 (logical axis numbers 7 and 8).

```
N10  #AX REQUEST NAM [X,7,3] [Y,8,4]   ;Request X/Y axes
```

Due to the logic switch NAM, the new additional axes X and Y in channel 1 are assigned their default names in the axis lists (e.g. X2 and Y2).

Assigning axis names, logical axis numbers and axis indices after axis requests:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
X2	7	3
Y2	8	4

Programming example continued:

In channel 1: Request Z axis from channel 2 (logical axis number 9).

```
N100 #AX REQUEST NAM IDX [Z,9,2]  ;Request Z axis
```

Due to the logic switches NAM and IDX, the new additional Z axis in channel 1 is adopted in the next free index (e.g. index 5) together with its default name in the axis list (e.g. Z2).

Assigning axis names, logical axis numbers and axis indices after second axis request:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
X2	7	3
Y2	8	4
Z2	9	5

Example 3:

Use the additional syntax (logic switches and axis exchange sequences in string format):

Assigning axis names, logical axis numbers and axis indices at program start in channel 1:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

Assigning axis names, logical axis numbers and axis indices at program start in channel 2:

Axis identifier	Logical axis number	Axis index
X_1	7	0
Y	8	1
Z	9	2

| A | 10 | 3 |
| B | 11 | 4 |

In channel 1: Request X_1/ Y and B axis from channel 2 (logical axis numbers 7, 8 and 11). The axis exchange sequence is stored in a macro.

```
N05 "ACHSEN_KANAL2" = "[X_1,7,0] [Y,8,1] [B,11,2]"
:
N10 #AX REQUEST NAM IDX "ACHSEN_KANAL2" ;Request axes
```

Due to the logic switches NAM and IDX, the new additional axes in channel 1 are adopted correctly.

Assigning axis names, logical axis numbers and axis indices after axis requests:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
X_1	7	3
Y2	8	4
B	11	5

12.2.2.2 Releasing axes (#AX RELEASE, #AX RELEASE ALL)

These NC command axes return axes in the axis group of the NC channel to the axis management. It is permitted to return axes which are not or no longer present and this generates no error message.

Syntax:

#AX RELEASE [*<axis_name>* {**,***<axis_name>* } **]**

| *<axis_name>* | Axis names of the axes currently present in NC channel. |

The logic switch NBR switches over evaluation from axis names to logical axis numbers (e.g. if axis names are unknown at the time of release).

Syntax:

#AX RELEASE [NBR] [*<ax_nr>* {**,***<ax_nr>* } **]**

| *<ax_nr>* | Logical axis number of the axis. |

This NC command returns all the axes in the axis group of the NC channel to the axis management.

Syntax:

#AX RELEASE ALL

Assigning axis names, logical axis numbers and axis indices at program start:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2
A	4	3
B	5	4

Releasing axes

```
N10 #AX RELEASE[X, A]      ;Release X/A axes
```

Assigning axis names, logical axis numbers and axis indices after axis release:

Axis identifier	Logical axis number	Axis index
Y	2	1
Z	3	2
B	5	4

Programming example continued:

```
...
N100 #AX RELEASE NBR[2]     ;Release Y axis
```

Assigning axis names, logical axis numbers and axis indices after second axis release:

Axis identifier	Logical axis number	Axis index
Z	3	2
B	5	4

Programming example continued:

```
...
N100 #AX RELEASE ALL       ;Release all existing axes of this
                           ;channel
```

Assigning axis names, logical axis numbers and axis indices after the third axis release:

Axis identifier	Logical axis number	Axis index

12.2.2.3   Definition of an axis configuration (#AX DEF)

This NC command defines a new axis configuration to replace the existing axis configuration. **Only exactly the axes which are programmed in NC command form the new axis configuration of the NC channel.**

Syntax:

#AX DEF [**NAM**, **NBR**, **IDX**] **[**<*axis_exchange_sequence*> {,<*options*>} **]**

{ **[**<*axis exchange sequence*> {,<*options*>} **] }**

NAM, NBR, IDX	Logic switches for handling conflicts:
	NAM: Handling redundant axis names
	NBR: Handling redundant axis numbers
	IDX: Handling redundant axis indices

ℹ️ Logic switches can be programmed individually or in combination.

<*axis_exchange_sequence*> consisting of:

<*axis_name*>	The permissible strings for axis designation start with the letters A, B, C, Q, U, V, W, X, Y and Z.

ℹ️ In case of conflicts within the programmed axis exchange sequence:

Redundant axis names → ERROR, program abort

In case of conflicts with existing axes in the NC channel:

Identical axis names, different axis numbers → ERROR, program abort

If logic switch **NAM** is set, a conflict is resolved as follows:

The axis receives the default axis name from its axis parameter list P-AXIS-00297. The user must ensure a clear definition of the default axis name in the list.

<*axis_number*>	The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known in the axis management.

ℹ️ In case of conflicts within the programmed axis exchange sequence:

Redundant axis numbers → ERROR, program abort

In case of conflicts with existing axes in the NC channel:

Axis number already exists in NC channel → WARNING

If logic switch **NBR** is set, a conflict is resolved as follows:

The axis request is ignored, i.e. it is not executed.

<*axis_index*>	The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1]. The axis index may not yet be assigned an axis.
	0: 1st main axis in the machining plane.
	1: 2nd main axis in the machining plane.
	2: 3rd main axis generally perpendicular to the machining plane.
	3: 1st tracking axis.
	...n_ (n-2) tracking axis.

NOTICE
To make programming easier, the axis index input can be left empty for tacking axes. In this case, the next free axis index after index 3 is then assigned automatically to this tracking axis. However, it must be noted that the index of a tracking axis is important with regard to various functionalities. For example, all transformational axes must be arranged after the main axes with no gaps in the case of kinematic transformation (RTCP). In such cases, it is then necessary to program the axis index explicitly for the tracking axes.

ℹ️ In case of conflicts within the programmed axis exchange sequence:

Redundant axis indices → ERROR, program abort

In case of conflicts with existing axes in the NC channel:

Axis index is already assigned in the NC channel, different axis → ERROR, program abort.

If logic switch **IDX** is set, a conflict is resolved as follows:

The next free index in the axis configuration of the NC channel is automatically determined for the axis.

<options>   Offsets are kept axis-specific. The adoption of different offsets can be controlled for requesting axes using the following keywords:

---: No adoption of offsets (default)

ALL: Adopt all offsets *

BPV: Adopt reference point offset

PZV: Adopt clamping offset

WZV: Adopt tool offset *

NPV: Adopt zero offset

MOFFS: Adopt measuring offset

SOFFS: Adopt command value/manual mode offset

PSET: Adopt actual value offset

NOTICE
* When **the tool is selected**, please note the following when adopting tool offsets with #AX DEF:

- If axes are only swapped (internal axis exchange) by #AX DEF and otherwise no additional axes are specified or requested, all offsets (including tool offsets) are also replaced and continue to remain active. Specifying keywords to adopt offsets has no effect.
  If a new tool is then selected, the replaced offsets are replaced by the new offsets of the tool.

- As soon as an axis release or an axis request is triggered by #AX DEF (external axis exchange), the tool offsets are re-adopted in the calculation of the sequence of axes indexed in the tool data. Therefore, any adopted tool offsets are replaced by the current tool offsets. If the original tool offsets in the corresponding axes should continue to apply, a new tool must be selected with offsets which have been adapted to the new axis arrangement.

You are therefore advised to run #AX DEF when the tool is deselected and to ensure the correct assignment of tool offsets in the data record of a newly selected tool by appropriate parameterisation.

	[0]	[1]	[2]	[3]
Index of tool offsets in tool data				
Parameterised tool offsets e.g. for T1	50	0	70	20
Axis configuration at program start	X	Y	Z	---
Included tool offsets after T1 selection	50	0	70	---
"Internal" #AX DEF {Z, X, Y}:	**Z**	**X**	**Y**	---
Tool offsets are also swapped or	*70*	*50*	*0*	*---*
"External" #AX DEF {Z, X, Y, B}:	**Z**	**X**	**Y**	**B**
Tool offsets are again recalculated depending on T1	*50*	*0*	*70*	*20*

12.2.2.4   Load the default axis configuration (#AX DEF DEFAULT)

If DEFAULT is specified, the default configuration contained in the channel parameter list [1] [▶ 786]-5 can be restored. A combination with logic switches and additional axes requests is also permitted.

```
Syntax:
```
#AX DEF DEFAULT

or

#AX DEF DEFAULT [NAM, NBR, IDX] { **[**<axis exchange sequence> {,<options>} **]** }

Assigning axis names, logical axis numbers and axis indices at program start:

The starting position of the following NC program is the default axis configuration listed in the table.

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

```
;Set axis configuration
;X axis remains in its place,
;Y axis is released since this axis is not contained
;               in the new definition
;Z axis is re-sorted acc. to Index 4,
;Y1 and Z1 axis are requested

%ACHSTAUSCH
N10 #AX DEF [X,1,0][Y1,4,2][Z1,5,3][Z,3,4]
:
```

Assigning axis names, logical axis numbers and axis indices after N10:

Axis identifier	Logical axis number	Axis index
X	1	0
		1
Y1	4	2
Z1	5	3
Z	3	4

Restoring default axis configuration:

```
%ACHSTAUSCH
N10 #AX DEF DEFAULT
:
```

Assigning axis names, logical axis numbers and axis indices after N10:

Axis identifier	Logical axis number	Axis index
X	1	0
Y	2	1
Z	3	2

12.3 Dwell time

```
Syntax:
```

#TIME *<time>*   non-modal

See Section Dwell time (G04), (#TIME) [▶ 80].

12.4 Flushing NC channel (#FLUSH, #FLUSH CONTINUE, #FLUSH WAIT)

The single NC blocks of an NC program are read in continuously by an interpreter module, converted to an internal representation and output to the NC channel for further processing (program decoding) In the NC channel, the data is then read in by further modules (tool radius compensation, contour preparation, etc.), edited and passed through to the interpolator.

This processing step has a buffering effect on the NC channel with the result that program decoding is always ahead of actual execution of commands in the interpolator.

The buffering effect on the NC channel can then lead to a delay in motion block supply to the interpolator and stop axis motions.

The commands **#FLUSH** and **#FLUSH CONTINUE** deselect the buffering effect of the NC channel. All buffered NC blocks are immediately processed and executed in their programmed sequence.

These commands are especially helpful to bridge the time gap of large non-motion relevant NC program parts between two motion blocks and to avoid needless stops in axis motion.

NOTICE
The commands for flushing the NC channel (#FLUSH, #FLUSH CONTINUE, #FLUSH WAIT) can **not** be used with active TRC, polynomial contouring or HSC mode.

> **i** As of Build V2.10.1503.08, the commands for flushing the NC channel can be used with active polynomial contouring. In this case, contouring is suppressed in the motion block before and after a command to flush the NC channel.

```
Syntax:
```

#FLUSH

The **#FLUSH** statement forces all NC blocks (control commands, path motions) decoded before the command to be passed through to the interpolator. However, decoding of the part program is continued without interruption.

The interpolator is generally stopped for a short time at the end of the last path motion before **#FLUSH** even if the next motion block generated by the continuation of part program decoding is already present.

Flush NC channel

A WHILE loop containing many non-motion relevant NC blocks is programmed between the motion blocks N10 and N210 (e.g. parameter calculations, variable accesses etc.). As a result, the motion block N10 in NC channel and the preparation of motion block N210 are delayed. **#FLUSH** forces the NC channel to pass through all motion blocks to the interpolator before the WHILE loop. It can then start with executing the path motion up to and including N10.

```
N05  G01 F1000 G90 X150
N10  X200
N20  #FLUSH
N30  $WHILE
N40    P1 = [P2 * P3] + V.E...
N50    V.L....
...
N200 $ENDWHILE
N210 X250
```

```
...
M30
```
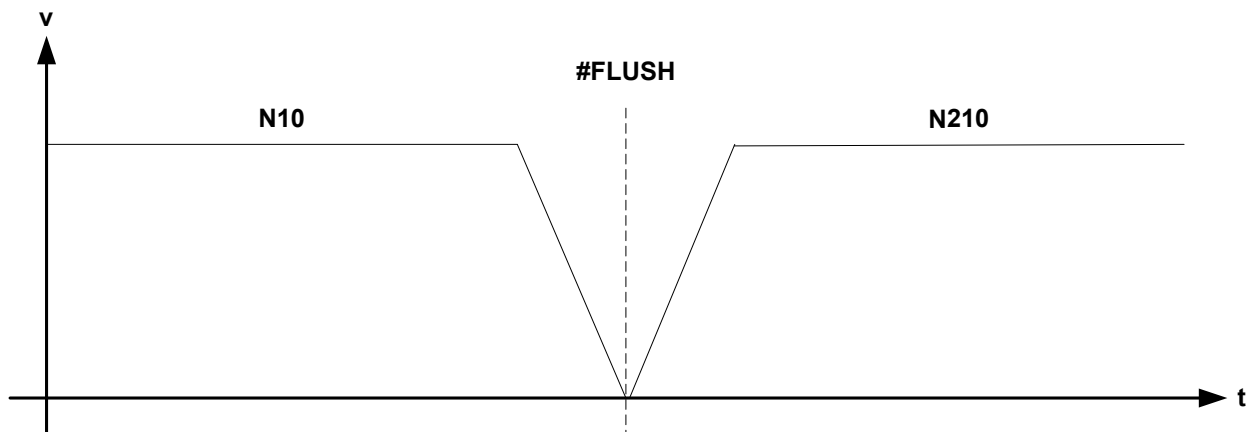

Fig. 98: Mode of operation of #FLUSH between 2 motion blocks

```
Syntax:
```
#FLUSH CONTINUE

A stop at block end of the last motion block can be avoided by adding CONTINUE. If the next motion block is already present in the interpolator, the motion is executed without any interruption. Only if no further motion block is present does the interpolator go to wait state.

#Flush Continue

A WHILE loop containing many non-motion relevant NC blocks is programmed between the motion blocks N10 and N210 (e.g. parameter calculations, variable accesses etc.). As a result, the motion block N10 in NC channel and the preparation of motion block N210 are delayed. **#FLUSH CONTINUE** forces the NC channel to pass through all motion blocks to the interpolator before the WHILE loop. It can then start with executing the path motion up to and including N10. If the next motion block N210 is present in the interpolator before the end of N10, the motion is continued without interruption.

```
N05  G01 F1000 G90 X150
N10  X200
N20  #FLUSH CONTINUE
N30  $WHILE ...
N40    P1 = [P2 * P3] + V.E...
N50    V.L....
...
N200 $ENDWHILE
N210 X250
...
M30
```

v

#FLUSH CONTINUE

| **N10** | **N210** |

t
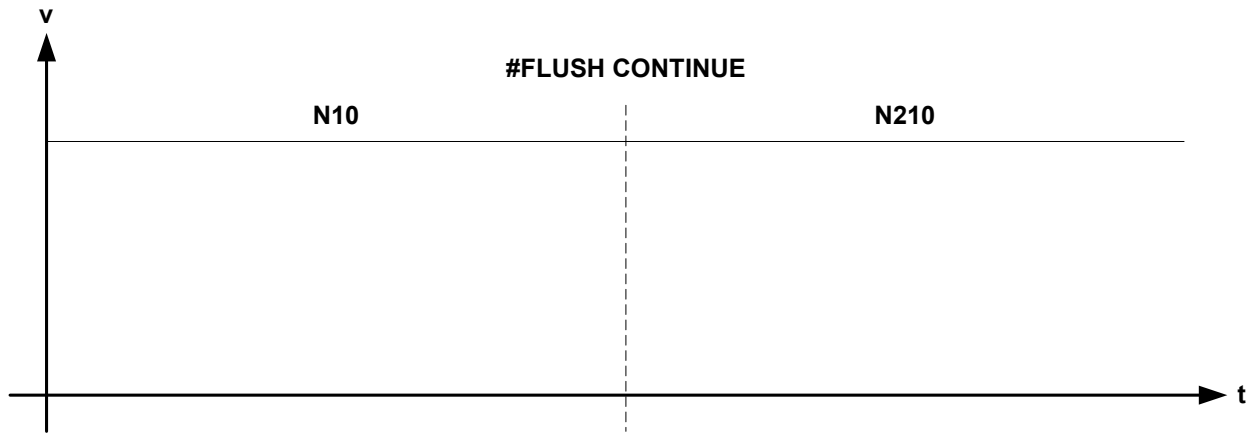
Fig. 99: Mode of operation of #FLUSH CONTINUE between 2 motion blocks

```
Syntax:
```
#FLUSH WAIT

By contrast to **#FLUSH** and **#FLUSH CONTINUE**, program decoding is also interrupted by the command **#FLUSH WAIT**. All NC blocks before this command are output to the NC channel and executed in interpolator The NC channel is then "flushed". All axes are stopped and the interpolator and program decoder have the same state, i.e. they are synchronised. Only when this state is reached does program decoding continue automatically.

NOTICE
If synchronised **or unsynchronised (MOS)** M/H functions or #MSG commands were output previously to the PLC, the controller waits until they are adopted and synchronised by the PLC. Program decoding is interrupted until this state is reached.

#Flush Wait

#FLUSH WAIT interrupts program decoding and forces the NC channel to pass through all motion blocks to the interpolator before the WHILE loop. When the interpolator has executed block N10 and position X200 is reached, program decoding continues.

```
N05  G01 F1000 G90 X150
N10  X200
N20  #FLUSH WAIT
N30  $WHILE ...
N40    P1 = [P2 * P3] + V.E...
N50    V.L....
...
N200 $ENDWHILE
N210 X250
...
M30
```
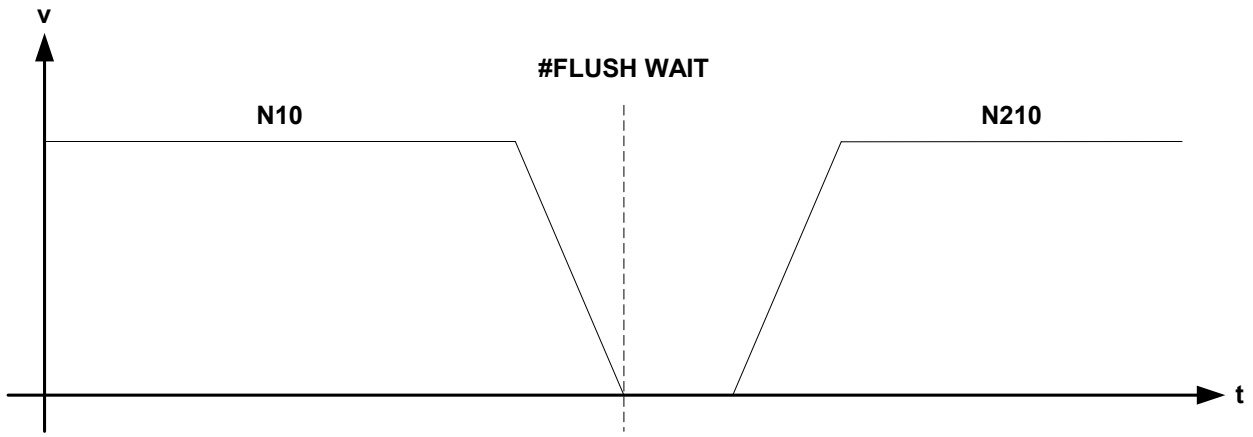
Fig. 100: Mode of operation of #FLUSH WAIT between 2 motion blocks

12.5 Cross-block comments (#COMMENT BEGIN/END)

Syntax:

#COMMENT BEGIN	Start of comment block
:	Disabled area. All lines are ignored by the
:	NC kernel without evaluation.
#COMMENT END	End of comment block

Both commands must be programmed in a separate NC line, i.e. further NC commands in the same block are not permissible. An exception is a line-by-line comment that must be bracketed "(" and ")".

Comment blocks can be defined inside or outside an NC program. However, they are restricted to the current file since calls from and returns from global subroutines included in the comments are not evaluated.

Application-specific nesting of comments can be enabled.

Cross-block comments

:	
#COMMENT BEGIN	Start of comment block 1
...	
#COMMENT BEGIN	Start of comment block 2
...	
#COMMENT END	End of comment block 2
...	
#COMMENT END	End of comment block 1
:	
Nesting disabled:	
:	
#COMMENT BEGIN	Start of comment block 1
...	
#COMMENT BEGIN	Start of comment block 2
...	
#COMMENT END	End of comment block 1 and 2
:	

In the following cases, the NC kernel generates error messages:

- "End of File" is read in within a comment block.
- #COMMENT END is read in without a previously programmed #COMMENT BEGIN.
- After commend commands, further NC commands are programmed in the **same** block.

12.6   Waiting for event (#WAIT FOR)

```
Syntax:
```

#WAIT FOR *<wait_condition>*

This command stops the decoding of the NC program until the arithmetical expression is fulfilled (TRUE or $\geq$0.5).

Waiting for an event

N10 **#WAIT FOR** V.E.EXT1 == 5	(Decoding of NC program is stopped at this) (point until the value of the external variable) (is 5.)
....	
N100 **#WAIT FOR** V.E.EXT2	(Decoding of the NC program is stopped at this (Point until the value of the (external variable $\geq$ is 0.5.)
....	

NOTICE

The controller look-ahead mechanisms may result is the state that a certain number of motion-relevant NC blocks programmed before #WAIT FOR are retained in the NC channel. If the retained motion blocks are relevant for generating the event itself, a deadlock is caused and execution of the NC program is stopped without any apparent reason.

By programming #FLUSH or #FLUSH CONTINUE directly before #WAIT FOR, the execution of all preceding motion blocks is forced and a deadlock is avoided.

Waiting for an event

....	
Nxx G01 X.. Y.. Z.. F..	
Nxx G01 X.. Y.. Z.. F..	
Nxx G01 X.. Y.. Z.. F..	
Nxx G01 X.. Y.. Z.. F..	
Nxx G01 X.. Y.. Z.. F..	
Nxx G01 X.. Y.. Z.. F..	
Nxx G01 X.. Y.. Z.. F..	
Nxx **#FLUSH CONTINUE**	;Forced execution of all buffered motion blocks
Nxx **#WAIT FOR** V.E.XX>123	;Forced execution of all buffered motion blocks
....	

12.7 Adapting minimum radius for tangential feed ((#TANGFEED))

> ℹ️ As of Build **V2.11.2010.02** the command **#TANGFEED [...]** replaces the command **#SET TANG-FEED RMIN [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

This command supplements the tangential adaptation of feed programmed with G10/G11 [▶ 529]. The minimum radius is considered in circular blocks containing the identifier for feed adaptation. The feed is only corrected if the programmed circular radius is greater than or equal to the minimum radius.

```
Syntax:
```
#TANGFEED [RMIN=.. **]**

RMIN=..   Minimum contour radius Rmin in [mm, inch] up to which the tangential feedrate is corrected.
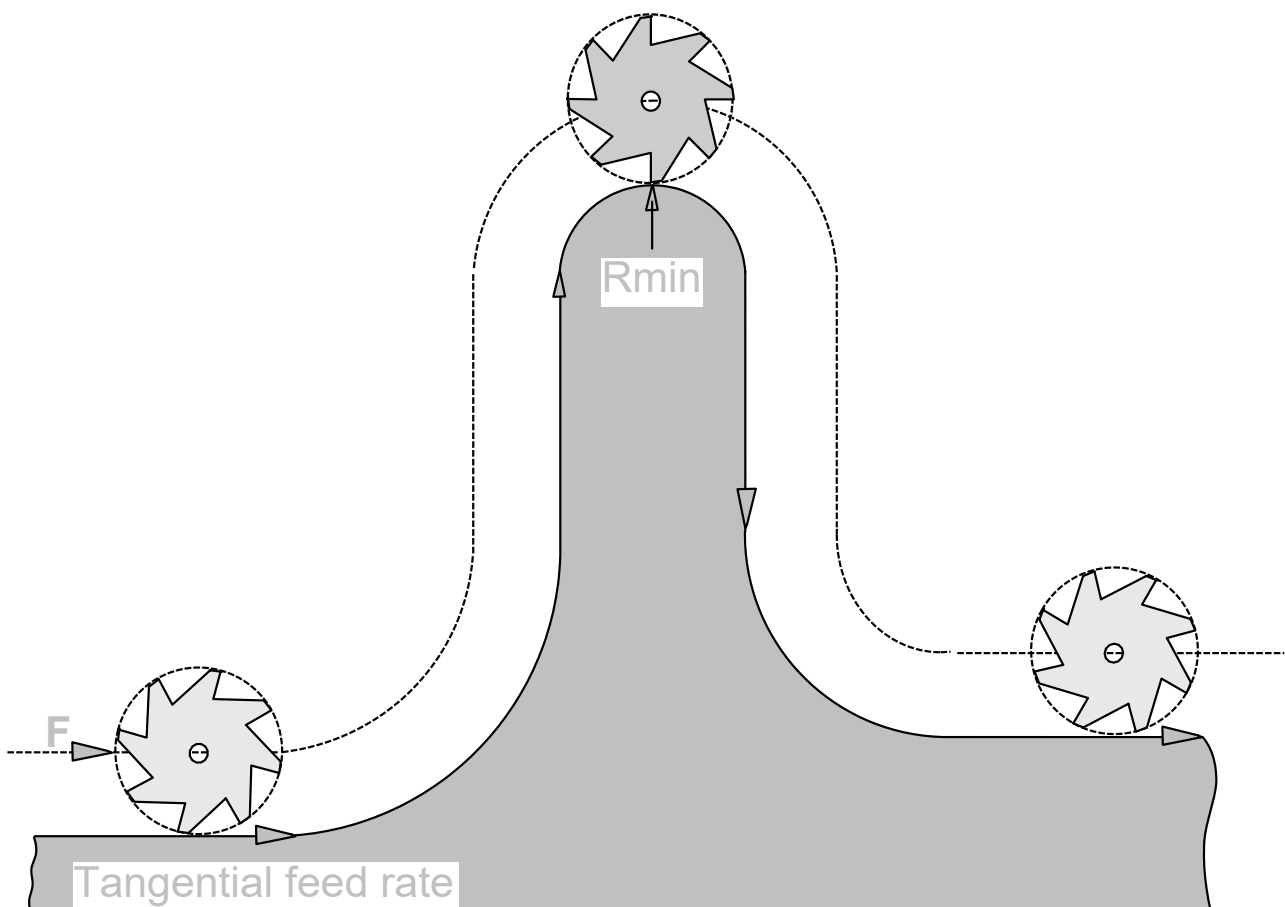
Fig. 101: Programming the tangential feedrate.

- A new programmed minimum radius is taken into account with the next relevant circular motion block.

- It is optional to specify the minimum radius. Tangential feed adaptation with G11 [▶ 529] is also executed if #TANGFEED [RMIN…] is not programmed or the radius is assigned the value zero.

Adapting minimum radius for tangential feed

```
(Contour with TRC and tangential feed adaptation)
N10   V.G.WZR=5
N20 #TANGFEED [RMIN=3]          (Define minimum radius rmin = 3mm)
N30 G41 G01 X0 Y20 G11 F600     (Select TRC and feed adaptation)
N40 X20 Y20
N50 G03 X40 R10                 (Feed adaptation)
N60 G01 Y40
```

```
N70 G02 X44 R2               (No feed adaptation → rprg < rmin)
N80 G01 Y20
N20 #TANGFEED [RMIN=6]       (Define new minimum radius rmin = 6mm)
N90 G03 X54 R5               (No feed adaptation → rprg < rmin)
N100 G01 Y50
G40 Y20 G11 G10      (Deselect TRC and feed adaptation)
N120 X0 Y0
```

12.8 Suppressing offsets (#SUPPRESS OFFSETS)

In combination with a motion block, this command causes an execution of the programmed axis positions without consideration of the active offsets.

All offsets in the NC block are suppressed if a particular offset type is not specified.

> **i** As of Build V2.11.2032.07 it is also possible to selected which active offsets (zero points, PSET, measurements etc.) are to be suppressed in the NC block.

Offsets due to active kinematic and/or Cartesian transformations (CS, ACS, ROTATION) are not suppressed by the command. To suppress all kinds of offsets, the **command** #MCS ON/OFF [▶ 709] must be used as an alternative.

```
Syntax:
```

| **#SUPPRESS OFFSETS** [**[ZERO ADD_ZERO PSET CLAMP TOOL MEASURE MANUAL]**] | non- |
| *<axis_name>*.. {*<axis_name>*..} | modal |

ZERO	Zero offsets
ADD_ZERO	Additive zero offsets and reference point offsets
PSET	Position presets
CLAMP	Clamping offsets
TOOL	Tool offsets
MEASURE	Measuring offsets
MANUAL	Manual mode offsets
<axis_name>..	Axis positions which are moved without offsets.

Suppressing offsets

```
%suppress_offsets
;Define zero offsets for G54
N05 V.G.NP[1].V.X=11
N10 V.G.NP[1].V.Y=22

;Select zero offsets X11 Y22
N15 G54

;Select and define additive zero offsets X10 Y20
N20 G92 X10 Y20

N25 X100 Y150                         ;Position X=121 Y=192

;Suppress all active offsets
N30 #SUPPRESS OFFSETS X50 Y100        ;Position X=50 Y=100

;Suppress additive offsets
N35 #SUPPRESS OFFSETS [ADD_ZERO] X50 Y100    ;Position X=61 Y=122

N40 X200 Y250                         ;Position X=221 Y=292
N99 M30
```

12.9 Settings for measurement

12.9.1 Switching measurement type (#MEAS MODE)

Using the following command, the measurement type can be defined by the NC program.

```
Syntax:
```
#MEAS MODE [**[**<*meas_type*>**]**

<meas_type>	Measurement type as described in Section <u>Switching the measurement type [▶ 87]</u>.

Programming #MEAS MODE without parameters selects the default measurement type which is specified in the channel parameter P-CHAN-00057.

Switching measurement type

```
N10 #MEAS MODE[3]       (Measurement type 3)
N20 G100 X150           (Measurement motion block)
..
N100 #MEAS MODE         (Default measurement type)
..
M30
```

12.9.2   Extended programming (#MEAS, #MEAS DEFAULT)

Alternatively to the #MEAS MODE command, the following command offers additional measuring parameters. The selected parameter settings remain valid until program end. When the program is restarted, the default settings in the configuration lists are again valid. To be able to change the measurement parameters of an axis, it must be identified as a measurement axis (i.e. the axis parameter P-AXIS-00118 must be set to 1:

```
Syntax:
```
#MEAS [[**TYPE=..**] [**ERR_NO_SIGNAL=..**] [[**SIMU_OFFSET=..**] | [**TRIGGER**]] |

   [{**AX**=*<axis_name>* | **AXNR=..**} [**SIGNAL**=*<ident>*] [**EDGE**=*<ident>*] [**INPUT=..**]

   [**G107** | **G108**]] **]**

TYPE=..	New measurement type as described in Section Measuring functions [▶ 87]. This measurement type is valid until the next change or until program end.
ERR_NO_SIGNAL=..	Reaction to undetected probe signal:
	0: no error message
	1: Error message when measurement run is deselected (default)
SIMU_OFFSET=..	This keyword is only effective in particular for a measuring simulation in connection with the axis parameter P-AXIS-00112=4. The value in [mm, inch] offsets the simulated default measuring point relative to the path motion based on the programmed target points.
	The **measurement type 2** shifts the default measuring point in a positive or negative direction by SIMU_OFFSET. Any additional offset by the axis parameter [2] [▶ 786]P-AXIS-00114 is not considered here.
TRIGGER	Triggering a programmed probe signal. Only used in conjunction with the global block edge banding function G107/G108 [▶ 103]. Is only effective if P-CHAN-00257 is active.

ℹ️ SIMU_OFFSET and TRIGGER are exclusive and may not be programmed in combination with the other axis-
specific keywords.

AX=*<axis_name>*	Name of the axis whose measurement parameters are to be changed. The axis must be configured as a measurement axis.
AXNR=..	Logical number of the axis whose measurement parameters are to be changed. The axis must be configured as a measurement axis, positive integer
SIGNAL=*<ident>*	Name of the probe signal source used for the measurement (see P-AXIS-00516). Valid identifiers:
	PLC:Probe signal via PLC
	DRIVE: Probe signal via drive position latch
	FIXED_STOP: Probe signal by moving to fixed stop
	PLC_FIRST_EVENT: Probe signal by PLC. Measurement run terminates when one axis receives the measuring event
	PLC_EXT_LATCH_CONTROL: Measure with measuring interval for external hardware (see [HLI//Measuring with measuring interval for external hardware])
EDGE=*<ident>*	Relevant measurement edge, (see P-AXIS-00518). Valid identifiers:
	POS: Positive (rising) measurement edge
	NEG: Negative (falling) measurement edge
INPUT=..	Name of the drive measurement input used for the measurement (see P-AXIS-00517).
	For probe signal DRIVE:
	1: 1st measurement input
	2: 2nd measurement input

For probe signal PLC_EXT_LATCH_CONTROL:

1 .. 255: Channel number of measurement input of the external measuring hardware

G107	Deselect edge banding function for this axis, i.e. no measured value is latched for this axis with edge banding.
G108	Select edge banding function for this axis. The precondition is that the "edge banding " function must be active for the axis in the axis parameters (see P-AXIS-00098).

The #MEAS command provides the following extensions for a modal measurement run (over several motion blocks) in conjunction with the edge banding function (measurement type 8).

Syntax:

#MEAS [**ON** | **OFF**] **[** [<measurement_parameter>] **]**

ON	Select measurement run for measurement type 8 All set or programmed axes are then measured in the following motion blocks.
OFF	Deselect measurement run for measurement type 8

Syntax:

#MEAS DEFAULT [**[** {**AX**=<*axis_name*> | **AXNR**=..} **]**

DEFAULT	Reset the *axis-specific* (AX, AXNR) parameter settings (SIGNAL, EDGE, INPUT, G107/G108) changed by the #MEAS command. The measurement settings of the axis parameter list are effective again.

ℹ The measurement edge (EDGE) and measurement input (INPUT) cannot be changed for SERCOS drives with position latch in the drive (SIGNAL=DRIVE) because this also requires parameter changes in the drive for this purpose.

Setting measurement parameters:

Selecting a different measurement type:

```
N100 #MEAS [TYPE=2]
```

Setting measurement positions for the measurement simulation for measurement type 2:

```
N10  G00 X0 Y0
N20  #MEAS [TYPE=2 SIMU_OFFSET=10]
N30  G100 X200
```

Target point of measurement (Default measuring point)
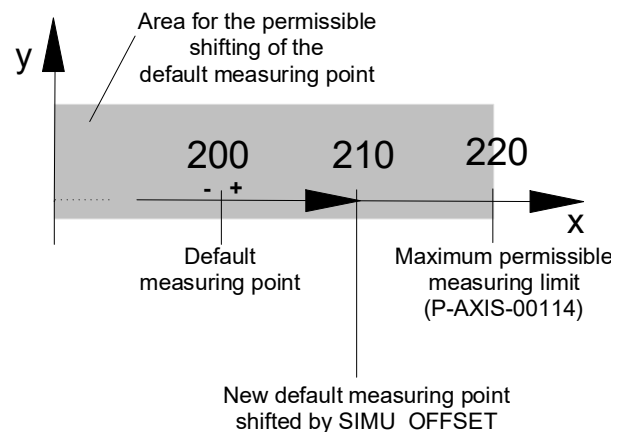
For all other measurement types only an offset in negative direction (opposite to the path motion direction) is possible.

```
N10   G00  X0 Y0
N20   #MEAS [TYPE=1 SIMU_OFFSET=-10]
N30   G100 X200
```

Target point of
measurement
(Default measuring point)

Area for the permissible
shifting of the
default measuring point

190   200

y

x

Default
measuring point

New default measuring point
shifted by SIMU_OFFSET
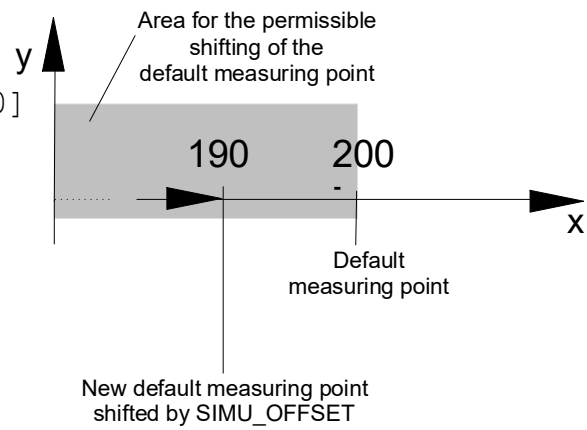
Activate measurement by moving to fixed stop for X and Y axes:

```
N100 #MEAS [AX=X AX=Y SIGNAL=FIXED_STOP]
```

Activate probe signal via PLC to negative edge:

```
N100 #MEAS [AXNR=1 SIGNAL=PLC EDGE=NEG]
```

Deactivate edge banding function for Y and Z axes:

```
N100 #MEAS [AX=Y AX=Z G107]
```

Restore the measurement settings in the axis parameters for all path axes:

```
N100 #MEAS DEFAULT
```

Restore the measurement settings in the axis parameters for X axis:

```
N100 #MEAS DEFAULT [AX=X]
```

Modal measurement run with all measurement axes:

```
N5 #MEAS [TYPE=8]
N10 G01 X100 Y100 F1000
N20 G01 Z200
N30 G01 X200 Y200
N40 #MEAS OFF
```

Modal measurement run with measurement in X and Y axes, probe signal via PLC, leading edge, no error message if probe signal not detected

```
N5 #MEAS ON [TYPE=8 AX=X AX=Z SIGNAL=PLC EDGE=POS ERR_NO_SIGNAL=0]
N10 G01 X100 Y100 Z10 F1000
N20 G01 X200 Y150 Z25
N30 #MEAS OFF
```

12.10 Selecting position preset (#PSET)

A new position preset is selected by:

`Syntax:`

#PSET *<axis_name><expr>* {*<axis_name><expr>*}   non-modal

<axis_name><expr>   New actual position of the axis in [mm, inch]

The actual positions are programmed as absolute. No path motion is linked to this command.

Axes which have no new position preset programmed retain their current position preset, i.e the corresponding offset does not change for the axis concerned.

The #PSET command can be repeated as often as required.

The offset of the coordinate system caused by the #PSET command is not valid across programs.

The offset of the coordinate system caused by the #PSET command deselected by the homing function (G74).

12.10.1 Deselecting position preset (#PRESET)

`Syntax:`

#PRESET {*<axis_name> <value>*}   non-modal

<axis_name><value>   The axis position preset is reset. The coordinate value in [mm, inch] is only required for syntax reasons, otherwise it is irrelevant.

If #PRESET is programmed without specifying an axis, the position preset is deselected in all axes.

> ℹ If tool radius compensation, mirroring or diameter programming are selected, #PSET or #PRESET may not be programmed.
>
> ("non modal" only applies to the commands #PSET and #PRESET. The position preset offset itself remains valid of course until it is again selected or deselected by #PRESET.)

Deselecting position preset

```
N10 X50 Y10 Z0
N20 #PSET X20 Y20        (Set position for X and Y)
N30 X10
N40 #PRESET X10          (Reset X position; value 10 not relevant)
N50 X30
N60 #PRESET              (Reset all axes)
N70 M30
```

The figure below shows the position of the x axis in machine coordinates after execution of a particular NC block:

after block N10:

50/50   x

after block N20:

50/20   x

after block N30:

40/10   x

after block N40:

40/10   x

after block N50:

30/30   x
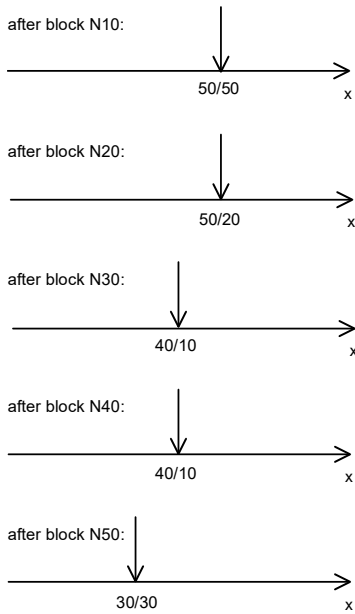
Fig. 102: Positions of the X axis in machine coordinates / programmed coordinates.

(In this example, no other coordinate transformations are selected).

```
N10 X50 Y10 Z0
N20 #PSET X20 Y20        (Set position for X and Y)
```

12.11   Synchronous operation

12.11.1   Programming axis couplings (#SET AX LINK, #AX LINK)

The following NC command can be defined in an NC program for axis couplings:

`Syntax:`
#SET AX LINK [*<coupling_group>***,** *<slave>=<master>* {**,***<slave>=<master>*} **]**

or alternatively

#AX LINK [**NBR**] **[** *<coupling_group>***,** *<slave>=<master>* {**,***<slave>=<master>*} **]**

<coupling_group>	Number of the coupling group [1]
<Slave>	Designation or logical axis number of the slave axis of coupling pair i [2]
<master>	Designation or logical axis number of the master axis of coupling pair i [2]
NBR	Evaluation can be changed from logical axes names to axes numbers with the logic switch NBR. The axis couplings must then be defined with logical axis numbers. The axes need not be present in NC channel. Their presence in the NC channel is only checked when the coupling group is activated.

> **i**   At least one master-slave coupling pair must be defined for each coupling group.
>
> The coupling of spindles is described in greater detail in section Synchronous spindle mode [▶ 651].

General handling and method of operation:

- The definitions of coupling groups apply program global. This means that they can be selected again in a subsequent NC program.
- The coupling group with coupling number '0' cannot be defined in the NC program. It is defined in channel parameter [1] [▶ 786]-2.
- A disabled coupling group can be modified at any time. The previously defined couplings are overwritten.
- An enabled coupling group may not be modified.
- Recursive axis couplings is not permissible. A master axis in a coupling group cannot be a slave axis at the same time and vice versa. A slave axis in one coupling group cannot be a master axis in another coupling group at the same time and vice versa.
- Master and slave axis of a coupling pair may not be identical.
- A main axis may not be a slave axis.
- A slave axis can only be assigned to one master axis, but a master axis can have several slave axes.
- When synchronous mode is activated, a check is made whether all master and slave axes exist in the axis group of the NC channel.
- Master and slave axes must be of the same axis type and be used in the same axis mode.
- The NC command used for programming the coupling group must be a single instruction in the NC block.
- The number of the coupling group can also be programmed via mathematical expressions. The result must be a positive integer.
- A slave axis may not be an active tracking axis at the same time (#CAXTRACK).

1 ... [Max. number of coupling groups–1], see [6] [▶ 786] -2.11

Max. number of coupling pairs, see [6] [▶ 786]-2.12

● The corresponding axis coupling must be enabled explicitly using **#ENABLE AX LINK**
[▶ 345].

Programming axis couplings

`#SET AX LINK[1, Z2=Z]`	`;Z2 is linked as slave to main axis Z`
`#SET AX LINK[2, Y2=Y, X2=X]`	`;Coupling of Y2 with Y and X2 with X`
`#SET AX LINK[3, X1=X, X2=X, X3=X]`	`;Coupling of X1, X2, X3 as slaves to the same master axis X`
`:`	
`or alternatively`	
`#AX LINK[1, Z2=Z]`	
`#AX LINK `**`NBR`**`[2, `**`8 = 2, 9 = 3`**`]`	**`;Coupling via logical axis numbers`**

12.11.2 Extended programming of axis couplings ("SOFT-GANTRY") (#SET AX LINK, #AX LINK)

> **ℹ** **The availability of this function depends on the configuration and on the version scope.**

Path axes can also be used as so-called gantry axes. Contrary to normal synchronous mode, additional position deviation monitoring mechanisms are active and specific error reactions apply. Due to the machine structure, the axes are permanently linked to each other in mechanical (and static) gantry operation and defined by the machine configuration. After controller start-up it not possible to make a dynamic change in the gantry coupling (see figure below).

In addition to path axes, spindles can also be run in synchronous mode. A detailed description is contained in section Spindle synchronous mode [▶ 651].
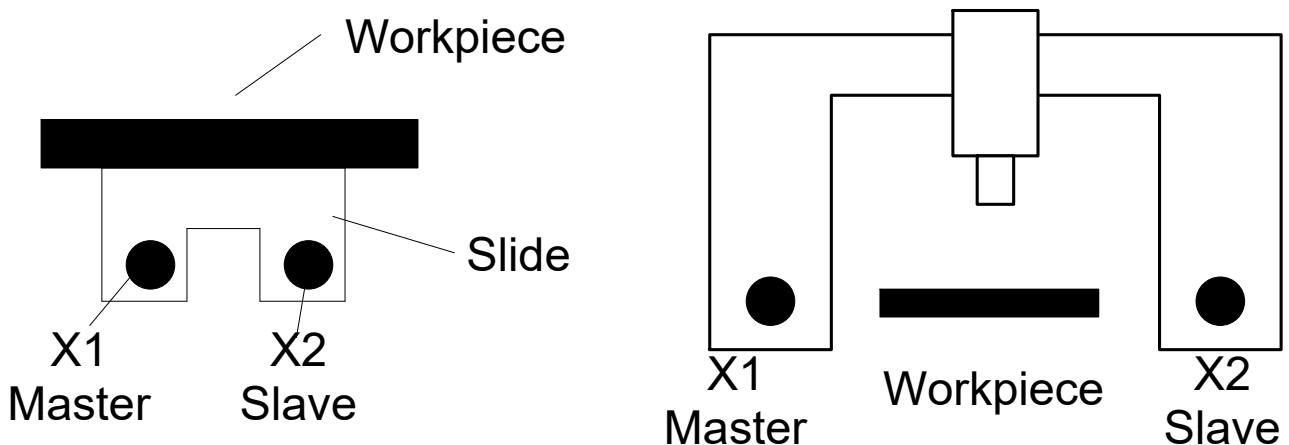
Fig. 103: Mechanical gantry operation

Machines that do not permit any mechanical gantry operation because of their basic structure, e.g. milling machines with two independent slides, can be operated in gantry mode by programming. For example, this is necessary when slides must be linked to one another for clamping and machining large workpieces (see Figure below)
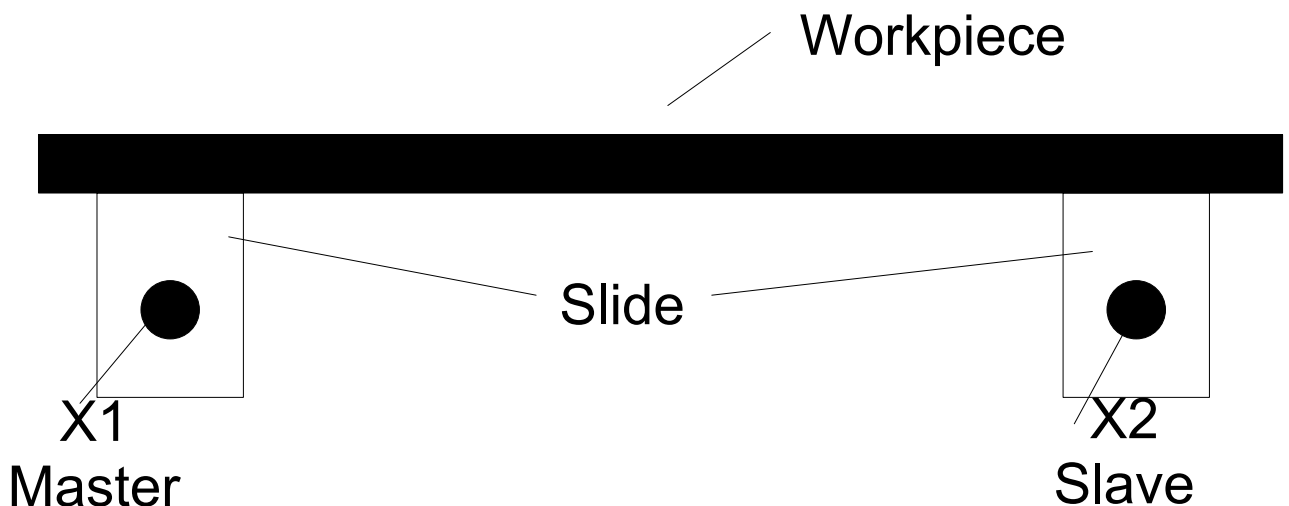
Fig. 104: Programmable gantry operation ("soft" gantry)

The #SET AX LINK and #AX LINK commands described in section Programming axis couplings [▶ 339] are available in an extended syntax for this purpose:

`Syntax:`

#SET AX LINK [*<coupling_group>*, **[** *<slave>=<master>*,**G** [,*<limit_1>*, *limit_2>*] **]**

                 {, **[** *<slave>=<master>*,**G** [,*<limit_1>*, *limit_2>*] **]** } **]**

or alternatively

#AX LINK [**NBR**] **[** *<coupling_group>*, **[** *<slave>=<master>*,**G** [,*<limit_1>*, *limit_2>*] **]**

                 {, **[** *<slave>=<master>*,**G** [,*<limit_1>*, *limit_2>*] **]** } **]**

<coupling_group>	Number of the coupling group [1]
<Slave>	Designation or logical axis number of the slave axis of coupling pair i [2]
<master>	Designation or logical axis number of the master axis of coupling pair i [2]
NBR	Evaluation can be changed from logical axes names to axes numbers with the logic switch NBR. The axis couplings must then be defined with logical axis numbers. The axes need not be present in NC channel. Their availability in NC channel is checked only at activation of the coupling group!
G	Keyword for "gantry coupling"

(1) 1 ... [Max. number of coupling groups–1], see [6] [▶ 786] -2.11

(2) Max. number of coupling pairs, see [6] [▶ 786] -2.12

When gantry coupling is used, the following values serve to monitor in two stages the permissible position difference of the gantry axes. Specified in [mm]. Positive real number:

<limit_1>	First monitoring limit in [mm, inch]. If this limit is exceeded, the motion is aborted and the controller assumes the error state. In the default case, the position difference is cancelled during RESET. Depending on the specific application, deviating motion can still be realised.
<limit_2>	Second monitoring limit in [mm, inch]. An error that cannot be RESET is output if this limit is exceeded. The controller must be switched off and the position difference must be eliminated manually.

> **i** If the monitoring limits are not programmed, the defaults apply from the axis parameter data records P-AXIS-00072 and P-AXIS-00071 of the slave axis.

Additions to general handling and mode of operation

- Gantry coupling takes place at precisely the positions where the axes are at the time when coupling is selected. There is no need to specify an offset in the NC command because the offset is calculated internally in the position controller via the nominal positions.

- The dynamic data of the slave axis is taken into account in the contouring motion.

- For safety reasons, a coupling which is still active on RESET or at program end is implicitly restored the next time the program is started. This reaction is parameterisable (P-CHAN-00104, P-CHAN-00105).

Extended programming of axis couplings (SOFT-GANTRY)

`N10 #SET AX LINK[1, [Y2=Y1,G,0.01,0.25]]`	Gantry coupling of Y1 as master axis and Y2 as slave axis. First limit is 10 µm, second limit is 250 µm.
`N20 #SET AX LINK[2, [Y2=Y1,G]]`	Gantry coupling of Y1 (master) and Y2 (slave). The monitoring limits of the axis parameter data block of the Y2 axis apply.
`N30 #SET AX LINK [3,[Y2=Y1]]`	Standard coupling of Y2 with Y1. No gantry operation.
or alternatively	
`N10 #AX LINK[1, [Y2=Y1,G,0.01,0.25]]`	
`N20 #AX LINK `**`NBR`**`[2, [`**`8=2`**`,G]]`	Gantry coupling via **log. axis numbers**

The parallel machining of workpieces with a symmetrical or scaled contour can also be programmed by means of an extended syntax of the #SET AX LINK and #AX LINK commands. Position differences are not monitored in these modes (mirroring or scaling).

`Syntax:`

#SET AX LINK [*<coupling_group>*, **[** *<slave>=<master>,<nominator>, <denominator>* **]**

                    {, [*<slave>=<master>,<hominator>,* **<denominator** *]* **}** **]**

or alternatively

#SET AX LINK [NBR] **[** *<coupling_group>*, **[** *<slave>=<master>,<nominator>, <denominator>* **]**

                    {, [*<slave>=<master>,<hominator>,* **<denominator** *]* **}** **]**

<coupling_group>	Number of the coupling group [1]
<Slave>	Designation or logical axis number of the slave axis of coupling pair i [2]
<master>	Designation or logical axis number of the master axis of coupling pair i [2]
NBR	Evaluation can be changed from logical axes names to axes numbers with the logic switch NBR. The axis couplings must then be defined with logical axis numbers. The axes need not be present in NC channel. Their availability in NC channel is checked only at activation of the coupling group!
<numerator>, <denominator>	Integers. Their purpose is to calculate a coupling factor between the master and slave axes.
	The following applies to the **resulting** coupling factor:
	-1 : mirror coupling
	1 : standard coupling; equivalent to the previous syntax
	0 : output of an error message

NOTICE
Factors that result in pure scaling (factor ≠ 1) or in scaling with simultaneous mirroring (factor ≠ -1) are currently not permitted. A warning is output and the coupling is handled as a standard coupling. This means that only **coupling factors 1 and –1** are permissible (see examples).

BECKHOFF

(1) 1 ... [Max. number of coupling groups–1], see [6] [▶ 786] -2.11

(2) Max. number of coupling pairs, see [6] [▶ 786] -2.12

Code	Description
N10 #SET AX LINK[1, [Y2 = Y1,1,-1]]	Mirroring coupling (factor -1)
N20 #SET AX LINK[1, [Y2 = Y1,-1,1]]	Mirroring coupling (factor -1)
N30 #SET AX LINK[1, [Y2 = Y1,-2,2]]	Mirroring coupling (factor -1)
N40 #SET AX LINK[1, [Y2 = Y1,1,1]]	Standard coupling
N50 #SET AX LINK[1, [Y2 = Y1,2,2]]	Standard coupling
N60 #SET AX LINK[1, [Y2 = Y1,0,1]]	Error message, program is aborted
N70 #SET AX LINK[1, [Y2 = Y1,1,0]]	Error message, program is aborted
N80 #SET AX LINK[1, [Y2 = Y1,1,2]]	Warning (factor 0.5), standard cpl.
N90 #SET AX LINK[1, [Y2 = Y1,2,3]]	Warning (Factor 0.666), Standard cpl.
N100 #SET AX LINK[1, [Y2 = Y1,3,2]]	Warning (factor 1.5), standard cpl.
N110 #SET AX LINK[1, [Y2 = Y1,-1,2]]	Warning (factor -0.5), standard cpl.
N120 #SET AX LINK[1, [Y2 = Y1,-3,2]]	Warning (factor -1.5), standard cpl.
or alternatively	
N40 #AX LINK[1, [Y2 = Y1,1,1]]	Standard coupling
N50 #AX LINK NBR[1, [8 = 2,2,2]]	Standard coupling via log. axis numbers

12.11.3   Enabling/disabling axis couplings (#ENABLE AX LINK, #DISABLE AX LINK)

A coupling group can be activated by the following NC command:

Syntax:

#ENABLE AX LINK [<coupling_group>]

or

#ENABLE AX LINK (coupling_group 0, defined in the channel parameters)

or alternatively

#AX LINK ON [<coupling_group>]

or

#AX LINK ON (coupling_group 0, defined in the channel parameters)

An active coupling group can be disabled by the following NC commands:

#DISABLE AX LINK [<coupling_group >]

or

#DISABLE AX LINK (Deselect the last enabled coupling group)

or alternatively

#AX LINK OFF [<coupling_group>]

or

#AX LINK OFF (Deselect the last enabled coupling group)
#AX LINK OFF ALL (Deselect all actrve coupling groups)

Handling and method of operation:

- No coupling group is active after start-up in the initial position of the NC kernel. Activation of axis coupling begins with programming in the NC program and ends, if not cancelled, when the program ends (M30, M02). If active axis couplings are to remain effective for the next program, i.e. program global, a specific channel parameter P-CHAN-00105 must be set.
- Multiple coupling groups can be enabled at the same time.
- Unassigned coupling groups cannot be activated. A coupling group is considered assigned if at least one valid master-slave coupling pair was defined.
- The NC command must be a single instruction in the NC block.
- The number of the coupling group can also be programmed via mathematical expressions.
- WRK must not be selected when synchronous operation is selected or cancelled.
- Manual mode with parallel interpolation (G201) may not be active for the slave axes when synchronous mode is selected
- Positions of slave axes may not be addressed in the NC program when synchronous mode is active.

Axis designations used: Master axis system X, Y, Z, CSlave axis system Y_S, Z_S, C_S

```
(Initialisation program)
%L UP_INIT_ACHS_KOPPL
(initialise axis coupling 1)
N10 #SET AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
(or #AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
N20 M17
(tool changing program)
%L UP_WZ
N30 #DISABLE AX LINK (oder #AX LINK OFF)
(Approach tool change position)
N40 G01 G90 Y1000 Z100 C0 Y_S=1000 Z_S=100 C_S=0
(Tool change; T10 contains all tool axis offsets and the tool lengths of master and slave tools; or
these values are explicitly included in the calculation.) )
N50 T10 D10
....................
(Further commands for physical tool change)
..................
(Approach old coupling position. The coupling position may also be defined by parameter programming
and then be used by the subroutine.)
N80 G01 G90 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50
N90 #ENABLE AX LINK[1] (or #AX LINK ON[1])
N110 M17

(Subroutine for contour machining)
%L UP1
N150 G01 G91 X10 Y10 Z-20 C90
N160 G02 X20 Y20 I10 J10
N170 LL UP_WZ
N180 G01 G91 X10 Y10 Z-20 C90
N190 G02 X20 Y20 I10 J10
N200 M17


(Main program; initial condition: Both tools were changed.)
(Move both axis systems to coupling position first.)
N300 G01 G91 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50 F300
(Start synchronous operation)
N310 #ENABLE AX LINK[1] (or #AX LINK ON[1])
N320 LL UP1
.....................................
.....................................
N400 #DISABLE AX LINK (or #AX LINK OFF)
N410 M30
```

12.11.4 Inquiring coupling state and coupling number via variables

Variables in the NC program can be used to request the current coupling state and the current or last active coupling group. The following new axis group-related variables are introduced for this purpose:

```
Syntax of Inquiring the current or last active coupling group:
```

V.G.AX_LINK.NR

```
Syntax of Inquiring the current coupling state:
```

V.G.AX_LINK.ACTIVE

Only read access is permissible for these variables. A write operation leads to an error message and to program abort.

12.12 Messages from the NC program

This functionality permits messages from the NC program to be output to any system device. The functionality corresponds to the 'printf' function in the operating systems.

Messages in the NC program can be programmed, e.g. to notify the operator of the current state of the process or the controller. Therefore, the message display may be optionally synchronised with the current processing state. This means that the message is only displayed when the interpolator reaches the point in the NC program.

A message may consist

- of ASCII characters,
- parameters and
- variables.

A message may not be programmed together with other NC commands in the same NC block (exception: #ADD).

The message is sent in ASCII format.

12.12.1 Programming a message (#MSG)

`Syntax:`

#MSG [<*mode*>] [<*receiver*>] **["**<*message_text*>**"]**

<mode>	The mode defines the time of the message output. If the message is to be output synchronously with the processing state in the interpolator, the **SYN** or **SYN_ACK** mode must be specified. If no mode or **ACK** is specified, the message is output immediately after decoding.
	---: Immediately after decoding (default).
	SYN: Synchronous with processing state in interpolator.
	SYN_ACK: Synchronous with processing state in interpolator with acknowledgement by the receiver (e.g. SPS). Processing is only continued in the interpolator after the acknowledgement is received.
	ACK: Immediately after decoding with acknowledgement by the receiver (e.g. SPS). Decoding is only continued after the acknowledgement is received.
<receiver>	The receiver of the message is specified by its receiver ID. If no receiver is specified, it is sent to the default receiver.
	ISG_DIAG_BED: Message is sent to the CNC diagnosis interface (default)
	DIAG: Message is sent to diagnosis data, see P-CHAN-00514
	HMI: Message is sent to the system-specific user interface
	PLC: Message is sent to the system-specific PLC

NOTICE
Messages must be read by the corresponding receiver, otherwise the CNC stops after sending 10 messages.

<message_text>   The message text must be enclosed in quotation marks "...".

Programming a message

```
#MSG HMI ["Text_1"]

#MSG SYN HMI ["Text_2"]

#MSG ["Text_3"] (message sent to default receiver)

#MSG SYN ["Text_4"]

#MSG SYN_ACK ["Text_5"]
```

Text_1 is sent immediately after decoding to the system-specific user interface; the Text_2 string is sent synchronously for processing in the interpolator.

Text_3 is sent immediately after decoding to the default receiver; the Text_4 string is sent to the same device synchronously for processing in the interpolator.

Text_5 is also sent to the default receiver synchronously for processing in the interpolator but processing is stopped until the acknowledgement is received.

The following format elements to output numerical values and to output strings are available:

%d or **%D**	Outputting decimal numbers with sign (signed integer)
%u or **%U**	Outputting decimal numbers without sign (unsigned integer)
%f or **%F**	Outputting real numbers (float)
%s or **%S**	Outputting strings (<String>, Macros, V.E.STRING)

i A maximum of 10 parameters can be output with %xx. The number of format characters must match the number of subsequent parameters.

Programming a message with format elements:

```
#MSG [ "Message text_%d and message text_2", 1 ]
```

Sent string: Message text_1 and message text_2

```
#MSG [ "Current measured value: %f", V.A.MEAS.ACS.VALUE.X]
```

Sent string: Current measured value: 3.4567800000E+001

```
#MSG [ "Current state: %s", "End of roughing"]
```

Sent string: Current state: End of roughing

The control sequence "%%" must be programmed to output the "%" character. The "\" character must precede quotation marks.

Programming a message with "%" and quotation marks:

```
#MSG [ "Text with %% character"]
#MSG [ "Text in quotation marks: \"TEXT\" "]
```

Sent string: Text with % character and text in quotation marks: "TEXT"

A message text may also be specified with parameters and variables.

Programming a message with parameters and variables:

```
P10 = 1
V.P.BSP = 2
#MSG SYN ["Text_%D and Text_%D", P10, V.P.BSP]
```

This example sends the Text_1 and Text_2 strings synchronously to processing in the interpolator to the default receiver.

12.12.2 Programming message information #MSG INFO)

The following command can be used to program user-specific information and display parameters for message processing. Information and parameters are also sent as an ASCII string.

Syntax:

#MSG INFO [*<mode>*] [*<receiver>*] [**"***<message_information>***"**]

Compared to a message in which the message receiver usually displays only the ASCII string, message information indicates that the ASCII string contains user-specific information that is encoded by the receiver itself.

SYN mode must be specified if the message information is to be sent to the receiver synchronously to the processing state in the interpolator. The message is output immediately after decoding if a mode was not specified (see previous section).

The same requirements as for the #MSG command apply to the receiver specifications (see previous section).

The message information must be enclosed in quotation marks "...". The same format elements are available for the output of numerical values as for the #MSG command (see previous section).

Programming message information

```
P1 = 20
#MSG INFO ["RED,%d,BOLD",P1]
```

In this example, the string RED,20,BOLD is sent as message information to the default receiver.

12.12.3   Including the 'Macro' functionality

i | **The availability of this function depends on the configuration and the scope of the version.**

Messages may also be defined as a macro for the message management across NC programs. This is particularly useful for messages that are used repeatedly or by several programs.

Example of a macro content as a message in the channel parameter list [1] [▶ 786]**:**

```
makro_def[1].symbol Meldung_1
makro_def[1].nc_code #MSG ["Message text"]
```

This permits the following message to be output in the NC program by the following macro call:

:	
"Message_1"	(The "Message text" string is sent to the) (default receiver of messages)
:	

12.12.4 Writing messages to a file (#MSG SAVE)

The command #MSG SAVE saves data directly from the NC program to a file on a storage device (e.g. hard disk). The possibilities of structuring a message text and logging data are fully identical to the scope of the commands #MSG or #MSG INFO.

`Syntax:`

#MSG SAVE [**EXCLUSIVE**] [**CONTINUE**] [**"**<*message_text*>**"**]

It is not necessary to specify a mode (SYN) or a receiver ID (e.g. HMI) as this would produce an error message since the message is written directly to the report file in the NC program after evaluation.

The message text must be enclosed in quotation marks "...". Every time #MSG SAVE is called, the message text is added to the end of the report file that already exists. To create a new report file, the user should delete an existing file before NC program start .

#MSG SAVE ["message text "] writes the message text to the report file in the format

< time stamp >

"message text".

#MSG SAVE EXCLUSIVE ["message text"] writes the message text to the report file without time stamp in the format

"message text"

.

#MSG SAVE CONTINUE ["message text"] can suppress a line break at the end of the message text. The next #MSG SAVE command then writes in the same line as the previous #MSG SAVE command.

The name of the report file can be previously defined with the command #FILE NAME. If this command is not used, #MSG SAVE automatically writes to a file with the name message.txt.

The output file path is defined in the start-up list by P-STUP-00018 using the path type P-STUP-00020. If no path information is entered for the report file, a default path is used depending on the control platform or the report file is stored in the main directory of the NC controller.

Writing messages to a file

```
:
#FILE NAME [MSG="example.txt"]        ;Name of output file
:
#MSG SAVE["Hello World %d",12345]     ;Write text "Hello World 12345" in output file example.txt
:
```

12.12.5 Outputting additional informations at block end (#ADD)

The #ADD command can be used to create additional information in the NC block. The possibilities for structuring this additional information (message texts) are fully identical to the scope of the #MSG commands. However, as opposed to the #MSG commands, different NC commands can be programmed before #ADD. Therefore, #ADD must always be programmed as the last command at NC block end. The following comments are permitted.

```
Syntax:
```
#ADD [*<receiver>*] **["***<additional_information>***"]**

It is not necessary to specify a mode (SYN) as this would produce an error message since the message is automatically always output synchronously to the processing state in the interpolator.

The same requirements as for the #MSG commands apply to the receiver specification.

Additional information must be enclosed in quotation marks "...".

Outputting additional information at block end

```
%add_block_info
N05 P1=20
N10 G00 X0 Y0 Z0
N15 T1            #ADD["toolT=%d active", V.G.T_AKT]
N20 G01 F2000 X10 #ADD["Approach X position"] (Comment)
N30 YP1           #ADD["Y position=%d", P1]
N40 Z30           #ADD["Z position"]
N50 Z33 #ADD["Z position"] X11 Y22 ->Error 21509
N999 M30
```

12.13 Jerk limiting slope

The slope function determines the velocity on the programmed path and maintains the specified permissible velocities, accelerations and jerks [2] [▶ 786]-1. The following modes are available :

- Step-shaped acceleration profile with restriction of acceleration without monitoring the jerk
- Trapezoidal acceleration profile with jerk monitoring
- Square-sinusoidal acceleration profile with jerk monitoring

The acceleration curve is generated depending on the slope function selected:

Step-shaped acceleration profile

Trapezoidial acceleration profile

Square-sinusoidial acceleration profile
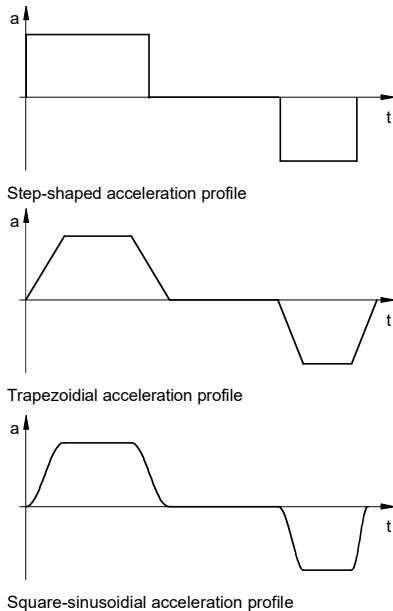
Fig. 105: Acceleration on the programmed path

The acceleration profile is parameterised axis-specific by accelerations and ramp times [2] [▶ 786]-1:
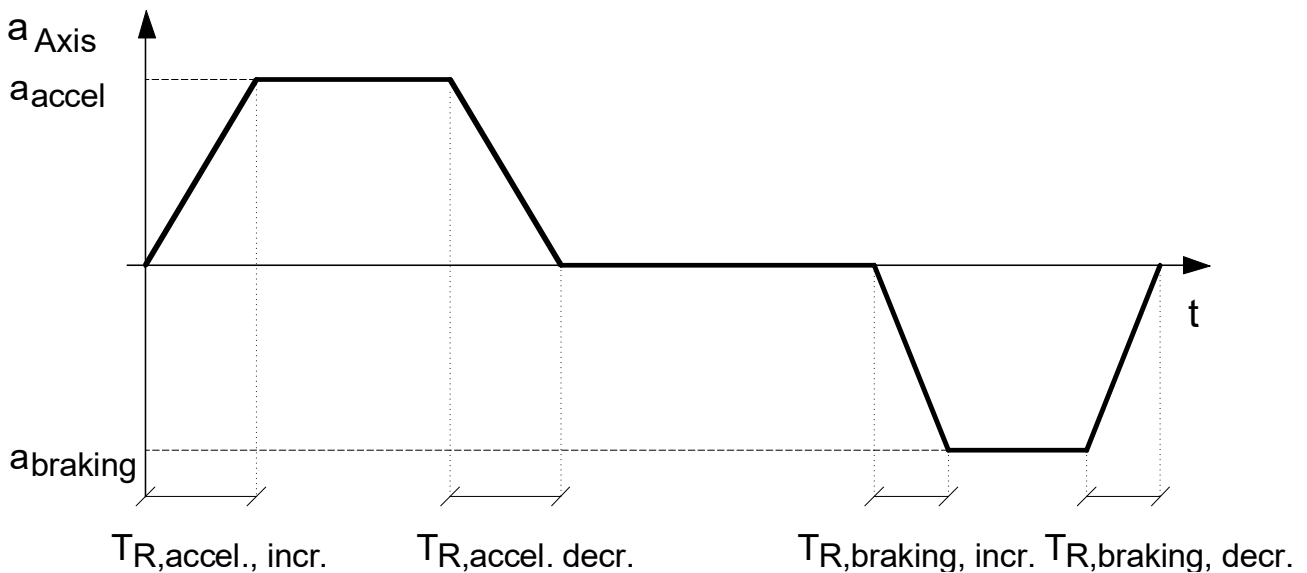
Fig. 106: Parameters of the acceleration profile.

12.13.1 Selecting operating mode (#SLOPE, #SLOPE DEFAULT)

i As of Build **V2.11.2010.02** the command **#SLOPE [...]** replaces the command **#SET SLOPE PRO-FIL [...]**. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

Syntax:

#SLOPE [TYPE=*<ident>* [**NO_OPT=..**] **]**

TYPE*<ident>*	Type of acceleration profile. Permitted identifiers:
	STEP: Step-shaped acceleration profile (default)
	TRAPEZ: Trapezoidal acceleration profile
	SIN2: Square-sinusoidal acceleration profile
	HSC: HSC slope, recommended for "Extended HSC programming" [▶ 242] *
NO_OPT=..	Switch the optimised use of jerk:
	0: Optimised use of jerk is active. This reduces processing time but requires greater computing resources. It must be checked whether the existing hardware is adequate.
	1: Optimised use of jerk is **not** active (default).

i * The use of this feature for selecting the HSC slope profile type requires a licence for the "HSC" extension package. It is not included in the scope of the standard license.

i The specific weighting adaptation of ramp time (G132/G133) and acceleration (G130/G131) is no longer supported by the command #SLOPE [...]. Weightings always act on all ramp times and accelerations (default).

Syntax:

#SLOPE DEFAULT

The programming of #SLOPE DEFAULT restores the initial state (as after start-up). This means that the slope type is set from the channel parameter list P-CHAN-00071.

Select the operation mode

```
N10 G01 X50 Y10 Z0 F1000        (step-shaped accel.profile, default)
N20 #SLOPE [TYPE=TRAPEZ]        (trapezoidal accel.profile)
N30 X10 Y30
N40 #SLOPE [TYPE=SIN2]        (sinusoidal accel.profile)
N50 X15
N60 Y50
N70 M30
```

The following velocity curve results on the programmed path:
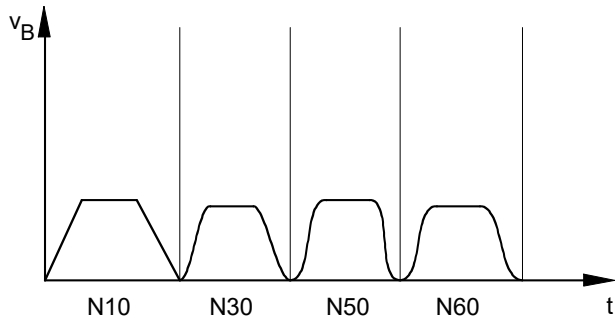
Fig. 107: Velocity curve depending on the programmed path.

12.14 Writing and reading SERCOS parameters and commands

12.14.1 SERCOS parameters (#IDENT)

The following NC commands are used to write and read SERCOS parameters. The original SERCOS format [4] [▶ 786] is used to simplify programming.

Additional information, e.g. axis name or logical axis number, codes and an attribute, are required to correctly process the parameter IDENT number in the drive. This information is programmed together with the IDENT number in the same NC command.

> **i** "Non-synchronous" means execution of the command in the decoding context.
>
> "Synchronous" means execution of the command synchronously with processing, i.e. at interpolator level.

12.14.1.1 Non-synchronised write (#IDENT WR)

Syntax:

#IDENT WR [AX=*<axis_name>* | **AXNR**=.. **ID**=*<Ident_nr>* **VAL**=.. **TYP**=.. **DEC=**.. *<Drive_type>* **]**

AX=*<axis_name>*	Name of SERCOS axis
AXNR=..	Logical axis number of SERCOS axis, positive integer
ID=*<Ident_nr>*	ID Number in SECOS format, e.g. *S-0-0047* or *P-0-0129*
VAL=..	Value to be written; real number
TYP=..	Data type of value (2 or 4 byte length):
	2: 2 bytes data length
	4: 4 bytes data length
DEC=..	Number of decimal places; positive integer
<Drive_type>	Drive type
	SERC**:** SERCOS drive (currently only one permitted)

Non-synchronised write (IDENT)

```
:
#IDENT WR [AX X ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
#IDENT WR [AXNR 1 ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
:
```

NOTICE
No plausibility check is made for logical axis number, identification number or the programmed attributes of data type and decimal places. The operator is solely responsible for making the correct entries.

12.14.1.2 Non-synchronised read (#IDENT RD)

Syntax:

#IDENT RD [AX=*<axis_name>* | **AXNR**=.. **ID**=*<Ident_nr>* **P**=*<variable>* **TYP**=.. **DEC**=.. *<Drive_type>* **]**

AX=*<axis_name>*	Name of SERCOS axis
AXNR=..	Logical axis number of SERCOS axis, positive integer
ID=*<Ident_nr>*	ID Number in SECOS format, e.g. *S-0-0047* or *P-0-0129*

<Variable>	Variable to which the value to be read is saved, e.g. P parameter or V.P. , V.L. or V.S. variables.
TYP=..	Data type of value (2 or 4 byte length):
	2: 2 bytes data length
	4: 4 bytes data length
DEC=..	Number of decimal places; positive integer
<Drive_type>	Drive type
	SERC: SERCOS drive (currently only one permitted)

Non-synchronised read (IDENT)

```
:
#IDENT RD [AX X ID S-0-0104 P=P1 TYP 4 DEC 2 SERC] or

#IDENT RD [AXNR 1 ID S-0-0104 P=V.P.KV_WERT TYP 4 DEC 2 SERC]
:
```

NOTICE
No plausibility check is made for logical axis number, identification number or the programmed attributes of data type and decimal places. The operator is solely responsible for making the correct entries.

12.14.1.3   Synchronised write (#IDENT WR SYN)

Syntax:

#IDENT WR SYN [AX=*<axis_name>* | **AXNR=.. ID=***<Ident_nr>* **VAL=.. TYP=.. DEC=..** <Drive_type>

[**NO_WAIT**] **]**

AX=*<axis_name>*	Name of SERCOS axis
AXNR=..	Logical axis number of the SERCOS axis, positive integer
ID=*<Ident_nr>*	ID Number in SECOS format, e.g. *S-0-0047* or *P-0-0129*
VAL=..	Value to be written; real number
TYP=..	Data type of value (2 or 4 byte length):
	2: 2 bytes data length
	4: 4 bytes data length
DEC=..	Number of decimal places; positive integer
<Drive_type>	Drive type
	SERC: SERCOS drive (currently only one permitted)
NO_WAIT	No waiting for successful writing of the parameter.
	If this keyword is not programmed, the interpolation always waits for the parameter to be written.

Non-synchronised write (IDENT)

```
:
#IDENT WR SYN [AX=X ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
#IDENT WR SYN [AXNR=1 ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
:
#IDENT WR SYN [AX Y ID S-0-0104 VAL655.35 TYP 4 DEC 2 SERC NO_WAIT]
:
```

NOTICE
No plausibility check is made for logical axis number, identification number or the programmed attributes of data type and decimal places. The operator is solely responsible for making the correct entries.

12.14.2   SERCOS commands (COMMAND)

The following NC commands serves to start and wait for execution of SERCOS commands. The original SERCOS format [4] [▶ 786] is used to simplify programming.

Additional information, such as axis name or logical axis number and code, is required for correct processing of the IDENT number command in the drive. This information is programmed together with the IDENT number in the same NC command.

> **i** "Non-synchronous" means execution of the command in the decoding context.
>
> "Synchronous" means execution of the command synchronously with processing, i.e. at interpolator level.

12.14.2.1   Non-synchronised write (#COMMAND WR)

`Syntax:`

#COMMAND WR [AX=*<axis_name>* | **AXNR**=.. **ID**=*<Ident_nr> <Drive_type>* **]**

AX=*<axis_name>*	Name of SERCOS axis
AXNR=..	Logical axis number of SERCOS axis, positive integer
ID=*<Ident_nr>*	Identification number of the command in SERCOS format,
	e.g. S-0-0148 (drive-controlled referencing)
	or S-0-0170 (tracing cycle)
<Drive_type>	Drive type
	SERC: SERCOS drive (currently only one permitted)

Non-synchronised write (COMMAND)

```
:
#COMMAND WR [AX=X ID S-0-0148 SERC]

#COMMAND WR [AXNR 1, ID S-0-0148, SERC]
:
```

NOTICE
No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries.

12.14.2.2   Synchronised write (#COMMAND WR SYN)

`Syntax:`

#COMMAND WR SYN [AX=*<axis_name>* | **AXNR=.. ID**=*<Ident_nr>* *<Drive_type>* **]**

AX=*<axis_name>*	Name of SERCOS axis
AXNR=..	Logical axis number of SERCOS axis, positive integer
ID=*<Ident_nr>*	Identification number of the command in SERCOS format,
	e.g. S-0-0148 (drive-controlled referencing)
	or S-0-0170 (tracing cycle)
<Drive_type>	Drive type
	SERC: SERCOS drive (currently only one permitted)

Synchronised write (COMMAND)

```
:
#COMMAND WR SYN [AX Y, ID S-0-0170, SERC]

#COMMAND WR SYN [AXNR 2 ID S-0-0170 SERC]
:
```

NOTICE
No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries.

BECKHOFF

12.14.2.3   Non-synchronised wait (#COMMAND WAIT)

`Syntax:`

#COMMAND WAIT [AX=*<axis_name>* | **AXNR**=.. **ID**=*<Ident_nr> <Drive_type>* **]**

AX=*<axis_name>*	Name of SERCOS axis
	ALL: Check all SERCOS axes existing in the system
AXNR=..	Logical axis number of SERCOS axis, positive integer
ID=*<Ident_nr>*	Identification number of the command in SERCOS format,
	e.g. S-0-0148 (drive-controlled referencing)
	or S-0-0170 (tracing cycle)
	If no identification number is specified, the system waits for all open commands.
<Drive_type>	Drive type
	SERC: SERCOS drive (currently only one permitted)

Non-synchronised wait (COMMAND)

```
...
#COMMAND WAIT [AX X, ID S-0-0148, SERC]
:
#COMMAND WAIT [AX ALL ID S-0-0148 SERC]
...
```

NOTICE
No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries.

12.14.2.4   Synchronised wait (#COMMAND WAIT SYN)

Syntax:

#COMMAND WAIT SYN [AX=*<axis_name>* | **AXNR**=*..* **ID**=*<Ident_nr>* *<Drive_type>* **]**

AX=*<axis_name>*	Name of SERCOS axis
	ALL: Check all SERCOS axes existing in the system
AXNR=..	Logical axis number of SERCOS axis, positive integer
ID=*<Ident_nr>*	Identification number of the command in SERCOS format,
	e.g. S-0-0148 (drive-controlled referencing)
	or S-0-0170 (tracing cycle)
	If no identification number is specified, the system waits for all open commands.
<Drive_type>	Drive type
	SERC: SERCOS drive (currently only one permitted)

Synchronised wait (COMMAND)

```
...
#COMMAND WAIT SYN [AX=X, ID=S-0-0148, SERC]
:
#COMMAND WAIT SYN [AX ALL, ID S-0-0148, SERC]
...
```

NOTICE
No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries.

i   The motion is not forcibly stopped to wait for a SERCOS command. But if a SERCOS command is not terminated at the end of the motion block, no further NC block is processed and the motion is therefore stopped.

The system waits until the S-0-0148 command terminates at the end of block N120.

```
..
N100 #COMMAND WR SYN [AX Y ID S-0-0148 SERC]
N110 G01 X1000 F100
N120 #COMMAND WAIT SYN [AX Y ID S-0-0148 SERC]
N130 G01 X2000
...
```

NOTICE
A ("WAIT") command can only check commands which were previously started from the same processing level (decoding context or synchronous to processing at interpolation level). For example, a synchronised command ("SYN") can only be checked synchronously at interpolator level.

No active command S-0-0148 is found in the N100 block, so no wait is initiated although the command could still actually be active. The real state of the command at interpolation level is only detected at block N120.

```
...
N100 #COMMAND WR SYN [AX Y ID S-0-0148 SERC]
N110 #COMMAND WAIT [AX Y ID S-0-0148 SERC]
N120 #COMMAND WAIT SYN [AX Y ID S-0-0148 SERC]
...
```

12.15   Channel synchronisation

When a multi-channel controller is used (particularly in the case of n > 2), situations may occur in which it is absolutely essential to maintain with specific run sequences between channels.
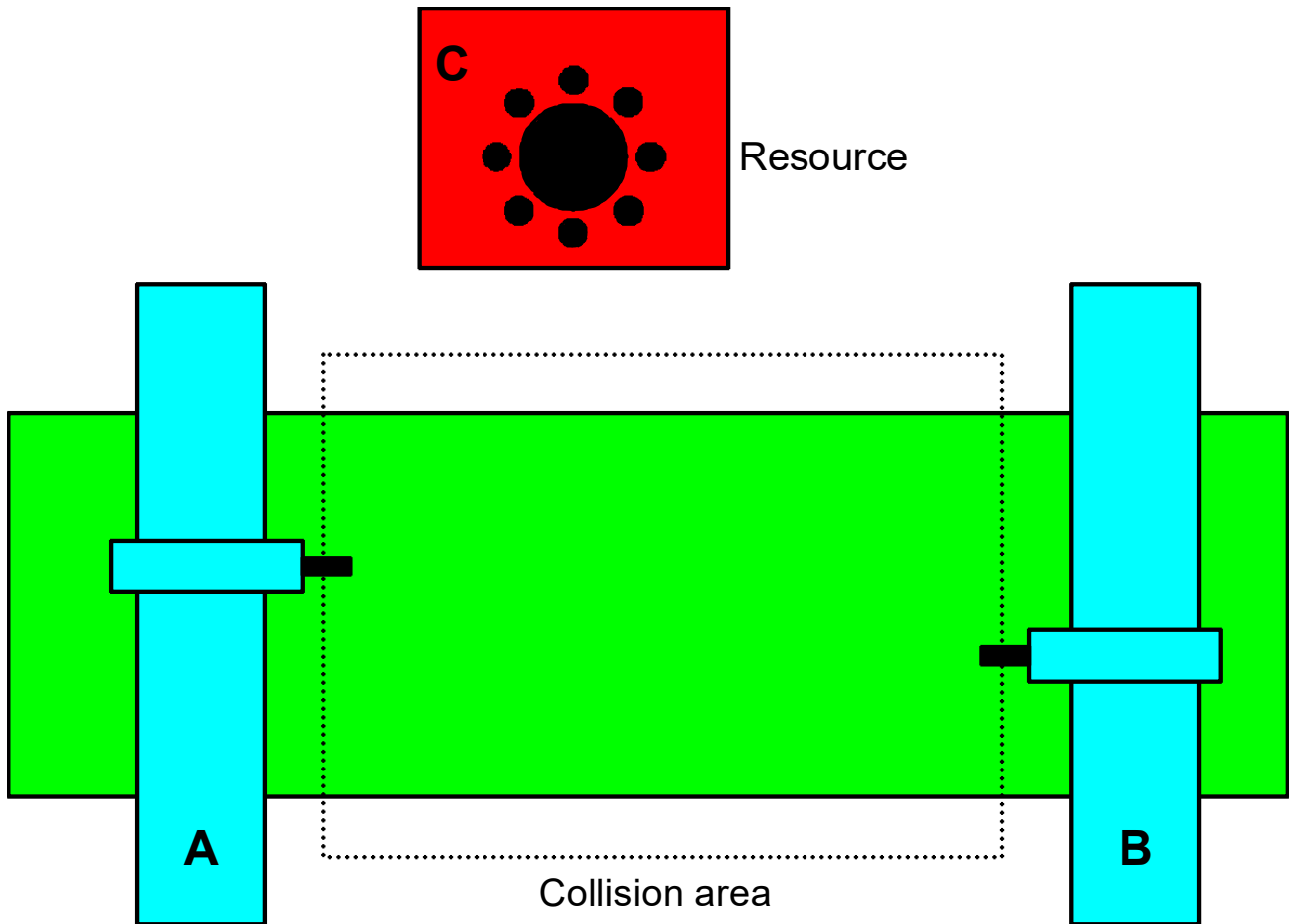
Fig. 108: Example application: double-column machine with tool changer

The example in the figure above shows such an application in which two machining units (A and B) share a common machining space. Similarly, both machines use the tool changer resource (C). In order to avoid collisions in such a machine configuration, the NC programs of the various controller channels must be synchronised with each other at specific points. For example, in the above case, column A may not enter the collision space while column B is still located there. Similarly, column B may not use the tool changer if column A is currently accessing it.

For example, the time sequence in the two channels of the controller as shown in the figure below results from an access to the tool changer resource.
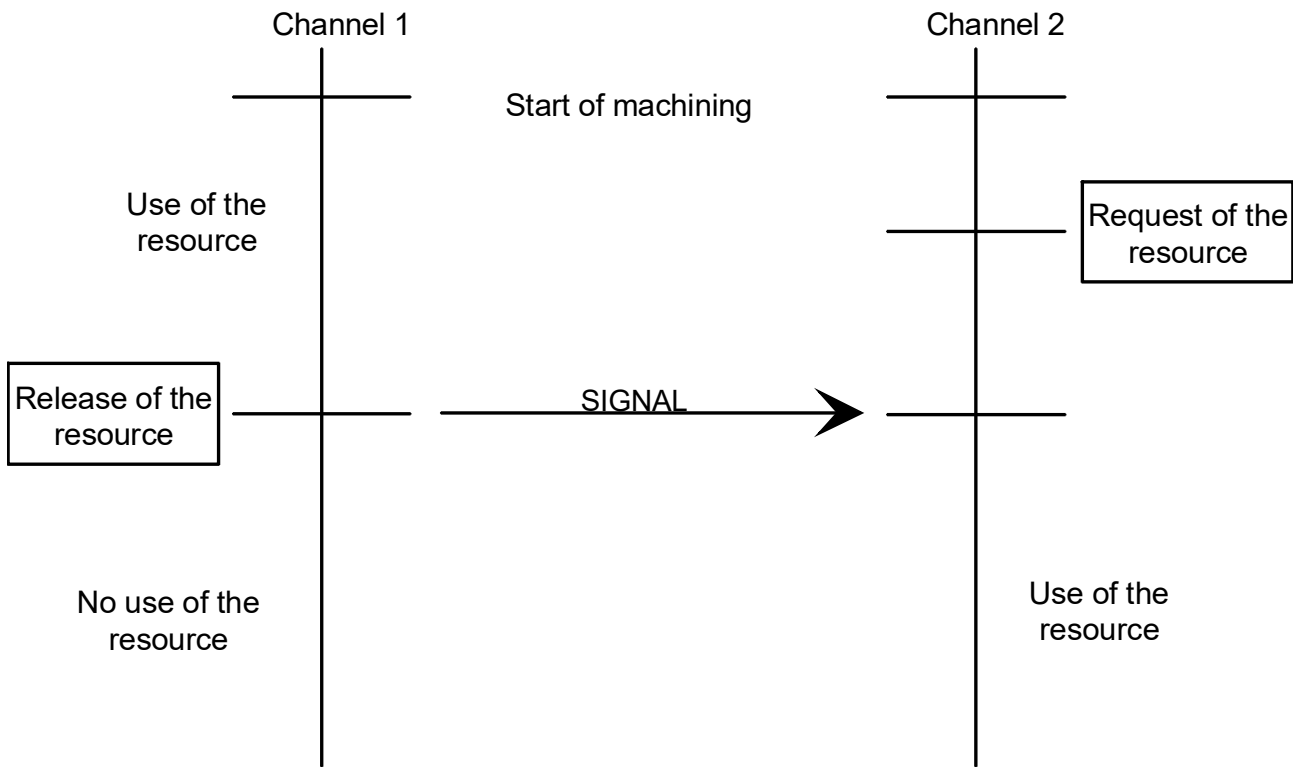
Fig. 109: Sequence in case of shared access to a resource

The required synchronisation is based on sending and waiting for signals and is performed by the NC commands in the NC programs described below.

12.15.1 Synchronisation scenarios

Synchronisation of 2 decoders on 2 channels
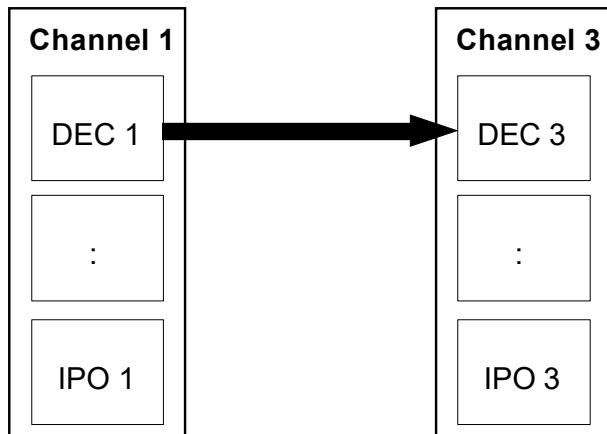
Fig. 110: Synchronisation of 2 decoders on 2 channels

- Decoder 3 waits for decoder 1, decoder 1 continues to operate without interruption

Synchronisation of 2 decoders on 2 channels

```
% kanal_1
...
(Signal P100)
(Synchronisation at DEC level)
(Synchronisation with channel 3)
(Parameter V.P.SYNC)

V.P.SYNC = 1000
P100 = 814

#SIGNAL [IDP100 P[0]= V.P.SYNC CH3]
...
```

```
% kanal_3
...
(Wait request 814)
(Synchronisation at DEC level)
(Synchronisation with channel 1)
(Parameter V.P.SIGNAL)

#WAIT [ID814 P[0]= V.P.SIGNAL CH1]
...
```

Synchronisation between decoder and interpolators on 3 channels
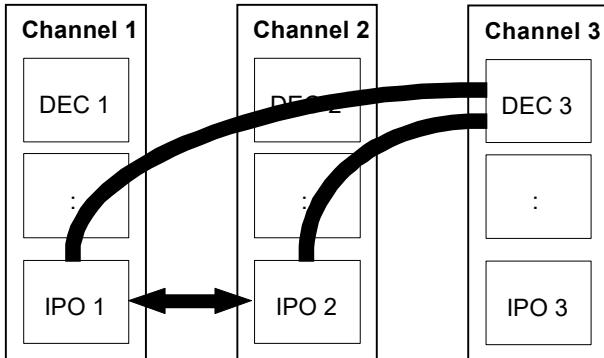
Fig. 111: Synchronisation between decoder and interpolators on 3 channels

- Interpolator 1 waits for interpolator 2 and decoder 3,
- Interpolator 2 waits for interpolator 1 and decoder 3,
- Decoder 3 signals to interpolator 1 and interpolator 2.

Synchronisation between decoder and interpolators on 3 channels

```
% kanal_1
...
(Wait request 968)
(Sync. At IPO level)
(Sync. with channels 2 and 3)

#WAIT SYN [ID968 CH2 CH3]
```

```
%kanal_2
...
(Wait request 968)
(Sync. At IPO level)
(Sync. with channels 3 and 1)

#WAIT SYN [ID968 CH3 CH1]
```

```
% kanal_3
...
(Signal 968)
(Sync. At DEC level)
(Sync. with channels 1 and 2)

#SIGNAL [ID968 CH1 CH2]
```

Synchronisation between interpolators on 3 channels
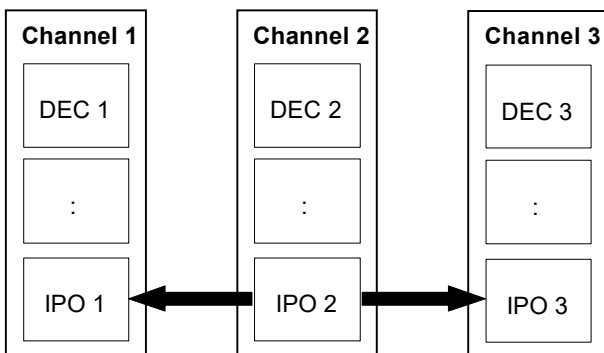
Synchronisation between interpolators on 3 channels

- Interpolator 1 waits for interpolator 2,
- Interpolator 3 waits for interpolator 2,
- Interpolator 2 signals to interpolator 1 and interpolator 3.

Synchronisation between interpolators on three channels

```% kanal_1 ... (Wait request 100) (Sync. At IPO level) (Sync. with channel 2) #WAIT SYN [ID100 CH2]```	```%kanal_2 ... (Signal 100) (Sync. At IPO level) (Sync. with channels 1 and 3) #SIGNAL SYN [ID100 CH1 CH3]```	```% kanal_3 ... (Wait request 100) (Sync. At IPO level) (Sync. with channel 2) #WAIT SYN [ID100 CH2]```

Synchronisation between decoder and interpolator of one channel
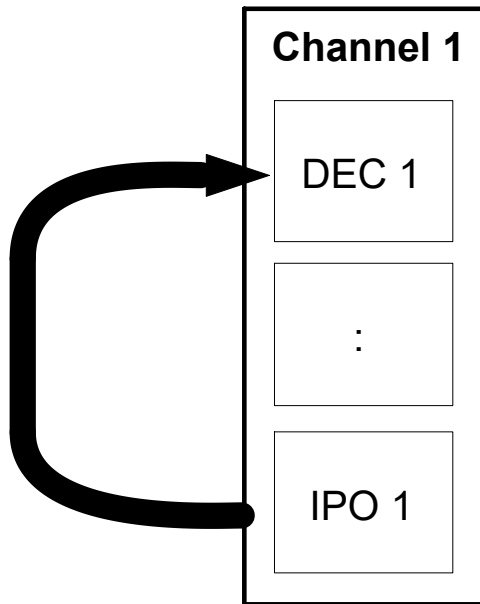
Fig. 112: Synchronisation between decoder and interpolator of one channel

- Decoder waits until interpolator has reached position X 250.
- Motion block "G01 X370 Z200 F80" is already on the NC channel and is processed after signalling.
- Motion block "G01 X900" is decoded only after synchronisation.

NOTICE
In the case of synchronisation requests between decoder and interpolator, states may occur in which the NC program cannot be decoded further since no acknowledgement has yet arrived. The acknowledgement is, however, not dispatched by the interpolator since the signal block does not reach the interpolator owing to the buffer effect of the NC channel. A #FLUSH which flushes the NC channel must be provided in such cases in order to avoid possible deadlocks.

Synchronisation between decoder and interpolator of one channel

```
% kanal_1

G00 X100 Y500
G01 X250 F300

(Signal 88)
(Synchronisation at IPO level)
(Synchronisation with channel 1)

#SIGNAL SYN [ID88 CH1]

(Operation)

G01 X370 Z200 F80

(Wait request 88)
(Synchronisation at DEC level)
(Synchronisation with channel 1)
#FLUSH
#WAIT [ID88 CH1]

G01 X900
...
```

12.15.2 Sending signals (# SIGNAL)

Basically, a distinction is made between signals <u>without</u> a specified receiver (also called broadcast signals) and non-broadcast signals where a channel is explicitly specified as the receiver.

The signals are identified by a unique number, although it is permitted to send signals with identical signal numbers.

In the case of non-broadcast signals, one or more NC channels must be explicitly specified as receivers. If several receivers are specified for one signal, this acts in the same way as the multiple transmission of the same signals to single channels.

Sending signals

```
#SIGNAL [ID4711 CH1 CH2 CH3]
```

acts in the same way as

```
#SIGNAL [ID4711 CH1]
```
```
#SIGNAL [ID4711 CH2]
```
```
#SIGNAL [ID4711 CH3]
```

These signals are only valid for a WAIT of the addressed receiver and are used up if the consumption counter COUNT is not specified for a WAIT from the receiver channel. If a consumption counter is specified, the same number of WAIT requests are possible until the signal is used up.

As opposed to this situation, broadcast signals can be received by a WAIT from any channel.

If no consumption counter COUNT is specified for broadcast signals, they are not used up by a WAIT. This means that they remain until they are explicitly cleared (see #SIGNAL REMOVE). If a consumption counter COUNT is specified, the exact same number of WAIT requests is possible until the broadcast signal is used up as is the case with non-broadcast signals.

```
Syntax:
```
#SIGNAL [*<mode>*] **[ID**=.. [**COUNT**=..] { **P[***idx*]**=**<*param*> } { **CH**=.. } [**KEEP_AT_RESET**] **]**

<mode>	Synchronisation mode. Permitted identifiers:
	---: Synchronisation at decoding level (initial state) For example, this synchronisation is required if it is necessary to synchronise to parameters or variables.
	SYN: Synchronisation at interpolator level. This synchronisation is required in the case of real-time requests, e.g. synchronisation of two machining units on a multi-column machine
ID=..	Signal number; must be unique system-wide. Positive integer.
COUNT=..	Consumption counter; defines how many times a signal can be called with #WAIT [▶ 376]. Positive integer.
P[*<idx>*] = *<param>*	Signal parameter. The real values *<param>* are entered in the parameter array of the signal according to the specified index.
	<idx>: Range for maximum possible number of parameters: 0 .. 11 (Max. number of coupling pairs[1])
CH=..	Number of the channel for which the signal is destined. 1...max. number of channels [2]
	If no channel number is specified, a so-called **broadcast signal** is sent to all available signal receivers in the system.
KEEP_AT_RESET	Flagged signal (standard signal with receiver) is not removed at reset. This means that the signal remains retained even after the channel to be sent is reset until it is consumed by a #WAIT [▶ 376] with the correct ID or removed by explicit programming (#SIGNAL REMOVE [▶ 374])

(1) see [6] [▶ 786]-6.45, (2) see [6] [▶ 786]-2.4

Sending signals

```
(Signal 812, synchronisation at DEC level, broadcast)
N500 #SIGNAL [ID812]

(Signal 4711, synchronisation at DEC level, to channel 2)
N100 #SIGNAL [ ID4711 CH2 ]

(Signal 4711, synchronisation at DEC level,
for 10 #WAIT requests, broadcast)
N100 #SIGNAL [ ID4711 COUNT10 ]

(Signal 815, synchronisation at interpolator level,
twice to channel 2 and once to 3)
N200 #SIGNAL SYN [ ID815 CH2 CH2 CH3 ]

(Signal 911, synchronisation at decoder level, to channel 3)
(1St signal parameter V.A.MEAS.ACS.VALUE.X, 2nd signal parameter P200,
3rd signal parameter 94.4)
N260 P200 = 924
N300 #SIGNAL [ IDP100 CH3 P[0]=V.A.MEAS.ACS.VALUE.X P[1]=P200 P[2]=94.4 ]
```

12.15.3   Removing (broadcast) signals (#SIGNAL REMOVE)

Signals are generally cleared after consumption by an assigned WAIT. In addition, non-broadcast signals are implicitly cleared for an NC reset of the receiver channel (see section Reset handling). Since broadcast signals are not cleared by a WAIT unless the consumption counter is specified, they must be cleared explicitly. An additional NC command exists for this purpose. This NC command can also be used to clear normal signals although in this case the identification number and the addressed channel must match.

If a single signal is specified for clearing, only one signal is cleared if more than one identical signals exist. However, when specifying a signal range [ID; IDMAX], i.e. including several identical signal numbers, all signals within this range are cleared.

```
Syntax:
```
#SIGNAL REMOVE [<*mode*>] **[ID=..** | **IDMIN**=.. [**IDMAX**=..] { **CH=..** } **]**

<mode>	Synchronisation mode. Permitted identifiers:
	---: Synchronisation at decoding level (initial state) For example, this synchronisation is required if it is necessary to synchronise to parameters or variables.
	SYN: Synchronisation at interpolator level. This synchronisation is required in the case of real-time requests, e.g. synchronisation of two machining units on a multi-column machine
ID=..	Number of broadcast signal to be cleared. Positive integer.
IDMIN=..	First broadcast signal in a range to be cleared. Also alternative to ID=... Positive integer
IDMAX=..	Last broadcast signal in a range to be cleared. Positive integer
CH=..	Number of channel for which the signal to be cleared is destined. 1...max. number of channels [1]
	If no channel number is specified, the corresponding **broadcast** signal is cleared

(1) see [6] [▶ 786]-2.4

Clearing (broadcast) signals

```
(Clear a broadcast signal 812, synchronisation at DEC level)
N500 #SIGNAL REMOVE [ID812] or
#SIGNAL REMOVE [IDMIN812]

(Clear a signal 812 to channel2, synchronisation at DEC level)
N500 #SIGNAL REMOVE [ID812 CH2]

(Clear all broadcast signals in 812-820,
synchronisation at DEC level)
N500 #SIGNAL REMOVE [IDMIN812 IDMAX820] or
#SIGNAL REMOVE [ID812 IDMAX820]

(Clear all signals 812 to channel 1, synchronisation at DEC level)
N500 #SIGNAL REMOVE [ID812 IDMAX812 CH1]

(Clear a broadcast signal 813, synchronisation at IPO level)
N600 #SIGNAL REMOVE SYN [ID813]
```

12.15.4   Waiting for signals (#WAIT)

Analogously to sending signals, it is possible with the WAIT command to wait for a corresponding SIGNAL. A broadcast WAIT waits only for a broadcast SIGNAL with the same signal number. A WAIT synchronised at decoder/interpolator level uses up one separate SIGNAL in each case.

```
Syntax:
```
#WAIT [<*mode*>] **[ID**=.. { **P[**<*idx*>**] =** <*param*> } { **CH**=.. } [AHEAD]]

<mode>	Synchronisation mode. Permitted identifiers:
	---: Synchronisation at decoding level (initial state) For example, this synchronisation is required if it is necessary to synchronise to parameters or variables.
	SYN: Synchronisation at interpolator level. This synchronisation is required in the case of real-time requests, e.g. synchronisation of two machining units on a multi-column machine
ID=..	Number of the signal for which the system is waiting. Positive integer.
P[<*idx*>] = <*param*>	Signal parameter as real number. While waiting for signals, parameters can also be sent by the signal sender. Parameters can also originate from different channels. They are assigned to the specified parameters or variables (<*param*>).
	<idx>: Range for maximum possible number of parameters: 0 .. 11 (Max. number of coupling pairs[1])
	After complete acknowledgement of the waiting condition (reception of all required signals), a check is made whether all programmed parameters were written. An error message is generated if id not the case.

NOTICE
Signal parameters can only be evaluated at decoder level. This means, for example, a #WAIT **SYN** [... P[0] = ...] is not allowed.

CH=..	Channel number from which a signal is expected. 1...max. number of channels [2]
	If no channel number is specified, the program waits for a broadcast signal from any user.
AHEAD	Keyword for execution of a "flying" WAIT. Used to reduce waiting times because of the buffer effect of the look-ahead function (up to 70 blocks in advance). If synchronised at interpolator level, WAIT is output at once. As a result, the following acknowledgement check (SIGNAL) is flying, i.e. a change can be made immediately to the next motion block without interruption.

(1) see [6] [▶ 786]-6.45

(2) see [6] [▶ 786]-2.4

Waiting for signals

```
(Wait flag 4711, synchronisation at DEC level, SIGNAL 4711 from
any channel)
N200 #WAIT [ID4711]

((Wait flag 815, synchronisation at interpolator level,
SIGNAL 815 from channels 2 and 3)
N100 #WAIT SYN [ID815 CH2 CH3]

(Wait flag 911, synchronisation at decoder level, from channel 3)
(1st signal parameter V.P.SIGNAL, 2nd signal parameter P200)
N250 P100 = 911
N300 #WAIT [IDP100 P[0]=V.P.SIGNAL P[1]=P200 CH3]
(The calculation below only takes place when)
(the signal is received)
N350 P20 = 10 * P200
```

Wait for signals with adoption of parameters (in channel 3):

```
%channel1
N10 #SIGNAL [ID 110014 P[0] = 1234 CH3]
N20 M30

%channel2
N10 #SIGNAL [ID 110014 P[1] = 200 CH3]
N20 M30

%channel3
N10 P1 = 1 (Stores value from channel 1
N20 P2 = 1 (Stores value from channel 2
N30 XP1 YP2
N40 #WAIT [ID 110014 P[0] = P1 P[1] = P2 CH1 CH2]
N50 XP1 YP2
N60 M30
```

12.15.5   Reading signals without waiting (#SIGNAL READ)

ℹ **This function is available as of CNC Build V2.11.2820.00**

The NC command #WAIT stops the program decoding of the interpreter if the requested signal is missing. This blocks further program processing if the signal never arrives.

The NC command #SIGNAL READ allows flexible program sequences without a program stop. This command stores the result of the signal read operation in the channel-specific variable V.G.SIGNAL_READ. When this variable is subsequently evaluated, a corresponding response is possible.

#SIGNAL READ behaves analogously to #WAIT with respect to the use of signals as well as the programming and use of parameters and broadcast signals.

NOTICE
The NC command #SIGNAL READ is only permitted at interpreter level. A #SIGNAL READ SYN [..] is not permitted and is indicated by an error message.

The status of the read access of #SIGNAL READ is indicated by the variable V.G.SIGNAL_READ. It is TRUE if the corresponding signal was present. The value of the variable remains until the next read access with #SIGNAL READ.

V.G.SIGNAL_READ	Status of read access of #SIGNAL READ
	TRUE: Signal present and read
	FALSE: No signal present, default

```
Syntax:
```
#SIGNAL READ [ID=.. { **P[**<*idx*>**] =** <*param*> } { **CH**=.. } **]**

ID=..	Number of the signal to be read. Positive integer.
P[<idx>] = <param>	Signal parameter as real number. When signals are read, parameters can also be sent by the signal sender. Parameters can also originate from different channels. They are assigned to the specified parameters or variables (<*param*>).
	<idx>: Range for maximum possible number of parameters: 0...max. Number of signal parameters [1]
	After complete acknowledgement of all the required signals, a check is made whether all programmed parameters were written. If this is not the case, the program stops with an error message
CH=..	Channel number from which a signal is expected. 1...max. number of channels [2]
	If no channel number is specified, the program waits for a broadcast signal from any user

(1) see [6] [▶ 786]-6.45

(2) see [6] [▶ 786]-2.4

Read signals and wait for result

```
(Wait flag 4711, synchronisation at interpreter level, SIGNAL 4711 from any channel)
:
N100 #SIGNAL READ [ID=4711]
N110 $IF V.G.SIGNAL_READ == TRUE
N120   LL UP1
```

```
N130 $ELSE
N140   #ERROR [..]
N150 $ENDIF
:
```

(Wait flag 815, synchronisation at interpreter level,
SIGNAL 815 from channel 1)

```
:
N100 #SIGNAL READ [ID=815 CH=1]
N110 $IF V.G.SIGNAL_READ == TRUE
N120   LL UP1
N130 $ELSE
N140   L Init.nc
N150 $ENDIF
:
```

(Wait flag 911, synchronisation at interpreter level, from channel 3)
(1st signal parameter V.P.SIGNAL, 2nd signal parameter P200)

```
:
N100 #SIGNAL READ [ID=911 P[0]=V.P.SIGNAL P[1]=P200 CH=3]
N110 $IF V.G.SIGNAL_READ == TRUE
N120   P20 = [10 * P200]-V.P.SIGNAL
N130 $ELSE
N140   P20 = 0
N150 $ENDIF
```

12.15.6 RESET handling

If a single channel is reset, the synchronisation events of the channel affected are cleared, i.e. all wait requests (#WAIT) which the channel in question has sent and all non-broadcast signals (#SIGNAL) destined for it are cleared.

Broadcast signals are not cleared if a channel is reset since these signals may still be expected by channels. They must be cleared explicitly (#SIGNAL REMOVE).

12.16 Rotate the coordinate system in the plane (#ROTATION ON/OFF)

This function rotates a coordinate system in the current plane (G17/G18/G19). Contours programmed in the machine coordinate system can be adapted quickly and easily to workpieces in offset positions.

Contour rotation acts directly on the programmed axis coordinates (contour) **before** all other contour-influencing functions, i.e. all offsets and mirroring operations are not influenced by the rotation and can be used as before (*).

Rotation may also be applied within an already rotated coordinate system (#(A)CS).

A change of plane with G17/ G18/ G19 automatically deselects an active contour rotation and a warning is output.

As a alternative to #ROTATION, contour rotation can be programmed using G68/G69 [▶ 181].

Syntax:

#ROTATION ON [**[** [**ANGLE=**..] [**CENTER1**=..] [**CENTER2**=..] **]**]
#ROTATION OFF

ANGLE=..	Rotation angle in [°]
CENTER1=..	Offset of the first main axis relative to the centre of rotation in [mm, inch]
CENTER2=..	Offset of the second main axis relative to the centre of rotation in [mm, inch]
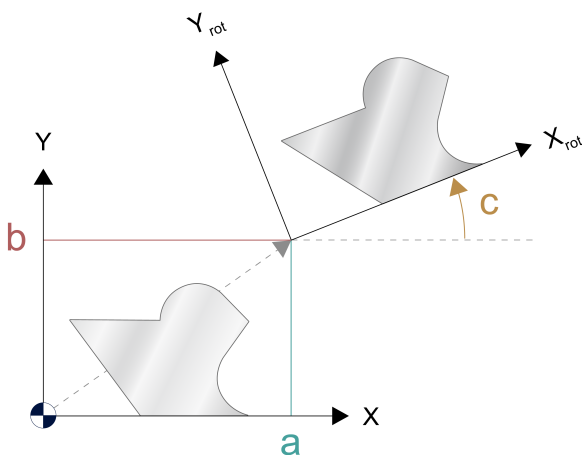
Fig. 113: Significance of rotation parameters in the main plane (example G17):

a: CENTER1	b: CENTER2	c: ANGLE

The programmed rotation parameters can be read with the following variables:

V.G.ROT_ACTIVE	Contains the value 1 if a rotation is active
V.G.ROT_ANGLE	Rotation angle
V.G.ROT_CENTER1	Offset of the first main axis relative to the centre of rotation

V.G.ROT_CENTER2   Offset of the second main axis relative to the centre of rotation

ℹ️ (*) It makes no difference whether the offsets (e.g. G54, G92 etc.) were programmed before or after the #ROTATION command; they always act in the axis directions of the basic coordinate system of the machine (MCS).

In addition, tool offsets always act independently of P-TOOL-00010 in the axis directions of the MCS.

Rotation in a plane (contour rotation)

```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

%ang1.nc
N100 G53 G17
N110 LL part
N130 #ROTATION ON [ANGLE -45 CENTER1=10 CENTER2=100]
N140 LL part
N150 G21 (mirroring of X coordinates)
N160 LL part
N170 G18 (warning expected)
N190
M30
```
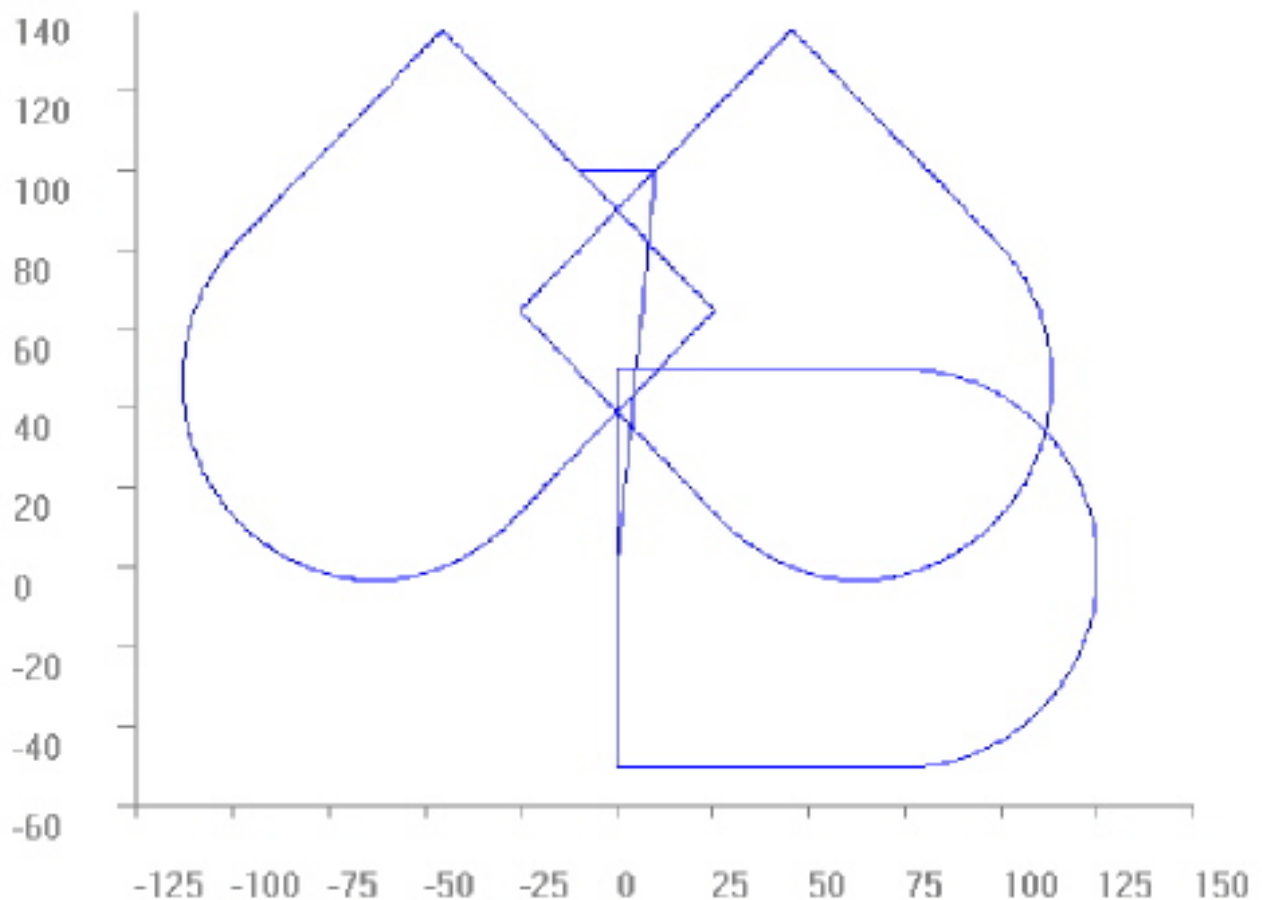
Same contour as in the previous program but within #CS of -15°.

```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

% ang1cs.nc
N99 #CS ON[0,0,0,0,0,-15]
N100 G53 G17
N110 LL part
N130 #ROTATION ON [ANGLE -45 CENTER1 10 CENTER2 100]
N140 LL part
N150 G21 (mirroring of X coordinates)
N160 LL part
N190 #CS OFF
M30
```
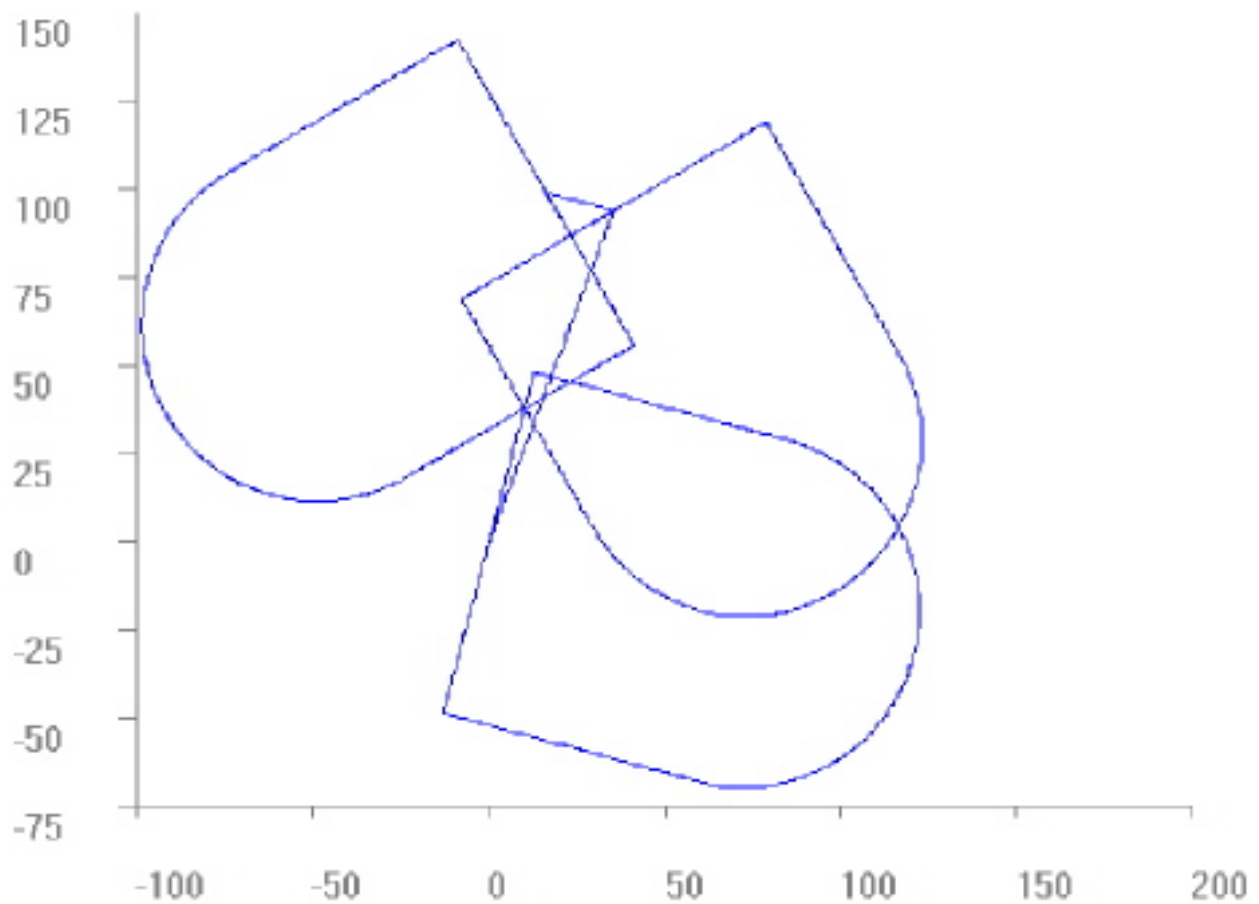
```
%L Trajectory0
N10 G54 G90 X0 Y0
N20 G0 X75 Y-50
N30 Y50
N40 X-75
N50 G3 X-75 Y-50 R50
N60 G0 X75
N70 X0 Y0
N80 M29

%ang2.nc
F1000
N100 LL Trajectory0
N200 G92 G90 Y-25
N400 #ROTATION ON [ANGLE 90 CENTER1 75 CENTER2=-50]
N600 LL Trajectory0
N700 G92 G90 Y25
N900 #ROTATION ON [ANGLE=-90 CENTER1 75 CENTER2 50]
N60 LL Trajectory0
N70 M30
```
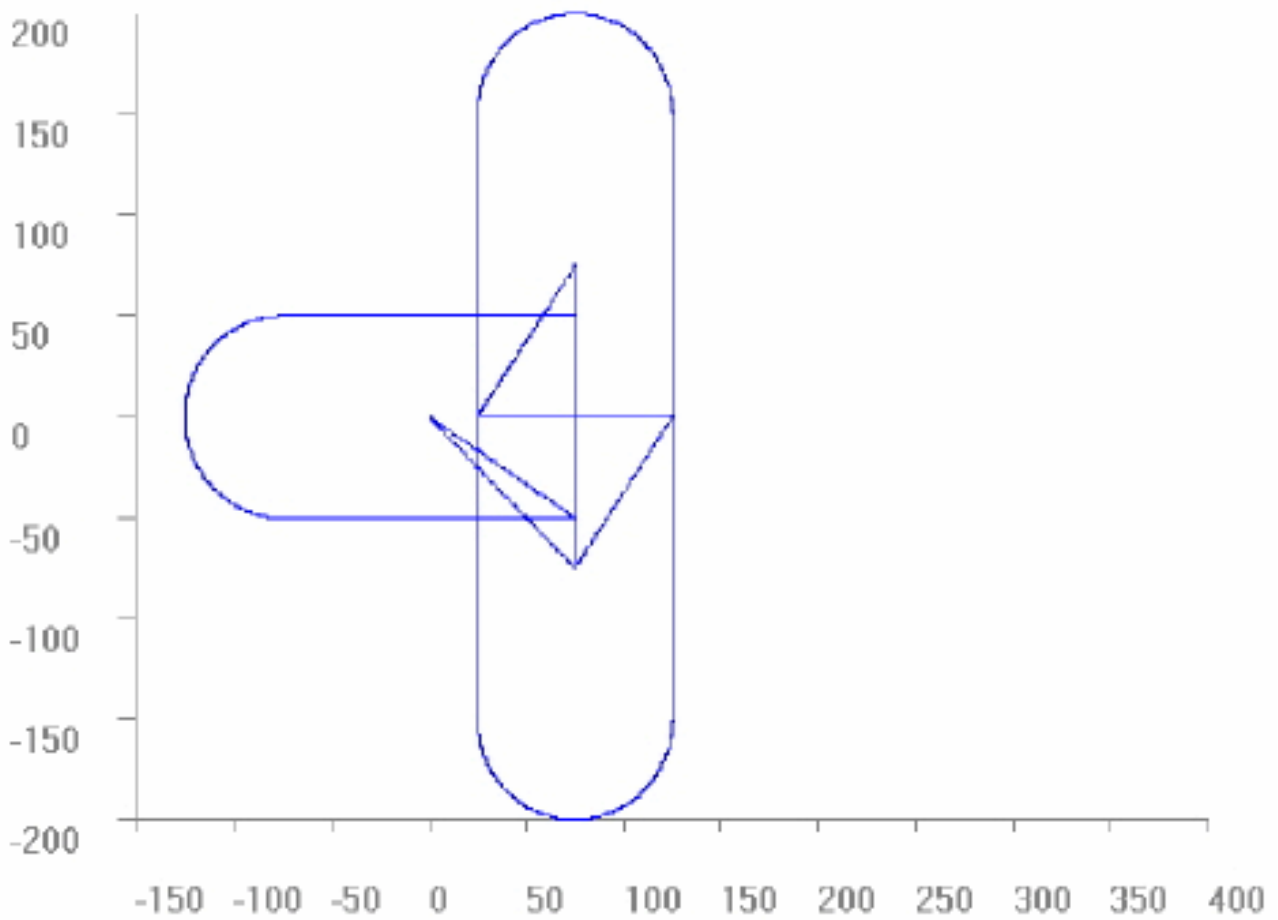
```
%L Trajectory0
N10 G54 G90 X0 Y0
N20 G1 X75 Y-50
N30 Y50
N40 X-75
N50 G3 X-75 Y-50 R50
N60 G1 X75
N70 X0 Y0
N80 M29

%ang3.nc
N10 F4000 G90
N15 #ROTATION ON
N20 LL Trajectory0
N30 G90 G92 Y100
N35 #ROTATION ON [ANGLE 180]
N40 LL Trajectory0
N50 G90 G92 Y-100
N55 #ROTATION ON [ANGLE 180]
N60 LL Trajectory0
N70 M30
```
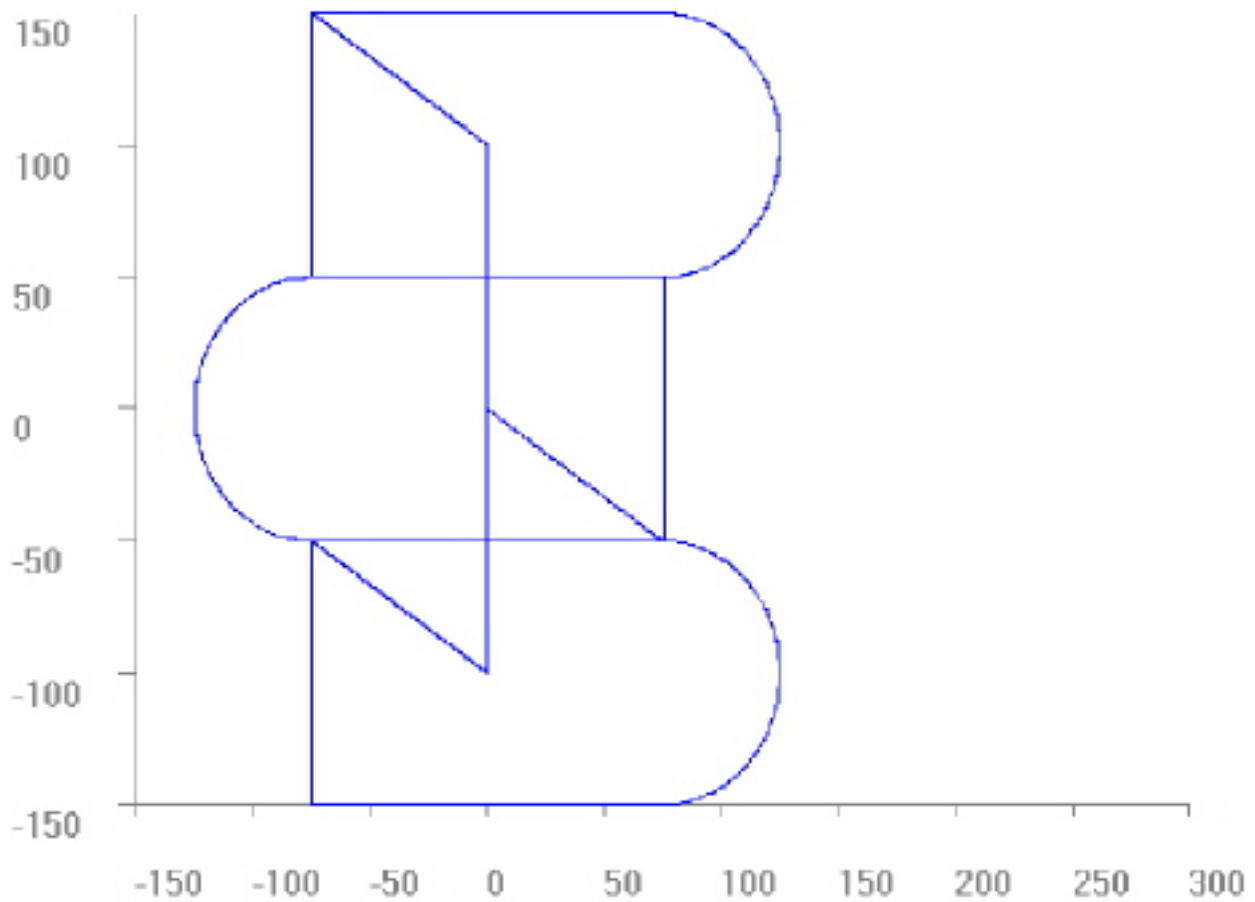
```
%L UPRG1
N1 X0 Y0 Z0
N10 X25
N30 X0
N40 Y25
N50 Y0
N60 X10
N70 Y10
N80 X0 Y0
N90 Y10
N100 X10 Y0
N110 G03 I-5 J5 Y10
N120 G1 X0 Y0
M17

%L UPRG2
N2 X0 Y0 Z0
N10 X25
N20 G02 I0.8
N30 G1 X0
N40 Y25
N45 G02 J0.8
N50 G1 Y0
N120 G1 X0 Y0
M17

%L UPRG3
N3 G1 X0 Y0 Z0
N10 X4 Y4
N20 G02 I1 J1
N30 G1 X0 Y0 Z0
M17

%ang4.nc
N1 G1 X0 Y0 Z0 F1000

N500 G92 X10 Y10
N510 LL UPRG1
N520 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 15]
N540 LL UPRG1
N550 G92 X20 Y25
N560 #ROTATION ON [ANGLE -35]
N570 LL UPRG1
N580 G92 X35 Y-10
N590 #ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG1
N610 #ROTATION ON [CENTER1 0 CENTER2 0]
N620 LL UPRG1
N630 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+117]
N640 LL UPRG1
N650 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]

N500 G92 X10 Y10
N510 LL UPRG2
N520 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 15]
N540 LL UPRG2
N550 G92 X20 Y25
N560 #ROTATION ON [ANGLE -35]
N570 LL UPRG2
N580 G92 X35 Y-10
N590 #ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG2
N610 #ROTATION ON [CENTER1 0 CENTER2 0]
N620 LL UPRG2
N630 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+117]
N640 LL UPRG2
N650 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]
N500 G92 X10 Y10
N510 LL UPRG3
N520 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 15]
N540 LL UPRG3
N550 G92 X20 Y25
N560 #ROTATION ON [ANGLE -35]
N570 LL UPRG3
N580 G92 X35 Y-10
N590 #ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG3
N610 #ROTATION ON [CENTER1 0 CENTER2 0]
N620 LL UPRG3
N630 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+117]
```

```
N640 LL UPRG3
N650 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]
M30
```
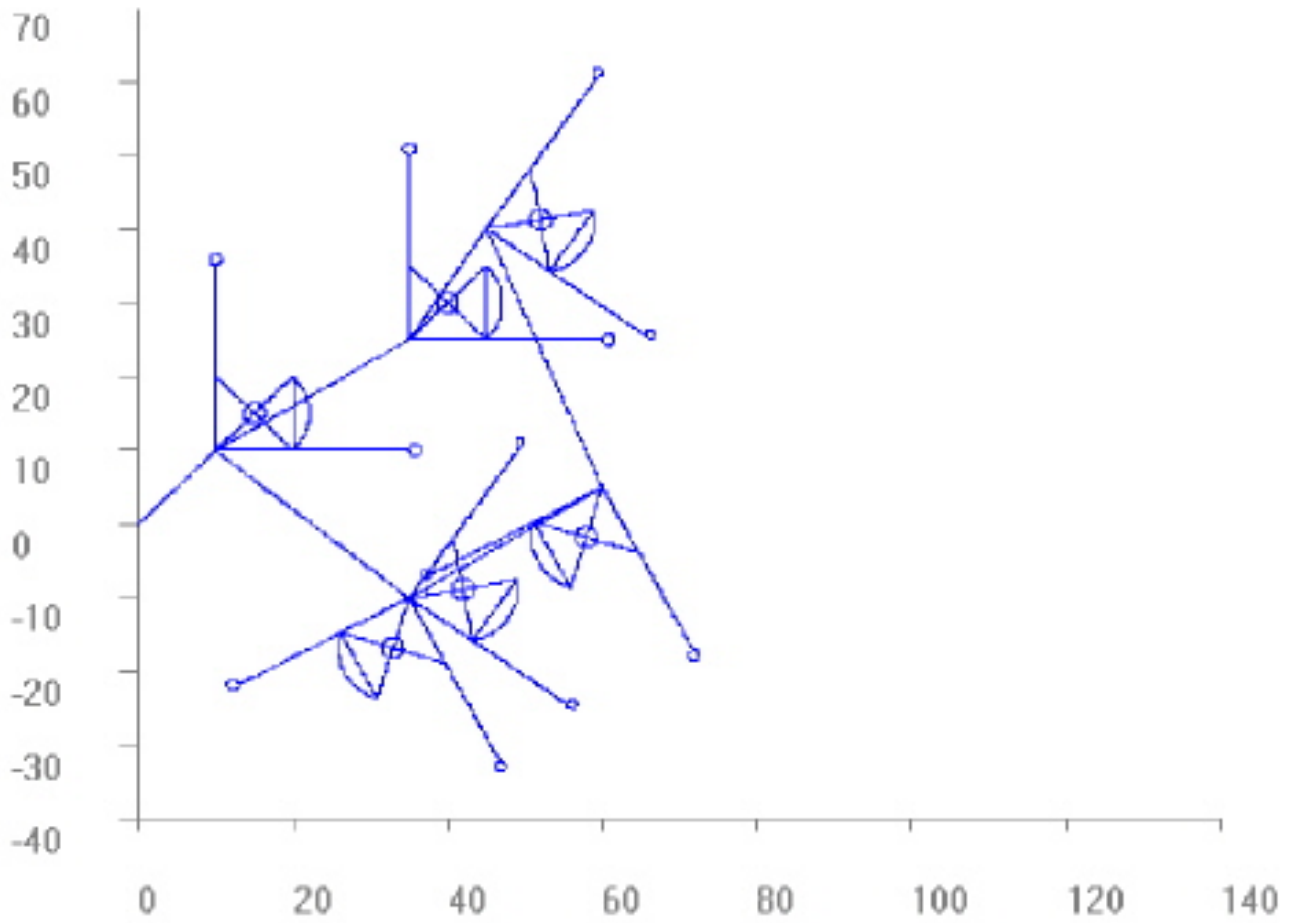
Test of relative and absolute programming:

```
%L contour_1
N1 G1 G91 (all positions with G91)
N2  X20
N3      Y20
N4  X20
N5      Y20
N6  X20
N7      Y20
N8  X20
N9      Y20
N10 X20
N11     Y20
N12         X5
N13         Y-3
N14         Y3
N15         X-5
N16 G90 X0
N17     Y0
N18         X5
N19         Y-3
N20         Y0
N21         X0
#MSG SYN["contour_1 finished"]
M17
%L contour_2
N100 G1 (same contour, X with G91, Y with G90)
N101 G91 X20
N102 G90 Y10 (transl. offset in Y is 10)
N103 G91 X20
N104 G90     Y30
N105 G91 X20
N106 G90     Y50
N107 G91 X20
N108 G90     Y70
N109 G91 X20
N110 G90     Y90
N111             G91 X8
N112             G91 Y-4
N113             G91 Y4
N114             G91 X-8
N115 G90 X0
N116 Y0
N117     X8
N118     Y-4
N119     Y0
N119     Y0
N120     X0
#MSG SYN["contour_2 finished"]
M17
%L contour_3
N200 G1 (same contour, Y with G91, X with G90)
N201 G90 X0 (transl. offset in X is 20)
N202 G91     Y20
N203 G90 X20
N204 G91     Y20
N205 G90 X40
N206 G91     Y20
N207 G90 X60
N208 G91     Y20
N209 G90 X80
N210 G91     Y20
N211             G91 X11
N212             G91 Y-5
N213             G91 Y5
N214             G91 X-11
N215 G90 X0
N216 Y0
N217     X11
N218     Y-5
N219     Y0
N220     X0
#MSG SYN["contour_3 finished"]
M17
%L contour_4
N300 G1 G90 (same contour with G90)
N301 X0 (transl. offset in X is 20)
N302 Y10 (transl. offset in Y is 10)
N303 X20
```

```
N304      Y30
N305 X40
N306      Y50
N307 X60
N308      Y70
N309 X80
N310      Y90
N111              G91 X14
N312              G91 Y-6
N313              G91 Y6
N314              G91 X-14
N315 G90 X0
N316 Y0
N317      X14
N318      Y-6
N319      Y0
N320      X0
#MSG SYN["contour_4 finished"]
M17
%ang5.nc
N5001 G0 G90 X0 Y0 F5000

N501 #ROTATION ON [ANGLE 0 CENTER1 20 CENTER2 10]
(Note: with angle != 0 the contours are
(     not congruent because of difference
(     of absolute and incremental movement!)

N502 #ROTATION ON
N503 LL contour_1
N504 #ROTATION OFF
N5002 G0 G90 Y0
N5003        X0

N505 #ROTATION ON
N506 LL contour_2
N507 #ROTATION OFF
N5004 G0 G90 Y0
N5005        X0

N508 #ROTATION ON
N509 LL contour_3
N510 #ROTATION OFF
N5006 G0 G90 Y0
N5007        X0

N511 #ROTATION ON
N512 LL contour_4
N513 #ROTATION OFF
N5005 G0 G90 Y0
N5006        X0


N210 M2
```
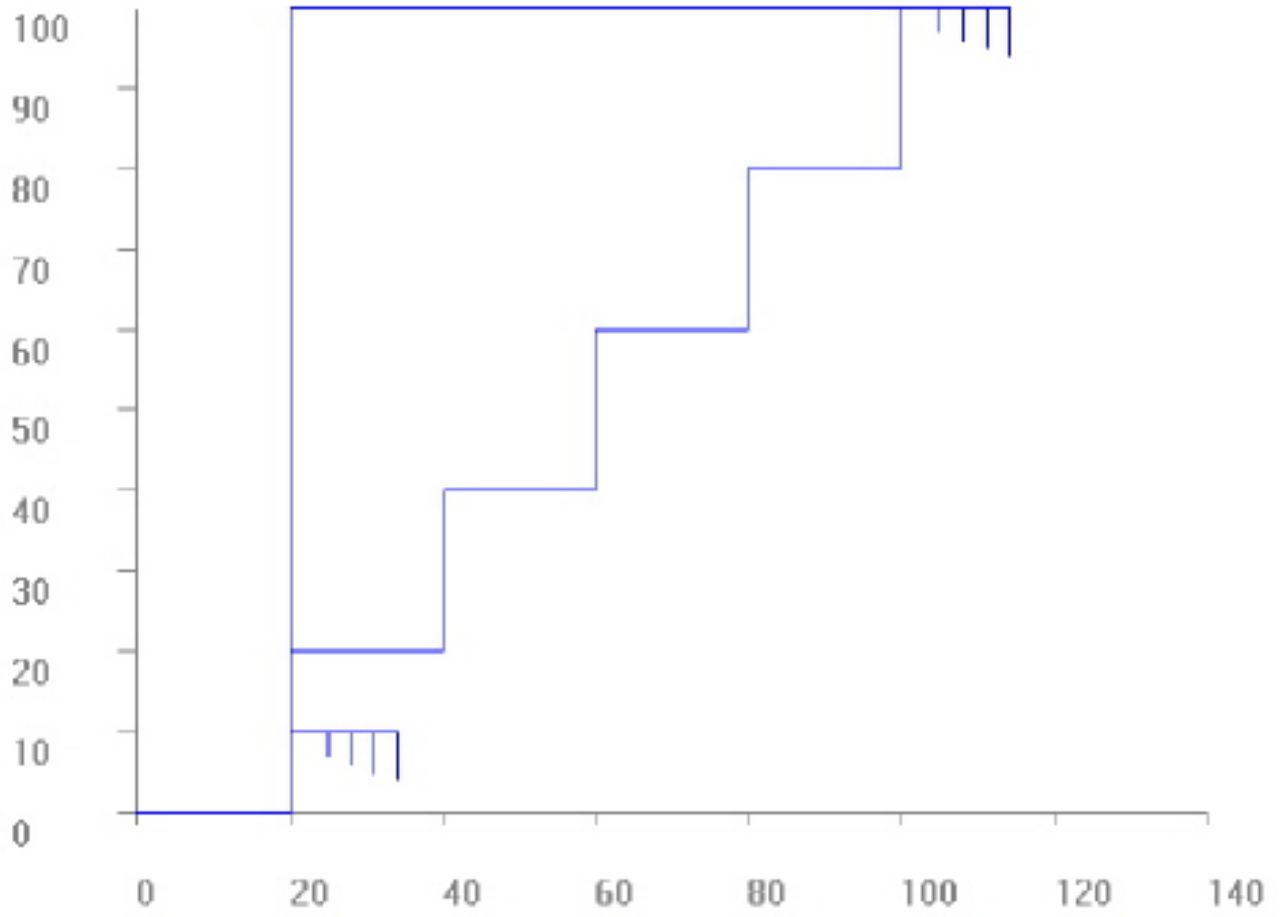
After selecting valid rotation angular. The offsets for the rotation point may firstly be considered with the first absolute programming (G90).

```
%ang6.nc
N10  G90 X0Y0Z0 G1 F200
N20 #ROTATION ON [ANGLE 30 CENTER1 10 CENTER2 10]
N30  X0 Y0
N40 G3 I1 J1 F500
N50  G01 X10
N60      Y10
N70 G90 X0
N80 G90 Y0
N90 X10 Y10
(New rotation parameters.
(Note: Centre offset has no effect until an absolute (G90) position
(has been programmed. However, the angle is effective.
N100 #ROTATION ON [ANGLE 10 CENTER1 -10 CENTER2 0]
N110 G3 I1 J1 F500
N120 G01 G91 X10
N130 G91 Y10
N140 G91 X-10
N150 G91 Y-10
(Make the new centre effective by first absolute position:
N200 G90 X0 Y0
N210 G3 I1 J1 F500
N220 G01 X10
N230 Y10
N240 X0
N250 Y0

M30
```
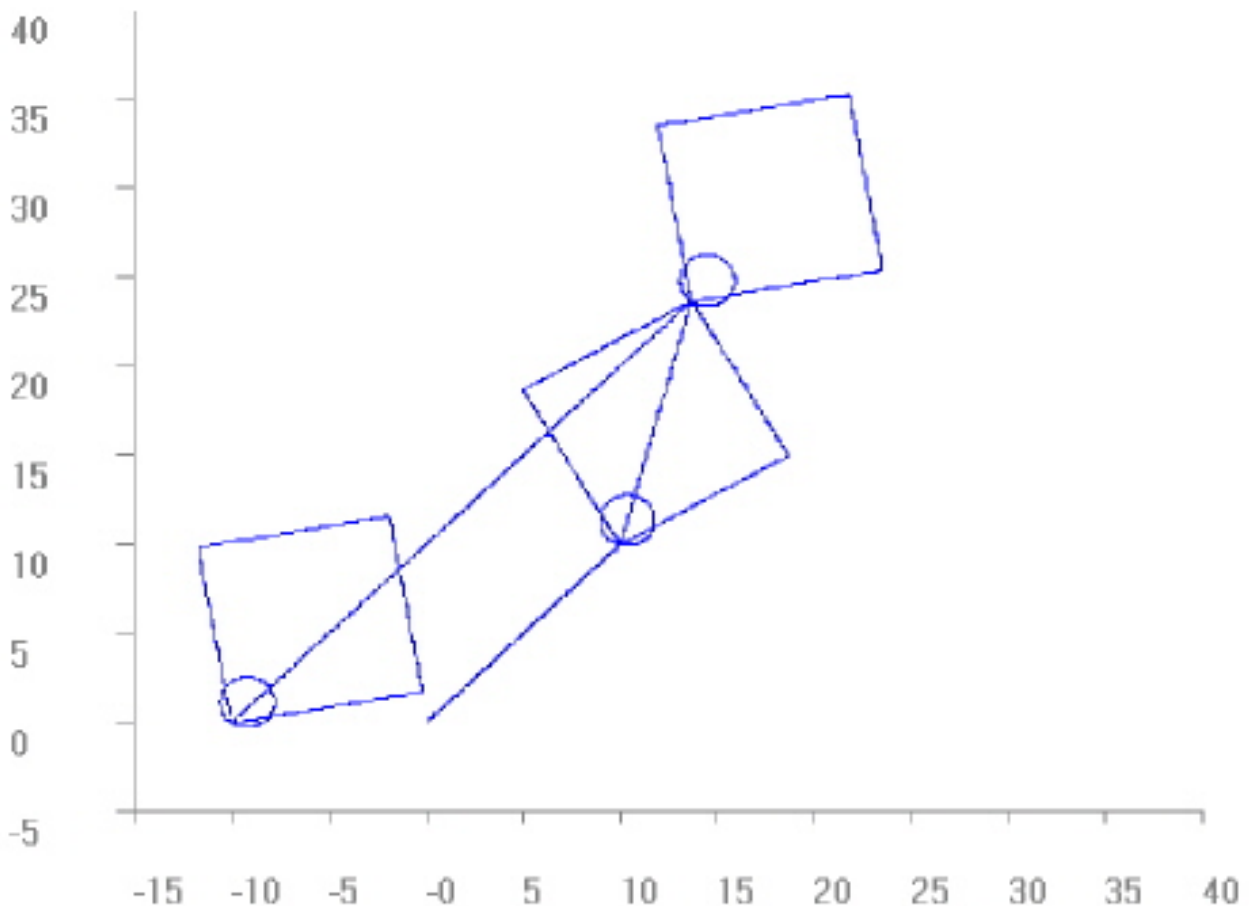
Transforming the absolute or relative programmed circle centre point:

```
%ang_cent.nc
N10 F2000 G53
N11 G0 X0 Y0 G90
(--------------------------------------------------------------------)
(4 times the same circle with different programming of circle centre point)
(--------------------------------------------------------------------)

N12 G0 X0 Y0 G90
N13 Y50
N14 X-75
N15 G3 X-75 Y-50 G161 I-75 J0 (absolute centre)
N16 G0 X0 Y0 G90
N17 Y50
N18 X-75
N19 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N20 G0 X0 Y0 G90
N28 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 -75]
(--------------------------------------------------------------------)
(The same with LIN and ANG offset active (ED=0))
(--------------------------------------------------------------------)

N80 G0 X0 Y0 G90
N90 Y50
N100 X-75
N110 G3 X-75 Y-50 G161 I-75 J0 (absolute centre)
N120 G0 X0 Y0 G90
N130 Y50
N140 X-75
N150 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N360 G0 X0 Y0 G90


(--------------------------------------------------------------------)
(The same rotated by 50° (unnecessary I / J omitted) )
(--------------------------------------------------------------------)
N370 #ROTATION ON [ANGLE 50]
N380 G0 X0 Y0 G90
N390 Y50
N400 X-75
N410 G3 X-75 Y-50 G161 I-75 (J0) (absolute centre, not prog. is 0)
N420 G0 X0 Y0 G90
N630 Y50
N640 X-75
N650 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N655 G0 X0 Y0 G90


(--------------------------------------------------------------------)
(The same rotated by 95° )
(--------------------------------------------------------------------)
N660 #ROTATION ON [ANGLE 95]
N670 G0 X0 Y0 G90
N680 Y50
N690 X-75
N700 G3 X-75 Y-50 G161 I-75 J0 (absolute centre)
N710 G0 X0 Y0 G90
N730 Y50
N740 X-75
N750 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N760 G0 X0 Y0 G90
M30
```
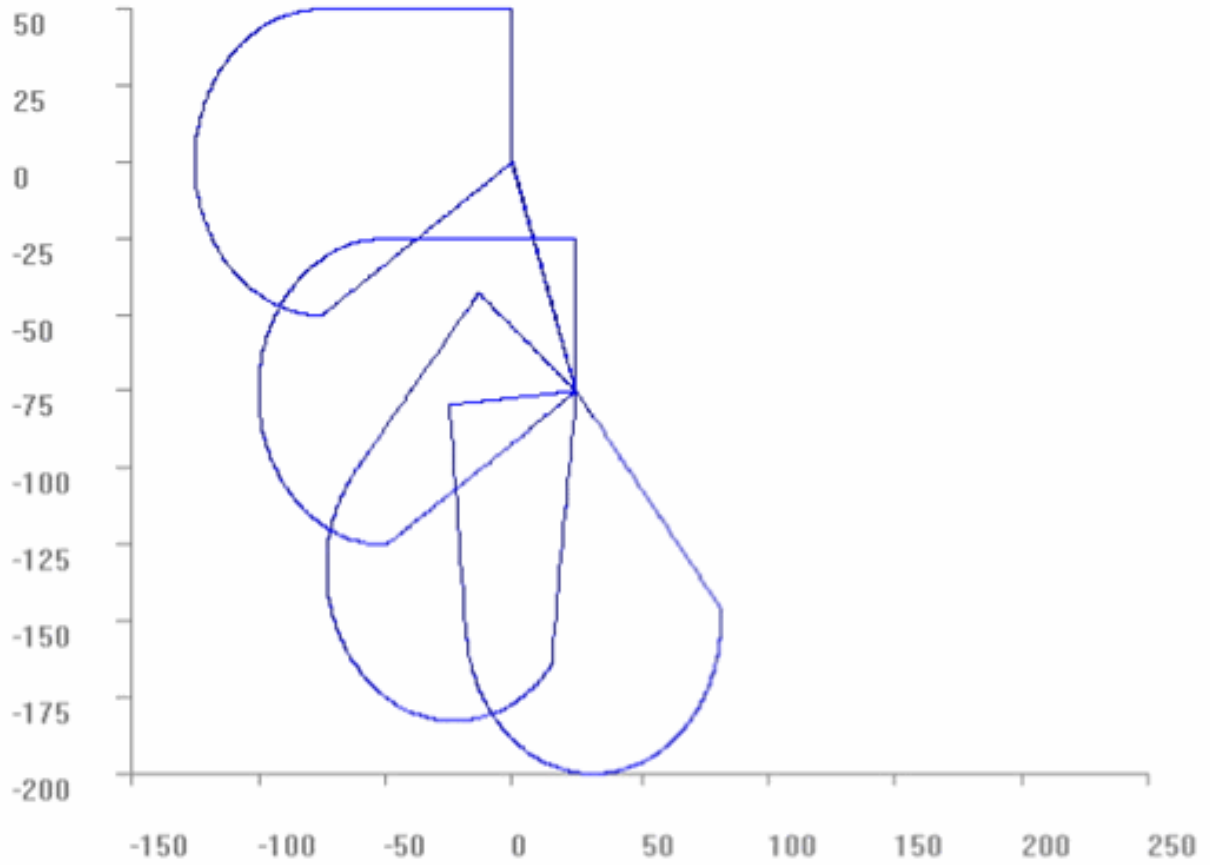
12.17 Automatic axis tracking (C axis tracking) (#CAXTRACK)

Machining operations, such as cutting various materials, require tool guidance generally associated with the C axis so that the tool is always aligned tangentially relative to the path followed.

It must be noted that the tangent is not unique at each point of the path (break points). Consequently, a solution requires strategies for handling block transitions which do not feature constant tangents.

One typical application is the technology sector for glass cutting. This involves machining level contours with the aid of cutting tools in the form of carbide metal cutting wheels using CNC machines. The flat workpiece is scored slightly at the machining point in accordance with the programmed contour (closed contour, e.g. ellipse). The required contour can then be broken out of the glass workpiece.

Starting point
C-axis in basic orientation
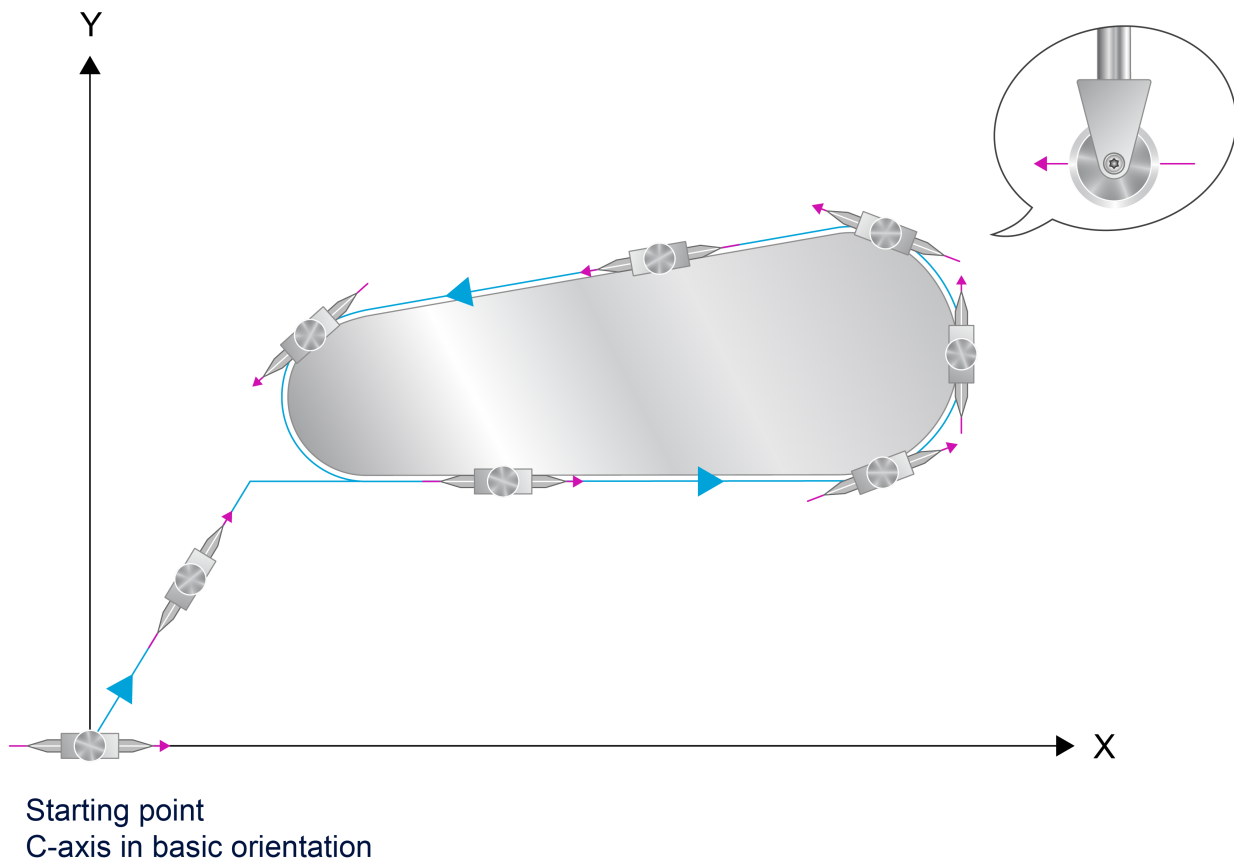
Fig. 114: Tracking the rotary C axis tangentially relative to the x-y contour

The C axis can also be guided tangentially relative to the path by explicit programming. However, the NC commands described below simplify programming considerably.

```
Syntax:
```
#CAXTRACK ON [**[** [**ANGLIMIT**=..] [**OFFSET**=..] [**OPTALIGN**=..] [**ROTMODE**=..] [**SCALEFACT**=..]

                 [**AX**=*<axis_name>* | **AXNR**=..] [**START_STROKE**] [**SYMMETRIC_TOOL**] **]**

#CAXTRACK OFF [**[ANGPOS=..]**]

ANGLIMIT=..	Limit angle in [°].
	This parameter is only considered for non-tangential continuous contour sections. For example, tangential continuous contour sections are created by the contouring function G61.
	If the angle between the tangents to the contour exceeds the limit angle on block transition, the contouring motion is halted and the dressing movement is performed by an inserted motion block at rapid traverse velocity. In this case, the inserted

movement forms one unit with the following second block. This means in particular that PLC synchronisation events in conjunction with the motion (M functions etc.) are possible only before or after this motion unit.

If the transition angle to second block is smaller than the limit angle, the dressing motion starts immediately on transition to the second block. In general limited axis acceleration in general causes a lower feedrate at the block transition. If this reaction is not acceptable, the axis dynamic monitoring of the tracking axis can be excluded. (e.g. G116 C1 [▶ 177])

OFFSET=..	Angle offset in [°].
	This specifies an angle offset in order to orient the tool opposite to the tangent against the contour.
OPTALIGN=..	When selected, orientation is automatically optimised if the orientation distance is greater than the defined angle value 'OPTALIGN' in [°].
	This parameter is only considered for active automatic aligning according to P-CHAN-00101 and rotary linear axes with limited motion range (no modulo axis). It is only effective during the automatic orientation process on the first contour element.
	After selecting the tangential tracking function with automatic orientation, the position of the tracking axis lies without any offset in the range of -180 to +180°. The parameter allows that the position of the tracking axis is considered during the automatic orientation process <u>before</u> the tangential tracking function is selected.
	The function is only useful if the tracking axis has approximately the correct position to the first contour element before automatic orientation is selected. However, for example, it has a pre-orientation of +-360°. If the internally calculated orientation angle exceeds the programmed angle 'OPTALIGN', alternative solutions for the orientation angle are taken into consideration. Then the smallest orientation distance of the solutions determines the real orientation angle. (*)

ⓘ (*) For modulo axes, the automatic orientation process is always executed on the shortest way.

ROTMODE=..	Boolean value indicating assignment of the tracking axis:
	0: The tracking axis is an axis in the tool (default).
	1: The tracking axis is an axis in the workpiece.
	The tool axis always has to be aligned vertically to the XY plane. Alternatively, the position of the tracking axis can also be defined in the channel parameters (P-CHAN-00185).

SCALEFACT=..	Scaling factor for the tracking angle in the range >0.0... ≤1.0.
	For values outside the permissible limits, the scaling factor is set to 1.0 (default). For special applications only.
ANGPOS=..	Position at deselection in [°].
	When deselected, the tracking axis can also be positioned. Rotary axes are positioned on the shortest way.
AX=*<axis_name>*	Specify the tracking axis by name. The axis is valid until program end (M30).
	After program start or if no tracking axis is programmed, the default axis defined in the channel parameter P-CHAN-00095 is valid.
AXNR=..	Specify the tracking axis by logical axis number, positive integer. The axis is valid until program end (M30).
	After program start or if no tracking axis is programmed, the default axis defined in the channel parameter P-CHAN-00095 is valid.
START_STROKE	Executes a reduced orientation sequence once, defined by #STROKE DEF CAXTRACK ALIGN BLOCK with #CAXTRACK ON. If automatic orientation is used with P-CHAN-00101, the orientation motion is additionally executed at the beginning of the orientation sequence. START_STROKE is non-modal and must be programmed for each time #CAXTRACK ON is activated. START_STROKE is only relevant when an orientation sequence is used.
SYMMETRIC_TOOL	Keywords activate tracking with a symmetric tool that can be moved in the material both in the 0° and in the 180° directions. The orientation motions are always executed on the shortest possible path and the flute side is selected automatically. SYMMETRIC_TOOL is non-modal and must be programmed for each time #CAXTRACK is activated

The tracking axis used must be an additional axis. This may not lie on a main axis index.

Automatic tracking of the axis is performed with the correct sign relative to the last position depending on the resulting contour transition angle.

Depending on the P-CHAN-00101 parameter, the **alignment of the tracking axis** in the required orientation (generally parallel to the contour) is as follows:

- Programmed aligning before selecting automatic tracking. No check of the position is conducted. The current angle position is frozen and the tracking axis is aligned at that angle.
- Automatic tangential alignment of the tracking axis (P-CHAN-00101) to the first programmed contour element when automatic tracking is selected.

Caution: The activation of automatic tangential alignment is absolutely necessary if there are programmed contour sections with polynomials (e.g. G261).

Tracking commences with the transmission of the first to the second relevant motion block after i is activated by #CAXTRACK ON. Automatic axis tracking operates in the main plane of circular interpolation (1st + 2nd main axes). This must be defined before activation (G17 / 18 / 19, #PUT AX / #CALL AX / #SET AX).

If the tracking axis already has the correct orientation when selected, the path motion is continued without interruption to the first relevant motion block provided by the parameter P-CHAN-00109.

If the contouring function (G261) is already active and parameter ANGLIMIT > 0, the following condition is required for a smooth motion transition:

- Contour elements before and after #CAXTRACK ON [..] must be tangential linear blocks to each other.

● Do not operate an active tracking axis in synchronous mode as a slave axis.
i

Automatic axis tracking (C axis tracking)

```
Example 1: Selecting axis tracking
N10 G00 G90 X0 Y0 Z0 C0
N20 X5 Y5 C45        ;straight line 45° to the X axis, tracking axis C
                     ;aligned parallel to the contour
N20 #CAXTRACK ON [ANGLIMIT 3, OFFSET 0] ;Enable axis
                                  ;tracking, limit angle 3°
                                  ;Angular offset 0°
N30 X10 Y10        ;Primary motion block, C axis is
                    ;already aligned
N40 X20            ;Angle to previous block: -45° >
                   ;Limit angle -> Block is inserted: End
                   ;position of C = 0
N50 M99 X30        ;If M function synchronisation before
                   ;motion > First synch., then motion C
                   ;at 0, then X at 30.
                   ;If sync. After block > motion C at 0
                   ;then X at 30, then sync.
N60 X40            ;C axis angle 0°
N70 X30            ;C axis angle 180°
N80 Y0            ;C axis angle -90°
N90 #CAXTRACK OFF   ;Disable axis tracking
M30
```

```
Example 2: Couple a slave axis (C2) to master tracking axis (C)
N20 G00 X0 Y0 Z0 C0 A0 C2=0 A2=0
N50 #SET AX LINK[1, C2=C]
N70 #ENABLE AX LINK[1]
N140 G01 X0 Y0 Z0 A0 C0 F2000
N170 #CAXTRACK ON [AX=C ANGLIMIT 0.1]
N190 LL SUB_1
N220 #CAXTRACK OFF
N250 #DISABLE AX LINK[1]
M30
```

12.18 User-defined error output (#ERROR)

The NC command #ERROR allows the output of user-defined error messages which are further processed by the higher-level GUI (GUI = Graphical User Interface). Additional parameters offer the option of specifying the error more precisely.

The error number (ID) and error message are assigned in a user-defined file (FCT-M7// Outputting user error messages). The memory location (path) and filename are entered in the parameter P-STUP-00169.

`Syntax:`

#ERROR [[ID=..] **[RC**=..] **[MID**=..] {**PV**<*i*>=..} {**PM**<*i*>=..} {**PIV**<*i*>=..} **]**

ID=..	Error number:
	1...1000: The numerical value determines the user-specific error number to be output.
RC=..	Error remedy class:
	0: Warning, no transition to error state. Program execution is continued.
	2: Error, transition to error state. Can be cleared with NC–RESET.
	7: Fatal error, transition to 'system error' state. Requires controller restart.
MID=..	Multiple ID. Counter acts as a distinguishing feature if the #ERROR command with the same error number (ID) is used several times in an NC program. MID must be a positive integer.
PV<*i*>=..	Max. 5 (1 <= i <= 5) user-specific numerical values (PV1...PV5) in real format can also be output in the error message.
PM<*i*>=..	The maximum of 5 (1 <= i <= 5) PM parameters (PM1...PM5) specify the meaning of the PV parameters more precisely.
	0: IGNORE, value has no meaning
	1: Limit value
	2: Current value
	3: Error value
	4: Expected value
	5: Corrected value
	6: Logical axis number
	7: Drive type
	8: Logical control element number
	9: State
	10: Transition
	11: Sender
	12: Class
	13: Instance
	14: Identification number
	15: State
	16: Ring number
	17: Block number
	18: Lower limit value
	19: Upper limit value
	20: Initial value
	21: Final value
PIV<*i*>=..	The maximum of 4 (1 <= i <= 4) PIV parameters (PIV1...PIV4) transfer additional information in real format.

For non-programmed parameters, the following default values are valid:

ID	1
RC	0
MID	0
PV1...PV5	0.0
PM1…PM5	1
PIV1...PIV4	0.0

User-defined error output

```
; -------
; Output of warning with ID 100, multiple identifier 10
#ERROR [ID100 RC0 MID10]
; ..
; -------
; Output of warning with ID 455 with parameter
; Error 455 with parameters
; Parameter 1 – current value is 1
; Parameter 1 – incorrect value is 4.999
#ERROR [ID455 RC2 PV1=5 PV2=4.999 PM1=2 PM2=3]
; ..
; -------
; Fatal error 999
#ERROR [ID999 RC7]
```

12.19   Time measurement (#TIMER)

The NC command #TIMER offers the option of time measurement in the NC program. The time recorded is represented in the unit milliseconds (ms).

NOTICE
Time counters are provided **cross-channel (global)**. For example, this permits the measurement of signal propagation times between channels.
For parallel independent time measurements in different channels, make sure that different counter numbers (IDs) are used. Otherwise, the measurements will influence each other.

```
Syntax:
```
#TIMER *<action>* [*<mode>*] **[ID=..]**

<action>	Determines the action with the designated counter (ID).
	START: Starts the designated counter (ID).
	STOP**:** Stops the designated counter (ID).
	READ**:** Reads out the designated counter (ID).
	The timer count is latched and saved to the assigned **V.G.TIMER[ID]** variable in milliseconds (ms).
	CLEAR**:** Resets <u>and</u> stops the counter (ID).
	The assigned V.G.TIMER variable is not deleted but is retained until the next READ action of the related counter.

NOTICE
The timer function records a maximum of 1193 hours.

<mode>	Synchronisation mode:
	---: Time measurement asynchronous relative to the interpolator at decoding level (initial state). Time measurement starts directly after decoding.
	SYN: Time measurement at interpolator level. The designated counter is set synchronous to the machining operations of the NC machine. The synchronous read function (<SYN>) in the interpolator interrupts decoding until the timer count at decoder level is adopted in the timer variable.

i   In order to measure program run-times, timers must always use the keyword SYN.

ID=..   Counter number:

0...127: A maximum of 128 channel global counters can be programmed. However, only one counter (ID) can be programmed per timer command.

Time measurement

```
:
#FILE NAME[ MSG="C:\timer.txt" ] Filename for time logging
:
#TIMER START [ID=10]                Start timer 10 (decoder level)
#TIMER START SYN[ID11]               Start timer 11 (interpolator level)
:
:
#TIMER READ [ID10]              Save timer count in V.G.TIMER[10]
#TIMER READ SYN [ID11]          Save timer count in V.G.TIMER[11]
#MSG SAVE["T10 = %d",V.G.TIMER[10]] Log timer count to file
#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
#TIMER STOP [ID10]              Stop timer 10
#TIMER CLEAR [ID10]              Reset timer 10
:
:
#TIMER READ SYN [ID11]          Save timer count in V.G.TIMER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
:
:
#TIMER READ SYN [ID11]          Save timer count in V.G.TIMER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
:
:
#TIMER READ SYN [ID11]          Save timer count in V.G.TIMER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
#TIMER STOP SYN [ID11]          Stop timer 11
#TIMER CLEAR SYN [ID11]         Reset timer 11
:
:
:
#TIMER START [ID=10, ID11]      Error, only one counter per timer
                                command permissible!
:
:
```

12.20 Definition of feed axes (#FGROUP, #FGROUP ROT, #FGROUP WAXIS)

#FGROUP defines the axes to which the programmed feedrate (F word) refers. For axes programmed in the #FGROUP command, a path in space is defined for execution at the programmed feedrate. All other axes are treated as tracking axes and they then reach their target points simultaneously with the path axes.

A characteristic feature of a path axis is that the distance to be moved is included in the feedrate. On the other hand, the tracking axis distance to be moved has no direct influence on path velocity.

```
Syntax of Defining feed axes:
```
#FGROUP [*<axis_name>* {,*<axis_name>* } **]**

| *<axis_name>* | Name of the axes which are members of the feed group |

```
Syntax of Selecting default setting:
```
#FGROUP

If no feed axes are programmed, the default setting defined in the channel parameters P-CHAN-00096 and P-CHAN-00011 are valid. If no feed axes are configured there, all main axes (Index 0, 1, 2) automatically form the channel feed group. This is indicated by the message 21209.

With linear interpolation, feed axes may be given any definition:

With circular and polynomial interpolation, the following exceptions apply:

- With circular interpolation, either all main axes must be feed axes or all defined feed axes must be tracking axes.
- With polynomial interpolation, all main axes form the feed group independent of the #FGROUP command. An exception to this is polynomial contouring in DIST_SOFT mode. Here the programmed #FGROUP is effective.

Definition of feed axes

```
N10 #FGROUP [X,Y]         ;X and Y are feed axes
N20 #FGROUP [X,Y,U,V]     ;X,Y,U and V are feed axes
:
N50 #FGROUP [A]           ;Tracking axis A is feed axis
N60 #FGROUP [X,Y,U,V,A,B] ;X,Y,U,V,A and B are feed axes
:
N100 #FGROUP              ;Feed axes acc. to default settings
:                         ;in the channel parameters
N999 M30
```

In the default setting, all feed axes defined with #FGROUP are equally weighted when determining the effective feed rate. By optionally specifying a factor when defining the feed axes, a weighting and therefore its effect on determining the effective feed can be defined for each axis. The extension of #FGROUP is described below.

```
Syntax of Determining feed axes with feed factors:
```
#FGROUP [*<axis_name>*=.. { *,<axis_name>*=..} **]**

<axis_name>=..	Name of the axes which form the feed group with associated feed factors. If no factor is specified for an axis, the factor 1.0 will apply.
	If an axis is fetched by axis exchange to the channel, its feed factor is initialised with a value of 1.0..
	The permissible value range for feed factor is [0.001..1000]

```
Syntax of Selecting default setting, all feed factors to 1.0:
```
#FGROUP

Definition of feed axes with feed weighting

```
N10 #FGROUP [X= 0.5, Y= 0.75]; X and Y are feed axes with
                          ;changed feed factors
:
N50 #FGROUP [A=0.85]          ;Tracking axis A is feed axis
                          ;with changed feed factor
N60 #FGROUP [X,Y,U,V,A=0.85,B=1.7] ;X,Y,U,V are feed axes
                              ;with feed factor 1.0
                              ,A and B are feed axes
                              ;with changed feed factors
:
N100 #FGROUP            ;All feed axes with feed factor 1.0
:
N999 M30
```

In order to machine cylindrical workpieces on a rotary workpiece axis, the real programmed feedrate [mm/min] should act at the tool contact point. This can be ensured either by selecting a suitable kinematic transformation (e.g. lateral surface transformation) or by using the command #FGROUP ROT[...]. After programming this command, the feedrate of the rotary axis in [°/min] is recalculated depending on the reference radius. When the rotary axis is programmed alone or together with linear axes, the required programmed feedrate is obtained at the reference radius.

`Syntax of Adapting feed for a rotary axis`

#FGROUP ROT [AX=*<axis_name>* **REF**=.. **]**

AX=*<axis_name>*   Name of the axis on which the reference radius is to act.

REF=..   Effective reference radius of the rotary axis in [mm, inch].

`Syntax of Deselecting adapting feed for a rotary axis`

#FGROUP

NOTICE
No check is made whether the axis "AX.." is really a rotary axis or not. The function can only be used for feed blocks (G01) and combined with G94.

> **i** Typically, this function is used for milling.
>
> Feedrate adaptations for turning work are programmed with G95 and G96.

Workpiece with reference radius R=10mm

```
N05 G00 C0
N10 G01 C180 F1000; Rotational speed of the workpiece 1000°/min
                  ;Feedrate at workpiece circumference 174.67 mm/min

N20 #FGROUP ROT[AX=C REF=10]
N30 G01 C360 F1000 ;feedrate at workpiece circumference 1000 mm/min
                  ;Rotational speed of workpiece 5727.6°/min
:
Nxx #FGROUP ROT     (Deselect)
:
N10 G00 X0 Y0 Z0
N15 #FGROUP ROT[AX=C REF=10] ;Feedrate at milling cutter contact point 1000 mm/min

N60 G01 G91 X10 C57.325 F1000 ;Diagonal on lateral surface
N70 G90 X0 C0
Nxx #FGROUP ROT              ;Deselect
:
```

Independent of the default setting in the channel parameters, the command #FGROUP WAXIS defines that the axis with the longest running time ("weakest axis") is moved automatically as the feed axis at the programmed feedrate (F word). All other axes are treated as tracking axes.

```
Syntax of Deselecting "weakest" axis as feed axis
```

#FGROUP WAXIS

```
N10 #FGROUP [X, Y]      ;X and Y are feed axes
N20 G00 X0 Y0
N30 #FGROUP WAXIS       ;Weakest axis is feed axis
N40 G01 F1000 X10 Y200  ;Y axis is axis with longest motion time
:
N999 M30
```

12.21 Adapt path dynamic limit values (#VECTOR LIMIT ON/ OFF)

ℹ️ As of Build **V2.10.1507.02** the command **#VECTOR LIMIT ON/OFF...** replaces the commands **#VECTORACC ON/OFF...** and **#VECTORVEL ON/OFF...** For compatibility reasons, the commands continue to be available but they should not be used in new NC programs.

The maximum permissible velocity, acceleration and deceleration on the path depend on the dynamic characteristics set in the axis-specific parameter lists and the programmed contour.

To ensure the best results in specific applications (e.g. high-intensity laser or plasma torch cutting processes), it should be possible to modify and adapt the dynamic characteristics on the path directly in NC program.

These path limit values can be adapted by the following NC commands during the dynamic phases of machining. They permit the activation/deactivation of self-defined limits and default limits in the NC program.

It is also possible to limit radial acceleration and radial jerk that occur in curved contour elements (polynomials, circles).

```
Syntax of Deselecting "weakest" axis as feed axis
```
#VECTOR LIMIT ON [[**ACC**=..] [**DEC**=..] [**RADIAL_ACC**=..] [**RADIAL_JERK**=..] [**JERK**=..]
[**TRANS_ACC**=..] [**VEL**=..] [**FEED**] [**RAPID**] { **** } **]**

or with default limits

#VECTOR LIMIT ON [[**ACC**] [**DEC**] [**RADIAL_ACC**] [**RADIAL_JERK**] [**JERK**]
[**TRANS_ACC**] [**VEL**] [**FEED**] [**RAPID**] { **** } **]**

or

#VECTOR LIMIT ON ALL

ACC=..	Path acceleration limit in [mm/min², inch/min²]. The unit can be switched by P-CHAN-00351 to [mm/sec², inch/sec²].
DEC=..	Path deceleration limit in [mm/min², inch/min²]. The unit can be switched by P-CHAN-00351 to [mm/sec², inch/sec²].

ℹ️ DEC is effective if the selected slope profile allows a separate definition of acceleration and deceleration parameters (e.g. slope type TRAPEZ [▶ 357]).

RADIAL_ACC=..	Radial acceleration limit in [mm/min², inch/min²]. The unit can be switched by P-CHAN-00351 to [mm/sec², inch/sec²]. The programming for this parameter is available as of Build V2.11.2033.05.
RADIAL_JERK=..	Limit for radial jerk in [mm/min², inch/min²]. The unit can be switched by P-CHAN-00351 to [mm/sec², inch/sec²]. The programming for this parameter is available as of Build V3.1.3076.02.

JERK=..	Limit for path jerk. The unit is always in [mm/sec³, inch/sec³] and cannot be switched over. The programming for this parameter is available as of Build V3.1.3079.12.
TRANS_ACC=..	Path acceleration limit in block transition in [mm/min², inch/min²]. The unit can be switched by P-CHAN-00351 to [mm/sec², inch/sec²]. The programming for this parameter is available as of Build V3.1.3072.02.
VEL=..	Velocity limit [mm/min, inch/min].
	If no limit values are programmed for the keywords ACC, DEC, RADIAL_ACC and VEL or the command #VECTOR LIMIT ON ALL is used, the default limit values from the channel parameter list are used (P-CHAN-00002, P-CHAN-00208, P-CHAN-00361, P-CHAN-00090). An internal limit value is set for RADIAL_JERK, JERK and TRANS_ACC.
	The command #VECTOR LIMIT OFF... causes a switch-over to the calculation of the dynamic limitation of the look ahead function. Switching can be programmed both for specific limits and for all limits.
FEED	Dynamic limitations are only effective for feedrate blocks (G01, G02, G03).
RAPID	Dynamic limitations are only effective for rapid traverse blocks (G00).
	If the keywords FEED and RAPID are not programmed or both keywords are programmed in combination, the dynamic limitations are effective for all motion blocks (G00, G01, G02, G03).
\	Separator ("backslash") for clear programming of the command over multiple lines.

Syntax of Deselecting adapting dynamic path limit values:

#VECTOR LIMIT OFF [[**ACC**] [**DEC**] [**RADIAL_ACC**] [**RADIAL_JERK**] [**JERK**]

[**TRANS_ACC**] [**VEL**] { **** } **]**

or

#VECTOR LIMIT OFF ALL

---

i The path dynamic limits are only used if they are smaller than the valid look-ahead limits. They have no influence on axis-specific motions such as homing, manual mode or independent axes and they act on both G01 and G00.

NOTICE
In connection with the channel parameter P-CHAN-00097 the user must take into consideration that machine deceleration during feedhold is also slower depending on the programmed limit.

---

i The path dynamic limits for G00 can also be influenced by a motion path depending on the weighting table defined in the channel parameters [1] [▶ 786]-7.

Adapting path dynamic limit values

```
%vec_limit
(Set dynamic limit data to specific values)
N10 #VECTOR LIMIT ON [ACC=3600000 DEC=4000000 VEL=3000 FEED]
N11 #VECTOR LIMIT ON [ACC=3600000 DEC=4000000 TRANS_ACC=3000000]
N12 #VECTOR LIMIT ON [ACC=3600000 VEL=3000 RADIAL_ACC=2000000 RAPID]
N13 #VECTOR LIMIT ON [ACC=3600000]
N14 #VECTOR LIMIT ON [DEC=4000000 VEL=3000 FEED RAPID]
N15 #VECTOR LIMIT ON [DEC=4000000]
N16 #VECTOR LIMIT ON [VEL=3000]

(Set dynamic limit data to default values)
N20 #VECTOR LIMIT ON [ACC DEC RADIAL_ACC TRANS_ACC VEL RAPID]
N21 #VECTOR LIMIT ON [ACC DEC RADIAL_JERK JERK]
N22 #VECTOR LIMIT ON [ACC VEL FEED]
N23 #VECTOR LIMIT ON [ACC]
N24 #VECTOR LIMIT ON [DEC VEL FEED RAPID]
N25 #VECTOR LIMIT ON [DEC]
N26 #VECTOR LIMIT ON [VEL]

(Mixed assignment of dynamic limit data)
N27 #VECTOR LIMIT ON [ACC=3600000 DEC]
N28 #VECTOR LIMIT ON [ACC VEL=3000]

(Set all dynamic limit data to default values)
N30 #VECTOR LIMIT ON ALL
(:= #VECTOR LIMIT ON [ACC DEC RADIAL_ACC RADIAL_JERK JERK TRANS_ACC VEL FEED RAPID])

(Set dynamic limit data by LOOK_AHEAD)
N40 #VECTOR LIMIT OFF [ACC DEC VEL]
N41 #VECTOR LIMIT OFF [ACC DEC]
N42 #VECTOR LIMIT OFF [ACC VEL]
N43 #VECTOR LIMIT OFF [ACC]
N44 #VECTOR LIMIT OFF [DEC VEL]
N45 #VECTOR LIMIT OFF [DEC]
N46 #VECTOR LIMIT OFF [VEL]

(Set all dynamic limit data by LOOK_AHEAD)
N50 #VECTOR LIMIT OFF ALL
(:= #VECTOR LIMIT OFF [ACC DEC RADIAL_ACC TRANS_ACC VEL])
N999 M30
```

12.22 Defining a minimum block transition velocity (#TRANSVELMIN ON/OFF)

At discontinuous block transitions (contour knee angles), path velocity is reduced to such an extent that the resulting axis accelerations do not exceed the limit values predefined in the axis parameters. From a technological aspect (e.g. flame or gas cutting), this velocity reduction at contour knee angles sometimes is undesirable. In this case, the following command defines a minimum velocity which may not be undershot at contour knee angles.

NOTICE
The programmed minimum velocity in the command acts only at discontinuous block transitions of circular and linear blocks. As a result, no contouring mode or spline interpolation may be selected in the NC program (e.g. G261, #HSC, G151).

```
Syntax:
```
#TRANSVELMIN ON [**[** *<min_vel>* **]**]
#TRANSVELMIN OFF

<min_vel>   Minimum block transition velocity in [mm/min, inch/min].

If no limit value is programmed after #TRANSVELMIN ON, the minimum transition velocity is set to 0. The command #TRANSVELMIN OFF causes a switch-over to the free calculation of the velocity limitation by the look ahead function.

The function works with linear and non-linear slopes but with a non-linear slope, jerk limitation for discontinuous block transitions with channel parameter P-CHAN-00009 (corr_v_trans_jerk=1) may not be active. In such cases, jerk limitation is prioritised.

The programmed minimum velocity is only valid until program end. It is set to zero at the next program start or RESET.

Pre-assignment in the channel parameter list is not possible.

12.23 Writing machine data (#MACHINE DATA)

i **The availability of this function depends on the configuration and on the version scope.**

This command permits a change in axis-specific machine parameters in the NC program. The new values are valid program global. They are overwritten by the next update of machine data lists at controller start-up.

i An active path interpolation is stopped until the new parameter is adopted and the value is effective. Any rotating spindles are **not** stopped.

Syntax:

#MACHINE DATA [*<mode>*] **[AX**<name> | **AXNR**=.. *<Param_ID>* =.. | *<Param_ID>{.<idx>}<value>* |

AXPARAM "<string>**"** [**WAIT**] **]**

<mode>	Synchronisation mode
	---: Synchronisation at decoding level (initial state)
	SYN: Synchronisation at interpolation (real time) level
AX=*<name>*	Name of the related axis or spindle where a new axis-specific parameter is to be written
AXNR=..	Logical axis number of the path axis or spindle where a new axis-specific parameter is to be written. Positive integer.

NOTICE
No plausibility check is conducted for the logical axis number. The operator is solely responsible for making the correct entries. A change to any parameter value with #MACHINE DATA causes the complete axis parameter list to be retransferred to the NC channel. Axis parameter values previously changed using other NC commands (e.g. software limit switches via G98/G99) are overwritten and are no longer valid in the NC channel.

<Param_ID> =..	Axis parameter in ISG notation (P-AXIS-xxxxx) with new value in the unit of the axis parameter list [AXIS].
<Param_ID> {.<idx>}<value>	With axis parameters that address an array, the corresponding element is written by the extended specification of point and index (e.g. P-AXIS-00209.0).

i **A slave axis in a hard gantry combination can only be addressed by the logical axis number.**

The following axis parameters are available via predefined keywords (Param_ID):

Param_ID	Meaning
P-AXIS-00001	Non-linear velocity profile: Axis acceleration at velocity increase
P-AXIS-00002	Non-linear velocity profile: Axis deceleration at velocity decrease
P-AXIS-00004	Acceleration in rapid traverse (G00)
P-AXIS-00005	Linear velocity profile: Acceleration step 1 in rapid traverse (G00)
P-AXIS-00006	Linear velocity profile: Acceleration step 2 in rapid traverse (G00)

P-AXIS-00008	Permissible axis dynamics: Maximum permissible axis acceleration
P-AXIS-00011	Linear velocity profile: Acceleration step 1 at feed interpolation (G01, G02, G03)
P-AXIS-00012	Linear velocity profile: Acceleration step 2 at feed interpolation (G01, G02, G03)
P-AXIS-00045	Minimum distance (safety distance) between two collision axes
P-AXIS-00056	Maximum difference after deselecting tracking mode
P-AXIS-00075	Gantry operation: Velocity for compensating for static offset between master and slave axes due to differences in measuring systems
P-AXIS-00099	Acceleration factor k_v for P positional control
P-AXIS-00103	Size of backlash
P-AXIS-00109	Maximum permissible velocity override for independent axis and spindles
P-AXIS-00151	Maximum transient time to reach the exact stop window
P-AXIS-00152	Absolute position of reference point
P-AXIS-00166	Remaining deviation for non-linear position lag monitoring
P-AXIS-00167	Factor for parameterising dynamic position lag monitoring
P-AXIS-00168	Maximum position lag
P-AXIS-00169	Minimum position lag
P-AXIS-00172	Type of position lag monitoring
P-AXIS-00195	Non-linear velocity profile: Ramp time for acceleration down-gradation
P-AXIS-00196	Non-linear velocity profile: Ramp time for acceleration up-gradation
P-AXIS-00197	Non-linear velocity profile: Ramp time for deceleration down-gradation
P-AXIS-00198	Non-linear velocity profile: Ramp time for deceleration up-gradation
P-AXIS-00200	Non-linear velocity profile: Ramp time in rapid traverse (G00)
P-AXIS-00201	Minimum permissible ramp time of the drive to limit axis jerk
P-AXIS-00208	Maximum feed of compensation motion after deselecting tracking operation
P-AXIS-00209	Rapid traverse velocity G00
P-AXIS-00211	Linear velocity profile: Changeover speed in rapid traverse (G00) between ramp 1 and ramp 2
P-AXIS-00212	Permissible axis dynamics: Maximum permissible axis velocity
P-AXIS-00216	Minimum permissible axis velocity for spindles. Below this velocity, the rotational speed monitor in the position controller supplies the "speed zero" state
P-AXIS-00217	Factor to calculate the actual rotational speed at which the "speed setpoint reached" state is signalled.
P-AXIS-00218	Minimum homing velocity
P-AXIS-00219	Maximum homing velocity
P-AXIS-00221	Linear velocity profile: Changeover speed in feed interpolation (G01, G02, G03) between velocity ramp 1 and velocity ramp 2
P-AXIS-00236	Size of control window for exact stop
P-AXIS-00414	Maximum position offset for distance control

AXPARAM "*\<string\>*"	Alternative syntax: Axis parameter with complete structure path and value in internal notation of the axis parameter list [AXIS] (see example). It is possible to write **all** axis parameters.
WAIT	This keyword may only be used in conjunction with the synchronised setting (SYN) of an axis parameter. If a WAIT is programmed, program decoding is also interrupted (implicit FLUSH) and the program waits until the new parameter is adopted and becomes effective in the NC channel.

BECKHOFF

Writing machine data

```
N10 G00 X100                    (Position at rapid traverse velocity acc.)
                                (to default setting after start-up)

N20 #MACHINE DATA SYN [AX=X P-AXIS-00209=80000]   (New rapid traverse velocity)

N30 ...     (The new rapid traverse velocity applies in the following program sequence).
:

Alternative with axis list notation:
:
N20 #MACHINE DATA SYN [AX=X AXPARAM="getriebe[i].vb_eilgang 80000"]
                                            (New rapid traverse velocity)
:

For a spindle axis:
Alternative with axis list notation:
N20 #MACHINE DATA SYN [AX=S P-AXIS-00109=1200]   (New velocity override)


Set a software limit switch and wait in the channel:
N20 #MACHINE DATA SYN [AX=X AXPARAM="kenngr.swe_pos 15000000" WAIT]


Set the rapid traverse velocity for gear stage 1 with index:
N20 #MACHINE DATA SYN [AX=X P-AXIS-00209.1=80000]
```

12.24 File operations

12.24.1 Definition of file names (#FILE NAME)

The command #FILE NAME defines file names in NC program file. The file names are used by specific NC functions, e.g. to create a report file or to call a NC program.

`Syntax:`

#FILE NAME [*<file_ID>*=**"**<filename>**"** { *<file_ID>*=**"**<filename>**"** } **]**

<file_id>	MSG: Name of the report file for #MSG SAVE.
	Default name is "*message.txt*".
	M6: Name of the global subroutine for implicit program call via M6 in the NC program. The name is valid until M30. M6 is <u>not</u> treated as an M function any more.
	The default name is set in P-CHAN-00118. M6 is always executed as the last action at block end.
	D: Name of the global subroutine for implicit program call by a D word in the NC program. The name is valid until M30.
	The default name is set in P-CHAN-00429. The subroutine call is always executed as the last action at block end.
	G80 – G89: Name of the global subroutines for implicit program calls at G80 –G89 in the NC program. The names are valid until M30.
	Default names are set in P-CHAN-00160 - P-CHAN-00169. G80-G89 is always executed as the last action at block end.
	G800- G819 or **G800- G839**: Names of the global subroutines in case of additional implicit program calls via G800 – G819 in the NC program. The names are valid until M30.
	Default names are set in P-CHAN-00187. A G800-G819** is always executed as the last action at block end.
	** Extended to 40 calls (G800 – G839) as of V3.1.3079.23

The equals sign (=) is optional.

The file names can be defined or changed in the NC program with #FILE NAME at any time.

Filenames are set to their default names at RESET and at NC program start.

BECKHOFF

Definition of filenames

```
%example1
N10 #FILE NAME[ MSG="prog_flow.txt" ]
N20 $IF V.E.PLC_START_HOME == 1
N30 G74 X1 Y2 Z3
N40 #MSG SAVE["Homing executed"]    ;Output in prog_flow.txt
N50 $ENDIF
;…
N120 #MSG SAVE["Roughing OK"]
;…
N950 #MSG SAVE["Finishing OK"]
N985 V.E.WP_CNTR = V.E.WP_CNTR+1
N990 #MSG SAVE["Workpiece No. %d OK", V.E.WP_CNTR]
M1000 M30
```

```
%example2
N10 #FILE NAME[ M6="tool_change.nc" ]
N20 G00 X100 Y100 Z0
N30 M6                              ;Call tool change program tool_change.nc
;…
M1000 M30
```

```
%example3
N10 #FILE NAME[ D="tool_data.nc" ]
N20 G00 X100 Y100 Z50
N30 D1          ;1:Request data of tool 1
                ;2:Implicit call of tool_data.nc
;…
M1000 M30
```

```
%example4
N10 #FILE NAME[ G80="g80_up_test.nc" ]
N20 G00 X100 Y100 Z50
N30 G80         ; Call subroutine g80_up_test.nc
;…
M1000 M30
```

```
%example5
N10 #FILE NAME[ G800="g800_up_test.nc" G815="g815_up_test.nc"]
N20 G00 X100 Y100 Z50
N30 G800         ; Call subroutine g800_up_test.nc
;…
N90 G815         ; Call subroutine g815_up_test.nc
;…
M1000 M30
```

12.24.2 Renaming a file (#FILE RENAME)

The command #FILE RENAME renames an existing file. All parameters must be specified. Omitting a parameter leads to a corresponding error message.

```
Syntax:
```
#FILE RENAME [PATHOLD="*<filename>*" **PATHNEW**="*<filename>*" **OVRMODE**=.. **]**

PATHOLD= "*<filename>*"	File to be renamed, possibly with directory specification. If this directory path or the file does not exist, the NC program is aborted with an error message.
PATHNEW= "*<filename>*"	New (destination) file, possibly with directory specification. If a different directory path is specified here than at PATHOLD, the file is moved to the directory with the new name, provided the directory exists. If this directory is not found, a corresponding error message is output.
OVRMODE=..	Boolean value indicating whether a file specified by PATHNEW should be overwritten provided it already exists. 0: File may not be overwritten. output of an error message 1: Existing file may be overwritten.

The equals sign (=) is optional.

ℹ The user must have write access to the directories PATHNEW and PATHOLD. Otherwise, renaming causes an error.

NOTICE
WRITE PROTECTION:
An error is generated if the file which is to be renamed is write-protected, it is an existing (destination) file and is protected.

NOTICE
RELATIVE DIRECTORIES:
If only the filename is specified at PATHOLD, a search is made for the file in the folders of the start-up/channel parameter list.
The search is for the sequence main program - subroutine - work directory.
If PATHNEW is specified as relative, the PATHOLD directory is used.

ℹ **In addition, the default directory for the file operation is used in TwinCAT. This setting is made in the System Manager.**

Renaming a file

```
%FileRename

N10 #FILE NAME[MSG="C:\Test.txt"] ;Create file
...
N40 #MSG SAVE["Write me into file"] ;Writes text to file

N60 #FILE RENAME[PATHOLD="C:\Test.txt" PATHNEW = "C:\NewName.txt" OVRMODE=1]
N70 M30
```

12.24.3 Deleting a file (#FILE DELETE)

The command #FILE DELETE is used to delete a file. The parameter must be specified. Otherwise, a corresponding error message is generated.

```
Syntax:
```
#FILE DELETE [PATH="*<filename>*" **]**

PATH=
"*<filename>*"

File to be deleted with directory specification.

The equals sign (=) is optional.

> **ℹ** The user must have write authorisation to the PATH Directory to be able to delete a file.

NOTICE
WRITE PROTECTION:
If the file is write-protect5ed, the error is generated by ID 21627.

NOTICE
RELATIVE DIRECTORIES:
If the PATH parameter is specified as relative, a search is made for the file in the folders of the start-up/ channel parameter list.
The search is for the sequence main program - subroutine - work directory.

> **ℹ** **In addition, the default directory for the file operation is used in TwinCAT. This setting is made in the System Manager.**

Deleting a file

```
%FileDelete

N10 #FILE NAME[MSG="C:\Test.txt"] ;Create file
...
N40 #MSG SAVE["Write me into file"] ;Writes text to file
…
N60 #FILE DELETE[PATH="C:\Test.txt" ] ;Delete file
N70 M30
```

12.24.4   Checking existence of a file (#FILE EXIST)

The command #FILE EXIST checks whether a file exists in the file system and can open it. After calling #FILE EXIST the call result is saved to V.G. variables. This is retained until the next time #FILE EXIST is called.

V.G.<var_name>	Meaning	Data type	Unit for input/ output	Permitted access Read / Write
FILE_EXIST	Result of data search. When the file is found, then 1	Boolean	0 , 1	R
FILE_EXIST_PATH_N AME *	Path with file name*	String	-	R
FILE_EXIST_DIRECT ORY *	Path without file name. The path ends with '\' *	String	-	R

* Paths that are entered in the table of search paths in the start-up list (P-STUP-00018) or channel list (P-CHAN-00401) .

```
Syntax:
```
#FILE EXIST [PATH="*<filename>*" **]**

PATH= "*<filename>*"	Name of the file whose existence is to be checked with or without directory specification.

The equals sign (=) is optional.

NOTICE
RELATIVE DIRECTORIES:
If the PATH parameter is specified as relative, a search is made for the file in the folders of the start-up/ channel parameter list.
The search is for the sequence main program - subroutine - work directory.

Checking the existence of a file

```
%FileExist

N010 #FILE EXIST[PATH = "C:\TestDir\test.nc"]

N030 $IF V.G.FILE_EXIST == TRUE
N040 #MSG ["FILE EXISTS"]
N050 $ELSE
N060 #MSG ["FILE DOES NOT EXIST"]
N070 $ENDIF

N090 M30
```

12.24.5   Create and manage backup files

Logging data

The user can write text to file in the NC program using #MSG SAVE […]. New data is appended to the file, causing the file size to gradually increase.

Defining size and number of log files

To check the size and access to large files, the user can limit the maximum number of lines in the file. When the maximum size is reached, the file is backed up automatically and the original file is deleted.

To track the chronology of file changes, several backups (log files) can be saved. The user can also define the maximum number of backups. When the maximum number of backup files is reached, the oldest back file is discarded.

The size and number of backup files is defined with the command #FILE NAME […]. This refers to the file which is written with #MSG SAVE.

```
Syntax:
```

#FILE NAME [BACKUP_LINES_MAX=.. **BACKUP_COUNT_MAX**=.. **]**

BACKUP_LINES_MAX=..	Number of lines in the file since controller start-up. If this is overshot, a backup is created. This applies to files last programmed with *<Number>*.
BACKUP_COUNT_MAX=..	Maximum number of backups saved in parallel. If the maximum number is reached when a new backup is created, the older backup is overwritten.
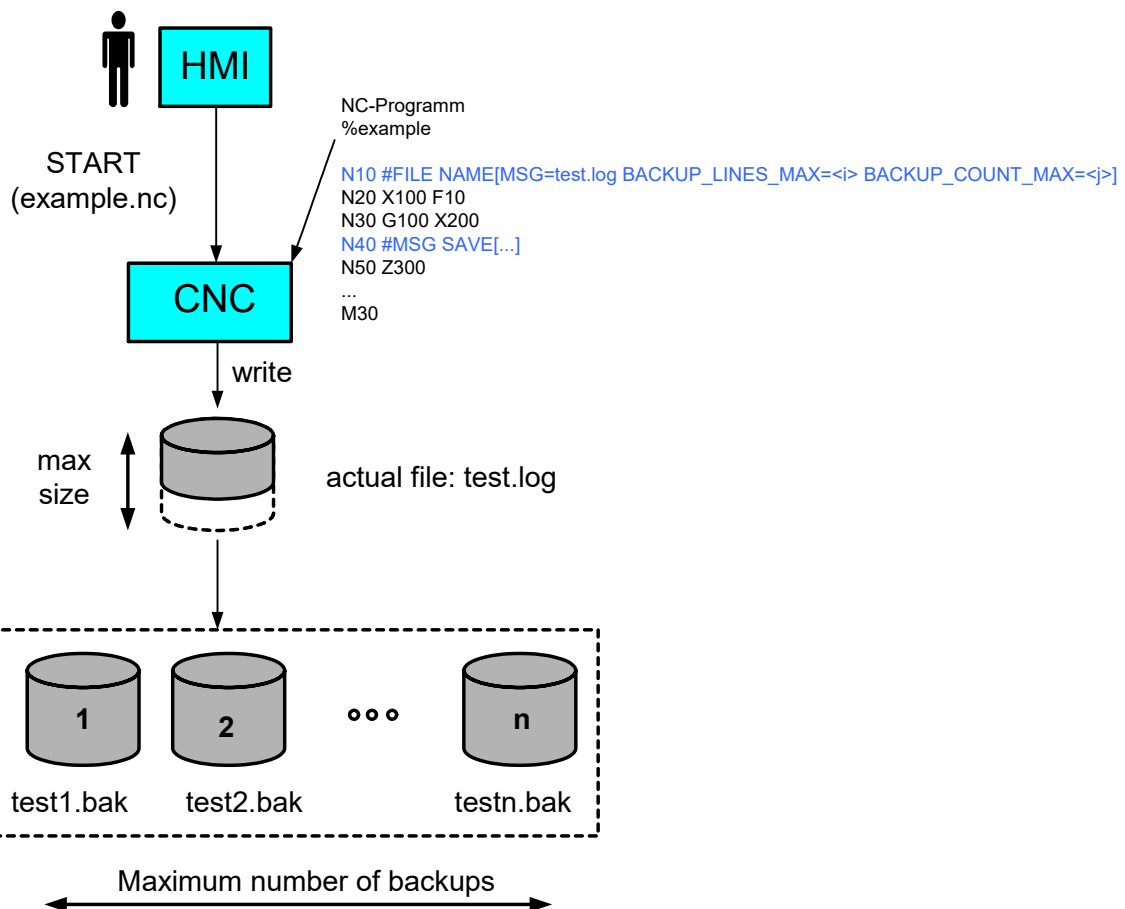
Fig. 115: Diagram of backup function
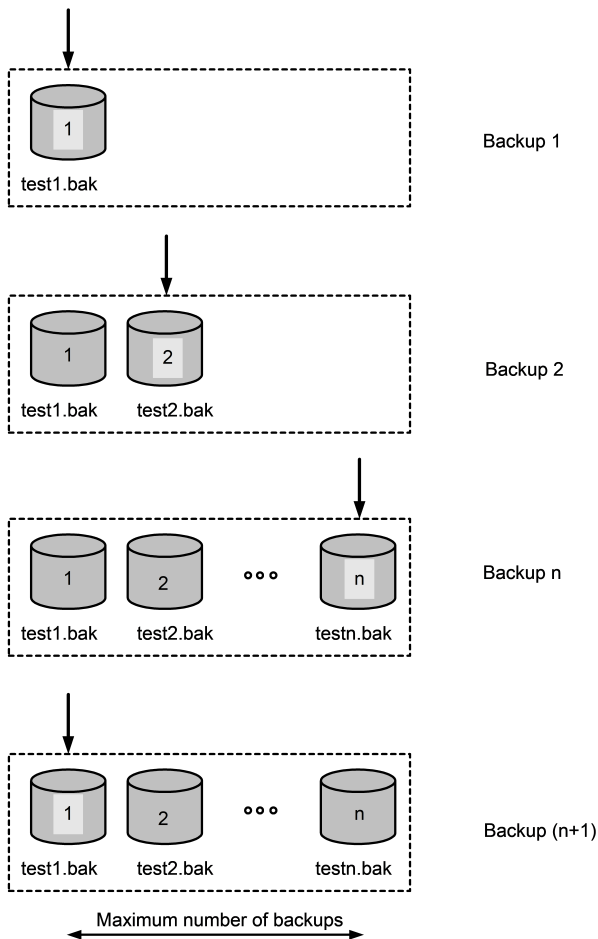
Fig. 116: Creating backup files

Activating and deactivating the backup function

The backup function is activated by specifying a maximum number of lines > 0 and a maximum number of files > 0.

Deactivation takes place if one of these parameters is programmed with 0 or none of these parameters is programmed or at program end M30.

ℹ The current maximum number of lines corresponds to the number of write accesses per #MSG SAVE command. In other words, the actual number of lines is not determined. This produces a difference in particular if a message containing several lines is written.

The original name of the backup file is specified by #FILE NAME [MSG=<name>]. The name of the backup consists of the original name, the count number <no> and the file extension .bak.

Example: Test.log (original file)

Test1.bak , Test2.bak , Test3.bak, etc.

Behaviour at controller restart

When the controller is started, back files may already exist. The first time a new backup is created, no search is made for the oldest backup. This means that after controller start, a new backup always starts with 1.

ℹ A backup is only deleted if it is necessary when a new backup is created.

No backup is automatically deleted at controller start-up, when the backup behaviour is defined by the NC command, in the event of an error or at NC reset.

The name of the last backup file and the current lines are saved. This means that the current state is always updated when the backup function is re-activated. This data is only deleted at controller restart. All other values for previously programmed filenames are discarded.

12.25   Monitoring the work space and protection space

The use of control areas prevents specific axis configurations from causing collisions between the tool and parts to be protected. A control area may be a work space or a protection space.

Control areas can be defined as three dimensional objects in cylindrical or polygonal form. The third dimension is defined by constant minimum and maximum values. Axes are assigned dependent on the currently active working plane (e.g. G17).
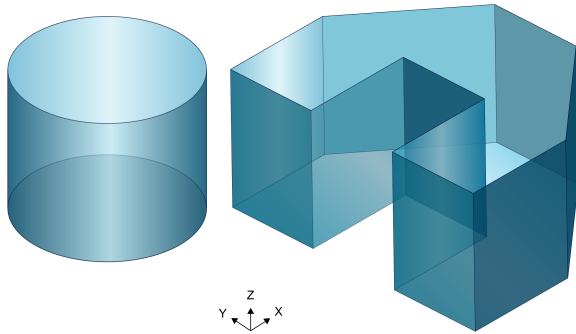
Fig. 117: Definition of 3D control areas (cylindrical, polygonal)
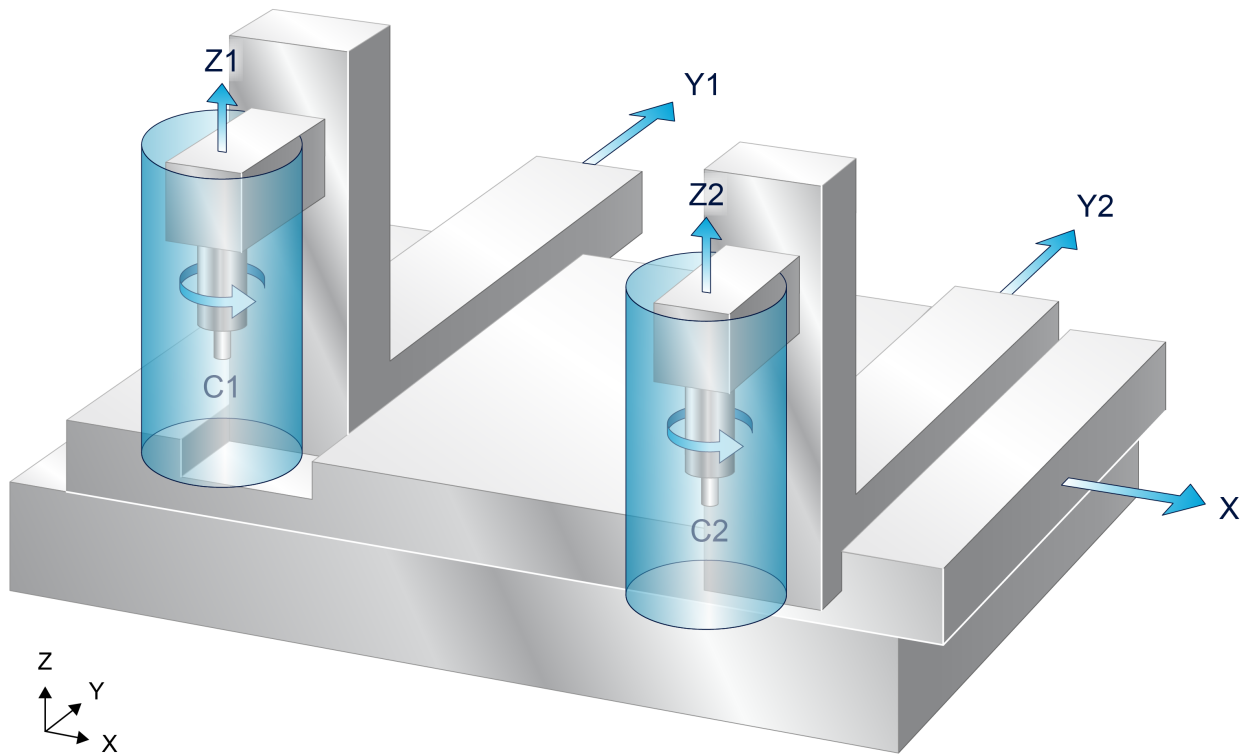
Fig. 118: Example of cylindrical workspace areas in an application

It is not possible to move out of a work space and into a protection space. Current tool data are included and the tool tip can be checked for validity within the control area.

Work and protection spaces are defined, activated and de-activated in the NC program. Several control areas may be active at the same time. Control areas can be redefined in de-activated state.

Work space and protection space monitoring with tool centre point monitoring is possible with

- automatic mode in conjunction with:
  ◦ Linear motion blocks
  ◦ Circular motion blocks (irrespective of their orientation G17/G18/ G19).
  ◦ Kinematic transformations
  ◦ Polynomial contours (contouring, HSC)
  ◦ Helical motions
  ◦ Reference point offsets with G92, G54
  ◦ Cartesian transformations #(A)CS available as of CNC Build V2.11.2015:
- active manual mode available as of CNC Build V3.1.3068.9: in conjunction with:
  ◦ Exclusive (G200) or inclusive mode (G201/G202)
  ◦ Kinematic transformations

12.25.1   Defining a control area (#CONTROL AREA BEGIN/END)

Time of definition

No workspace/protection areas are predefined when the controller starts up. A definition in the configuration lists is not possible.

A work or protection space is defined directly in the NC program in a sequence of path motions embedded in plain text commands.

In this case, path motions must always be programmed in absolute dimensions. The contour of the control area in the plane is defined either by a closed polygon formed in any shape by linear blocks (target point and starting point of the block sequence must be identical) or by a full circle. The excursion in the third dimension and further characteristics of the control area are defined in the assigned plaintext command.

```
Syntax for Start of control area definition:
```
#CONTROL AREA BEGIN [ID=.. **WORK | PROT POLY | CIRC MIN_EXCUR**=.. **MAX_EXCUR**=..

[EXCUR_AX=.. **|EXCUR_AXNR**=..] **[MONITOR_LVL**=..] **]**

ID=..	Identification number of the control area (ID). The definition is global valid after program end and RESET. Up to 20 different control areas can be defined.
WORK	Control area is a workspace.
PROT	Control area is a protection space.
POLY	Contour of a control area is defined as a polygonal shape.
CIRC	Contour of a control area is defined as a full circle.

MIN_EXCUR=..	Limitation of the control area in the third dimension in negative direction in [mm, inch].
MAX_EXCUR=..	Limitation of the control area in the third dimension in positive direction in [mm, inch].
EXCUR_AX=..	Optional specification of an axis identifier for the third excursion direction of the workspace or protection area (as of CNC Build V2.11.2025.00). By default the third main axis is used.
EXCUR_AXNR=..	Optional specification of a logical axis number for the third excursion direction of the workspace or protection area (as of CNC Build V2.11.2025.00). By default the third main axis is used.
MONITOR_LVL=..	Specification of the monitoring level, see Control levels (as of CNC Build V3.1.3079.42)
	IMCS: Intermediate coordinate system (only practical with multi-step transformations)
	MCS: Machine coordinate system (default)

Syntax for End of control area definition:

#CONTROL AREA END

Each control area must be closed by the command #CONTROL AREA END. Only then can further control areas be defined.

ℹ️ A deactivated area can be overwritten by reprogramming using the same ID.

NOTICE

Active Cartesian transformations #(A)CS are not considered in the definition of monitored spaces. Workspaces and protection areas are always defined as Cartesian in the MCS coordinate system.

Define control areas

```
(Define a polygonal workspace:
:
N10 #CONTROL AREA BEGIN [ID1 WORK POLY MIN_EXCUR=-50 MAX_EXCUR=50]
N20 G01 F1000 G90 X-150 Y75 (Starting point)
N30 X-50  Y150
N40 X50   Y150
N50 X150  Y75
N60 X150  Y0
N70 X50   Y0
N80 X50   Y75
N90 X-50  Y75
N100 X-50  Y0
N120 X-150 Y0
N130 X-150 Y75 (End point identical to starting point)
N140 #CONTROL AREA END
:
(Define a cylindrical protection area:
:
N210 #CONTROL AREA BEGIN [ID2 PROT CIRC MIN_EXCUR=-70 MAX_EXCUR=70]
N220 G01 X100 Y0 F10000 (Starting point for cylindrical prot. area)
N230 G02 G162 I50 J0
N240 #CONTROL AREA END
:
```
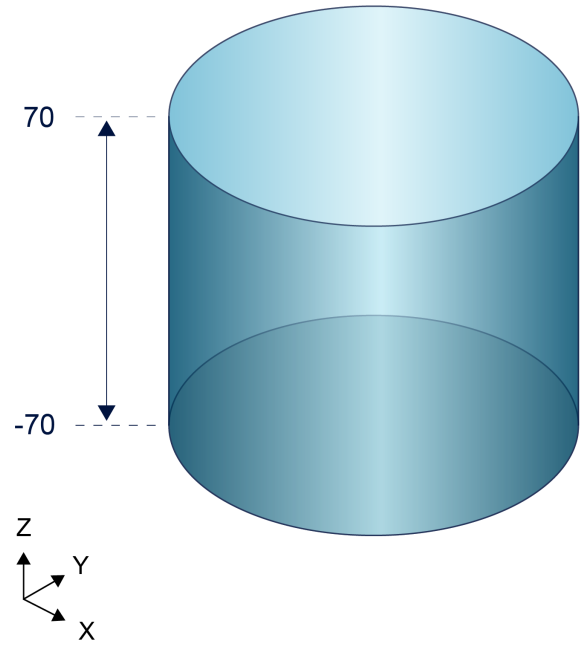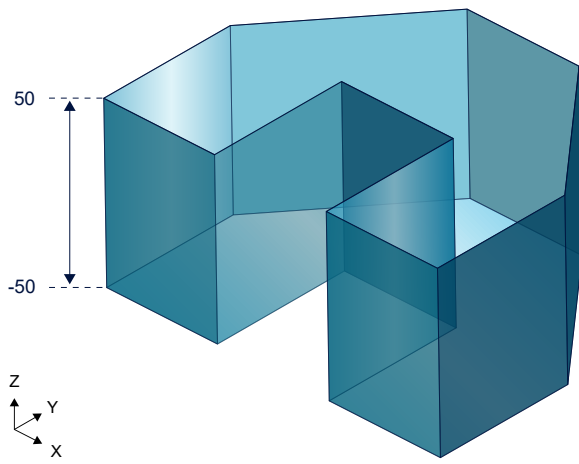
12.25.2 Selecting/deselecting control areas (#CONTROL AREA ON/ OFF)

When workspaces are activated, the TCP must already be located in the valid workspace. In the same way, when a work space is activated, the TCP may not cause a violation at the current position.

Syntax of Selecting a control area

#CONTROL AREA ON [**ALL**] | **[ID**=. **]**

ID=..	Identification number of the control area (ID).
ALL	Activate all defined control areas.

Deselecting and selecting control areas

```
#CONTROL AREA ON [ID3] (Activate specific control area)
#CONTROL AREA ON ALL (Activate all defined control areas)
```

Syntax of Deselecting a control area

#CONTROL AREA OFF [**ALL**] | **[ID**=..**]**

ID=..	Identification number of the control area (ID).
ALL	Deactivate all currently defined control areas.

```
#CONTROL AREA OFF [ID3] (Deactivate specific control area)
#CONTROL AREA OFF (Deactivate control area last selected)
#CONTROL AREA OFF ALL (Deactivate all active control areas)
```

12.25.3   Clearing control areas (#CONTROL AREA CLEAR)

Syntax of Clearing a control area

#CONTROL AREA CLEAR [**ALL**] | [**ID**=..**]**

| ID=.. | Identification number of the control area (ID). |
| ALL | Delete all defined control areas. |

Deleting control areas

```
#CONTROL AREA CLEAR [ID3] (Clear specific control area)
#CONTROL AREA CLEAR ALL   (Clear all defined control areas)
```

12.25.4   Monitor additional axes

Besides the main axes. X, Y, Z can adopt additional axes in the monitoring function for work spaces and protection spaces. In this case, the definition of the assigned control areas is limited to polygonal shapes. The control areas are defined using the associated axis identifiers.

Define a work space for the additional axes X2, Y2 and Z2

```
:
N10 #CONTROL AREA BEGIN [ID4 WORK POLY MIN_EXCUR=-50 MAX_EXCUR=50]
N20 G01 F1000 G90 X2=100 Y2=100   ;Starting point
N30 X2=-100
N40 Y2=-100
N50 X2=100
N60 X2 = 100 Y2= 100   ; End point identical with starting point
N70 #CONTROL AREA END
:
N500 #CONTROL AREA ON  ALL
:
N1000 M30
```

12.25.5   Special features in manual mode

Monitoring in manual mode is carried out in the real-time part of the CNC based on the defined and activated control areas.

Error response is identical to ACS limitation or approaching manual mode offset limits. If an IMCS / MCS limit is reached, all axes ahead of the limit without any error.

The start of the deceleration process ahead of the limit is dependent on the manual mode velocity and acceleration.

Output a warning message

The reason for motion stop is displayed to the user by the output of a warning message. To achieve this, P-MANU-00014 must be set.

Exclusive manual mode (G200)

In response to an error in automatic mode, an error message is output and the program is aborted. However in manual mode, only an axis motion stop occurs as an error reaction when a person enters the protection space or leaves the workspace.

Inclusive manual mode (G201/G202)

If motions from automatic and manual mode are superimposed (parallel interpolation), work space and protective space violations may occur.

Suppress workspace monitoring

The parameter P-CHAN-00442 influences or even suppresses workspace monitoring in manual mode.

Suppress workspace monitoring with P-CHAN-00442

Initial situation for all cases:
The machine runs in automatic or manual mode. Before activating manual mode, activate workspace monitoring, e.g. in a subroutine.

Case 1:

P-CHAN-00442 is assigned the value 1.
Workspace monitoring is not activated in manual mode although its definition and activation were executed in the NC program. The machine can move back and forth across workspace boundaries.

Case 2:

P-CHAN-00442 is assigned the value 0.

Workspace monitoring is active in manual mode in combination with the Suppress error output from workspace monitoring in manual mode control unit.

When manual mode is active, workspace monitoring can be deactivated by the signal set in the Suppress error output from workspace monitoring in manual mode control unit. The machine can then move back and forth across workspace boundaries.

When manual mode is activated the TCP must be within the permitted range. If this is not the case, an error is output.

Error ID 50961, if the workspace was left

Error ID 50962, if the protection space was left.

Case 3:

P-CHAN-00442 is assigned the value 2.

Workspace monitoring is activated in manual mode in combination with the Suppress error output from workspace monitoring in manual mode control unit.

When manual mode is activated, the TCP may be located outside the permitted range. The position of the TCP is not checked, Workspace monitoring can be deactivated by the control unit. The machine can move back within the permitted range.

12.26 Influence forward/backward motion on path

ℹ **The availability of this function depends on the configuration and the scope of the version.**

For more information on these commands and forward/backward motion on the path see the functional description [FCT-C7].

12.26.1 Skipping program sequences (#OPTIONAL EXECUTION)

In the NC program, the programming command #OPTIONAL EXECUTION ON/OFF flags a sequence that is to be skipped during backward motion or simulate motion.

Skipping is activated in the PLC. The flagged program part is skipped if

- backward motion is active (backward motion control unit)
- or with simulated ("simulate motion" control unit)

The flagged area is then skipped at interpolator level. However, transition conditions between blocks before and after the skipped area are not recalculated.

ℹ **The P-STUP-00033 function must be parameterised in order to use it**

`Syntax:`

#OPTIONAL EXECUTION [ON | OFF | CLEAR] [[SIMULATE] | [SIMULATE MASK=..] | [**APPROACH]]**

ON	Activate skipping
OFF	Deactivate skipping

The syntax below is available as of CNC Build V3.3107.12

CLEAR	Every motion before this command is deleted. This ensures that, in the case of an Escape motion, no motion block can be executed by the Escape function before this flag.
SIMULATE	The programmed sequence between #OPTIONAL EXECUTION ON and OFF is only skipped if the signal of the "simulate motion" control unit is active.
SIMULATE MASK=..	64-bit mask for the specification.
	The sequence between #OPTIONAL EXECUTION ON and OFF is only skipoped if the signal of the "simulate motion" control unit is active and the programmed mask has bitwise matches with the "simulate motion mask" control unit.
APPROACH	Implicitly flags the start and the end of the approach geometry in the Escape channel. This approach geometry is appended to the dynamically generated Escape path in the backward direction.
	The keyword can only be used for EDM and may only be programmed in the Escape channel.

Therefore any interpolator state, especially the axis positions, must remain unchanged to prevent a discontinuous transition of path, velocity and acceleration.

ℹ **Axis positions must be identical before and after the skipped sequence.**

If axis position are changed the error message ID 50452 is output.

Synchronisation of M/H functions with #OPTIONAL EXECUTION

The sequences flagged with #OPTIONAL EXECUTION ON/OFF are only skipped if backward motion or "simulate motion" are active. No M/H functions are output.

The behaviour or possibilities of M or H functions **outside** the sequence are described in [FCT-C7//"M/H function handshake with PLC"].

When the command #OPTIONAL EXECUTION [SIMULATE] ON/OFF is used, the "simulate motion" control unit must be active in order to skip the sequence. Therefore, skipping the sequence with backward motion requires backward motion and "simulate motion" to be active.

Skipping program sequences

```
%t_storag.nc
X10 Y0
N10 G91 G00 X10 F1000

N11 #OPTIONAL EXECUTION ON
N12 Z123
N13 S1000 M3
N14 Z-123
N15 M101
N16 #OPTIONAL EXECUTION OFF

N20 G90 G01 X0
N30 G02 I10
N40 G03 J10
M30
```

The CNC checks and monitors only the continuous path of axes whether blocks are skipped or not. Any further conditions must be ensured by the user itself and are not checked by the CNC.

Nesting of multiple commands optional execution on/off is not considered.

Before terminating the program level (M17, M29) where OPTIONAL EXECUTION was selected (ON), the function must also be deselected (OFF). This also applies to terminating the main program level (M30). If the program level is terminated without deselection of the function, the error ID 21719 is output.

It is only possible to terminate complete section. If backward motion or "simuate motion" are activated when the program is executed within an OPTIONAL_EXECUTION section, the section is not skipped.

> **i** **It is not practical to use the NC command #OPTIONAL EXECUTION together with contour-changing functions, e.g. tool radius compensation or polynomial contouring.**

Skipping program sequences with "SIMULATE MASK"

In the NC program below, 3 skipped sequences are flagged and are each provided with an identifier in the form of a binary bitmask. These sequences are only skipped when the "simulate motion" control unit is active and also when the programmed mask has bitwise matches with the "simulate motion mask" control unit.

```
N010 X10 Y0
N020 G91 G00 X10 F1000
N030 #OPTIONAL EXECUTION ON [SIMULATE MASK='2#000001']
N040 X20
N050 M3
N060 X0
N070 M101
N080 #OPTIONAL EXECUTION OFF

N090 #OPTIONAL EXECUTION ON [SIMULATE MASK='2#000010']
N100 X30
N110 M3
N120 X0
N130 M102
N140 #OPTIONAL EXECUTION OFF

N150 #OPTIONAL EXECUTION ON [SIMULATE MASK='2#000100']
N160 X40
N170 M3
N180 X0
```

```
N190 M103
N200 #OPTIONAL EXECUTION OFF

N210 X50
N220 X0
N230 M30
```

12.26.2 Clearing backward storage (#BACKWARD STORAGE CLEAR)

The NC command #BACKWARD STORAGE CLEAR explicitly clears the previous backward storage. This ensures that the function is stopped after this program position is crossed.

```
Syntax:
```
#BACKWARD STORAGE CLEAR

Clearing backward storage

```
%backward-storage

N000 G01 X0 F10000
N010 X100 Y123
N020 X100
N030 X200 Y10
N040 X300 Y20

N050 #BACKWARD STORAGE CLEAR

N060 X400 Y-20
N070 X500 Y-3

N060 #BACKWARD STORAGE CLEAR

N080 X444 Y10
N090 X333 Y3
N100 X222 Y10
N110 X111 Y3
N120 X000 Y10
N130 X-111 Y3

N140 #BACKWARD STORAGE CLEAR

N1000 M30
```

12.27 Tool change with active synchronous mode (#FREE TOOL CHANGE)

For security reasons, it is generally not permitted to adopt new tool data (T (with implicit D), D or #TOOL DATA) with active synchronous mode. This is indicated by error message 20169. Locking can be switched by the following NC command:

```
Syntax:
```

#FREE TOOL CHANGE ON | OFF

At the latest, the NC command must be programmed before the first tool change selected in active synchronous mode. The synchronous mode lock is disabled by the keyword "ON".

Locking of synchronous mode and tool change are re-enabled by the keyword "OFF". In addition, the NC command is deselected at RESET and implicitly at the end of the main program.

At program start, locking of synchronous mode and tool change is always active for compatibility and security reasons (default).

ⓘ If the channel parameter P-CHAN-00100 (move_tool_offsets_directly) is active at the same time, the error message 20169 continues to be generated since the immediate exiting of tool offsets in the slave axes could lead to machine damage when synchronous mode is active. This means that when #FREE TOOL CHANGE is used, P-CHAN-00100 should be deactivated.

Tool change with active synchronous mode

```
%TOOL_AXLINK
N05 X0 Y0 Z0 C100 G53 D0
N10 #AX LINK [1,C=X]
N15 #FREE TOOL CHANGE ON
N20 #AX LINK ON [1]
N30 X100 Y50 Z30
N40 D2
N50 X200 Y75 Z40
N60 D1
N65 X300 Y100
N70 #AX LINK OFF [1]
N75 #FREE TOOL CHANGE OFF
N70 X25 Y25 Z25 C25
N80 M30
```

12.28 Locking program areas for single-step mode (#SINGLE STEP)

The command #SINGLE STEP [DISABLE / ENABLE] locks any program areas in the NC program for single-step mode. This allows the operator to skip the labelled program areas in one step and jump faster to the NC lines to be analysed.

If locked areas are nested, the single-step lock includes the area starting at the first activation to the first deactivation (see example 2).

The user can continue to define an incremental width at a specific block number single-step resolution #SINGLE STEP [RESOLUTION…] at which the single-step stop is to act (see example 3).

For further information on single-step block mode, see the functional description [FCT-M2].

Syntax:

#SINGLE STEP [DISABLE | ENABLE | RESOLUTION=.. **]**

DISABLE	Start of single-step lock.
ENABLE	End of single-step lock.
RESOLUTION=..	Specific block number incremental width at which the single-step stop acts.

Locking program areas for single-step mode

Example 1:

Single-step mode is not effective in the area of the NC blocks N40–N100 including all included subroutines called in it.

```
%SINGLE_STEP
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #SINGLE STEP [DISABLE]
N50 X20
N60 Y20
N65 L GSP.nc
N70 Z20
N80 X30
N90 Z30
N100 #SINGLE STEP [ENABLE]
N110 Y30
N120 X40
N130 Z40
N999 M30
```

Example 2:

Area of single-step lock with nesting includes N40-N75

```
%SINGLE_STEP
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #SINGLE STEP [DISABLE]
N50 X20
N55 #SINGLE STEP [DISABLE]
N60 Y20
N65 L GSP.nc
N70 Z20
N75 #SINGLE STEP [ENABLE]
N80 X30
N90 Z30
N100 #SINGLE STEP [ENABLE]
N110 Y30
N120 X40
N130 Z40
N999 M30
```

Example 3:

Block numbering with user-specific single-step resolution (steps of 10) and internal numbering (single-step width). The black lines show the single-step stop. There is no stopping within the grey area.

```
%SINGLE_STEP

N010 #SINGLE STEP [RESOLUTION = 10]

N090 Y0
N091 Y1
N092 Y2
N093 Y3
N094 Y4

N100 Y5
N101 Y6
N102 Y7

N110 Y8

…
```

12.29 Programmable path override (#OVERRIDE)

This command influences path feed in the NC program if this needs to be different for feed and rapid traverse blocks. The programmed override for path motions is active if at least one axis moves. This does not affect the mode of operation of real-time influencing of feed by the PLC.

An additional function is provided for programmable axis override [▶ 769].

If an axis override is also defined for an axis involved in path motion, the effective path override results from multiplication of the two override values.

> ● The G166 function [▶ 179] suppresses the programmed override values.
> **i**

Syntax:

#OVERRIDE [FEED_FACT=.. RAPID_FACT=..]

FEED_FACT=..	Override factor for feed blocks [0.1%-200%].
RAPID_FACT=..	Override factor for rapid traverse blocks [0.1%-200%].

Programmable path override

```
%path_override

N10 G00 G90 X0 Y0 Z0
   (Path override G01 122.765%, G00 155.7%)
N20 #OVERRIDE [FEED_FACT=122.765 RAPID_FACT=155.7]
N30 G01 X100 Y100 Z100 F1000           Effective feed=1227.65
   (Path override G01, G00 100%)
N40 #OVERRIDE [FEED_FACT=100 RAPID_FACT=100]
N50 G01 X200 Y200 Z200 F1000           Effective feed=1000

M30
```

12.30 Drive-independent switching of drive functions

As opposed to SERCOS-specific commands as described in Section <u>Writing and reading drive parameters and commands [▶ 359]</u>, the following commands permit the **drive-independent** setting of drive parameters. The parameters can only be written synchronously, i.e. the command is executed at interpolator level at the time of processing. In the default setting, the #DRIVE command is executed in parallel to the following processing of the NC program, i.e. program processing is not stopped. The "WAIT" parameter can be used to stop interpolation until the drive function is completed.

The command #DRIVE WAIT SYN can be used to check the successful switching of the drive function at a **later** time.

12.30.1 Synchronised write (#DRIVE WR SYN)

`Syntax:`

#DRIVE WR SYN [AX=<*axis_name*> **| AXNR=**.. **[MOTOR=**..**] [PARAM_SET=**..**] KEY=**<*string*> **VAL=**.. **[WAIT]]**

AX=<*axis_name*>	Name of the (drive) axis
AXNR=..	Logical axis number of the (drive) axis, positive integer
MOTOR=..	Selectable motor number in the drive amplifier on motor switching; positive integer
PARAM_SET=..	Switch-over parameter set in the drive amplifier;
	Positive integer
KEY=<*string*>	Keyword of addressed function as described in [2] [▶ 786]
VAL=..	Value to be transmitted, negative or positive integer
WAIT	Stopping interpolation and waiting for successful switching of drive function.
	If this keyword is not specified, a #DRIVE WAIT SYN can be programmed in the subsequent part of the NC program to check the successful switching of the drive function.

NOTICE
When continuous rotation is active for spindles, no motor switching and no changing of the parameter set (MOTOR=<expr>, PARAM_SET=<expr>) may be programmed.

12.30.2 Synchronous waiting for acknowledgement (#DRIVE WAIT SYN)

The following command checks whether all previous #DRIVE WR SYN were completed for an axis. The interpolator is stopped until all #DRIVE WR SYN are executed in the drive. This applies to both path axes and spindle axes.

```
Syntax:
```
#DRIVE WAIT SYN [AX=<axis_name> **| AXNR=.. SWITCH_OK]**

AX=<axis_name>	Name of the (drive) axis
AXNR=..	Logical axis number of the (drive) axis, positive integer
SWITCH_OK	Check whether <u>all</u> previous #DRIVE WR SYN are completed.

Synchronous waiting for acknowledgement

```
Synchronous writing with immediate waiting for acknowledgement:

%TOOL_AXLINK1
N05 X0 Y0 Z0
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY=torque_limit VAL=400 WAIT]
N20 X100 Y50 Z30 G01 F3000
N30 X200 Y75 Z40
N65 X300 Y100
N70 X25 Y25 Z25 C25
Nxx
N80 M30


Synchronous writing with subsequent waiting for acknowledgement:

%TOOL_AXLINK2
N05 X0 Y0 Z0
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY=torque_limit VAL=400]
N20 X100 Y50 Z30 G01 F3000
N30 X200 Y75 Z40
N60 #DRIVE WAIT SYN [AX=X SWITCH_OK]
N65 X300 Y100
N70 X25 Y25 Z25 C25
Nxx
N80 M30
```

12.31 Velocity-optimised motion control by segmentation (#SEGMENTATION)

In some applications (e.g. kinematics where singular areas can appear) it may be an advantage to improve the programmed block segmentation on the CNC side or to convert circular blocks (G2/G3) into linear blocks (G1) by segmentation. In addition, circular blocks can also be divided into circle segments to improve the utilisation of machine dynamics. This can be achieved in the NC program by the following command.

```
Syntax:
```
#SEGMENTATION [**ON** | **OFF**] [**ALL**] [**[** [**LIN**] [**LENGTH**=..] [**CIR**] [**OPMODE**=..] [**PARAM**=..] **]**]

ON	Select segmentation
OFF	Deselect segmentation
ALL	Segment linear and circular blocks
LIN	Segment linear blocks
LENGTH=..	Length of the resulting linear blocks in [mm, inch]
CIR	Segment circular blocks
OPMODE=..	Mode of circle segmentation:
	0: Preset required block length.
	1: Specify required chordal error. Block length is calculated automatically (default).
	2: Preset required block length, output as circle segments.
	Chordal error and block length are defined by the keyword PARAM.
PARAM=..	Either chordal error or length of the resulting linear blocks, depending on the selected OPMODE in [mm, inch]

If no other parameterisation is programmed except for LIN and/or CIR when segmentation is activated, the following initial state is valid:

LENGTH	1 mm
OPMODE	0
PARAM	0.1 mm

BECKHOFF

Velocity-optimised motion control by segmentation

Select linear segmentation with default parameterisation:

```
N20 G00 X0 Y0 Z0 F10000
N30 #SEGMENTATION ON [LIN]
N40 X3     Y25
N50 X15    Y15
N60 X23    Y12
N70 X25    Y25
N80 X30    Y35
N90 #SEGMENTATION OFF [LIN] ;Deselect
N100 M30
```

Select linear segmentation + parameterisation:

```
N20 G00 X0 Y0 Z0 F10000
N30 #SEGMENTATION ON [LIN LENGTH 0.5]
N40 X3     Y25
N50 X15    Y15
N60 X23    Y12
N70 X25    Y25
N80 X30    Y35
N90 #SEGMENTATION OFF [LIN] ;Deselect
N100 M30
```

Select linear and circular segmentation + parameterisation:

```
N30 #SEGMENTATION ON [LIN LENGTH 0.5 CIR OPMODE 1 PARAM 0.5]
N40 X3     Y25
N50 X15    Y15
N60 X23    Y12
N70 X25    Y25
N80 X30    Y35
N90 #SEGMENTATION OFF ALL ;Deselect
N100 M30 ;Alternative: #SEGMENTATION OFF [LIN CIR]
```

Combined selection of linear and circular segmentation:

```
N30 #SEGMENTATION ON ALL
N40 X3     Y25
N50 X15    Y15
N60 X23    Y12
N70 X25    Y25
N80 X30    Y35
N90 #SEGMENTATION OFF ALL ;Deselect
N100 M30
```

12.32   Enlarging/reducing contours (#SCALE ON/OFF))

The #SCALE command can enlarge or reduce the scales of contours or positions by specifying axis-specific factors. When different factors are specified, it also permits the expansion or compression of contours.

Scaling acts on

- Path axes (linear axes, e.g. X, Y, Z, U, V, W)
- Positioning axes (independent axes, oscillating axes).

Offsets are generally not scaled, regardless of whether an offset is selected or programmed before or after #SCALE ON.

Scaled values always refer to the zero point of the currently active coordinate system. In particular when a contour is described in absolute coordinates, you are advised, before selecting the scaling, to place the zero point on the starting point of the contour by means of the appropriate G functions G53 to G57, G159, G92 or by the functions for defining the rotated coordinate systems #ROTATION and #CS.

```
Syntax:
```
#SCALE [ON] *<axis_name>*.. { *<axis_name>*.. }
#SCALE OFF

ON	Activate scaling
OFF	Deactivate scaling
<axis_name>..	Axis-specific scaling factors. The factor must be > 0.
	An error message is issued if factors are ≤ 0.

ℹ️ The definition of scaling factors and their selection can be specified together or in separate steps. This means that it is possible, for example, to define the scaling factors first and then to activate scaling in a second command.

The programmed scaling factors remain stored up by M30 to program end and can be used again if #SCALE ON/OFF is used several times.

NOTICE

The scaling factors can be modified when scaling is deselected.

Modifying the scaling factors while scaling is active leads to the output of an error message.

The following must be noted when scaling and circle programming are combined:

- Scaling circles only makes sense if the scaling factors of the axes involved in circular interpolation are identical. This applies in particular to programming circles with radius specification via R or G163 because the factor for the radius is based here on the scaling factor programmed with "X".

- When circles are programmed with I, J, K, different scaling factors are also possible; - then, however, they normally cause errors in the circle centre point correction or create circle segments without any practical meaning.

Enlarge and reduce contours

```
;Scale an absolute programmed contour with identical factors

%L Cont1_abs
N01 G01 G90 F2000
N02 X90 Y0
N03 G301 I20
N04 X90 Y120
N05 G302 I20
N06 X50 Y120
N07 X50 Y100
N08 X40 Y100
N09 G03 X0 Y100 I-20 J0
N10 G01 X0 Y20
N11 G03 X20 Y0 R20
```

```
N13 M17

%scale
N015 G53
N020 G00 X0 Y0 Z0
N065 LL Cont1_abs ; Basic contour
N075 #SCALE X0.5 Y0.5 ;Definition of scaling factors
;Definition of zero point origin of the scaled contour
N085 V.G.NP[1].V.X = 30
N090 V.G.NP[1].V.Y = 20
N095 G54
N100 G90 G0 X0 Y0 Z0
N105 #SCALE ON
N110 LL Cont1_abs
N115 #SCALE OFF
N140 M30
```

```
;Scale an absolute programmed contour with different ;factors

%L Cont1_abs
N01 G01 G90 F2000
N02 X90 Y0
N03 G301 I20
N04 X90 Y120
N05 G302 I20
N06 X50 Y120
N07 X50 Y100
N08 X40 Y100
N09 G03 X0 Y100 I-20 J0
N10 G01 X0 Y20
N11 G03 X20 Y0 R20
N13 M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N065 LL Cont1_abs ; Basic contour
N075 #SCALE X0.6 Y0.3 ;Definition of scaling factors
;Definition of zero point origin of the scaled contour
N085 V.G.NP[1].V.X = 30
N090 V.G.NP[1].V.Y = 20
N095 G54
N100 G90 G0 X0 Y0 Z0
N105 #SCALE ON
N110 LL Cont1_abs
N115 #SCALE OFF
N140 M30
```
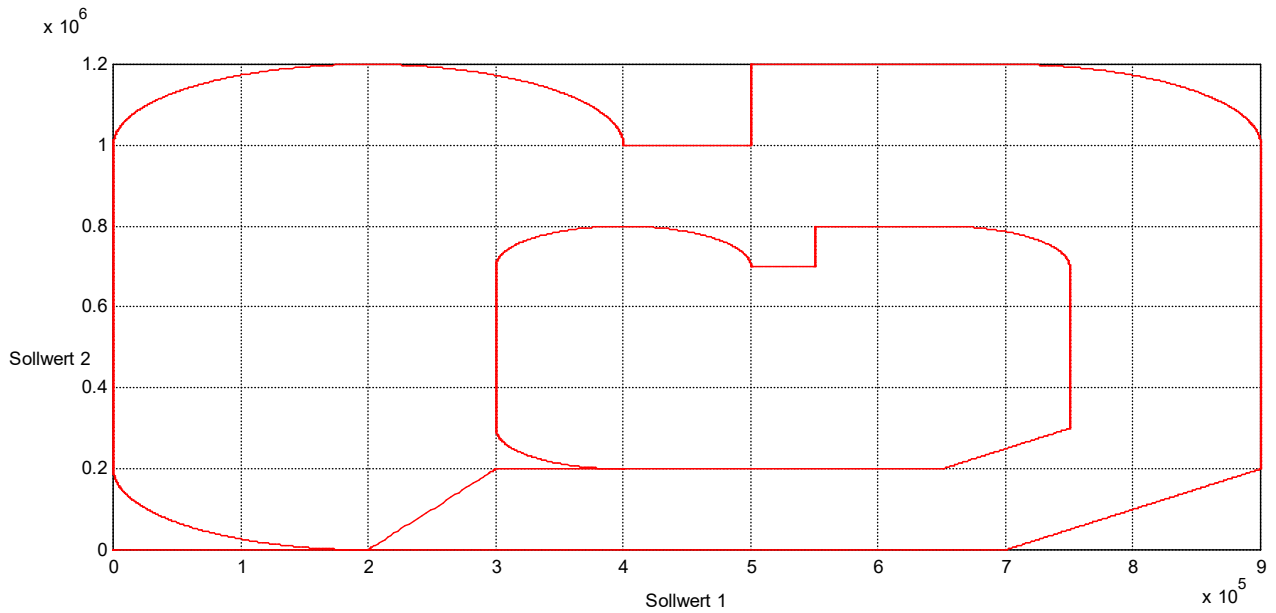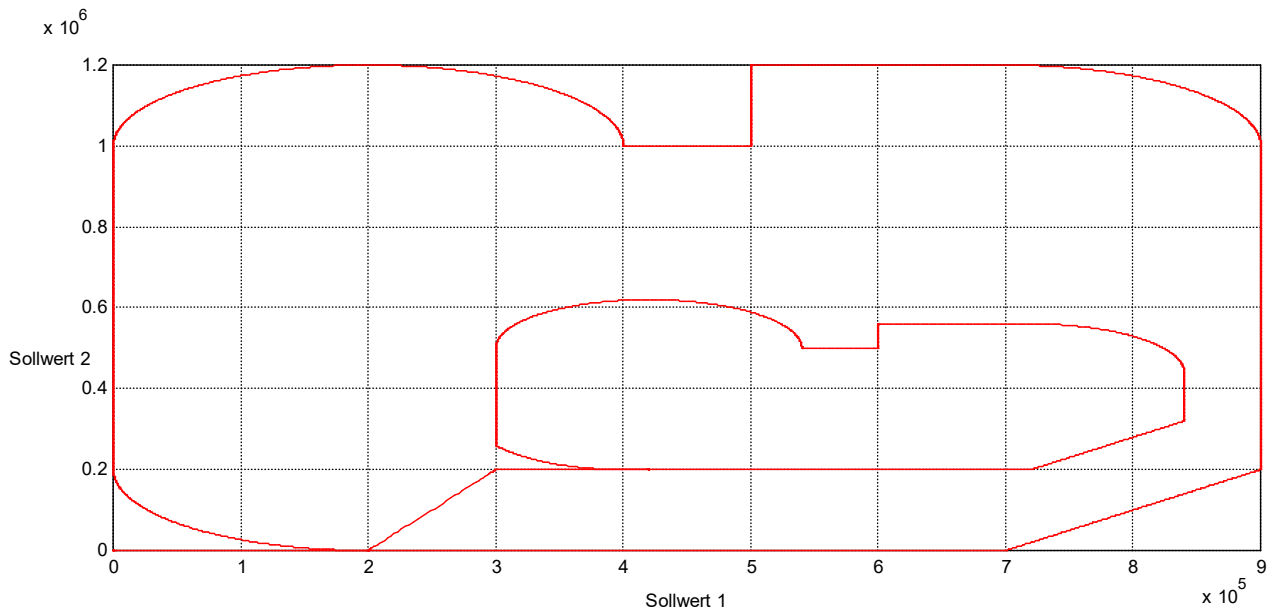
```
;Multiple different scaling of a relative programmed
;contour

%L Cont1_rel
N01 G01 G91 F2000
N02 X90 Y0
N03 G301 I20
N04 X0 Y90
N05 G302 I20
N06 X-40 Y0
N07 X0 Y-20
N08 X-10 Y0
N09 G03 X-40 Y0 I-20 J0
N10 G01 X0 Y-50
N11 G03 X20 Y-20 R20
N13 M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
; Basic contour
N030 LL Cont1_rel
N040 #SCALE X0.3 Y0.3 ;Definition 1 of scaling factors
;Definition 1 of starting point of the scaled contour
N055 G90 G0 X10 Y50
N060 #SCALE ON
N065 LL Cont1_rel
N070 #SCALE OFF
N075 G90 G00 X20 Y0
N085 #SCALE X0.5 Y0.5 ;Definition 2 of scaling factors
;Definition 1 of starting point of the scaled contour
N100 G90 G0 X40 Y10
N105 #SCALE ON
N110 LL Cont1_rel
N115 #SCALE OFF
N125 M30
```
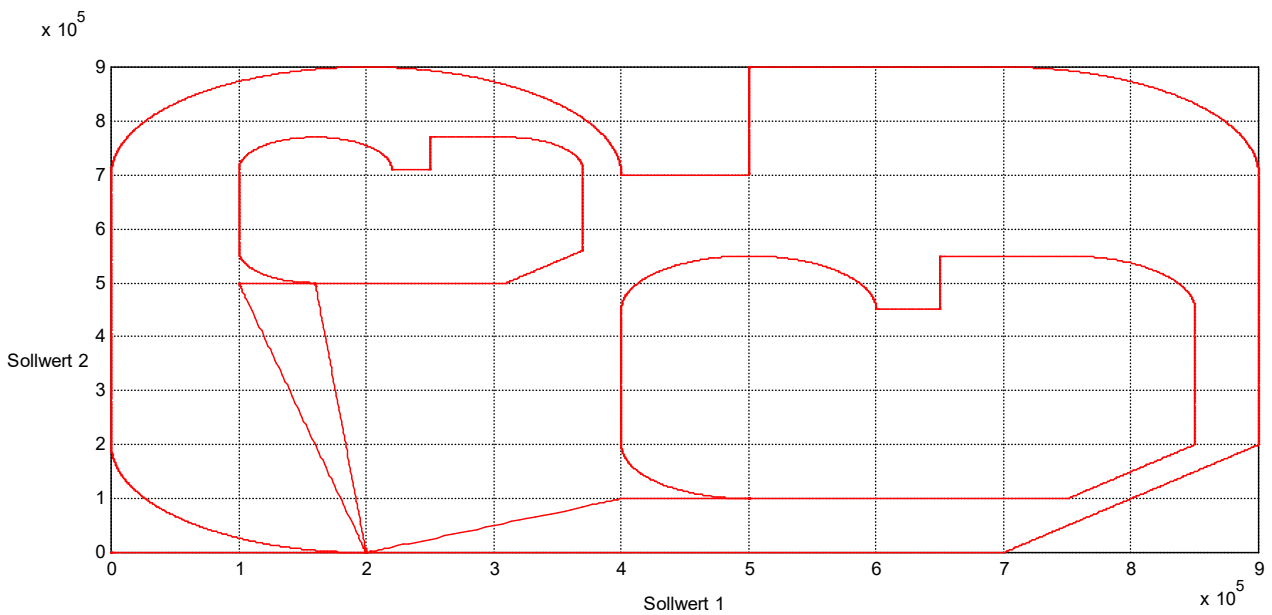
```
;Scale a contour in a coordinate system which is shifted and
rotated with #CS

%L Cont1_rel
N01 G01 G91 F2000
N02 X90 Y0
N03 G301 I20
N04 X0 Y90
N05 G302 I20
N06 X-40 Y0
N07 X0 Y-20
N08 X-10 Y0
N09 G03 X-40 Y0 I-20 J0
N10 G01 X0 Y-50
N11 G03 X20 Y-20 R20
N13 M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
; Basic contour
N030 LL Cont1_rel
N040 #SCALE X0.6 Y0.6 ;Definition of scaling factors
N045 #CS ON [40,5,0,0,0,20]
N055 G90 G0 X0 Y0
N060 #SCALE ON
N065 LL Cont1_rel
N070 #SCALE OFF
N0525 #CS OFF
N125 M30
```
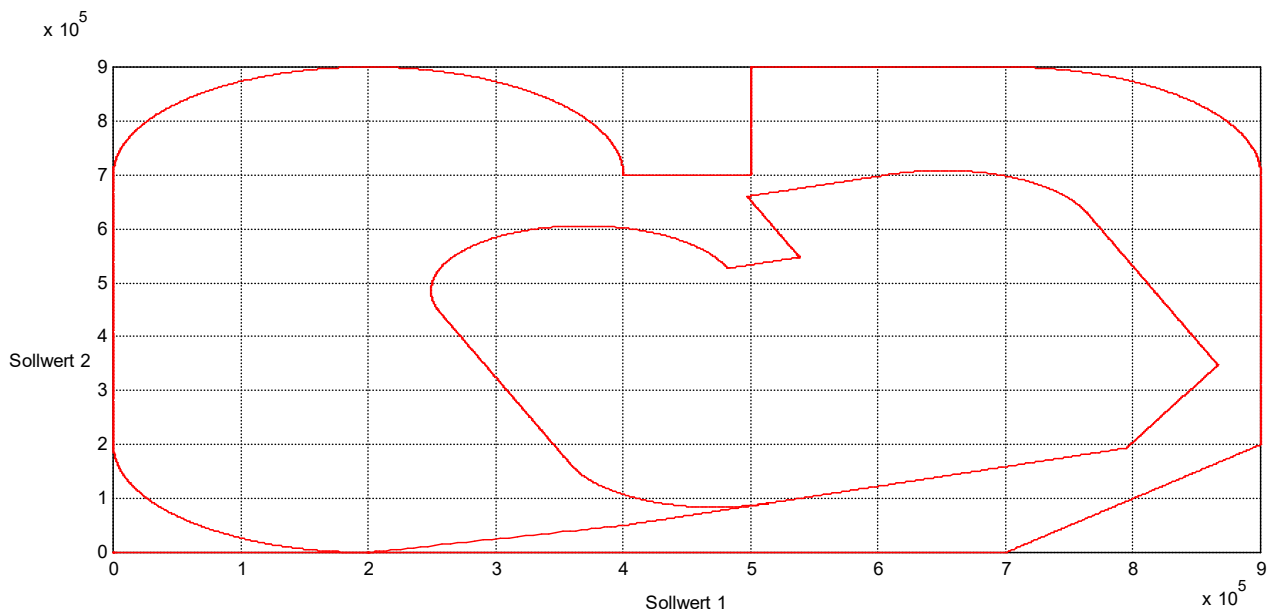
```
;Scale a semicircle, programmed with I, J

%L Circle1
G02 G91 X80 Y80 I40 J40
M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N025 G01 X10 Y10 F1000
; Basic semicircle
N030 LL Circle1
N040 #SCALE X0.7 Y0.7 ;Definition of scaling factors
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle1
N070 #SCALE OFF
N075 G90 G0 X0 Y0

M30
```
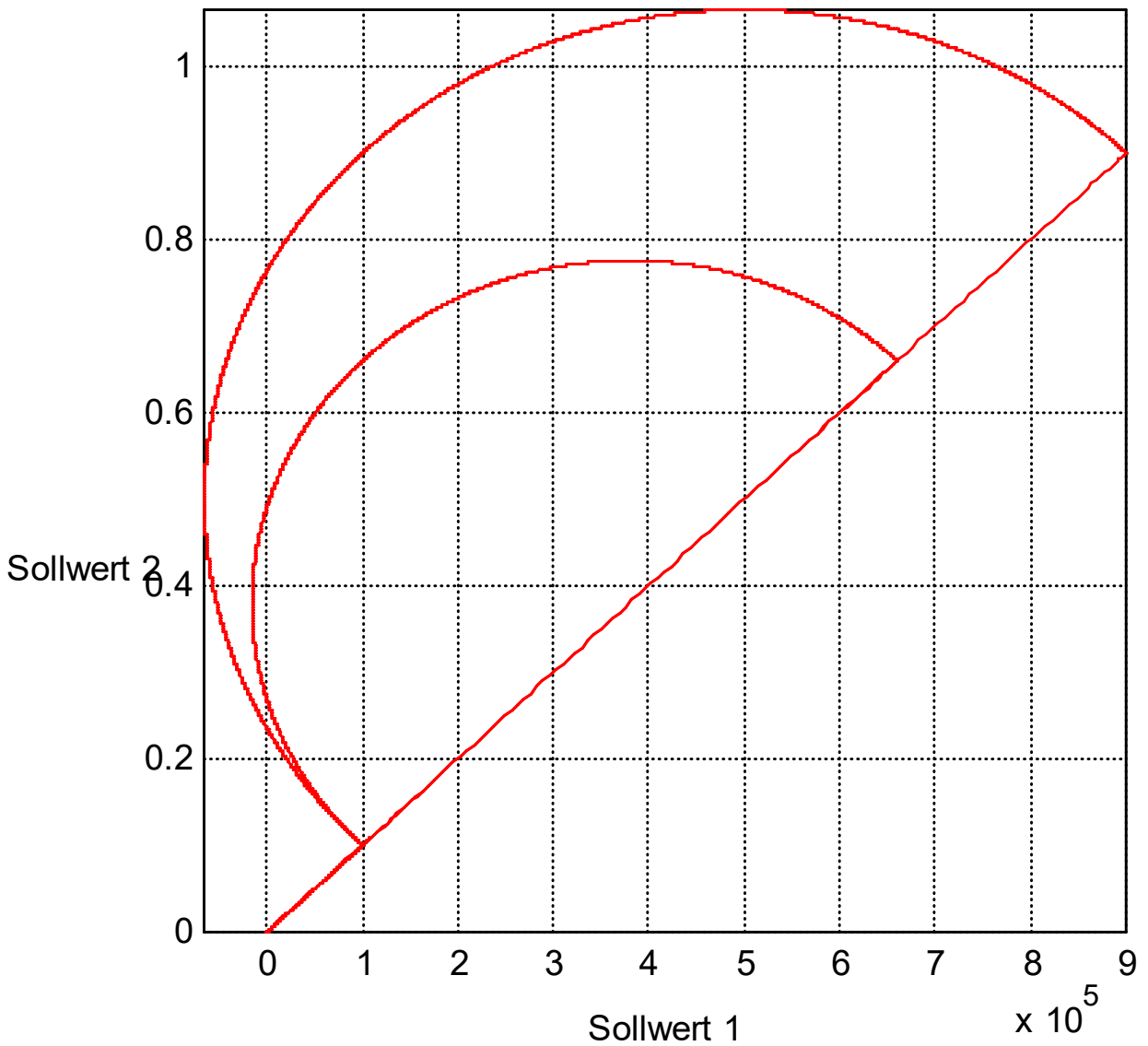
```
;Scale a semicircle, programmed with R

%L Circle2
G03 R33 X-66 Y0
M17
%scale

N015 G53
N020 G00 X0 Y0 Z0 F1000
; Basic semicircle
N030 LL Circle2
N040 #SCALE X1.5 Y1.5 ;Definition of scaling factors
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle2
N070 #SCALE OFF
N075 G90 G0 X0 Y0

M30
```
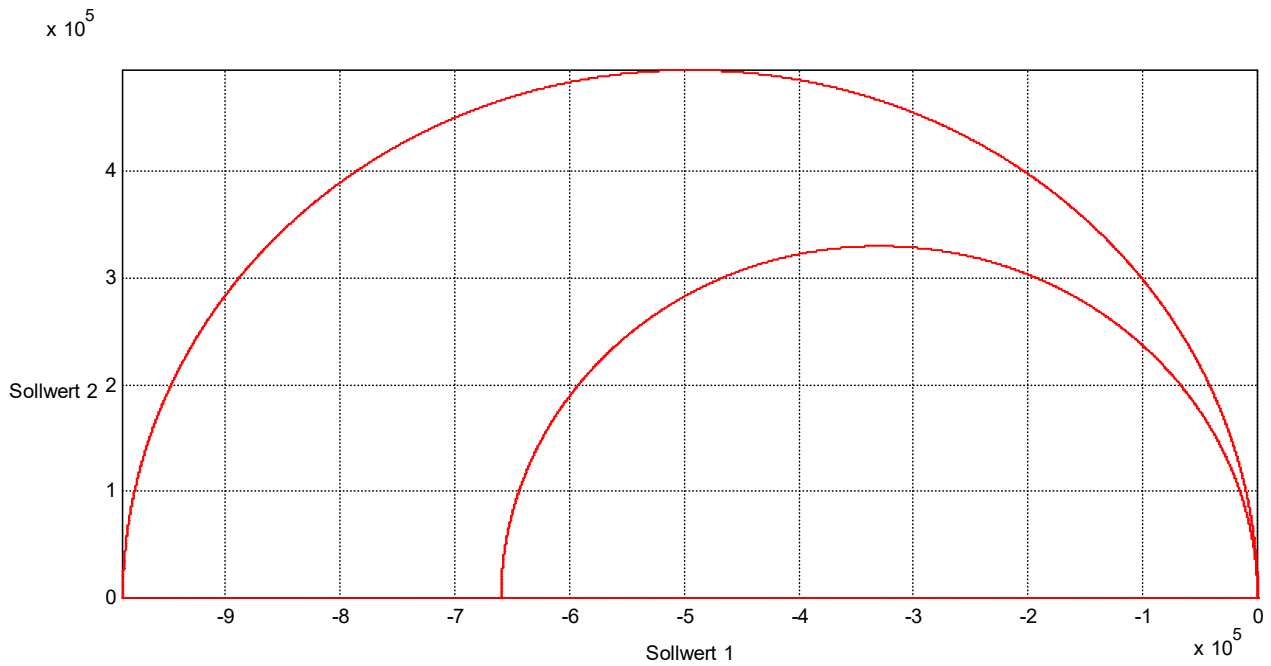
```
;Asymmetrical scaling of a semicircle programmed with I, J

%L Circle1
G02 G91 X80 Y80 I40 J40
M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N025 G01 X10 Y10 F1000
; Basic semicircle
N030 LL Circle1
N040 #SCALE X0.3 Y0.8 ;Definition of scaling factors
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle1
N070 #SCALE OFF
N075 G90 G0 X10 Y10

M30
```
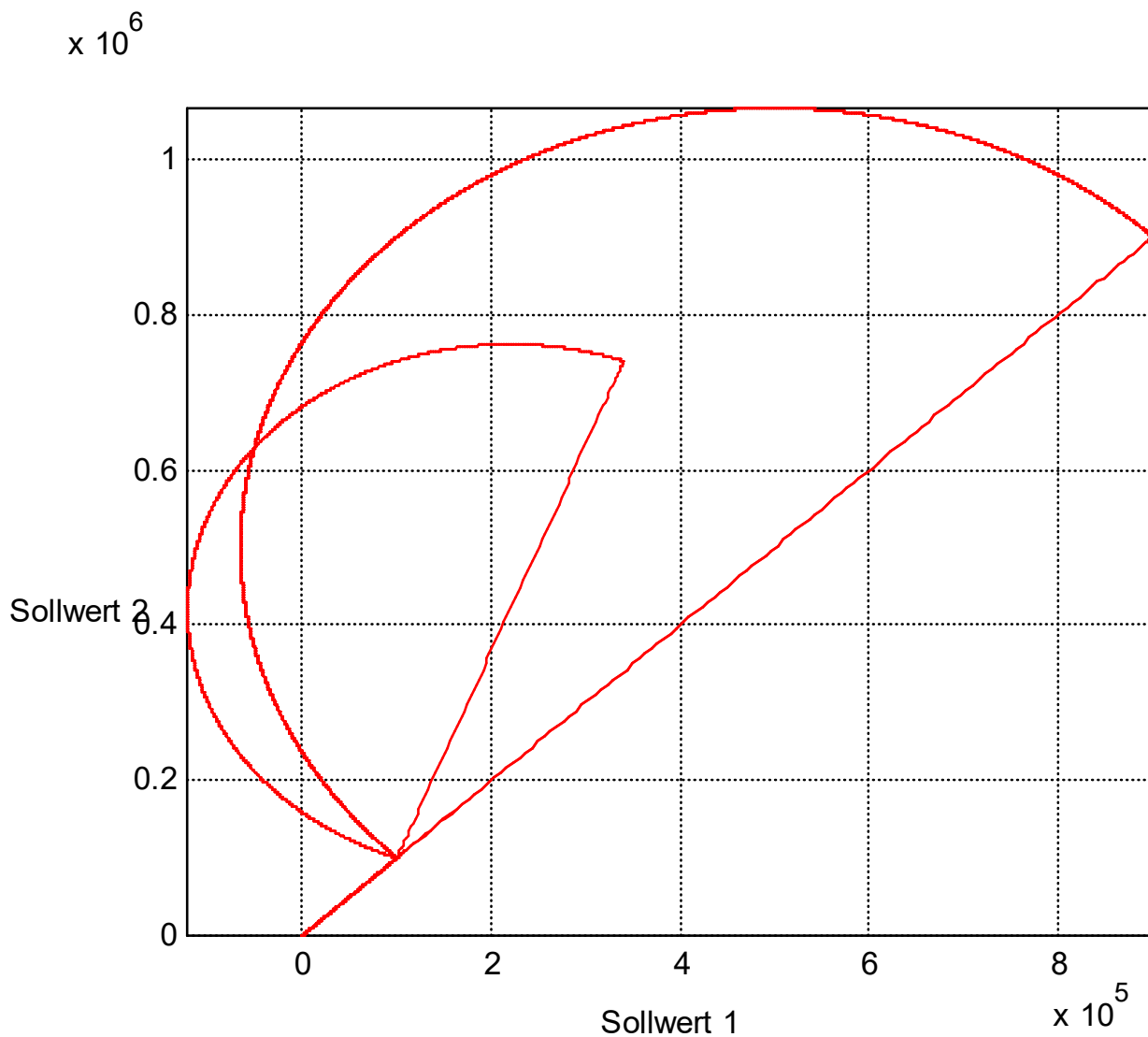
12.33   Punching and nibbling

With punching and nibbling operations, automatic stroke motions are executed in a programmable number of partial steps. Users can freely define the stroke motion sequence. The stroke motion is only triggered in the active machining plane with active path motions.

12.33.1   Splitting up motion path and programming (#STROKE DEF, #PUNCH ON/OFF, #NIBBLE ON/OFF)

The motion path is split up either by specifying the length of a partial segment or the number of partial segments. Programming is modal.

Splitting up by specifying the length of a partial segment takes place so that the travel block is uniformly subdivided into partial segments. The length of each real partial segment is less than or equal to the length of the programmed partial segment.

When the motion path is split up by specifying the number of partial segments, the segment length is calculated automatically.
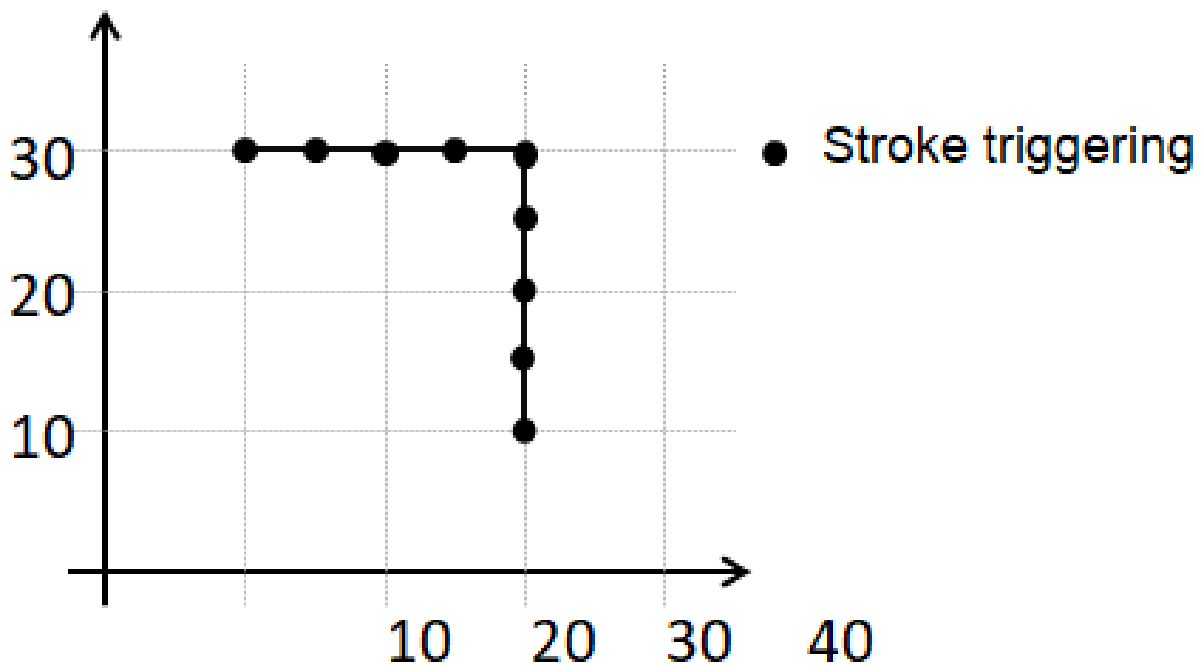
Fig. 119: Splitting up linear blocks

Programmed circular blocks are converted to linear partial motions and splitting of the motion path refers to the arc length of the circle.
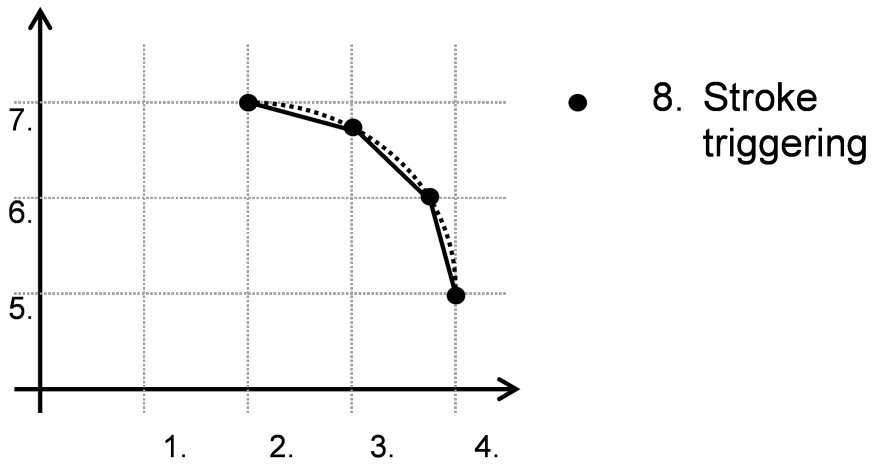
Fig. 120: Splitting up circular blocks

To define the reciprocating motion, a sequence limited to a maximum of *10 blocks* is programmed, which is delimited with the following two commands:

`Syntax of Defining stroke motion:`

#STROKE DEF BEGIN	Start of stroke definition
#STROKE DEF END	End of stroke definition

Only the following restricted scope of functions is allowed for stroke definition:

#STROKE DEF BEGIN

G0, G01, M, H, G261, G61, G260, G60, G04, #TIME, G90, G91

#STROKE DEF END

i The use of any other NC commands within the stroke definition leads to an error.

`Syntax of Activating/deactivating punching and nibbling functions:`

#PUNCH ON | **OFF [** [**LENGTH**=.. | **NUMBER**=..] **]**

#NIBBLE ON | **OFF [** [**LENGTH**=.. | **NUMBER**=..]

ON	Activate punching/nibbling
OFF	Deactivate punching/nibbling
LENGTH=..	Length of a partial segment after which a stroke motion is automatically inserted in [mm, inch]. With #NIBBLE ON, a stroke motion is executed during the first path motion in the active machining plane, including at the start.
NUMBER=..	Number of partial segments to be generated within one motion command. A stroke motion is triggered after every partial segment.

i LENGTH and NUMBER are exclusive, i.e. splitting up is based either on partial segment length or the number of partial segments.

Split up motion path and programming

```
%Nibble
N10 G90 G17
N20 #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50 Z-10
N60 #STROKE DEF END

N70   X10    Y30 C0
N80 #NIBBLE ON [LENGTH 5]
N90   X30    Y30 C180
N100  X30    Y10 C360
N110 #NIBBLE OFF
N120 M30
```
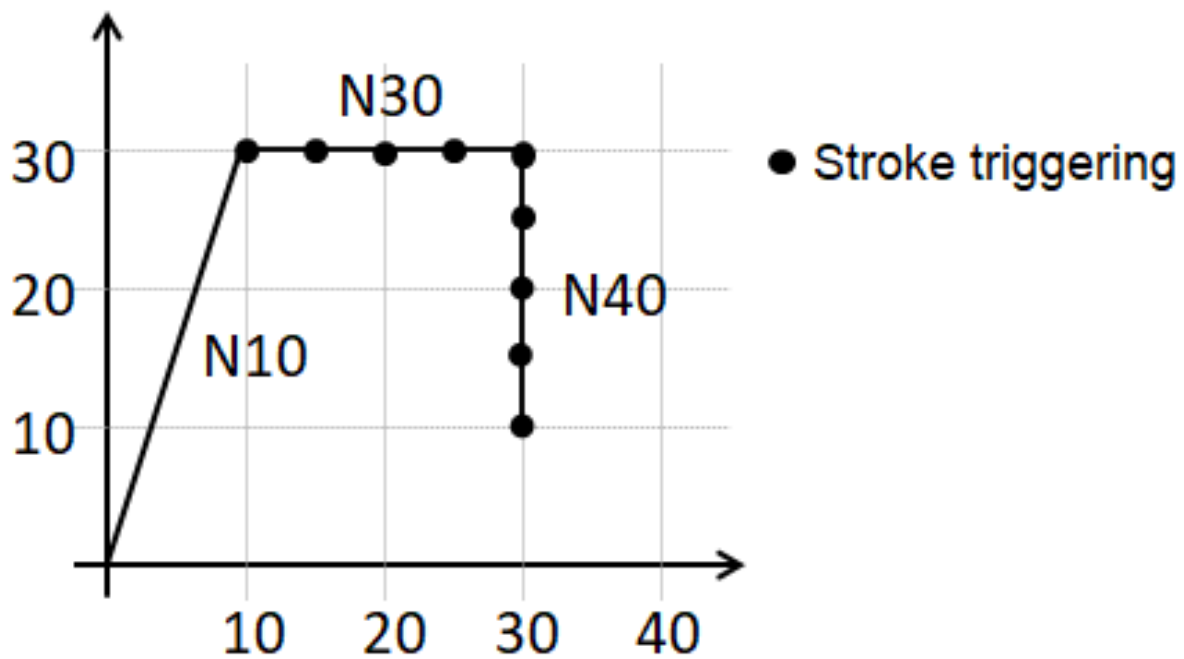
```
%Punch
N10 G90 G17
N20 #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50     Z-10
N60 #STROKE DEF END

N70   X10    Y30
N80 #PUNCH ON [LENGTH 5]
N90   X30    Y30
N100  X30    Y10
N110 #PUNCH OFF
N120 M30
```
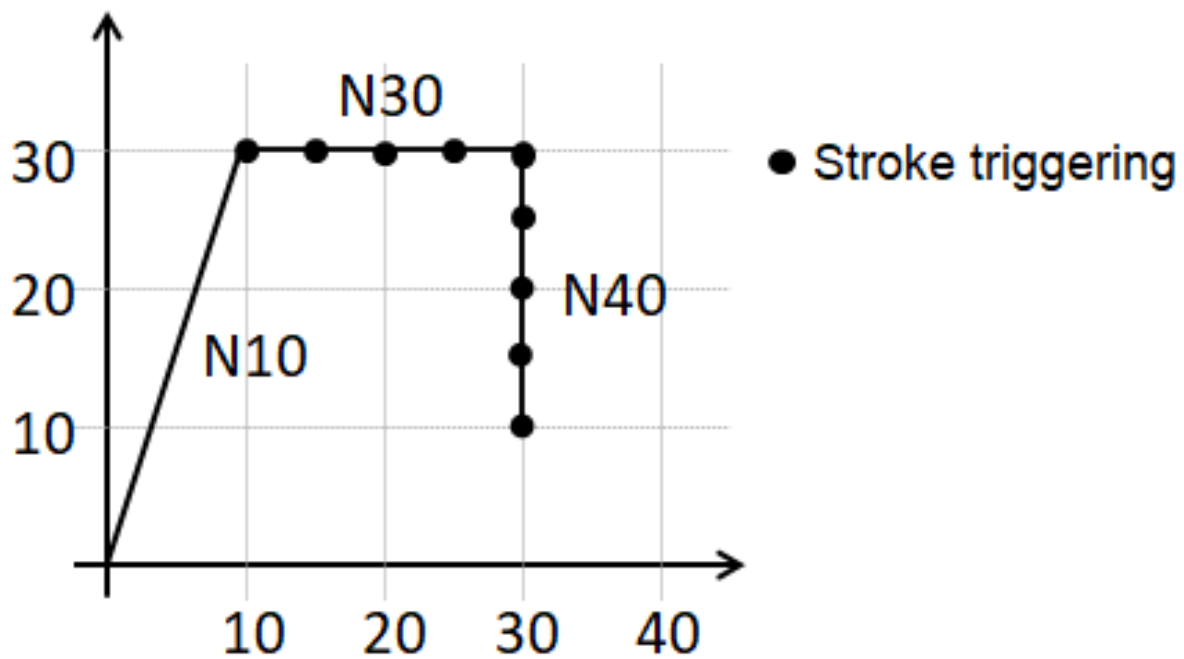

Configuration:

The following setting must be entered in P-STUP-00060 in order to use this function:

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_PUNCHING**

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_NIBBLING**

12.33.2   Further functions

Influencing the stroke triggering time:

The stroke triggering time can be influenced to achieve a good machining result. Stroke **pre-**triggering permits compensation of the constant dead time in signal processing. Stroke **post-**triggering can compensate for the transient response of the axes or, for example, for the hold-down time of pressure pads in punch operations.

Depending on the setting, influencing of the stroke triggering time refers to one of the following events:

- Interpolation target point reached →G00,G01,G02,G03
- In-Pos window of the slowest axes →G60

The dwell time G04 together with an M function is used for stroke post-triggering.

An M function of the MET_SVS type is used for pre-triggering.

> **ⓘ**   The offset timing resolution depends on the CNC controller cycle time as the smallest unit.

Stroke pre-triggering:

The user-specific M function M97 is to be output to the PLC 40 millimetres before the synchronisation time in the block sequence is reached.

Excerpt from the channel parameter list:

```
# Definition of M functions and synchronisation types
# =======================================================
m_synch[97] 0x02000000 MET_SVS

# Setting the pre-output time, pre-output path with MET_SVS type
# =============================================================
m_pre_outp[97] 40000 in us
N10 #STROKE DEF BEGIN
N30 M97
N40 #STROKE DEF END
```

Stroke post-triggering:

M96 is the user-specific M function for triggering the stroke motion. The stroke motion should only take place 50 milliseconds after the stroke positions are reached.

```
N10 #STROKE DEF BEGIN
N20 G04 0.05
N30 M96
N40 #STROKE DEF END
```

Example of output of the M function referred to the In-Pos window:

M96 is the user-specific M function for triggering the stroke motion. The stroke motion should only take place 50 milliseconds after the physical stroke positions of all axes are reached (actual position is in the window).

```
N10 #STROKE DEF BEGIN
N20 G04 0.05
N30 M96
N40 #STROKE DEF END
```

```
G90 G17
N10   X10      Y30
N20 #PUNCH ON [LENGTH 5]
N30   G60 X30     Y30
N40   G60 X30     Y10
N90 #PUNCH OFF
```

12.33.3   Restrictions

The use of rotations that tilt the active machining plane is not permitted. By contrast, rotations about the Z axis when the XY plane is active are permitted. This restriction applies only if axis motions are programmed in the stroke sequence. The output of M functions, for example, is still possible.

12.34 Controlling edge machining (#EDGE MACHINING)

Depending on the machining technology it may be necessary to control the machining process especially on sharp contours (edges). In the case of a sharp edge (defined by the angle difference between two contour elements), the path velocity at the edge is modified depending on pre-defined parameters. Linear or circular blocks can be programmed as contour elements. No check is made here whether they are inner or outer contours.

The edge machining function is configured in the channel parameter list and is effective as of program start. The exact description of parameters is described in the documentation [MDS-CHAN//Section Settings for edge machining].

In addition, the edge machining function can also be selected/deselected and parametrised in the NC program by the following NC command:

```
Syntax of Selecting edge machining:
```
#EDGE MACHINING ON [[**ANGLE_LIMIT**=..] [**WAIT_TIME**=..] [**PRE_DIST**=..] [**PRE_FEED**=..]

[**POST_DIST**=..] [**POST_FEED**=..] [**DISABLE_FEED_ADAPTION**=..] **]**

or together with setting default values

#EDGE MACHINING ON [[**ANGLE_LIMIT**] [**WAIT_TIME**] [**PRE_DIST**] [**PRE_FEED**]

[**POST_DIST**] [**POST_FEED**] [**DISABLE_FEED_ADAPTION**] **]**

or

#EDGE MACHINING ON DEFAULT

ON	Select edge machining.
ANGLE_LIMIT=..	Limit knee angle in [°]. If the knee angle between two contour elements is smaller than this critical angle, the special edge function is executed.
WAIT_TIME=..	Wait time in the edge in [s]
PRE_DIST=..	Distance after the edge in [mm, inch] for which a PLC signal is generated.
PRE_FEED=..	Feed before the edge in [mm/min, m/min, inch/min] used for the motion from PRE_DIST to the stop in the edge.
POST_DIST=..	Distance after the edge in [mm, inch] for which a PLC signal is generated.
POST_FEED=..	Feed after the edge in [mm/min, m/min, inch/min] used for the motion up to POST_DIST. The motion then continues at the originally programmed feed rate.
DISABLE_FEED_ ADAPTION=..	Switching feed adaptation, Boolean:
	0: Feed adaptation is active
	1: Feed adaptation is inactive, PRE_FEED and POST_FEED are not effective.
DEFAULT	When combined with ON selection, the default values apply from the channel parameter list [P-CHAN-00221 - P-CHAN-00226, P-CHAN-00300].

If all or some single keywords ANGLE_LIMIT, WAIT_TIME, PRE_DIST, PRE_FEED, POST_DIST, POST_FEED or DISABLE_FEED_ADAPTION and programmed without values, the default values from the channel parameter list [P-CHAN-00221 - P-CHAN-00226, P-CHAN-00300] are used.

The command #EDGE MACHINING OFF deselects the edge machining function. In combination with deselection, no other parameters may be specified.

```
Syntax of Deselecting edge machining:
```

#EDGE MACHINING OFF

Configuration:

To use this function, the following setting must be made in the start-up list ([STUP]):

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_EMF**

NOTICE
It is permitted to change parameters when edge machining is active. However, it is recommended to program the change only after relevant contour elements are completed. This avoids an undefined reaction of the edge machining function.

Control edge machining

```
%edge_machining

(Select edge machining and set parameters with value)
#EDGE MACHINING ON [ANGLE_LIMIT=90 WAIT_TIME=0.2 PRE_DIST=5 PRE_FEED=800
POST_DIST=10 POST_FEED=1600]

(Select edge machining and set parameters to default)
#EDGE MACHINING ON [ANGLE_LIMIT PRE_DIST PRE_FEED POST_DIST POST_FEED
WAIT_TIME DISABLE_FEED_ADAPTION]

or

(Select edge machining and set parameters to default)
#EDGE MACHINING ON DEFAULT

(Deselect edge machining)
#EDGE MACHINING OFF
```

BECKHOFF

12.35 Switching dynamic weighting (#DYNAMIC WEIGHT)

Path and radius dependent weighting of dynamic limits are configured in the channel parameter list and activated by parameters P-CHAN-00190 and P-CHAN-00230. Configuration examples for path-dependent weighting and radius-dependent weighting are contained in [CHAN].

The weightings are then active after control start-up.

The following NC command can switch the dynamic weightings during NC program runtime, i.e. they can be activated and deactivated as required.

Syntax:

#DYNAMIC WEIGHT [**ON** | **OFF**] **[** [**PATH** | **CURVE**] **]**

ON	Weighting on
OFF	Weighting off
PATH	Path length dependent weighting (see P-CHAN-00190)
CURVE	Circle radius dependent weighting (see P-CHAN-00230)

i For activation or deactivation, at least one keyword PATH or CURVE must be programmed.

Switch dynamic weighting

```
N20 G00 X0 Y0 Z0 F10000
N30 #DYNAMIC WEIGHT ON [PATH]    ;Select path-dependent dynamic weighting
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 #DYNAMIC WEIGHT OFF [PATH]  ;Deactivate
N100 M30
```

12.36 Weighting of external feedrate (# FF)

The NC channel includes the option of commanding an external path feed via the PLC [HLI// Control commands of a channel]. This can be additionally weighted in the NC program by the feed factor #FF.

The effect of #FF is controlled by setting the parameter P-CHAN-00422.

Syntax:

#FF = *<feed_factor>*

<feed_factor>   Weighting of external feed in [%]

●

ℹ **The feedrate factor #FF only acts on the specified external feed.**
Before Build V3.01.3079.0, the feed factor #FF is limited to maximum 10000% and minimum to >0. As of this build, it is limited to maximum 1000000% and minimum to 0.1%.

The NC command is available as of CNC Build V3.01.3064.02.

12.37 Axis clamping and monitoring

To ensure high quality and accuracy in a machining process (milling, turning, erosion machining, etc.), the tool-carrying axis can by physically fixed by the drive brake. Mechanical axis clamping is commanded by synchronised M/H functions in the NC program and enabled in the PLC. This ensures that an axis is clamped before the next motion block is executed.

The status 'clamped axis' is effective at axis driver level and remains valid cross-channel until it is deselected, even with axis exchange operations.

While an axis is clamped, neither the interpolator nor an additional interface can command it to drive by command values. The monitor required here is enabled and disabled by the following NC command. If a clamped axis is to be moved, the error message P-ERR-70525:is output and the CNC changes to error state.

The monitoring status is modal and remains active even after NC program end or RESET.

```
Syntax of Monitoring specific clamped axes:
```
#CLAMP MONITORING ON | OFF [{ **AX**=*<axis_name>* | **AXNR**=.. } **]**

AX=*<axis_name>*	Name of clamped axis to be monitored.
AXNR=..	Logical number of the axis to be monitored, positive integer

```
Syntax of Monitoring all clamped axes:
```
#CLAMP MONITORING ON | OFF ALL

Axis clamping and monitoring

```
%clamp.nc
N010 X0 Y0 Z0 A0 B0 C0
N020 A[M300] C[M300] ;Mechanical clamping of axes A + C on
N030 #CLAMP MONITORING ON ALL ;Enable monitoring of axes A + C
..
N100 #CLAMP MONITORING OFF ALL ;Disable monitoring of axes A + C
..
N110 #CLAMP MONITORING ON [AX=A AX=C] ;Enable monitoring of axes A + C
N120 Y10
N130 X15
N140 #CLAMP MONITORING OFF [AX=A AX=C] ;Disable monitoring of axes A + C
N150 X10
N160 #CLAMP MONITORING ON [AXNR=4 AXNR=6] ;Enable monitoring of axes A + C
N120 Y20
N130 X25
N150 A[M301] C[M301] ;Release mechanical clamping of axes A + C
N140 #CLAMP MONITORING OFF [AXNR=4 AXNR=6] ;Disable monitoring of axes A + C
..
N999 M30
```

ℹ The command does not cause any mechanical axis clamping. Instead, it enables the monitor for un-authorised axis motions. With the function 'Backward motion on path', the monitor status is inverted accordingly.

ℹ Axis motions which are generated by active compensation functions in the axis driver (e.g. Cross compensation, Volumetric Compensation, etc.) are not monitored.

The monitoring status of an axis can be read by the ADS access.

Index group: 0x20300

Index offset: 0x10189 (axis 1)

.

12.38   Gantry start-up

NOTICE
Only use the #GANTRY command for start-up
Possible machine damage if command used incorrectly.

For more information, see [FCT-C11// Gantry start-up]

Syntax for Disable gantry combination:

#GANTRY OFF [{ **AXNR**=.. | **AX**=*<axis_name>* } **]**

AXNR=..   Logical axis number (P-AXIS-00016) of master axis

AX=*<axis_name>*   Name of the master axis of a gantry combination

Syntax for Disable all gantry combinations:

#GANTRY OFF ALL

Syntax for Restore a gantry combination:

#GANTRY ON [{ **AXNR=..** | **AX=**<*axis_name*> } **]**

AXNR=..	Logical axis number (P-AXIS-00016) of master axis
AX=<*axis_name*>	Name of the master axis of a gantry combination

Syntax for Restore a gantry combination:

#GANTRY ON ALL

12.39 Position controller-based axis couplings (#GEAR LINK)

The motion of a **target axis** by a position controller-based axis coupling is influenced additively or even exclusively by the motions of other axes. The target axis is either an additional interpolator axis or a spindle.

Axes that influence a target axis via coupling specifications are referred to as **source axes**. The motion part of sources axes are weighted accordingly by defining specific factors. This permits the implementation of an electronic gear.
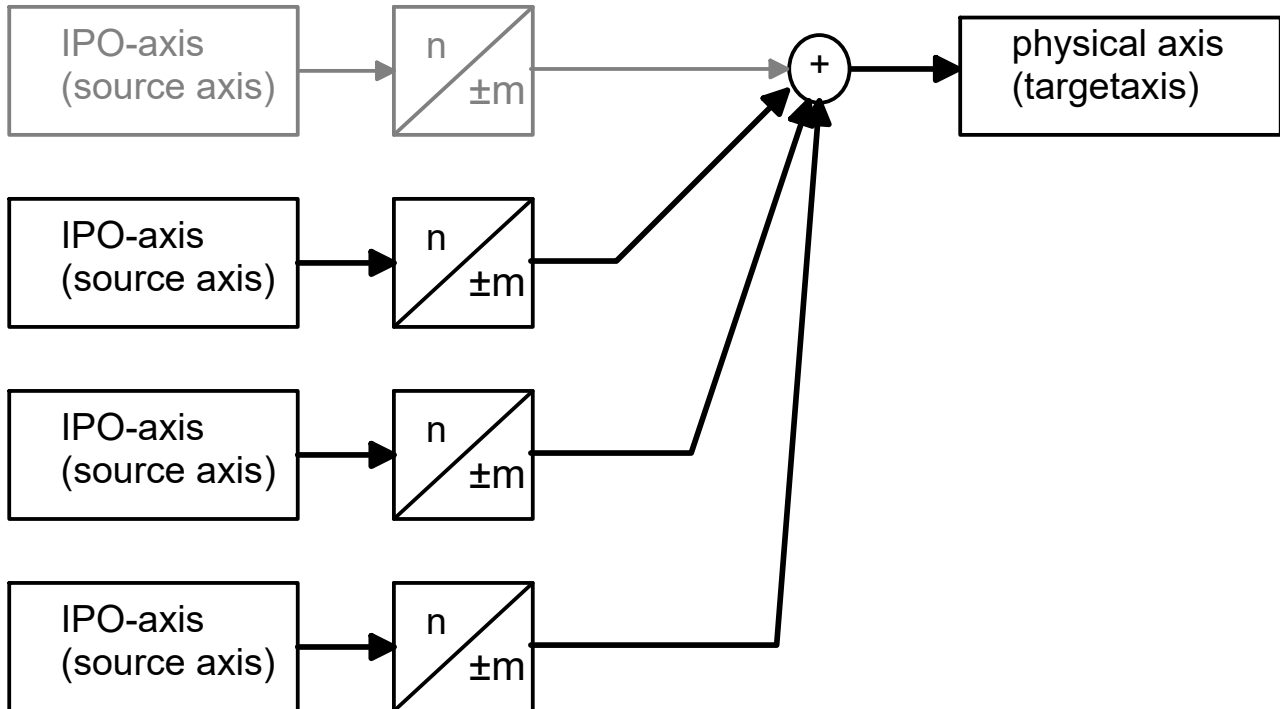

Fig. 121: View of the difference between source and target axis

This function can be controlled by the HLI and is described in detail in [FCT-A9]. The NC command #GEAR LINK parameterises and enables this type of axis coupling, even in a subroutine.

```
Syntax of GEAR-LINK programming, parameterisation by axis names:
```
#GEAR LINK [TARGET=*<axis_name>* **AX***<i>***=***<axis_name>* **NUM***<i>***=**.. **DENOM***<i>***=**.. {**AX***<i>***=***<axis_name>*

        NUM*<i>***=**.. **DENOM***<i>***=**..} [**MODE**=*<mode>*] [**ACC**=..] { **** } **]**

TARGET=*<axis_name>*	Name of coupling target axis
AX<i>=*<axis_name>*	Name of source axis i where i =1 .. 4
	Caution: Only channel-specific axis names may be programmed as they are the only ones that are unique.
NUM*<i>*=..	Coupling factor NUM<i> / DENOM<i> for source axis i,
DENOM*<i>*=..	numerator and denominator are negative or positive integers.
MODE=*<mode>*	Mode for coupling/decoupling:
	DIRECT: Coupling requires all the axes involved to be at standstill (default). If this is not the case, stop occurs with error message P-ERR-70200
	SOFT: Soft coupling using slope; axes can move
ACC=..	Acceleration limit on the axis with coupling/decoupling in [mm/sec², inch/sec²]. If ACC is not specified, P-AXIS-00053 is used as default acceleration limit.
\	Separator ("backslash") for clear programming of the command over multiple lines.

```
Syntax of GEAR-LINK programming, parameterisation by axis numbers:
```

#GEAR LINK [TARGETNR=.. **AXNR**<i>**=.. NUM**<i>**=.. DENOM**<i>**=.. {AXNR**<i>**=..**

   NUM<i>**=.. DENOM**<i>**=..} [MODE**=<mode>**] [ACC**=..**] { \ }]**

TARGETNR=.	Logical number of coupling target axes, positive integer
AXNR<i>=..	Logical number of source axis i where i =1 .. 4, positive integer
	Caution: All logical axis numbers known in the system may be programmed as they are always unique in multi-channel systems.
NUM<i>=..	Coupling factor NUM<i> / DENOM<i> for source axis i;
DENOM<i>=..	numerator and denominator are negative or positive integers.
MODE=<mode>	Mode for coupling:
	DIRECT: Coupling requires all the axes involved to be at standstill (default). If this is not the case, stop occurs with error message P-ERR-70200
	SOFT: Soft coupling using slope; axes can move. If ACC is not specified, P-AXIS-00053 is used as default acceleration limit.
ACC=..	Acceleration limit on the axis with coupling/decoupling in [mm/sec², inch/sec²]
\	Separator ("backslash") for clear programming of the command over multiple lines.

```
Syntax of Enabling and parameterising in one step:
```

#GEAR LINK ON [.. **]**

…or disable only

A previously parameterised coupling is enabled. An error message is output if no coupling was parameterised.

#GEAR LINK ON [TARGET=<axis_name> | **TARGETNR**=.. **]**

…or disable only:

#GEAR LINK OFF [TARGET=<axis_name> | **TARGETNR**=.. **]**

Response at program end and reset:

Active axis couplings are disabled at program end and reset.

Defining and selecting a coupling specification

```
Parameterisation by axis name:
#GEAR LINK [TARGET=U AX1=X AX2=Y NUM1=1 DENOM1=2 NUM2=-1 DENOM2=1\
            MODE=SOFT ACC=400]
:
TARGET=U          ;Target axis is U AXIS
AX1=X             ;Source axis 1 is X axis
NUM1=1 DENOM1=2   ;Coupling factor for X axis is NUM1/DENOM1=0.5
AX2=Y             ;Source axis 2 is Y axis
NUM2=-1 DENOM2=1  ;Coupling factor Y axis is NUM2/DENOM2=-1
MODE=SOFT         ;Soft coupling/decoupling
ACC=400           ;Acceleration limit on the axis at 400 mm/sec²

Enable coupling:
#GEAR LINK ON [TARGET=U]
Disable coupling:
#GEAR LINK OFF [TARGET=U]


Alternative parameterisation by axis number:
#GEAR LINK [TARGETNR=4 AXNR1=1 AXNR2=2 NUM1=1 DENOM1=1 NUM2=-1 \
            DENOM2=1]
Enable coupling:
#GEAR LINK ON [TARGETNR=4]
Disable coupling:
#GEAR LINK OFF [TARGETNR=4]
```

12.40 Settings for turning functions (# TURN)

i ● **This function is available as of CNC Build V3.1.3079.03.**

The #TURN command can influence rotary functions.

Syntax:

#TURN [[**ROT_FEED_CPL=**..] [**TAPPING_ACT_POS=**..] [**TAPPING_N_CYCLES=**..]**]**

ROT_FEED_CPL=..	Influence of axis couplings on the revolution feedrate with G95 [▶ 602]
	0 : Axis couplings are not considered (default)
	1 : Axis couplings are considered
TAPPING_ACT_POS=..	Activate tapping with actual spindle positions.
	1: The linear axes are coupled to the actual spindle positions.
	0: The spindle is coupled to the command positions of the linear axis.
	Available as of V3.1.3080.04
TAPPING_N_CYCLES=..	Number of filter cycles to filter actual spindle positions. Value range: 0 … 20
	0: Filter is deactivated.
	Available as of V3.1.3080.04

BECKHOFF

12.41 Distance to go display in a program section

ℹ **This function is available as of CNC Build V3.1.3079.27.**

A geometry sequence can be flagged in the NC program with the NC commands **#DIST TO GO BEGIN/ END** . Within a sequence, the distance to go of each axis is displayed on the HLI up to the end of the geometry sequence by dist_to_geom_end.

Requirement: The parameter P-STUP-00033 must be parameterised in order to use the display function.

```
Syntax:
```

#DIST TO GO BEGIN

#DIST TO GO END

Characteristics:

- The distance to go of the axis is always displayed in forward direction. Backward motion on the path makes the distance displayed greater.

- The displayed values are no longer correct if a geometry changes takes place with the "Delete distance to go" function. This error is signalled by the warning ID 51047.

- The entire geometry sequence is required to obtain a valid display of the axis distances to go up to #DIST TO GO END.
  Especially with large program sequences and small values defined in P-STUP-00033, this may result in a delay of the display datum dist_to_geom_end and the warning ID 51048.
  If the values in P-STUP-00033 are too small, the sequence is output without stopping and the display datum dist_to_geom_end remains at 0 until a valid value can be determined for the display.

- If NC commands that result in a channel synchronisation are programmed within the flagged geometry sequence. The commands prevent the look-ahead distance to go calculation up to #DIST TO GO END. The distance to go display is then only displayed correctly after this NC command.
  Example of these NC commands:

  - #CHANNEL INIT [▶ 169]

  - #FLUSH WAIT [▶ 322]

  - or reading a synchronous V.E. variable

Distance to go display in a program section

```
N010 G01 X0 Y0 F5000
N020 #DIST TO GO BEGIN
N030 G01 X10
N040 G01 Y10
N050 G01 X5
N060 #DIST TO GO END
; …
```

In the example, the display would show at the start of the sequence for X 15 and Y 10 and 0 in each case at the end.

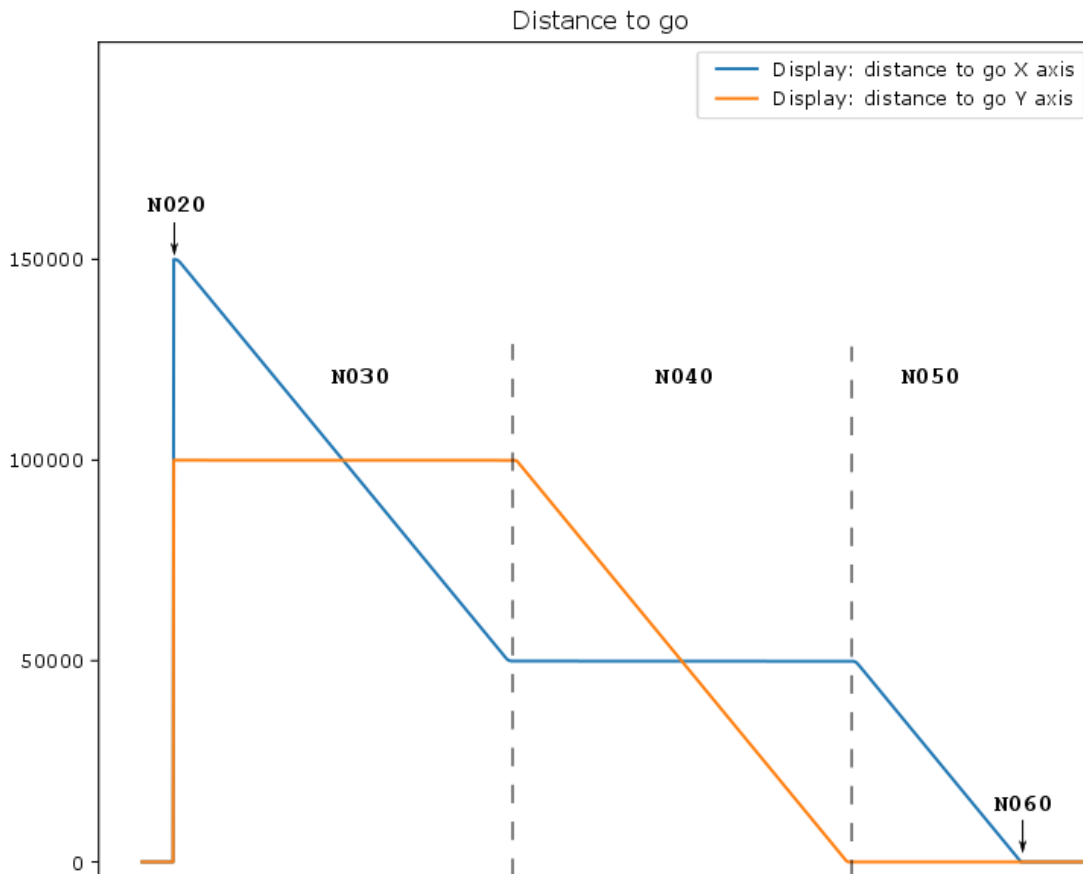The display values is presented in the figure below:

Fig. 122: Distance to go display in a program section

13   Tool geometry compensation (D)

Tool geometries are compensated for length, radius and axis offset. The corresponding compensation data for tool geometry are provided either by:

tool data lists containing data records for each tool and loaded at controller start-up (see [TOOL]), or an external tool management system.

Tool compensation data

Tool compensation data is selected using the D word or is selected automatically by the T word (P-CHAN-00014) if this is parameterised.

Syntax:

D..	Selecting tool compensation with number of the compensation data block. Positive integer.	modal

The D word defines a tool data block that contains the following values:

- Tool length (perpendicular to main plane)
- Tool radius (in the main plane)
- Axis offset coordinates (in all axis directions)
- Measuring unit of the numerical statements (mm/inch)
- Tool validity code
- Tool dynamic data

NOTICE
The time when the tool compensation data for the tool length (perpendicular to the main plane) and the axis offset coordinate (in the axis directions) become effective is programmed by the channel parameter P-CHAN-00100. The lines below must also be noted:

A distinction is made between 2 modes:

The compensating movement is executed directly with the D word without specifying a path preparatory function.

For safety reasons, the controller only executes the compensating movement at the next absolute motion block in the corresponding axis (default mode).

In G91 mode, the following applies to the inclusion of tool compensation data:

The programming of...

```
N10 D16
N20 G0 X0 G91
```

... may not cause any movement of the X axis (corresponds to relative movement about 0). In this case, tool compensation data for this axis only acts if the next motion information is programmed in absolute coordinates (G90).

Please note for both modes together: The tool radius is transferred to the tool radius compensation and influences the equidistant calculation. The compensating movement always take place combined with a path preparatory function.

The rules for execution of the compensating movement also apply to deselection of tool compensation.

Syntax:

D00	Deselect tool compensation	modal

13.1   Tool length compensation

On Cartesian machines, tool length compensation (TLC) always acts in the direction of the 3rd main axis. With G17, this is generally the tool head axis.

If the tool is oriented in the negative direction to the Z axis, a compensating motion takes place to the positive direction of the Z axis.
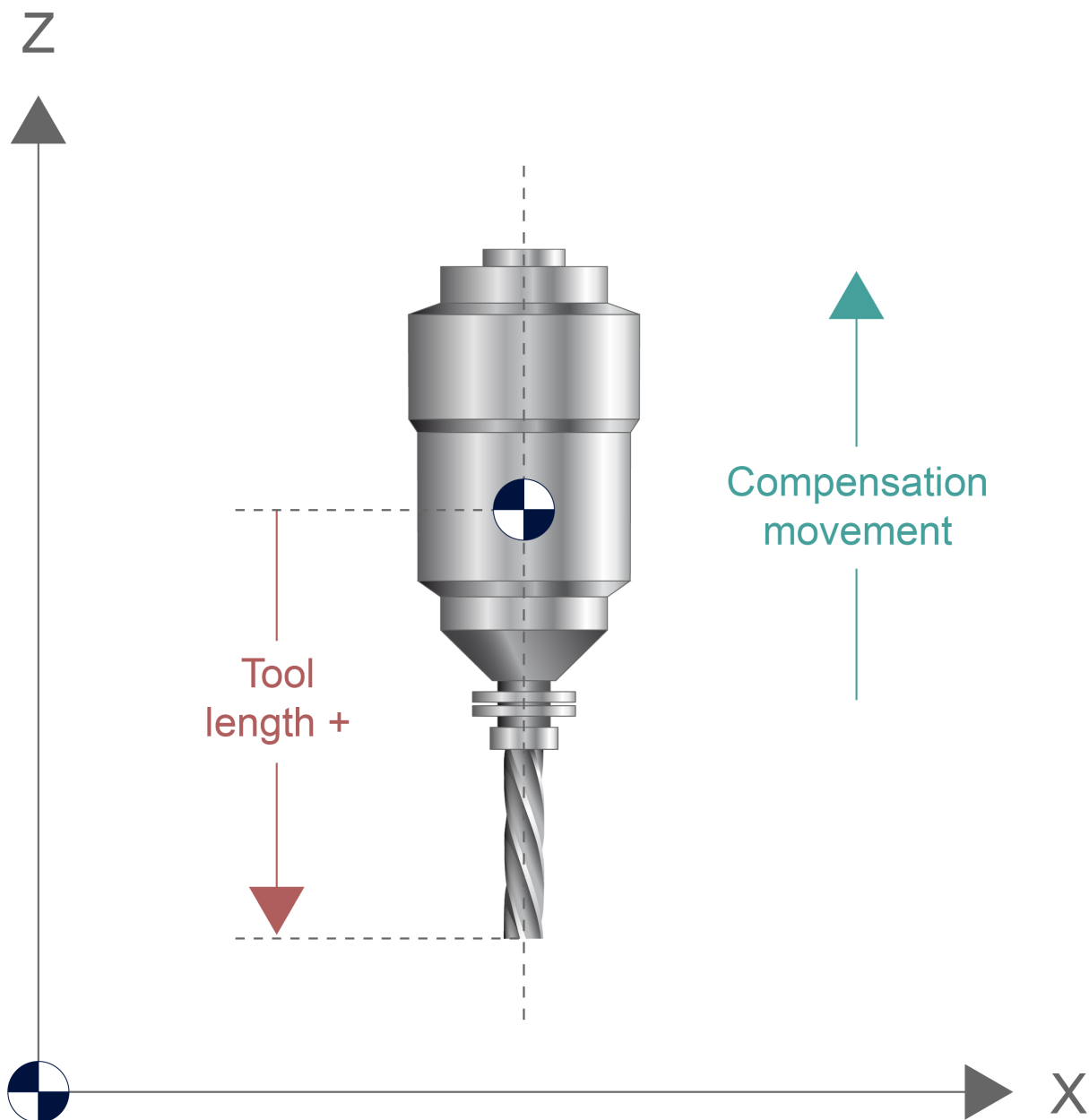

Fig. 123: Compensate tool length by compensating motion

NOTICE
If the mechanical machine structure defines a different tool head axis, the user must choose (G18, G19, #CS, #TRAFO) a suitable programming coordinate system (PCS) in order to calculate the tool length correctly.

In the example below, tool length compensation is carried out in the Z direction. When the compensation data block D16 is selected in block N30, the compensation movement occurs in the Z direction jointly with the motion data block in N30.

```
N10 G01 F900 G17 ;X-Y plane, length compensation in Z+ (default))
N20 X150 Y10 Z10
N30 D16 Y40 Z15 ;Select length compensation D16. The
:                ;compensation motion is executed.
:
N100 D20 ;Select tool compensation D20. The
:         ;compensation motion only occurs with
:         ;the next absolute motion block in the Z direction.
:
N200 G0 D0 X0 Y0 Z0 ;Deselect TLC

Compensation data block
D0: Length = 0 Radius = 0
D16: Length = 5 Radius = 5
D20: Length = 12.5 Radius = 5
```
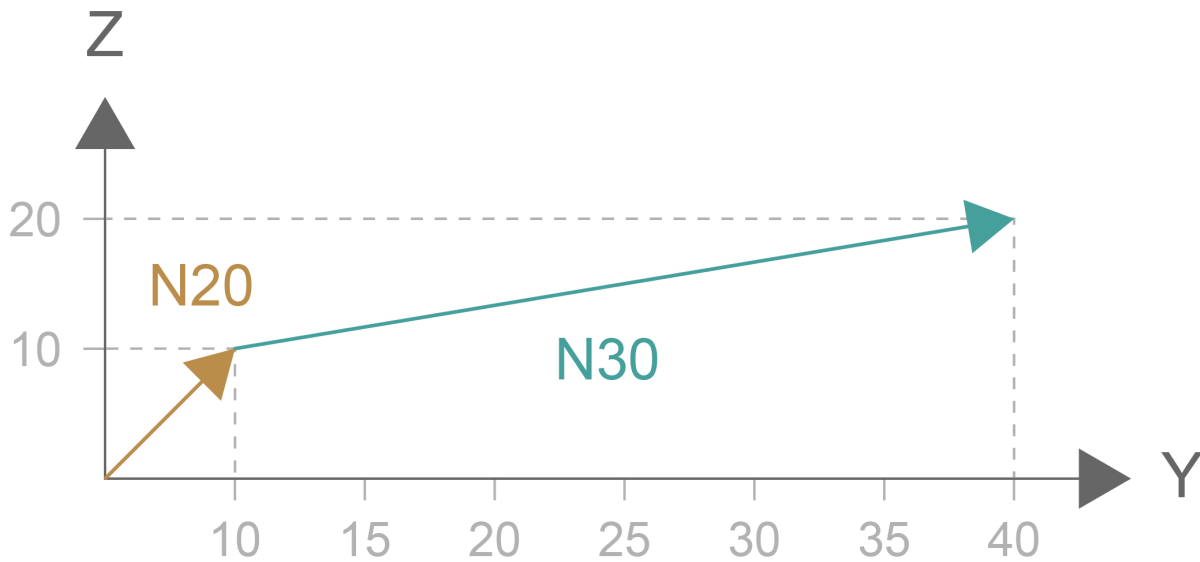

Fig. 124: Example of tool length compensation

13.1.1 Axis-specific assignment of tool length compensation (#TLAX, #TLAX DEFAULT)

In initial state, the tool length is always considered in the third main axis of the new place when the plane changes (G17, G18, G19). This is useful if the tool head (or tool axis) is orientable and machining takes place in the relevant plane.

This behaviour is not required with non-orientable (fixed) tool heads. If a change in place is only intended to program an approach motion, the tool length and its orientation can remain in the original main axis or can be assigned fixed to a specific main axis by the following NC command:

Syntax:

#TLAX [*<axis_name>* **+** | **-]**

    <axis_name>   Name of main axis in which tool length is included and orientation is specified + or -

The following rule applies to orientation assignment:
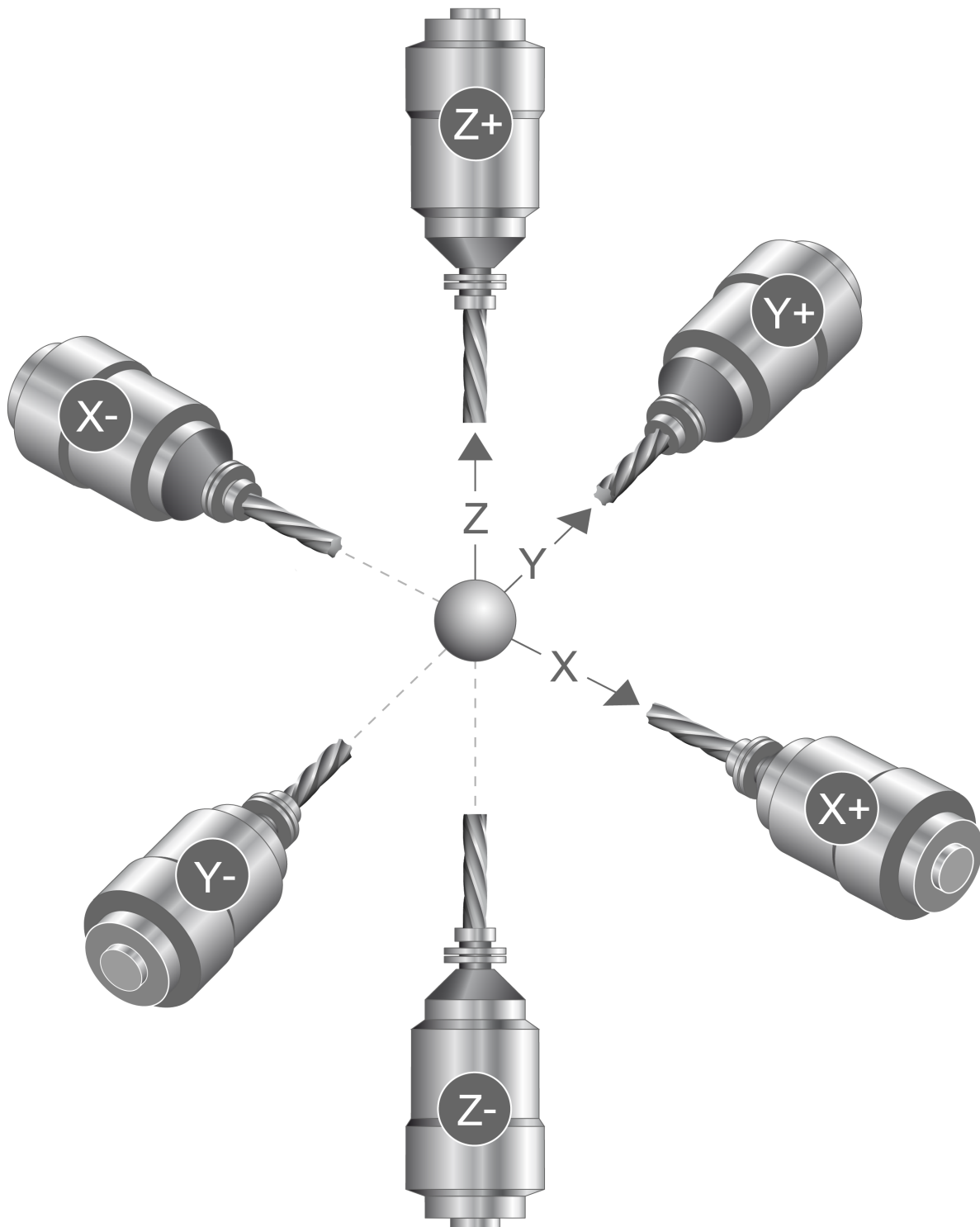
Fig. 125: Assignment rule of tool length compensation

In initial state (G17), tool length compensation is included in the third (Z). The direction (orientation) is specified by the sign '+'. This corresponds in the X-Y plane to the command #TLAX [Z+].

On machines with non-orientable tool heads that are mainly used for 3D machining in G17, this behaviour can also be preset by the channel parameter P-CHAN-00420; it is then not necessary to program #TLAX.

The command below cancels the fixed assignment of tool length compensation to a specific axis. Tool length is then considered in the third main axis of the current plane again.

```
Syntax:
```

#TLAX DEFAULT

```
Contour circular motions in different planes with tool length compensation in constant orientation
%tlax
N010 G0 X0 Y0 Z0
N020 V.G.WZL=33 G161
N030 X0 Y0 Z0
N040 #TLAX [Z+]  ;Tool length compensation in Z+
N050 G18
N060 G01 X0 Z50 F2000
N070 G02 X50 Z0 I50 K50 F1000
N080 G17
N090 G03 X100 Y50 I50 J50
N100 G19
N110 G03 Y0 Z50 J50 K50 F1000
N120 G18
N130 G02 X0 Z50 I50 K50 F1000
N140 #TLAX DEFAULT  ;Deselect tool length compensation in Z+,
                    ;Tool length is included in Y (G18)
N150 G17
N160 M30
```
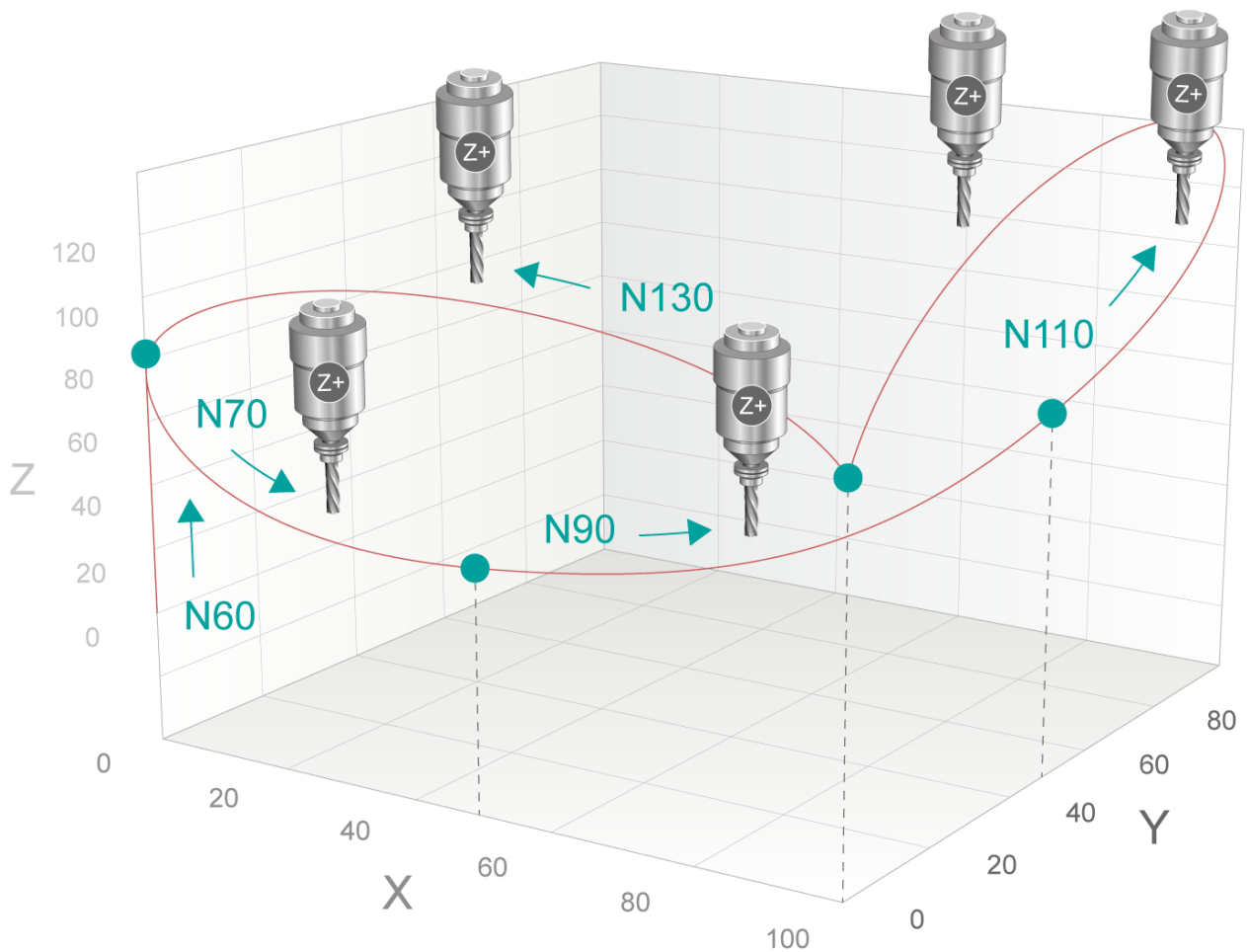

Fig. 126: Contour approach motions in G17, G18, G19 with tool length compensation in constant orientation

13.2 Tool radius compensation (TRC)

Tool radius compensation (TRC) allows programming of the workpiece contour independently of tool geometry. If TRC is selected (G41, G42), a tool path equidistant from this programmed tool contour is calculated at distance "tool radius".

Tool radius compensation acts in the plane selected with G17, G18 or G19. The tool compensation values used are the tool compensation synchronisation modes stored under the D words (see Tool geometry compensation [▶ 470]).

Tool radius can also be changed when TRC is active by selecting a new D word or writing the variable V.G.WZ_AKT.R.

When a negative tool radius is used, the selection side of the TRC is changed automatically.

Fig. 127: Mode of operation and terms of tool radius compensation

Overview of all TRC-relevant G functions:

Selecting/deselecting TRC

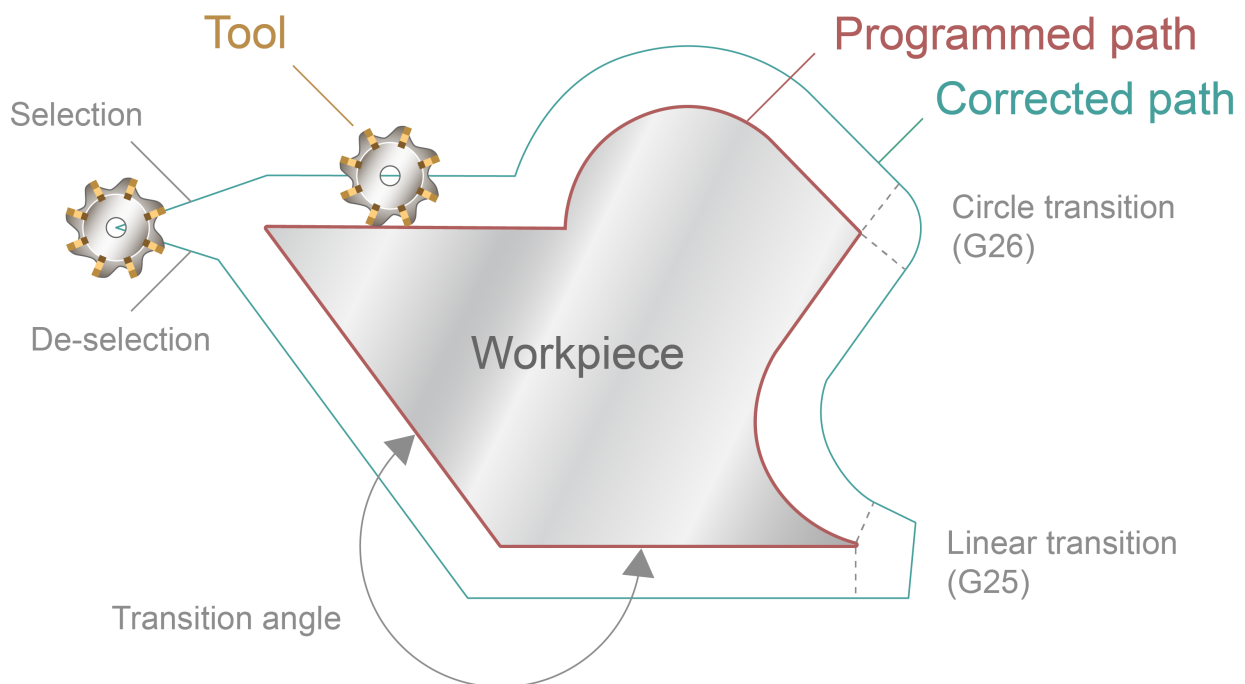| G40	Deselecting TRC	(modal, initial state)
G41	Selecting TRC left of contour	(modal)
G42	Selecting TRC right of contour	(modal)

Selecting/deselecting TRC

G138 [▶ 484]	Direct TRC selection/deselection	(modal)
G139 [▶ 488]	Indirect TRC selection/deselection	(modal, initial state)
G236 [▶ 512]	Direct TRC selection/deselection on the path	(modal)
G237 [▶ 498]	Perpendicular TRC selection/deselection	(modal)
G238 [▶ 504]	Selecting inside corner of TRC	(modal)
G239 [▶ 507]	Selecting/deselecting TRC directly without block	(modal)
G05 [▶ 526]	Tangential TRC selection/deselection	(non-modal)

NOTICE
It is not permitted to change the modal selection method to G238 when TRC is selected.

Feed adaption:

G10 [▶ 529]	Constant feedrate	(modal, initial state)
G11 [▶ 529]	Adapted feedrate	(modal)

Contour masking:

G140 [▶ 530]	Deselecting contour masking	(modal, initial state)
G141 [▶ 530]	Selecting contour masking	(modal)

Selecting external transitions:

Outside corners must be bypassed if tool radius compensation is selected. For this purpose, tool radius compensation inserts transition blocks. Use G25 and G26 to select between linear block insertion and circular block insertion.

G25	Linear transition	(modal, initial state)
G26	Circular transition	(modal)

Presentation of the different selection options when tool radius compensation is used. This shows the commands for direct, indirect and tangential selection/deselection of TRC (G139/G138/G05) in combination with the TRC transition types for linear and circular transition (G25/G26).The test programs are executed with a tool radius of 10 mm.

Example 1	G138	G139	G05
G25	**Path 1**	**Path 2**	**Path 3**

```
%WZKG25 (Contour for G25)
N1 G00 G90 T1 D1 X0 Y0 Z0 G17
(Display of contour)
N15 G01 X20 Y20 F1000
N20 G91
N25 G1 X10
N30 X5 Y-5
N35 Y-5
N40 X-5 Y-3
N45 G01 G90 X0 Y0 F1000

(Path 1)

N100 G138 G41          (Select directly and TRC left of contour)
N105 G01 X20 Y20 F1000 (Required compensation motion after G41)
N110 G25               (G25 linear transitions)
N115 G1 G91 X10
N120 X5 Y-5
N125 Y-5
N130 X-5 Y-3
N135 G138 G40          (Deselect directly and deselect TRC)
N140 G01 G90 X0 Y0 F1000 (Required compensation motion after G40)

(Path 2)

N200 G139 G41          (Select directly and TRC left of contour)
N205 G01 X20 Y20 F1000 (Required compensation motion after G41)
N210 G25               (G25 linear transitions)
N215 G1 G91 X10
N220 X5 Y-5
N225 Y-5
N230 X-5 Y-3
N235 G139 G40          (Deselect indirectly and deselect TRC)
N240 G01 G90 X0 Y0 F1000 (Required compensation motion after G40)

(Path 3)

N300 G05 G41           (Tangential selection and TRC left of contour)
N305 G01 X20 Y20 F1000 (Required compensation motion after G41)
N310 G25               (G25 linear transitions)
N315 G1 G91X10
N320X5 Y-5
N325 Y-5
N330 X-5 Y-3
N335 G05 G40           (Tangential deselection and deselect TRC)
N340 G01 X20 Y20 F1000 (Required compensation motion after G41)

N999 M30
```
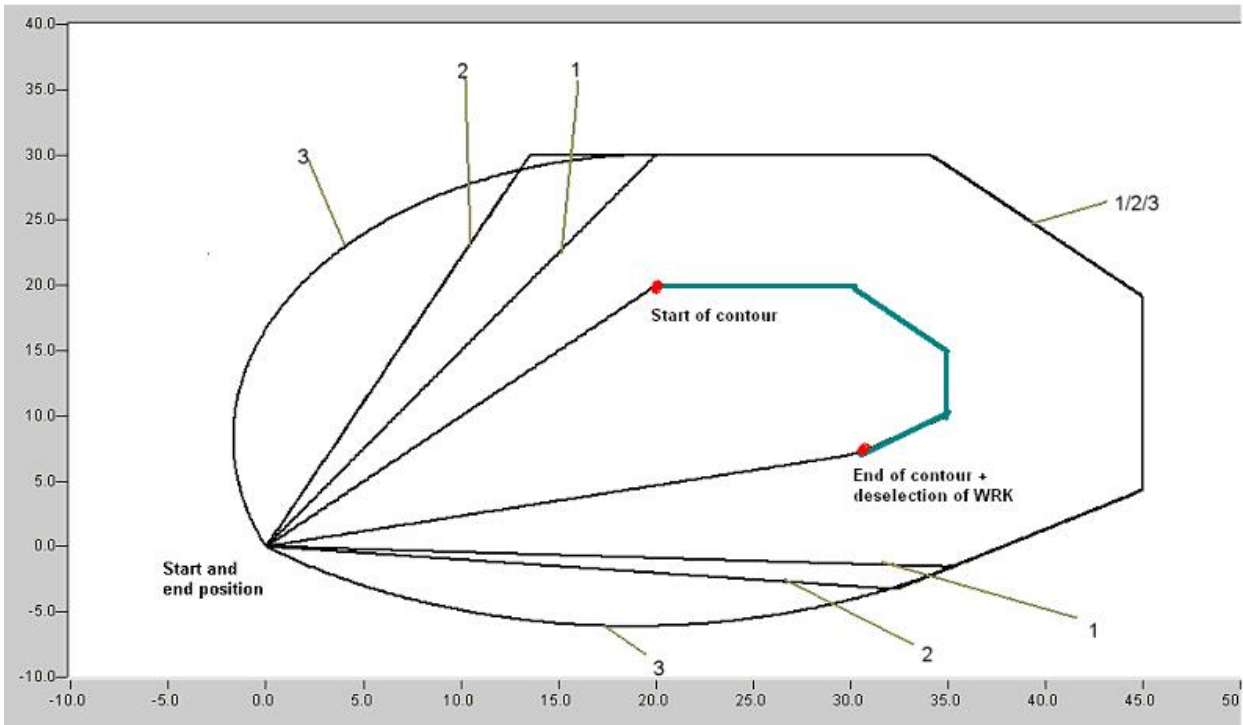
Example 2	G138	G139	G05
G26	**Path 4**	*Path 5*	**Path 6**

```
%WZKG26 (Contour for G26)
N10 G00 G90 T1 D1 X0 Y0 Z0 G17
(Display of contour)
N15 G01 X20 Y20 F1000
N20 G91
N25 G1 X10
N30 X5 Y-5
N35 Y-5
N40 X-5 Y-3
N45 G01 G90 X0 Y0 F1000

(Path 4)

N400 G138 G41           (Select directly and TRC left of contour)
N405 G01 X20 Y20 F1000  (Required compensation motion after G41)
N410 G26                (G26 Circular transitions)
N415 G1 G91 X10
N420 X5 Y-5
N425 Y-5
N430 X-5 Y-3
N435 G138 G40           (Deselect directly and deselect TRC)
N440 G01 G90 X0 Y0 F1000 (Required compensation motion after G40)

(Path 5)

N500 G139 G41           (Select directly and TRC left of contour)
N505 G01 X20 Y20 F1000  (Required compensation motion after G41)
N510 G26                (G26 Circular transitions)
N515 G1 G91 X10
N520 X5 Y-5
N525 Y-5
N530 X-5 Y-3
N535 G139 G40           (Deselect indirectly and deselect TRC)
N540 G01 G90 X0 Y0 F1000 (Required compensation motion after G40)

(Path 6)

N600 G05 G41            (Tangential selection and TRC left of contour)
N605 G01 X20 Y20 F1000  (Required compensation motion after G41)
N610 G26                (G26 Circular transitions)
N615 G1 G91X10
N620 X5 Y-5
N625 Y-5
N630 X-5 Y-3
N635 G05 G40            (Tangential deselection and deselect TRC)
N640 G01 X20 Y20 F1000  (Required compensation motion after G41)

N999 M30
```
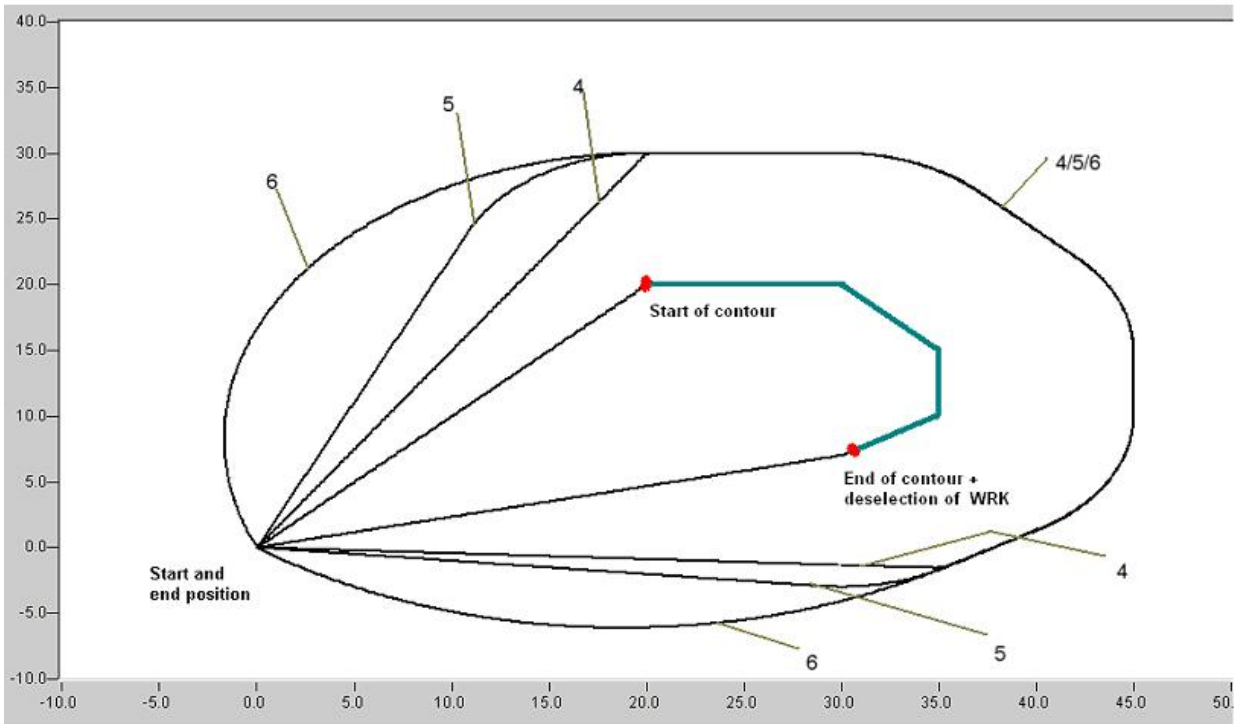
BECKHOFF

Change tool data

Change tool data

```
:
N30 G0 D0 X0 Y0 Z0 G17   (X-Y plane)
N40 G0 D100 X10 Y10      (Select TLC)
N50 G1 Z0
N60 G0 Z100
N70 G41                  (Select TRC with data block D100)
N80 G1 Z0
N90 G2 X10 Y10 I-15      (Full circle with radius 15)
N100 G0 Z100
N110 D2 Z200             (Other compensation data block, i.e.)
(other values for TLC and TRC)
N120 G1 Z0               (Compensating motion)
                         (motion of TLC takes place here)
N130 G1 X20 Y20          (Compensating motion of TRC)
N140 G02 X10 Y10 I-15    (TRC is now executed with data)
                         (block D2)
N150 G0 Z100
N160 G40                 (Deselect TRC)
N170 X0
:
```

Dynamic change of tool radius:

Another option to change tool radius is to assign variables (see also Chap. 13 [▶ 533]). For example, this takes into consideration the wear and tear of grinding tools during motion blocks.

Dynamic change of tool radius:

```
N00 G1 G90 X0 D6 F5
N10 G41 G26 (Select TRC)
N20 X0 Y250                    (Starting point)
N30 V.P.VERSCHLEISS = 0.010
N100 $FOR V.P.LAUF = 0,100,10  (Grinding cycle)
N110 X300
N120 Y200
N130 X0
N140 Y250                      (Tool radius gradually becomes smaller)
N150 V.G.WZ_AKT.R = V.G.WZ_AKT.R - V.P.VERSCHLEISS
N160 $ENDFOR

N200 G40 X300                  (Deselect TRC)
N999 M30
```

13.2.1   Direct/indirect deselection of TRC

When TRC is selected directly or indirectly, an approach block is generated in the selected compensation plane with the next motion block in the NC program.

When TRC is selected, G138 selects an alternative approach strategy (direct selection of TRC) besides the standard approach strategy (indirect selection of TRC).

The figures below show all the possible approach blocks for the permitted contour transitions. Two NC blocks NC10 and NC20 are illustrated for the 3 relevant contour transition angles.

13.2.1.1   Direct selection (G138/G41/G42)
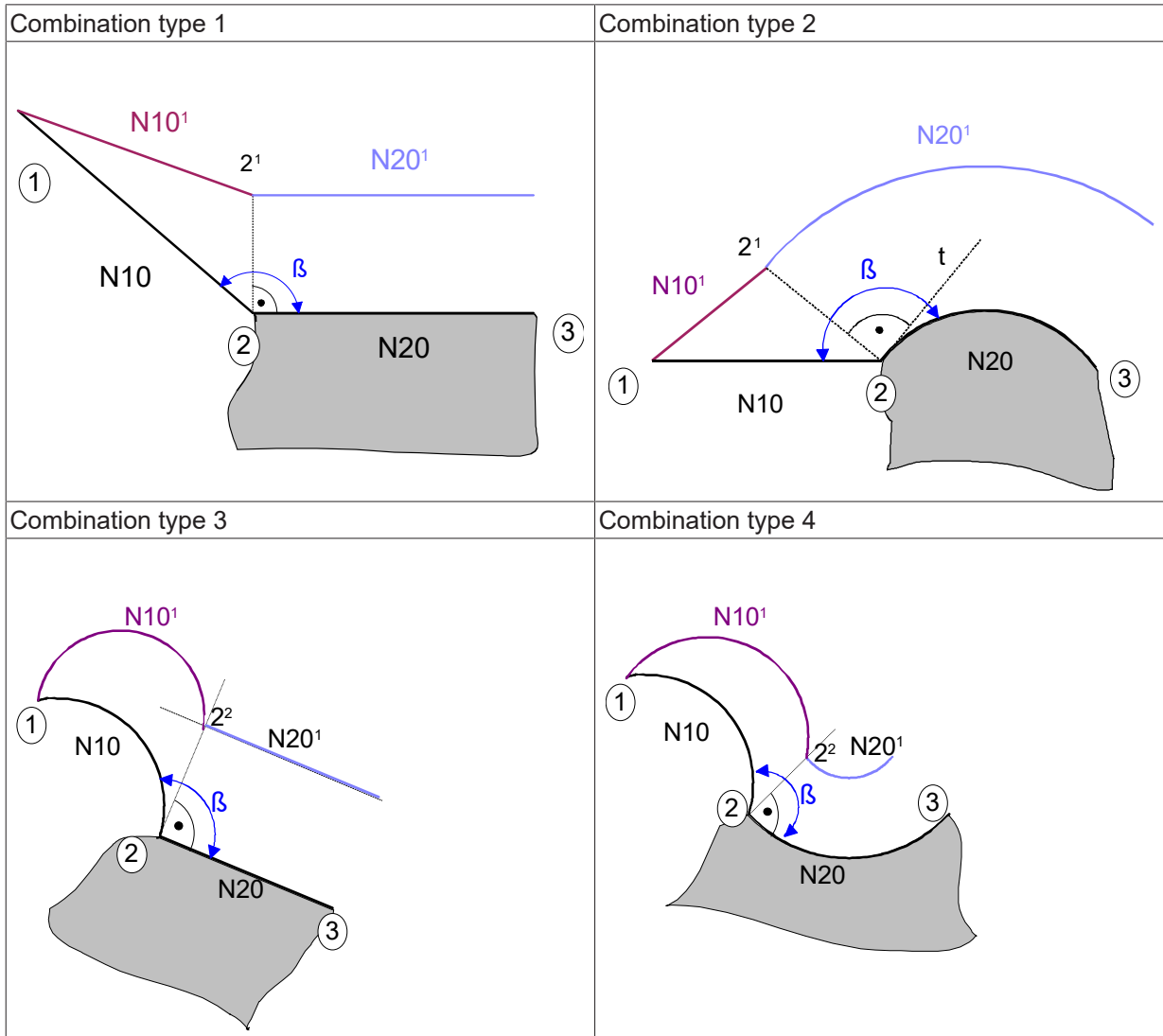
Contour transition range 0° < ß ≤ 180°
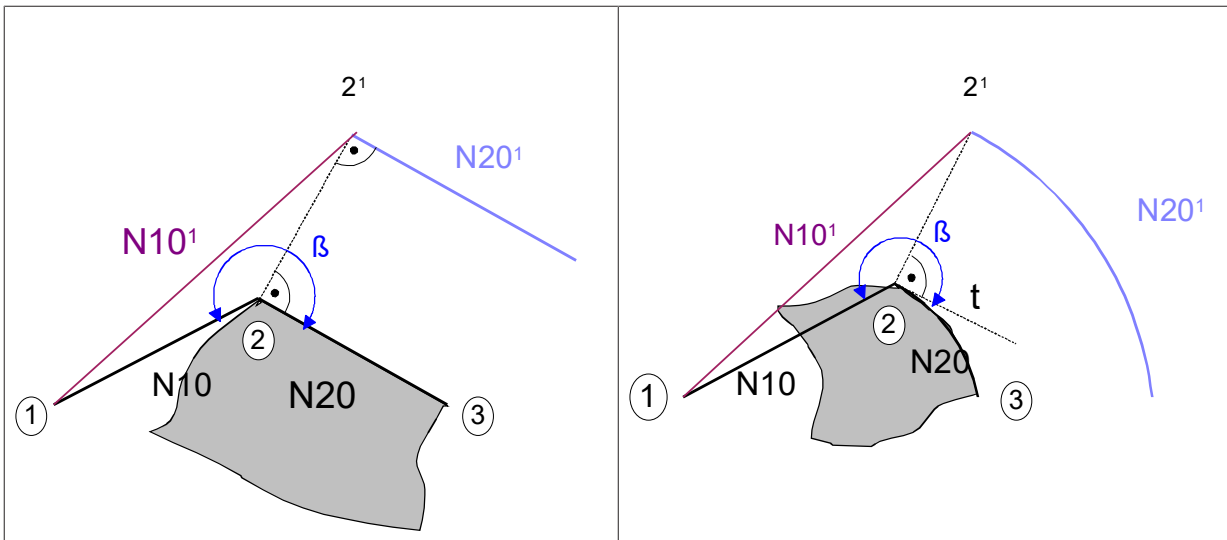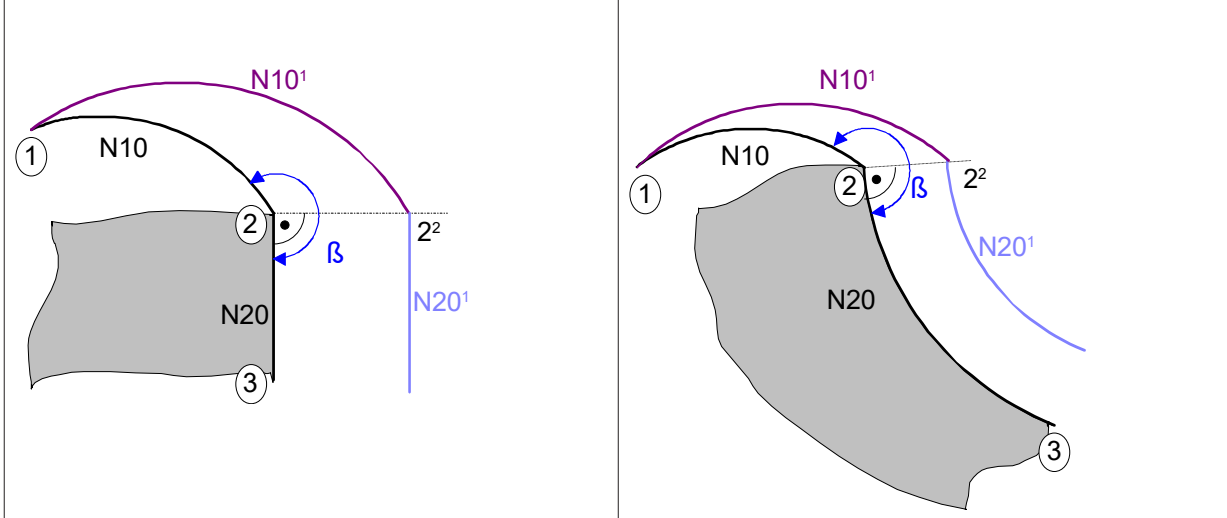
Combination type 1	Combination type 2

| Combination type 3 | Combination type 4 |

Contour transition range 180° < ß ≤ 270°

| Combination type 1 | Combination type 2 |

| Combination type 3 | Combination type 4 |
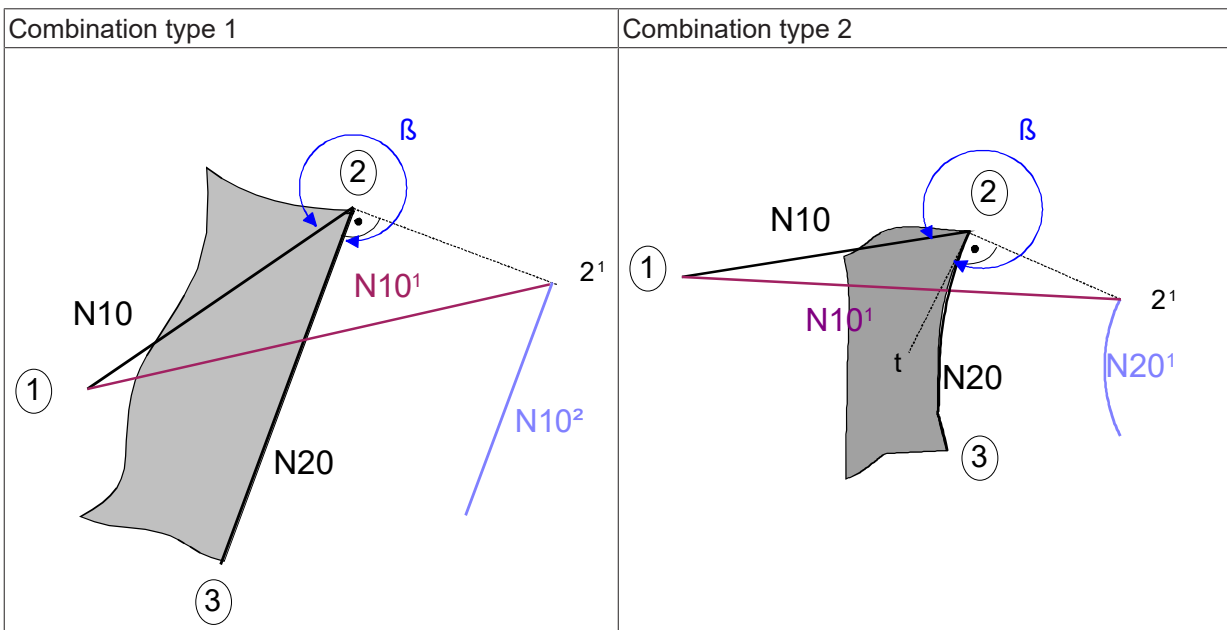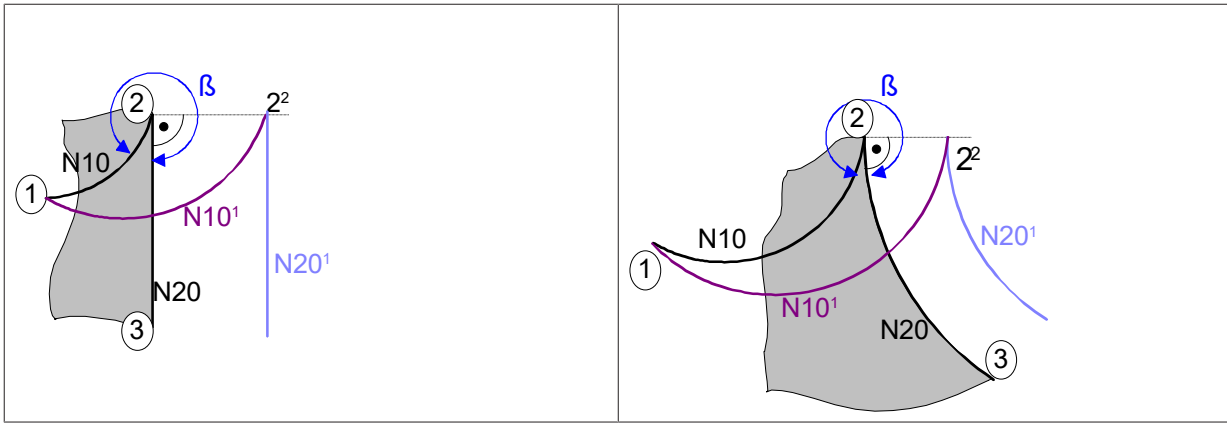
Contour transition range 270° < ß ≤ 360°

13.2.1.2 Indirect selection (G139/G41/G42) with G25

Contour transition range 0° < ß ≤ 180°

Combination type 1	Combination type 2

Combination type 3

Combination type 4
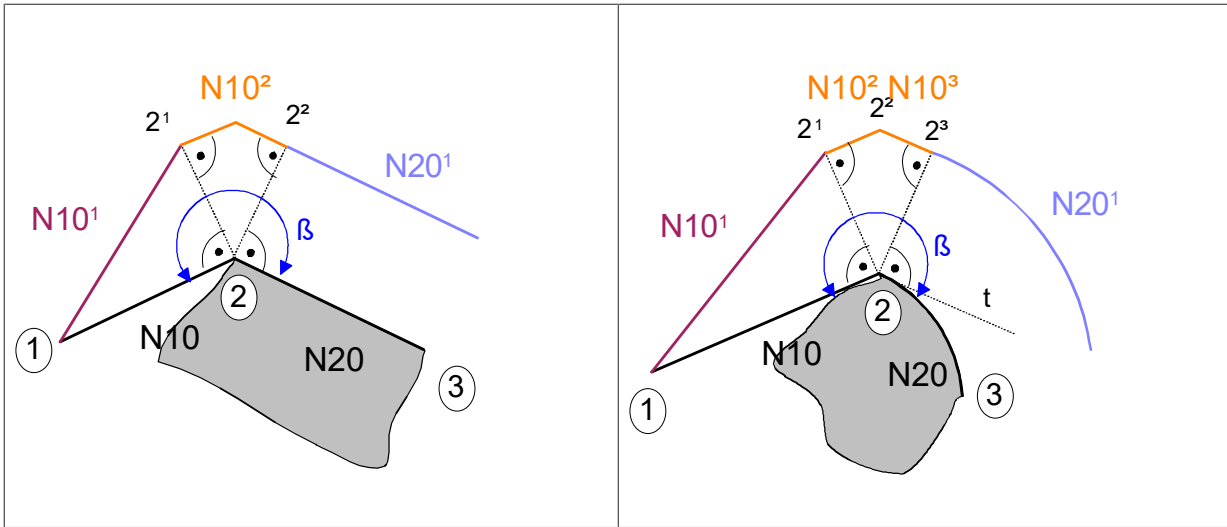
Contour transition range 180° < ß ≤ 270°
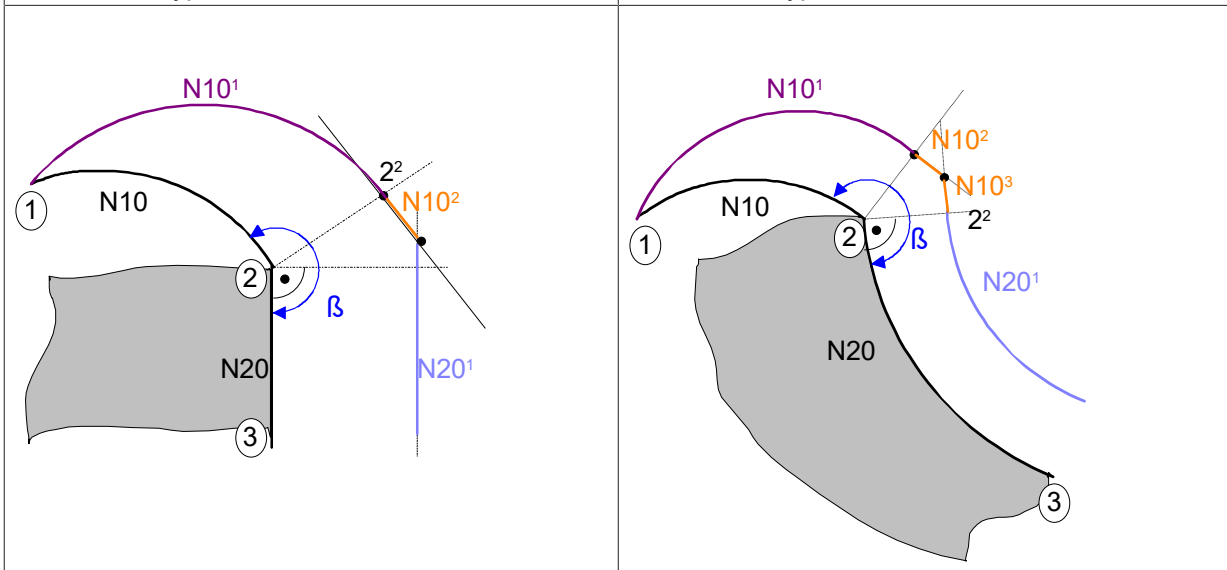
Combination type 1

Combination type 2

Combination type 3

Combination type 4

Contour transition range 270° < ß ≤ 360°

13.2.1.3 Indirect selection (G139/G41/G42) with G26

Combination type 1	Combination type 2
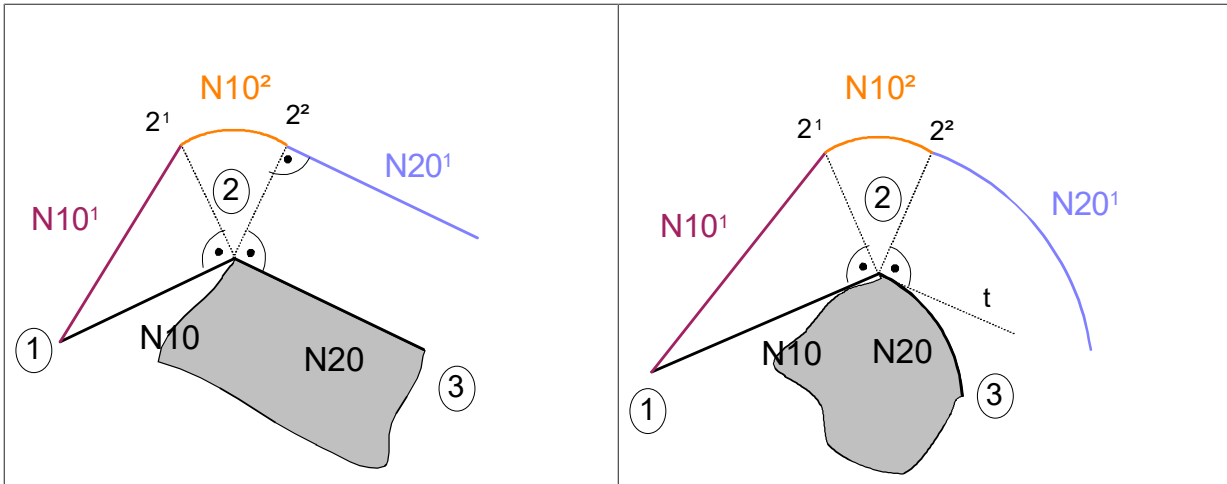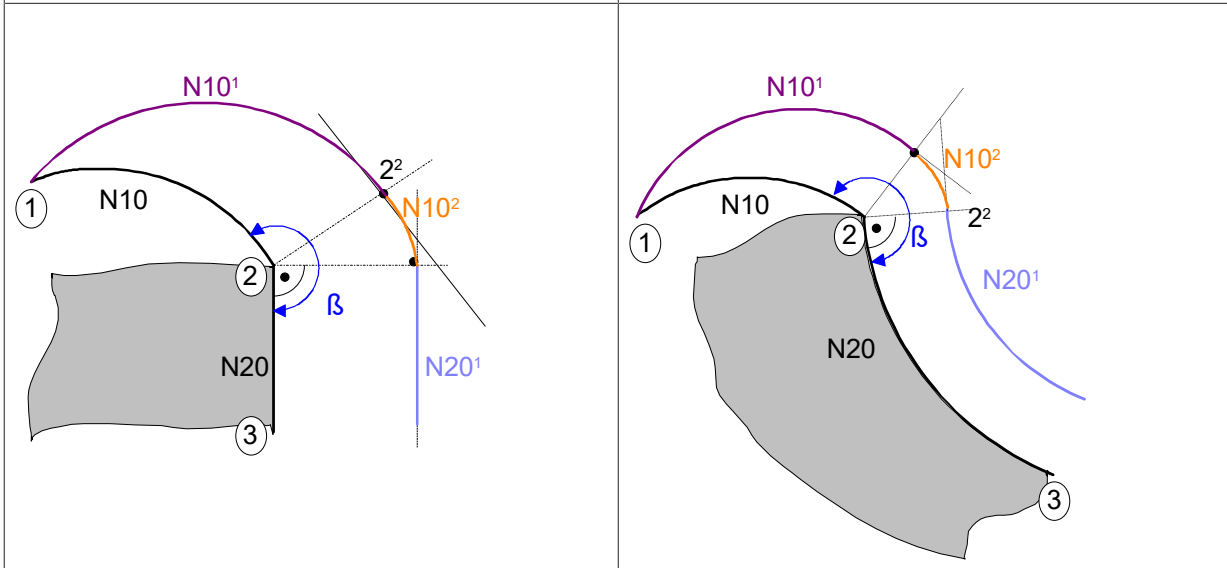
Combination type 3	Combination type 4

Contour transition range 0° < ß ≤ 180°

Combination type 1	Combination type 2

Combination type 3	Combination type 4

Contour transition range 180° < ß ≤ 270°

Combination type 1	Combination type 2

Combination type 3	Combination type 4

BECKHOFF
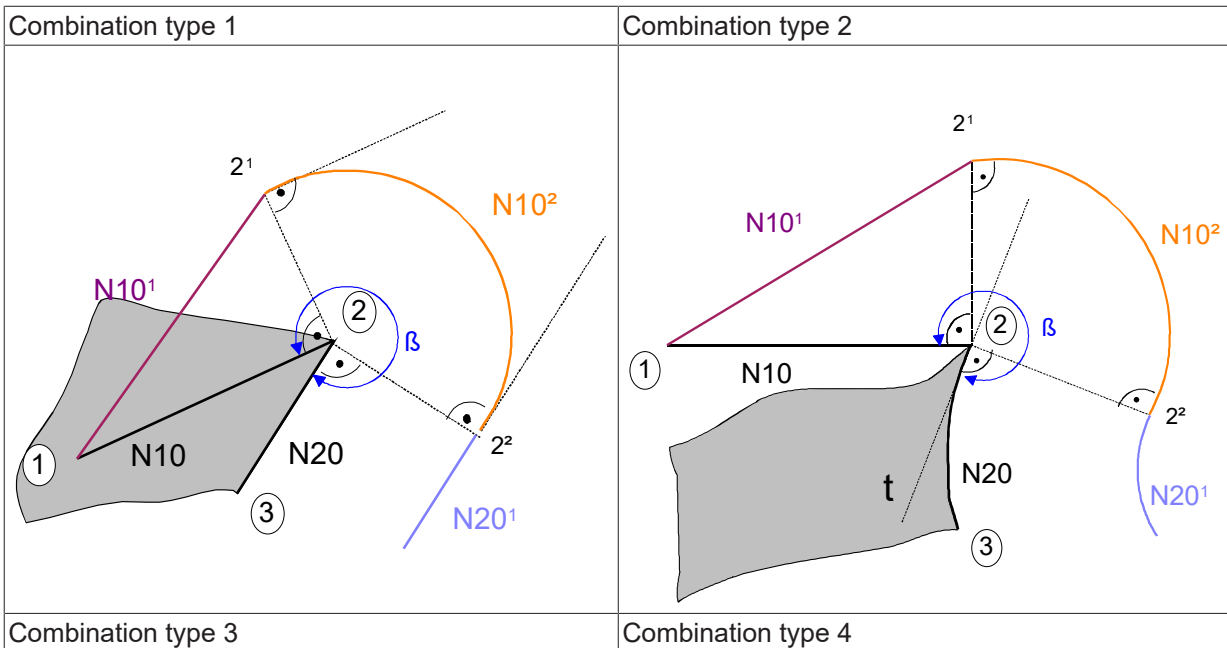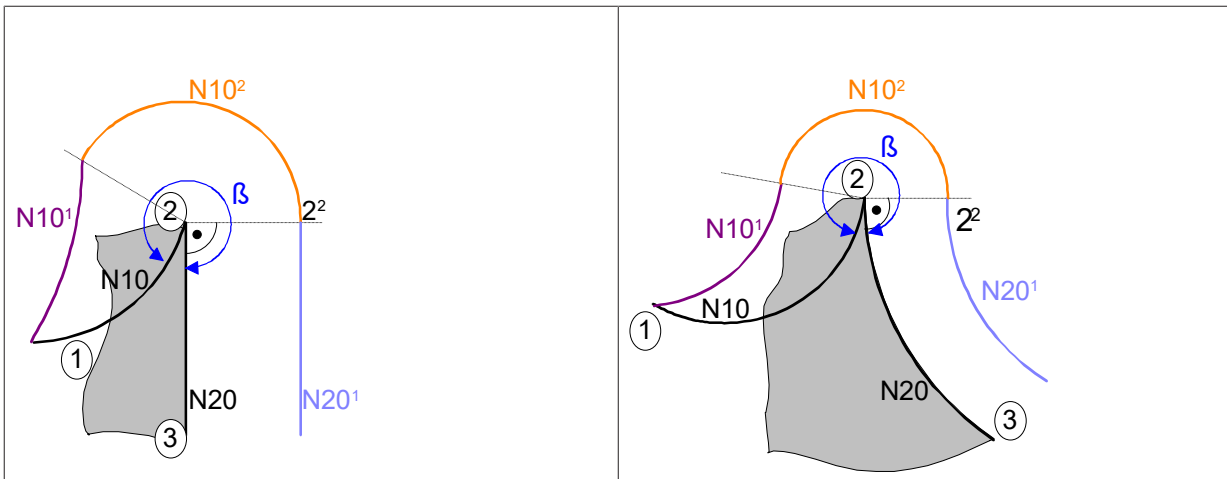
Contour transition range 270° < ß ≤ 360°

13.2.2 Direct/indirect deselection of TRC

When TRC is selected directly or indirectly, an exit block is generated in the selected compensation plane with the next motion block after a programmed G40.

When TRC is deselected, G138 can select an alternative exit strategy (direct deselection of TRC) besides the standard exit strategy (indirect deselection of TRC).

The figures below show all the possible exit blocks with all contour transitions. Two NC blocks NC10 and NC20 are shown for the 3 relevant contour transition angles.

13.2.2.1 Direct deselection (G138/G40)
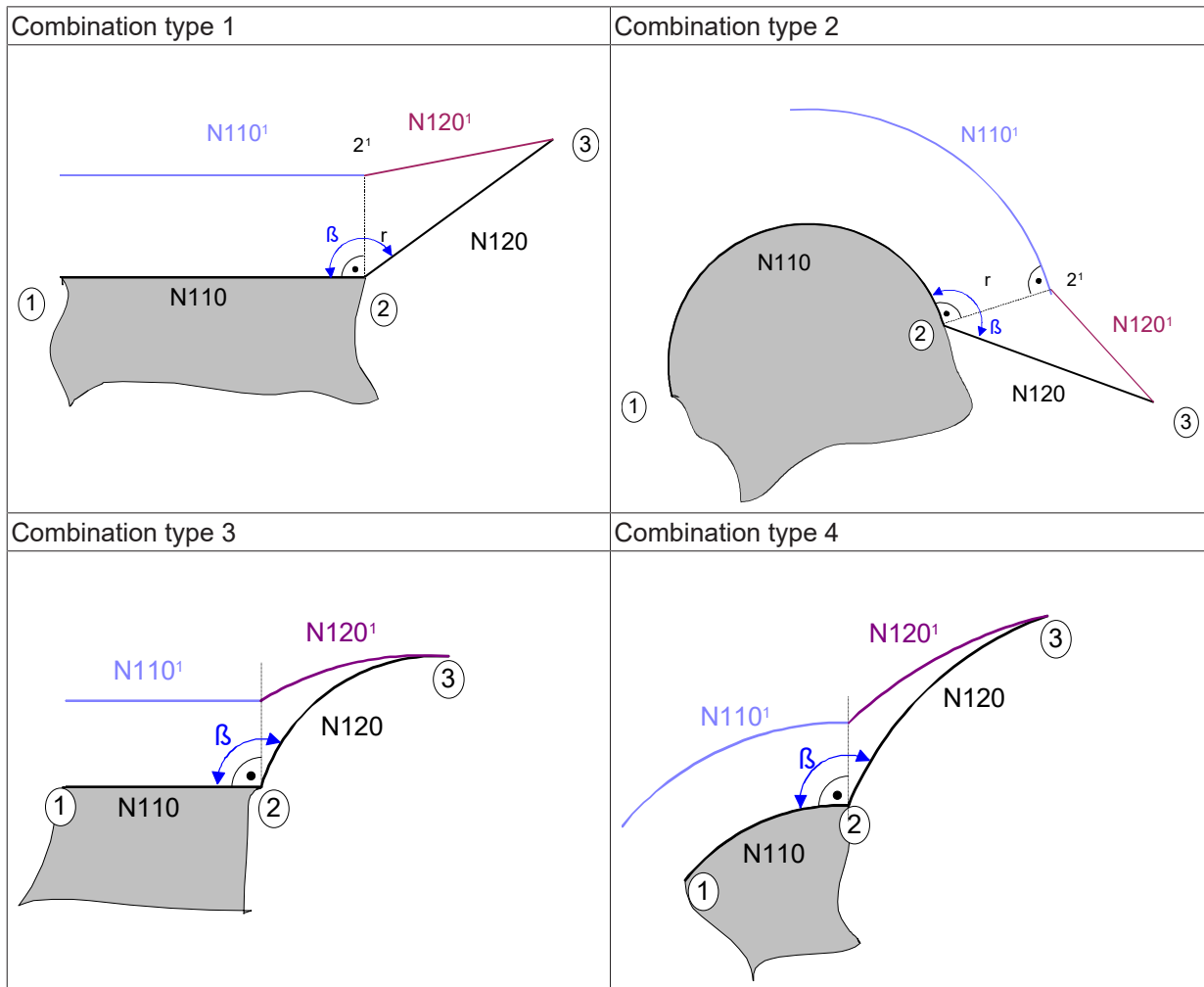
Combination type 1	Combination type 2
Combination type 3	Combination type 4
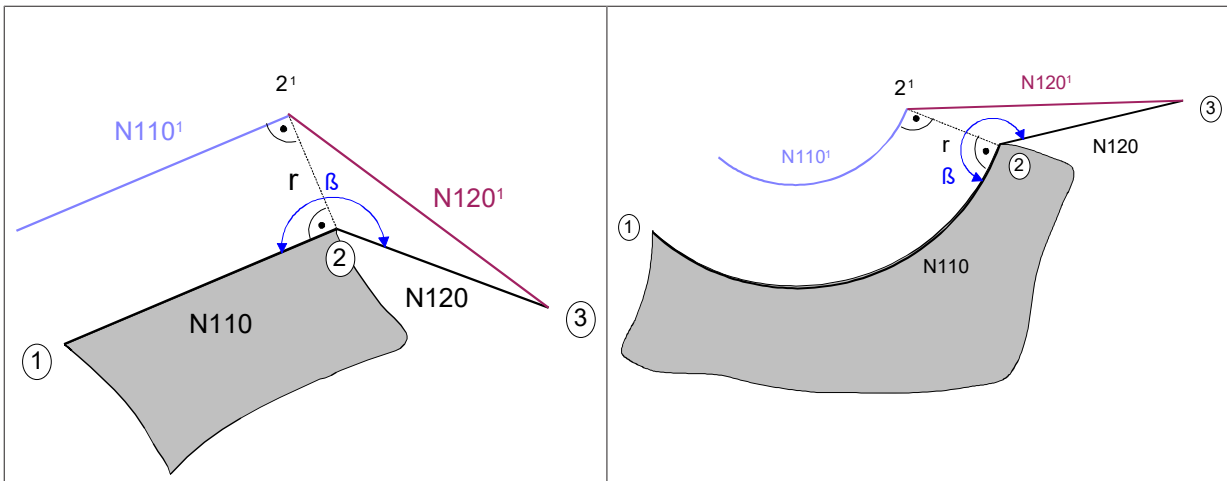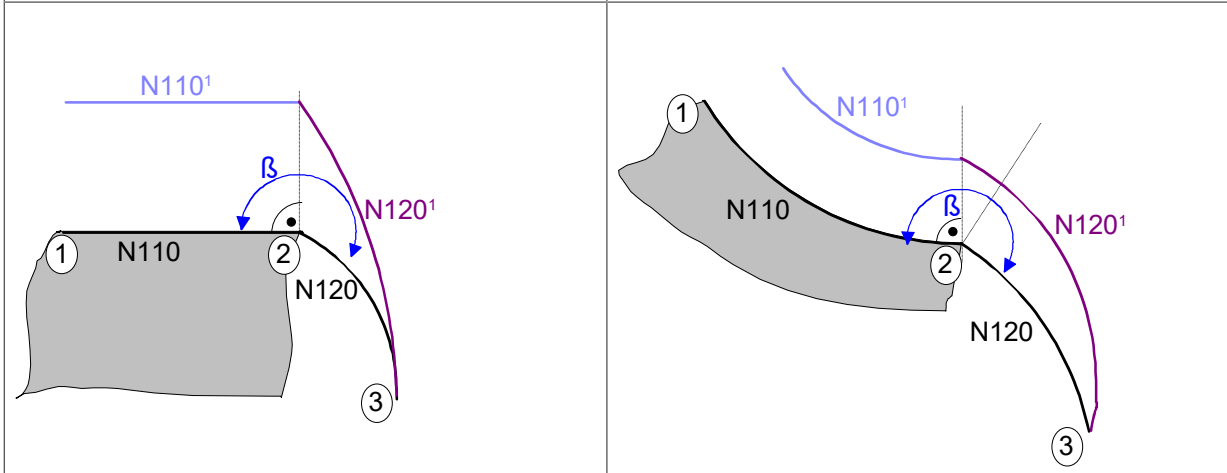
Contour transition range 0° < ß ≤ 180°

Combination type 1	Combination type 2

Combination type 3	Combination type 4

Contour transition range 180° < ß ≤ 270°

Combination type 1	Combination type 2
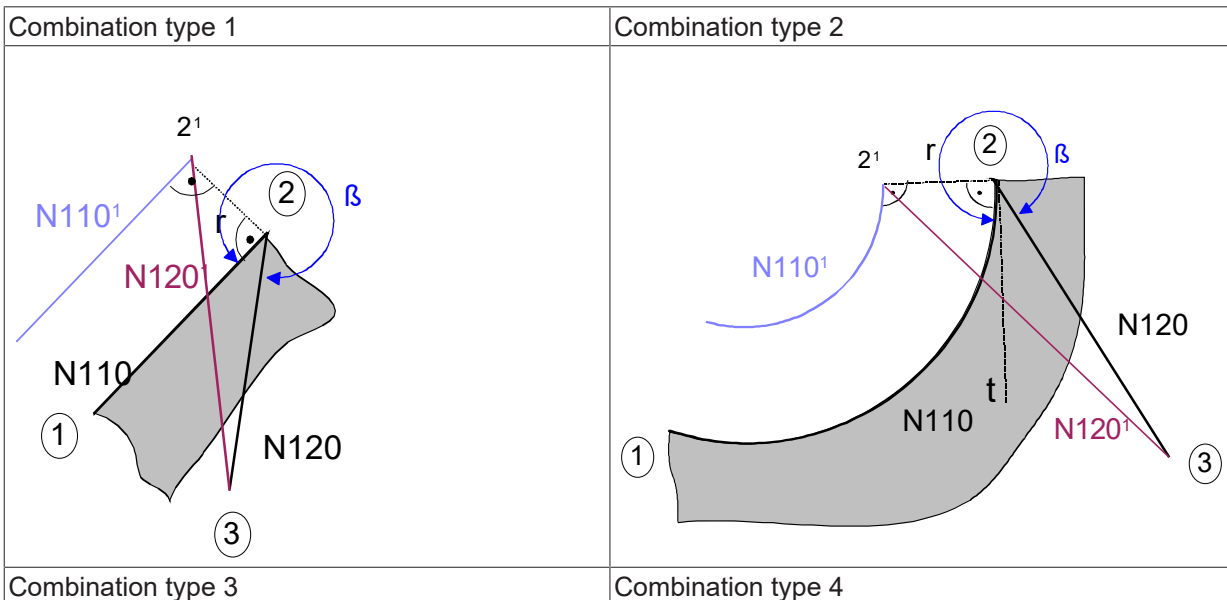
Combination type 3	Combination type 4
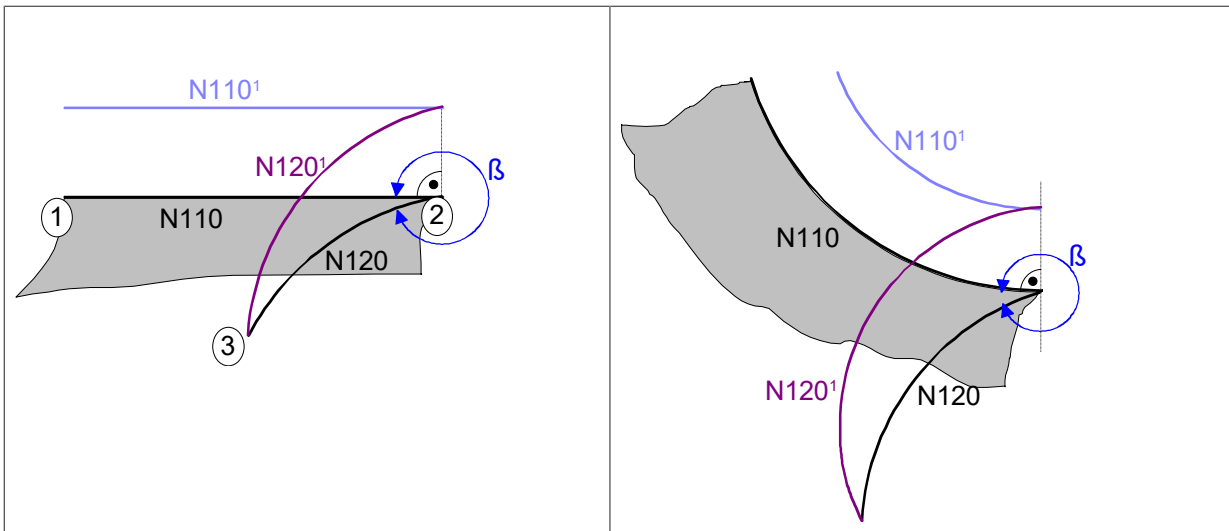
Contour transition range 270° < ß ≤ 360°

13.2.2.2 Indirect deselection (G139/G40) with G25
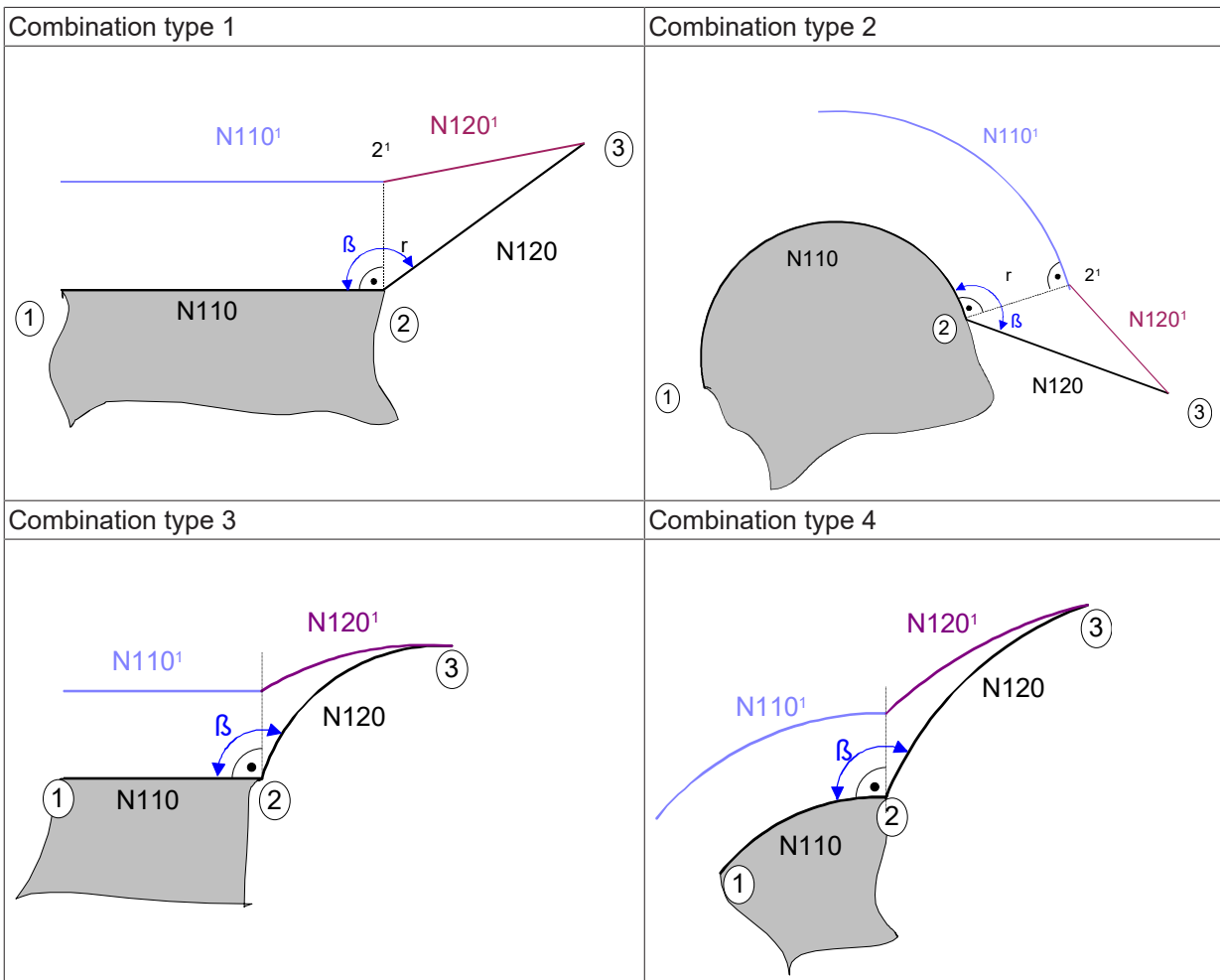
Contour transition range 0° < ß ≤ 180°
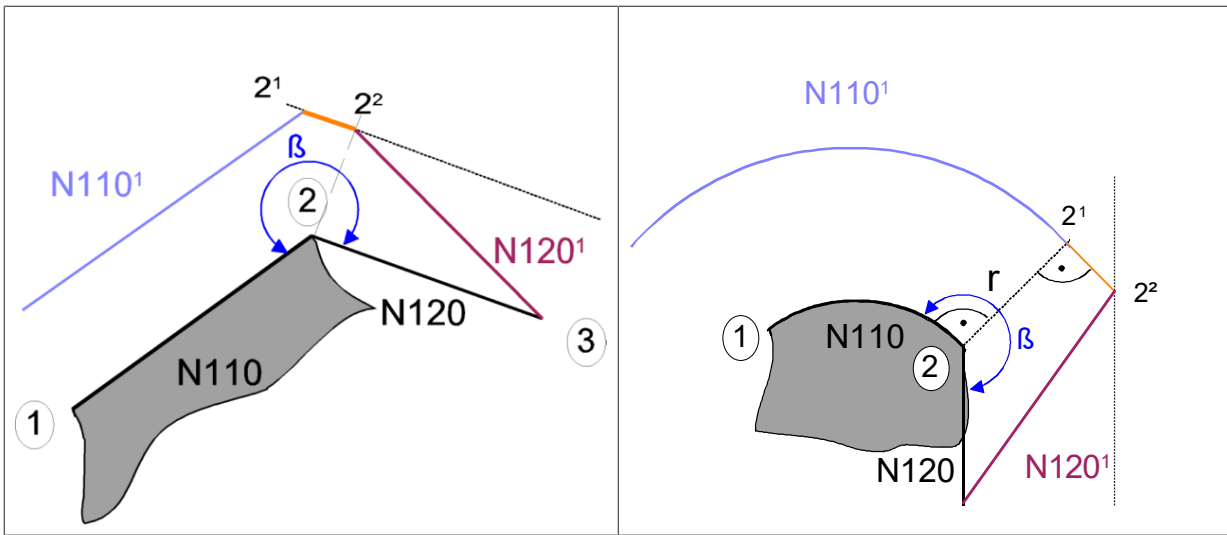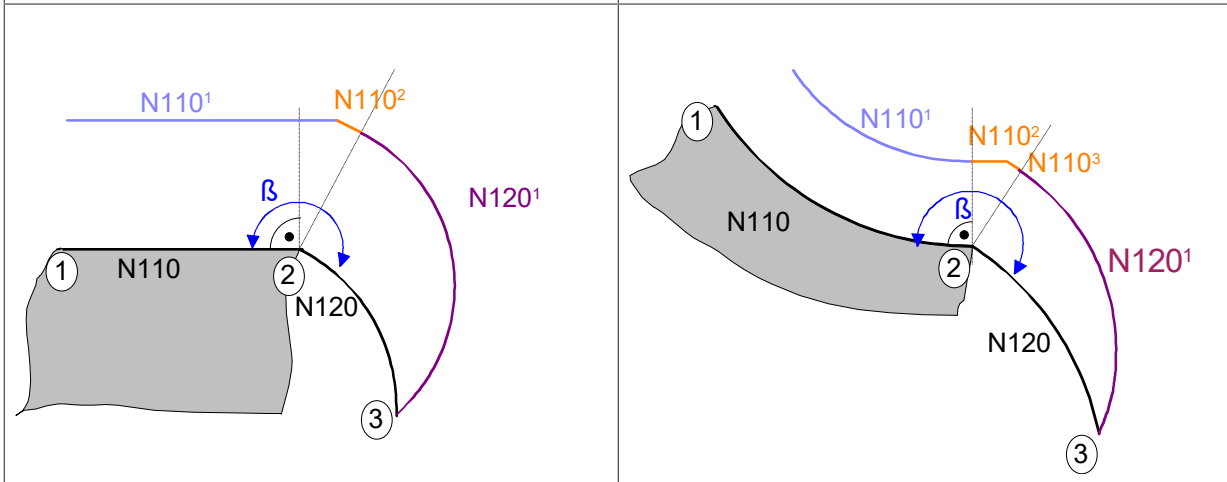
Combination type 1	Combination type 2

BECKHOFF

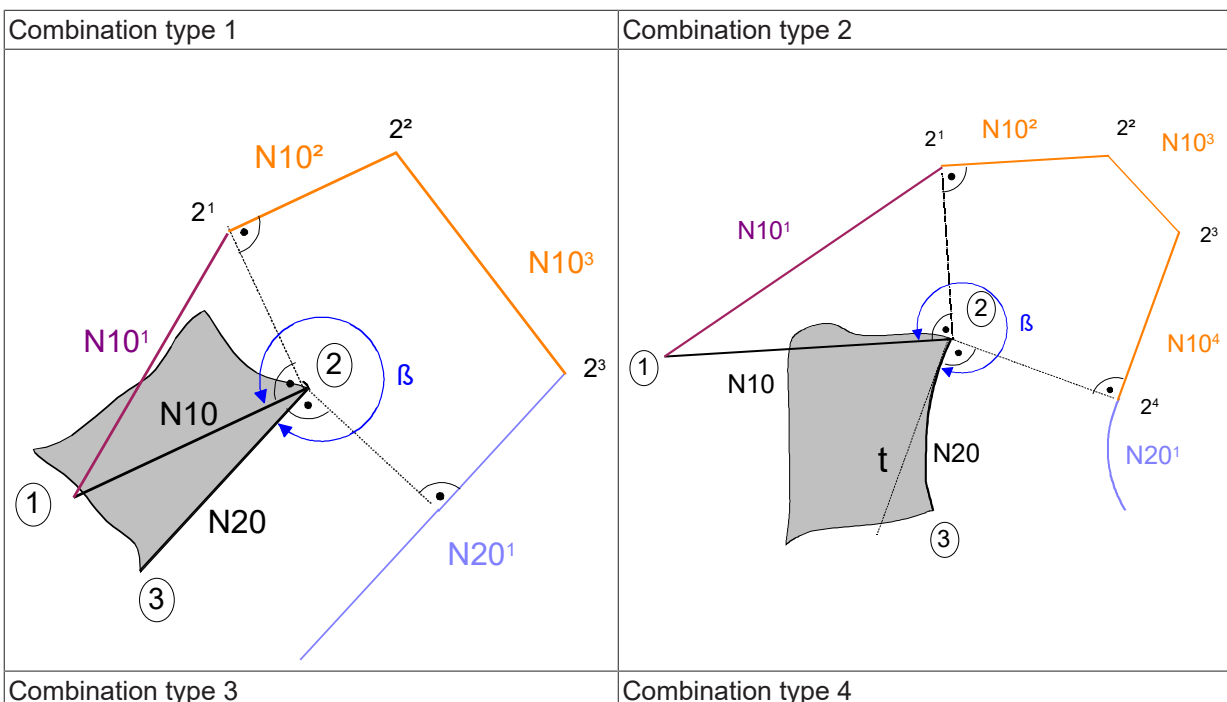Combination type 3

Combination type 4

Combination type 3

Combination type 4

Contour transition range 180° < ß ≤ 270°
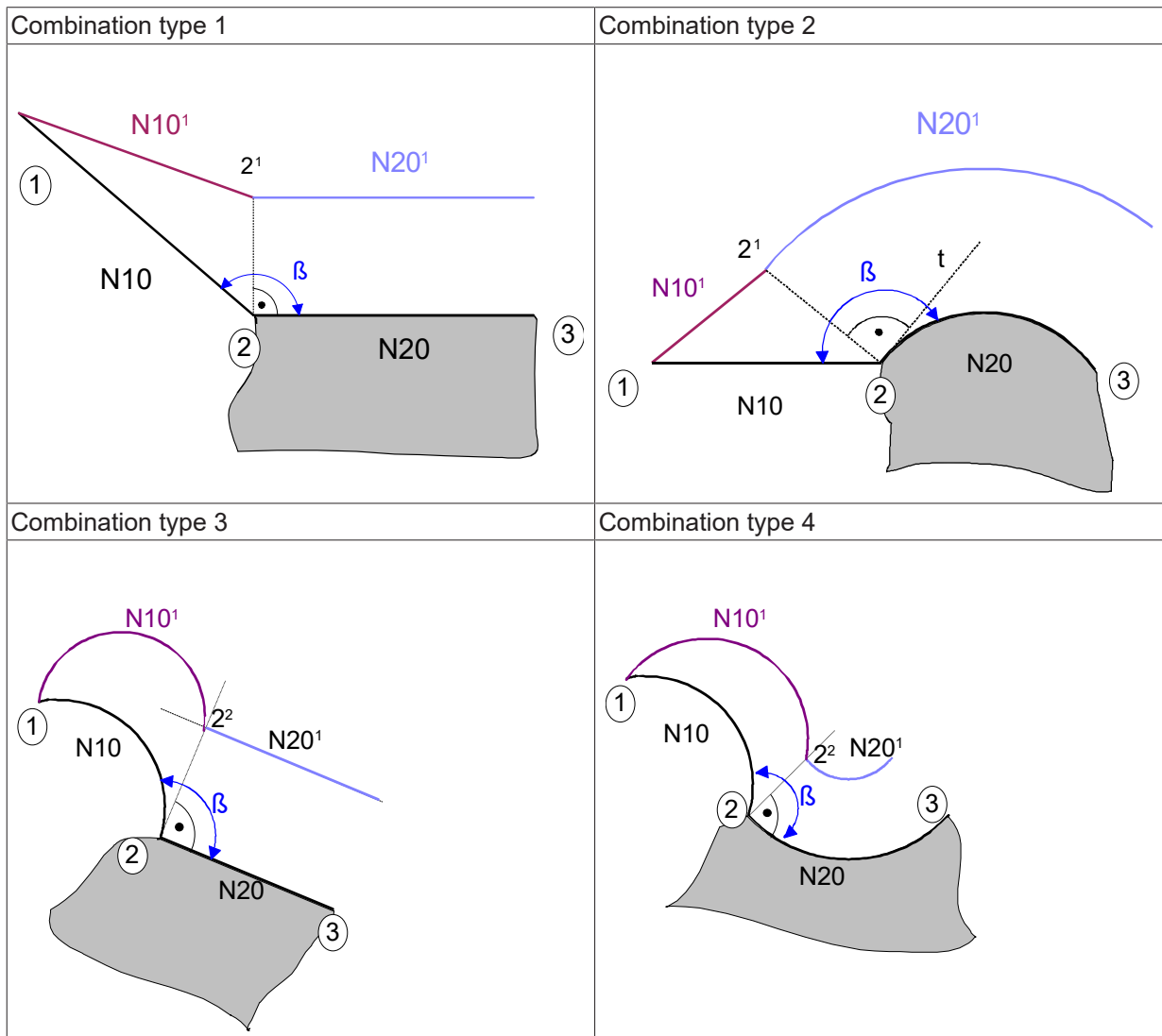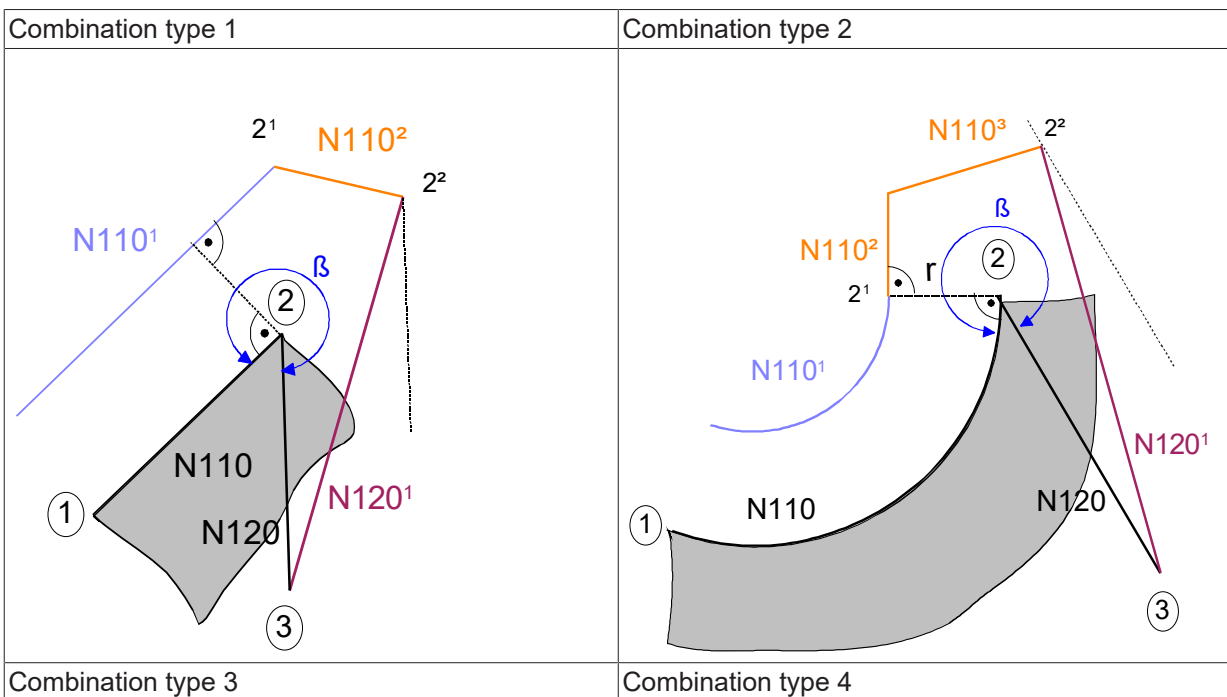
Combination type 1

Combination type 2
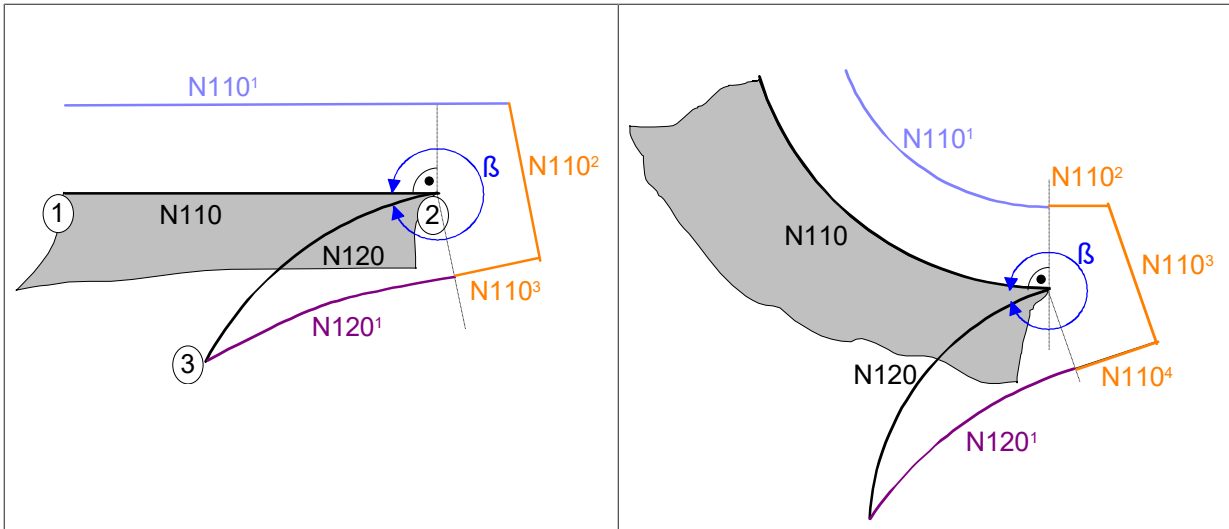
Contour transition range 270° < ß ≤ 360°

13.2.2.3 Indirect deselection (G139/G40) with G26
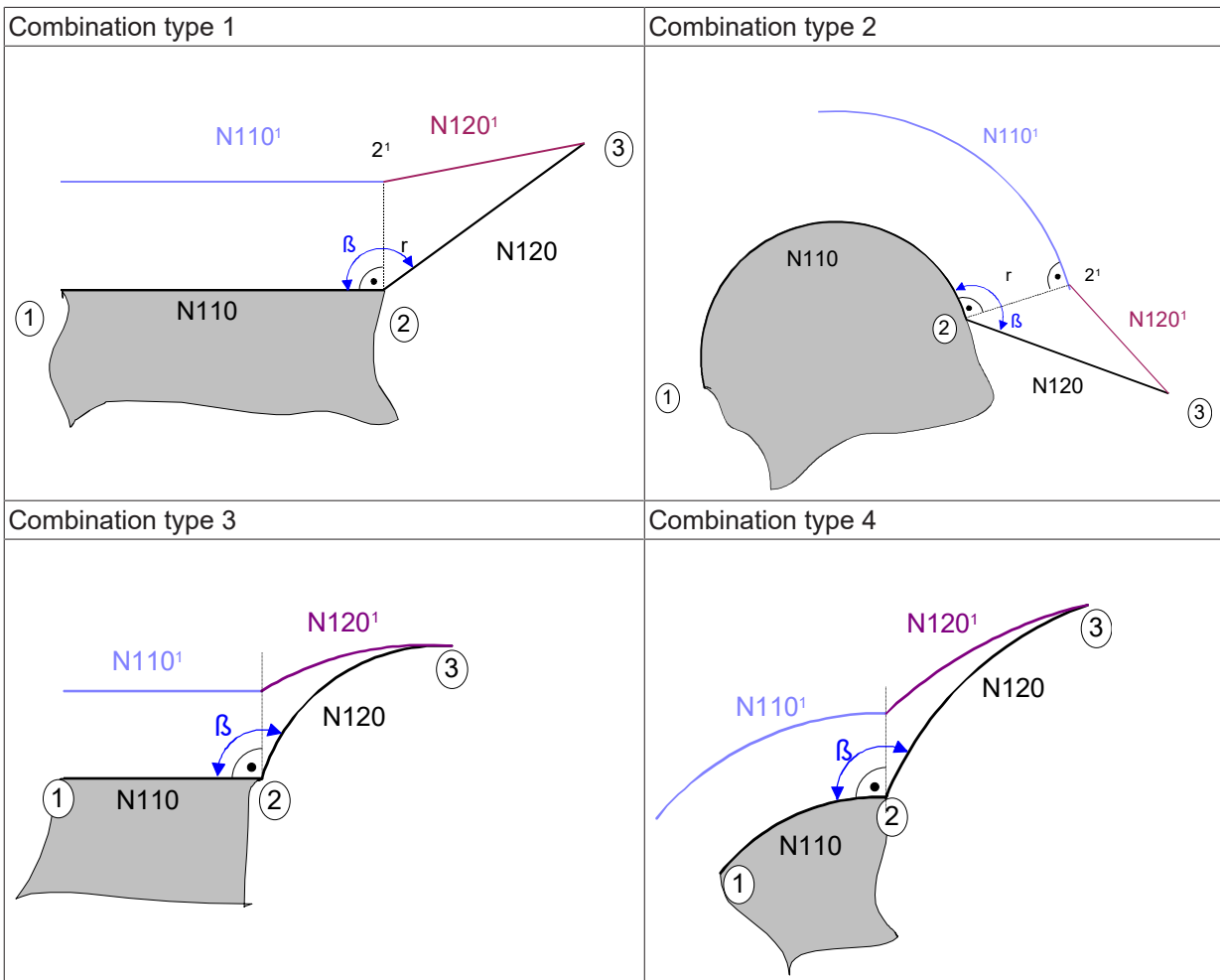
Combination type 1	Combination type 2
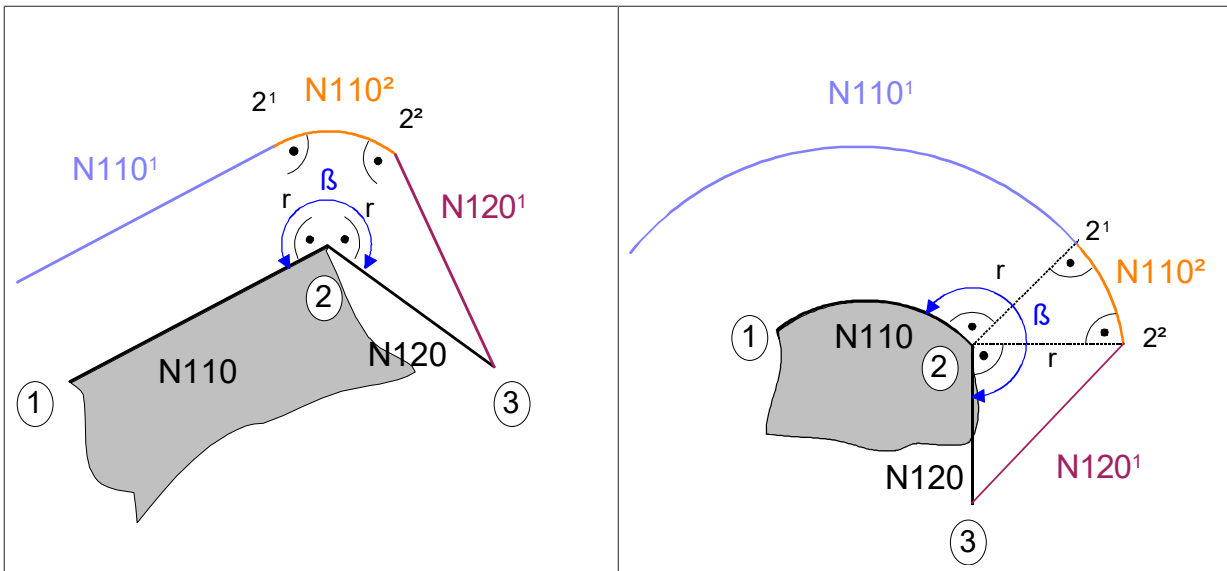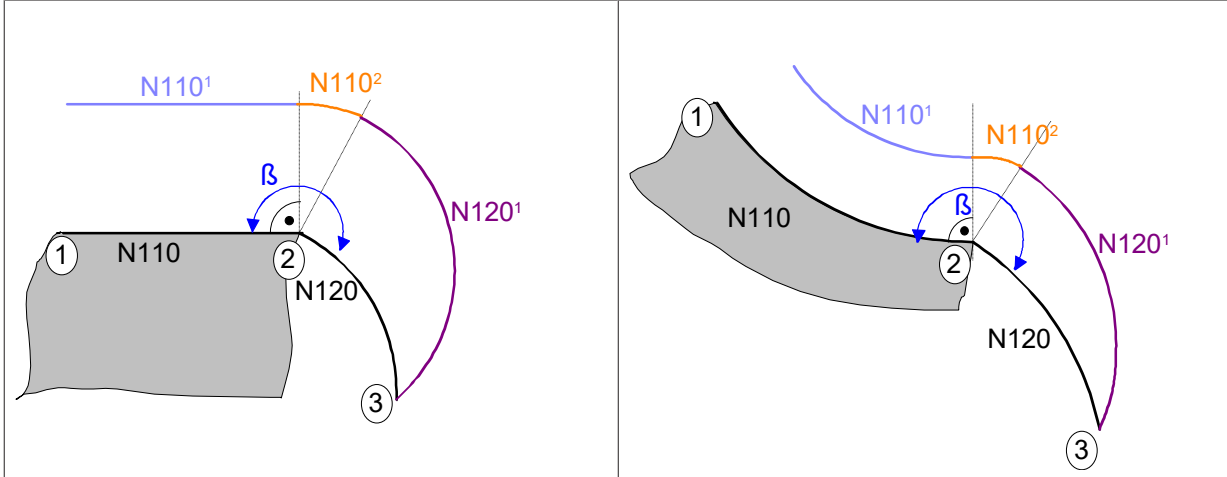
Combination type 3	Combination type 4

Contour transition range 0° < ß ≤ 180°

Combination type 1	Combination type 2

BECKHOFF

Combination type 3

Combination type 4

Combination type 3

Combination type 4

Contour transition range 180° < ß ≤ 270°

Combination type 1

Combination type 2
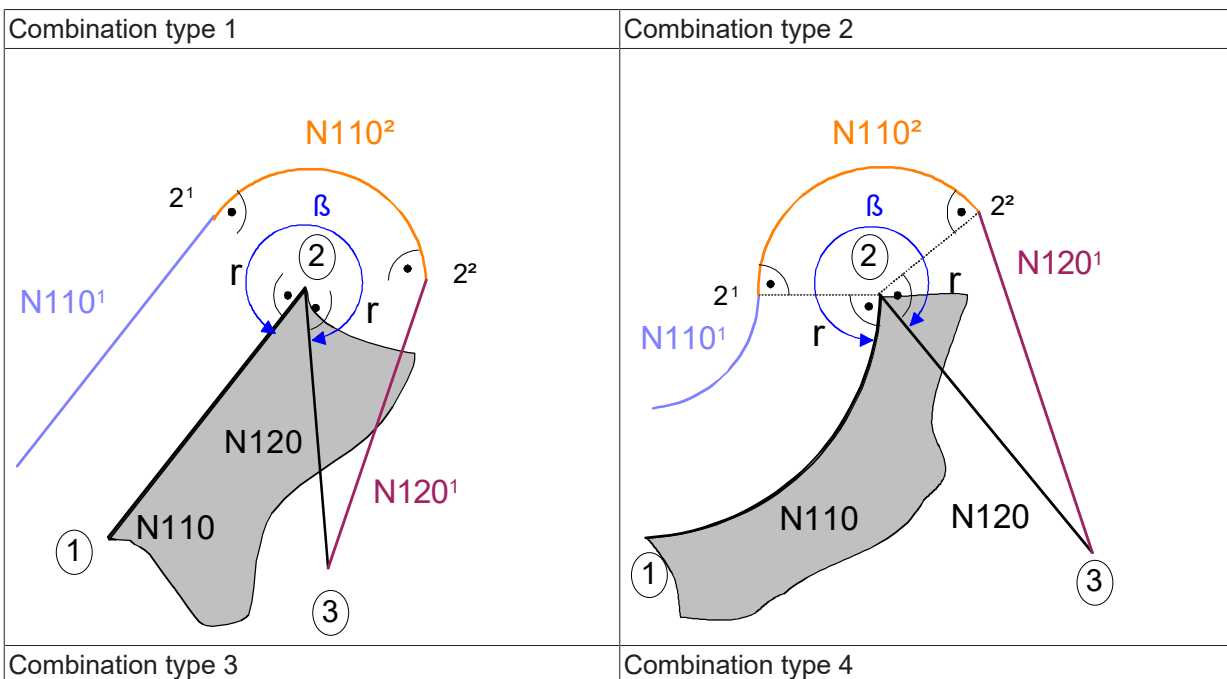
Combination type 3

Combination type 4
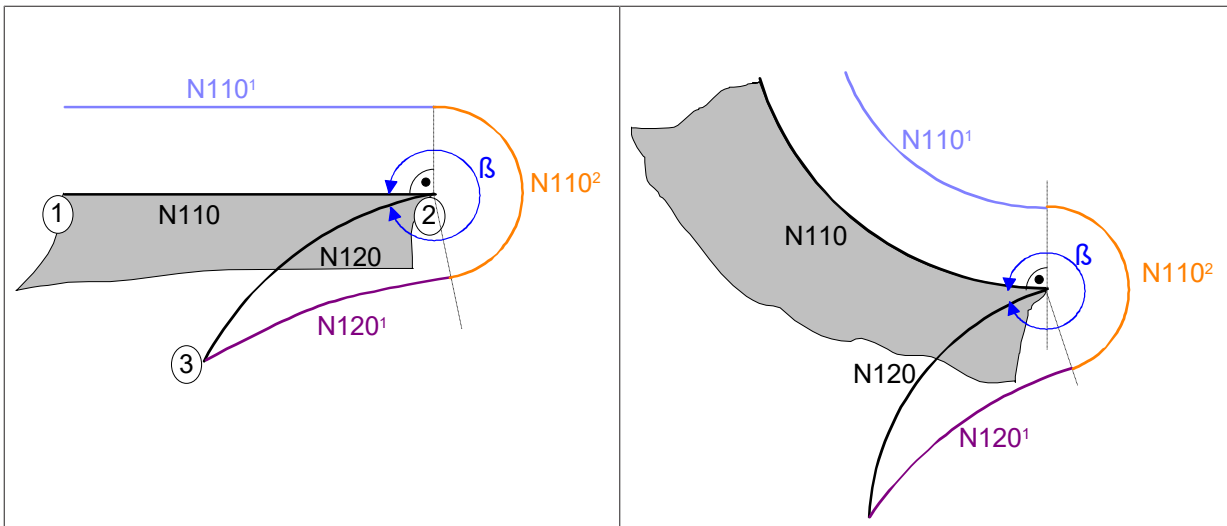
Contour transition range 270° < ß ≤ 360°

13.2.3   Perpendicular selection/deselection of TRC (G237)

If the perpendicular mode of selection and deselection is used, there are no restrictions concerning the sequence of blocks as for the direct or indirect selection and deselection mode.

It is also possible to activate TRC for single blocks.

When perpendicular TRC is selected, a block is added running orthogonally to the programmed path. This block is output before the first motion block and represents the distance (tool radius) of the compensated path relative to the programmed path.

When TRC with the perpendicular mode is deselected, a block is added after the deselection block and runs orthogonally to the programmed path. This motion block reverses the distance generated to the programmed path.

Perpendicular selection and deselection (G237) of TRC

```
%bsp01.nc
N10 G00 X0 Y0 Z0 G17
N20 X10Y10
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237               (perpendicular selection activated)

; Corrected path
N50 G42                (Select TRC right)
N60 G03 X60Y60J50F1000
N70 G01 Y100
N80 G03 X10Y150J50
N90 G40                (Deselect TRC)

; Presentation of the original contour
N100 G00 X0Y0
N110 X10Y10
N120 G03 X60Y60J50F1000
N130 G01 Y100
N140 G03 X10Y150J50
N150 G00 X0Y0
N999 M30
```
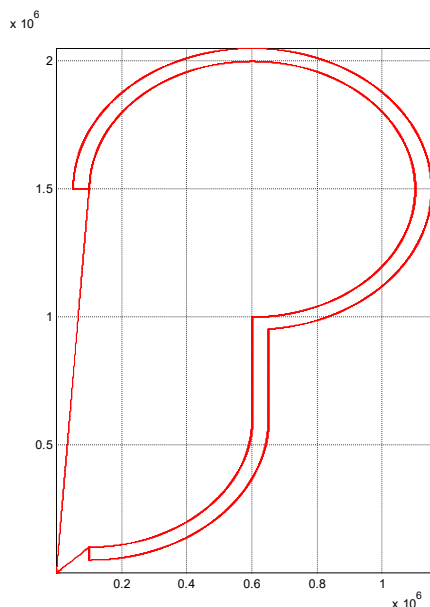

Fig. 128: Contour example with G237

13.2.3.1 Technology functions

The placement of technology functions in the NC program is especially important with perpendicular selection or deselection since this determines the output time.

When perpendicular TRC is selected, the arrangement of selection block, technology function and the first motion block is decisive for the output time.

When perpendicular TRC is deselected the arrangement of technology block and the last compensated motion block is decisive for the output time.

The programming examples below and corresponding figures illustrate this.

Technology function 1

```
%bsp02.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237               (perpendicular selection activated)
N50 G91                (relative programming)
N60 G01 X30 Y10

; Corrected path
N50 G41 M7 X20 Y70     (Select TRC left)
N55 M8
N60 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N85 M9
N90 G40                (Deselect TRC)
N100 G90 X200 Y0       (absolute programming)
N110 X0

; Presentation of the original contour
N200 G91 (relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0 (absolute programming)
N270 G00 X0
N999 M30
```

The technology function M9 in the figure below is output immediately before before the output of the perpendicular deselection block. The technology function must be placed between the last block to be compensated and the perpendicular deselection of TRC.
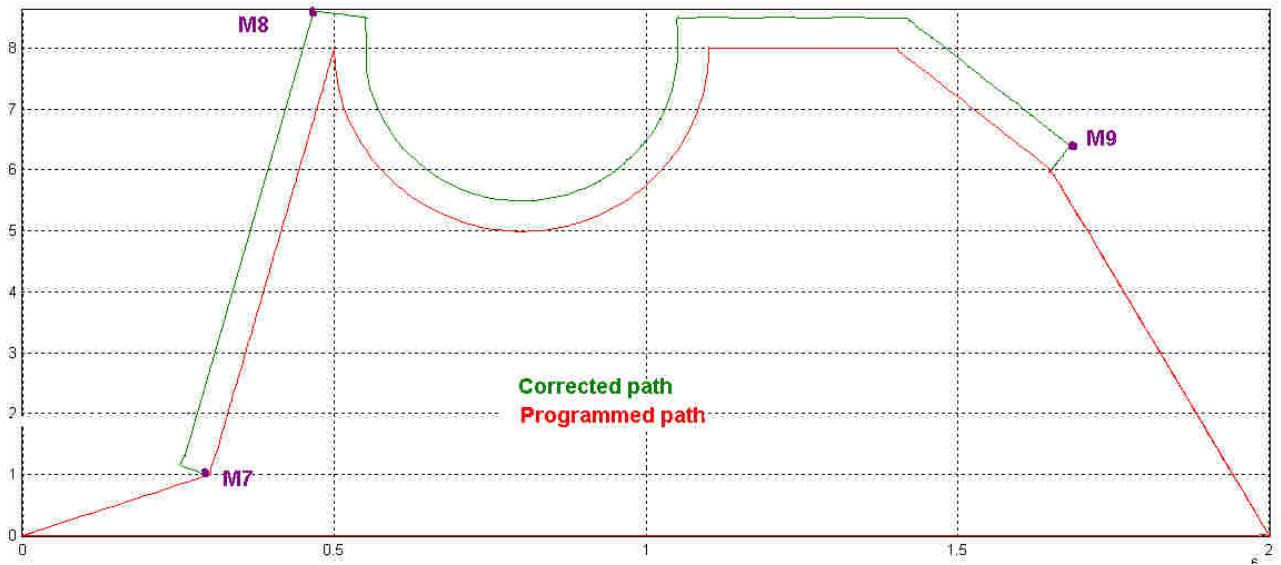
Fig. 129: Contour example with technology function 1

Technology function 2

```
%bsp03.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5     (Tool radius)
N40 G237               (perpendicular selection activated)
N50 G91                 (relative programming)
N60 G01 X30 Y10
; Corrected path
N50 G41 M7             (Select TRC left)
N55 M8 X20 Y70
N60 M9 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N90 G40                (Deselect TRC)
N100 G90 X200 Y0 (absolute programming)
N110 X0
; Presentation of the original contour
N200 G91 (relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0 (absolute programming)
N270 G00 X0
N999 M30
```

The technology function M8 is executed after the selection block generated. For this placement, it is absolutely necessary that it is not placed in the same block as TRC selection and that there is no motion block between the selection block and the technology function.
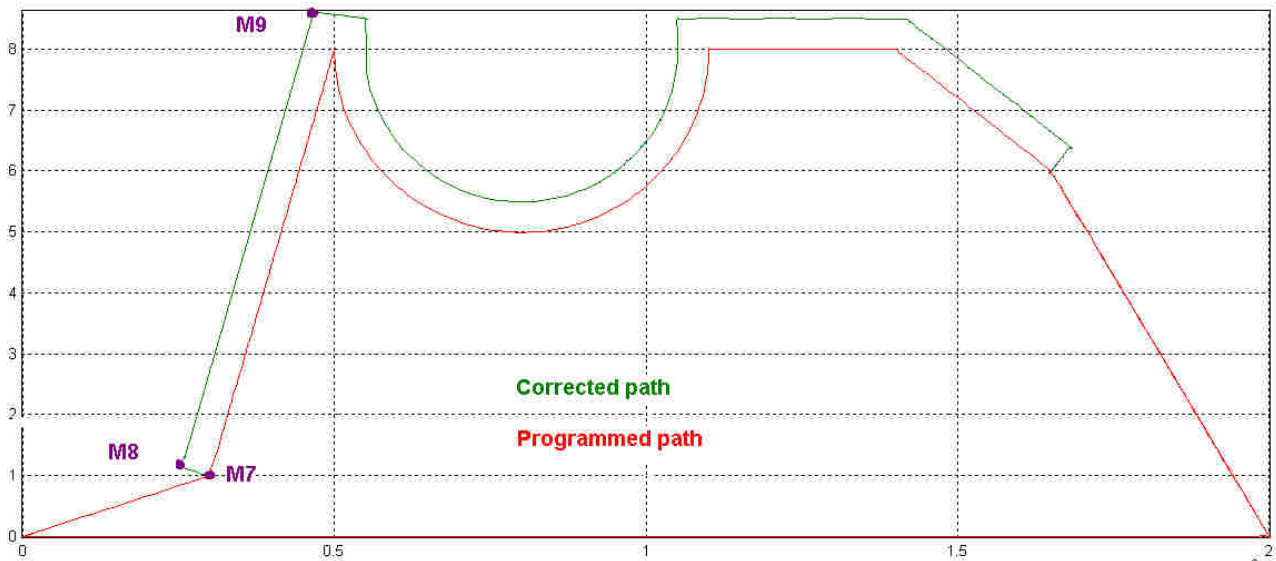
Fig. 130: Contour example with technology function 2

Technology function 3

```
%bsp04.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5       (Tool radius)
N40 G237                 (perpendicular selection activated)
N50 G91                  (relative programming)
N60 G01 X30 Y10
; Corrected path
N50 G41 (Select TRC left)
N51 M7
N52 M8
N53 M8
N55 X20 Y70
N60 G03 X60 I30
```

BECKHOFF

```
N70 G01 X30
N80 X25 Y-20
N90 G40                 (Deselect TRC)
N100 G90 X200 Y0        (absolute programming)
N110 X0
; Presentation of the original contour
N200 G91                    (relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0        (absolute programming)
N270 G00 X0
N999 M30
```
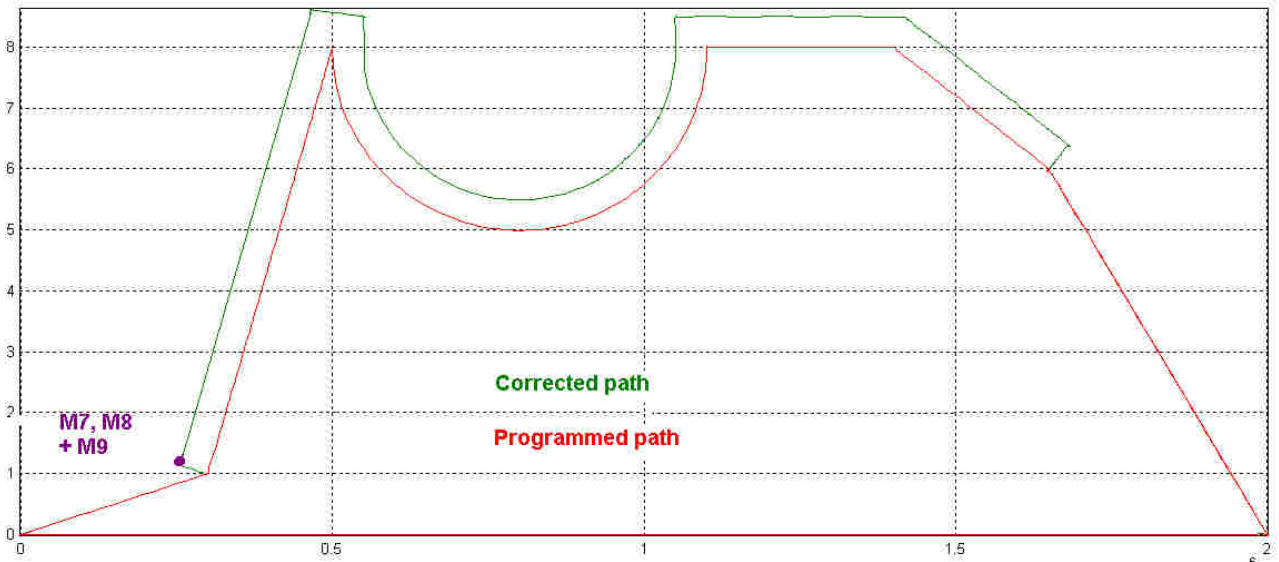

Fig. 131: Contour example with technology function 3

13.2.3.2   Technology function in single block

Technology function in single block

```
%bsp05.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237               (perpendicular selection activated)
N60 G01 X10 Y10
; Corrected path
N50 G41 M7             (Select TRC left)
N55 M8 X70 Y30
N60 M7
N90 G40                (Deselect TRC)
N100 G90 X100 Y0
N110 X0
; Presentation of the original contour
N210 G01 X10 Y10
N220 X70 Y30
N240 X100 Y0
N270 G00 X0
N999 M30
```
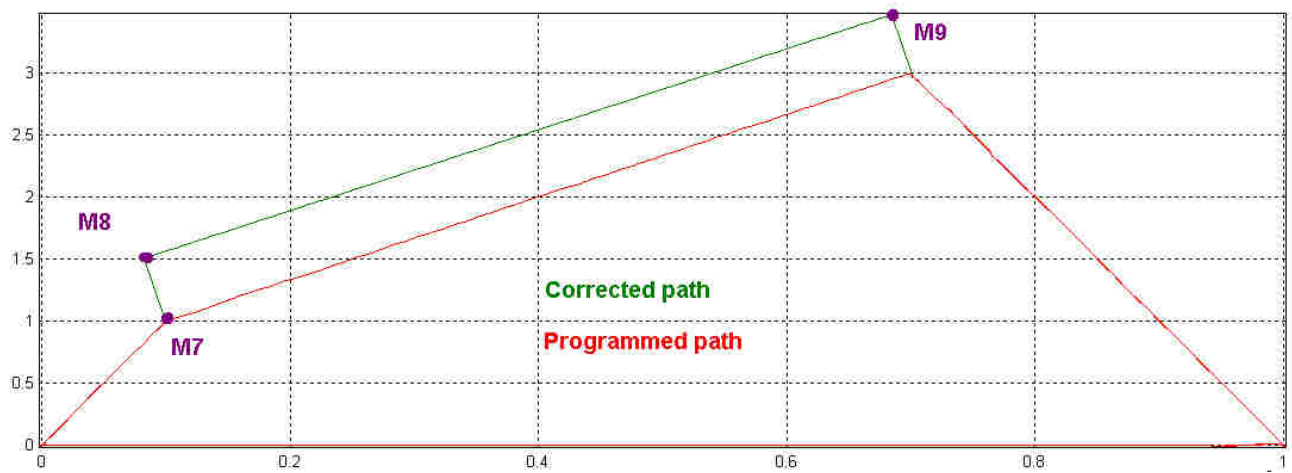

Fig. 132: Contour example with technology function in single block

13.2.4 Selecting inside corner of TRC (G238)

Inside corner selection means that the TRC should be selected at an inside corner of a closed contour.

The designations below are used in connection with inside corner selection:

- **Approach block:**: linear motion block which starts at the point of selection
- **1st motion block**: Motion block that starts at the end of the approach block. It is also the 2nd motion block after selection of TRC with motion data
- **last motion block:** last motion block before deselection of TRC with motion data

Inside corner selection (G238) of TRC

In star-shaped contours the TRC should be selected in a pointed inside corner.

```
%musterstern.nc
N1 G74 X1Y2Z3
N2 G17 G00 X0Y0Z0 G90
N4 F10000

(Display of contour)
N100 G01 X0 Y100
N110 X-20 Y20
N120 X-100 Y0
N130 X-20 Y-20
N140 X0 Y-100
N150 X20 Y-20
N160 X100 Y0
N170 X20 Y20
N180 G01 X0 Y100

N200 G00 X0Y0
(Path display)
N210 G238 (Inside corner selection activated)
N220 V.G.WZR = 10 (Define tool radius)
N230 G41     (Select TRC left of contour)

N240 G01 X0 Y100    (Approach block)

N310 X-20 Y20
N320 X-100 Y0
N330 X-20 Y-20
N340 X0 Y-100
N350 X20 Y-20
N360 X100 Y0
N370 X20 Y20
N380 G01 X0 Y100

N390 G40 (Deselect TRC)
N400 G00 X0Y0
N999 M30
```
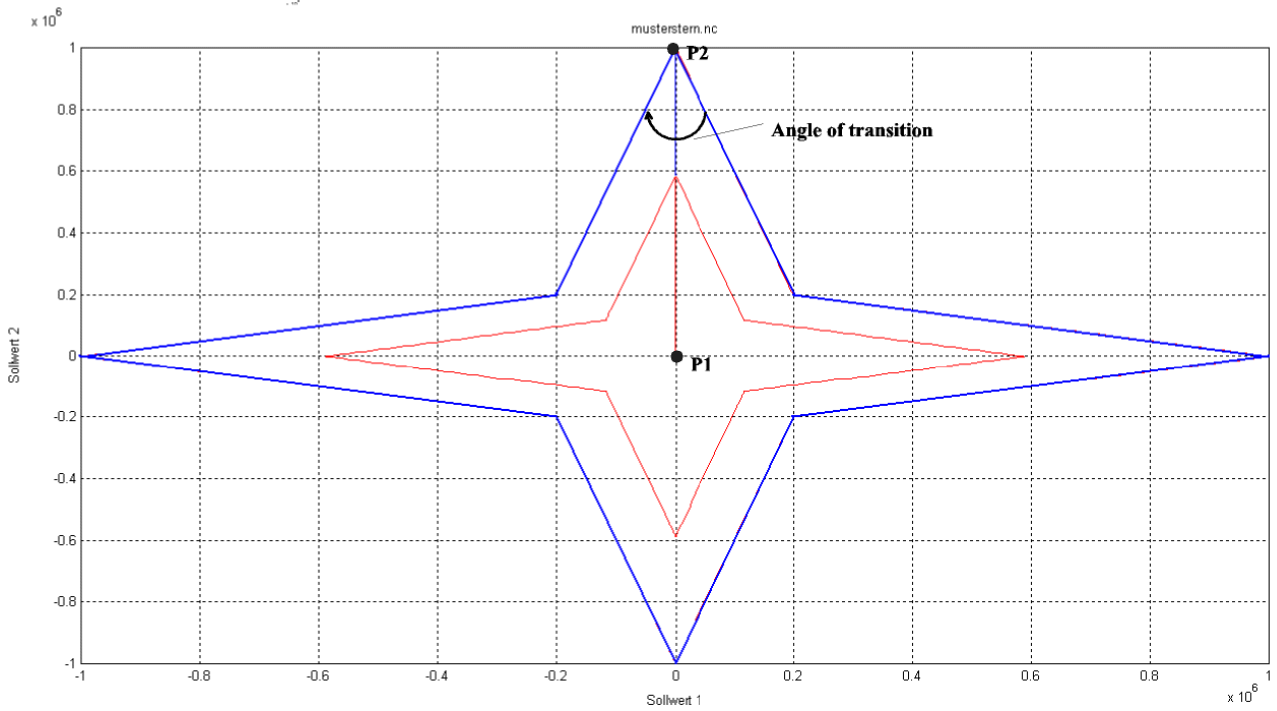
Fig. 133: Contour example for inside corner selection (G238)

Note:

The selection command G41 takes place at point P1; point P2 is the end point of the first linear block after G42 and must simultaneously be the target point of the last motion block before G40.

This outer (or blue) line marks the programmed contour; the inner (or red) line shows the tool path.

The closed contour means that the end point of the last motion block before TRC is deselected must be identical with the end point of the linear block after selection.

There are no restrictions for block transition between the last and the first contour motion block, neither for the transition angle nor with block combinations.

> **i** After the programmed G40 to deselect TRC, contour violations are not monitored in the subsequent motion block.

13.2.4.1   Restrictions of inside corner selection

- The programmed contour must be closed.
- The last motion block may not be a full circle.
- The first motion block must be a linear block; this also applies to the first motion block after TRC is deselected.
- The point P1 must at least have the distance of the tool radius to all contour elements.
- The first motion block may not cross a contour element, with the exception of the first and the last motion blocks.
- It is not permitted to change the selection side in selected state.
- It is not permitted to make changes to the tool radius of tool in selected state.
- The selection side of the TRC must be selected from the position of selection soy that the tool does not cross the programmed contour.
- The enclosed angle of the first and the second motion blocks after selection may not exceed 180°.

Causes of angle restriction:
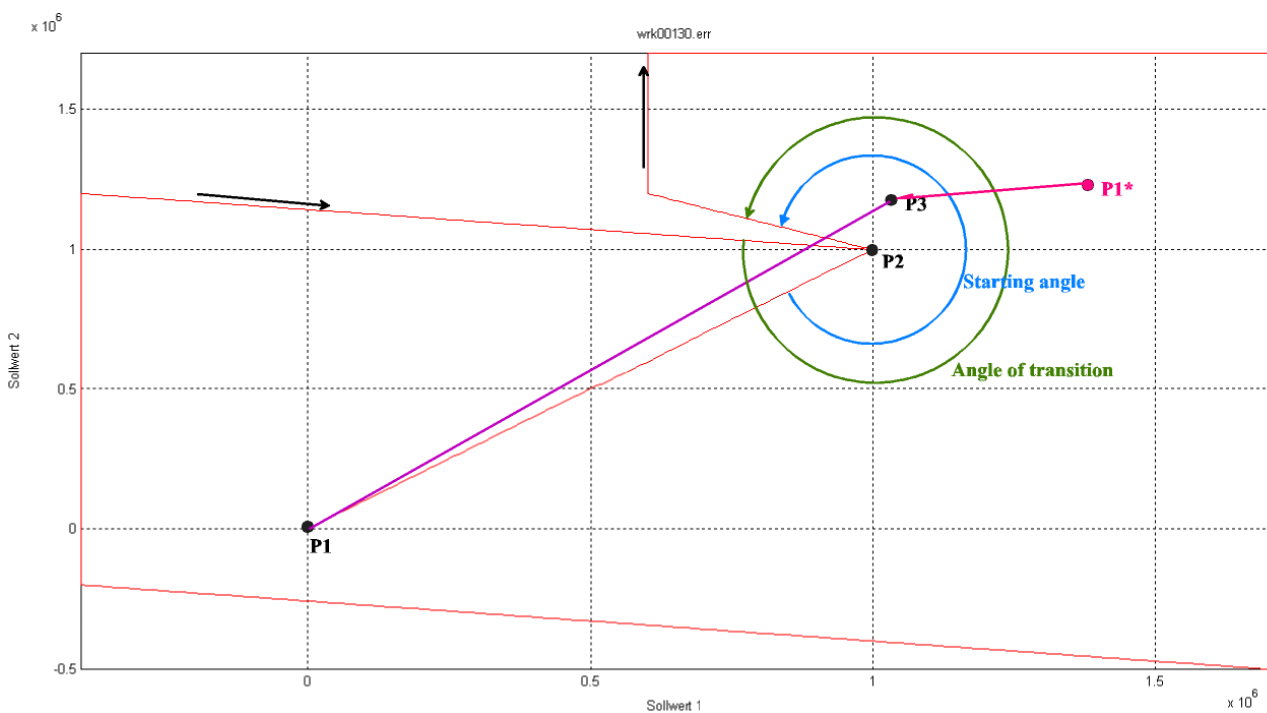
In the above graphic, the TRC should be selected to the right of the contour. As of a transition angle of over 180° between the last and the first contour blocks, the first point of the equidistant path is determined using the direct selection method. This is P3. The diagram illustrates that the direct connection P1 to P3 violates the contour marked in red.

The angle shown as the starting angle in the graphic may be maximum 180° in order not to damage the contour. The alternative point of selection P1* prevents contour damage.

NOTICE
If one of the restrictions is violated, an error is output.

13.2.5 Direct selection/deselection of TRC without block (G239)

The approach motion with G239 is executed immediately as with G138.

When TRC is deselected (G40) the last equidistant point is the current position. A subsequent motion block to reduce the tool radius is not required.

NOTICE
An arc programmed directly after G40 causes an error message because then the real deselection position and the programmed starting point of the circle are no longer identical.

In the examples below the programmed path is displayed in black and the equidistant path is displayed in blue.

Example 1

Then after TRC is deselected, a position is programmed in both main axes and executed in this way.

```
%G239_Demo1.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40 (Deselect TRC without block)

N100 G00 X150 Y0
N110 M30
```
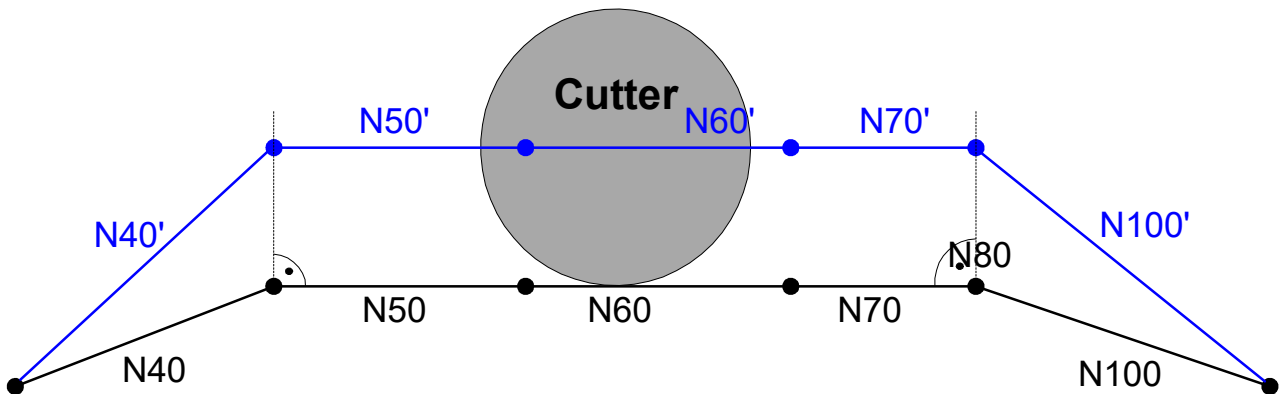

Fig. 134: Motion block sequence of G239_a position in both main axes

Example 2:

Here only one of the main axes is programmed after TRC is deselected. The position of the 2nd main axis remains the same.

```
%G239_Demo2.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40 (Deselect TRC without block)

N090 G00 X150
N100 G00 X200

N110 M30
```
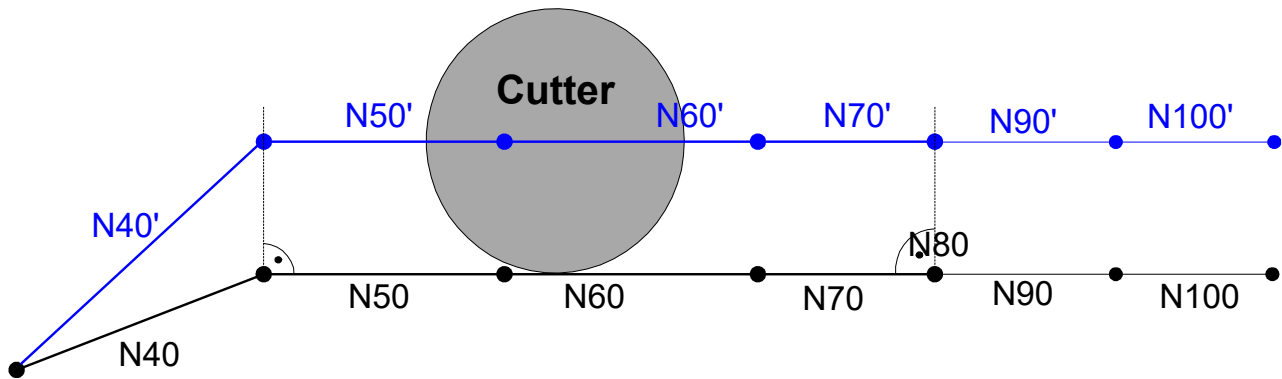
Fig. 135: Motion block sequence of G239_only one main axis programmed

Example 3:

When TRC is selected again directly after G40, only one of the two main axes is programmed.

```
%G239_Demo3.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40 (Deselect TRC without block)
(Y is not programmed again)
N090 G41 G01 X150       (Reselect TRC)
N100 G00 X200

N110 M30
```
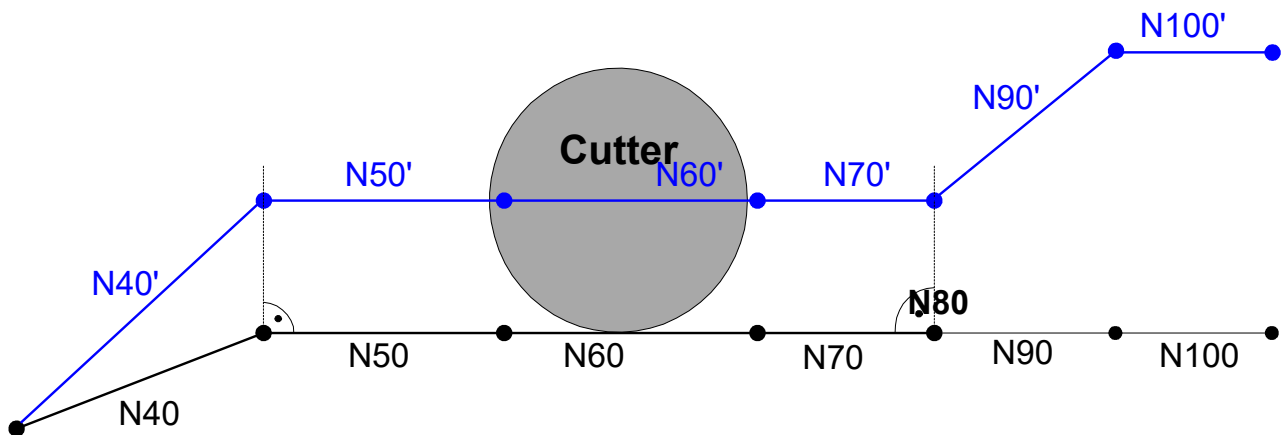

Fig. 136: Motion block sequence of G239_only one main axis programmed

Example 4:

Here the 2nd motion block is only programmed in the 2nd motion block after TRC is reselected.

```
%G239_Demo4.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40 (Deselect TRC without block)

N090 G41 G01 X150       (Reselect TRC)
N100 G00 X200 Y20

N110 M30
```
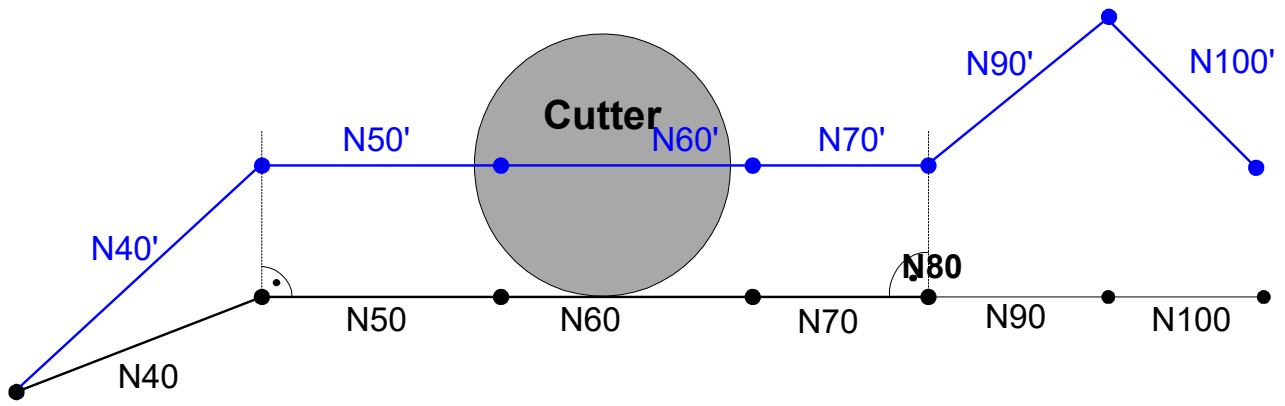
Fig. 137: Motion block sequence of G239_Program the 2nd main axis in 2nd motion block

Example 5:

In this example the position of second main axis is programmed as for the first selection (G41) after TRC is reselected after G40 .

```
%G239_Demo5.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40 (Deselect TRC without block)

N090 G41 G01 X150 Y20   (Reselect TRC)
N100 G00 X200

N110 G40
N120 M30
```
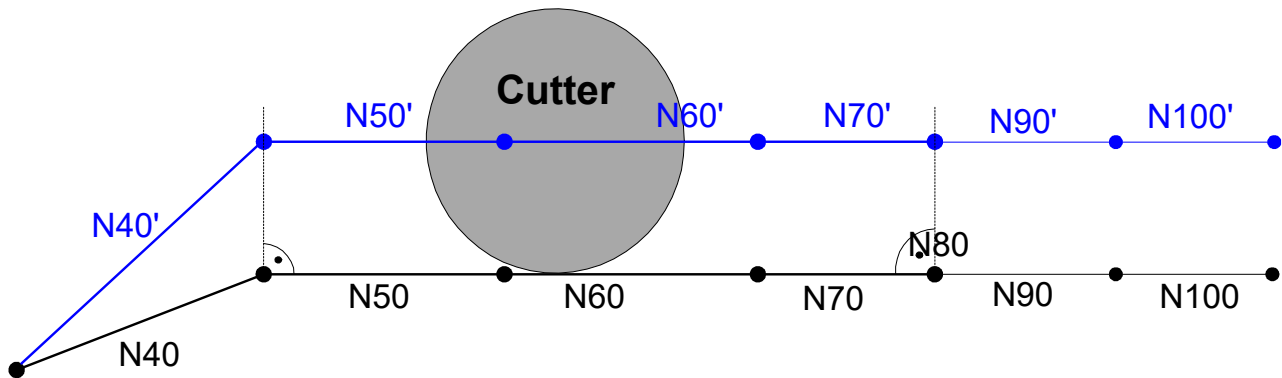

Fig. 138: Motion block sequence of G239_Program the 2nd main axis as for the 1st selection

Example 6:

The arc programmed after TRC is deselected cannot be moved because the specified centre point (M) no longer corresponds to the (new) starting point (S) and the (programmed) end point (E).

```
%G239_Demo_Kreis.nc
N005 G162
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 30
N040 G41 G01 X30 Y20       (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G40 (Deselect TRC without block)
N080 G02 X140 Y-30 J-50 (Error, centre point not correct)
N100 G00 X200
N110 G40
N120 M30
```
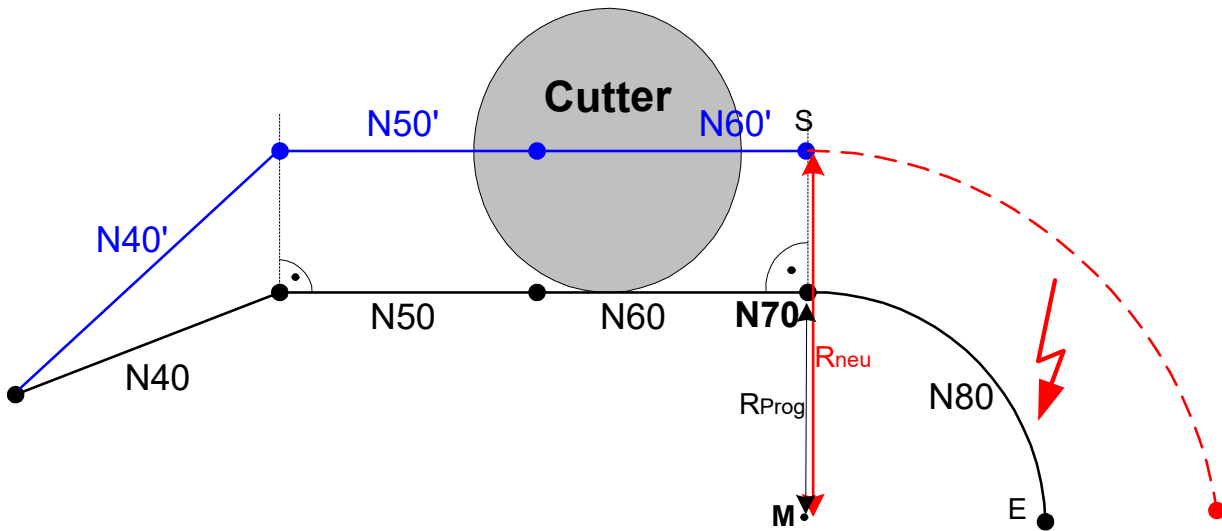

Fig. 139: Motion block sequence of G239_Centre point does not match starting and end points

A message is output with ID 20035.

13.2.6   Direct selection/deselection of TRC on the path (G236)

2 motion blocks containing motion information are required to select or deselect G236.

The programmed motion blocks in the figures below run from S to E1 and from E1 to E2.

It is permitted to select or deselect all combinations of linear and circular blocks.

> **i**   If the transition strategy G25 (Insert linear blocks) is used with G236, an arc is integrated at the corresponding transition angle when TRC is selected or deselected.

> **i**   If a 2-path configuration is used, only linear blocks can be selected or deselected. With circular selection and deselection, the associated error message 90181 or 90182 is output.

Selection

The tool radius is built up of the following:
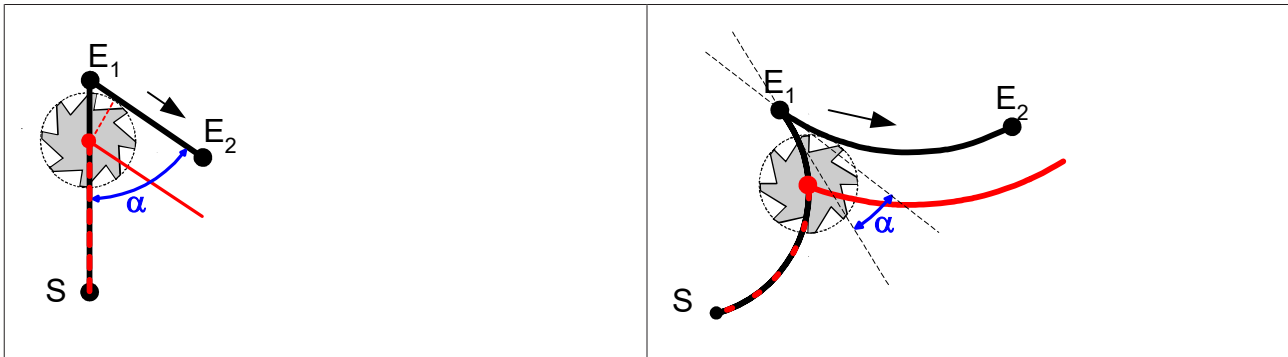
Transition angle 0° < α < 90°
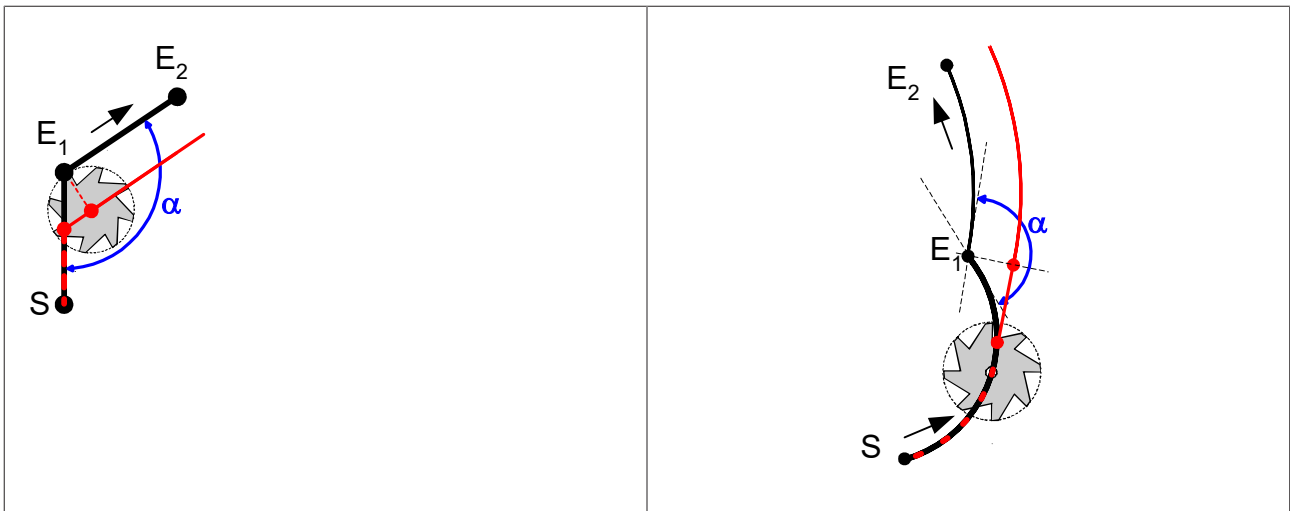
Transition angle 90° <= α <= 180° , TRC option G236_LIN= 0 (default)

Transition angle 90° <= α <= 180° , TRC option G236_LIN= 1
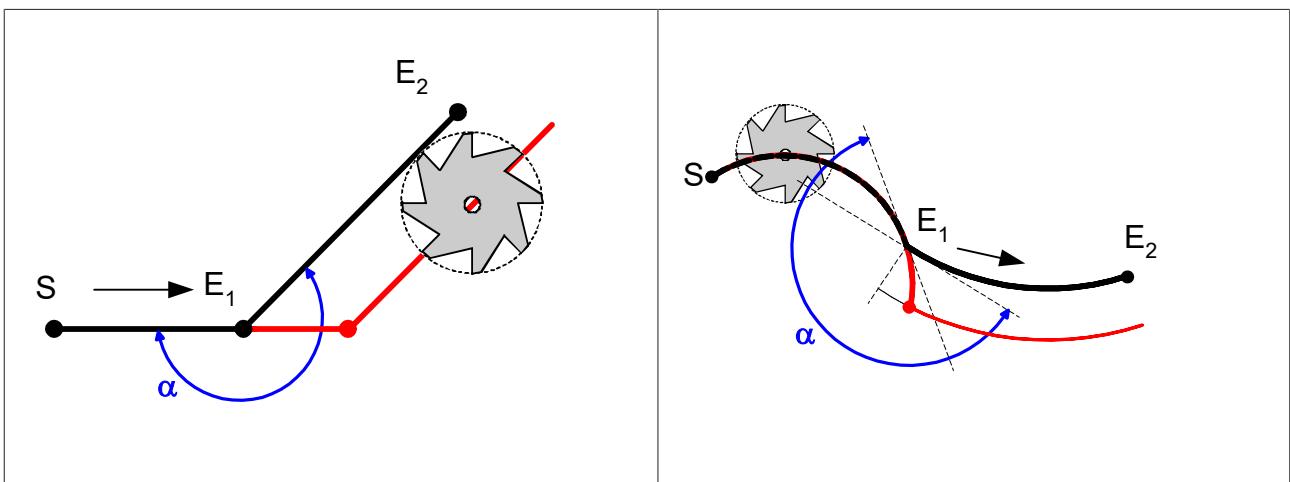
Transition angle 180° < α <= 270°
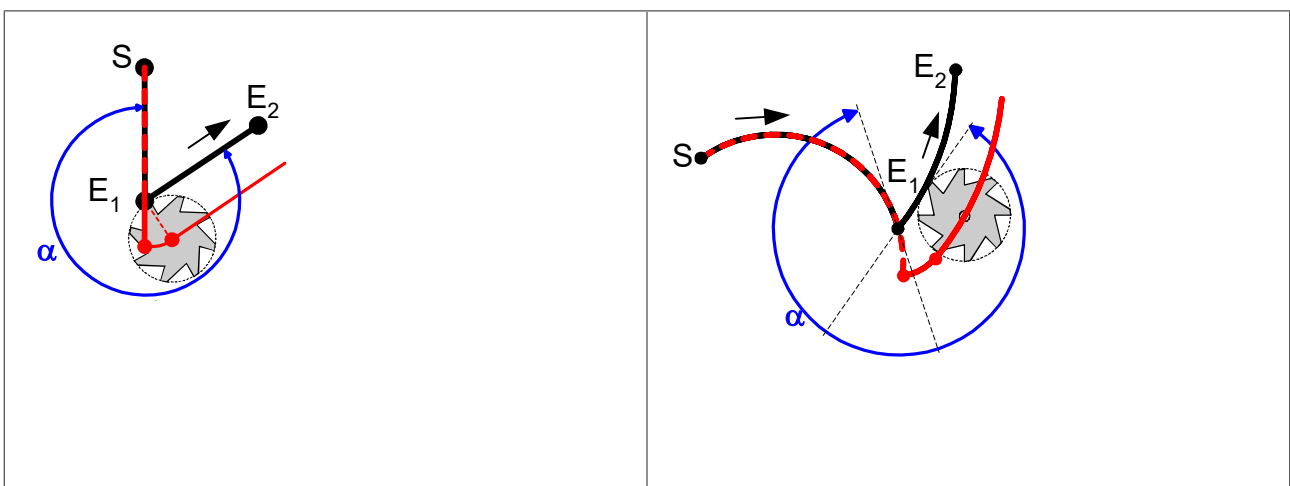
Transition angle 270° < α < 360°

deselecting

In analogy to the tool radius composition, tool radius is reduced when TCR is deselected as follows:
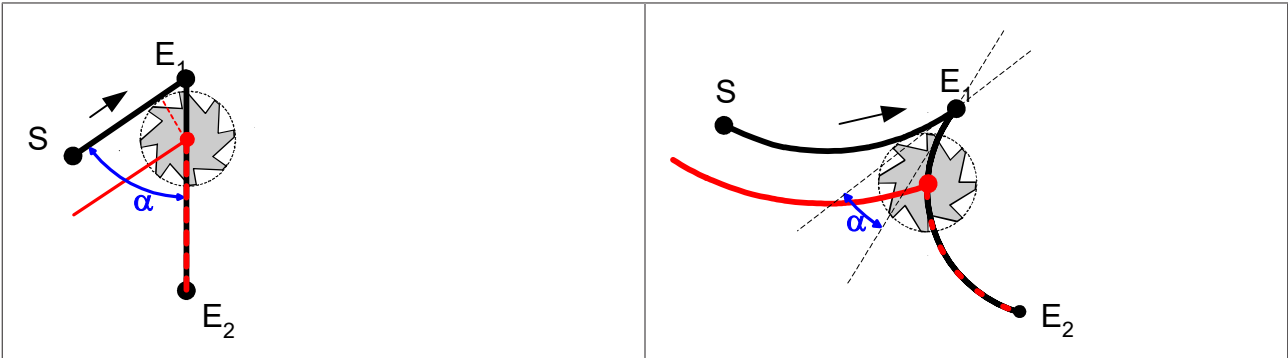
Transition angle 0° < α < 90°
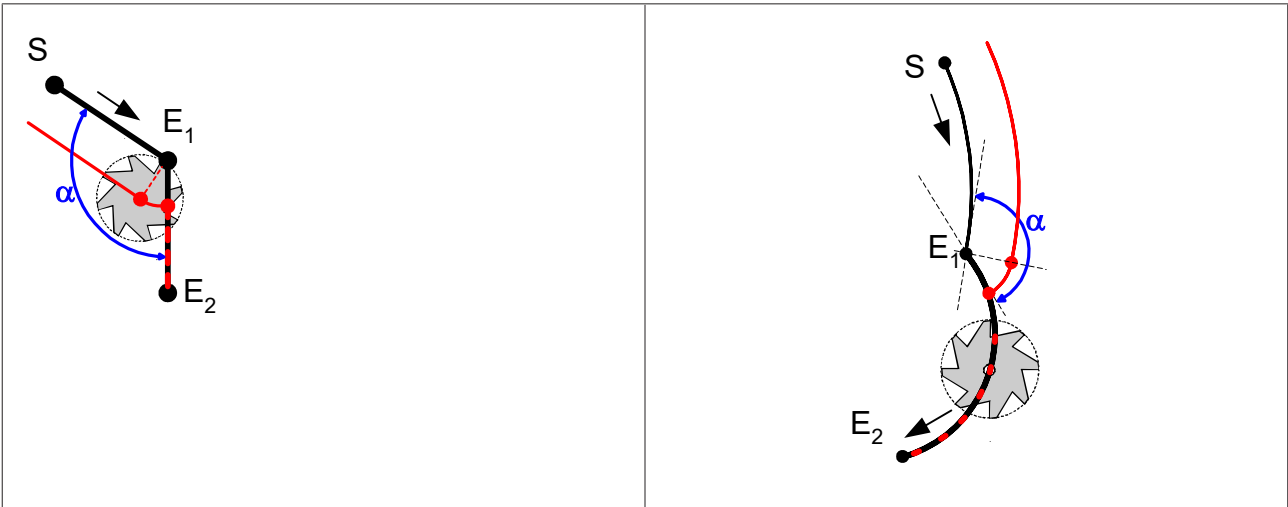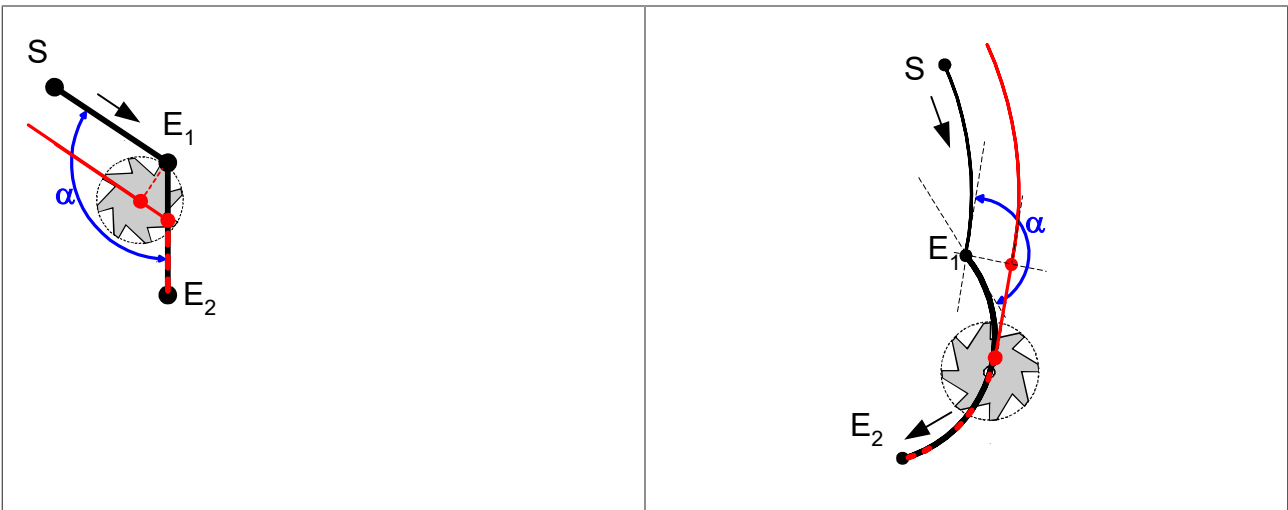
Transition angle 90° <= α <= 180° , TRC option G236_LIN= 0 (default)

Transition angle 90° <= α <= 180° , TRC option G236_LIN= 1

Transition angle 180° < α <= 270°

Transition angle 270° < α < 360°

BECKHOFF
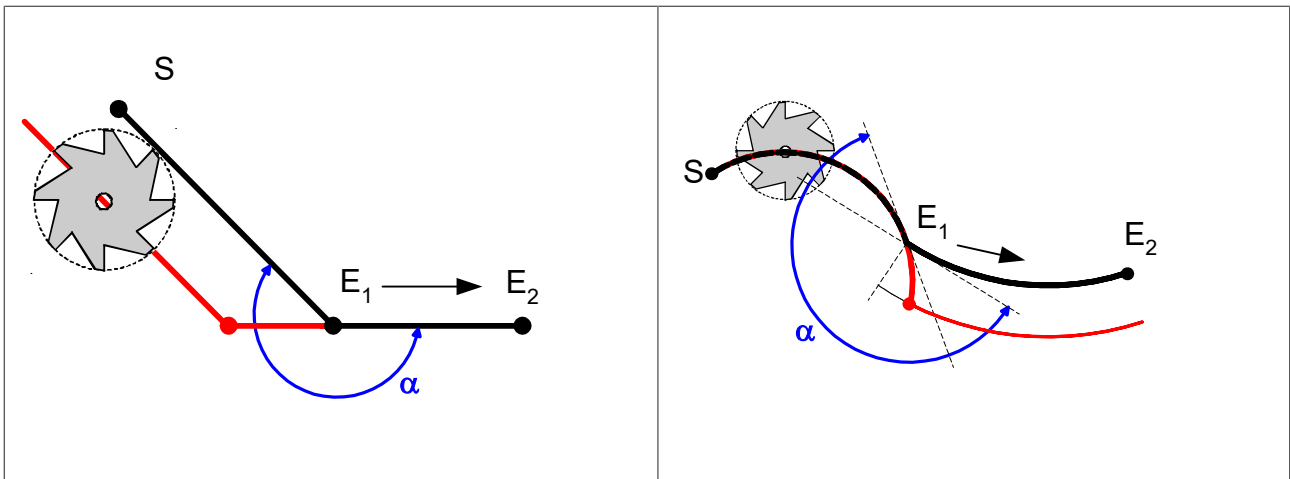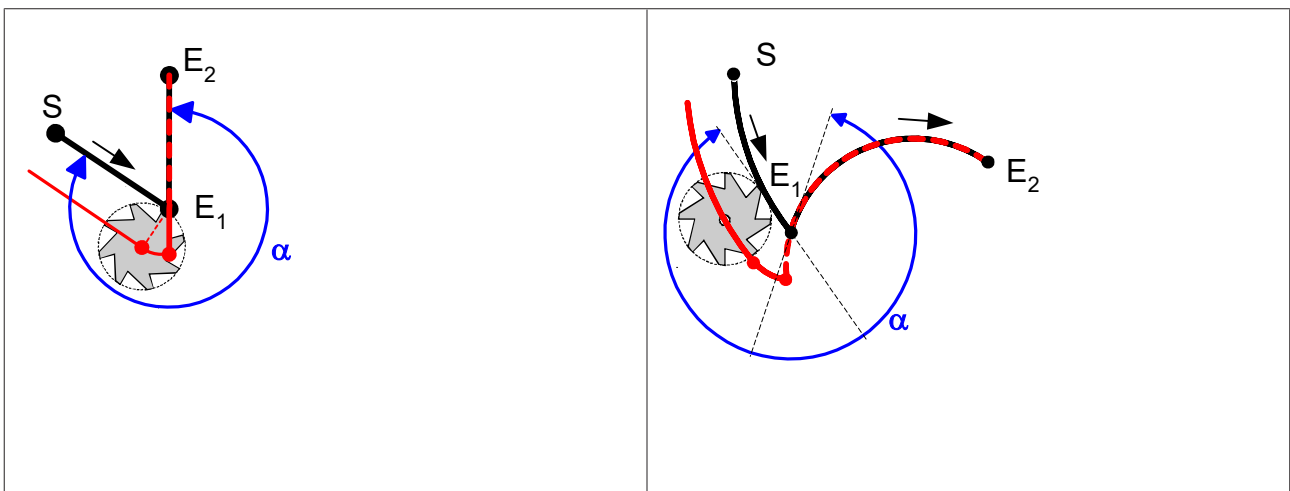
13.2.6.1 Selecting/deselecting G236 with closed contours

13.2.6.1.1 Selection/deselection at inside corners

Selection and deselection at an inside corner means that the transition angle β between the first and last motion block of the contour is less than 180°.
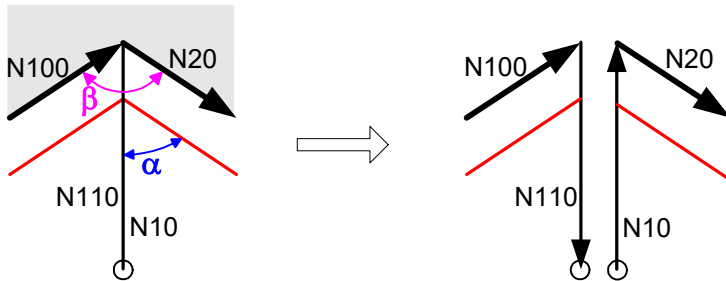
Fig. 140: Selection and deselection of closed contours at inside corner

It is recommended to place the selection and deselection point of the TRC on the bisecting line in the selected corner to avoid damaging the contour.

The contour is damaged if the selection and deselection point is incorrectly placed.
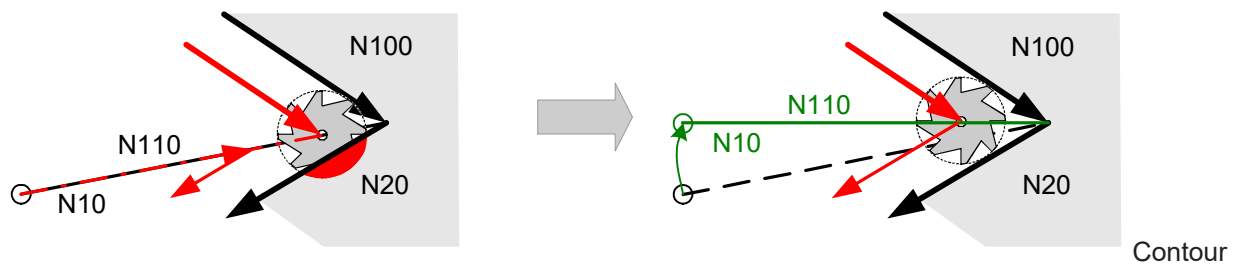
violation with G236 at inside corner

Select and deselect with G236

```
%g236_test.nc
N01 G236 D1 F1000
N02G01 X10 Y10 (Selection point)
N05 G42
N10 G01 X30 Y30
N20 G01 X30 Y20
;
N100 G01 X10 Y30
N105 G40
N110 G01 Y10 Y10 (Deselection point)
;
N999 M30
```

13.2.6.1.2 Selection/deselection at outside corners

Selection and deselection at an outside corner means that the transition angle β between the first and last motion block of the contour is greater than 180°.

It is recommended to place the selection and deselection point of the TRC within the area marked green, as depicted in the fígure below.
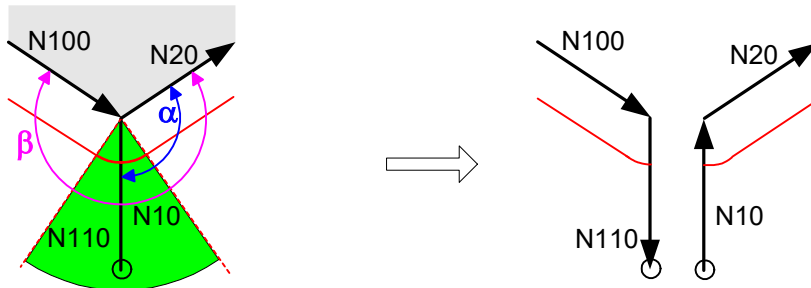
Fig. 141: Selection and deselection of closed contours at outside corner

13.2.7   Generate compensation blocks

The generation of compensation blocks is not subject to restrictions with respect to the NC block sequence as applies to direct or indirect TRC selection. The next block after the current block is used to calculate contour transitions between the programmed NC blocks.

The contour transition is on straight lines (G25) by default – or optionally on arcs (G26) – with the option of feedrate adaptation (G10/G11).

The table below and the supplementary diagrams show all the possible contour transitions, whereby both the possible transitions (linear and circular interim blocks) are shown.

Insert blocks by the TRC						
LIN: Linear block, CIR: Circular block						
G25: Insert linear transition						
G26: Insert circular transition						
(2 LIN : The TRC inserts 2 linear blocks at the transition).						
	Angle range					
	0° to 180°		180° to 270°		270° to 360°	
progr. block sequence	G25	G26	G25	G26	G25	G26
LIN-LIN	0	0	0	1 CIR	1 LIN	1 CIR
LIN-CIR / CIR-LIN	0	0	1 LIN	1 CIR	2 LIN	1 CIR
CIR-CIR	0	0	2 LIN	1 CIR	3 LIN	1 CIR
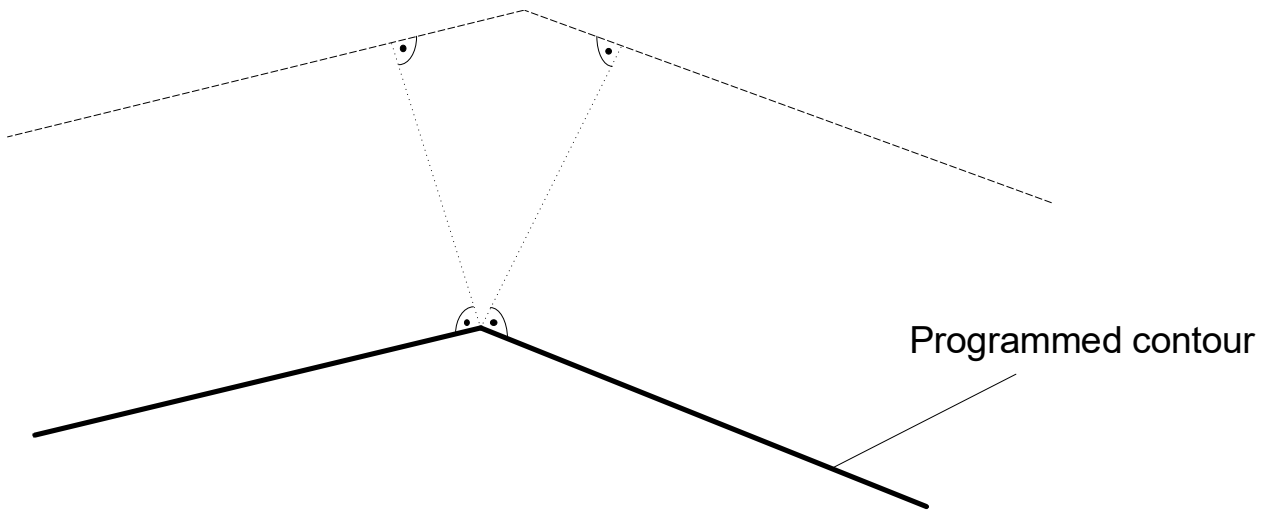
Fig. 142: Example of contour transition on straight lines for linear-linear block sequence
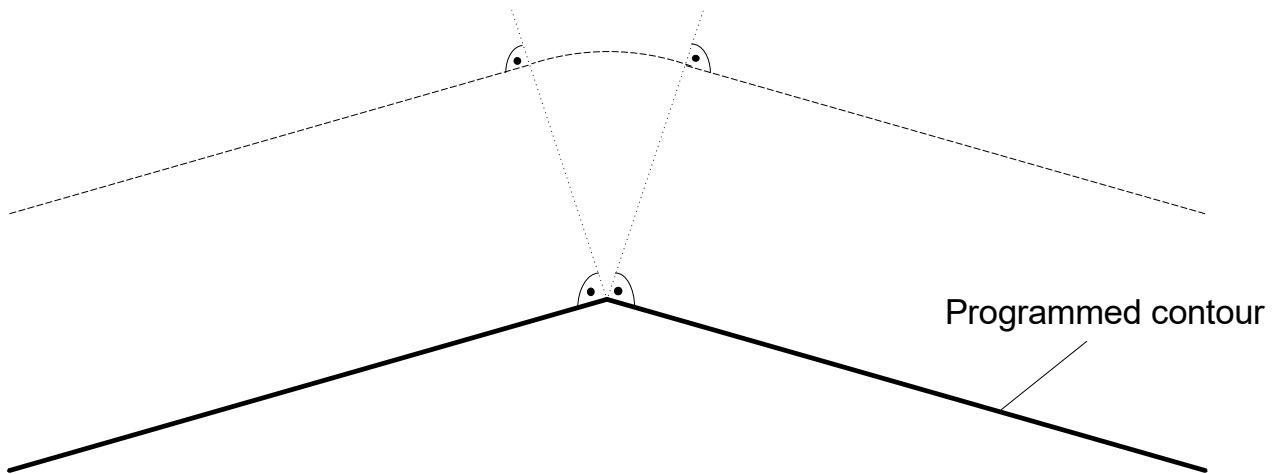
Fig. 143: Example of contour transition to an arc for linear-linear block sequence

Combination type 1: 0° < ß ≤ 180°, G41	Combination type 3: 0° < ß ≤ 180°, G41
Combination type 2: 0° < ß ≤ 180°, G41	Combination type 4: 0° < ß ≤ 180°, G41
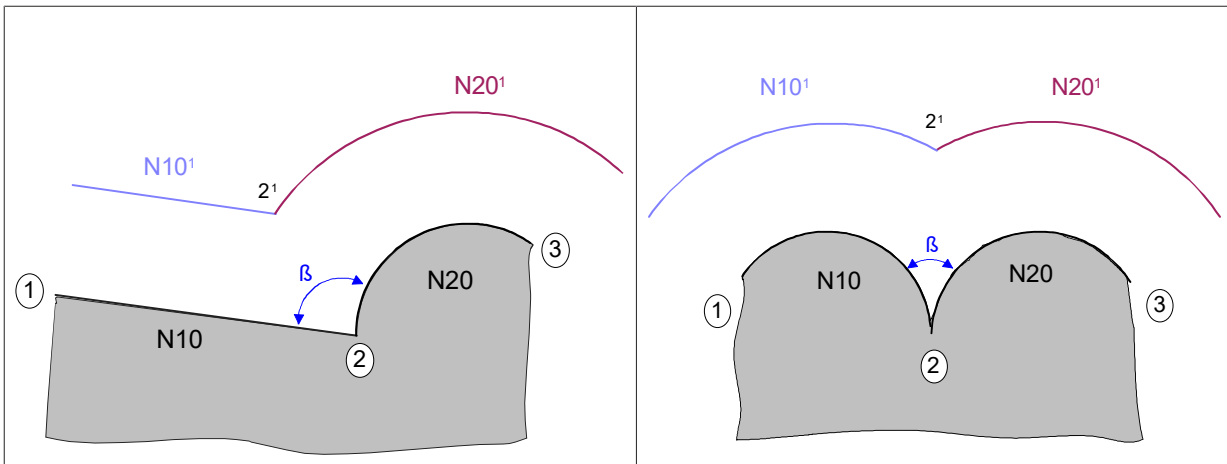
TRC transitions: Combination cases for 0° < ß ≤180 ° (independent of G25 or G26 since no compensation blocks are added)
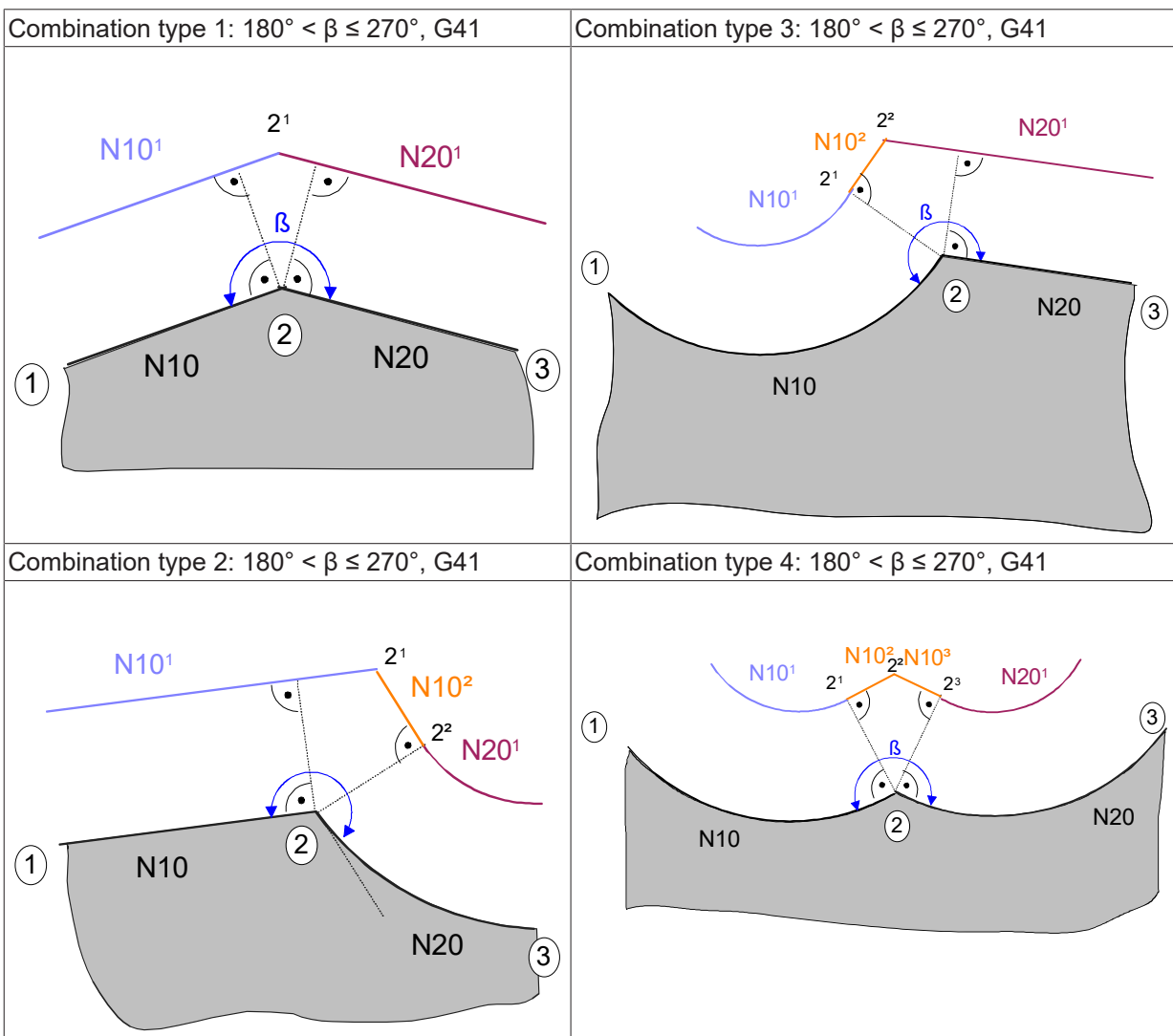
Combination type 1: 180° < β ≤ 270°, G41	Combination type 3: 180° < β ≤ 270°, G41
Combination type 2: 180° < β ≤ 270°, G41	Combination type 4: 180° < β ≤ 270°, G41

TRC transitions: Combination cases for 180° < ß ≤ 270° with linear interim blocks

Combination type 1: 270° < ß ≤ 360°, G41	Combination type 3: 270° < ß ≤ 360°, G41

Combination type 2: 270° < ß ≤ 360°, G41 | Combination type 4: 270° < ß ≤ 360°, G41

TRC transitions: Combination cases for 270° < ß ≤ 360° with linear interim blocks

Combination type 1: 180° < β ≤ 270° , G41, G26 | Combination type 3: 180° < β ≤ 270° , G41, G26

Combination type 2: 180° < β ≤ 270° , G41, G26 | Combination type 4: 180° < β ≤ 270° , G41, G26
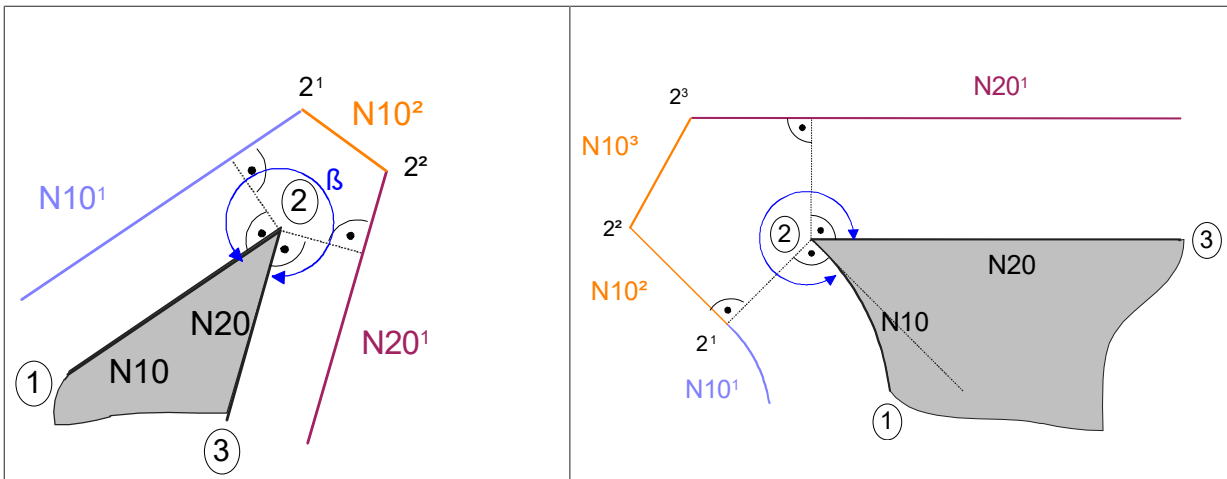
TRC transitions: Combination cases for 180° < ß ≤ 270° with circular interim blocks

Combination type 1: 270° < ß ≤ 360°, G41, G26	Combination type 3: 270° < ß ≤ 360°, G41, G26
Combination type 2: 270° < ß ≤ 360°, G41, G26	Combination type 4: 270° < ß ≤ 360°, G41, G26
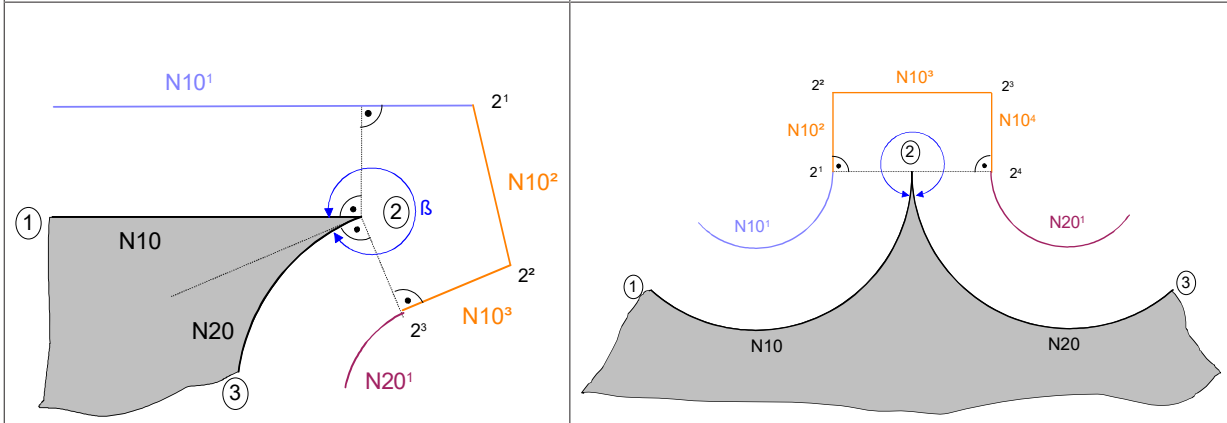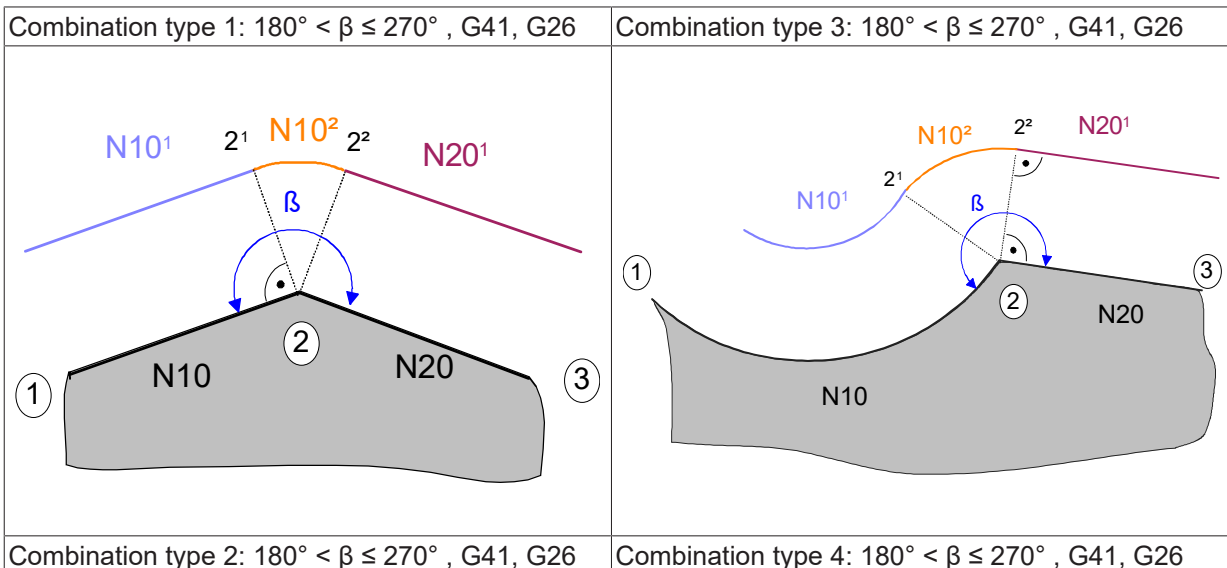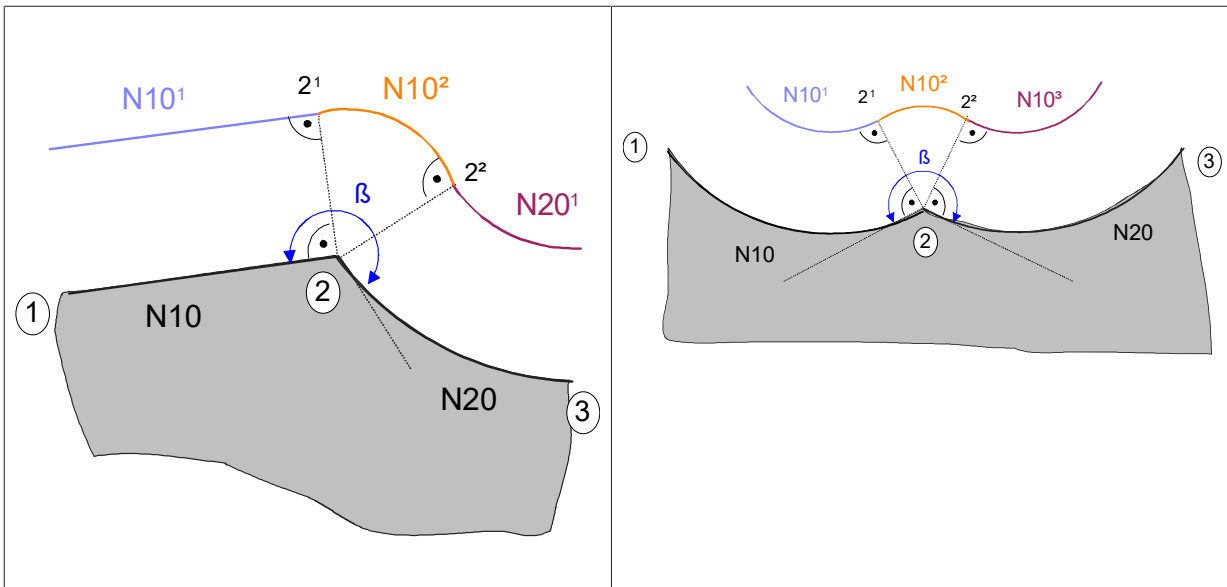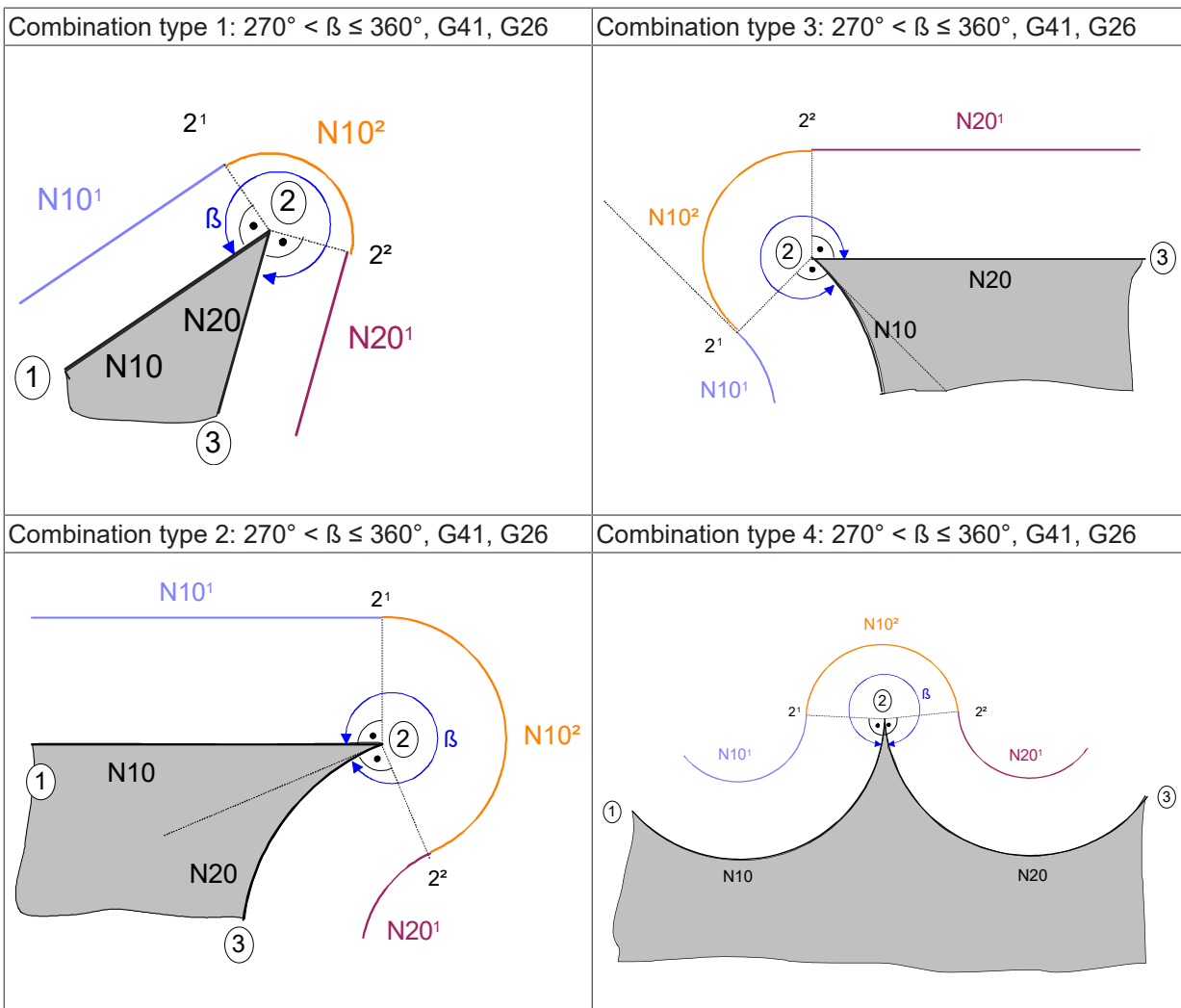
TRC transitions: Combination cases for 270° < ß ≤ 360° with circular interim blocks

13.2.8 Reaction on contour change

A direct contour change, e.g. from G41 to G42, is permitted without any explicit deselection (G40) of TRC. The contour change corresponds to a TRC deselection and selection. Contour change should only be carried out for linear NC blocks for exit and approach blocks. Circular blocks are also permitted directly before and after a contour change.

> **i** If a circular block is programmed during a contour change, an error message is output.

Reaction on contour change

N05 G01 Y10 F100 G41 V.G.WZR=2	(Linear block with select. of TRC left) (of the contour)
N10 X20 Y25	
N20 X40 G42	(Linear block with select. of TRC right) (of the contour)
N30 X60 Y10	
N40 X0 Y0 G40	(Deselect TRC)
N50 M30	
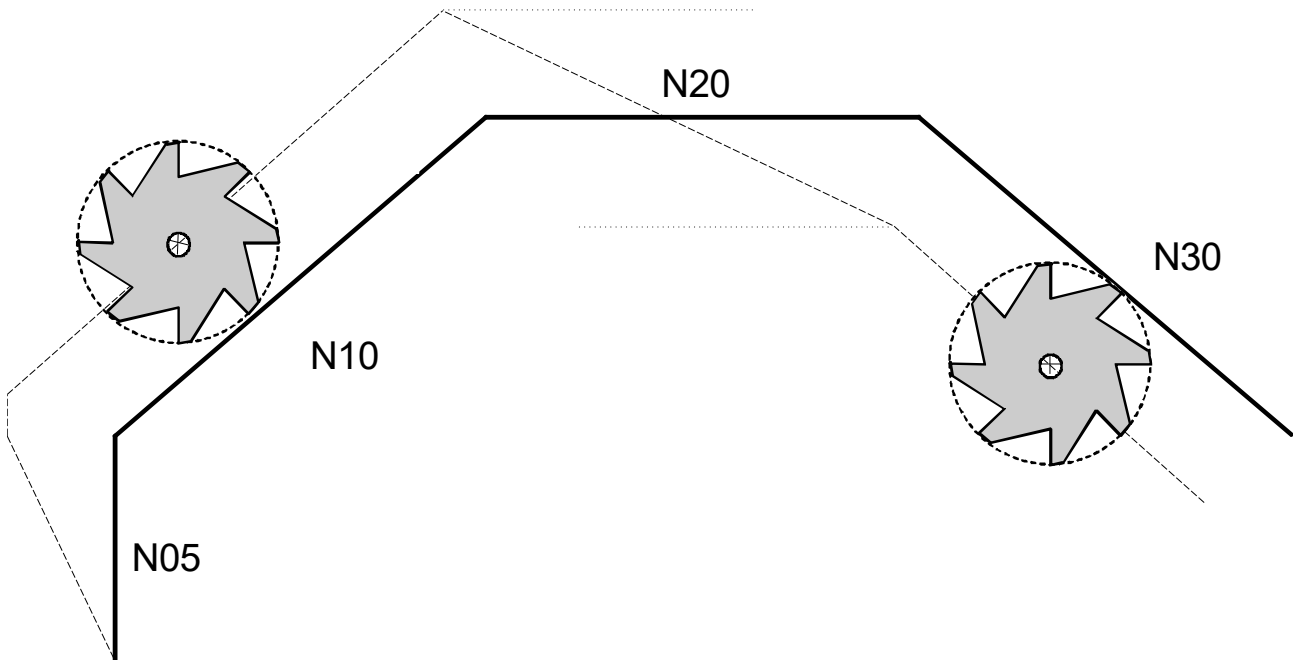
Fig. 144: Example of a selection change without deselection

13.2.9   Reaction to change in tool radius

A change in tool radius is possible both within a linear block and within circular blocks.

Change within a linear block

```
%wr_lin.nc

N10 V.G.WZ_AKT.R = 10
N20 G00 X0 Y0 Z0 F1000
N25 G41 G1 X20 Y20
N30 G01 X100
N35 V.G.WZ_AKT.R = 20
N40 G01 X200
N50 X240 Y100
...
N200 G40 X500
N999 M30
```
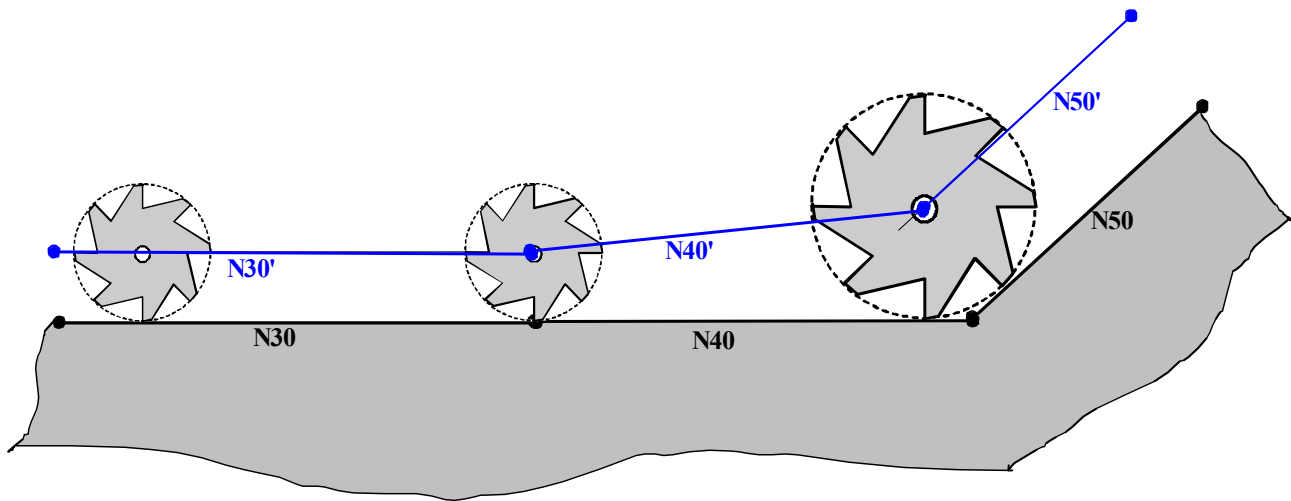
Fig. 145: Tool radius change within linear block

Change within a circular block

```
%wr_rad.nc

N10 V.G.WZ_AKT.R = 10
N20 G00 X0 Y0 Z0 F1000
N25 G41 G1 X20 Y20
N30 G01 X100
N35 V.G.WZ_AKT.R = 20
N40 G02 X180 Y100 R50
N50 X240 Y150
...
N200 G40 X500
N999 M30
```
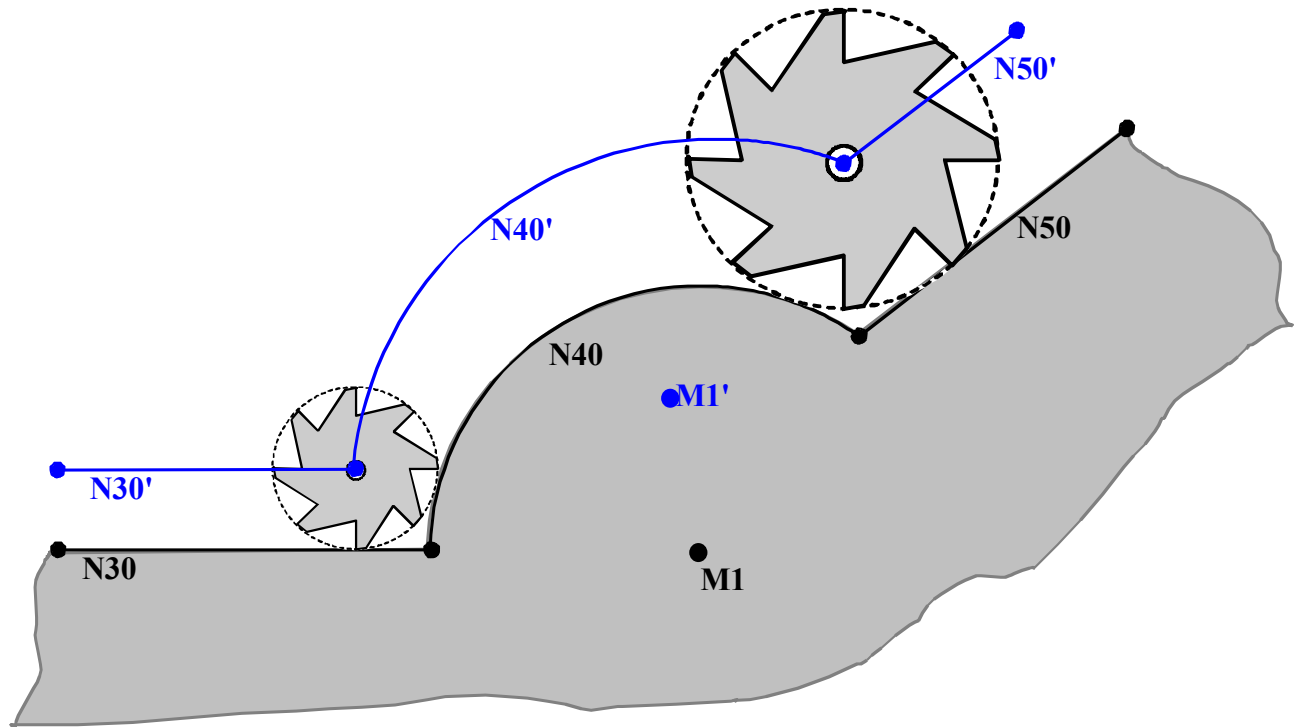
Fig. 146: Tool radius change within circular block

13.2.10   Tangential selection/deselection of TRC (G05)

When TRC is directly selected/deselected, it generally results in a kink on the path contour at the start of machining. With angles greater than 180°, the contour of the block after selection and the block before deselection are violated.

Tangential entry and exit provides a solution to avoid these contour violations in direct selection/deselection mode and to minimise the jerk that occurs at the kink points on the path.

G05 must be programmed in the same block in conjunction with G40, G41, G42. This then derives whether a tangential transition should take place at the start or at the end of the contour.

From the current position, the next contour element G01, G02... is approached tangentially in a circle at the programmed feedrate; where necessary, the feedrate must be adapted with G10/G11.

G05 in conjunction with G41 or G42 causes tangential entry at contour start; G05 and G40 cause a tangential exit at contour end when G41/G42 is active. This converts the selection/deselection block into a circular block.

The motion blocks triggered by the G05 function when TRC is selected and deselected are illustrated on the next two pages. A total of four curves are shown resulting from different starting points (P1, P1') with the same programming.

Selection in tangential mode:

The selection point (AnP) is calculated in the same as in conventional direct selection. The direction of rotation of the circular selection block is specified based on the orientation of the first selected block and the position of the starting point. The circle centre point (MP) results from the intersecting point of the centre perpendiculars from the starting point (P1 or P1') and the selection point (AnP) and the straight line starting point of the selection block (P2) – selection point (AnP).

Deselection in tangential mode:

The last compensated end point (AbP), referred to as last selection point here, is calculated in the same way as in conventional direct deselection mode. The direction of rotation of the circular deselection block is specified based on the orientation of the last selected block and the position of the deselection point (P4 or P4'). The circular centre point results from the intersecting point of the centre perpendiculars to the connecting line of the last selection point (AbP) and the deselection point (P4 and P4') and the straight lines of the last selection point (AbP) - last programmed point (P3) before deselection.
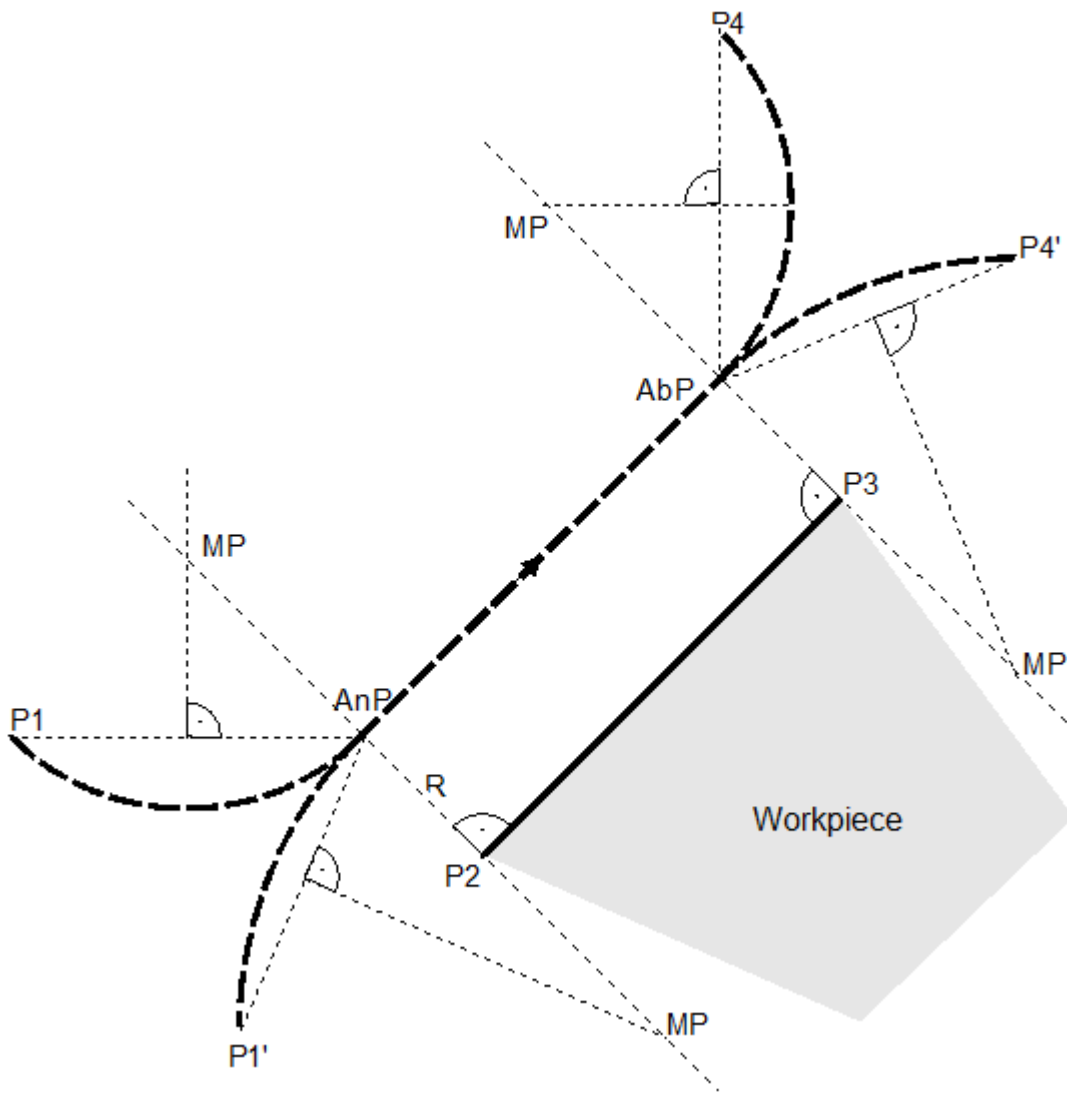
Fig. 147: Selection and deselection of TRC in tangential mode
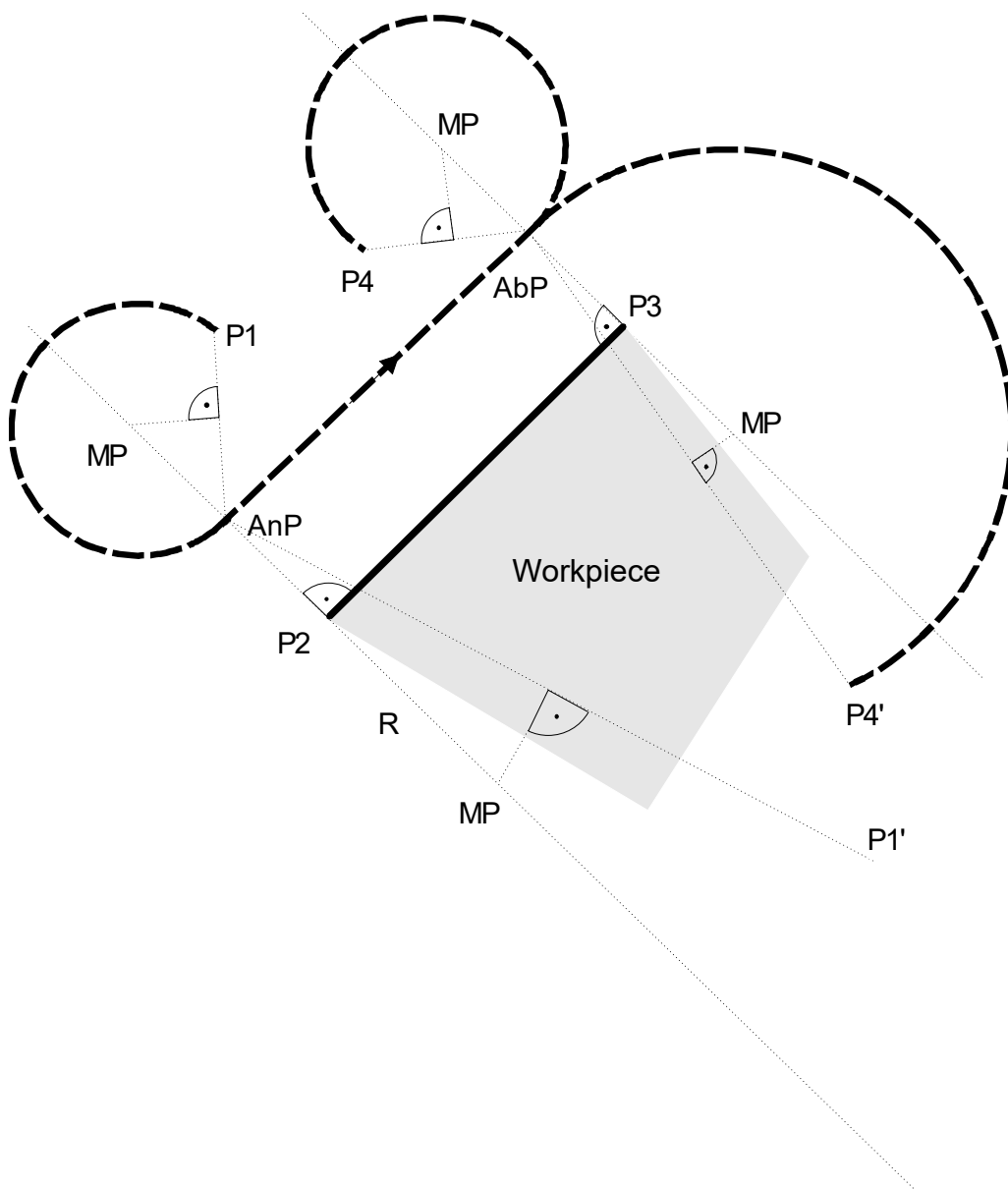
Fig. 148: Selection and deselection of TRC in tangential mode

13.2.11   Adapting feed of TRC (G10/G11)

Syntax:

G10	Constant feedrate	modal, initial state
G11	Adapted feedrate	modal

When G11 is active, transition blocks inserted by tool radius compensation are travelled at a higher velocity. In addition, the programmed feedrate for circular blocks acts on the tool engagement point (see figure below).
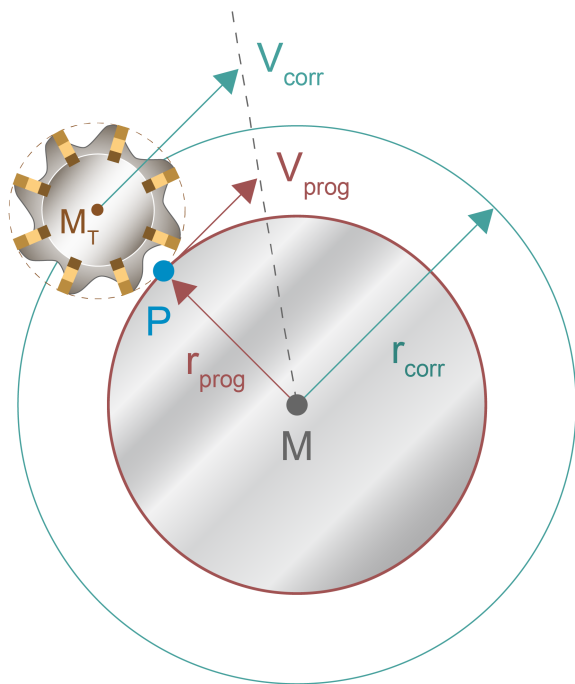
$$V_{corr} = \frac{r_{corr}}{r_{prog}} \times V_{prog}$$

P = Tool engagement point

M_T = Tool centre point

r_{prog} = programmed radius

r_{corr} = Corrected radius

V_{prog} = programmed velocity

V_{corr} = corrected velocity

Adapting feedrate with compensated circular interpolation

13.2.12 Selecting/deselecting TRC contour masking (G140/G141)

Syntax:

G140	Deselecting contour masking	modal, initial state
G141	Selecting contour masking	modal

If a rough-machining operation with a large diameter tool is executed before finishing, there is a danger of contour violation if single contour elements have smaller dimensions than that of the tool geometry. To move along contiguous contour objects, certain contour elements can be masked if violations are detected by G140/G141 when TRC is active and then to process the next contour to avoid damaging it.

No error message is output if contour masking is still selected at program end. At program start the initial state G140 is active.

NOTICE
Contour masking remains selected if tool radius compensation is deactivated by G40 and is re-activated when tool radius compensation is reselected.
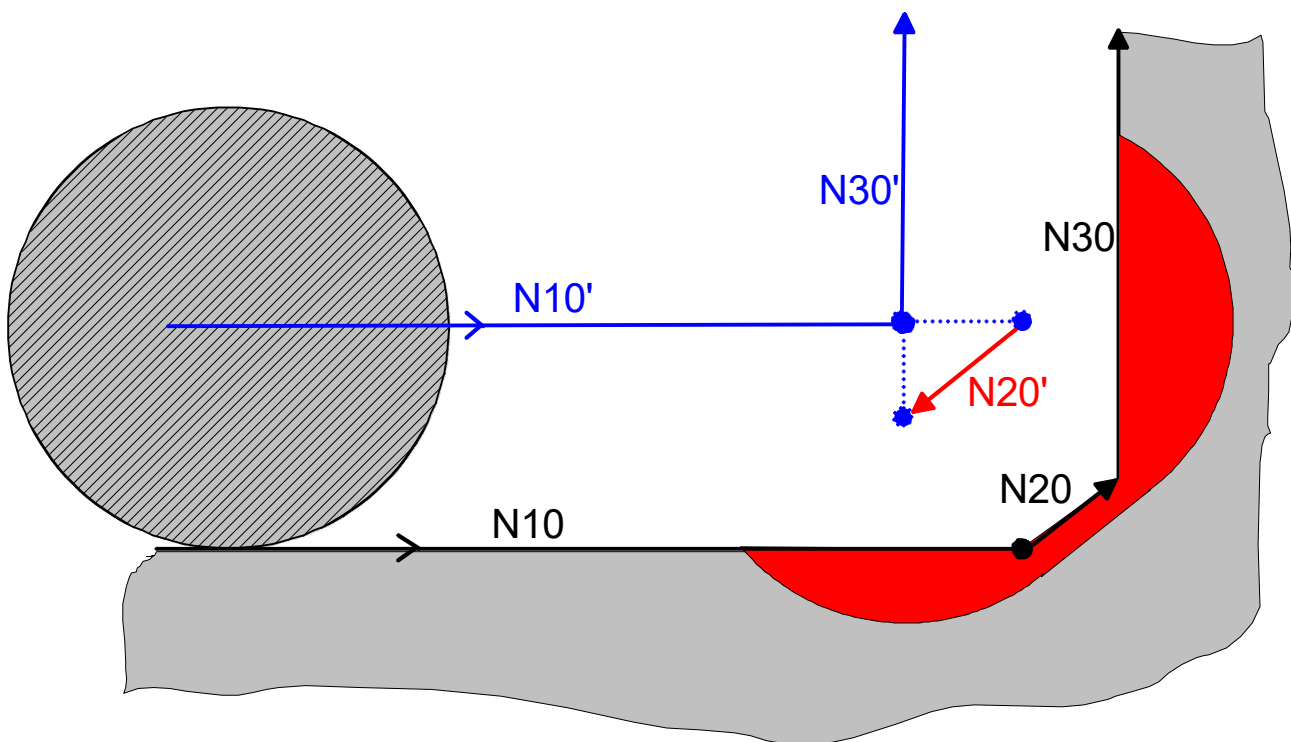
Fig. 149: Masking of N20 to avoid contour violation.

13.2.13   Limits of TRC

The TRC takes into account the current and two further relevant NC blocks to calculate interim blocks. "Relevant" here means: motion blocks in the selected main plane in which tool radius compensation takes place. "Non-relevant" would refer to techno blocks, time delays (G04) or path motions with one axis perpendicular to the main plane. If an interim block running in the opposite direction to the programmed contour is calculated within these three blocks, a contour violation is detected. The figure below shows an example of this.

If a bottleneck position is caused by one or more blocks which are more than 3 blocks away from current block, this contour violation is not detected by the TRC. The figure on the next page shows an example of this.
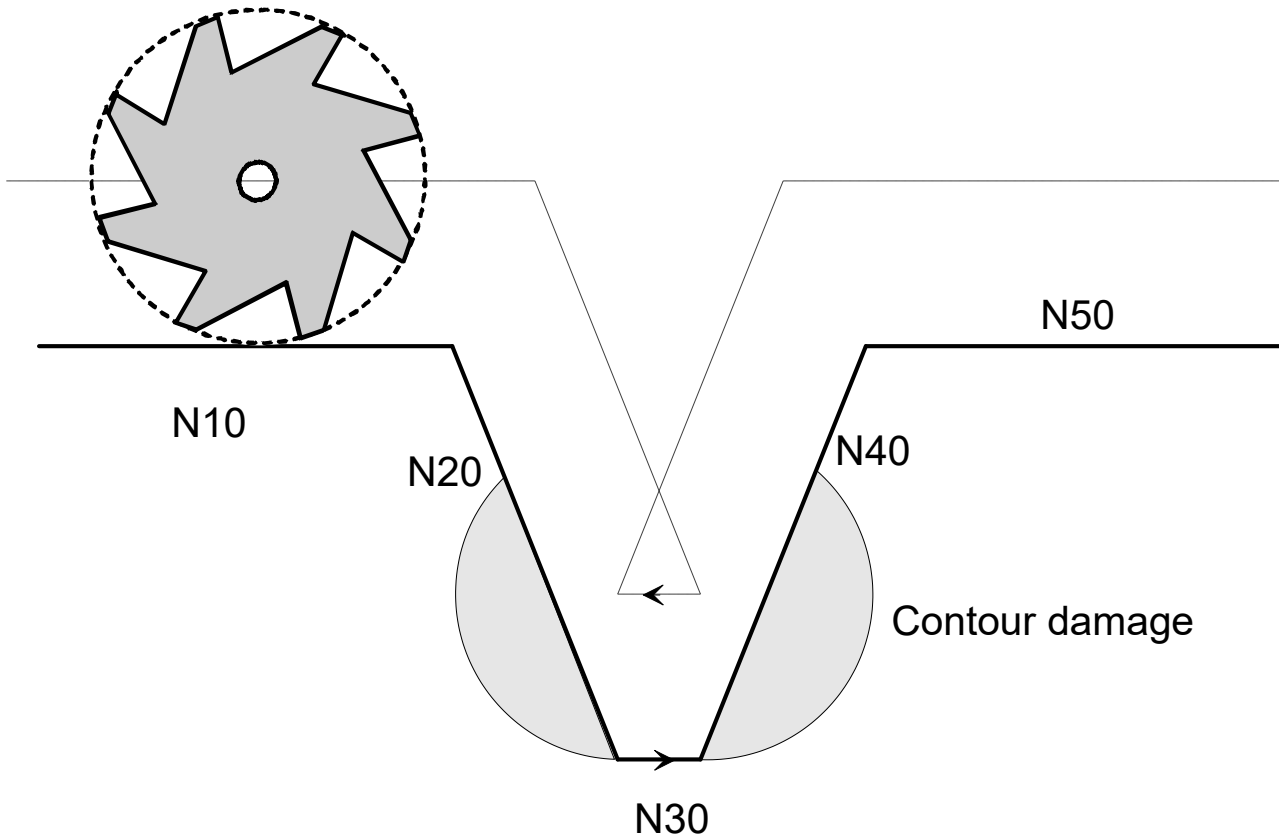
Fig. 150: Example of detected contour violation
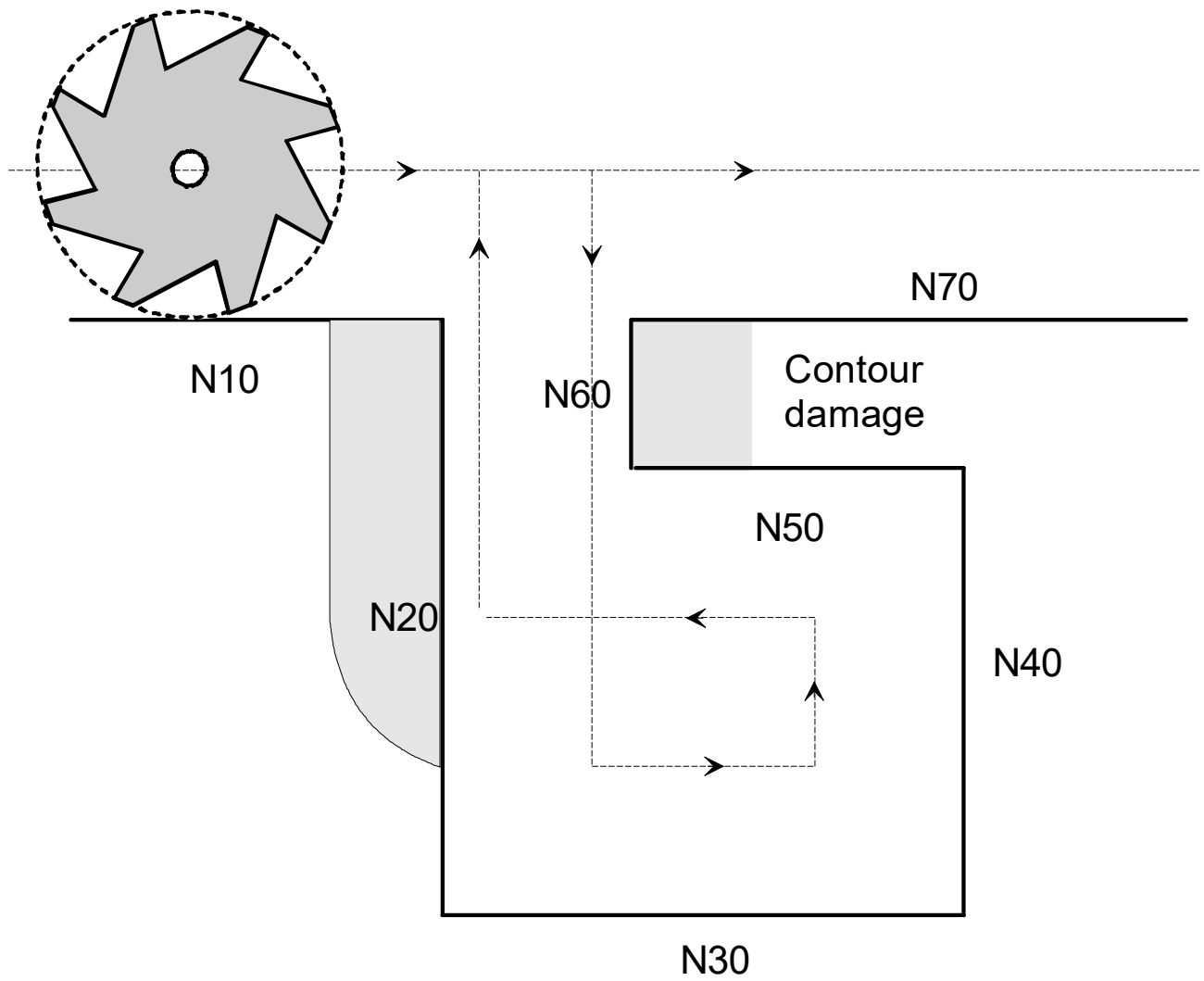
Fig. 151: Example of undetected contour violation

13.2.14   Programmable additional options of TRC (#TRC)

The following command permits the programming of additional optional TRC functions.

> **i** After selection, these optional TRC functions remain active up to main program end or RESET, but they can also be deselected at any time while the NC program is running.

Syntax:

#TRC [[CONV_CIR_TO_LIN=..] **[KERF_MASKING**=..] **[REVERSE**=..] [**IGNORE_CONT_DAMAGE**=..]
        [REMOVE_MASKED_BLOCKS=..]] **[EXT_ANGLE_BLOCK_INTERSECTION**=..]]**

CONV_CIR_TO_LIN=..   In circular blocks in which the tool radius is greater than the programmed radius of the circle element, this parameter converts the circular block directly into linear blocks. The effectiveness of this function is dependent on whether contour masking processes are active (G141).

0: No conversion of circular blocks (default).

1: Conversion of circular blocks into linear blocks.

KERF_MASKING=..   This parameter explicitly selects the masking of kerfs.

0: Kerf masking deactivated (default).

1: Kerf masking activated.

> **i** When contour masking is used, this TRC option is implicitly activated and/or deactivated.

The functionality of kerf masking is based on offsetting a programmed point as soon as the program detects that a kerf cannot be traversed by a tool.

BECKHOFF
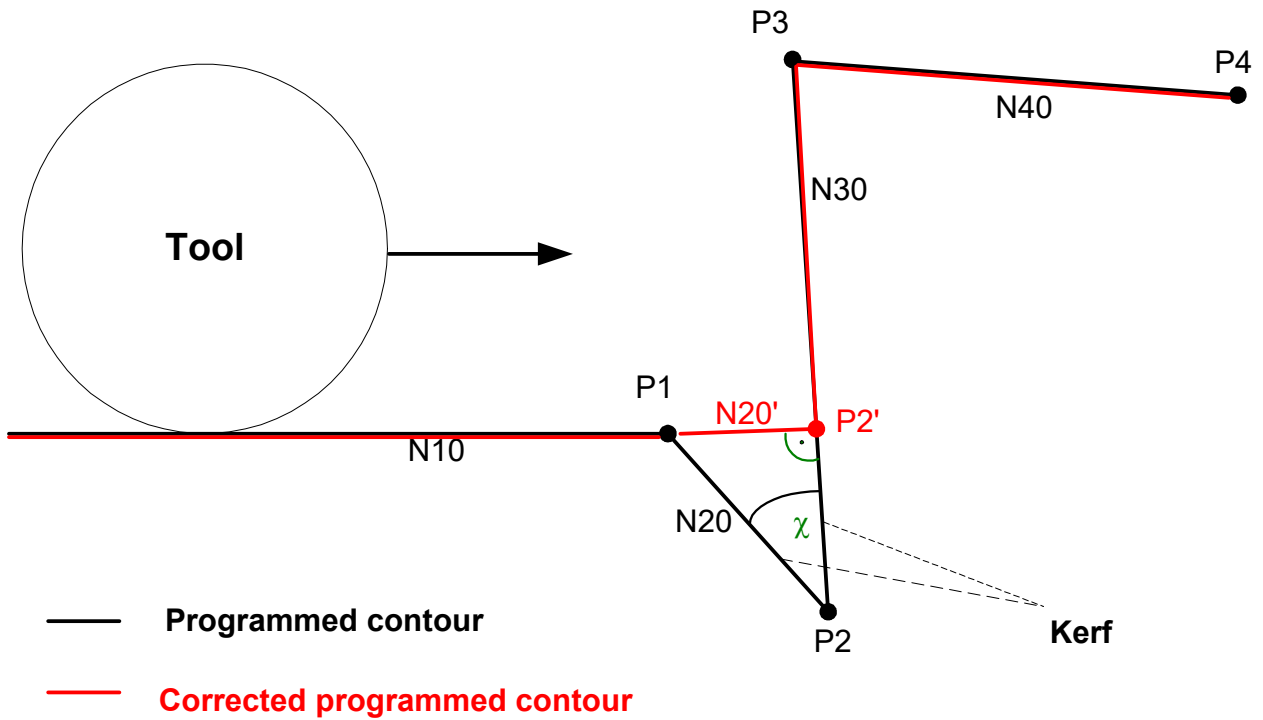
Fig. 152: Illustration of kerf masking

REVERSE=..	This parameter permits the direct change (G41<->G42) of selection side with reversing motions when TRC is active.
	0: Direct change of selection side deactivated (default)
	1: Direct change of selection side activated

The change of selection side of TRC always takes place at the point of reversal. With linear blocks, this is dependent on exactly reversing motions.

With circular blocks, the tangents of both circles must be identical at the point of reversal and the directions of both circles must be different.

IGNORE_CONT_ DAMAGE=..	This parameter explicitly ignores contour violations.
	0: Contour violations not ignored (default)
	1: Contour violations ignored

REMOVE_MASKED_ BLOCKS=..	This parameter deletes contour loops detected by contour masking. Pure motions of tracking axes are retained. From the viewpoint of TRC, this also includes a motion of the 3rd main axis.
	The parameter is specially suited for contours with very short blocks. Effectiveness is dependent on whether contour masking is active (G141).
	0: Closed contour loops are not deleted (default).
	1: Closed contour loops are deleted.
EXT_ANGLE_BLOCK_ INTERSECTION=..	This parameter changes the limit of the transition angle between two motion blocks from 180° to 181°. This avoids the creation of additional TRC transition blocks in this angle range.
	The value of the limit itself cannot be modified.
	0: Limit of transition angle is 180° (default)
	1: Limit of transition angle is 181°

Conversion of circular blocks:

```
N1000 V.G.WZ_AKT.R=1      (Tool radius)
N2300 G140 (Deactivate contour masking))
....
N2450 #TRC [CONV_CIR_TO_LIN=1] (Activate option)
N2500 G41 (Select TRC left of contour)
...
(Circular element with radius less than tool radius)
N2550 G03 X3557.83 Y-577.61 I0.00 J0.60
(no direct conversion to a linear block)
...
N3000 G141              (Activate contour masking)
...
(Circular element with radius less than tool radius)
N3550 G03 X3557.83 Y-577.61 I0.00 J0.60
(direct conversion to a linear block)


...
N3600 G40               (Deselect TRC)
```

Direct change of selection side:

```
N090 G90
N100 #TRC[REVERSE=1]
N110 G00 X0 Y0
N120 G01 X100 Y100
N130 G41 G01 X150    (Select TRC left of contour)
N140 G01 X250
N150 G01 Y200        (Reversal point)
N160 G42             (Change selection side, now right of contour)
N170 G01 X100
N180 G01 X100 Y150
N160 G40 G01 X0
N170 G01 Y0
…
```

13.2.14.1 TRC option STRETCH_FACTOR

STRETCH_FACTOR=.	This parameter can influence the bypass strategy with acute outside corners. The parameter value determines the maximum block stretch. The length is dependent of the tool radius used.
	The parameter only affects outside corners with a transition angle greater than 270 degrees.
	1-9: Factor for possible block stretch
	Default value: 1st

ⓘ **If the STRETCH_FACTOR is programmed as less than 1 or greater than 9, a warning P-ERR-22007 is output and the default value 1 is set.**

ⓘ **With highly pointed outside corners and large STRETCH_FACTOR selected, block stretch may be very large.**

Extreme block stretches are possible as well as possible overshoots of the software limit switch with corresponding error messages (P-ERR-120002/P-ERR-120003)

The option behaviour is illustrated by means of examples of block transitions linear-linear and circular-circular. The function can also be used for linear-circular and circular-linear transitions.

	%stretch_test1.nc N010 G139 G25 N020 V.G.WZ_AKT.R = 5 (Use default value) N040 #TRC**[STRETCH_FACTOR= 1**] N080 G0 X0 Y0 Z0 N090 G41 F10000 N100 G1 X10 Y20 **N110 G1 X35 Y30** **N120 G1 X30 Y20** N130 G1　 Y10 N140 G40 N150 G1 X20 Y0 N160 G0 X0 Y0 N180 M30
	%stretch_test2.nc N010 G139 G25 N020 V.G.WZ_AKT.R = 5 N040 #TRC**[STRETCH_FACTOR=2**] N080 G0 X0 Y0 Z0 N090 G41 F10000 N100 G1 X10 Y20 **N110 G1 X35 Y30** **N120 G1 X30 Y20** N130 G1　 Y10 N140 G40 N150 G1 X20 Y0 N160 G0 X0 Y0 N180 M30

|  | ```
%stretch_test3.nc
N010 G139 G25
N020 V.G.WZ_AKT.R = 5

N040 #TRC[STRETCH_FACTOR=3 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G1 X35 Y30
N120 G1 X30 Y20
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
``` |
|  | ```
%stretch_test4
.nc N010 G139 G25
N020 V.G.WZ_AKT.R = 5
( Use default value )
``` <br> ```
N040 #TRC[STRETCH_FACTOR = 1 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G2 X35 Y30 R50
N120 G3 X25 Y15 R40
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
``` |
|  | ```
%stretch_test4.nc
N010 G139 G25
N020 V.G.WZ_AKT.R = 5

N040 #TRC[STRETCH_FACTOR = 3 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G2 X35 Y30 R50
N120 G3 X25 Y15 R40
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
``` |

If the STRETCH_FACTOR is further increased with the current geometries, the parallel path does not change. The corner point is used as with factor 3.

13.2.14.2 TRC option PERPENDICULAR_RADIUS_CHANGE

PERPENDICULAR_
RADIUS_CHANGE=..

This parameter can extend the change perpendicularly if there is a change in tool radius.

0: No perpendicular extension of the new tool radius (default)

1: Perpendicular extension of the new tool radius

In the program examples, a change is made to the tool radius between the programmed motion blocks N50 and N60.

To illustrate the behaviour, a very large change in tool radius is chosen. Normally, a change in tool radius relate to very small corrections.

Several block transitions are shown in the examples below. All combinations of linear and circular blocks are permitted.

Change of tool radius at inside corner

	%Test.nc N05 G0 X0 Y0 F1000 #TRC [**PERPENDICULAR_RADIUS_CHANGE=1**] N10 **V.G.WZ_AKT.R= 5** N20 G42 N40 G01 X20 Y0 N50 G01 X50 Y50 N55 **V.G.WZ_AKT.R = 6** N60 G01 X100 N70 G40 X120 Y0 N99 M30
	%Test.nc N05 G0 X0 Y0 F1000 #TRC [**PERPENDICULAR_RADIUS_CHANGE=1**] N10 **V.G.WZ_AKT.R= 6** N20 G42 N40 G01 X20 Y0 N50 G01 X50 Y50 N55 **V.G.WZ_AKT.R = 5** N60 G01 X100 N70 G40 X120 Y0 N99 M30

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G42
N30 G01 X10 Y20
N40 G01 X20
N50 G03 X50 Y25 R30
N55 V.G.WZ_AKT.R = 6
N60 G03 X90 Y18 R40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```



```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42
N30 G01 X10 Y20
N40 G01 X20
N50 G03 X50 Y25 R30
N55 V.G.WZ_AKT.R = 5
N60 G03 X90 Y18 R40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```

Change of tool radius at outside corner


```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42 G25
N40 G01 X20 Y20
N50 G01 X50
N55 V.G.WZ_AKT.R = 5
N60 G01 X80 Y50
N70 G01 X120
N80 G40 X140 Y0
N99 M30
```

	%Test.nc N05 G0 X0 Y0 F1000 #TRC [**PERPENDICULAR_RADIUS_CHANGE=1**] N10 **V.G.WZ_AKT.R= 6** N20 G42 **G26** N40 G01 X20 Y20 N50 G01 X50 N55 **V.G.WZ_AKT.R = 5** N60 G01 X80 Y50 N70 G01 X120 N80 G40 X140 Y0 N99 M30
	%Test.nc N05 G0 X0 Y0 F1000 #TRC [**PERPENDICULAR_RADIUS_CHANGE=1**] N10 **V.G.WZ_AKT.R= 5** N20 G42 **G25** N40 G01 X20 Y20 N50 G01 X50 N55 **V.G.WZ_AKT.R = 6** N60 G01 X80 Y50 N70 G01 X120 N80 G40 X140 Y0 N99 M30
	%Test.nc N05 G0 X0 Y0 F1000 #TRC [**PERPENDICULAR_RADIUS_CHANGE=1**] N10 **V.G.WZ_AKT.R= 5** N20 **G25** N30 G42 X10 Y10 N40 G01 X20 N50 G02 X60 Y0 R30 N55 **V.G.WZ_AKT.R = 6** N60 G02 X90 Y12 R50 N70 G01 X100 N80 G40 X120 Y0 N99 M30
	%Test.nc N05 G0 X0 Y0 F1000 #TRC [**PERPENDICULAR_RADIUS_CHANGE=1**] N10 **V.G.WZ_AKT.R= 5** N20 **G26** N30 G42 X10 Y10 N40 G01 X20 N50 G02 X60 Y0 R30 N55 **V.G.WZ_AKT.R = 6** N60 G02 X90 Y12 R50 N70 G01 X100 N80 G40 X120 Y0 N99 M30

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G25
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G01X80 Y40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```



```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G26
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G01X80 Y40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```

13.2.14.3   TRC option SPLIT

The SPLIT or SPLIT_PATH option splits the selection and deselection block of the TRC into several segments each with its own feedrate.

Splitting is optionally defined by the path of an main level axis or by the path distance.

At selection or deselection, splitting can only be executed with transition angles less than 180°. If the angle is greater than 180°, the block is not split. The programmed feedrate is used.

> **ℹ**   **This function is available as of CNC Build V3.1.3080.05.**

Splitting is not permitted under the following conditions:

- Transition angle greater than 180°
- Selection mode G237, G238, G05
- Selection mode G236 and transition angle greater than 90°
- Selection or deselection with circular block
- Multipath programming
- Corresponding distance less than the specified segment length

```
Syntax:
```
#TRC [SPLIT | SPLIT_PATH | SPLIT_OFF [AX=<*axis_name*> | AXNR=..] COMBINED | POST | PRE
      DIST_SEG1=.. FEED_SEG1=.. [DIST_SEG2=.. FEED_SEG2=..] { \ }]

SPLIT	Split by distance of an axis
SPLIT_PATH	Split by path distance
SPLIT_OFF	Deactivate splitting
AX=<*axis_name*>	Specify the axis name when SPLIT is used
AXNR=..	Specify the axis number when SPLIT is used
COMBINED	Split on selecting and deselecting TRC
PRE	Split on selecting TRC
POST	Split on deselecting TRC
DIST_SEG1=..	Specify the length of the first segment in [mm]
FEED_SEG1=..	Feedrate of the first segment in [mm/min]
DIST_SEG2=..	Specify the length of a second additional segment in [mm], optional
FEED_SEG2=..	Feedrate of a second additional segment in [mm/min], optional
\	Separator ("backslash") for clear programming of the command over multiple lines.

Split by programmed distance of an axis

Split the selection and deselection block of the TRC by specifying the axis SPLIT and e.g. AX=Y

Selection   **Deselection**

Split by programmed path distance

Split the selection and deselection block of the TRC by path distance SPLIT_PATH

Selection   **Deselection**

Behaviour with short distance

If the length of the path or axis distance is shorter and the sume of the segment length when two segments are specified, only the 2nd segment is split as programmed and the corresponding feedrate is used.,

If only one segment is specified and the distance is too short, the entire distance is travelled at the programmed segment feedrate.

Parameterising the SPLIT option

```
;SPLIT with specification of the axis name only when PRE selection is default
#TRC [SPLIT AX=X FEED_SEG1=1111 DIST_SEG1=100 FEED_SEG2=2222 DIST_SEG2=200]
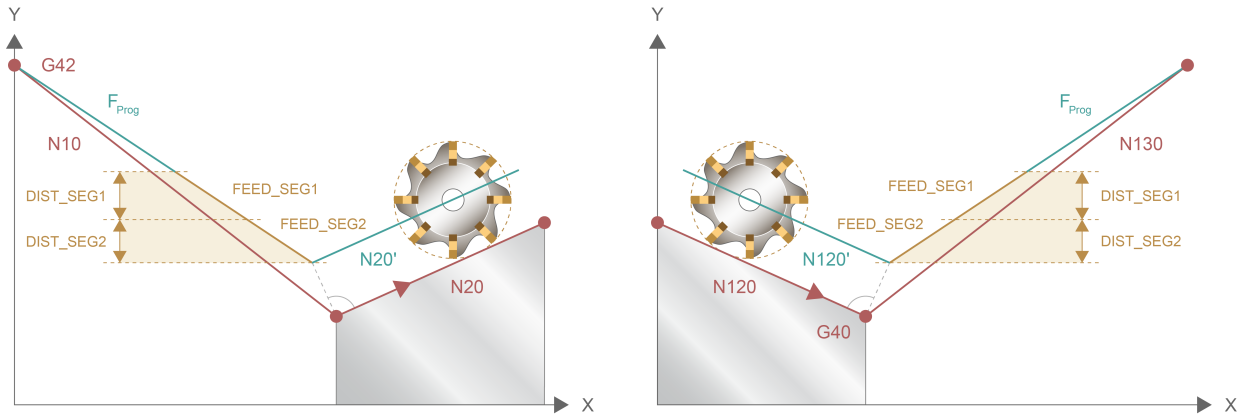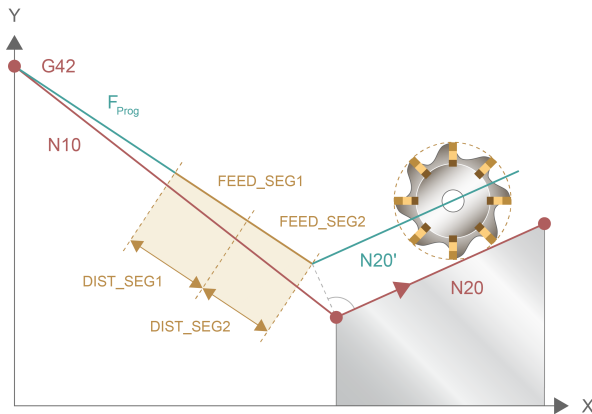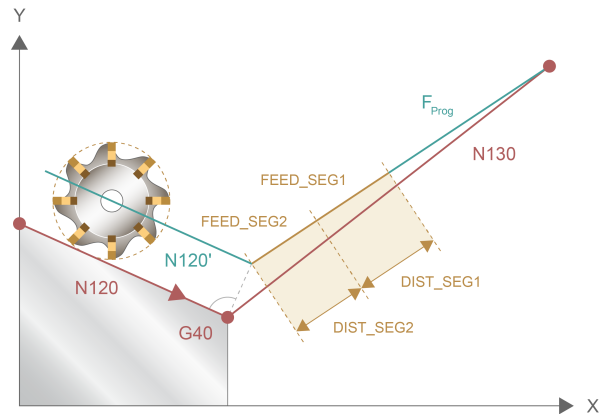
;SPLIT with specification of axis number only when POST is deselected
#TRC [SPLIT POST AXNR=2 FEED_SEG1=1000 DIST_SEG1=100 FEED_SEG2=2222 DIST_SEG2=2000]

;SPLIT with specification of axis number when COMBINED is selected and deselected
#TRC [SPLIT COMBINED AXNR=1 FEED_SEG1=1000 DIST_SEG1=100 FEED_SEG2=2222 DIST_SEG2=2000]

;SPLIT by specifying the distance SPLIT_PATH onyl when PRE is selected
#TRC [SPLIT_PATH PRE FEED_SEG1=1111 DIST_SEG1=100 FEED_SEG2=2222 DIST_SEG2=200]

;Split by specifying the distance SPLIT_PATH when COMBINED is selected and deselected with only one
additional segment
#TRC [SPLIT_PATH COMBINED FEED_SEG1=1111 DIST_SEG1=100]

;Specify splitting over several lines
#TRC [SPLIT_PATH COMBINED \
FEED_SEG1=1111 DIST_SEG1=100]

;Deselect splitting
#TRC [SPLIT_OFF]
```

;SPLIT by specifying the distance SPLIT_PATH onyl when PRE is selected

13.2.14.4 TRC option G236_LIN

This option is only effected when the G236 selection mode [▶ 512] is used and with a transition angle in the selection and deselection range of the tool radius compensation between 90 and 180°

`Syntax:`

#TRC [[**G236_LIN =**..] **]**   modal

G236_LIN=..	Define whether a circular or linear block is inserted in the described angle range of 90 to 180°
	0 : Insert a circular block (default)
	1 : Insert a linear block

13.2.15 Exception list of commands with active TRC/SRK

Below is a list of commands which are not permitted when TRC/SRK (G41/G42) are active:

(This refers to the section in the NC program between G41/G42 and G40.)

> **i** **Error 20651 is output between G41/G42 and G40 if they are followed by commands.**

> **i** **When the TRC modes G139/G138/G236/G05 are used, an additional motion block is required after G40 to reduce the tool radius.**
>
> If subsequent commands are used **directly** after G40, **error 90050** is output.

G command	Note
G200/ G201	
G303	Circles in space

A read access by synchronous V.E. variables is also not permitted.

Additional command	Note
#AX DEF	
#AX DEF DEFAULT	
#AX LINK ON/OFF/OFF ALL	
#AX REQUEST	
#AX RELEASE	Release axes. Axes that are not in the active plane may be released.
#AX RELEASE ALL	
#CALL AX	
#CAX	
#CAX OFF	
#CHANNEL INIT	
#CLEAR CONFIG	
#CS ON/OFF	
#CS MODE ON/OFF	
#CYL	
#CYL OFF	
#CYL ORI LATERAL	
#CYL ORI PROFILE	

#FACE	
#FACE OFF	
#FLUSH	
#FLUSH CONTINUE	Output is shifted to CNC but is permissible.
#FLUSH WAIT	
#GET CMDPOS	
#GET ACTPOS	
#GET MANUAL OFFSETS	
#LOAD CONFIG	
#MCS ON/OFF	
#MCS TO WCS	
#OTC OFF	
#PRESET	
#PTP ON/OFF	
#PUT AX	Release axes. Axes that are not in the active plane may be released.
#PUT AX ALL	
#SET AX	
#SET AX LINK	
#TRACK CS ABS/ON/OFF	
#TRAFO ON / OFF	
#WCS TO MCS	

14   Variables and calculation of variables

A complete variable programming list is contained in the overview of commands under Variable programming (V.) [▶ 800].

On the one hand, variables mean an internal data item of the decoder with fixed name assignment; on the other hand, they refer to self-defined variables whose designation is essentially freely selectable. With the exception of external variables (V.E...), their validity and utilisation are limited exclusively to a particular NC channel.

```
General syntax:
```
V.<NAME_1>**.**<NAME_2>**.**<NAME_3>**.{**<NAME_n>**.}**

V.	indicates access to the variables		
<NAME_1>.	Defines the global data designation:		
	"A."	represents axis-specific variables	
	"SPDL." "SPDL_PROG."	represents spindle-specific variables	
	"G."	represents inter-axis variables globally valid in the channel	
	"E."	represents external variables	
	"P."	represents self-defined variables valid up to the end of the main program (M2, M30),	
	"S."	represents self-defined variables valid throughout the main program.	
	"L."	represents self-defined local variables that are valid until the current program level is left by return (M17, M29),	
	"CYC."	represents variables that may only be used in cycle programs (L CYCLE).	
<NAME_2>.	specifies the data name		
<NAME_3>.	e.g. indicates the index if a distinction is made between several identical data.		

> **ⓘ Programming axis identification**
>
> The last identification code represents the axis code of axis-specific and several group-specific variables. Here, the designations
>
> > ".X" or "[0]"
> >
> > ".Y" or "[1]"
> >
> > ".Z" or "[2]"
>
> must be selectively used if the name "X" is assigned to the axis with the index 0, the name "Y" is assigned to the axis with the index 1 and the name "Z" is assigned to the axis with index 2 in the channel parameter list.

Absolute value of the X axis:

V.A.ABS.**X** or V.A.ABS**[0]**Analogous to spindle-specific variables, the spindle names or the corresponding indices must be used as listed in the channel parameter synchronisation
mode.   ".S" or "[0]"   ".S2" or "[1]" etc.

Logical axis number of the S spindle:

Logical axis number of the **S** spindle: V.SPDL.LOG_AX_NR.**S** or V.SPDL.LOG_AX_NR**[0]**

The lines N20/N30 cause a linear interpolation in X direction by the value of the variables V.A.BZP.Y, i.e. the reference point offset in Y direction.

```
N10 G92 X0 Y40 Z0                          ;Reference point offset
N20 G91 G01 F1000 XV.A.BZP.Y
```

```
N30 XV.A.BZP[1]                              ;here: axis index 1 == Y
N40 M30
```

The content can be read by all variables and a value can also be assigned to several of them. The access type is firmly given for each variable, however generally only a reading access is allowed. Because for most of the variables a writing access is not practical.

Here, the 2nd zero offset vector for the axis with the index 1 is assigned the value 100:

```
N10 V.G.NP[2].V[1] = 100
...
```

The EXIST function (see Section <u>Arithmetical expressions <expr> [▶ 32]</u>) checks whether a variable exists at all.

The EXIST request for an axis-specific variable checks whether a specific axis is found at all in the NC channel.

```
…
N10 G90 Y0
N20 $IF EXIST[V.A.LOG_AX_NR.X] == TRUE
N30 X-10                        ;X axis is in channel, approach position -10
N40 $ELSE
N50 #CALL AX [X,1,0]            ;X axis not in channel, request first
N60 $ENDIF
…
M30
```

Before access to an external variable, a check is made whether access is possible at all:

```
…
N10 G90 Y0
N20 $IF EXIST[V.E.POS_1] == TRUE
N30 XV.E.POS_1                          ;Move X axis to position POS_1
N40 $ELSE
N50 #MSG ["V.E.POS_1 not found!"]       ;message is output.
N55 M0                                  ;Stop
N60 $ENDIF
…
M30
```

14.1 Axis-specific variables (V.A.)

The code for axis-specific variables is "V.A. ...".

The axis-specific identifier can be specified in 2 variants:

1: Axis name according to channel list (represented with "X" as example below)

2: Axis index [i] according to channel list where <i>: 0…31

Example:

V.A.ABS.X or V.A.ABS[0]

NOTICE
V.A. variables can only be programmed for linear and rotary axes but not for spindles.

V.A.<var_name>	Meaning	Data type	Unit for input/output	Permitted access Read / Write
MENT.X	Virtual coordinate of the previous NC block (see Section Mirroring G20-G23 [▶ 110])	Real	[mm, inch]	R
PROG.X	Programmed coordinate of the previous NC block During active contour rotation (#ROTATION) the variable supplies the coordinate value mapped onto the machine axes.	Real	[mm, inch]	R
ABS.X	Absolute coordinate of the previous NC block or current absolute coordinate after NC command #CHANNEL INIT [▶ 172] each in the currently active coordinate system	Real	[mm, inch]	R
ACS.ABS.X	Current actual axis position in the present coordinate system without offsets.	Real	[mm, inch]	R
ACS.ABS.X	Current actual position during active transformation mapped onto the machine axes.	Real	[mm, inch]	R
-SWE.X	Current effective negative software limit switch	Real	[mm, inch]	R
+SWE.X	Current effective positive software limit switch	Real	[mm, inch]	R
-SWE_MDS.X	Configured negative software limit switch (acc. to P-AXIS-00177).	Real	[mm, inch]	R
+SWE_MDS.X	Configured positive software limit switch (acc. to P-AXIS-00178).	Real	[mm, inch]	R
REF.X	Machine reference point (only assigned after successful machine reference search)	Real	[mm, inch]	R
BZP.X	Reference point offset	Real	[mm, inch]	R
PZV.X	Clamp position offset	Real	[mm, inch]	R

| MESS.X | After measurement run completed, supplies the axis-specific measured value in the coordinate system in which measurement took place. The value includes all offsets in the calculation

With 2.5D: ACS values or with CS / TRAFO: PCS values | Real | [mm, inch] | R |

> **i** As of Build **V2.11.2020.07,** the axis-specific variables **V.A.MEAS.ACS.VALUE** and **V.A.MEAS.PCS.VALUE** supplement the variable **V.A.MESS**. The additional variables supply the measured value both in the axis coordinate system including all offsets and also the measured value in the programming coordinate system without offsets.

MEAS.ACS.VALUE.X	After measurement run completed, supplies the axis-specific measured value in the axis coordinate system (ACS). The value contains all offsets.	Real	[mm, inch]	R
MEAS.PCS.VALUE.X	After measurement run completed, supplies the axis-specific measured value in the programming coordinate system (PCS). The value does <u>not</u> contain any offsets	Real	[mm, inch]	R
MOFFS.X	Measurement offset	Real	[mm, inch]	R
MERF.X	Measurement run completed? If yes, then 1	Boolean	0 , 1	R
MEIN.X	Included measuring offset	Real	[mm, inch]	R
RERF.X	Homing completed? If yes, then 1	Boolean	0 , 1	R
MANUAL_OFFSETS.X or SOFFS.X	Motion path in manual mode. Only practical in conjunction with NC command <u>#GET MANUAL OFFSETS</u> [▶ 169].	Real	[mm, inch]	R
MODE.X	Current axis mode	Integer	-	R
MODULO_VALUE.X	Modulo range	Real	[°]	R
LOG_AX_NR.X	Logical axis number of an axis	Integer	-	R
AX_LIST_NAME.X	Configured axis name (acc. to P-AXIS-00297)	String	-	R
AXIS_DEACTIVATED.X	Variable indicates whether the axis was parked via the HLI. Available as of V2.11.2813	Boolean	-	R
ENCODER2_VALUE.X	Current value of a second encoder (optional) in the drive	Real	[mm, inch]	R
MIRROR.X	Mirror status of axis (1: no mirroring -1: mirroring)	Integer	-	R

14.2 Spindle-specific variables (V.SPDL., V.SPDL_PROG.)

The identifier for variables that permit access to configuration-specific spindle data is "V.SPDL. ...".

The identifier for spindle data assigned by programming is "V.SPDL_PROG. …".

V.SPDL.\<name>	Meaning	Data type	Unit for input/ output	Permitted access Read / Write
LOG_AX_NR.S	Logical axis number of the spindle	Integer	-	R
PLC_CONTROL.S	Is spindle a PLC spindle? If yes, then 1	Boolean	0 , 1	R
NBR_IN_CHANNEL	Total number of available spindles in the current NC channel	Integer	-	R
M_FCT_FREE	What is the classification of the M functions M3, M4, M5, M19? Explicitly defined as spindle M functions: 0 Freely available for other technology functions: 1	Boolean	0 , 1	R/W

V.SPDL_PROG.\<name>	Meaning	Data type	Unit for input/ output	Permitted access Read / Write
SPEED.S	Current speed S.. of spindle	Real	[rpm]	R
MOVE_CMD.S	Current motion type of spindle: For M3: 3 , for M4: 4 , for M5: 5	Integer	-	R
POSITION.S	Current set position S.POS.. of spindle for M19	Integer	[°]	R
MAX_SPEED.S	Maximum speed G196 S.. of spindle with G96. Only for main spindles.	Real	[rpm]	R
CONST_CUT_SPEED.S	Constant cutting speed G96 S.. for turning work. Only for main spindles.	Real	[m/min, ft/min *]	R
DWELL_ROT_COUNT.S	Dwell time G04 S.. in number of spindle revolutions. Only for main spindles.	Real	[rpm]	R
GEAR_DATA_STEP.S	Current gear stage G112 S.. Only for main spindles.	Integer	-	R

* [as of Build V2.11.2032.08 with G70 and P-CHAN-00360 = 1]

NOTICE
Write access causes permanent change to internal channel parameters (P-CHAN-00098)

A check is first made before programming a spindle whether it is known in the channel:

```
…
N10 G90 Y0
N20 $IF EXIST[V.SPDL.LOG_AX_NR.S] == TRUE
N30  M3 S1000        (Spindle S at speed 1000 rpm)
N40 $ELSE
N50 #MSG ["Spindle S is not present!"] (Output message and stop)
N55  M0
N60 $ENDIF
...
M30
```

14.3 Global variables (V.G.)

The identifier for inter-axis globally valid variables in the channel is "V.G. ...".

The axis-specific identifier can be specified in 2 variants:

1: Axis name according to channel list (represented with "X" as example below)

2: Axis index [i] according to channel list where <i>: 0…31

Example: V.G.NP_AKT.V.X or V.G.NP_AKT.V[0]

V.G.<var_name>	Meaning	Data type	Unit for input/ output	Permitted access: Read/ Write
BLOCK_NR	The last programmed NC block number	Integer	-	R
MASS_MM	If measuring unit [mm], then 0	Boolean	0 , 1	R
MASS_360	If measuring unit [°], then 0	Boolean	0 , 1	R
I	I coordinate of circular programming. With active centre point compensation (G165), access to corrected value	Real	[mm, inch]	R
J	J coordinate of circular programming. With active centre point compensation (G165), access to corrected value	Real	[mm, inch]	R
K	K coordinate of circular programming. With active centre point compensation (G165), access to corrected value	Real	[mm, inch]	R
R	The radius traversed during circular interpolation	Real	[mm, inch]	R
FEEDRATE	The last programmed feedrate (F word)	Real	[mm/min, m/min, inch/min]	R
FEEDRATE_MODE	G number of current feedrate mode: 93: G93 active, 94: G94 active, 95: G95 active, 194: G194 active **[as of V2.11.2039.01]**	Integer	-	R
FEEDRATE_SCALE	Permits adaptation of a feedrate programmed in mm/min to the unit specified in 'prog_start.feedrate_factor' (P-CHAN-00108). Supplies the value 1000 for 'prog_start.feedrate_factor' = 0.1 (m/min) 1 for 'prog_start.feedrate_factor' = 100 (mm/min) Programming example: F2000 / V.G.FEEDRATE_SCALE results for 1000: F2 (m/min) 1: F2000 (mm/min) **[as of V2.11.2024.00]**	Integer	-	R
MERR[i]	Compensation offset of circle centre point in the main axes of the current plane where <i> := 0 , 1	Real	[mm, inch]	R

The "WZ[j]" variables permit read access to the data of **any** tool. They are both available with an external tool management system (transparent access) and for use of an internal tool table (<j> then corresponds to the index of the tool (or the tool number) in the tool list [5] [▶ 786]).

Write access is only permitted if an internal tool table is used.

WZ[j].R	Radius of the tool	Real	[mm, inch]	R/W*
WZ[j].L	Length of the tool	Real	[mm, inch]	R/W*
WZ[j].P[i]	Tool parameter where <i>: 0 … 59	Real	-	R/W*
WZ[j].V[i] or WZ[j].V.X	Offset in axis <i> or "X" of the tool where < i>: 0 … 31	Real	[mm, inch]	R/W*
WZ[j].ME	Measuring unit of radius, length and axis offsets, always supplies 0 (for [mm]) when a tool list is used, otherwise the variable has no significance	Boolean	0 , 1	R
WZ[j].OK	Validity flag of the tool; if valid, then 1	Boolean	0 , 1	R/W*
WZ[j].SPDL_AX_NR	Logical axis number of the assigned spindle	Integer	-	R/W*
WZ[j].KIN_PARAM[i]	Kinematic parameters of the tool in internal unit where < i>: 0 … 69	Real	[0.1 μm, 10^{-4}°]	R/W*
WZ[j].KIN_ID	Kinematics ID of the tool	Integer	-	R/W*
WZ[j].TYPE	Tool type (0: Milling tool 1: Turning tool 2: Grinding tool)	Integer	-	R
WZ[j].TOOL_FIXED	Tool is alignable or fixed	Boolean	0 , 1	R/W*
WZ[j].SRK_ID	Cutter orientation of a turning tool	Integer	-	R/W*
WZ[j].S_MIN_SPEED	Minimum rotational speed (tool dynamic data)	Real	[rpm]	R/W*
WZ[j].S_MAX_SPEED	Maximum rotational speed (tool dynamic data)	Real	[rpm]	R/W*
WZ[j].S_MAX_ACC	Maximum acceleration (tool dynamic data)	Real	[°/s^2]	R/W*
WZ[j].SISTER_VALID	Validity flag of sister tool (TOOL ID)	Boolean	0 , 1	R/W*
WZ[j].SISTER	Number of valid sister tool	Integer	-	R/W*
WZ[j].VARIANT_VALID	Validity flag of variant tool (TOOL ID)	Boolean	0 , 1	R/W*
WZ[j].VARIANT	Number of valid variant tool	Integer	-	R/W*
WZ[j].GOBJECT[i].*	Access to the subelements of a specific graphical object **[as of Build V3.01.3018.00]** where <i>: 0 … 4	-	-	…
WZ[j].LINKPOINT.*	Access to the subelements of associated linkpoints **[as of Build V3.01.3018.00]**	-	-	...

S*: Write access to data of the internal tool management system as of CNC Build V3.1.3079.08

The variables "WZ_AKT", "T_AKT" and "D_AKT" permit access to the data of the **currently selected** tool. These variables are available both for an external tool management system and for the use of an internal tool table.

T_AKT	Number of the selected tool	Integer	-	R
D_AKT	Number of the selected tool compensation record	Integer	-	R

ℹ A write access always causes the temporary change of tool data as long as this function is selected. When a new tool is selected (Dxx) or deselected (D0), the changed data are lost.

Exception:

When an external tool management system is used, the so-called additional tool data (V.G.WZ_AKT.P[i]) is adopted and saved when a new tool is selected or a tool is deselected (P-CHAN-00103).

WZ_AKT.R	Radius of the selected tool	Real	[mm, inch]	R/W
WZ_AKT.L	Length of the selected tool	Real	[mm, inch]	R/W
WZ_AKT.P[i]	Free parameters of the selected tool where <i>: 0 … 59	Real	-	R/W
WZ_AKT.V[i] or WZ_AKT.V.X	Offset in axis <i> or "X" of the selected tool where <i>: 0 … 31	Real	[mm, inch]	R/W
WZ_AKT.ME	Measuring unit of radius, length and axis offsets of the selected tool, always supplies 0 (for [mm]) when a tool list is used, otherwise the variable has no significance	Boolean	0 , 1	R
WZ_AKT.OK	Validity flag of the selected tool; is always 1 since only data of valid tools are adopted. An error message is output if invalid tools are requested	Boolean	0 , 1	R/W*
WZ_AKT.SPDL_AX_NR	Logical axis number of the assigned spindle	Integer	-	R/W*
WZ_AKT.KIN_PARAM[i]	**CAUTION**: Note on write access: Value must be programmed in internal units. Kinematic parameters of the selected tool where <i>: 0 … 69	Real	[0.1 µm, 10^{-4}°]	R/W
WZ_AKT.KIN_ID	Kinematic ID of the selected tool	Integer	-	R/W*
WZ_AKT.TYPE	Tool type of the selected tool (0: Milling tool 1: Turning tool 2: Grinding tool)	Integer	-	R
WZ_AKT.TOOL_FIXED	Tool is alignable or fixed	Boolean	0 , 1	R/W

WZ_AKT.SRK_ID	Cutter orientation of the selected turning tool	Integer	-	R/W*
WZ_AKT.S_MIN_SPEED	Minimum rotational speed (tool dynamic data)	Real	[rpm]	R/W*
WZ_AKT.S_MAX_SPEED	Maximum rotational speed (tool dynamic data)	Real	[rpm]	R/W*
WZ_AKT.S_MAX_ACC	Maximum acceleration (tool dynamic data)	Real	[°/s^2]	R/W*
WZ_AKT.SISTER_VALID	Validity flag of sister tool (TOOL ID)	Boolean	0 , 1	R/W*
WZ_AKT.SISTER	Number of valid sister tool	Integer	-	R/W*
WZ_AKT.VARIANT_VALID	Validity flag of variant tool (TOOL ID)	Boolean	0 , 1	R/W*
WZ_AKT.VARIANT	Number of valid variant tool	Integer	-	R/W*
WZ_AKT.WEAR_RADIUS	Total radius wear with radius compensation (OTC) (sum of discrete + continuous wear)	Real	[mm, inch]	R

WZ_AKT.WEAR_RADIUS_CONT	Continuous radius wear with radius compensation (OTC)	Real	[mm, inch]	R
WZ_AKT.WEAR[i] or WZ_AKT.WEAR.X	Wear in axis <i> or "X" with length compensation (OTC) where < i>: 0 … 31	Real	[mm, inch]	R
WZ_AKT.WEAR_CONST	Wear constant (OTC)	Real	[0.1 µm/m]	R/W
WZ_AKT.GOBJECT[i].*	Access to the subelements of a specific graphic object (see FCT-C15) **[as of V3.01.3018.00]** where < i>: 0 … 4	-	-	-
WZ_AKT.LINKPOINT.*	Access to the subelements of associated link points (see FCT-C15) **[as of V3.01.3018.00]**	-	-	-

S*: Write access to these current tool data as of CNC Build V3.1.3079.08

NP[j].V[i] or NP[j].V.X	Zero offset of an axis <i> or "X" where <j>: 0 … 96 and <i>: 0 … 31 **CAUTION**: Write access causes permanent changes to internal zero offset data.	Real	[mm, inch]	R/W
NP[j].ALL	Address all axes of a zero offset group where <j>: 0 … 96 **CAUTION**: Write access causes permanent changes to internal zero offset data.	Real	[mm, inch]	R/W
NP_AKT.V[i] or NP_AKT.V.X	Currently (active) zero offset of an axis <i> or "X" where <i>: 0 … 31	Real	[mm, inch]	R/W
NP_AKT.ALL	Address (currently active) zero offsets of all axes	Real	[mm, inch]	R/W
NP_AKT.IDX	Index of the (currently active) zero offset group (e.g. 0 with G53, 1 with G54, etc.)	Integer	-	R
NP_DEFAULT	Index of zero offset group after start-up effective in initial state	Integer	-	R/W

The following variables can be used with a CS stack that was programmed with #(A,B)CS DEF or #(A,B)CS ON without ID specification. **[as of V3.01.3080.03]**

BCS_COUNT_DEF	Number of defined BCS	Integer	-	R
BCS_COUNT_FREE_DEF	Number of free places to define BCS	Integer	-	R
BCS_COUNT_ACTIVE	Number of active (linked) BCS	Integer	-	R
ACS_COUNT_DEF	Number of defined ACS	Integer	-	R
ACS_COUNT_FREE_DEF	Number of free places to define ACS	Integer	-	R
ACS_COUNT_ACTIVE	Number of active (linked) ACS	Integer	-	R
CS_COUNT_DEF	Number of defined CS	Integer	-	R
CS_COUNT_FREE_DEF	Number of free places to define CS	Integer	-	R
CS_COUNT_ACTIVE	Number of active (linked) CS	Integer	-	R

The following variables can be used with a CS stack that was programmed with #(A,B)CS ADD or #(A,B)CS SELECT. **[as of V3.01.3080.03]**

COORD_SYS_BCS_COUNT_DEF	Number of defined BCS	Integer	-	R
COORD_SYS_ACS_COUNT_DEF	Number of defined ACS	Integer	-	R
COORD_SYS_CS_COUNT_DEF	Number of defined CS	Integer	-	R

COORD_SYS_COUNT_ALL_DEF	Total number of defined coordinate systems (BCS+ACS+CS)	Integer	-	R
COORD_SYS_COUNT_ALL_ACTIVE	Total number of active coordinate systems after #CS SELECT	Integer	-	R

CNC_CHANNEL	Channel number	Integer	-	R
IPO_COUNT	System time counter	Integer	-	R
CS	Active tool compensation (1: RTCP 2: TLC 3: 2.5D)	Integer	-	R
TOOL_COMP	Current clamp position offset index	Integer	-	R
AX_LINK.NR	Number of current or last active coupling group	Integer	-	R
AX_LINK.ACTIVE	Is an axis coupling (#AX LINK ON) active? If active, then 1	Boolean	0 , 1	R
AX_LINK_GROUP[i].ACTIVE	Is a specific coupling group with the number < i>: 0 … 4 active? If active, then 1	Boolean	0 , 1	R
ROT_ACTIVE	Contour rotation (#ROTATION...) active? If active, then 1	Boolean	0 , 1	R
ROT_ANGLE	Angle of contour rotation	Real	[°]	R
ROT_CENTER1	Offset of 1st main axis to rotation point with contour rotation	Real	[mm, inch]	R
ROT_CENTER2	Offset of 2nd main axis to rotation point with contour rotation	Real	[mm, inch]	R
TIMER[i]	Counter value of timer with number <i>: 0 … 127	Integer	[ms]	R
PROG_ABS	Measuring system, 0: G91 active 1: G90 active	Boolean	0 , 1	R
ACT_PLANE	G number of current active plane: 17: G17 active, 18: G18 active, 19: G19 active	Integer	-	R
CNC_RELEASE	Build number of CNC version (old syntax)	Integer	-	R
CNC_VERSION.MAJOR	Major version of CNC **[as of V2.11.2800]** (e.g. **2** with **V2.11.2807.42**)	Integer	-	R
CNC_VERSION.MINOR	Minor version of CNC **[as of V2.11.2800]** (e.g. **11** with **V2.11.2807.42**))	Integer	-	R
CNC_VERSION.BUILD	Build number of CNC version **[as of V2.11.2800]** (e.g. **2807** with **V2.11.2807.42**))	Integer	-	R
CNC_VERSION.PATCH	Patch number of CNC version **[as of V2.11.2800]** (e.g.. **42** with V2.11.2807.**42**)	Integer	-	R
WCS_POSLIMIT_1 WCS_POSLIMIT_2 WCS_POSLIMIT_3	Motion limits in the main axes in WCS. Only practical in conjunction with the NC command #GET WCS POSLIMIT [▶ 710]	Real	[mm, inch]	R

SIGNAL_READ	Read a signal without waiting **[as of V2.11.2820.00]**	Boolean	0 , 1	R
	1: Signal present and read,			
	0: Signal not present			
	Only practical in conjunction with the NC command #SIGNAL READ [▶ 378]			
EXECUTION_MODE	Execution mode in the NC program job, for meaning of value see HLI description	Integer	-	R

The following variables permit access to the data of kinematics defined in the channel parameter list.

In versions up to V2.11.28xx and for single step transformations from V3.00, access uses V.G.KIN[j].* (where j:= kinematic ID).

For multi-step transformations (as of Build V3.00.3012.00), access uses V.G.KIN_STEP[i].ID[j].*

(where i:= transformation step 0 or 1 and j:= kinematic ID).

NOTICE
Write access causes a permanent change to internal channel parameter data. Value must be programmed in internal units.
The changes are retained until the next start-up or updating of the channel parameter list.

(single-stage up to V3.00.3017.01): **V.G.KIN[j].*** (multi-stage up to V3.00.3018.00): **V.G.KIN_STEP[i].ID[j].***	Meaning	Data type	Unit for input/ output	R/W*
PARAM[k]	Kinematic parameter where <k>:= 0 ... 69	Real	[0.1 µm, $10^{-4°}$]	R/W

* Permitted access: R/W = Read/Write

In addition, the following variables listed are provided for access to the data of the **Universal Kinematic** (ID 91, [FCT-C27])

ZERO_ORIENTATION [k]	Initial orientation of the tool where <k>:= 0, 1, 2 (X, Y, Z)	Real	-	R/W
ZERO_POSITION[k]	Initial position of the tool where <k>:= 0, 1, 2 (X, Y, Z)	Real	[0.1 µm, $10^{-4}°$]	R/W
PROGRAMMING_MODE	Programming mode, see P-CHAN-00112	Integer	-	R/W
RTCP	Angle transformation	Boolean	-	R/W
NUMBER_OF_AXES	Number of axes in kinematic chain	Integer	-	R/W
AXIS[k].*	Axis of kinematic chain where <k>:= 0 … 5	-	-	-
AXIS[k].TYPE	Axis type (1: translatory, 2: rotary)	Integer	-	R/W
AXIS[k].ORIENTATION[l]	Orientation vector of axis (direction) where <l>:= 0, 1, 2 (X, Y, Z)	Real	-	R/W
AXIS[k].POINT[l]	Interpolation vertex on axis where <l>:= 0, 1, 2 (X, Y, Z)	Real	[0.1 µm, $10^{-4}°$]	R/W
CHAIN[k]	Description of axis sequence in the kinematic chain where <k>:= 0 … 5	Integer	-	R/W

The 3 variables below permit read access to the currently selected kinematic ID:

V.G.KIN_ID	**Currently selected kinematic ID for single-step transformations**	**Integer**	**-**	**R**
V.G.KIN_ID_STEP[0]	Currently selected kinematic of the first step with multi-step transformations **[as of Build V3.00.3018.00]**	Integer	-	R
V.G.KIN_ID_STEP[1]	Currently selected kinematic of the second step with multi-step transformations **[as of Build V3.00.3018.00]**	Integer	-	R
TOTAL_KIN_OFFSET[i]	Effective total offset of the selected kinematics, consisting of the sum of kinematic offsets of the active tool and the kinematic offsets in the channel parameter list where <i>:= 0 … 69 **[as of V2.11.2024.00]**	Real	[0.1 µm, $10^{-4}°$]	R

MAIN_FILE_NAME	File name of the NC main program. With MDI or manual block the variable supplies the filename "-".	String	-	R
MAIN_PROG_NAME	Name (%...) of the NC main program	String	-	R
MAIN_PROG_NR	Number of the NC main program if the program name is a digit	Integer	-	R
FILE_NAME	File name of the currently active NC program	String	-	R
PROG_NAME	Name (%...) of the currently active NC program	String	-	R
PROG_NR	Number of the currently active NC program if the NC program name is a digit	Integer	-	R
PROG_LEVEL	Program level of the currently active NC program: 1: Main program ≥ 2: Local or global subroutine	Integer	-	R
PATH_NR	Logical path number of the currently active NC main or global subroutine contained in the start–up list (P-STUP-00019)	Integer	-	R
FILE_OFFSET	File position of start of the NC block where V.G.FILE_OFFSET is programmed	Integer	-	R
MIRROR_ACTIVE	Mirroring (G351 X... or G21, G22, G23) active? If active, then 1	Boolean	0 , 1	R
BLOCK_SEARCH_ACTIVE	Block search active? If active, then 1	Boolean	0 , 1	R
TRAFO_ACTIVE	Kinematic transformation (#TRAFO...) active? If active, then 1	Boolean	0 , 1	R
MCS_ACTIVE	Temporary transition to the machine axis coordinate system (#MCS...) active? If active, then 1	Boolean	0 , 1	R
HSC_ACTIVE	Free-form surface machining (#HSC...) or spline interpolation (#SPLINE... or G151) active? If active, then 1	Boolean	0 , 1	R
HSC_SURFACE_ACTIVE	Is free-form surface machining (#HSC [SURFACE...]) active? If active, then 1	Boolean	0 , 1	R
CONT_MODE_ACTIVE	Polynomial contouring (G261) active? If active, then variable supplies the value of the current contouring type: 3 with contouring type DIST 4 with contouring type DEV 5 with contouring type POS 6 with contouring type DIST_SOFT 6 with contouring type DIST_MASTER 7 with contouring type PTP 0 with G260	Integer	-	R

TRC_ACTIVE	Tool radius compensation (G41, G42) active? If active, then 1	Boolean	0 , 1	R
OTC_ACTIVE	Online tool compensation (#OTC…) active? If active, then 1	Boolean	0 , 1	R
CAXTRACK_ACTIVE	Is C axis tracking (#CAXTRACK…) active? If active, then 1	Boolean	0 , 1	R
CYCLE_ACTIVE	Is the current program level a cycle? If the current program was called with L or LL CYCLE... or G8xx with parameter transfer, then 1	Boolean	0 , 1	R

@P[i].VALID	Is transfer parameter programmed in the cycle call [▶ 203] or subroutine call programmed with G8xx? 0: Parameter is not programmed 1: Parameter is programmed Where <i>: = 1 ... 200 the number of the parameter subjected to a validity check is specified. (E.g. 1 for @P1 or on request of the first parameter when called with G8xx). NOTE for read access: The variable can only be used inside a cycle or in global subroutines G8xx with parameter transfer.	Boolean	0 , 1	R
TIME_STAMP	Current time stamp <date time> in format <*DD.MM.YYYY hh:mm:ss:zzz*>: D: Day, two-digit M: Month, two-digit Y: Year, four-digit h: Hours, two-digit m: Minutes, two-digit s: Seconds, two-digit z: Milliseconds, three-digit (e.g. 16.06.2015 14:08:10:123) **[as of V2.10.1507.02]**	String	-	R
TIME_STAMP_FILE_NAME	Current time stamp <date time> in the format as per ISO 6801:2004 <*YYYYMMDDThhmmsszzz*> without '.' and ':' e.g. for use in names of log files where: Y: Year, always four-digit M: Month, two-digit D: Day, two-digit h: Hours, two-digit m: Minutes, two-digit s: Seconds, two-digit z: Milliseconds, three-digit (e.g. 20150616T140810123) **[as of V2.11.2024.03]**	String	-	R
LIFT_ACTIVE	Lifting/lowering an axis (X[LIFT...]) active? If active, then 1	Boolean	0 , 1	R

The following variables permit access to data of the M/H functions defined in the channel parameters.

> ℹ️ When M/H functions are defined with pre-output function (MEP_SVS, MET_SVS), the following must be taken into consideration:
>
> If write access (S) to the pre-output values and the synchronisation modes of the M/H functions do not match, the synchronisation mode is implicitly adapted.
>
> If read access (L) to the pre-output values and the synchronisation modes of the M/H functions do not match, an error message is output.

NOTICE				
The changes are retained until the next start-up or updating of the channel parameter list.				
M_FCT[i].SYNCH	Synchronisation type of an M function <i> defined in the P-CHAN-00041 channel parameters. The value of the synchronisation type is specified in hexadecimal notation 'Hex' [▶ 22].	Integer	-	R/W
M_FCT[i].PRE_OUTP_PATH	Pre-output path *m_pre_output* of an M function <i> of type MEP_SVS defined in the P-CHAN-00070 channel parameters	Real	[mm, inch]	R/W
M_FCT[i].PRE_OUTP_TIME	Pre-output time *m_pre_output* of an M function <i> of type MET_SVS in the P-CHAN-00070 channel parameters	Real	[s]	R/W
H_FCT[i].SYNCH	Synchronisation type of an H function <i> defined in the P-CHAN-00027 channel parameters. >. The Synchronisation type of H functions (P-CHAN-00027) is specified in hexadecimal notation 'Hex' [▶ 22].	Integer	-	R/W
H_FCT[i].PRE_OUTP_PATH	Pre-output path *h_pre_output* of an H function <i> of type MEP_SVS defined in the P-CHAN-00107 channel parameters	Real	[mm, inch]	R/W
H_FCT[i].PRE_OUTP_TIME	Pre-output time *h_pre_output* of an H function <i> of type MET_SVS in the P-CHAN-00107 channel parameters	Real	[s]	R/W

The "V.G.SPEED_LIMIT" variables permit access to parameters pre-assigned in the channel parameters for speed limit look ahead [1] [▶ 786]-6.

i
With write access the changed parameterisation is output to the channel. A possibly active motion is interrupted.

If write access (S) to these values and the unit of the distance parameters do not match, the unit is implicitly adapted.

If read access (L) to these values and the unit of the distance parameters do not match, an error message is output.

NOTICE				
The changes are retained up to program end or RESET. After program start the settings of the channel parameters are valid again.				

SPEED_LIMIT.ENABLE	Selecting/deselecting speed limit look ahead (0: deselect 1: select)	Boolean	0 , 1	R/W
SPEED_LIMIT.VEL_LIMIT	Definition of speed limit value	Integer	[%]	R/W
SPEED_LIMIT.TIME	Defining the unit of the distances to and from the "corner" (0: path 1: time)	Boolean	0 , 1	R/W
SPEED_LIMIT.DIST_TO_CORNER	Path distance to "corner"	Real	[mm, inch]	R/W
SPEED_LIMIT.DIST_FROM_CORNER	Path distance from "corner"	Real	[mm, inch]	R/W
SPEED_LIMIT.TIME_TO_CORNER	Time distance to "corner"	Real	[s]	R/W
SPEED_LIMIT.TIME_FROM_CORNER	Time distance from "corner"	Real	[s]	R/W
SPEED_LIMIT.OVERRIDE_WEIGHT	Weighting of speed limit value (0: no weighting 1: weighting by override)	Boolean	0 , 1	R/W

MAX_NC_BLOCKS_AHEAD	Block-related decoder block ahead limiting: all NC blocks	Integer	-	R/W
MAX_MOTION_BLOCKS_AHEAD	Block-related decoder block ahead limiting: NC motion blocks only	Integer	-	R/W
MAX_TIME_AHEAD	Time-related decoder block ahead limiting	Real	[s]	R/W

CIRCLE_CP_ABS	Circle centre point definition, 0: G162 active 1: G161 active	Boolean	0 , 1	R
CIRCLE_CPC_ACTIVE	Is circle centre point correction (G165) active? If active, then 1 0 with G164	Boolean	0 , 1	R

SEGMENTATION_ACTIVE	Is segmentation (#SEGMENTATION...) active? If active, then 1	Boolean	0 , 1	R
STROKE_DEF_ACTIVE	Is definition of a stroke motion (#STROKE DEF…) active? If active, then 1	Boolean	0 , 1	R
PUNCH_ACTIVE	Punching (#PUNCH...) active? If active, then 1	Boolean	0 , 1	R
NIBBLE_ACTIVE	Nibbling (#NIBBLE...) active? If active, then 1	Boolean	0 , 1	R
EDM_ACTIVE	EDM machining (#EDM...) active? If active, then 1 **[as of CNC Build V3.1.3019]**	Boolean	0 , 1	R
FRICTION_ACTIVE	Recording friction compensation values (#FRICTION ON […] active? If active, then 1 **[as of CNC Build V2.11.2022.05]**	Boolean	0 , 1	R
VECTOR_OFFSET_ACTIVE	Is cutting edge compensation (#VECTOR OFFSET...) active? If active, then 1	Boolean	0 , 1	R
DIAMETER_PROG_ACTIVE	Is diameter programming (G51) active? If active, then 1 **[as of CNC Build V2.11.2051.00]**	Boolean	0 , 1	R
DIAMETER_PROG.ZERO_OFFSET	Supplies the value of P-CHAN-00091 if the following applies: - Diameter programming (G51) is active - Face turning axis is configured **[as of CNC Build V2.11.2051.00]**	Boolean	0 , 1	R

FILE_EXIST	Result of request with #FILE EXIST [▶ 417] If file check is positive, then 1	Boolean	0 , 1	R

MEAS_TYPE	Value of currently active measuring type [▶ 87] **[as of V2.11.2022.03]**	Integer	-	R

TRACK_CS.X TRACK_CS.Y TRACK_CS.Z TRACK_CS.A TRACK_CS.B TRACK_CS.C	Values of master position with dynamic coordinate system (see [FCT-C30]).	REAL	[mm, inch]	R

The following V.G. variables permit the readout of the number of all write accesses to the contents of each list since controller start-up. [available as of Build V2.11.2037.03]

CHAN_LIST_INVOKE	Channel parameter list	Integer	-	R
CLAMP_LIST_INVOKE	List of clamping offsets	Integer	-	R
TOOL_LIST_INVOKE	Tool list	Integer	-	R
VE_VAR_LIST_INVOKE	List of external variables	Integer	-	R
ZERO_POINT_LIST_INVOKE	Zero offset list	Integer	-	R

The number of write accesses to individual zero offset groups is also possible [available as of V2.11.2037.03].

NP[j].INVOKE_COUNT	Number of all write accesses to a zero offset since controller start-up where <j>: 0 … 96	Integer	-	R
NP_AKT. INVOKE_COUNT	Number of all write accesses to current zero offset group since controller start-up	Integer	-	R

Read the number of write accesses to a zero offset group

```
N010 P1 = V.G.NP[1].INVOKE_COUNT ; Read counter

N020 V.G.NP[1].V.X = 15
N030 V.G.NP[1].V.X = 16

N040 $IF P1 +2!= V.G.NP[1].INVOKE_COUNT ; Read counter again
N050   #ERROR [ID1]
N060 $ENDIF

N070 G54
N080 $IF V.G.NP_AKT.INVOKE_COUNT != V.G.NP[1].INVOKE_COUNT
N090   #ERROR [ID2]
N100 $ENDIF

N110 G53
N120 M30
```

The following V.G. variables for TCP velocity limiting are available as of V3.1.3079.26

LIMIT.KIN[i].TOOL.LENGTH	Tool length where \<i\>:= 0, 1 index of the configured kinematic	Real	[mm]	R/W
LIMIT.KIN[i].TOOL.KIN_PARAM[j]	Kinematic parameter of the tool where \<j\>:= 0..69 Index of the kinematic parameter	Real	[0.1 µm,10-4°]	R/W

14.3.1   Versioning of NC programs

NC programs can be provided with a version number using the V.G. variable V.G.PROG_VERSION.

---

i **The complete version number must always have the format "<Major>.<Minor>.<Build>.<Patch>".**

If a different format is used, the error ID 22015 is output.

The complete version number is composed of the following:

"Complete"="<Major>.<Minor>.<Build>.<Patch>"

For example, the complete version number or individual elements of the version number can be defined.

```
V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
V.G.PROG_VERSION.PATCH = 4
```

PROG_VERSION.MAJOR	Major version of the NC program (e.g. **4** with **4**.1.2.3)	UNS08	-	R/W
PROG_VERSION.MINOR	Minor version of the NC program (e.g. **1** with 4.**1**.2.3)	UNS08	-	R/W
PROG_VERSION.BUILD	Build version of the NC program (e.g. **2** with 4.1.**2**.3)	UNS08	-	R/W
PROG_VERSION.PATCH	Patch version of the NC program (e.g. **3** with 4.1.2.**3**)	UNS08	-	R/W
PROG_VERSION.COMPLETE	Major.Minor.Build.Patch	STRING	-	R/W

The version information is inherited to called subroutines. A self-defined version number can also be assigned in subroutines and it is then inherited to its called subroutines.

Versioning of NC programs

Example 1

```
%L UP_1
N110 V.G.PROG_VERSION.COMPLETE = "5.1.2.3"
N120 #MSG ["Version UP_1: %s", V.G.PROG_VERSION.COMPLETE]
N130 M17

%MAIN
N010 V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
N20 LL UP_1
N30 #MSG ["Version Main: %s", V.G.PROG_VERSION.COMPLETE]
N040 M30
```

The following is output:

Version UP_1: 5.1.2.3

Version Main: 4.1.2.3

Example 2

```
%L UP_1
( --- no separate version specified ---)
N120 #MSG ["Version UP_1: %s", V.G.PROG_VERSION.COMPLETE]
N130 M17

%MAIN
N010 V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
N20 LL UP_1
N30 #MSG ["Version Main: %s", V.G.PROG_VERSION.COMPLETE]
N040 M30
```

The following is output:

Version UP_1: 4.1.2.3

Version Main: 4.1.2.3

14.4   Self-defined variables (#VAR, #ENDVAR, #DELETE)

Self-defined variables are created and possibly initialised in the NC main program or subroutine **after** the program name in a declaration block which starts with **#VAR** and ends with **#ENDVAR**.

ℹ️ Self-defined variables have prefix identifier V.P. , V.S. and V.L. They may be assigned values in REAL format. As of Build V2.11.2032.08 self-defined variables are available with the prefix identifier V.CYC. [▶ 577].

Syntax of Creating a declaration block for self-defined variables:

#VAR	Start of declaration block
:	
:	Declaration and initialisation part
:	
#ENDVAR	End of declaration block

Due to the introduction of <u>V.CYC. variables [▶ 577]</u>, the following extensions are available for all types of self-defined variables (as of: V2.11.2032.08, V2.11.2832.00, V3.1.3079.41, V3.1.3107.30)

Besides specifying the variable name, the declaration also includes a definition of the data type and an initial value. The variable is assigned the initial value of each data type without assigning an initial value. If a data type is not specified, the variable is always created in REAL format.

Syntax of the declaration and initialisation:

V.P | S | L | CYC.<*Name*> **:** <*Data_type*> **=** <*Initial_value*> **|** **"**<*Initial_string*>**"**

<name>	User-defined name of the self-defined variable
<data_type>	Data type identifiers (optional):
	BOOLEAN
	SGN08, UNS08
	SGN16, UNS16
	SGN32, UNS32
	REAL
	STRING[i] where <i>:= 0..126
<Initial_value>, *"<Initial_string>"*	Initial value or string of the variable (optional) depending on data type

```
Create variables with and without type declaration
%test_var_def_1
:
#VAR
  V.P.VAR_1
  V.P.VAR_2 = 10.67
  V.P.VAR_3 : UNS32 = 10
  V.L.NAME_1 : STRING[20] = "BASEPLATE"
  V.L.VAR_1 : REAL = 23.45
  V.L.VAR_2 : SGN08
#ENDVAR
```

```
For a better overview, initialisation of a variable array can be written across several NC lines
using the "\" character.
%test_var_def_2
:
#VAR
  V.P.ARRAY_1[3][6] = [10,11,12,13,14,15, \
                       20,21,22,23,24,25, \
                       30,31,32,33,34,35 ]
  V.L.MY_ARRAY[3][6] = [10,11,12,13,14,15, 20,21,22,23,24,25, 30,31,32,33,34,35]
#ENDVAR
```

> **i** Access to array variables starts with index 0. In the above example, access V.L.MY_ARRAY[0][5]
> then supplies the value 15.

Self-defined variables and variable arrays can also be deleted in the NC program. The #DELETE command
is provided for this.

```
Syntax:
```
#DELETE V.<*name*> {**, V.**<*name*>}

```
#DELETE V.P.ARRAY_1, V.L.MY_ARRAY, V.P.VAR_1, V.L.VAR_1, V.S.VAR_1
```

In addition, the SIZEOF and EXIST functions are provided (see Section <u>Arithmetical expressions <expr></u> [▶ 32]) to determine the dimensional size of variable arrays and check for the existence of self-defined variables.

The EXIST request for a self-defined V.S. array variable (with any valid index) checks whether this variable was already defined in a previous NC program or if this variable must still be defined:

```
…
N10 $IF EXIST[V.S.EXAMPLE[0]] == TRUE
N20  V.S.EXAMPLE[2] = 10  ;assign a value to V.S. variable[2]
N30 $ELSE
N40   #VAR
N50     V.S.EXAMPLE[5] = [1,2,3,4,5 ]
N60   #ENDVAR
N70 $ENDIF
…
M30
```

14.4.1 Global, valid up to end of main program (V.P.)

The identifier "V.P. ...." permits the definition of self-defined variables which are detected (global) at the current program level and at all other program levels but are not valid after main program end. V.P. variables may be assigned values in REAL format. As of Build V2.11.2032.08 <u>declarations of other data types [▶ 570]</u> with initial values are possible.

V.P. variables can also be created as multi-dimensional arrays. A maximum of 4 dimensions is possible, e.g. V.P.TEST[1][2][3][4].

```
Syntax:
```
V.P.<FREE_DEF**>**   global variable not valid after (main) program end

<FREE_DEF>   User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

Global variable not valid after (main) program end (V.P.)
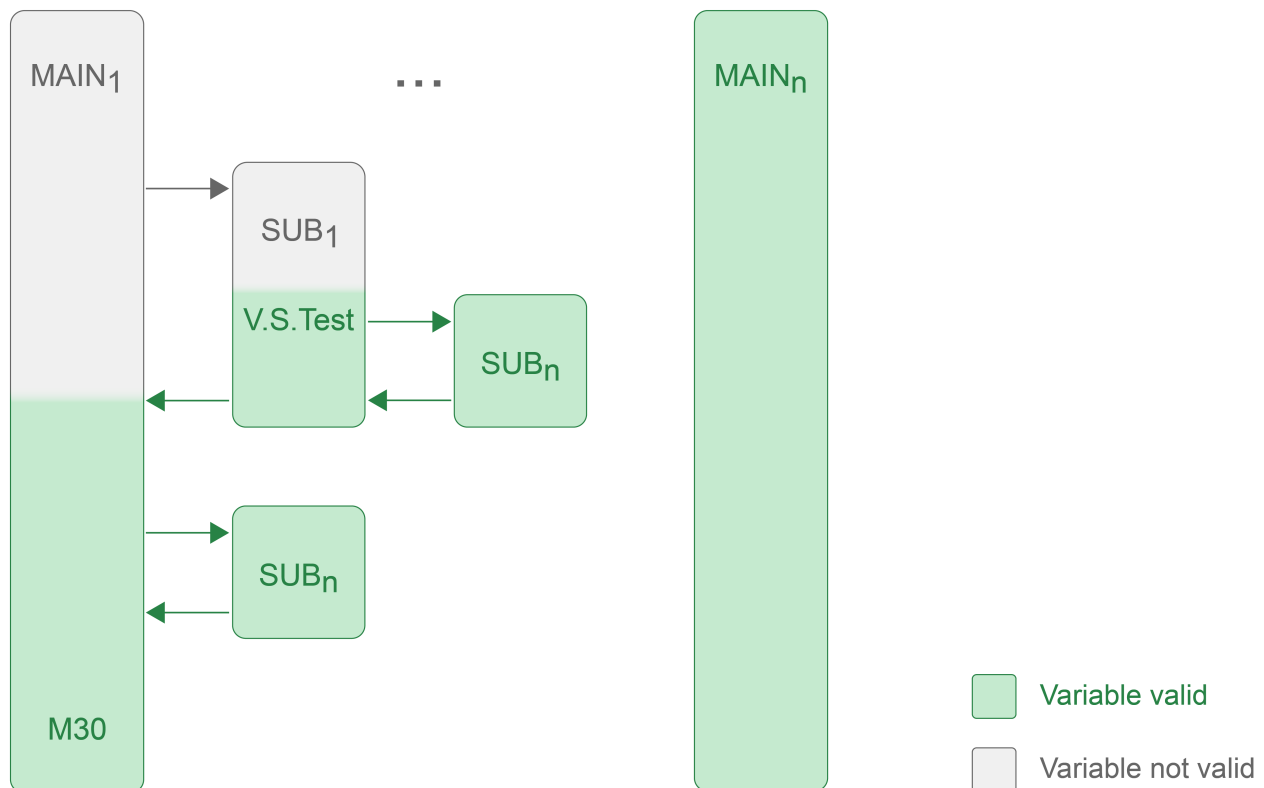
Create V.P. variables with and without assigning data types and initial values.

```
:
#VAR
  V.P.VAR_1                          ;REAL64, 0.0
  V.P.VAR_2 : UNS32 = 200            ;UNS32, 200
  V.P.VAR_3 : REAL64 = 11.34         ;REAL64, 11.34
  V.P.VAR_4 : STRING[20] = "END_OK"  ;STRING, END_OK
  V.P.VAR_5 = 20                     ;REAL64, 20.0
#ENDVAR
:
XV.P.VAR_5        ;X20.0
:
```

The maximum number of self-defined V.P. variables is fixed [6] [▶ 786]-6.21. At program start, all names and values of V.P. variables are deleted.

14.4.2 Global, valid throughout main program (V.S.)

The code "V.S. ...." permits the definition of self-defined variables which retain the same names and the last values assigned at all program levels and all following (main) HC programs. After RESET, these variables also remain valid. The values of the variables can only be changed by overwriting. The variables themselves can only be deleted by #DELETE or a controller restart. V.S. variables may be assigned values in REAL format. As of Build V2.11.2032.08 declarations of other data types [▶ 570] with initial values are possible.

```
Syntax:
```

V.P.<FREE_DEF>   global variable not valid after (main) program end

<FREE_DEF>   User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

Global, valid (main) program global (V.S.)

Create V.S. variables with and without assigning data types and initial values.

```
:
#VAR
  V.S.VAR1                    ;REAL64, 0.0
  V.S.VAR2 : UNS32 = 200      ;UNS32, 200
  V.S.VAR3 : REAL64 = 11.34   ;REAL64, 11.34
  V.S.VAR4 : BOOLEAN          ;BOOLEAN, FALSE or 0
  V.S.VAR[5] = [5,10,10,15,20]  ;Array with REAL64 values
#ENDVAR
:
XV.S.VAR[4]        ;X20.0
:
```

The maximum number of self-defined V.S. variables is fixed [6] [▶ 786]-6.22.

14.4.3   Local, valid throughout subroutine (V.L.)

The code "V.L. ...." permits the definition of self-defined variables which are valid local at the current program level and in directly called subroutines. They are deleted when the program level in which they were created is left (return).

A V.L. variable may be redefined with the same name at a lower program level and assigned a new value. This value is valid until this program level is left (return). The V.L. variable with the same name then has the original value.

V.L. variables may be assigned values in REAL format. As of Build V2.11.2032.08 <u>declarations of other data types [▶ 570]</u> with initial values are possible.

V.L. variables can also be created as multi-dimensional arrays. A maximum of 4 dimensions is possible, e.g. V.L.TEST[1][2][3][4].

```
Syntax:
```

V.P.<FREE_DEF>   local global variable in programs and subroutines

<FREE_DEF>   User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

Local valid after subroutine end (V.L.)
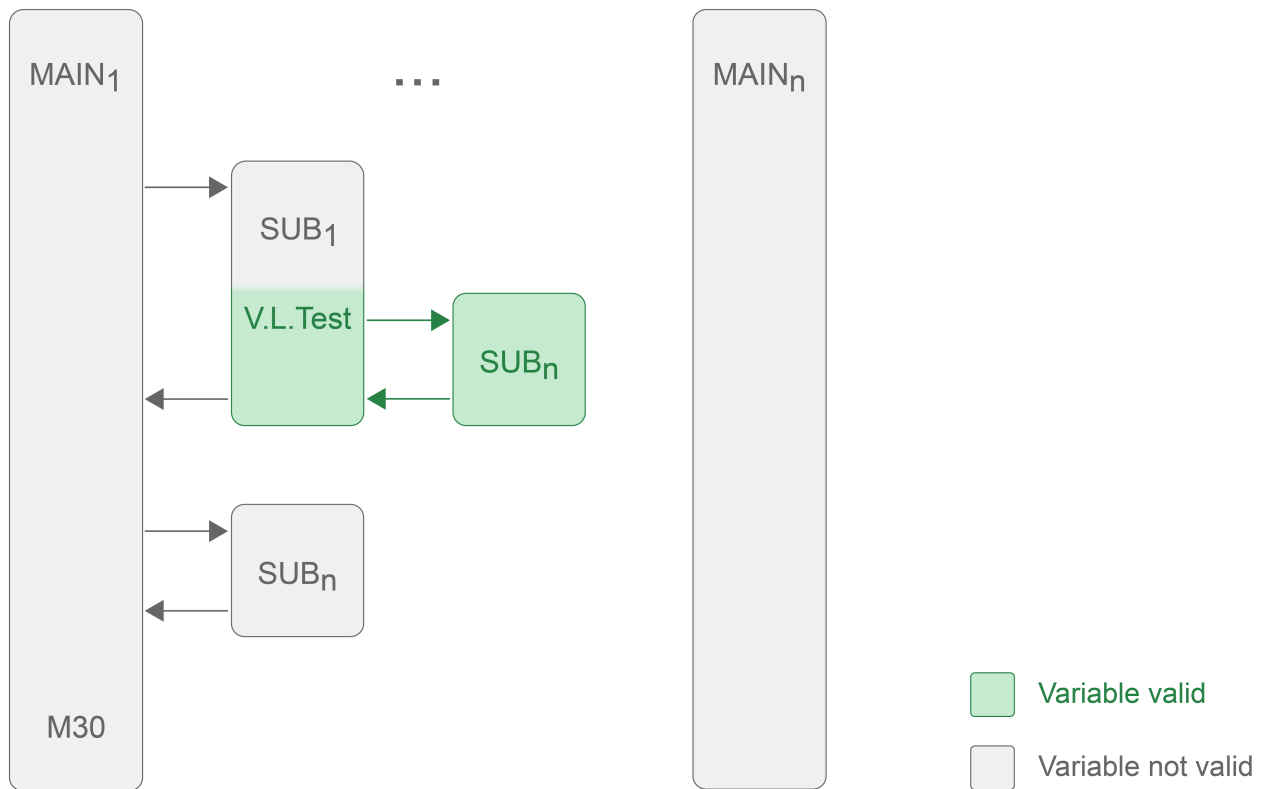
Create V.L. variables with and without assigning data types and initial values.

```
:
#VAR
  V.L.LOC_VAR1                    ;REAL64, 0.0
  V.L.LOC_VAR2 : UNS32 = 200      ;UNS32, 200
  V.L.LOC_VAR3 : REAL64 = 11.34   ;REAL64, 11.34
  V.L.LOC_VAR4 : BOOLEAN          ;BOOLEAN, FALSE or 0
  V.L.LOC_VAR5 = 10               ;REAL64, 10.0
#ENDVAR
:
XV.L.LOC_VAR5         ;X10.0
:
```

The maximum number of self-defined V.L. variables is fixed [6] [▶ 786]-6.23. At the start of the program, all names and values of those V.L. variables are deleted.

14.4.4   Cycle variables (V.CYC.)

i   **This function is available as of CNC Build V2.11.2032.08**

i   A precondition for using V.CYC. variables is that memory space must be reserved via the channel parameter P-CHAN-00418.

The identifier "V.CYC. ..." addresses self-defined variables that must be used by preference within cycle programs. In addition, V.CYC. variables can also be used in standard main programs and subroutines. Besides specifying the variable name, the statement also includes a definition of the data type. The variables are valid as from their declaration in the current program level and in all other directly called program levels (subroutines). They are deleted when the program level in which they were created is left (return) (see Validity and visibility [▶ 578]).

V.CYC. variables also have the option of creating multi-dimensional arrays. A maximum of 4 dimensions is possible, e.g. V.CYC.Test[1][2][3][4].

Syntax:

V.CYC.<FREE_DEF>   Cycle/program-specific variable

<FREE_DEF>   User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

V.CYC statement

```
%CYCLE_TEST.cyc
P1 = 3          ;first index of the array
P2 = 2          ;second index of the array
P3 = 10         ;default maximum string length
#VAR
  V.CYC.TEST_A[P1][P2] : STRING[P3]
  V.CYC.TEST_B : STRING[P3] = "TEXT"
  V.CYC.TEST_C : REAL64 = 1.0
#ENDVAR
:
M30
```

14.4.4.1 Validity and visibility

A V.CYC. variable is visible and usable starting with the declaration at the program level and in the program levels below it.
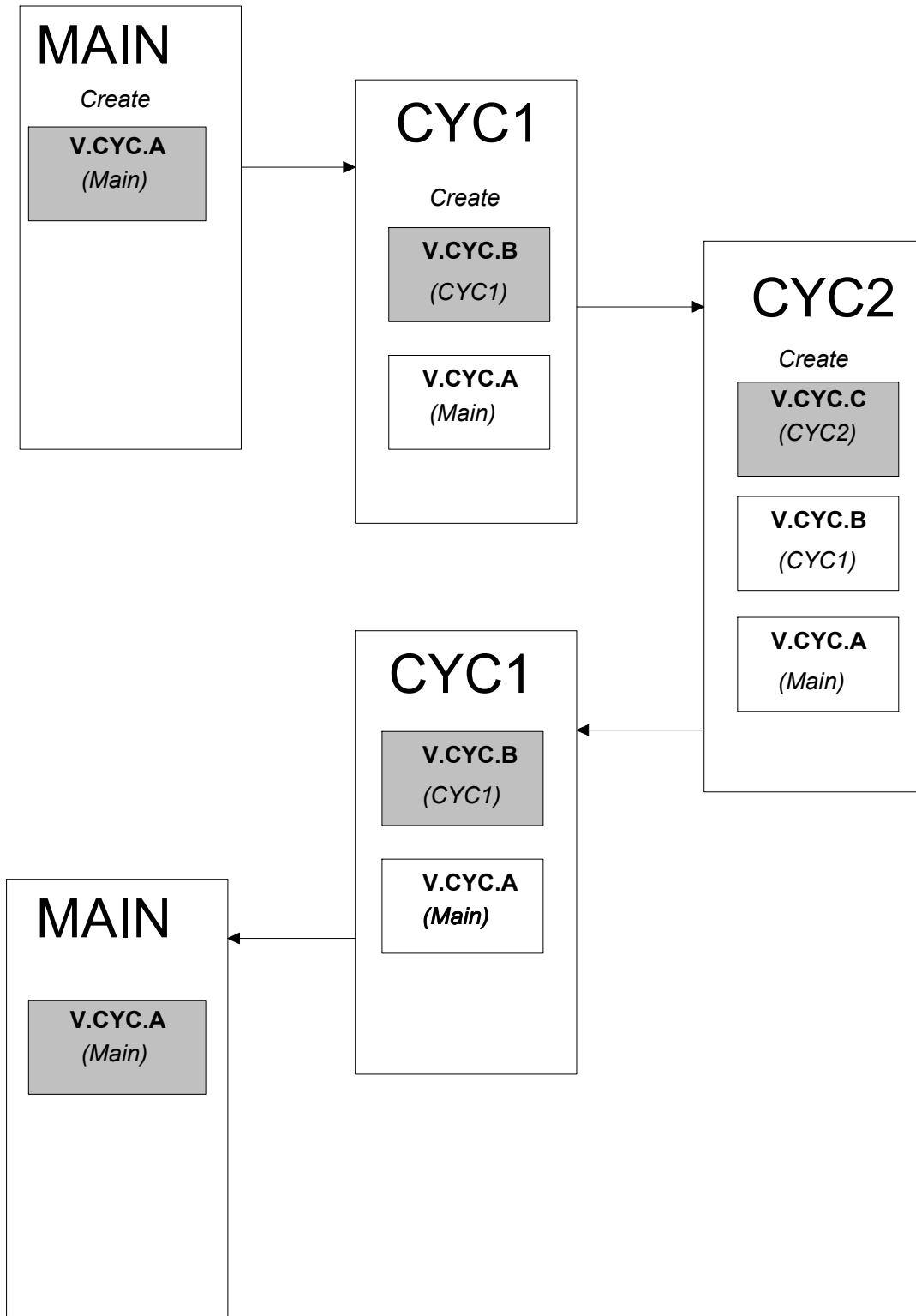

Fig. 153: Validity of self-defined V.CYC. variables

If a V.CYC: variable is created with the same name in a lower program level, the 'most local" declaration is visible and usable.
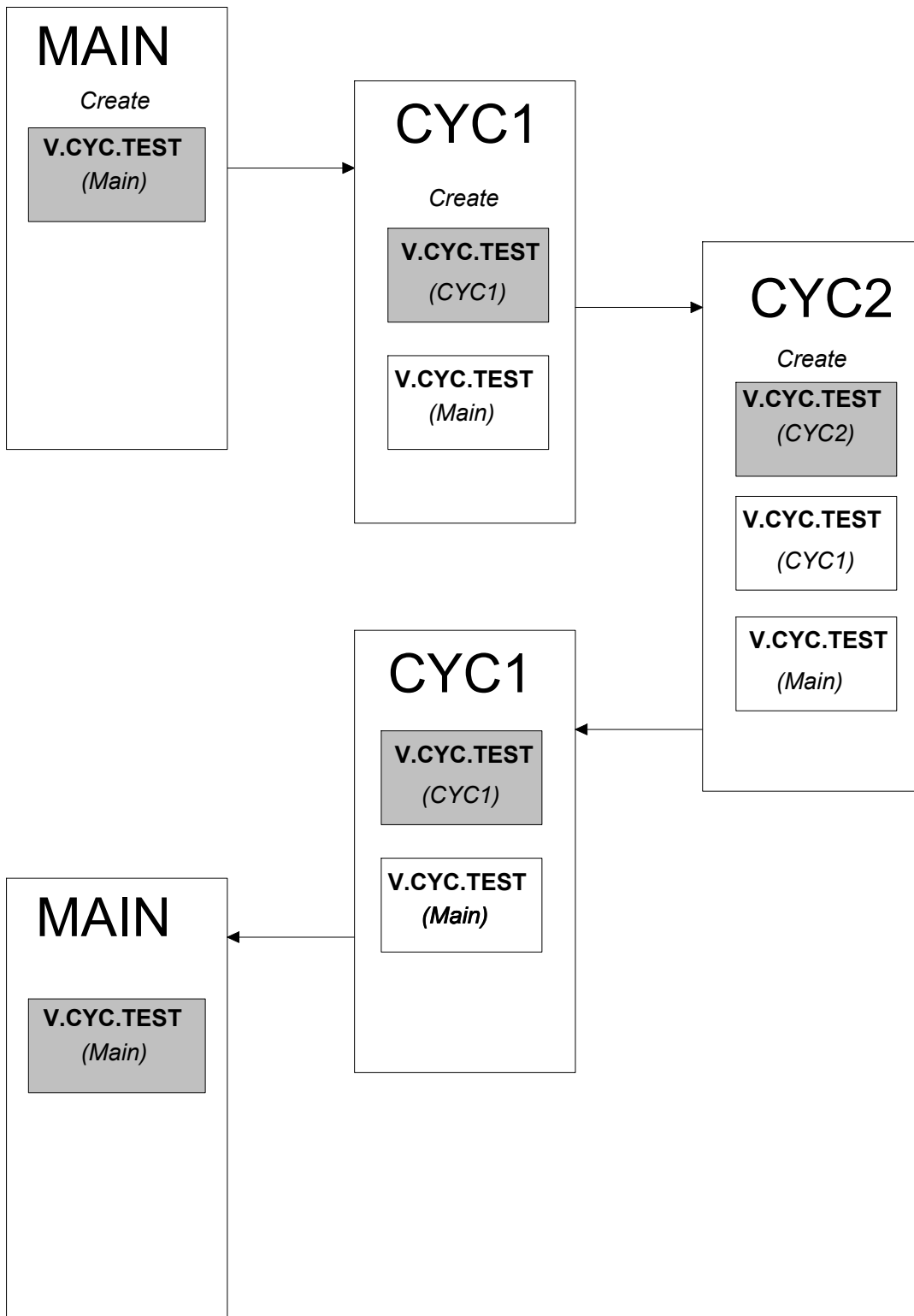
Fig. 154: Validity of V.CYC. variables of the same name

14.4.4.2 Delete V.CYC. variables

A V.CYC. variable is automatically deleted when the program exists the cycle or the program level in which the variable was created.

Alternatively, a V.CYC. variable can also be deleted by the NC command #DELETE in the program level in which it was created.

For example,

#DELETE V.CYC.MyVar

ⓘ #DELETE can only delete V.CYC. variables in the program level in which it was created.

Deleting a non existing V.CYC. variable is ignored. No warning or error message is output.

If an attempt is made in a program level to delete an existing V.CYC. variable which was created in a higher program level, the error P-ERR-20933 is output.

14.5   External variables (V.E) (#INIT V.E.)

The command "V.E. ...." writes to external addresses in the NC program and/or reads from external addresses. This is permitted by direct memory-linked communication between the NC channel and external users, typically the PLC.

Access from the NC channel can be executed synchronously by the interpolator or asynchronously by the decoder.

> **i**   **Reading a synchronous V.E. variable causes the NC channel to be flushed.**
>
> This is not permitted when TRC, polynomial contouring or HSC mode are active.

For more information on configuration and parameterisation, see the descriptions of external variables in the documentation [8] [▶ 786].

External variables (V.E)

```
N100 $IF V.E.EXT1 >= 100            (corr. to the value of V.E.EXT1)
                                    (branched to)
                                    (various cases)
N110 G01 X100 Y100 F1000

N120 $ELSE
N130 G01 X100 YV.E.EXT1 F1000      (linear interpolation in Y direction)
                                    (with the value of V.E.EXT1)
N140 $ENDIF
N150 N150 = V.A.ABS.X          (The external variable is assigned)
                                (absolute X coordinates)
N160 G01 X                 (linear interpolation in X direction)
                                (the value of V.E.EXT2)
```

After controller start-up, the configured V.E. variables are initialised with zero.

Then V.E. variables can be re-initialised in the NC program using the #INIT command. The command can be followed by one or several V.E. variables which are initialised completely. Besides individual V.E. variables, complete V.E. arrays, V.E. structures and subelements of V.E. structures can also be initialised.

```
Syntax:
```
#INIT V.E.<*name*> {**, V.E.**<*name*>}

NOTICE
Access rights:
If a variable only has read access rights, it cannot be initialised with the #INIT command. The same applies to V.E. structures if they contain at least one subelement that can only be read.

NOTICE
Synchronous V.E. variables:
As soon as a V.E. structure contains one synchronous variable, the entire initialization operation with #INIT is synchronous, i.e. it is only executed in the interpolator context. Therefore, possible asynchronous subelements are also affected by this operation because they may not yet be reinitialised by a subsequent read access.
To achieve complete synchronism in these cases, users should therefore manually program a #FLUSH WAIT command before the #INIT command.
Tip:
When the #INIT command is used, it is recommended to create V.E. structure variables so that all elements are completely synchronous or completely asynchronous.

```
Initialisation of single V.E. variables:
Nxx #INIT V.E.EXT1, V.E.EXT2, V.E.EXT3


Initialisation of a V.E. array variable:
Nxx #INIT V.E.ARRAY1


Initialisation of specific V.E. array variables:
Nxx #INIT V.E.ARRAY1[5], V.E.ARRAY1[8], V.E.ARRAY1[20]


Initialisation of a V.E. structure variable:
Nxx #INIT V.E.STRUCT1


Initialisation of specific elements of a V.E. structure variable:
Nxx #INIT V.E.STRUCT1.NBR_POINTS, V.E.STRUCT1.POINTS


Combined initialisation of V.E. variables:
Nxx #INIT V.E.EXT2, V.E.ARRAY1[5], V.E.STRUCT1.POINTS
...
```

15   Spindle programming

Spindle programming is performed in accordance with the standard syntax defined in ISO and ISO + extensions. This is required in particular for compatibility reasons and owing to certain standard functionalities such as turning, tapping and gear changing etc.

In order to comply with the requirements of new machine concepts and production technologies in flexible spindle programming, each spindle present on the channel also features the additional option of axis-specific programming.

This syntax allows several spindles to be addressed independently in multi-spindle systems in one NC block simultaneously (P-CHAN-00082, [6] [▶ 786]-8.8). In this case, it must be noted that only one spindle can be programmed at a time, the so-called "main spindle", both in the standard syntax and in the spindle-specific syntax.

All other spindles can be addressed only using the spindle-specific syntax (Section Spindle override in DIN syntax (G167) [▶ 632]).

Fig. 155: Correct use of DIN syntax and spindle-specific syntax

The spindles and the main spindle are defined in the channel parameter list [1] [▶ 786]-3. This configuration is available after controller start-up. The main spindle can be changed in the NC program by means of an NC command (#MAIN SPINDLE, see Section Programmable spindle override [▶ 647]).

The table below shows what NC commands must be used only in the DIN syntax in conjunction with spindle programming and what NC commands are also permitted within the extended spindle-specific syntax.

Overview of spindle commands

Description	DIN syntax	Spindle-specific syntax
Spindle M functions	M3, M4, M5, M19	M3, M4, M5, M19
Speed	W	REV
Position	S.POS	POS
User-specific M/H functions	Mxx/Hxx	Mxx/Hxx
Gear change (mechanical)	M40-M45	
Thread cutting	G33	
Tapping	G63	
Turning	G95, G96, G97, G196	
C axis	#CAX	
Gear change (new data)	G112	
Homing	G74	G74
Override 100%	G167	G167
Explicit request for a spindle axis		CALLAX
Explicit release of a spindle axis		PUTAX
Adopt tool dynamic data		GET_DYNAMIK_DATA
Feedforward control		G135, G136, G137
Feed linking		FEED_LINK

15.1 Parameterising spindles

15.1.1 Axis parameters

Axis channel parameter lists must be defined both for closed-loop controlled spindles and for open-loop controlled (PLC) spindles to enter the spindle-specific parameters [2] [▶ 786].

15.1.2 Channel parameter

Further entries which are made in the channel parameters are required for programming spindles in the NC program [1] [▶ 786].

In this case, each spindle to be addressed by this channel must be declared. For this purpose, a string (axis name) and the corresponding logical axis number are defined for each spindle. The axis names of the spindle can be freely selected but they must always start with "S" (e.g. S, S_MAIN, S1, SPINDEL_1).

In addition, the synchronisation modes must be defined spindle-specific for the spindle M functions (M3, M4, M5 and M19) and for the S word. The appropriate meaning of M3, M4, M5 and M19 must then be switched (P-CHAN-00098).

i	The synchronisation method of the S function has no effect if a spindle M function is programmed in the NC block. Synchronisation only takes place based on the settings for the spindle M function. The following priorities apply: **M19 > M3/M4/M5 > S**

If the spindles are to be considered in the "Production time calculation" simulation mode, the data required for this can also be parameterised spindle-specific.

To ensure that this remains compatible with previous programming, one spindle must be declared as the main spindle (P-CHAN-00051, P-CHAN-00053). The main spindle can then be programmed together with specific standard functionalities (e.g. tapping and gear changing etc.) in the conventional DIN syntax. Even if there is only one spindle in the system, it must be configured as the main spindle.

The optional gear changing function for the main spindle (P-CHAN-00052) is also enabled by setting a flag.

The default configuration defined in the channel parameters [1] [▶ 786] is provided after controller start-up.

Example 1:

Configuration of a 1-channel system with 3 spindles. The spindle with axis number 6 is to be the main spindle. Gear change for this spindle is deactivated.
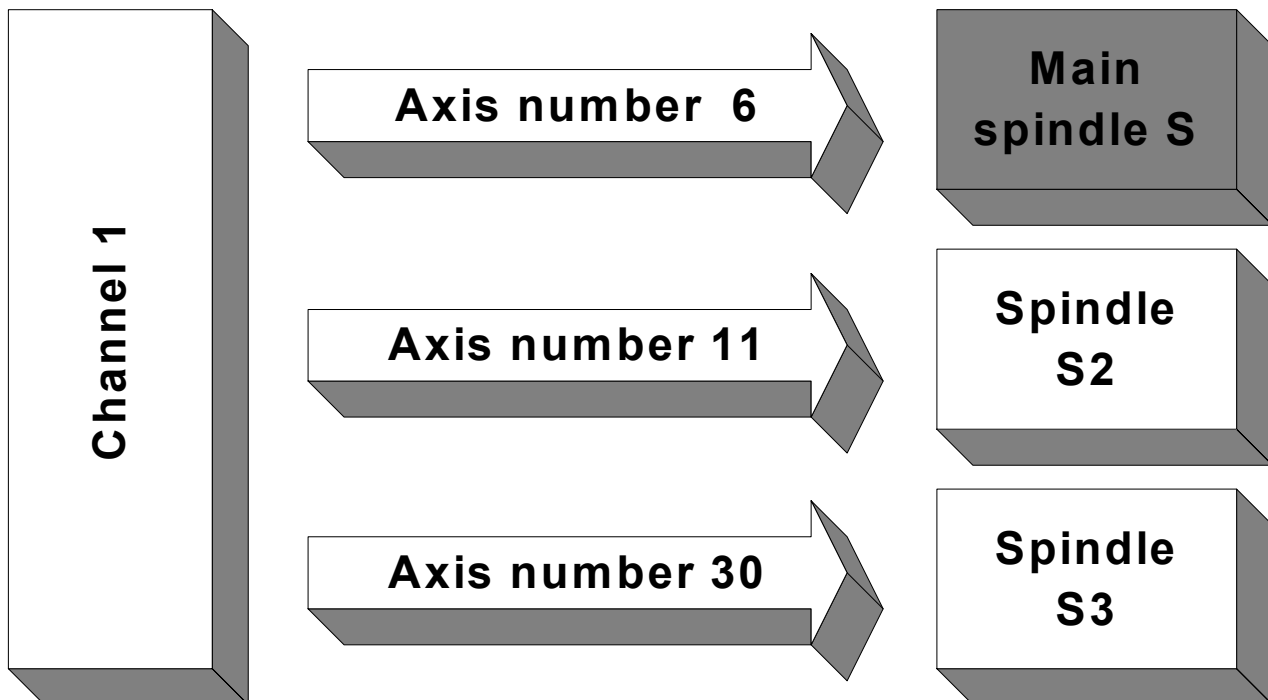
Channel parameter list [1] [▶ 786]:

```
:
spdl_anzahl                                        3
:
main_spindle_ax_nr                                 6 -> -> ->-
main_spindle_name                                  S          |
main_spindle_gear_change                           0          |
#                                                             |
spindel[0].bezeichnung                             S1         |
spindel[0].log_achs_nr                             6 -< -< -<-
spindel[0].s_synch                                 0x00000001
spindel[0].m3_synch                                0x00000002
spindel[0].m4_synch                                0x00000004
spindel[0].m5_synch                                0x00000008
spindel[0].m19_synch                               0x00000001
spindel[1].bezeichnung                             S2
spindel[1].log_achs_nr                             11
spindel[1].s_synch                                 0x00000001
spindel[1].m3_synch                                0x00000002
spindel[1].m4_synch                                0x00000004
spindel[1].m5_synch                                0x00000008
spindel[1].m19_synch                               0x00000001
spindel[2].bezeichnung                             S3
spindel[2].log_achs_nr                             30
spindel[2].s_synch                                 0x00000001
spindel[2].m3_synch                                0x00000002
spindel[2].m4_synch                                0x00000004
spindel[2].m5_synch                                0x00000008
spindel[2].m19_synch                               0x00000001
:
```

After start-up, the spindle with the logical axis number 6 is the main spindle. It is addressed via the spindle name "S" and can be programmed in conventional DIN syntax or in spindle-specific syntax. Spindles "S2" and "S3" can only be programmed in spindle-specific syntax.
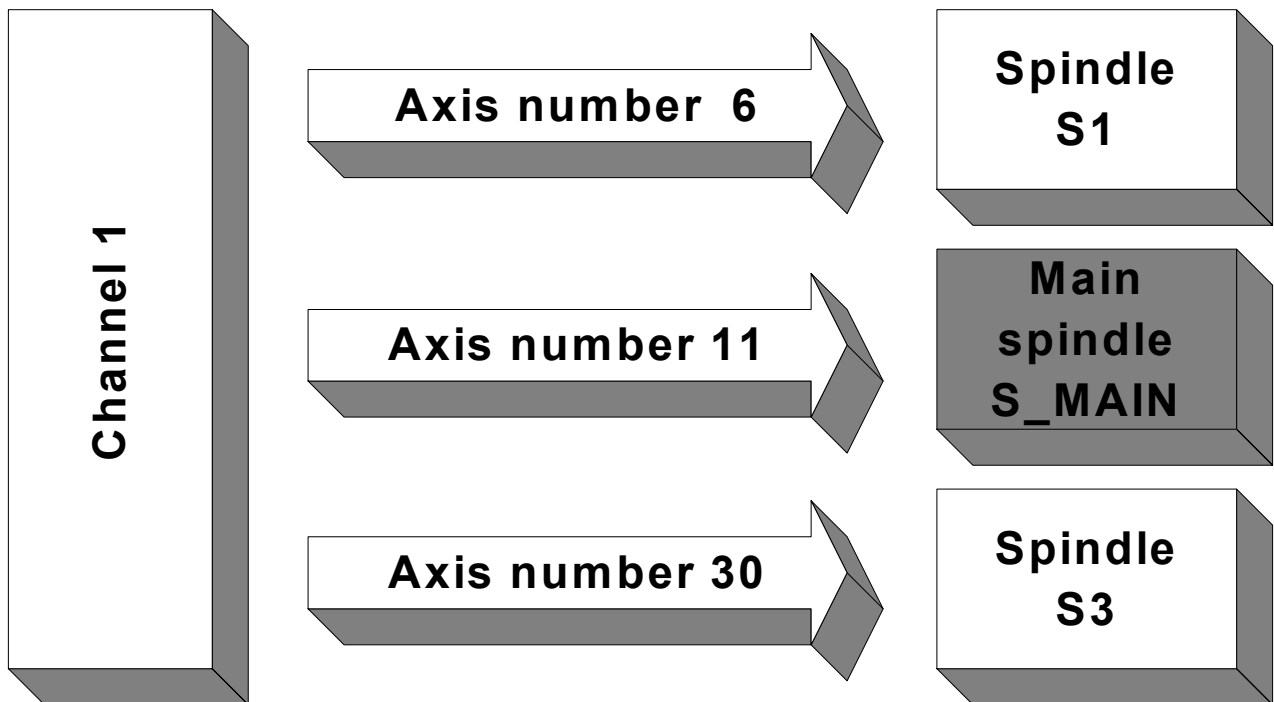
Example 2:

Configuration of a 1-channel system with 3 spindles. The spindle with axis number 11 is to be the main spindle. Gear changing for this spindle is deactivated.

Channel parameter list [1] [▶ 786]:

```
:
spdl_anzahl                                     3
:
main_spindle_ax_nr                              11 -> -> -> ->
main_spindle_name                               S_MAIN       |
main_spindle_gear_change                        0            |
#                                                            |
spindel[0].bezeichnung                          S1           |
spindel[0].log_achs_nr                          6            |
spindel[0].s_synch                              0x00000001   |
spindel[0].m3_synch                             0x00000002   |
spindel[0].m4_synch                             0x00000004   |
spindel[0].m5_synch                             0x00000008   |
spindel[0].m19_synch                            0x00000001   |
spindel[1].bezeichnung                          S2           |
spindel[1].log_achs_nr                          11 -< -< -< -<
spindel[1].s_synch                              0x00000001
spindel[1].m3_synch                             0x00000002
spindel[1].m4_synch                             0x00000004
spindel[1].m5_synch                             0x00000008
spindel[1].m19_synch                            0x00000001
spindel[2].bezeichnung                          S3
spindel[2].log_achs_nr                          30
spindel[2].s_synch                              0x00000001
spindel[2].m3_synch                             0x00000002
spindel[2].m4_synch                             0x00000004
spindel[2].m5_synch                             0x00000008
spindel[2].m19_synch                            0x00000001
:
```
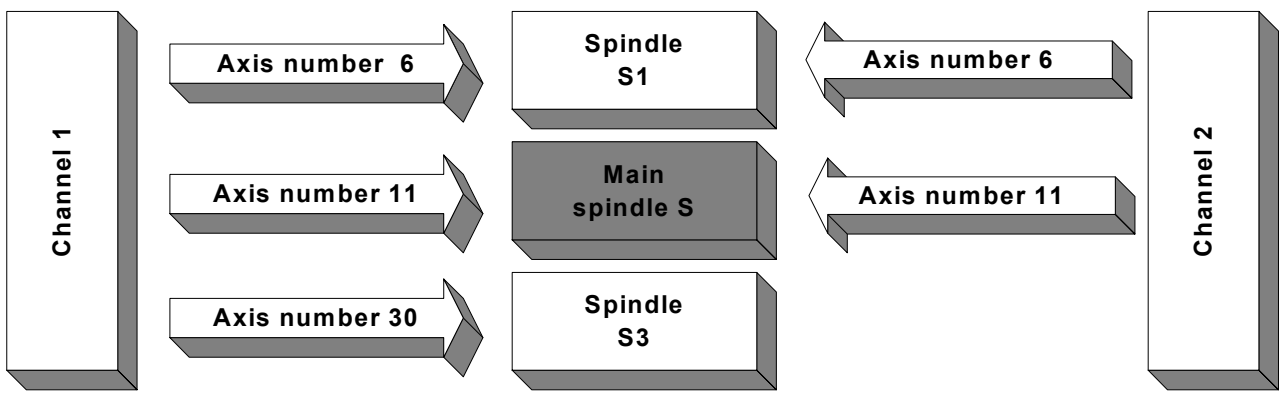
After start-up, the spindle with the logical axis number 11 is the main spindle. It is addressed by the spindle name "S_MAIN" and can be programmed in conventional DIN syntax or in spindle-specific syntax. Spindles "S1" and "S3" can only be programmed in spindle-specific syntax.

Example 3:

Configuration of a 2-channel system with a total of 3 spindles:

Channel 1: 3 spindles. Spindle with the axis number 11 is to be the main spindle.

Channel parameter list [1] [▶ 786]:

```
:
spdl_anzahl                              3
:
main_spindle_ax_nr                       11 -> -> -> ->
main_spindle_name                        S           |
main_spindle_gear_change                 0           |
#                                                    |
spindel[0].bezeichnung                   S1          |
spindel[0].log_achs_nr                   6           |
spindel[0].s_synch                       0x00000001  |
spindel[0].m3_synch                      0x00000002  |
spindel[0].m4_synch                      0x00000004  |
spindel[0].m5_synch                      0x00000008  |
spindel[0].m19_synch                     0x00000001  |
spindel[1].bezeichnung                   S2          |
spindel[1].log_achs_nr                   11 -< -< -< -<
spindel[1].s_synch                       0x00000001
spindel[1].m3_synch                      0x00000002
spindel[1].m4_synch                      0x00000004
spindel[1].m5_synch                      0x00000008
spindel[1].m19_synch                     0x00000001
spindel[2].bezeichnung                   S3
spindel[2].log_achs_nr                   30
spindel[2].s_synch                       0x00000001
spindel[2].m3_synch                      0x00000002
spindel[2].m4_synch                      0x00000004
spindel[2].m5_synch                      0x00000008
spindel[2].m19_synch                     0x00000001
:
```

Channel 2: 2 spindles. Spindle with the axis number 11 is to be the main spindle.

Channel parameter list [1] [▶ 786]:

```
:
spdl_anzahl                                          2
:
main_spindle_ax_nr                                   11 -> -> ->-
main_spindle_name                                    S          |
main_spindle_gear_change                             0          |
#                                                               |
spindel[0].bezeichnung                               S1         |
spindel[0].log_achs_nr                               6 -< -< -<-
spindel[0].s_synch                                   0x00000001
spindel[0].m3_synch                                  0x00000002
spindel[0].m4_synch                                  0x00000004
spindel[0].m5_synch                                  0x00000008
spindel[0].m19_synch                                 0x00000001
spindel[1].bezeichnung                               S2
spindel[1].log_achs_nr                               11
spindel[1].s_synch                                   0x00000001
spindel[1].m3_synch                                  0x00000002
spindel[1].m4_synch                                  0x00000004
spindel[1].m5_synch                                  0x00000008
spindel[1].m19_synch                                 0x00000001
:
```

After start-up, the spindle with the logical axis number 11 can be addressed as the main spindle by spindle name "S" for both channels. It can be programmed in conventional DIN syntax or in spindle-specific syntax. Spindle "S1" can also be programmed from both channels in spindle-specific syntax. Spindle "S3" is only available in channel 1. This spindle is not known in channel 2.

15.2 Programming in DIN syntax

15.2.1 The spindle M functions

15.2.1.1 Move spindle in DIN syntax ((M3/M4/M5)

```
Syntax:
```

M03	Spindle rotation clockwise (cw)	modal
M04	Spindle rotation counter-clockwise (ccw)	modal
M05	Stop spindle	modal

The spindle M functions M03, M04 and M05 define the spindle operation mode and must be used in conjunction with the S word (Sec. Spindle speed (S word) [▶ 593]). They are modal and each may only be programmed alone in the NC block.

Spindle rotation is activated if M03 or M04 is programmed and a valid speed is set.

M05 stops spindle rotation. Note that this spindle M function is the default spindle mode after controller start-up and initial program start.

If no M05 is set at program end, the spindle continues to rotate.

Move spindle (M3, M4, M5)

```
N10        S1000     (Speed 1000 rpm is stored, none)                 (Spindle rotation
because M05 is default)
N20  M03              (Spindle rotation cw at 1000 rpm)
N30  M04              (Spindle rotation ccw at 1000 rpm)
N40   S500       (Spindle rotation ccw at 500 rpm)
N50 M05  S300     (Spindle stop, speed 300 rpm is)                 (stored)
N60  M04              (Spindle rotation ccw at 300 rpm)
N70  M05             (Spindle stop)
N80  M03   S1000   (Spindle rotation cw at 1000 rpm)
N90 M30              (Program end)
```

Channel parameter list [1] [▶ 786]:

The synchronisation modes must be defined spindle-specific for M3, M4, M5. The M function is not executed for synchronisation mode "0" (NO_SYNCH).

```
:
spindel[0].bezeichnung                        S1
spindel[0].log_achs_nr                        6
spindel[0].s_synch                            0x00000001
spindel[0].m3_synch                           0x00000002
spindel[0].m4_synch                           0x00000002
spindel[0].m5_synch                           0x00000008
spindel[0].m19_synch                          0x00000001
```

15.2.1.2   Position spindle in DIN syntax ((M19,*. POS)

Syntax:

M19	Positioning spindle	non-modal
<spindle_name>.**POS**=..	Spindle position	modal

Spindle positioning can be represented by the following syntax:

M19 [*<spindle_name>*.**POS**=..] [**M03** | **M04**] [*<spindle_name>*=..]

M19   Positioning spindle

<spindle_name>.POS= Spindle position in [°]. Designation of the main spindle as per P-CHAN-00053
..

<spindle_name>=..   Spindle speed in [rpm].. Designation of the main spindle as per P-CHAN-00053

M03/M04 or spindle speed in the same NC block are optional. However, a valid spindle speed (> 0) must be set.

The spindle position is modal and need not be respecified if M19 is reprogrammed. If no spindle position was previously programmed, the motion is moved to position "zero" by default.

If the spindle is not rotating, positioning is executed with the shortest motion path.

Spindle positioning with M19 is only permitted for position-controlled spindles.

Positioning spindle (M19, *.POS)

The spindle is positioned at 180° in each of the following examples. The "=" character is optional:

```
M19 S.POS180
M19 S.POS 180
M19 S.POS=180
M19 SPINDEL.POS=180
M19 S1.POS=180
```

The spindle does not rotate during positioning. The shortest motion path is calculated.

N10 M05 S100	(Spindle stop, speed 100 rpm is stored) (is stored as default)
N20 M19 S.POS180	(Position at 100 rpm at position 180) (The direction of rotation results from the shortest) (motion path)
N30 M19 S200 S.POS90	(The direction of rotation results from the) (shortest motion path) (Position at 200 rpm ccw at position 90)

Channel parameter list [1] [▶ 786]:

The synchronisation mode must be defined spindle-specific for M19. The M function is not executed for synchronisation mode "0" (NO_SYNCH).

```
:
spindel[0].bezeichnung                    S1
spindel[0].log_achs_nr                    6
spindel[0].s_synch                        0x00000001
spindel[0].m3_synch                       0x00000002
spindel[0].m4_synch                       0x00000002
spindel[0].m5_synch                       0x00000008
spindel[0].m19_synch                      0x00000001
```

15.2.2 Spindle speed (S)

`Syntax:`

<spindle_name> ..   modal

<spindle_name>..   Designation of the main spindle as per P-CHAN-00053 with spindle speed in [rpm]

In the configuration, the main spindle can be assigned a string in the channel parameter list (P-CHAN-00053). In order to avoid ambiguities, there must be an equals sign in front of the speed entry after all spindle names consisting of more than one character.

Values can be assigned to the S word directly or by means of parameters. Decimal numbers are also permitted (REAL format).

A distinction is made between the following types of use of the S word in conjunction with spindle M functions:

1. S word in conjunction with M03, M04, M19:
   If the S word or the string used for it is programmed in conjunction with M03/M04 or M19, the value following the S word is interpreted as the spindle speed and is output to the spindle.
2. S word in conjunction with M05:
   The value following the S word is adopted as the spindle speed in the working data in conjunction with M05, but it is not output to the spindle.

The S word alone does not generate a motion in the NC program. This requires that a spindle mode M03, M04, M19 is known. Accordingly, programming M03, M04 and M19 only causes motion when the S word is set (> 0, analogous to G01, G02 and G03 in which traversing only occurs when the feedrate and the axes to be traversed are specified).

● In the case of a negative S value, an error message is output.

i A negative S value is only permitted in conjunction with G63 (tapping) since it triggers reversal of the direction of rotation on withdrawal out of the thread bore.

BECKHOFF

Programming with spindle S

```
N10        S300      (Speed 300 rpm is stored)
N20   M04           (Spindle rotation ccw at 300 rpm)
N30   M03   S1000   (Spindle rotation cw at 1000 rpm)
N40        S500      (M03 active, spindle rotation cw at 500 rpm)
N50 M05        S100    (Spindle stop, speed 100 rpm is stored)
N60   M04           (Spindle rotation ccw at 100 rpm)
N70   M05           (Spindle stop)
N80 M30 (Program end)
```

Channel parameter list [1] [▶ 786]:

The synchronisation mode must be defined spindle-specific for the S word. An error message is generated in the case of synchronisation mode "0" (NO_SYNCH) since an S word may not be ignored.

```
:
spindel[0].bezeichnung                   S1
spindel[0].log_achs_nr                   6
spindel[0].s_synch                       0x00000001
spindel[0].m3_synch                      0x00000002
spindel[0].m4_synch                      0x00000002
spindel[0].m5_synch                      0x00000008
spindel[0].m19_synch                     0x00000001
```

Spindle speed (S word)

```
N10   M03  S100             (Spindle rotation at 100 rpm)
N20   M19  S.POS90          (At 100 rpm cw to position 90)
N30    M04               (Spindle rotation ccw at 100 rpm)
N40   M19  S200  S.POS 180 (At 200 rpm cw to position 180)
N50 M05    S150     (Spindle stop, speed 150 rpm)
                           (is stored)
N60   M19  S.POS=135       (Position at 150 rpm on)
                           (the shortest path to position 135.)
N70   M03  S300             (Spindle rotation cw at 300 rpm)
N80   M19  S200  S.POS270  (At 200 rpm cw to position 270)
N90   M03  S400  S.POS45   (Spindle rotation cw at 400 rpm,)
                           (position 45 is stored)
N100  M19                  (At 400 rpm cw to position 45)
N110  M04  S800            (Spindle rotation ccw at 800 rpm)
N120  S1200                (Spindle rotation ccw at 1200 rpm)
N130  M5                   (Spindle stop)
N140  M03                  (Spindle rotation cw at 1200 rpm)
N150  M19                  (At 1200 rpm cw to position 45)
```

15.2.3   Spindle gear change (M40 - M45)

A spindle gear change is programmed with M40...M45. These M functions define max. 6 gear stages. A separate gear data record must be parameterised for each gear stage in the associated spindle axis list [AXIS].

The M functions can be programmed together with the spindle speed and the M function for the direction of rotation in the same NC block. Functions M40 to M45 are used to select the gear stage and to trigger mechanical gear change.

```
Syntax:
```
M40 | M41 | M42 | M43 | M44 | M45 [<s*pindle_name*>..] **[M03 | M04]**   modal

M40 to M45	Gear stages 1 to 6
<spindle_name>=<*expr*>	Spindle speed consisting of spindle name according to P-CHAN-00053 and speed value in [rpm].

Spindle gear change (M40 - M45)

```
S800 M41 M03 ;Spindle speed 800, gear stage 2, rotation cw
```

- The decoded functions M40 to M45 are modal and are activated at block start. M40 to M45 cancel each other out.

- The system permits definition of max. 6 spindle gear stages (M40... M45). The minimum and maximum speeds are defined for each gear stage in a "Table of speed stages" in [1] [▶ 786]-4 (unit = rpm).

- The maximum speed for the 10 V output in the case of position-controlled spindles is defined for analogue drives via the Multi-Gain Factors P-AXIS-00128 and P-CHAN-00129..

- In a system with automatic gear stage selection, this is determined solely by programming the speed S. M40 to M45 then do not need to be programmed.

- M40 to M45 can only be programmed for closed-loop position-controlled spindles.

- The NC kernel always attempts to minimise gear changing (for example, if a new speed can be used with the current gear stage, a gear change is suppressed even if it was explicitly programmed with M40 to M45).

Channel parameter list [1] [▶ 786]:

- Definition of M functions M40 - M45 and definition of synchronisation modes.

```
:
m_synch[1]          0x00000001        MOS
m_synch[2]          0x00000002        MVS_SVS
:
m_synch[40]         0x00000002        MVS_SVS
m_synch[41]         0x00000002        MVS_SVS
m_synch[42]         0x00000002        MVS_SVS
m_synch[43]         0x00000002        MVS_SVS
m_synch[44]         0x00000002        MVS_SVS
m_synch[45]         0x00000002        MVS_SVS
m_synch[48]         0x00000008        MNS_SNS
m_synch[49]         0x00000002        MVS_SVS
```

- Activate gear change:

```
main_spindle_gear_change   1     0:OFF          1:ON
```

- Parameterise the spindle gear (search direction, speed ranges):

```
:
spindel[0].range_way       0     0:bottom up   1: top down

#
spindel[0].range_table[0].min_speed      50      (M40)
spindel[0].range_table[0].max_speed      560     (M40)
spindel[0].range_table[1].min_speed      400             (M41)
spindel[0].range_table[1].max_speed      800             (M41)
spindel[0].range_table[2].min_speed      700     (M42)
spindel[0].range_table[2].max_speed      3500    (M42)
spindel[0].range_table[3].min_speed      3501            (M43)
spindel[0].range_table[3].max_speed      4000            (M43)
spindel[0].range_table[4].min_speed      3800    (M44)
spindel[0].range_table[4].max_speed      5500    (M44)
spindel[0].range_table[5].min_speed      5400            (M45)
spindel[0].range_table[5].max_speed      7000            (M45)
#
:
```

Spindle gear change (M40 - M45)

Automatic determination of the gear stage: ON

```
:
spindel[0].autom_range 1
:

NC program:
S650 M03                 OK, M41☐ PLC
S750                     OK, no change, M41 already selected
S950                     OK, automatic change, M42=> PLC
S1050                    OK, no change, M42 already selected
S750                     OK, automatic change, M41=> PLC
S500                     OK, no change, M41 already selected

S8000                    Error, speed too high

A programmed gear stage is always checked:
M41 S750                 OK, "automatic" change, M41=> PLC
...but
M40 S750                 Error, incorrect gear stage
```

Spindle gear change (M40 - M45)

Automatic determination of the gear stage: OFF

```
:
spindel[0].autom_range 0
:

NC program:
M41 S650 M03             OK, M41 => PLC
M41 S750                 OK, no change, M41 already selected
M42 S950       OK, change, M42=> PLC
M42 S1050      OK, no change, M41 already selected
M41 S750       OK, change, M41=> PLC
M41 S500                 OK, no change, M41 already selected

M41 S200 Error, program different gear stage (M40)
S950       Error, no gear stage (M42) programmed
```

15.2.4   Turning functions

15.2.4.1   Diameter programming (G51/G52)

Syntax:

| **G51** | Select diameter value | modal |
| **G52** | Deselect diameter value | modal, initial state |

Fig. 156: Reference points and diameter programming for turning

W	Workpiece zero point	P	Cutting point
X	Face turning axis	Z	Longitudinal turning axis
M	Machine zero point	A	Fixed stop point
C	Control zero point	d	Programmed dimension for diameter programming

When diameter programming is selected, the positional values for the face turning axis in a motion block are interpreted as diameter values relative to the turning centre point.

It should be noted that the programmed coordinates of the face turning axis only correspond to the workpiece diameter if the zero point of the face turning axis is located at the turning centre point (irrespective of whether offsets act as a diameter; see Programming example).

The axis parameters can parameterise axes in "face turning" mode:

- G51 with absolute programming (G90) (P-AXIS-00058)
- … and/or G51 with relative programming (G91) (P-AXIS-00059
- Reference point programming (G92) and zero point programming (G53 – G59) in the diameter (P-CHAN-00091)

G51 acts on the axis which is operated in "face turning" mode. When the face turning axis is selected, the face turning axis must exist in the machining plane (G17, G18, G19).

The coordinates of the circle centre point (I, J, K) and circle radius programming (R) are not programmed in the diameter.

Diameter programming is deselected with G52.

Diameter programming (G51/G52) in G18

```
;General settings (optional):
;Display position values in the diameter P-CHAN-00256 (TRUE, 1)

;Settings of X axis:
;Face turning axis, translatory: P-AXIS-00015 (0x41)
;G51 with G90: P-AXIS-00058 (TRUE, 1)
;G51 with G91: P-AXIS-00059 (FALSE, 0)(optional)
;G92, G53-G59 in the diameter: P-CHAN-00091 (TRUE, 1) (optional)

;Settings of Z axis:
;Longitudinal turning axis, translatory: P-AXIS-00015 (0x81)

:
N05 G18
N10 G90 G01 F1000
N20 G51 X80          ;Diameter 80 mm
N30 G92 X10          ;G92 by 10 mm in the diameter
N40 X0               ;Position 0 + G92 => diameter 10 mm
N50 G91 X50          ;X relative +50 mm, not in the diameter
N80 G52              ;Deselect diameter programming
:
```

15.2.4.2 Cutter radius compensation (G40/G41/G42)

`Syntax:`

G40	Deselect CRC	modal, initial state
G41	CRC left of contour	modal
G42	CRC right of contour	modal

Tool tip radius compensation (SRK) acts in the machining plane selected using G17, G18, G19 for turning work. In this plane, one of the axes must be operated in "face turning" mode and the other in "longitudinal turning" mode. (axis mode: P-AXIS-00015)

The data records stored in the D words are used as tool compensation values. For turning tools, the orientation of the cutter edge relative to the machining plane (face/longitudinal turning axis) must be specified in the parameter P-TOOL-00002 by an additional identifier 1...9 (see figure).

Fig. 157: Orientation of cutter edge to machining plane.

A typical turning tool is characterised by the following values/parameters:

- Tool type   1 (turning tool)
- SRK orientation   1...9
- Tool radius   Tool tip radius
- Tool length   --
- Tool offset (see figure below)

S: Cutter radius mid-point
r: Cutter radius
P: Theoretical tool cutter tip
F: Tool slide reference point
TV: Tool shift vector

Fig. 158: Tool gauging for tool offset compensation.

When specifying tool axis offsets, their mathematical sign should be noted since it refers to components of the tool offset vector in the machining plane. In the example of a turning tool shown in the above figure, the offsets in the direction of the X and Y axis both have negative (mathematical) signs.

Tool offsets must be specified up to the theoretical tool top (point P).

A change between turning tool and milling tool is permitted when G41 or G42 is selected. With absolute programming (G90) the current axis offset values of the new tool are included in the calculation for the next motion block depending on the tool type.

15.2.4.3   Feedrate per revolution (G95)

Syntax:

| **G95** | Feedrate in mm/revolution | modal |

During turning with active G95, a constant chip thickness can be defined with the F Word in [mm/rev, inch/rev] irrespective of spindle speed.

Here, the axis feedrate is linked to the rotational speed of the position-controlled spindle. It is only valid in conjunction with the G function with which it was programmed. This means that if a change is made from G95 to G94 or G93 (Section "Machining time or feedrate" [▶ 149]) the valid F word for G95 is not adopted..

G95 can also be programmed in combination with a controlled (PLC) main spindle (P-CHAN-00069). Please note here that the resulting feedrate per revolution is only dependent on the programmed values (F word, S word). Real-time influences on the spindle speed are not considered, e.g. override changes. This type of programming in only permitted in a channel in conjunction with the assigned main spindle as of Build V3.1.3066.02.

Feedrate per revolution (G95)

```
N10 F1000 G01 X0 Z100 M3 S1200  ;Feedrate 1000 mm/min (G94)
N20 G95 F1.5            ;Feedrate per revolution 1.5 mm/rev,
                       ;speed 1200 rpm
N30 Z50                ;Feedrate 1800 mm/min
N40 G94 X50            ;Feedrate 1000 mm/min valid from N10
N50 G93 F20 X20        ;Machining time 20 s
N60 G95 Y200 S2000     ;Feedrate per revolution 1.5 mm/rev valid from N20,
                       ;speed 2000 rpm
N70 M30
```

Feedrate per revolution (G95) and axis coupling with variable gear ratio (gear coupling)

As of Build 3.1.3079.03 the feedrate of the path axes can be coupled to the speed of the main spindle with #TURN [ROT_FEED_CPL=1]. The feedrate is adapted depending on the gear ratio settings of the axis couplings. This permits the activation of the feedrate per revolution (G95) while spindles are rotating.

Axis couplings with variable gear ratios can be linked by

- the NC command #GEAR LINK [▶ 464] or

- yia the HLI [FCT-A9// Axis coupling via HLI]

.

In the default setting of #TURN [ROT_FEED_CPL=0], the speed of the main spindle is used with no consideration given to the axis couplings.

If the main spindle is changed, the setting of ROT_FEED_CPL is adopted automatically for the new main spindle and deactivated for the previous main spindle.

G95 - Spindle speed with gear coupling

```
;Axis 5 is configured as main spindle, Axis 6 (S2) is configured as auxiliary spindle.
; ----------------------------------------------
; Axis 5 is coupled to Axis 6 and to itself.
N10 #GEAR LINK ON [TARGETNR=5 AXNR1=6 AXNR2=5 NUM1=1 DENOM1=1 NUM2=1 DENOM2=1]
N20 M3 S500         ; commanded speed of Axis 5 = 500 rpm
N30 N30 [M3 REV1500] ; commanded speed of Axis 6 = 1500 rpm
; speed of Axis 5 = 500 rpm + 1500 rpm = 2000 rpm
; speed of Axis 6 = 1500 rpm
; feedrate = 1000 mm/min
N40 F1000 G01 X100
; Spindle speed related to axis coupling is used for G95
N50 #TURN [ROT_FEED_CPL=1]
N60 G95 F1.5        ; feedrate per revolution = 1.5 mm/rev
; feedrate = 2000 rpm * 1.5mm/rev = 3000 mm/min
N70 X200
; Spindle speed without axis coupling is used for G95
N80 #TURN [ROT_FEED_CPL=0]
; feedrate = 500 rpm * 1.5mm/rev = 750 mm/min
N90 X400
N100 M05
N110 SC[M05]
N120 #GEAR LINK OFF [TARGETNR=5]
M30
```

15.2.4.4   Constant cutting speed (G96/G97/G196)

`Syntax:`

G96	Selecting constant cutting speed	modal
G97	Deselecting constant cutting speed, selecting spindle speed	modal, initial state
G196	Maximum spindle speed for G96	G196 non modal
		max. speed modal

Using the G functions G96, G97 and G196, it is possible to optionally change the interpretation of the S word:

G96	S in [m/min or ft/min *] (cutting speed)
G97	S in [rpm] (spindle speed)
G196	S in [rpm] (max. spindle speed during G96)

* [as of Build V2.11.2032.08 with G70 and P-CHAN-00360 = 1]

When selected with G96, the starting rpm of the spindle is calculated from the programmed cutting speed and the distance of the tool tip to the turning centre point. This distance results from the last (not in the current NC block) programmed position and the reference point offset of the face turning axis. Exactly one face turning axis must be present in the current machining plane (G17, G18, G19).

A cutting speed programmed for G96 using the S word is only valid until it is deselected by G97. With G96, constant cutting speed is only activated when the S word is programmed.

Specifying a maximum spindle speed with G196 in conjunction with the S word is optional and only active during G96. Spindle speed limiting must be programmed before G96 is selected.

Fig. 159: Spindle speed with active G96

ⓘ   **Extended G function G196**

As of Build V3.1.3057.04

Alternatively, the maximum spindle speed can be programmed as an additional value in [rpm] in conjunction with G196. It is modal.

This syntax permits the programming of G196 and G96 in the same NC block. A separate specific NC block is not required.

`Syntax:`

G196 = *<Max_spindle_speed>*

G196 non modal,

max. speed modal

Close to the turning centre point, the programmed maximum spindle speed (G196) or the maximum spindle speed specified in the assigned axis parameters P-AXIS-00212 defines the limits of the constant cutting speed.

When deselected with G97, the last spindle speed set is retained.

Motion blocks of the face turning axis in rapid traverse (G00) lead to an interruption of G96 to prevent undesired speed value changes when the tool is positioned. The next motion block with G01, G02 or G03 cancels suppression of G96.

Constant cutting speed (G96/G97/G196)

```
; X is the face turning axis
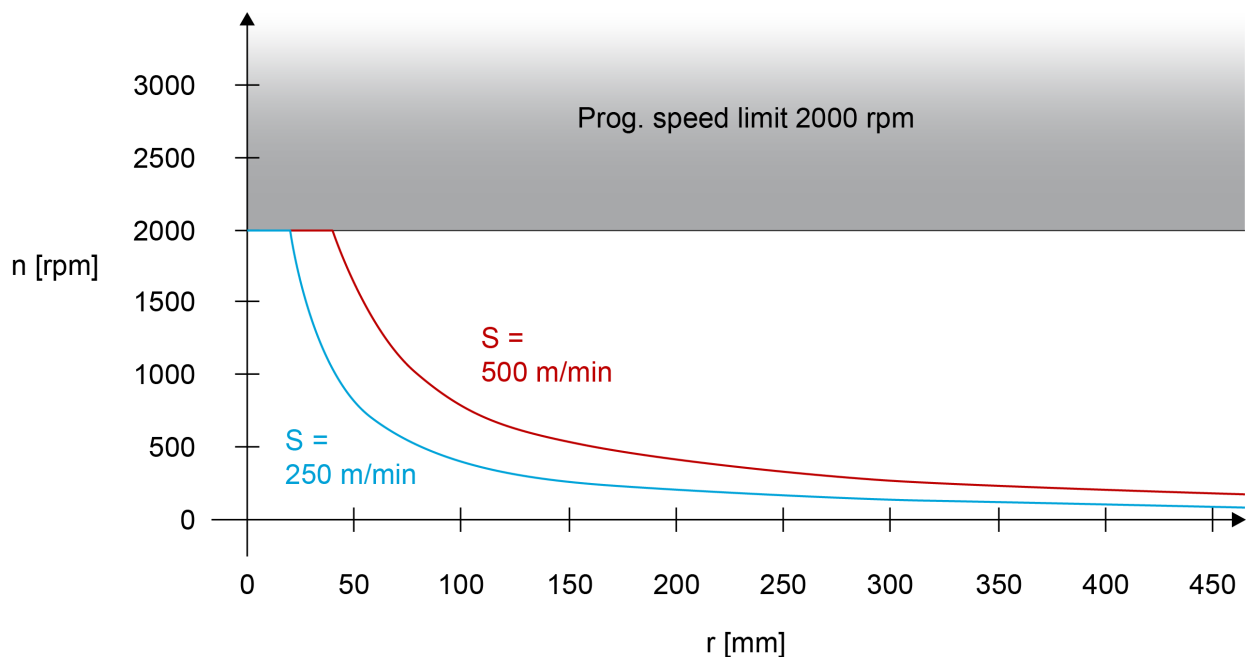
N10 M03 S1000 G01 F1500 X100
N20 G196 S6000    ;max. speed 6000 rpm
N30 G96 S63 ;select const. cutting speed 63 m/min,
            ;workpiece radius 100mm corresp. to X coordinates
N40 X80
N50 S4 X50     ;new cutting speed 4m/min; workpiece radius 80mm,
            ;at block end 50mm
N60 G97    ;max. speed 6000 rpm not effective here!
N80 G92 X-10  ;reference point offset in X by -10mm
N90 G96 X60   ;cutting speed from N50 not valid: const.
            ;cutting speed not active, speed 8000 rpm
N100 S25 X70    ;cutting speed 25m/min, workpiece radius 50mm,
            ;(=60mm+BPV), const. cutting speed active
N110 G00 X450 ;rapid traverse: speed remains constant
N115 X70
N120 G01 X40  ;suppress G96 cancelled
N110 M30
```

15.2.4.5   Thread cutting with endlessly rotating spindle (G33)

Single-start/multi-start threads

When thread cutting with an endlessly rotating spindle (G33), the path motion is synchronised to the zero passage of the spindle rotation. Therefore, the thread can also be cut in several passes in succession. When an offset angle is specified as option, multi-start threads can also be produced.

To achieve a good machining result and to minimise path errors, feedforward control can be selected for the spindle and for path axes.

Programming

```
Syntax example for ZX plane (longitudinal axis Z, feed axis X):
```
G33 Z.. K.. [<s*pindle_name*>**.OFFSET=..**]   modal

G33	Thread cutting with endlessly rotating spindle. The G33 function is modal. The next motion block with a modal block type (G00, G01, G02, G03, spline, polynomial) deselects thread cutting.
Z..	Target point ("thread length") in [mm, inch]
K..	The thread pitch is programmed with active thread cutting in the unit [mm/rev, inch/rev] without a mathematical sign using the address letters I, J and K. They are assigned to the X, Y and Z axes according to DIN 66025.
	The thread pitch is modal up to program end and should not be zero on when G33 is selected. The feed is not programmed using the F word but results from the spindle speed and the thread pitch.
	The pitch of longitudinal or tapered threads at an inclination angle less than 45° is specified by the address letter K if the Z axis is the longitudinal turning axis. With facing or tapered threads with a pitch greater than or equal to 45°, the pitch is specified by I if the X axis is used as the face turning axis, and by J if the Y axis is used. The figure below shows examples for specifying thread pitch using the address letters in the Z-X plane.
<spindle_name>.OFFSET=..	Thread offset angle in [°] in spindle modulo range. Only required as an option for multi-turn threads. The offset angle is modal up to program end. Spindle name according to P-CHAN-00053. The "=" character is optional.

Pitch values I, K with longitudinal thread

Fig. 160: Value of thread pitch for longitudinal thread

Pitch values I, K with tapered thread

Fig. 161: Value of thread pitch for tapered thread

Thread cutting with endlessly rotating spindle (G33)

G33 Z.. K.. [S.OFFSET=..]

Fig. 162: Representation of geometry example

Cutting a longitudinal thread (M70x1.5) with several cuts:

```
%L longit_thread
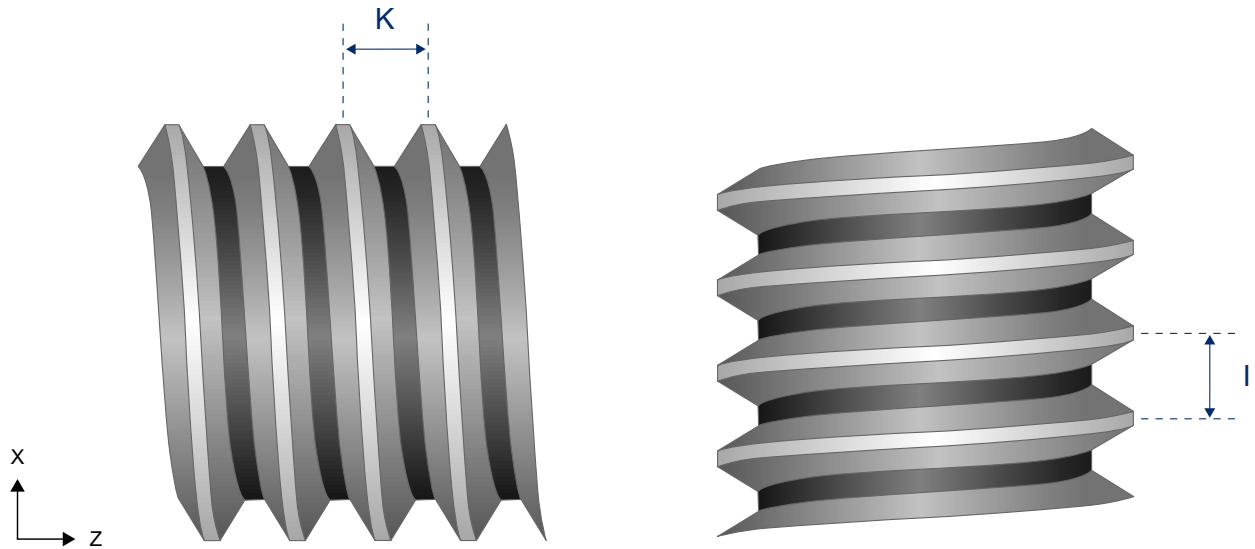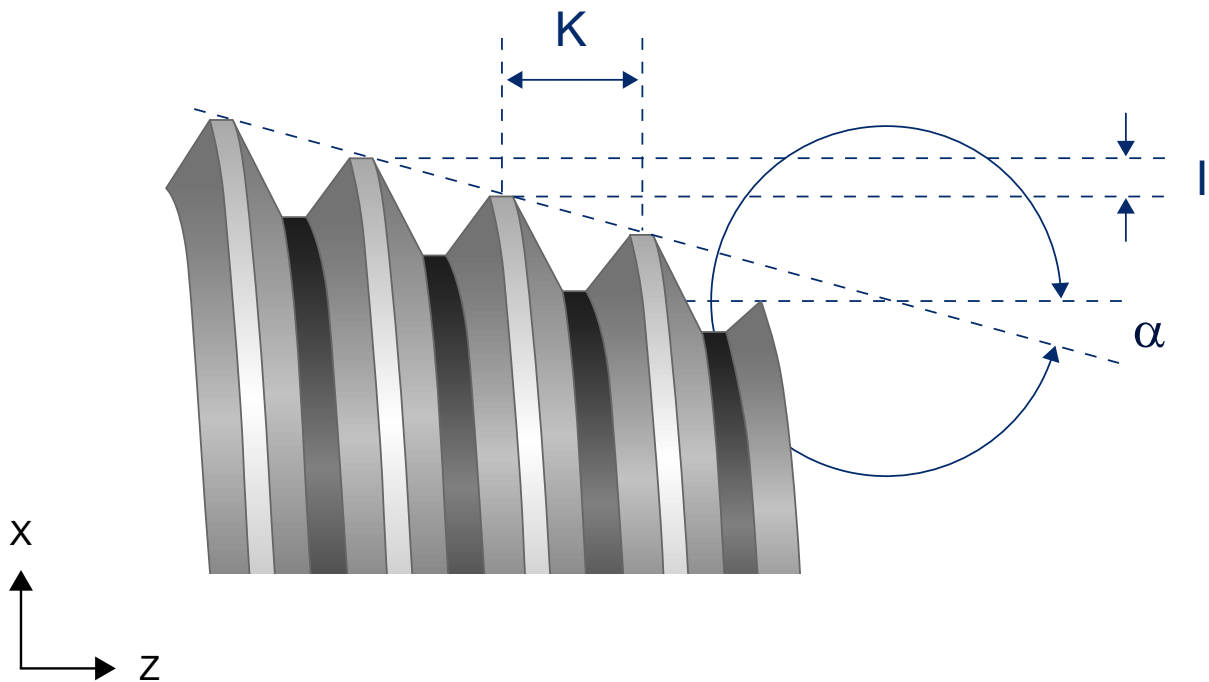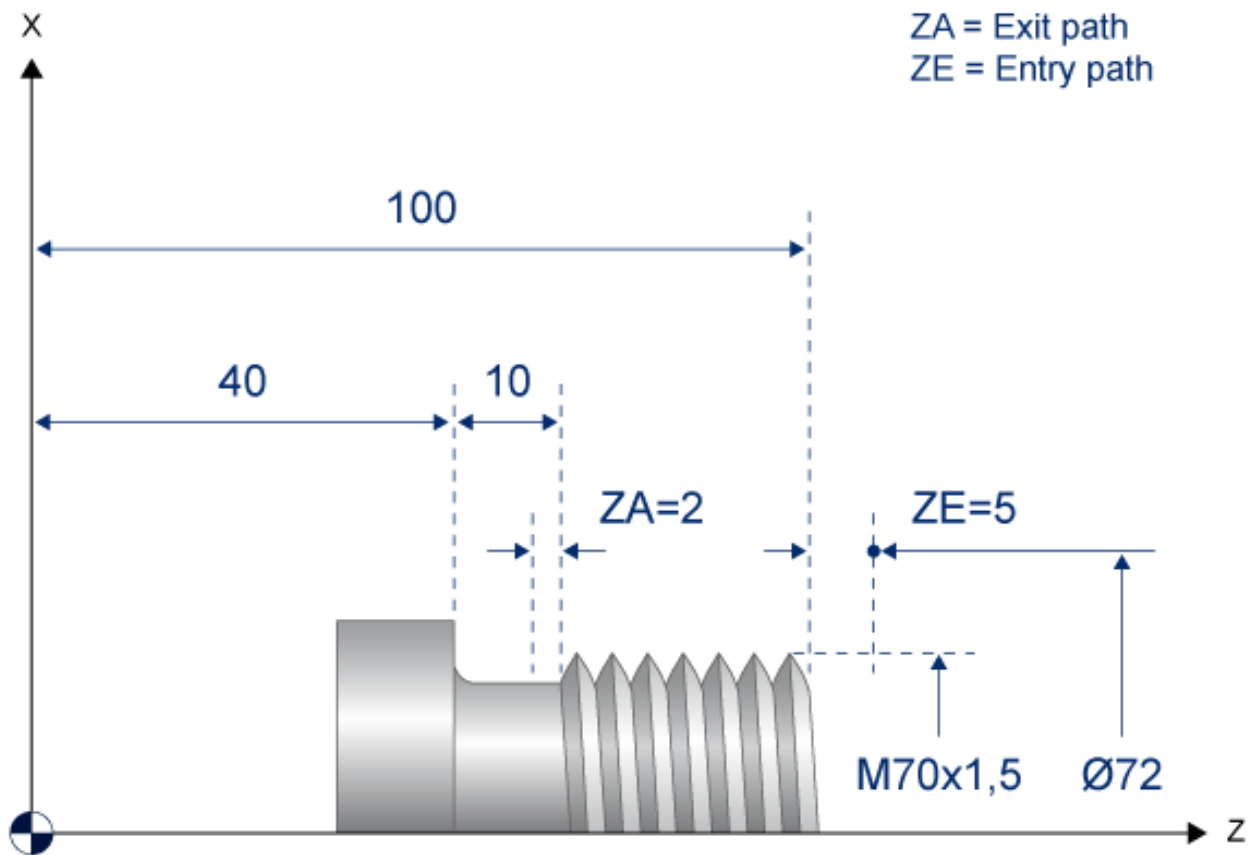
N100 G33 Z48 K1.5                 ;Cut thread turn
N110 G00 X72                      ;Retract and move
N120 Z105                         ;to start position
N130 M29                          ;Subroutine end


%G33 (thread depth 0.92 mm)
N10 G51                           ;Select diameter programming
N15 T1 D1 M03 S400                ;Select tool, start spindle
N20 G00 X72 Z105                  ;Approach

N25 G01 X69.54 F1000              ;Position at 1st cutting depth
N30 LL longitudinal thread            ;1st thread cut

N35 G01 X69.08                    ;Position at 2nd cutting depth
N30 LL longitudinal thread            ;2nd thread cut

N35 G01 X68.62                    ;Position at 3rd cutting depth
N30 LL longitudinal thread            ;3rd thread cut

N35 G01 X68.16                    ;Position at final depth
N30 LL longitudinal thread            ;4th thread cut

N35 G01 X68.16                    ;Reposition at final depth
N30 LL longitudinal thread        ;Empty cut

N60 M05 X150 Z200                  ;Moving to end position
N65 M30                           ;Program end
```

Cut a 2-turn longitudinal thread (M70x1.5)

```
%G33_2 (2-turn thread, thread depth 0.92 mm)
N10 G51                              ;Select diameter programming
N15 T1 D1 M03 S40                    ;Select tool, start spindle
N20 G00 X72 Z105                     ;Approach
N25 G01 X68.16 F1000                 ;Position at thread depth
N30 G33 Z48 K1.5                     ;Cut 1st thread turn
N35 G00 X72                          ;Retract and move
N40 Z105                             ;to next
N45 G01 X68.16                       ;start position
N50 G33 Z48 K1.5 S.OFFSET=180        ;Cut 2nd thread turn at 180°
N55 G00 X72                          ;Retract and move
N60 M05 X150 Z200                  ;to end position
N65 M30                           ;Program end
```

Cutting a tapered thread

```
%L tapered thread
N010 G33 Z90 X1 I5.0                      ;Cut thread turn (reference I)
; N010 G33 Z90 X1 K5.0                     ;Cut thread turn (reference K)
N020 G00 X72                      ;Retract and move
N030 Z105                         ;to start position
N040 M29                          ;Subroutine end

%G33
N050 G00 X0 Y0 Z0
N060 G18
N070 G51                             ;Select diameter programming
N080 D1 M03 S1                 ;Select tool, start spindle
N090 G00 X105 Z105                  ;Start
N100 G01 X100 F1000             ;Position at 1st cutting depth
N110 LL tapered thread            ;1st thread cut
N120 M05 X150 Z200              ;Move to end position
N130 M30                       ;Program end
```

15.2.5   Tapping (G63)

Syntax example of tapping in Z direction:

G63 Z.. **F**.. *<spindle_name>*..   modal

G63	Thread tapping
Z..	Thread depth (target point) in the tapping axis in [mm, inch]
F..	Feed rate in [mm/min, m/min, inch/min]
<spindle_name>..	Spindle speed consisting of spindle name according to P-CHAN-00053 and speed value in [rpm]

With this kind of tapping (G63) the position-controlled spindle is tracked by the CNC synchronously to the path motion. In this case the spindle and the feed motion of the participating axes are matched precisely and dynamically. A compensatory chuck is not required. The programmed feed rate must match the programmed spindle speed and the thread pitch and is calculated as follows:

Feed rate F [mm/min] = speed S [rpm] * pitch [mm/rev]

G63 is deselected by the selecting a different modal block type (e.g. linear motion G01). A non-modal block type (e.g. dwell time with G04) does not deactivate G63.

The path feed rate (F word) and spindle speed (S word) do not necessarily need to be specified in the same NC block as G63. The feed rate calculation must always be based on the last values programmed.

An error message is output if the path feed rate or spindle speed are equal to zero with G63 is selected.

M03, M04, M05, M19 cannot be programmed in combination with G331/G332.

NOTICE
The spindle (or the thread tapping drill) must be at standstill when G63 is selected. This can be achieved by previously programming M05 (Stop spindle) or M19 with S.POS (Position spindle).

Cutting a left-hand thread or movement out of a thread hole is programmed with a **negative S value**.

In C axis mode, the gear stage can be defined using the parameter P-AXIS-00052.

Tapping (G63)

Tap a right-hand thread with pitch 1.25 mm, thread depth 50 mm. At a programmed spindle speed S of 200 rpm the calculated feedrate is:

F = 200*1.25 = 250 mm/min

```
; …
G01 F2000 G90 X0 Y0 Z0 ; position axes
M19 S.POS=0 M3 S100    ; stop and position spindle
; …
G63 Z-50 F250 S200     ; tap
    Z0 S-200           ; retract from threaded bore
G01 F1000 X100         ; reposition, deselect tapping
:
```

Tapping (G63)

```
%Tapping_G63

N05 X0 Y0 Z0
N10 G91 Z100
N20 M19 S.POS180 M3 S100    ; position spindle

N30 G63 Z-50 F300 S200      ; tap
N40 Z100 S-200              ; retract from threaded bore

N50 G01 X200 F3000    ; reposition, deselect tapping

N60 G63 Z-70 F300 S200      ; tap
N70 Z100 S-200              ; retract from threaded bore

N80 M05 G01 X300 F1000
N90 M30
```

15.2.6   Tapping (G331/ G332)

ℹ️   **This function is available as of CNC Build V3.1.3067.01.**

```
Syntax example of tapping in Z direction:
```

G331 Z.. **K**.. *<spindle_name>*..	Thread tapping	modal
G332 Z.. [**K**..] [*<spindle_name>*..]	Thread tapping, retract	modal

G331	Thread tapping
Z..	Thread depth (target point) in the tapping axis in [mm, inch]
K..	Thread pitch in assigned interpolation parameter in [mm/rev, inch/rev]
<spindle_name>..	Spindle speed consisting of spindle name according to P-CHAN-00053 and speed value in [rpm]
G332	Retract from threaded bore (Retract). G332 causes an automatic direction reversal of the spindle on retraction.
Z..	Retract position of tapping axis after tapping in [mm, inch]
K..	Thread pitch in assigned interpolation parameter in [mm/rev, inch/rev]. The thread pitch must be the same pitch as used for the threaded bore assigned in G331. The parameter is optional. If not programmed, the pitch in block G331 applies.
<spindle_name>..	Spindle speed consisting of spindle name according to P-CHAN-00053 and speed value in [rpm]. The parameter is optional. If not programmed, the speed in block G331 applies.

This type of tapping (G331/ G332) requires a position-controlled spindle which is tracked by the CNC synchronous to the path motion. In this case the spindle and the feed motion of the participating axes are matched precisely and dynamically. A compensatory chuck is not required.

The thread type is defined by specifying a sign for thread pitch.

- Pitch without or with positive sign (+): Right-hand thread, e.g. K2 or K+2
- Pitch with negative sign (-): Left-hand thread, e.g. K-2

The thread tapping axis feedrate is a product of the programmed pitch and the spindle speed. The permissible speed limits apply to the internal calculation. An error message is output if these limits are violated.

The feed rate continues to apply after tapping is completed. With the following G331/G332, the feedrate is again calculated from the related programmed or saved values of pitch and spindle speed.

G331/G332 is deselected by selecting a different modal block type (e.g. linear motion G01) and the spindles are released from the coordinated motion. A non-modal block type (e.g. dwell time with G04) does not deactivate G331/ G332.

An error message is output if the pitch or spindle speed with G331/G332 are equal to zero or the tapping axis and pitch parameters fail to match. Valid combinations are X with I, Y with J and Z with K.

M03, M04, M05, M19 cannot be programmed in combination with G331/G332.

NOTICE
The spindle (or the thread tapping drill) must be at standstill when G331 is selected. This can be achieved by previously programming M05 (Stop spindle) or M19 with S.POS (Position spindle).

Tapping (G331/ G332)

Tap right-hand thread with pitch 2 mm, thread depth 50 mm, spindle speed S 200 rpm, Z is tapping axis:

```
;…
G01 F2000 G90 X0 Y0 Z0 ; position axes
M19 S.POS=0 M3 S100    ; stop and position spindle
;…
G331 Z-50 K2 S200      ; tap in Z
G332 Z10 K2 S200 ; retract
G01 F1000 X50          ; reposition, deselect tapping
G331 Z-50 K2 S200      ; tap in Z
G332 Z10  K2 S400      ; retract at increased speed
G01 F1000 X100         ; reposition, deselect tapping
G331 Z-50 K2 S200      ; tap in Z
G332 Z10               ; retract, K and S from G331
G01 F1000 X150         ; reposition, deselect tapping
;…
```

Tap right-hand thread with pitch 1.5 mm, thread depth 60 mm, spindle speed S 150 rpm, X is tapping axis:

```
;…
G01 F2000 G90 X100 Y0 Z0 ; position axes
M19 S.POS=0 M3 S100    ; stop and position spindle
;…
G331 X40 I1.5 S150     ; tap in X
G332 X110 I1.5 S150 ; retract
:
```

Tap at relative speed

```
%Tap at relative speed

N010 G91 G19 G0 X100 M03 S2000
N020 S2[MC_GearIn Master=S1 \ ; couple tool
RN=1 RD=1 Mode=256 \          ; spindle S2 to the
PhaseShift=1800000 WAIT_SYN] ; main spindle S1

N030 #MAIN SPINDLE[S2]        ; main spindle tool spindle S2
N040 G331 Z-100 K1.5 S200     ; tap right-hand thread
N050 G332 Z100 K1.5 S200      ; retract from threaded bore
N060 G01 X300 F1000
N070 S2[MC_GearOut WAIT_SYN]  ; release coupling to main spindle
N080 #MAIN SPINDLE[S1]
N090 M30
```

```
%Tap at relative speed
```

N040 G331 Z-100 K1.5 S200   ; tap right-hand thread

15.2.7 C axis machining

This functionality supplements the existing turning functions and permits the face and lateral surface machining of cylindrical workpieces on lathes and milling machines with revolving base. The workpiece is moved by the rotary axis or spindle (C axis) and the driven tool (e.g. milling cutter) by the two translatory axes X (or Y) and Z. Settings are required in the parameters P-CHAN-00008 and P-AXIS-00015 for C axis machining.

Face and lateral surface machining can be described in Cartesian coordinates.

All interpolation types (such as linear, circular or spline interpolation) are supported on the end face and lateral surface. The functionality also permits the machining of path contours running through the turning centre point. The C axis is automatically aligned on lathes.

The 2.5D tool radius compensation can be used with the familiar G commands.

The use of extended dynamic monitoring can specifically prevent dynamic axis characteristics from being exceeded with the C axis function and also with contours running close to the turning centre point.

The main axes for all machining modes are X, Y (depending on machine type), Z and C.

15.2.7.1   Exchange spindles in coordinated motion (# CAX, #CAX OFF)

This "basic mode" is required in particular for C axis machining on lathes because in this case the position-controlled spindle has to be converted into a rotary path axis (e.g "C").

> **ⓘ**   C axis machining can also be executed on milling machines or machining centres which are designed with rotary workpiece fixtures (e.g. turntable). In this case, it is not necessary to select #CAX.

The three physical axes X, Y, Z and the C axis replaced in coordinated motion can be directly programmed. Linear axes are programmed in Cartesian coordinates and the C axis in angle units.

Radius and diameter programming depends on G52/G51.

Two linear axes define the main plane: ZX (G18) or YZ (G19).

Syntax:

#CAX [**[** [*<main_spindle_name>*,] *<C_axis_name>* **]**]

<main_spindle_name>	Only the main spindle name can be programmed according to P-CHAN-00053. If a spindle other than the C axis is used, it must first be declared as the main spindle (see programming example in Section [PROG// Changing the main spindle [▶ 649]]). Otherwise, an error message is output.
<C_axis_name>	Freely definable name of the C axis in the NC program. If no C axis name is programmed, the default name from P-CHAN-00010 is used.

The main plane (circular interpolation, tool radius compensation, etc.) remains the same as before activation of the C axis.

An error is generated if a command for this spindle (M3, M4, M5 [▶ 590], etc.) is programmed although the axis is still declared as a C axis in the coordinated motion.

The C axis is deselected, i.e. the axis is released to the spindle interpolator, by the following:

Syntax:

#CAX OFF

C axis machining

Exchange spindles in coordinated motion

```
;…
#CAX                        ; Assuming: default C axis is "C"
G01 G90 X50 Z10 C90 F200
#CAX OFF                    ; deselect C axis mode
;...
#CAX[S, C] or #CAX[C]       ; Assuming: main spindle is "S"
G01 G90 X50 Z10 C90 F200
#CAX OFF
;...
#MAIN SPINDLE [S2]          ; "S2" becomes new main spindle "S"
#CAX[S, C]                  ; select C axis mode
G01 G90 X50 Z10 C90 F200
;...
#CAX OFF                    ; deselect C axis mode
;...
#CAX[S3, C]; Error, "S3" is not the main spindle
```

15.2.7.2   Face machining (#FACE, #FACE OFF)

This mode is selected for lathes and machining centres. The desired contour on the face is programmed in millimetres or inches using a virtual Cartesian coordinate system.

Fig. 163: Face machining

> **ℹ** **As of CNC build V3.00, the parameter P-CHAN-00262 must be assigned with the kinematic ID used, depending on P-CHAN-00008, in order to perform face machining applications.**
> - For face transformation 1 with P-CHAN-00008=1 - **ID 13**
> - For face transformation 2 with P-CHAN-00008=2 - **ID 14**

The three logical axes X, Y (or C) and Z are provided to program the contour on the face in Cartesian coordinates.

Fig. 164: Front view of face machining process

The figure below shows each of the main planes in face machining. Only the G17 plane is of technological importance.

Fig. 165: Main places of face machining

```
Syntax:
```
#FACE [*<name 1. main_axis>***,** *<name 2. main_axis>* **]**

<name of 1st main axis>	Name of the first main axis according to the current main plane.
<name of 2nd main axis>	Name of the second main axis according to the current main plane (virtual Cartesian axis).

When selected. the main plane (circular interpolation, tool radius compensation, etc.) is always defined by the 1st and 2nd main axes (G17). It is not permitted to change the main plane with G18, G19 while face machining is active.

ℹ️ Programmed tracking axes are not affected by the transformation.

This mode is deselected by:

```
Syntax:
```
#FACE OFF

The above command returns to the last active mode (e.g. mode 1). This means that the last active main plane is selected automatically and the last active axis offsets are restored.

Programming example for lathes

Example with axis name "C" for second main axis. main axis

```
; …
#CAX[S, C]                   ; Assuming: main spindle is "S"
#FACE[X, C]                  ; select face machining
; …
G01 X40 C-30 Z50 F1000       ; pre-position
G01 Z30                      ; approach
G01 X10 C40                  ; travel contour
G01 Z50                      ; retract
; …
#FACE OFF
#CAX OFF
; …
M30
```

Example with axis name "Y" for second main axis. main axis.

Note: No other axis with the identical name "Y" may exist in NC channel.

```
; …
#CAX[S, Y]                   ; Assuming: main spindle is "S"
#FACE[X, Y]                  ; select face machining
; …
G01 X40 Y-30 Z50 F1000       ; pre-position
G01 Z30                      ; approach
G01 X10 Y40                  ; travel contour
G01 Z50                      ; retract
; …
#FACE OFF
#CAX OFF
; …
M30
```

Programming example for machining centres

The rotary axis (workpiece axis) in the channel is "C2". It is not necessary to program the #CAX command.

```
;…
#FACE[X, C2]                  ; select face machining
;…
G01 X40 C2=-30 Z50 F1000      ; pre-position
G01 Z30                       ; feed
G01 X10 C2=40                 ; travel contour
G01 Z50                       ; retract
;…
#FACE OFF
;…
M30
```

15.2.7.3 Surface machining (#CYL, #CYL OFF)

This mode can be selected for lathes and machining centres. The desired contour on the cylindrical surface is programmed in millimetres or inches using a virtual coordinate system.

Fig. 166: Lateral surface machining

> **ℹ** **As of CNC build V3.00, the parameter P-CHAN-00262 must be assigned with the vale 15 for this transformation in order to perform lateral surface machining applications.**

The three logical axes X, Y, Z are provided to program the contour on the lateral surface in Cartesian coordinates. In this mode, the reference radius R of the workpiece must also be programmed.

The main plane in lateral surface machining is formed by Z-C.


```
Syntax:
```

#CYL [<1st_main_axis_name>**,** <2nd_main_axis_name>**,** <3rd_main_axis_main> etc.**]**   modal

<1st_main_axis_name>	Name of the first main axis according to the current main plane.
<2nd main_axis_name>	Name of the second main axis according to the current main plane (virtual linear axis, development).
<3rd_main_axis_name>.	Axis name of the third main axis according to the current main plane with specification of the reference radius in [mm, inch].

When selected. the main plane (circular interpolation, tool radius compensation, etc.) is always defined by the 1st and 2nd main axes (G17). It is not permitted to change the main plane by G18, G19 while lateral surface machining is active.

i Programmed tracking axes are not affected by the transformation.

This mode is deselected by:

```
Syntax:
```
#CYL OFF

The above command returns to the last active mode (e.g. mode 1). This means that the last active main plane is selected automatically and the last active axis offsets are restored.

Programming example for lathes

Example with axis name "C" for second main axis. main axis

```
;…
#CAX [S, C]        ; Assuming "S" is main spindle
G01 X60 C45        ;feed and positioning movement; X:60mm C:45°
#CYL [Z, C, X60]   ;select lateral surface machining
G00 G90 Z0 C0      ;Z: 0mm C:0mm!
G01 C100 F500
G02 Z100 R50
G01 C0
Z0
;…
#CYL OFF
#CAX OFF
M30
```

15.2.7.4   Switching between face and lateral surface machining

All C axis modes are deselected normally using the deselection commands described (for example using the #CYL OFF command). It is also permitted to change directly to a different machining mode, e.g. between face and lateral surface machining, without previously deselecting the current active mode. The programming example below shows a typical NC sequence for changing between C axis modes:

Switch between C axis modes

```
N10 #CAX [..]      ;Adopt the LR spindle in the coordinated motion
;..............
N120 #FACE [..]    ;Select face machining
;..............
N230 #CYL [..]     ;Direct transition to lateral surface machining
....... ..
N300 #CYL OFF      ;Deselect lateral surface machining and transition
                   ;to conven. machining with physical C axis
N400 #CAX OFF      ;Return the C axis to the position-controlled spindle
N500 M30
```

15.2.7.5   Tool offsets

The commands #FACE and #CYL result in the implicit selection of kinematics. For this reason, neither a kinematic ID requires a #KIN ID [▶ 683] [..] nor transformation activation with #TRAFO ON [▶ 677].

Tool offsets for face machining

Face machining supports 2 machine types (lathe/milling machine). The corresponding tool offsets must be entered in the channel parameters in the assigned offset data of the kinematic IDs 13 and 14. Alternatively, this can also be executed in the tool data.

Fig. 167: Tool offsets for face machining

Examples of entries in channel parameters

for CNC Builds as of V3.00

FACE[], Face machining on a <u>lathe</u> (KIN-ID 13):

```
trafo[0].id                 13
trafo[0].param[0]           1080000   Z offset [0.1µm]
trafo[0].param[1]           0         C angular offset [10⁻⁴°]
trafo[0].param[2]           900000    X offset [0.1µm]
```

FACE[], Face machining on a <u>milling machine</u> (KIN-ID 14):

```
trafo[1].id                 14
trafo[0].param[0]           1080000   Z offset [0.1µm]
trafo[0].param[1]           0         C angular offset [10⁻⁴°]
trafo[1].param[2]           900000    X offset [0.1µm]
```

CNC Builds < V3.00

FACE[], Face machining on a <u>lathe</u> (KIN-ID 13):

```
kinematik[13].param[0]      1080000   Z offset [0.1µm]
kinematik[13].param[1]      0         C angular offset [10⁻⁴°]
kinematik[13].param[2]      900000    X offset [0.1µm]
```

FACE[], Face machining on a <u>milling machine</u> (KIN-ID 14):

```
kinematik[14].param[0]      1080000   Z offset [0.1µm]
kinematik[14].param[1]      0         C angular offset [10⁻⁴°]
kinematik[14].param[2]      900000    X offset [0.1µm]
```

Tool offsets for lateral surface machining

Lateral surface machining implicitly executes the selection of the kinematic with ID 15.

Fig. 168: Tool offsets for lateral surface machining

#CYCL[], Lateral surface machining lathe (KIN-ID 15):

CNC Builds as of V3.00

```
trafo[0].id              15
trafo[0].param[0]        700000    X offset [0.1µm]
trafo[0].param[0]        0         C angular offset [10⁻⁴°]
trafo[2].param[0]        1200000   Z offset [0.1µm]
```

CNC Builds < V3.00

```
kinematik[15].param[0]   700000    X offset [0.1µm]
kinematik[15].param[1]   0         C angular offset [10⁻⁴°]
kinematik[15].param[2]   1200000   Z offset [0.1µm]
```

15.2.8   Gear change (G112)

Syntax:

G112 *<spindle_name>*.   non-modal

G112   Gear change

<spindle_name>..   Gear data block consisting of the spindle name as described in the channel
                         parameter list and number of the data block

As opposed to gear changing via M40-45 in which the mechanical gear change operation is also performed implicitly, programming G112 together with the S word only triggers the update of the spindle gear data (dynamic values) of one step.

The user must explicitly program the mechanical changing of the correct gear stage, e.g. by self-defined M functions or as mentioned above using M40-45.

NOTICE
The spindle must be at standstill before the spindle gear data are switched over. This can be achieved in the previous NC block by programming a spindle stop (M5) or a spindle positioning (M19…)

Gear change (G112)

```
%Test_G112
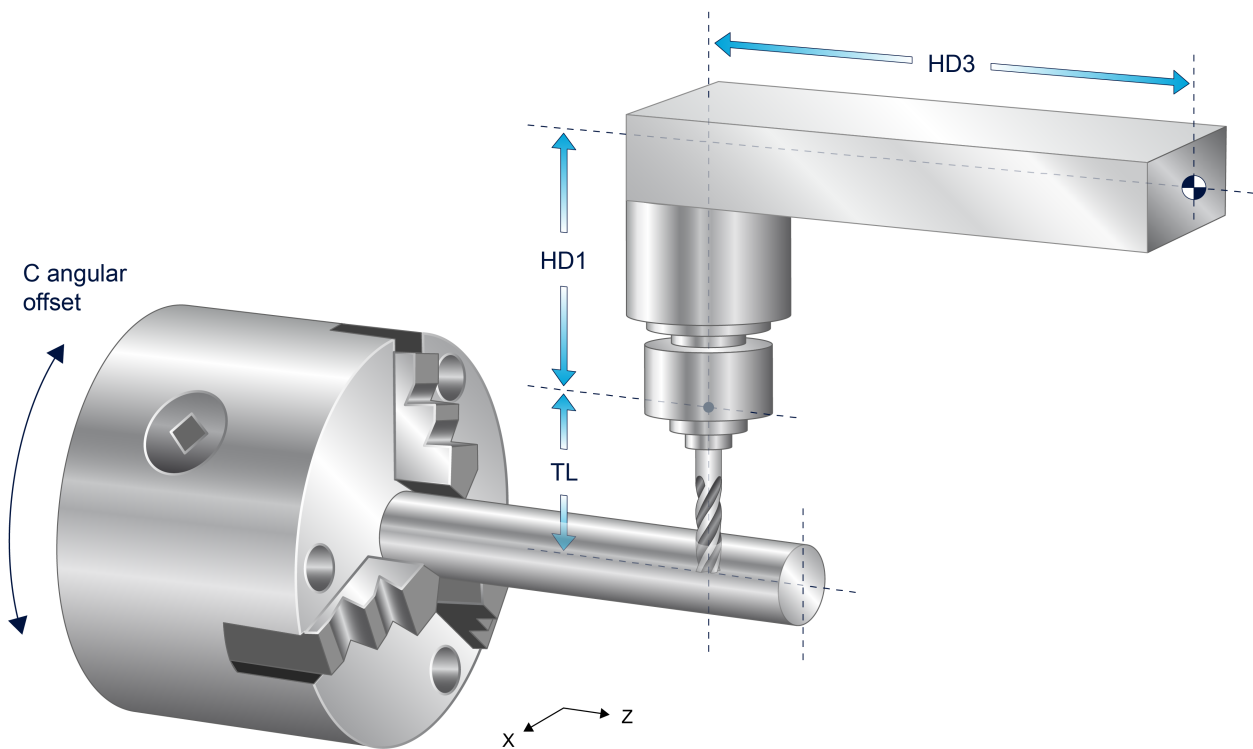
N010 G112 S2    (Load dynamic data of gear stage 2 for spindle "S")
N020 M3 S9000
N030 M5         (Stop spindle)
N040 G112 S1    (Load dynamic data of gear stage 1 for spindle "S")
N050 M4 S8000
N060 M30
```

15.2.9   Homing in DIN syntax(G74)

Syntax:

G74 *<spindle_name>*.   non-modal

G74   Homing

<spindle_name>..   Spindle name according to P-CHAN-00053 with specification of a value.

Homing can be conducted for closed-loop position-controlled spindles. As opposed to referencing linear axes, the values programmed with the spindle name have no significance relating the referencing sequence and are only required to represent a complete syntax.

It is not permitted to program spindle M functions in the same NC block as G74.

A distinction must be made in the following cases when referencing spindles:

Homing (G74)

```
Case 1:
Spindle referencing starts simultaneously with
Y axis referencing:
  Nxx G74 X2 Y1 S1

Case 2:
Same as 1.! The system continues to the next NC block without waiting until the spindle is
referenced so that the X axis is referenced quasi simultaneously:
  Nxx G74 S1
  Nyy G74 X1 Y2

Case 3:
Axes X and Y are first referenced. Spindle referencing then starts:
  Nxx G74 X1 Y2
  Nyy G74 S1
```

15.2.10   Spindle override in DIN syntax (G167)

G167 *<spindle_name>*.   non-modal

G167	Set spindle override to 100%
<spindle_name>..	Spindle speed consisting of spindle name according to P-CHAN-00053 and specification of a value. The value has no significance but is only required to represent a complete syntax.

The G167 function deactivates external influencing of spindle override and implements the speed actually programmed.

Override (G167)

```
:
N10 M3 S1000      (When override is 50%, speed is 500 rpm)
N20 G167 S1000    (Override influencing off, speed is 1000 rpm)
N30 S3000         (Override influencing active again, speed is 1500 rpm)
Nxx
```

15.3   Programming in spindle-specific syntax

Spindle-specific syntax offers the advantage that several spindles can be programmed mutually independent in the same NC block.

This is carried out within a bracketed expression attached to the spindle name. Only specific commands are permitted in this bracketed expression and these commands are always handled and executed spindle-specific. The main spindle can only be programmed by its main spindle name (P-CHAN-00053).

```
Syntax:
```
<spindle_name> **[** [**M3** | **M4** | **M5** | **M19**] [**REV**..] [**POS**..] { **M**.. } { **H**.. } [**G74**] [**G167**]

　　　　　　　[**CALLAX** | **PUTAX**] [**GET_DYNAMIC_DATA**] [**G130**] [[**G135** | **G137**] [**G136**..]]

　　　　　　　　[**FEED_LINK**..] [**OVERRIDE**..] { **** } **]**

　　　　　　　{ *<spindle_name>* **[** .. **]** }

<spindle_name>	Spindle name according to [1] [▶ 786]-3 and P-CHAN-00053
M3, M4, M5, M19	Spindle M functions
REV=..	Spindle speed
POS=..	Spindle position
M=..	User-specific M functions
H=..	User-specific H functions
G74	Homing
G167	Spindle override 100%
CALLAX	Call spindle axis
PUTAX	Release spindle axis
GET_DYNAMIK_DATA	Adopt new tool dynamic data
G130	Acceleration weighting
G135, G136=.., G137	Feedforward control
FEED_LINK…	Spindle feed link
OVERRIDE…	Spindle override
\	Separator ("backslash") for clear programming of the command over multiple lines.

Programming in spindle-specific syntax

```
:
N10 S[M3 REV500 M19 POS45 M18 M15 H20 …]    S2[M4 REV5000]
Nxx
:
```

15.3.1   The spindle M functions

15.3.1.1   Moving spindle in spindle-specific syntax ((M3/M4/M5)

Syntax:

M03	Spindle rotation clockwise (cw)	modal
M04	Spindle rotation counter-clockwise (ccw)	modal
M05	Stop spindle	modal

The spindle M functions M03 and M04 define the spindle direction of rotation and must be used in conjunction with the spindle speed (REV word). M05 stops spindle rotation. Note that this spindle M function is the default spindle mode after controller start-up and initial program start. These M functions are modal and may only be programmed on their own within the bracketed expression.

Spindle rotation is activated if M03 or M04 are programmed and a valid speed (REV) is set.

If no M05 is set at program end, the spindle continues to rotate.

Programming of one spindle "S":

```
N10 S[REV1000]      (Speed 1000 rpm is stored,)
                    (no spindle rotation because M05 is default)
N20 S[M03]          (Spindle rotation cw at 1000 rpm)
N30 S[M04]          (Spindle rotation ccw at 1000 rpm)
N40 S[REV500]       (Spindle rotation ccw at 500 rpm)
N50 S[M05 REV300]   (Spindle stop, speed 300 rpm is stored)
N60 S[M04]          (Spindle rotation ccw at 300 rpm)
N70 S[M05]          (Spindle stop)
N80 S[M03 REV1000]  (Spindle rotation cw at 1000 rpm)
N90 M30             (Program end)
```

Programming two spindles "S2"and "S2":

```
N10 S[M03 REV1000]  S2[M04 REV2000]  (S cw 1000 rpm,S2 ccw 2000 rpm)
N20 S[M05]          S2[REV1500]      (S stop, S2 ccw at 1500 rpm)
N30 S[M04]                           (S ccw 1000 rpm)
N40                 S2[M05]          (S2 stop)
N50 S[M05 REV300]                    (S stop, store speed 300 rpm)
N60 S[M04]          S2[M04]          (S ccw 300 rpm, S2 ccw 1500 rpm)
N70 S[M05]          S2[M05]          (S stop, S2 stop)
N80 M30                              (Program end)
```

Channel parameter list [1] [▶ 786]:

The synchronisation modes must be defined spindle-specific for M3, M4, M5. The M function is not executed for synchronisation mode "0" (NO_SYNCH).

```
:
spindel[0].bezeichnung                              S1
spindel[0].log_achs_nr                              6
spindel[0].s_synch                                  0x00000001
spindel[0].m3_synch                                 0x00000002
spindel[0].m4_synch                                 0x00000002
spindel[0].m5_synch                                 0x00000008
spindel[0].m19_synch                                0x00000001
```

```
N10 S[M03 REV1000]  S2[M04 REV2000]  (S cw 1000 rpm,S2 ccw 2000 rpm)
N20 S[M05]          S2[REV1500]      (S stop, S2 ccw at 1500 rpm)
N30 S[M04]                           (S ccw 1000 rpm)
```

15.3.1.2 Positioning spindle in spindle-specific syntax (M19, POS)

`Syntax:`

M19	Positioning spindle	non-modal
POS=..	Spindle position in [°]	modal

The spindle M function M19 executes spindle positioning and must be used in conjunction with the POS word. M03/M04 and the spindle speed (REV) in the same NC block are optional. However, a valid spindle speed (> 0) must be set. M19 may not be used together with M5 (spindle stop).

Spindle position POS in [°] is modal and need not be respecified if M19 is programmed again. If no spindle position was previously programmed, the motion is moved to position "zero" by default.

If the spindle is not rotating, positioning is executed with the shortest motion path.

Spindle positioning with M19 is only permitted for position-controlled spindles.

Programming of one spindle "S”:

```
N10 S[REV100 POS45 M19 M3]     (Move cw at 100 rpm to position 45)
N20 S[M03 REV1000]             (Spindle rotation cw at 1000 rpm)
N30 S[M19 M4 REV150]           (Move ccw at 150 rpm to position 45)
                               (->POS is modal!)
N40 S[M05]                     (Spindle stop)
N50 S[M19]                     (Move on shortest path at 150 rpm to)
                               (position 45)
N60 S[REV200 M19 POS90]        (Move on shortest path at 200 rpm to)
                               (position 90)
N70 S[M03 REV1000]             (Spindle rotation cw at 1000 rpm)
N80 S[M05]                     (Spindle stop)
N90 M30            (Program end)
```

Programming two spindles "S2"and "S2":

```
(For N10: Move both spindles cw at 1000 rpm to position 45)
N10 S[M03 REV200 M19 POS45] S2[M04 REV200 POS45 M19]
N20 S[REV1000] S2[REV1500]              (S E FAC 1000 rpm)
                                        (S2 ccw at 1500 rpm)
N30 S[M04]                      (S ccw 1000 rpm)
N40 S2[M19 POS0]                        (Move S2 ccw to position 0)
N50 S[M05 REV300]               (S stop, store speed 300 rpm)


(For N60: Move S cw at 300 rpm to position 90,)
(For N60: Move S2 cw at 200 rpm to position 45)
N60 S[M03 M19 POS90] S2[M03 REV200 POS45 M19]
N80 M30                                 (Program end)
```

Channel parameter list :

The synchronisation mode must be defined spindle-specific for M19. The M function is not executed for synchronisation mode "0" (NO_SYNCH).

```
:
spindel[0].bezeichnung                          S1
spindel[0].log_achs_nr                          6
spindel[0].s_synch                              0x00000001
spindel[0].m3_synch                             0x00000002
spindel[0].m4_synch                             0x00000002
spindel[0].m5_synch                             0x00000008
spindel[0].m19_synch                            0x00000001
```

15.3.2   Spindle speed (REV)

```
Syntax:
```

| **REV**=.. | Spindle speed in [rpm]. | modal |

Values can be assigned to the REV word directly or by means of parameters in [rpm] and decimal numbers are also permitted (REAL format).

A distinction must be made between the following types of use in conjunction with the spindle M functions in the case of the REV word:

1. REV word in conjunction with M03, M04, M19:
   If the REV word is programmed in conjunction with M03/M04 or M19, the value following the REV word is interpreted as the spindle speed and is output to the spindle.
2. REV word in conjunction with M05:
   Together with M05, the value following the REV word is transferred to the working data of the spindle speed but is not output to the spindle.

The REV word on its own does not generate a motion in the NC program. This requires that a spindle mode M03, M04, M19 is known. Accordingly, programming of M03, M04 and M19 only results in a motion if the REV word is set.

> **i** An error message is output if the REV value is negative.

Programming with spindle S1:

```
N10 S1[REV300]        (Speed 300 rpm is stored)
N20 S1[M04]           (Spindle rotation ccw at 300 rpm)
N30 S1[M03 REV1000]   (Spindle rotation cw at 1000 rpm)
N40 S1[REV500]        (M03 active, therefore spindle rotation cw at 500 rpm)
N50 S1[M05 REV100]    (Spindle stop, speed 100 rpm is stored)
N60 S1[M04]           (Spindle rotation ccw at 100 rpm)
N70 S1[M05]           (Spindle stop)
N80 M30               (Program end)
```

Channel parameter list [1] [▶ 786]:

The synchronisation mode must be defined spindle-specific for the S word. An error message is generated in the case of synchronisation mode "0" (NO_SYNCH) since an S word may not be ignored.

```
:
spindel[0].bezeichnung                          S1
spindel[0].log_achs_nr                          6
spindel[0].s_synch                              0x00000001
spindel[0].m3_synch                             0x00000002
spindel[0].m4_synch                             0x00000002
spindel[0].m5_synch                             0x00000008
spindel[0].m19_synch                            0x00000001
```

15.3.3 User-specific M/H function in spindle-specific syntax

All user-specific M/H functions (technology information) programmed within the bracket expression are **always** handled and output spindle-specific.

Program one spindle "S2" with two user-specific M functions:

```
N10 S2[M3 REV300 M10 M11](S2 cw rotates at 300 rpm and outputs M10/M11)
N20 M30                  (Program end)
```

Program two spindles "S" and S2:

```
N10 S[M10 M11] S2[REV1000 M3 M11]  (S executes M10 and M11, S2 rotates)
                                   (at 1000 rpm and outputs M11)
N20 M30                            (Program end)
```

Channel parameter list [1] [▶ 786]:

The user-specific M/H functions are defined in P-CHAN-00027 and P-CHAN-00041. The M/H function is not executed in the case of synchronisation mode "0" (NO_SYNCH).

```
:
# Definition of M functions and synchronisation types
:
# Definition of M functions and synchronisation types
m_synch[1]                  0x00000001                  MOS
m_synch[2]                  0x00000002                  MVS_SVS
:
m_synch[10]                 0x00000002                  MVS_SVS
m_synch[11]                 0x00000008                  MVS_SVS
m_synch[12]                 0x00000004                  MVS_SVS

m_synch[48]                 0x00000008                  MNS_SNS
m_synch[49]                 0x00000002                  MVS_SVS
:                           :                           :
# Definition of H functions and synchronisation types
h_synch[1] 0x00000001 MOS
h_synch[2] 0x00000001 MOS
:
```

15.3.4 Homing in spindle-specific syntax (G74)

Homing can be executed spindle-specific. Programming the spindle M functions is not permitted together with G74.

Programming two spindles "S2"and "S2":

```
N10 S[G74] S2[G74]          ;Homing S and S2
N20 M30                      ;Program end
```

15.3.5 Spindle override in spindle-specific syntax (G167)

The G167 function deactivates external influencing of spindle override spindle-specific and executes the actually programmed rotational speed. The effect of the programmed override value for one spindle [▶ 647] is retained.

Programming one spindle "S2":

```
N10 N10 [M3 REV1000]    ;If override is 50%, speed is 500 rpm
N20 S2[G167 REV1000]  ;Override influencing off,
                        ;spindle rotates at 1000 rpm
N30 S2[REV3000]         ;Override influencing active again,
                        ;speed 1500 rpm
N40 M30                 ;Program end
```

15.3.6   Releasing/requesting spindle axes (PUTAX/CALLAX)

ℹ️ **The availability of this function depends on the configuration and the scope of the version.**

These commands can be used to program the release or call of the spindle axis spindle-specific. The commands may not be used simultaneously with other spindle-specific commands.

```
%s-putcallax
(Move axis as spindle)
N10  S[CALLAX]
N20  M03 S1000
N30  G04 X2
N40  M05

(Exchange axis from spindle interpolator in channel)
N50  S[PUTAX]
N60  #CALL AX[ C, 4, 3]
(Drive axis in channel)
N70  X10 Y20 Z30 C40
N80  X-10 Y-20 Z-30 C--40
(Exchange axis from channel in spindle interpolator)
N90  #PUT AX[C]
N100 S[CALLAX]
(Move axis again as spindle)
N110 M04 S1000
N120 G04 X2
N130 M05
N140 M30
```

Programming multiple spindles:

```
:
N10 S[CALLAX] S2[CALLAX] S3[PUTAX] (S, S2 call their axes,
                                   (S3 releases its axis)
N20 S[M3 REV200] S2[G74]          (S rotates cw at 200 rpm),
                                   (S2 is homing)
N30 S3[ M4 REV3000]        (Error, S3 currently has no axis)
N40 S[PUTAX REV400)        (Error, PUTAX may not be programmed alone)
:
```

15.3.7   Adopt tool dynamic data (GET_DYNAMIC_DATA/ DEFAULT_DYNAMIC_DATA)

i   The availability of this function depends on the configuration and on the version scope.

The tool dynamic data (minimum/maximum speed, max. acceleration) takes effect automatically on transition of the spindle from standstill to interpolation after programming a new tool (D word, #TOOL DATA). Changed tool dynamic data is adopted and considered for a rotating spindle by the spindle-specific command "GET_DYNAMIC_DATA".

When a spindle is at standstill, a change back to tool-independent default dynamic data can be executed by the command "DEFAULT_DYNAMIC_DATA". The currently active gear stage (P-TOOL-00016/P-TOOL-00017) is not changed.

These commands may not be used simultaneously or in combination with other spindle-specific commands.

Adopt dynamic tool data

```
N10 T1 D1                  ;Supply tool dynamic data to the
                           ;tool data
N15 M6                     ;Change of tool 1
N20 S[M3 REV2002]          ;S rotates at 2000 rpm with tool dynamic data
                           ;from D1
N25 X100 Y100                  ;Motion block with tool dynamic data D1
N30 T99 D99                ;Supply tool dynamic data to the
                           ;tool data while spindle rotates (N20)
N35 X200 Y150             ;Motion block with tool dynamic data D1
N40 S[GET_DYNAMIC_DATA]    ;Adopt tool dynamic data D99
N45 X150 Y200              ;Motion block with tool dynamic data D99
N50 X50 Y50               ;Motion block with tool dynamic data D99
N60 S[M05] Z100            ;Spindle stop
N70 S[DEFAULT_DYNAMIC_DATA] ;Set to default dynamic data

N99 M30                    ;Program end
```

15.3.8   Commanding spindle feedforward control (G135/G136/G137)

i   The availability of this function depends on the configuration and on the version scope.

These commands permit the spindle-specific programming of feedforward control. The commands may not be used simultaneously with other spindle-specific commands.

Activation is programmed with G135.

Spindle-specific, percentage weighting of the calculated feedforward control variables takes place with G136. It is limited to 100%.

G137 deactivates feedforward control. It is also possible to specify the selection and weighting of the feedforward control in the same block.

If feedforward control is disabled or enabled during the NC program, the weighting factors remain at the values set by G136 or, if no G136 is programmed, to 100%.

Commanding spindle feedforward control

```
S[G135]                ;Activate feedforward control for S
S[G136=80]             ;Define weighting in percent
S[G137]                 ;Deactivate feedforward control
S2[G135 G136=90] ;Activate at 90% weighting for S2
S2[G136=0]             ;Change weighting to 0%
S1[G135]                Activate at 100% default weighting for S1
```

15.3.9   Spindle feed link (FEED_LINK)

Normally, the speed of a spindle is only controlled by the program. In general, there are no other options to influence spindle speed depending on other parameters. For some special technologies (e.g. HSC milling, edge banding in wood machining) and to obtain a uniform milling profile on the workpiece surface, it is necessary to influence spindle speed by external variables.

The following command can link the spindle-specific speed dynamically to path feed by specifying various parameters. When set accordingly, the spindle speed then adapts automatically to various feedrates. This is necessary in particular for materials which may be damaged by unsuitable cutting parameters (e.g. scorch marks on wooden workpieces or melted plastic etc.).

NOTICE

Spindle dynamic data are not considered. To avoid excessive deviation between the commanded and the actual link factors during active feed linking, the user must ensure that either the spindle dynamic is high enough or the path dynamic is reduced depending on the requirements.

`Syntax for programming a spindle feed link:`

<spindle_name> **[FEED_LINK** [**ON** | **OFF**] [[[**FACT=**..] [**CORR=**..]] | **AUTO**] [**MAIN**] **SRC=**.. **]**

<spindle_name>	Name of the feed link spindle
FEED_LINK	Identifier for spindle feed link. Must always be programmed as the **first** keyword.
ON	Select spindle feed link
OFF	Deselect spindle feed link
FACT=..	**Synchronised** link: Predefined link factor in [0.1%] between path feed (F word) and the basic spindle speed. The link factor can be reprogrammed at any time during active linking.
CORR=..	Correction factor in [0.1%] to modify the link factor. The resulting link factor then results from the product of FACT and CORR (priority = FACT*CORR).
AUTO	**Link on the fly**: The link factor is calculated automatically from the current commanded spindle speed and the current path feed and it remains constant until the link is deselected.
MAIN	Spindle feed link is only effective if at least one main axis is involved in motion. The spindle speed is not influenced if only tracking axes are moved.
SRC=..	Identifier of feed link source: • FEED_VEL – path feed (default) • EDGE_VEL – edge velocity for edge banding

- The keyword AUTO can only be programmed in combination with ON. FACT and CORR can even be reprogrammed while a feed link is active.

- The keyword MAIN can only be programmed in combination with ON.

- When the link is selected, the spindle must be in the state of endless rotation at a sped of !=0. The path feed must also be !=0 when the link factor is defined automatically.

- The spindle speed which is active when the feed link is deselected is retained as the commanded spindle speed. If necessary, the spindle speed must be redefined using the S word after the feed link is deselected.

NOTICE
A spindle command via the S word has no effect when the feed link is active.

> **ℹ** A tool with dynamic parameters is to be defined and activated. The minimum speed of this tool (P-TOOL-00013) can in general prevent the spindle from coming to a standstill or jamming while milling with active feed link at path velocity = 0.

Example 1

Manually programmed link factor, the effective link factor is changed by programming the correction factor

```
%feed_link1

N10 G00 X0 Y0 Z0
N20 G01 G90 X20 F2
N30 M03 S15
N40 G01 G90 X40
N50 S[FEED_LINK ON FACT=500]
N60 G01 G91 X5 F2
N70 S[FEED_LINK CORR=1100]
N80 X10 F2
N90 S[FEED_LINK CORR=900]
N100 X20 F2
N110 S[FEED_LINK CORR=1100]
N120 X40 F2
N130 S[FEED_LINK CORR=900]
N140 X80 F2
N150 S[FEED_LINK CORR=1100]
N160 X40 F2
N170 S[FEED_LINK CORR=900]
N180 X20 F2
N190 S[FEED_LINK CORR=1100]
N200 X5 F2
N210 S[FEED_LINK OFF]
N220 M30
```

Example 2

Link factor is generated automatically from the current commanded spindle speed and the current path feed.

```
%feed_link2

N10 G00 X0 Y0 Z0
N20 G01 G90 X20 F1
N30 M03 S15
N40 G01 G90 X40
N50 S[FEED_LINK ON AUTO]
N60 G01 G91 X5 F1
N70 X10 F2
N80 X20 F4
N90 X40 F8
N100 X80 F16
N110 X40 F8
N120 X20 F4
N130 X10 F2
N140 X5 F1
N150 S[FEED_LINK OFF]
N160 M30
```

Example 3

Link factor is generated automatically, feed link only acts with main axis motions.

```
%feed_link3

N10 M03 S200
N20 G01 X0 Y0 Z0 C0 F10
N30 S[FEED_LINK ON AUTO MAIN]
N40 G01 X50
N50 G01 X70 Y30
N60 G01 C45 ;Spindle speed remains constant
N70 G01 Z40 F15
N80 G01 Z60
N90 G01 C90 ;Spindle speed remains constant
N100 G01 C120 ;Spindle speed remains constant
N110 G01 X50 Y50 Z50
N120 S[FEED_LINK OFF]
N130 M03 S200
N140 G01 X10
N150 M05 S0
N160 M30
```

15.3.10 Programmable spindle override

This function influences spindle speed in NC program. The spindle-specific programmed override is active when the assigned spindle moves.

If an external override is also defined, the effective override results from the multiplication of the two override values.

> **i** The G167 [▶ 640] function only suppresses the effect of the external override value.

Syntax:

\<spindle_name> **[OVERRIDE SPEED_FACT=..]**

\<spindle_name>	Name of the spindle
OVERRIDE	Identifier for spindle-specific override programming. Must always be programmed as the **first** keyword.
SPEED_FACT=..	Override factor for spindle speed blocks [0.1%-200%]

Programmable spindle override (OVERRIDE)

```
%spdl_override
N10 G01 X100 Y100 Z100 F1000
N20 S[M3 REV1000]
N40 S[OVERRIDE SPEED_FACT=20]        ;Spindle override 20%
N50 X0                               ;G01 motion with S200
N60 Y0                               ;G01 motion with S200
N70 Z0                               ;G01 motion with S200
N60 S[OVERRIDE SPEED_FACT=100]       ;Spindle override 100%
N90 Y100                             ;G01 motion with S1000
N100 Z100                            ;G01 motion with S1000
N110 X200 Y200                       ;G01 motion with S1000
N120 X300 Y300 Z200                  ;G01 motion with S1000
N130 M5
M30
```

15.3.11 Acceleration weighting (G130)

The G130 function can change the acceleration of the spindle axis.

Acceleration can be influenced by a percentage change in the associated acceleration characteristic values. With a jerk-limited profile, these values are the axis parameters P-AXIS-00001 and P-AXIS-00002.

If programming takes place with G130, all axes which are not programmed or not yet programmed are set to 100%. Every time these functions are selected, the 100% weighting is taken irrespective of previous programming.

Therefore, 50% programmed twice in succession means the setting is made to 50% and not to 25%. A weighting of over 100% is possible up to maximum axis acceleration P-AXIS-00008.

NOTICE
At program end, the G130 weighing factor is restored to 100%.

Acceleration weighting (G130)

```
N10 S[G130=70]  ;Spindle acceleration is limited to 70%
N20 M03 S1000 ;Endless rotation CW
N30 S[G130=60]  ;Spindle acceleration is limited to 60%
N40 M04 S1000 ;Endless rotation CCW
```

15.4 Changing the main spindle (#MAIN SPINDLE)

Syntax:

#MAIN SPINDLE [**[** *<spindle_name>* | *<spindle_number>* **]**]

<spindle_name>	Default spindle name according to [1] [▶ 786]-3.
<spindle_number>	log. axis number of the spindle according to [1] [▶ 786]-3.

The #MAIN SPINDLE command can be used to change the definition of the main spindle in the NC program. The new main spindle is selected by specifying the default name (P-CHAN-00053) or the related logical axis number.

The initial state (as after start-up) can be restored without programming a spindle name, i.e. the spindle preset in the channel parameter list P-CHAN-00051 becomes the main spindle again.

Channel parameter list [1] [▶ 786]:

Configuration example of a 1-channel system with 3 spindles. Spindle with axis number 6 is the main spindle:

```
# Spindle data
# ============
spdl_anzahl             3

main_spindle_ax_nr      6
main_spindle_name       S

spindel[0].bezeichnung  S1
spindel[0].log_achs_nr  6

spindel[1].bezeichnung  S2
spindel[1].log_achs_nr  11

spindel[2].bezeichnung  S3
spindel[2].log_achs_nr  30
```

Configuration after start-up:

S1 is the main spindle with the name "S".

"S2" and "S3" are other spindles.

Changing the main spindle

```
%
N10 S100 M3 S2[REV200 M3] S3[REV300 M4]
N20 #MAIN SPINDLE [S2]    (S2 is new main spindle "S")
N30 S110 M3 S1[REV210 M3] S3[REV310 M4]
N40 #MAIN SPINDLE [S3]    (S3 is new main spindle "S")
N50 S120 M3 S1[REV220 M3] S2[REV320 M4]
N60 #MAIN SPINDLE         (Back to initial state S1 -> "S")
N70 S150 M3 S2[REV250 M3] S3[REV350 M4]
N80 M5 S2[M5] S3[M5]      (All spindles STOP)
N99 M30
```

> ℹ️ As long as a spindle is a main spindle, it can either be programmed with the defined main spindle name P-CHAN-00053 or with its default name [1] [▶ 786]-3. It can only be addressed exclusively by its default name again after another main spindle is selected with #MAIN spindle[ ].

The following applies in the example above:

Permissible names	Spindle 1	Spindle 2	Spindle 3
...after start-up	**S** or **S1**	S2	S3
...after #MAIN SPINDLE [S2]	S1	**S** or **S2**	S3
...after #MAIN SPINDLE [S3]	S1	S2	**S** or **S3**
...after #MAIN SPINDLE	**S** or **S1**	S2	S3

As already mentioned, the main spindle can be programmed in the conventional DIN syntax. In this case, all commands in the table in Section Spindle programming [▶ 583] can be used. The main spindle may also be programmed in spindle-specific syntax. However, in this case only the restricted command set then is available (see also the table in Section Spindle programming. [▶ 583]).

The following NC lines are also permissible for the main spindle:

```
:
N10 S1=1000 M3   or
N20 S1000 M3    or
N30 S1[REV1000 M3]    or
N40 S[REV1000 M3]
:
```

15.5 Synchronous spindle operation

Besides the synchronous mode of path axes (definition, activation, deactivation), the LINK command can also be used to define master/slave relationships for spindle axes.

```
Syntax:
```
#SET AX LINK [*<coupling_group>*, *<slave>* = *<master>* {, *slave>* = *<master>* }]

or alternatively

#AX LINK [NBR] [*<coupling_group>*, *<slave>* = *<master>* {, *<slave>* = *<master>* }]

<coupling_group>	Number of the coupling group [1]
<slave>	Designation or logical axis number of the slave spindle of coupling pair i [2]
<master>	Designation or logical axis number of the master spindle of coupling pair i [2]
NBR	The logic switch NBR can change the evaluation from axis names to logical axes numbers. The axis couplings must then be defined with logical axis numbers.

In this case, the following rules apply in addition to section <u>Synchronous mode [▶ 339]</u>:

- Coupling pairs are defined by the <u>names</u> or <u>logical axis numbers</u> of the spindles which are known in the channel. I.e. only spindles known in the channel can be linked.

- In the channel parameter list <u>[1] [▶ 786]</u>-2 coupling group 0 can be preset as the default group. This can be addressed directly with #ENABLE AX LINK or #AX LINK ON after start-up. It can <u>not</u> be redefined in the NC program.

Synchronous spindle mode, programming and selection/deselection of a spindle coupling

Parameterisation in the channel parameter list <u>[1] [▶ 786]</u>: S (main spindle name S1), S1, S2, S3 The coupling pairs may be formed with spindle names S, S2 and S3.

```
N10 #SET AX LINK[1, S2=S, S3=S] ;Main spindle is master for S2+S3
N20 #ENABLE AX LINK[1]          ;Select spindle couplings
N30 S1000 M3          ;Rotate main spindles S+S2+S3 cw 1000 rpm
N40 #DISABLE AX LINK    ;Deselect spindle couplings

or alternatively

N10 #AX LINK[1, S2=S, S3=S]
N20 #AX LINK ON[1]
N30 S1000 M3
N40 #AX LINK OFF

or alternatively

N10 #AX LINK NBR[1, 11=6, 17=6] ;Coupling via log. axis numbers
N20 #AX LINK ON[1]
N30 S1000 M3
N40 #AX LINK OFF
N50 M30                    ;Program end
```

(1) 1 ... [Max. number of coupling groups–1], see [6] [▶ 786]-2.11

(2) Max. number of coupling pairs, see [6] [▶ 786]-2.12

- Different axis types may not be defined in a #SET AX LINK or #AX LINK command within a coupling pair. However, coupling pairs may form a coupling group with coupling pairs of another axis type.

Examples:

#SET AX LINK [1, B=S, S2=X] **WRONG**

#SET AX LINK [1, B=X, S2=S] **PERMISSIBLE**

- During an active coupling group, a spindle present in this group may not be declared the main spindle with #MAIN SPINDLE [..], otherwise, this would result in inconsistencies between decoder and interpolator.
- Moreover, the programmer must be aware that using #MAIN SPINDLE [..] may possibly mean that already defined coupling groups can no longer be activated since the spindle names are no longer consistent.
- The technology information for M03, M04, M05 is synchronised by the NC kernel both for the master spindle and for the slave spindle.
- External movement influencing of the master spindle with FEEDHOLD and OVERRIDE does not act on slave spindles. During active coupling the slave spindles also continue evaluating their own OVERRIDE and FEEDHOLD interfaces.
- Master and slave spindles move to the same absolute position with M19. The position may possibly not be reached simultaneously if the start position or the dynamic data of the spindles are different.
- Coupling is cancelled when the program is aborted.
- If a spindle is active as master or slave, it may only be commanded by the channel which activated the link.

Channel parameter list [1] [▶ 786]:

```
:
# Pre-assignment of possible axis links for synchronous mode
# ======================================================
#synchro_data.koppel_gruppe[0].paar[0].log_achs_nr_slave 4
#synchro_data.koppel_gruppe[0].paar[0].log_achs_nr_master 1
#synchro_data.koppel_gruppe[0].paar[0].mode 0 ->AX_LINK
#synchro_data.koppel_gruppe[0].paar[1].log_achs_nr_slave 11
#synchro_data.koppel_gruppe[0].paar[1].log_achs_nr_master 6
#synchro_data.koppel_gruppe[0].paar[1].mode 1 ->SPDL_LINK
:
```

15.6   Cross-block synchronisation (Late Sync)

15.6.1   Implicit synchronisation

Spindle acceleration and deceleration operations may lead to substantial dead times in program execution since when the machine is at standstill, the spindle frequently needs to be set to the required speed first (e.g. M03 of type MVS_SVS) or during a positioning block with G00 (M3 of type MVS_SNS).

With M functions, implicit synchronisation provides the option of only checking acknowledgement if there is a switch over to a machining operation with G01/G02/G03/G151 etc. This reaction is achieved with synchronisation mode MVS_SLM. The identifier can only be used exclusively with other synchronisation (P-CHAN-00027).

Implicit synchronisation

```
:
N10 G00 M03 S1000 Z600               (M03: Synchronisation mode MVS_SLM)
N20 X100 Y100
N30 Z400
N40 G01 Z200                         (Check whether M03 is acknowledged)
:
```

In N40, the interpolator checks for acknowledgement of the M function at the start of the braking instant. If the acknowledgement is output, there is no stop at block end. If no acknowledgement is output, deceleration occurs and if no acknowledgement is output by block end, the system stops at the target point.

It is possible to program further channel-specific M functions up to synchronisation by a motion block. Synchronisation of channel-specific M functions is handled entirely in parallel with axis-specific synchronisation.

15.6.2 Explicit synchronisation (#EXPL SYN)

If G01 is used for positioning, no implicit synchronisation can be executed according to Sec. Implicit synchronisation [▶ 653].

The #EXPL SYN command is provided here for cross-block synchronisation and this permits explicit synchronisation of the M function.

```
Syntax:
```
#EXPL SYN   non-modal

An M function which is to be synchronised with this additional command is defined with synchronisation mode MVS_SLP in the channel parameter list P-CHAN-00027. The identifier can only be used exclusively with other synchronisation (P-CHAN-00027).

Explicit synchronisation

```
:
N10 G01 M03 S1000 Z200 F5000   ;M03: synchronisation mode MVS_SLP
N20 X100 Y100
N30 Z400
N40 #EXPL SYN                  ;Check whether M03 is acknowledged
:
```

At the braking instant, the path checks whether the acknowledgement has arrived based on the statement "#EXPL SYN" . A ramp-down occurs if this is not the case.

Further channel-specific and axis-specific M functions can be processed before the synchronisation command.

15.7 Synchronisation of spindle M functions

Synchronisation between the interpolator and the relevant spindle is executed directly, i.e. the acknowledgement of M03, M04 (speed reached) and M05 (speed zero) is executed by the spindle itself. Bit PLC_INFO which can be set in addition to the existing synchronisation P-CHAN-00027 determines whether the PLC is also to be acknowledged. In this case, note the following:

In general the PLC automatically acknowledges each spindle M function for speed-controlled spindles. It is therefore not necessary to additionally set the PLC_INFO bit.

It is practical to use the PLC_INFO bit for position-controlled spindles. In this case, the PLC_INFO bit can also be set for each spindle M function in addition to the synchronisation mode, thus causing the PLC to send an acknowledgement.

Channel parameter list [1] [▶ 786]:

Spindle S1 is to be a position-controlled spindle.

```
:
spindel[0].bezeichnung              S1
spindel[0].log_achs_nr              6
spindel[0].s_synch                  0x00020001    PLC_INFO, MOS
spindel[0].m3_synch                 0x00020002    PLC_INFO, MVS_SVS
spindel[0].m4_synch                 0x00020004    PLC_INFO, MVS_SNS
spindel[0].m5_synch                 0x00020002    PLC_INFO, MVS_SVS
spindel[0].m19_synch                0x00000008              MNS_SNS
spindel[0].s_prozess_zeit           0
spindel[0].m3_prozess_zeit          0
spindel[0].m4_prozess_zeit          0
spindel[0].m5_prozess_zeit          0
spindel[0].m19_prozess_zeit         0
:
```


Fig. 169: Diagram of synchronisation of the spindle M function

15.8   PLCopen programming

A complete list of PLCopen functions is contained in the overview of commands in the Appendix under PLCopen programming [▶ 656].

The scope of the Motion Control Platform (MCP) provides a number of function blocks (FB) for motion tasks. These FBs act on a single axis and are operated via the SPS. Each axis in the system is configured in the system as a so-called Single Axis Interpolator (SAI).

Alternatively these axes also can be addressed by the NC program because an SAI is always configured in the system as a conventional spindle. Special NC commands are therefore provided for the following FBs. These commands permit PLCopen-conforming programming in NC syntax.

MC_Home	Homing
MC_MoveAbsolute	Axis motion to an absolute position
MC_MoveAdditive	Relative axis motion to the commanded position
MC_MoveRelative	Relative axis motion to the current position
MC_MoveSuperImposed	Relative axis motion to a motion already active
MC_MoveVelocity	Endless axis motion at the specified velocity
MC_Stop	Stop an axis motion
MC_GearIn	Gear coupling with a gear ratio
MC_GearOut	Release a gear coupling
MC_Phasing	Phase offset of couplings
MC_TouchProbe	Measurement of axis positions

The topology below displays the basic arrangement of SAI (spindle) axes within the overall system:

Fig. 170: SAI axes in CNC topology

An SAI axis is addressed in the NC program in spindle-specific programming syntax. It must therefore be configured in the NC channel by its address letters and other data analogous to the configuration of a spindle in the channel parameter list. The most important settings in the channel parameter list are:

- spdl_anzahl (P-CHAN-00082) – Total number of (SAI) spindles
- bezeichnung (P-CHAN-00007) – Name of the (SAI) spindle
- log_achs_nr (P-CHAN-00036) – Logical axis number of the (SAI) spindle

For more information, see the documentation [1] [▶ 786] Section: Configuring spindles- and the Section Parameterising spindles [▶ 585].

The PLCopen functions

- MC_MoveSuperImposed
- MC_GearIn
- MC_GearOut
- MC_Phasing
- MC_TouchProbe

require additional specific SAI characteristics of the (spindle) axis which are configured in the axis parameters. The required settings are contained in the documentation [2] [▶ 786]-Section: SAI settings.

Each NC command of the corresponding FB is presented below. The syntax of these NC commands and the units of the programmed values are based on the corresponding input pins (VAR_INPUT) of the assigned FBs.

```
General syntax of an (SAI) NC command:
```
*<spindle_name>***[** *<FB name> <Input_pin1> < Input_pin2> < Input_pin n...>* **{ \ }]**

The axis name at the start of the NC command addresses the (SAI) spindle axis which is addressed by the NC channel.

The description of the input pins and the units and value ranges are also contained in the documentation [9] [▶ 786].

ℹ️ All input pin values are programmed in metric units. In initial state the values must be specified in the specified internal units (e.g. 0.1 µm). The parameter P-CHAN-00182 can be changed to specify values in default units (e.g. mm).

The input pin "Execute" is always assigned implicitly by programming the NC command. This is why no specific keyword is provided for this pin.

The line separator '\' can be used within the [...] brackets to obtain a clear programming of the command over multiple lines.

The keywords "Id" and "WaitSyn" for job synchronisation in the NC program have no corresponding PINs in the PLC. The two keywords are available as of CNC Build V3.01.3100.01

By default PLCopen functions are executed irrespective of other NC program processing. There is no synchronisation between PLCopen single-axis jobs and path motion.

However, wait conditions can be defined to synchronise PLCopen functions with the program run. There are two options here:

1) Synchronisation at block end:

Defining the "WaitSyn" keyword causes the CNC to wait for the completion of the PLCopen job before continuing to the next NC block. If several PLCopen jobs are programmed in the NC line, continuation only takes place when all the jobs specified for the "WaitSyn" keyword are completed.

```
Syntax:
```
<spindle_name>[*<FB_name>* [**WaitSyn**] *<input_pin1> <input_pin n...>* { \ }]

```
N10 G01 X100 F10000
N20 S[MC_MoveAbsolute WAIT_SYN POSITION=900000 …]
;Continue to block N30 when spindle S reaches ;position 90°
N30 G01 X200 F100
```

2) Late synchronisation:

The "Id" keyword can be used to assign a job ID to a PLCopen job. The #WAIT MC_STATUS SYN [ID<JobNo>] command can wait for the end of the PLCopen job at a later time.

```
Syntax:
```
<spindle_name>[*<FB_name>* [**Id**=*..*>] *<input_pin2> <input_pin n...>* { \ } **]**

#WAIT MC_STATUS SYN [ID=.. { **ID**=.. } **]**

If several PLCopen jobs with identical job numbers are started, the job number is assigned to the last PLCopen function commanded. It is then possible that the job last started may be subject to later synchronisation at job end.

```
N10 G01 X100 F10000
N20 S[MC_MoveAbsolute Id100 POSITION=900000 …]
N30 G01 X200
N40 S2[MC_MoveVelocity Id200 Velocity=10000 …}
N50 G01 X300
N60 #WAIT MC_STATUS SYN [ID100 ID200]
; Continue to block N70 takes place when spindle S
; reaches position 90° and spindle S2 reaches the speed 10°/s
N70 G0 X0
```

15.8.1   MC_Home command

MC_Home commands a homing run (homing) for the axis. How an axis reacts to this command basically depends on the type of referencing operation.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```

*<axis_name>*__[MC_Home__ [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

MC_Home command

```
S[MC_Home]
```

15.8.2   MC_MoveAbsolute command

MC_MoveAbsolute commands an axis motion to an absolute position. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```

*<axis_name>*__[MC_MoveAbsolute Position__=.. **Velocity**=.. [**Acceleration**=..]
    [**Deceleration**=..] [**Jerk**=..] **Direction**=.. [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

Input pin	Unit
Position	[0.1 µm or 10^{-4} °]
Velocity	[1 µm/s or 10^{-3} °/s]
Acceleration	[1 mm/s^2 or 1°/s^2]
Deceleration	[1 mm/s^2 or 1°/s^2]
Jerk	[1 mm/s^3 or 1°/s^3]
Direction	1 positive direction
	2 shortest path
	3 negative direction
	4 current direction

```
MC_MoveAbsolute
Axis                          Axis
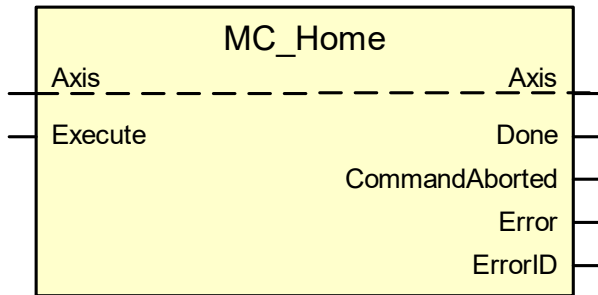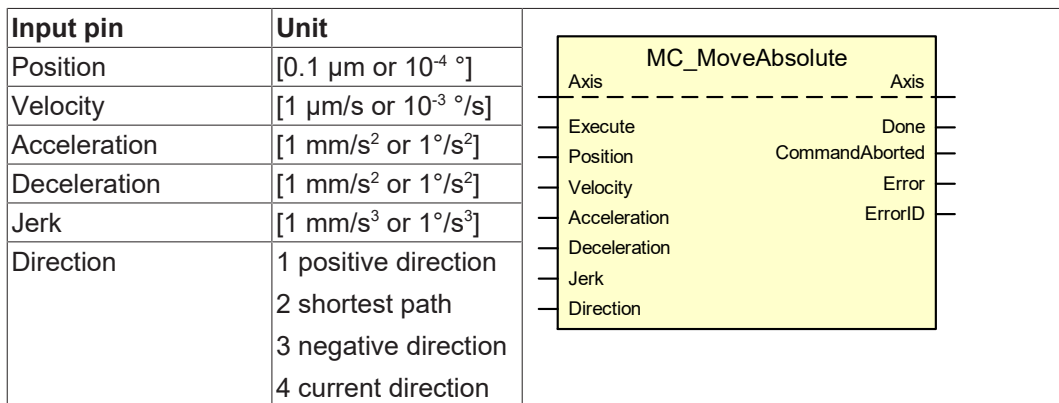
Execute                       Done
Position              CommandAborted
Velocity                     Error
Acceleration                ErrorID
Deceleration
Jerk
Direction
```

MC_MoveAbsolute

```
S[MC_MoveAbsolute Position=133 Velocity=1000 Acceleration=500 \
Deceleration=600 Jerk=20000 Direction=2]
```

15.8.3 MC_MoveAdditive command

MC_MoveAdditive commands a relative motion in addition to the commanded position if the axis is in "Discrete Motion" state. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

If the axis is in "Continuous Motion" state and receives a command from this command, the relative distance from the current position at the time of command is added.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
<axis_name>[**MC_MoveAdditive Distance**=.. **Velocity**=.. [**Acceleration**=..]
        [**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ }]

Block diagram of the function block in PLCopen:

Input pin	Unit
Distance	[0.1 µm or 10^{-4} °]
Velocity	[1 µm/s or 10^{-3} °/s]
Acceleration	[1 mm/s^2 or 1°/s^2]
Deceleration	[1 mm/s^2 or 1°/s^2]
Jerk	[1 mm/s^3 or 1°/s^3]

```
                    MC_MoveAdditive
       Axis                               Axis
       — — — — — — — — — — — — — — —
    —  Execute                    Done  —
    —  Distance          CommandAborted  —
    —  Velocity                  Error  —
    —  Acceleration            ErrorID  —
    —  Deceleration
    —  Jerk
```

MC_MoveAdditive command

```
S[MC_MoveAdditive Distance=277 Velocity=1100 Acceleration=550 \
Deceleration=660 Jerk=22000]
```

15.8.4 MC_MoveRelative command

MC_MoveRelative commands a relative motion in addition to the current position. This is regardless of whether the axis is in "Discrete Motion" or "Continuous Motion" state. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
<*axis_name*>**[MC_MoveRelative Distance**=.. **Velocity**=.. [**Acceleration**=..]
        [**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

Input pin	Unit
Distance	[0.1 µm or 10^{-4} °]
Velocity	[1 µm/s or 10^{-3} °/s]
Acceleration	[1 mm/s^2 or 1°/s^2]
Deceleration	[1 mm/s^2 or 1°/s^2]
Jerk	[1 mm/s^3 or 1°/s^3]

```
                    MC_MoveRelative
Axis                                          Axis
- - - - - - - - - - - - - - - - - - - - - - - - - -
Execute                                       Done
Distance                              CommandAborted
Velocity                                      Error
Acceleration                                ErrorID
Deceleration
Jerk
```

MC_MoveRelative command

```
S[MC_MoveRelative Distance=321 Velocity=1200 Acceleration=555 \
Deceleration=666 Jerk=22000]
```

15.8.5 MC_MoveSuperImposed command

MC_MoveSuperImposed commands a relative motion in addition to a motion already active. The active motion is not interrupted but is superimposed over the commanded one. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

Since "Acceleration" values are also superimposed in the case of superimposed interpolation, corresponding axis parameters must be defined to ensure that the axis is not dynamically overloaded.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
*<axis_name>***[MC_MoveSuperImposed Distance**=.. **VelocityDiff**=..
        [**Acceleration**=..] [**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

Input pin	Unit
Distance	[0.1 µm or 10^{-4} °]
VelocityDiff	[1 µm/s or 10^{-3} °/s]
Acceleration	[1 mm/s^2 or 1°/s^2]
Deceleration	[1 mm/s^2 or 1°/s^2]
Jerk	[1 mm/s^3 or 1°/s^3]

```
          MC_MoveSuperimposed
Axis                         Axis
─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─
─ Execute                  Done ─
─ Distance        CommandAborted ─
─ VelocityDiff            Error ─
─ Acceleration         ErrorID ─
─ Deceleration
─ Jerk
```

MC_MoveSuperImposed command

```
S[MC_MoveSuperImposed Distance=321 VelocityDiff=783 Acceleration=811 \
Deceleration=922 Jerk=45000]
```

15.8.6   MC_MoveVelocity command

MC_MoveVelocity commands an endless motion at the specified velocity. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

To stop the motion, the command must be interrupted by another command that sends a new command to the axis.

In conjunction with an MC_MoveSuperImposed command, the "InVelocity" output remains TRUE.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
<axis_name>[**MC_MoveVelocity Velocity**=.. [**Acceleration**=..] [**Deceleration**=..]
          [**Jerk**=..] **Direction**=.. [**Id**=..] [**WaitSyn**] { \ }]

Block diagram of the function block in PLCopen:

Input pin	Unit
Velocity	[1 µm/s or 10^{-3} °/s]
Acceleration	[1 mm/s^2 or 1°/s^2]
Deceleration	[1 mm/s^2 or 1°/s^2]
Jerk	[1 mm/s^3 or 1°/s^3]
Direction	1 positive direction
	3 negative direction
	4 current direction

MC_MoveVelocity command

```
S[MC_MoveVelocity Velocity=1333 Acceleration=770 Deceleration=880 \
Jerk=10000 Direction=1]
```

15.8.7   MC_Stop command

MC_Stop leads to a controlled motion stop and places the axis in "Stopping" state. The motion stop is always jerk-limited at the constant set in "Jerk" to build up the deceleration rate.

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

It aborts every ongoing command by other (SAI) motion commands.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
*<axis_name>***[MC_Stop** [**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

Input pin	Unit
Deceleration	[1 mm/s^2 or 1°/s^2]
Jerk	[1 mm/s^3 or 1°/s^3]

```
            MC_Stop
  Axis                      Axis
  ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─
  Execute                   Done
  Deceleration              Error
  Jerk                      ErrorID
```

MC_Stop command

```
S[MC_Stop Deceleration=999 Jerk=25000]
```

15.8.8   MC_GearIn command

MC_GearIn commands a gear coupling at a gear ratio. The gear ratio defines the velocity ratio between master and slave axes. Synchronisation to the velocity is jerk-limited. "Jerk" is set in the command.

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

The slave axis can be linked either to master setpoint values or to actual master values. The selection is made in the "Mode" parameter.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

The "PhaseShift" parameter specifies the required phase of master and slave axes which are set during automatic phase compensation (Mode = 256). The value is programmed in metric units [0.1 µm or 10^-4°]. This parameter has the default value 0.

```
Syntax of the NC command:
```
<axis_name>[**MC_GearIn Master**=.. **RatioNumerator**=.. **RatioDenominator**=..
　　　　[**Acceleration**=..] [**Deceleration**=..] [**Jerk**=..] **Mode**=.. [**Id**=..]
　　　　[**WaitSyn**] [**PhaseShift**=..] { \ } **]**

Block diagram of the function block in PLCopen:

Input pin	Unit		
Master *	Logical axis number of the master axis		
RatioNumerator *	Gear ratio numerator		
RatioDenominator *	Gear ratio denominator		
Acceleration	[1 mm/s^2 or 1°/s^2]		
Deceleration	[1 mm/s^2 or 1°/s^2]		
Jerk	[1 mm/s^3 or 1°/s^3]		
Mode	0	Type of coupling between master axis and slave: Coupling based on command values.	
	128	Type of coupling between master axis and slave: Coupling based on actual values.	
	256	Automatic phase compensation: ON.	

```
                MC_GearIn
Master  _ _ _ _ _ _ _ _ _ _  Master
Slave   _ _ _ _ _ _ _ _ _ _  Slave

── Execute                    InGear ──
── RatioNumerator      CommandAborted ──
── RatioDenominator            Error ──
── Acceleration              ErrorID ──
── Deceleration
── Jerk
── Mode
```

*As a supplement to PLCopen the following options are available for these input pins:

Master	As an alternative to the logical axis number the axis name of the master spindle can also be programmed.
RatioNumerator	Alternative abbreviation RN
RatioDenominator	Alternative abbreviation RD

MC_GearIn command

```
S[MC_GearIn Master=11 RatioNumerator=2 RatioDenominator=3 \
Acceleration=500 Deceleration=600 Jerk=20000 Mode=0]

Commanding with master axis names, default dynamic values and abbreviation of gear ratio:
S[MC_GearIn Master=S2 RN=1 RD=3 PhaseShift=25 Mode=256 WaitSyn]
```

15.8.9   MC_GearOut command

MC_GearOut releases the coupling of the slave axis to the master axis that is specified by a velocity ratio. The current velocity of the slave is retained (endless motion).

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
*<axis_name>***[MC_GearOut** [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

MC_GearOut command

```
S[MC_GearOut]
```

15.8.10 MC_Phasing command

MC_Phasing is used to achieve an offset of the slave axis relative to the master axis. Accordingly, a phase offset of the master axis is specified from the point of view of the slave axis and the slave axis tries to eliminate this offset by accelerating or decelerating. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to ≤ 0, the dynamic values are taken from the corresponding axis list.

The mechanical analogy is to release the coupling of the master and slave axes for a limited period.

With **camming,** this command results in a change in the "apparent" master position from the slave's point of view. With **gearing**, a phase offset is produced between the master and slave by commanding a superimposed motion in the slave. Therefore with gearing, MC_Phasing has the same effect as MC_MoveSuperImposed (into which it is actually converted in the control system).

The dynamic values: "Velocity", "Acceleration" and "Deceleration" refer to the change in the "apparent" master position from the point of view of the slave with camming; but with gearing, they refer to the superimposed motion of the slave axis itself.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
<axis_name>[**MC_Phasing Master**=.. **PhaseShift**=.. **Velocity**=..
         [**Acceleration**=..] [**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

Input pin	Unit
Master	Logical axis number of the master axis
PhaseShift	[0.1 µm or 10^{-4} °]
Velocity	[1 µm/s or 10^{-3} °/s]
Acceleration	[1 mm/s^2 or 1°/s^2]
Deceleration	[1 mm/s^2 or 1°/s^2]
Jerk	[1 mm/s^3 or 1°/s^3]

```
            MC_Phasing
Master ─ ─ ─ ─ ─ ─ ─ ─ ─ ─  Master
Slave  ─ ─ ─ ─ ─ ─ ─ ─ ─ ─  Slave
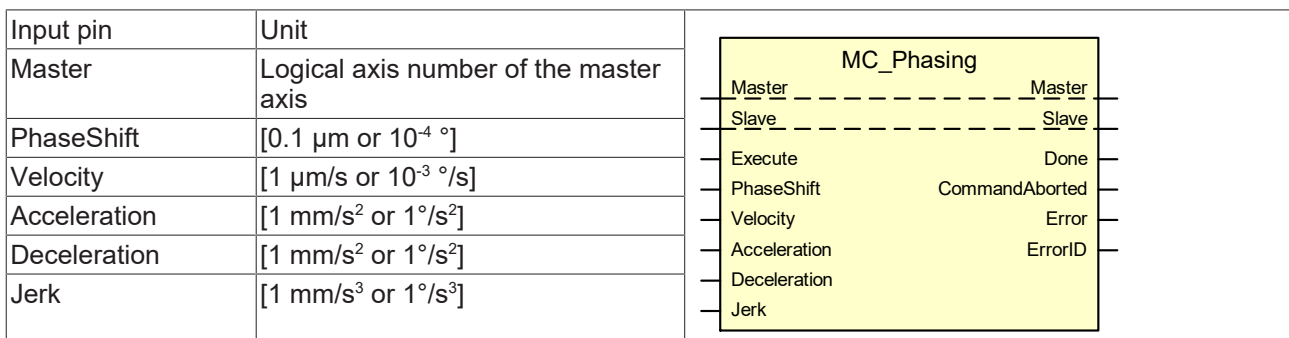
Execute                     Done
PhaseShift          CommandAborted
Velocity                    Error
Acceleration               ErrorID
Deceleration
Jerk
```

MC_Phasing command

```
S[MC_Phasing Master=11 PhaseShift=25 Velocity=1000 Acceleration=500 \
Deceleration=600 Jerk=20000]
```

15.8.11 MC_TouchProbe command

The MC_TouchProbe command records an axis position when a trigger event occurs. The measurement channel and method (rising or falling edge of the trigger signal) are defined via the reference for the trigger signal source.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

```
Syntax of the NC command:
```
<axis_name>[**MC_TouchProbe Channel**=.. [**Mode**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Block diagram of the function block in PLCopen:

Input pin		Unit	
TriggerInput	Channel	----	
	Mode	---	If no mode is specified, probe signal evaluated acc. to P-AXIS-00518
		1	Probe signal on falling edge
		2	Probe signal on rising edge

```
          MC_TouchProbe
Axis                           Axis
TriggerInput           TriggerInput
Execute                        Done
                               Busy
                              Error
                            ErrorID
                     CommandAborted
                    RecordedPosition
```

MC_TouchProbe

```
S[MC_TouchProbe Channel=1 Mode=2]
```

16 Macroprogramming (# INIT MACRO TAB)

Macros permit the assignment of an alias name (macro name) to an executable NC code (macro content). The macro content may consist of arithmetical expressions and NC commands. The related NC code is executed whenever the corresponding macro name (macro call) is specified.

Macros support the generation of low-maintenance and clear NC programs. Changes need only be made once in the macro content.

Macros may only be configured in the channel parameter list [1] [▶ 786]-1 and also defined in the NC program.

```
Syntax of a macro definition
```

"*<macro_name>*" = "*<macro_content>*"

<macro_name>   Name des Makros (alias)

<macro_content>   Executable NC code

- Macro name and macro content must be enclosed in quotation marks "...".
- The macro name makes a distinction between uppercase and lowercase letters.
- The redefinition of predefined macros is configurable [6] [▶ 786]-6.39.
- Macro definition remain valid after program end (M30) or CNC reset. They are usable until the next controller start-up or #INIT MACRO TAB.
- Macro definitions that were configured in the channel parameter list cannot be deleted by #INIT MACRO TAB.
- The maximum number of macro definitions [6] [▶ 786]-6.25 and their length [6] [▶ 786]-6.37/-6.38 as fixed. The values can be set as of Builds V3.1.3079.17 and V3.1.3107.10
  - Maximum number of expression labels P-CHAN-00509
  - Maximum number of local subroutine definitions P-CHAN-00510
  - Maximum number of characters in macro name P-CHAN-00511
  - Maximum number of characters in macro content P-CHAN-00512
- Macro definitions and other NC commands can be combined in the same NC block.

A macro call is initiated by the macro name and can be combined in the same NC block with other NC commands.

```
Syntax of a macro call:
```

"*<macro_name>*"

Macro definition and use

```
N10  "POSITION_1" = "X200 Y200 Z300"     ;Macro definition
N20  "POSITION_2" = "X300 Y100 Z50"      ;Macro definition
:
N200"POSITION_1"     (Macro call, X200 Y200 Z300 is executed)
:
N500"POSITION_2"     (Macro call, X300 Y100 Z50 is executed)
```

> ⓘ As of Build **V2.11.2010.02** the command **#INIT MACRO TAB** replaces the command **#INIT MAKRO TAB.** For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

```
Syntax of Initiating the macro table:
```

#INIT MACRO TAB

This command clears all macros previously defined in the macro table. The macros pre-assigned by the channel parameters [1] [▶ 786]-1 are retained.

16.1 Nesting macros

Macros may be used on the right side of the assignment (called nesting) in combination with the NC code. The maximum nesting depth is fixed [6] [▶ 786]-6.40.

Nesting is displayed by a '\' character that precedes the delimiting quotation marks. A macro should always represent complete expressions of an NC block (NC command, mathematical expression, term). This excludes the possibility of a macro only represented by the address letter of an NC command without the related mathematical expression. This connection will be discussed in the next sections.

```
Syntax of Nesting macros
```

"<*macro_name*>**" = "**<*NC_code*> **\"**< *macro_name_i*> **\"** <*NC_code*>**"**

NOTICE
A macro may not be called by its own nested macro name. Only nesting of other macro names is permitted.

Nesting macros

```
Example 1:
N10 "POS_1" = "X500 Y200"          (Macro definition)
N20 "MOVE1" = "G01 \"POS_1 \"F1000"  (Macro definition with nesting)
N30 "MOVE1"                        (Macro call)
M30


Example 2:
N10 " STRING_1 " = " 5*12 "          (Macro definitions)
N20 " STRING_2 " = " G \"STRING_1\" + 5 "
N30 " STRING_3 " = " M \" STRING_1\ " \" STRING_2 \" "
:
N200 " STRING_3 "                    (Call the nested macro)
:                                    (Corresponds to: N200 M60 G65         )
```

16.2 Use in mathematical expressions

Macro names may be assigned to arithmetic expressions and parts of them. Recursive treatment (nesting) may also be used within mathematical expressions.

It must be ensured that a string in a macro content always combines complete levels (i.e. terms whose results are not influenced by inserting '[' at the start and ']' at the end).

Use in mathematical expressions

```
Right:
N10 "STRING1" = "0.5"
N20 "STRING2" = "5 * 12"
N30 "STRING3" = "SIN[89.5 + \"STRING1\"]"
N40 X[-2 * "STRING1" + "STRING2" + "STRING3"]          (move to X60)
M30

Wrong: Macros contain only incomplete mathematical expressions
N10 "STRING1" = "COS["
N20 "STRING2" = "90]"
N30 "STRING3" = " \"STRING1\" \"STRING2\" "            Error
```

16.3 Separating address letter and mathematical expression

It is also possible to only assign a macro name the address letter and program or define the mathematical expression in a second macro.

This permits macros that reference the main axes to be defined in a higher-level NC program. The macro may then be used in the subroutine or cycle, providing a certain independence from the selected processing level.

Separating address letter and mathematical expression

```
Only the address letter is contained in the macro content. The mathematical expression is programmed
or defined in a separate macro:
"1.PA" = "X" "2.PA" = "Y" primary axis)
"Ziel_1.HA" = "V.E.POS1 + P12"
"1.PA" "TARGETl_1.PA" "2.PA" 100
```

16.4   Restrictions

No end-of-line or end-of-string (´\0´) characters may be included in macro content. The limits the macro definition to one block.

```
"Macro_Move" = "X100 G01 \0"
"Macro_Move2" = "X100
                G01"
...
M30
```

Macro content may not contain any control block statements ($).

```
"IF"     = "$IF"
"END_IF" = "$ENDIF"

P1 = 0

"IF" P1 == 0
P2 = 2
"ENDIF"
...
M30
```

Macro content may not contain any strong constants. However, strong functions or V.E. variables of the string type are permitted.

Recursive macro calls cause an error when the macro is executed.

```
"Macro_Recursive" = "G01 X100 \"Macro_Recursive\""
...
M30
```

17 5-Axis functionality

17.1 Rotation Tool Centre Point (RTCP)(# TRAFO OFF)

> **i** The use of this feature requires a license for the "Transformations" extension package. It is not included in the scope of the standard license.

`Syntax:`

#TRAFO ON	Selection
#TRAFO OFF	Deselection

The RTCP function represents tool compensation in space.

After RTCP is selected, the contact point of the current tool remains stationary relative to the workpiece when tool orientation is changed (please note the assignments of the tool tip offset parameters P-CHAN-00094 and P-TOOL-00009 when assigning parameter values to kinematic transformation).

Fig. 171: Motion control with/without RTCP

The resulting motion is shown on the left of the figure when the rotary machine axis is moved.

RTCP shifts the centre point of rotation to the tool tip (centre point of tool rotation). The resulting offsets in the axes X, Y, Z based on tool motion are compensated in each cycle by corresponding opposing motions.

Only the axes X, Y, Z are outputs from kinematic transformation; the rotary machine axes are programmed as usual.

The RTCP function may not be selected when TLC is active (see section Tool Length Compensation (#TLC ON/OFF) [▶ 678]).

NOTICE

When kinematic transformation is active, axis-specific tool offsets in *ax_versatz[<ax_index>]* (P-TOOL-00006) are only taken into consideration if axes are not influenced by the transformation function. Depending on the transformation type, they typically refer to all axes with index > 2 when RTCP is used.

The axis-specific tool offsets of the first three axes (index 0, 1, 2) are **not** taken into account when transformation is active. If tool offsets should also be effective for these axes when transformation is active, enter the values in the kinematic offsets of the tool (P-TOOL-00009) mentioned above.

RTCP example 1

```
N10 T1 D1              (Select 2.5D tool compensation)
N20 #TRAFO ON          (Select RTCP)

N30 G01 F100 B45 C30   (Programming rotary axes modifies tool)
                       (Orientation. Tool contact point remains)
                       (stationary)
N40...
.
.
N100 #TRAFO OFF        (Deselect RTCP)
N200 M30
```

RTCP example 2

```
N10 #KIN ID [1]                  (Select machine kinematic)
N20 #TRAFO ON                    (Select RTCP)
N30 G01 G18 X20 Y0 Z25 B90 F500  (Move workpiece from right)
N40 G91 X-8                      (Move to start position)
N50 G90 G02 X-12 I-12 B-90 F2000 (Process)
N60...
```


Fig. 172: Motion control with RTCP

17.2 Tool Length Compensation (#TLC ON/OFF)

i Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

```
Syntax:
```

#TLC ON [*<delta_tool_length>* **]**   Select TLC

#TLC OFF   Deselect TLC

<delta_tool_length>   Tool length difference in [mm, inch].

TLC permits the reuse of NC programs which were created by a programming system and which consider a specific tool length, even if the tool length was changed on the machine. It must be noted that no new tool offsets or radii may be compensated.

> **The TLC function may not be selected when RTCP is active.**
>
> The error is output with ID 20669. The two functions mutually exclude each other.

Fig. 173: When the tool length is changed, TLC transforms ΔL in each cycle.

BECKHOFF

Tool Length Compensation

```
N10 #TLC ON [15]   (Select TLC with extended tool,ΔTL =15 mm)
N20 .
.
N100 #TLC OFF      (Deselect TLC)
N200 #TLC ON [-20]  (Select TLC with extended tool, ΔTL = -20 mm))
N210 .
.
N300 #TLC OFF      (Deselect TLC)
N200 M30
```

N10 #TLC ON [15] (Select TLC with extended tool,ΔTL =15 mm)
N20 .
.
N100 #TLC OFF (Deselect TLC)

17.3   Orienting tool (#TOOL ORI CS)

#TOOL ORI CS   Select tool orientation

The first motion block that follows #TOOL ORI CS aligns the tool in parallel to the 3rd main axis of the current PCS (W_0- or MCS also possible). Programming the rotary axes in this block overrides the positions that are valid for orientation.

Fig. 174: Tool aligned perpendicularly to the X-Y machining plane

Orienting tool

```
N10 B10 C20          (Slant tool)
N20 #TOOL ORI CS     (in next motion block, align tool in parallel to)
                     (the Z axis of current PCS, here MCS)
N30 X0 Y0            (Motion block in the MKS, tool is aligned B=0,C=0)
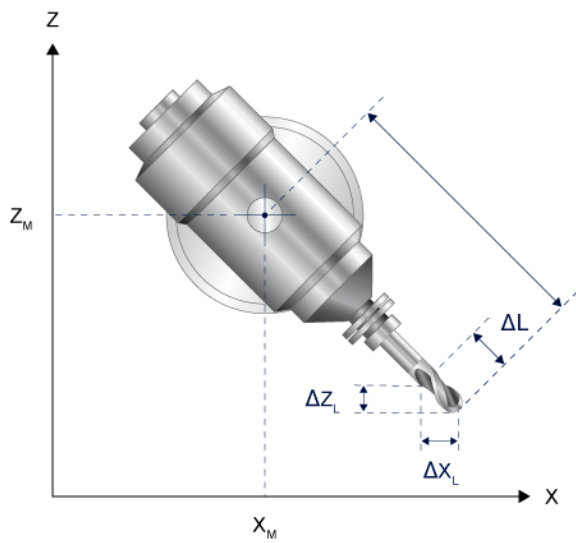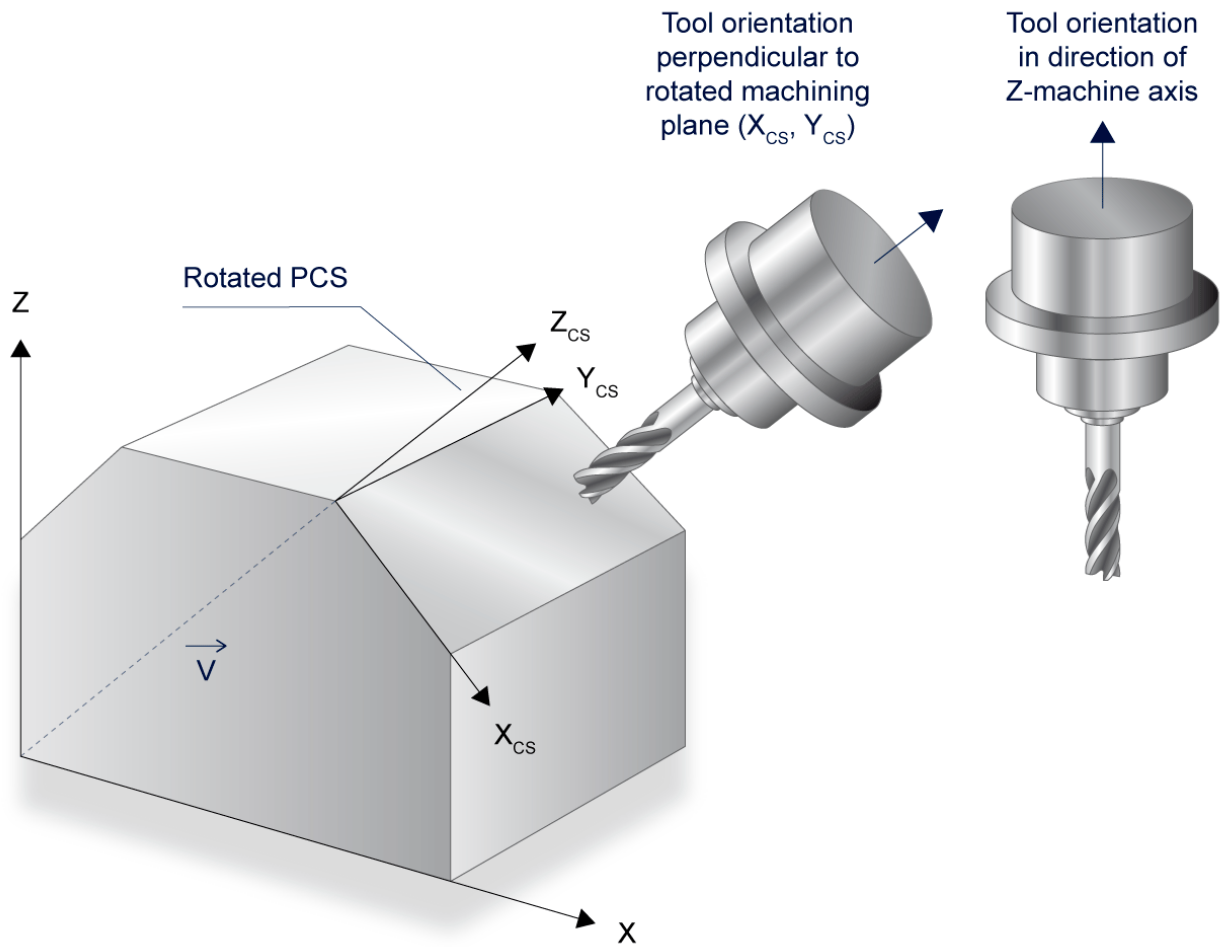
N40 B25 C-80          (Slant tool)
N50 #TOOL ORI CS     (Align tool in next motion block)
N60 #TRAFO ON        (Select RTCP)

N70 #CS ON[0,0,0,-80,-30,45]  (Transition to a rotated PCS)
N80 X100             (Motion block in PCS. align tool from N50, refers)
                     (to the MCS, i.e. B=0,C=0)
N90 #TOOL ORI CS     (Align tool in next motion block)
N100 Y150            (Too aligned in parallel to Z axis of PCS)

N110 #TOOL ORI CS
N120 Z100 B45 C10    (#TOOL ORI CS has no effect when programming)
                     (rotary axes)
N130 G18             (Change to the Z-X interpolation plane)
N140 #TOOL ORI CS    (Align tool in parallel to the Y axis)
N150 X0              (Tool aligned perpendicularly to X-Z machining plane)
M30
```

17.4 Machine kinematics (#KIN ID)

ℹ️ Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

`Syntax of Defining machine/tool head kinematics:`

#KIN ID [[*<kin_id>*]]   Defining machine/tool head kinematics

<kin_id>   Kinematic ID The purpose of the kinematic ID is to identify the machine or tool head-specific kinematic types implemented in the controller. Their default setting after controller start-up is parameterised in P-CHAN-00032.

The default kinematic ID is set by programming #KIN ID without parameters.

In addition, a kinematic change can be executed automatically when the tool is changed by assigning the "kin_id" element in the tool data list.

An unknown kinematic ID causes the output of an error message and a decoding stop when RTCP, TLC or TOOL ORI CS is selected.

When kinematic ID 0 is selected, no kinematic is enabled with no warning or error message.

NOTICE
It is not permitted to change kinematics with #KIN ID... when RTCP or TLC is active.

Machine kinematics

```
N10 #TOOL ORI CS     (Align tool, default kinematic)
                     (valid from P-CHAN-00032.)
N20 T1 D1            (Select tool)
N30 #TRAFO ON        (Select RTCP, default kinematic)
                     (valid from P-CHAN-00032)
.
.
N40 #TRAFO ON        (Deselect RTCP)
N50 #KIN ID[2]       (Select kinematic which is saved in the)
                     (internal library under ID '2' )

N60 #TRAFO ON        (Select RTCP, kinematic 2)
.
.
N70 #TRAFO ON        (Deselect RTCP)
N80 M30
```

17.5   Modify kinematic characteristics (#KIN DATA)

This command permits characteristics of the active kinematics to be modified when the transformation is active (#TRAFO ON [▶ 677]). This includes, for example, the possibility of specifying which axes should not move in cases of redundant degrees of freedom of a kinematic system.

> **ⓘ** **This function block is available as of CNC Build V3.1.3080.**

> **ⓘ** **Only kinematic ID 210 is supported.**
>
> If the command is used with other kinematics, it has no effect.

Syntax:

#KIN DATA [LOCKDOF { **AX**=*<axis_name>*} | { **AXNR**=..} **]**	Select axes to be locked
#KIN DATA [UNLOCKDOF { AX=*<axis_name>*} | { **AXNR**=..} **]**	Select axes to be released

LOCKDOF	Set the entry in parameter P-CHAN-00458 for the specified axes. As opposed to the variable V.G.KIN[i].LOCK_DOF[<AXIDX>], the command can also be used with active transformation.
UNLOCKDOF	Reset the entry in parameter P-CHAN-00458 for the specified axes. As opposed to the variable V.G.KIN[i].LOCK_DOF[<AXIDX>], the command can also be used with active transformation.
AX=*<axis_name>*	Names of locked axes to be released
AXNR=..	Logical numbers of locked axes to be released, positive integers

Using the #KIN DATA command

The starting point for the following programming example is a simplified configuration of a coupling kinematic with a stationary robot on an X linear axis. When TCP is programmed (axis identifier X), the X linear axis moves first due to the motion priority in P-CHAN-00450.

Extract from parameterisation of channel parameters:

```
gruppe[0].achse[00].log_achs_nr 1
gruppe[0].achse[00].bezeichnung X
gruppe[0].achse[00].default_feed_axis 0
...
gruppe[0].achse[06].log_achs_nr 7
gruppe[0].achse[06].bezeichnung X_LIN
gruppe[0].achse[06].default_feed_axis 0
gruppe[0].achse[07].log_achs_nr 8
gruppe[0].achse[07].bezeichnung X_ROB
gruppe[0].achse[07].default_feed_axis 0
...

trafo[0].id 210
trafo[0].group[0].name LIN_ROB
trafo[0].group[0].chain[0] LIN
trafo[0].group[0].chain[1] ROB
trafo[0].group[0].move_prio[0] LIN
trafo[0].group[0].move_prio[1] ROB

trafo[1].id 91
trafo[1].name LIN
...
trafo[2].id 45
trafo[2].name ROBOT
...
```

The example below locks the X linear axis in block N01. The robot can then only move the programmed motion.

The X linear axis is unlocked in block N03. The axis then moves the program X motion since it has a higher motion priority than the robot.

```
N01 #KIN DATA [LOCKDOF AXNR=7]
N02 G00 G90 X1500
( Cartesian axis positions: X=1500, X_LIN=0, X_ROB=0
N03 #KIN DATA [UNLOCKDOF AX=X_LIN]
N02 G00 G90 X1000
( Cartesian axis positions: X=1000, X_LIN=-500, X_ROB=1500
```

17.6 Positioning without compensation motion (#PTP ON/ OFF, #AX LOCK ALL, #AX UNLOCK ALL)

> **i** Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

`Syntax:`

#PTP ON	Select PTP motion control when transformation is active
#PTP OFF	Deselect PTP motion control when transformation is active

Tool positioning and alignment after kinematic transformation is selected causes a compensation motion in the machine axes because the tool centre point (TCP) is moved on the path. If these compensation motions are undesired, a more time-optimised motion can be executed with the commands listed above.

For 5-axis machines the motion is based on the reference point of the tool head; - in contrast to the TCP, the reference point moves on a straight line at the programmed feedrate (F word) or at rapid traverse (see Fig. blow).

On non-Cartesian machine structures (e.g. robots, tripods) neither the TCP nor the reference point moves on a straight line. The programmed feed (F word) or rapid traverse acts on the machine axes. However at the end of the motion, it is ensured that the TCP is located on the programmed PCS target point.

Motion programming is identical to PCS programming; the controller executes the conversion of PCS coordinates into MCS coordinates. As opposed to using the #WCS TO MCS command, adopting offsets and tool data is executed in the same way as with active kinematic transformation.

NOTICE
When PTP motion control is active, ACS values are displayed in the PCS coordinates in the real-time part of CNC.

Fig. 175: Motion control with/without #PTP

NOTICE
When PTP motion control is active, it is not permitted to select or deselect additional coordinate transformations (#(A)CS ON/OFF, #MCS ON/OFF etc.).

Positioning without compensation motion (PTP)

```
Example of previous figure;:
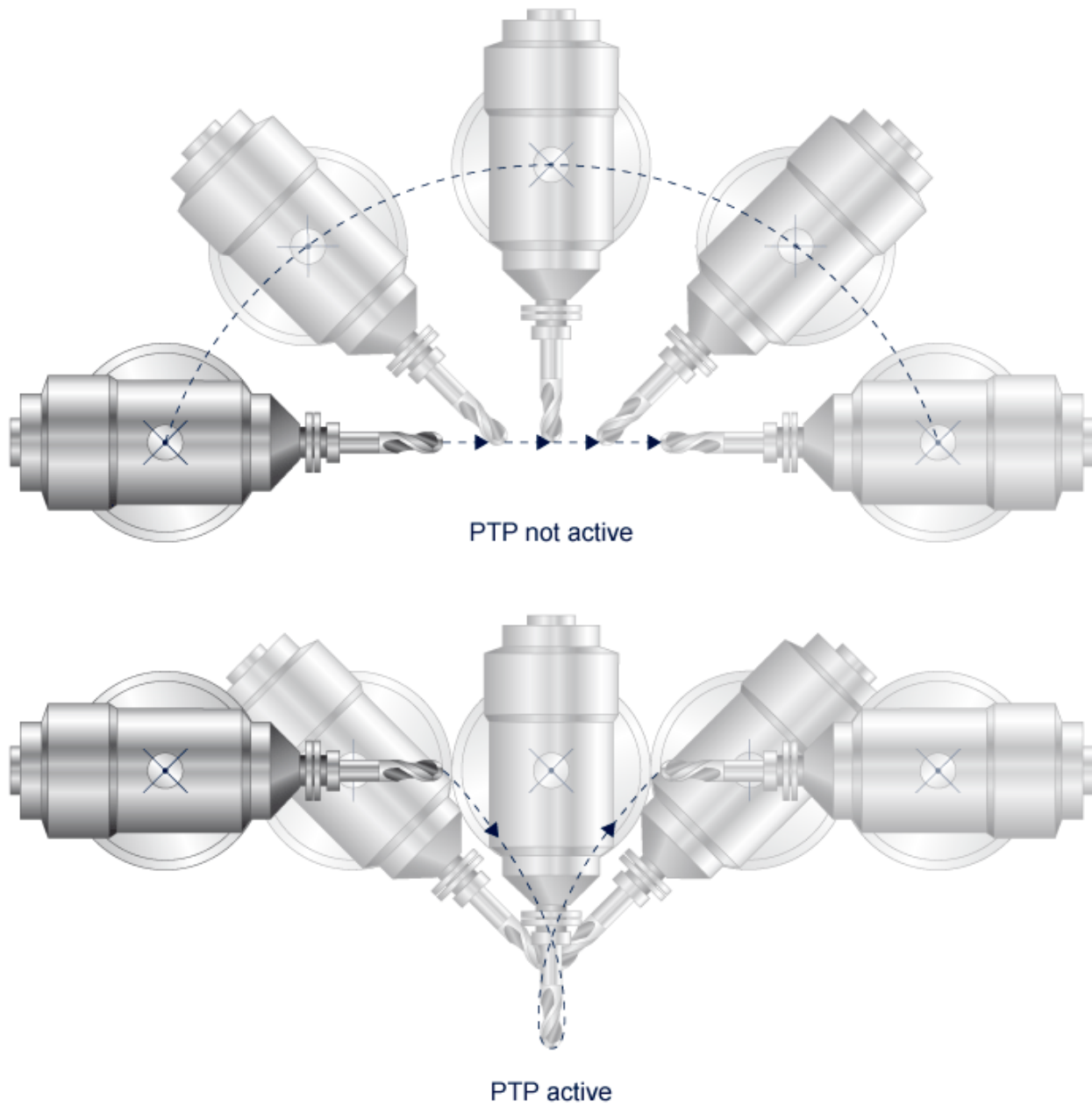
N10 T1 D1                ;Select tool
N20 G00 X-200 Y0 Z0 A90  ;MCS start position
N30 #TRAFO ON            ;Select RTCP, default kinematic
                         ;valid from P-CHAN-00032
N40 #PTP ON              ;Transformation PTP motion on
N50 G00 X200 Y0 Z0 A-90  ;PCS target point
...
N180 #PTP OFF             ;Transformation PTP motion off
N185 G01 X100 Y150 F5000
…
N500 #TRAFO OFF          ;Deselect RTCP
N999 M30
```

```
Example with automatic tool alignment:

N10 T1 D1                ;Select tool
N20 #TRAFO ON            ;Select RTCP, default kinematic)
                         ;valid from P-CHAN-00032
N30 #TOOL ORI CS     ;Align tool, default kinematic
                         ;valid from P-CHAN-00032.
N40 #PTP ON              ;Transformation PTP motion on
N50 G00 X0 Y0 Z100 G90     ;PCS start position
...
N180 #PTP OFF               ;Transformation PTP motion off
N185 G01 X100 Y150 F5000
…
N500 #TRAFO OFF          ;Deselect RTCP
N999 M30
```

ℹ︎ In conjunction with PTP motion control, it may sometimes be necessary to suppress the resulting path motions for certain axes for technological reasons (e.g. In order to remain within the limits of software limit switches).

For this purpose, the #AX LOCK/UNLOCK ALL command temporarily locks motions of single axes when PTP programming is active.

Syntax:

#AX LOCK [{ **AX=**<axis_name> } | { **AXNR=..** } **]**   Select axes to be locked

AX=<axis_name>	Name of axes to be locked
AXNR=..	Logical numbers of axes to be locked, positive integers

Syntax:

#AX UNLOCK ALL	Release locked axes. If several axes are locked, they can only be released together. The release of only one specific axis is not possible.
	If only one single axis is locked, it must also be released with ALL.

Programming of #AX LOCK, #AX UNLOCK ALL is only permitted when #PTP is active.

Locked ACS output axes of the kinematic transformation do not move with G00 and G01.

Positioning without compensation motion (PTP)

Select the axis to be locked or release all locked axes.

```
Correct:
N10 #CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N20 #PTP ON
N30 #AX LOCK[AX=Z AX=B] ;alternative: #AX LOCK[AXNR=3 AXNR=5]
N40 G00 G90 U30
........................
N60 #AX UNLOCK ALL  ;With implicit position request
N70 #PTP OFF
N80 G01 G90 Z0 F3000
N90 G01 U40 F2000
..........

Wrong:
N10 #AX LOCK[AX=Z] ;Programming before PTP
N20 #CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N30 #PTP ON
N40 G00 G90 U30
......................
N60 #AX UNLOCK ALL ;With implicit position request
N70 #PTP OFF
N80 G01
..........
N200 #CYL OFF
```

17.7 Coordinate systems

The number of possible coordinate systems is limited to [SYSP// Number 6.17].

17.7.1 Standard programming

17.7.1.1 Defining a machining coordinate system (#CS DEF, #CS ON/OFF, #CS MODE ON/OFF)

```
Syntax of CS programming
```
Defining and storing a CS:

#CS DEF [**[**<*CS-ID*>**]**] **[** *<v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3>* **]**

Define and store with simultaneous activation:

#CS ON [**[**<*CS-ID*>**]**] **[** *<v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3>* **]**

#CS ON [<*CS-ID*>**]**	Select a stored CS
#CS ON	Select the last CS defined
#CS OFF	Deselect the last CS activated
	The CS-ID parameter may not be programmed here since it is only permitted to deselect the last CS activated.
<CS-ID>	Coordinate system ID. The CS-ID is assigned the default value 1 at program start. If the CS-ID is **not** programmed with #CS DEF or #CS ON, the next free CS-ID is calculated automatically. However, a CS of this type is not longer available after it is deselected with #CS OFF!
	At the same time, a maximum of 5 CS definitions can be stored.
<vi>	Components of the translatory offset vector in [mm, inch]. (They refer to the main axes in the sequence contained in G17).
<φi>	Angle of rotation in [°].
	Note: Separate the translation and rotation components by commas. Gaps in the sequence can be marked by consecutive commas ", ,". However, to improve legibility, we recommend programming these components with 0.
	Example: [,,10,,,45] ↔ [0,0,10,0,0,45]
	Abbreviated programming of components is also permissible:
	Example: [0,0,10]　　　translation only
	[0,0,10,0,30]　　rotation with 2 rotation angles
#CS OFF ALL	Deselect all machining coordinate systems CS

A CS (PCS processing coordinate system) is characterised by the relative offset (V2 in figure below) and the rotation relative to the current work piece coordinate system (WCS). Current zero offset, clamp position offset and reference point offset (V1 in figure below) determine the position of the CS relative to the machine coordinate system (MCS).

Fig. 176: Machining on an inclined plane

Default setting of rotation sequence and rotation mode:

If the rotations φ1, φ2 and φ3 are defined, they are executed in the default setting in the mathematical positive direction (figure below) in the sequence as listed below:

1st rotation at angle φ3 about the 3rd axis (e.g. z)

2nd rotation at angle φ2 about the new 2nd axis (e.g. y´)

3rd rotation at angle φ1 about the 1st axis (e.g. x´´)

This rotation sequence is also referred to as YAW - PITCH - ROLL. The rotations always refer here to the new axes of the currently rotated CS (rotation mode).

(The specified axis sequence of the axes always corresponds to the sequence of the main axes at G17, irrespective of G17/G18/G19).

BECKHOFF

Fig. 177: Definition of a CS by 3 rotations referred to the new axes

Free definition of rotation sequence and rotation mode:

Every orientation in space can be reached by concatenating three basic rotations. There are 6 possible rotation sequences about two axes (known as classic Euler angles) and 6 rotation sequences about 3 axes (known as Tait-Bryan angles).

Rotations either refer to fixed axes in space (extrinsic rotation) or to the new axes of the currently rotated CS (YAW, intrinsic rotation). The figure below shows this difference compared to the figure above.

Fig. 178: Definition of a CS by 3 rotations about fixed axes in space

The rotation sequence can be configured with P-CHAN-00394. The configured rotation sequence can be changed in the NC program by the following command.

#CS MODE ON [ROTATION_SEQUENCE=*<rot_sequence>*]

<rot_sequence>	Rotation sequence as string according to:	
	Euler angles	Tait-Bryan angles
	XYX, XZX, YXY, YZY, ZXZ, ZYZ	XYZ, XZY, YXZ, YZX, ZXY, **ZYX** (Default)

Syntax of Deselecting and restoring the default setting:
#CS MODE OFF [ROTATION_SEQUENCE]

The rotation mode is also configure by P-CHAN-00393. The configured rotation sequence can be changed in the NC program by the following command.

Syntax of Rotation mode:
#CS MODE ON [ROTATION_MODE_FIXED]

ROTATION_MODE_FIXED	Rotation about (fixed) axes in space of the coordinate system at rotation state

Syntax of Deselecting and restoring the default setting:
#CS MODE OFF [ROTATION_MODE_FIXED]

Without #CS MODE OFF [...] all settings remain active until main program end (M30) or RESET. After next program start, the default settings are again valid.

A CS that has been defined via #CS DEF [*<CS-ID>*] [...] or CS ON [*<CS-ID>*] [...] is stored in relation to its position with respect to the current MCS and can be re-selected via #CS ON [<CS-ID>] without a specification of parameters. However, if the overall offset in the MCS is modified in the meantime, the CS has a new position relative to the MCS.

Zero offsets and reference point offsets may be programmed in the CS during machining. However, these values are only valid until the CS is deselected and are not saved. The axis designations are retained in the CS.

BECKHOFF

Example 1

```
N005 P1 = 2
N010 #CS DEF [1][P1,15,5,20,30,45]  (Define and store a CS)
                                     (under ID 1)
                                     (Relative offsets: X2, Y15, Z5)
                                     (Rotation Z:45° Y':30° X'':20°)
N020 #CS ON[1]                       (Activate CS with ID 1)
:
N100 #CS OFF                         (Deselect CS with ID 1)
:
N200 P1=10
N210 #CS ON [P1,15,5,2,3,60]         (Define and activate a CS)
                                     (with the automatically defined ID 2)
:
N300 #CS OFF     (Deselect the last CS activated (ID 2))
                 (Then the CS is deleted with ID 2)
:
N400 M30
```

Example 2

```
N05 P1 = 2
N10 #CS DEF [3][P1,15,5,2,3,4.5]     (Define and store)
                                     (a CS with ID 3)
N20 #CS DEF [2][P1,15,5,2,3,4.5]     (Define and store)
                                     (a CS with ID 2)
N30 #CS DEF [5][0,1,2,0,30,30]       (Define and store)
                                    (a CS with ID 5)
N30 #CS ON                           (Activate the CS with the)
                                     (last ID 5 programmed)
:
N50 #CS OFF
N60 #CS ON[3]                        (Activate CS with ID 3)
:
N80 #CS OFF
N90 #CS DEF [3][1.1. 2,1,3,0,0.33]     (Redefine the CS with ID 3)
:
M30
```

Example 3

If several coordinate systems are selected in succession, e.g. with CS ON [...] (without *CS_ID*), they form a new linked total CS. This must be deselected step by step by a corresponding #CS OFF.

It is permitted to select combined CS's with and without CS IDs but this is not recommended for the sake of program clarity.

Example of multiple programming of CS (without *CS_ID*):

```
N010 #CS ON [0,0,0,0,0,20]  (Define and activate a CS with)
                            (automatically defined ID 1)
                            (No offsets, only rotation 20° about Z)
:
N050 #CS ON [0,0,0,0,0,30]  (Define and activate a CS with)
                            (automatically defined ID 2)
                            (No offsets, only rotation 30° about Z)

->(This results in a total CS with a rotation of 50° about Z)
:
N100 #CS OFF    (Deselect the CS with ID 2, then the CS is)
                (deleted with ID 2.)

->(CS remains active with ID 1 with a rotation of 20° about Z)
:
N200 #CS OFF    (Deselect the CS with ID 1, then the CS is)
                (deleted with ID 1 and all CS's are deselected.)
:
N400 M30
```

Changing the rotation mode and rotation sequence

```
Switching over from rotations about fixed axes (extrinsic rotation)
:
#CS MODE ON [ROTAION_MODE_FIXED]     ;Rotation about fixed axes
#CS MODE ON [ROTATION_SEQUENCE=XYZ] ;Rotation about axes X->Y->Z
#CS ON [0,0,0,90,0,90]               ;Rotation X:90° Y:0° Z:90°
:
#CS OFF
#CS MODE OFF [ROTATION_SEQUENCE]  ;Return to rotation sequence ZYX
#CS MODE OFF [ROTAION_MODE_FIXED] ;Deselect rotation mode about fixed axes
:
Switching over to rotations about new axes (intrinsic rotation)
:
#CS MODE ON [ROTATION_SEQUENCE=XYZ] ;Rotation about axes X->Z'->X''
#CS ON [0,0,0,90,0,90]               ;Rotation X:90° Y:0° Z:90°
:
#CS OFF
#CS MODE OFF [ROTATION_SEQUENCE]  ;Return to rotation sequence ZYX
```

17.7.1.2   Defining/activating a coordinate system for fixture adaptation (#ACS)

The fixture adaptation coordinate system (ACS) compensates a sloping position of the workpiece or workpiece pallet. It is defined, selected and deselected in the same way as the machining coordinate system (CS).

`Syntax of ACS programming`

Defining and storing an ACS:

#ACS DEF [**[**<*ACS-ID*>**]**] **[** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

Define and store with simultaneous activation:

#ACS ON [**[**<*ACS-ID*>**]**] **[** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*>

#ACS ON [<*ACS-ID*>**]**	Selecting a stored ACS
#ACS ON	Selecting the last ACS defined
#ACS OFF	Deselecting the last ACS activated
	Parameter ACS-ID may not be programmed here since it is only permitted to deselect the last ACS activated.
<ACS-ID>	Coordinate system ID. The ACS-ID is assigned the default value 1 at program start. If the CS-ID is not programmed with #CS DEF or #CS ON, the next free CS-ID is calculated automatically. ACS programmed in this way are not available again after they are deselected with #ACS OFF.
<vi>	Components of the translatory offset vector in [mm, inch]. (These refer to the main axes in the sequence contained in G17).
<φi>	Angle of rotation in [°].
	Note: Separate the translation and rotation components by commas. Gaps in the sequence can be marked by consecutive commas ", ,". However, to improve legibility, we recommend programming these components with 0.
	Example: [,,10,,,45] ↔ [0,0,10,0,0,45]
	Abbreviated programming of components is also permissible:
	Example: [0,0,10] translation only
	[0,0,10,0,30] rotation with 2 rotation angles

The ACS is modal and may be selected and deselected independent of a CS.

Zero offsets and reference point offsets may be programmed in the ACS. However, they are only valid until the ACS is deselected and they are not stored.

#ACS OFF ALL   Deselect all ACS

BECKHOFF

ACS example 1

```
N005 P1 = 2
N010 #ACS DEF [1][P1,15,5,20,30,45](Define and store)
                                    (a CS under ID 1)
                                    (Relative offsets: X2, Y15, Z5)
                                    (Rotation about Z:45°Y': 30° X'':20°)
N020 #ACS ON[1]                     (Activate ACS with ID 1)
:
N100 #ACS OFF                        (Deselect ACS with ID 1)
:
N200 P1=10
N210 #ACS ON [P1,15,5,2,3,60]         (Define and activate an ACS)
                                      (with the automatically defined ID 2)
:
N300 #ACS OFF      (Deselect the last ACS activated (ID2))
                   (Then the ACS is deleted with ID 2)
:
N400 M30
```

ACS example 2

```
N5 P1 = 2
N10 #ACS DEF [1][10,15,5,2,3,4.5]      (Define and store)
                                        (an ACS with ID 1)
N20 #ACS DEF [3][0.15,5,2,3,4,5]       (Define and store)
                                        (an ACS with ID 3)
N30 #ACS DEF [P1+3][2*P1,1,2,0,30,30]  (Define and store)
                                        (an ACS with ID 5)
N30 #ACS ON                            (Activate the ACS with the)
                                        (last ID 5 programmed)
:
N50 #ACS OFF
N60 #ACS ON[3]                          (Activate ACS with ID 3)
:
N80 #ACS OFF
N90 #ACS DEF [3][0.1. 2,1,3,0,0.3]      (Redefine the ACS with ID 3)
:
M30
```

ACS example 3

If several coordinate systems are selected in succession, e.g. with ACS ON [...] (without A*CS_ID*), they form a new linked total ACS. This must be deselected step by step by corresponding #ACS OFF.

It is permitted to select the combined ACS with and without ACS IDs but it is not recommended for the sake of NC program clarity.

Example of multiple programming of ACS (without A*CS_ID*):

```
N010 #ACS ON [0,0,0,0,0,20]  (Define and activate an ACS with)
                             (automatically defined ID 1)
                             (No offsets, only rotation 20° about Z)
N020 #ACS ON [0,0,0,0,0,30]  (Define and activate an ACS with)
                             (automatically defined ID 2)
                             (No offsets, only rotation 30° about Z)

->(This results in a total ACS with a rotation of 50° about Z)
:
N100 #ACS OFF    (Deselect the ACS with ID 2, then the ACS is)
                 (deleted with ID 2.)

->(ACS remains active with ID 1 with a rotation of 20° about Z)
:
N200 #ACS OFF     (Deselect the ACS with ID 1, then the ACS is)
                  (deleted with ID 1 and all ACS are deselected.)
:
N400 M30
```

17.7.1.3 Linkage of coordinate systems

New coordinate transformations may be formed by combining ACS and CS.

Fig. 179: The combination of ACS and CS permits machining on an inclined plane with a slanted clamped workpiece.

Several ACS and CS are linked separately in the sequence they are selected. The resulting ACS is then linked to the resulting CS for overall transformation. Linkage always takes place with the ACS first irrespective of programming.

A maximum of 10 ACS/CS combinations can be linked to form an overall transformation.

Individual ACS's are deselected in the opposite sequence to selection. The same applies to the CS. To simplify this, #(A)CS OFF is programmed without an ID parameter (see the two figures in.Defining a machining coordinate system (#CS DEF, #CS ON/OFF, #CS MODE ON/OFF) [▶ 690]).

Linkage of coordinate systems

```
N100 #CS ON [1]              (                     CS[1])
N110 #ACS ON [2]            (ACS[2] o             CS[1])
N120 #ACS ON [1]            (ACS[2] o ACS[1] o    CS[1])
N130 #CS ON [2]             (ACS[2] o ACS[1] o    CS[1] o CS[2])
N140 #ACS OFF              (ACS[2] o             CS[1] o CS[2])
N140 #CS OFF               (ACS[2] o             CS[1])
N150 #ACS OFF              (                     CS[1])
N160 #CS OFF
M30
```


Fig. 180: Activating or changing the ACS without deselecting the CS's which are already active

It must be noted that the relative linkage of ACS or CS may in general result in a changed sequence of selection and lead to different results (see figure below).

Fig. 181: Result of a CS linkage depending on the sequence of selection (CS[1] - CS[2] or CS[2] - CS[1]).

The CS (or ACS) with the same ID may also be selected several times and linked to itself.

Linkage of coordinate systems

```
N10 #CS DEF[1][0,0,0,0,0,20]
N20 LL TEILEPRG                    (Contour in the system X-Y)
N30 #CS ON[1]
N40 LL TEILEPRG                    (X'-Y')
N50 #CS ON[1]
N60 LL TEILEPRG                    (X''-Y'')
N70 #CS OFF
N80 #CS OFF
M30
```


Fig. 182: Linkage of coordinate systems

The following NC commands store the currently active overall transformation:

Syntax:

#CS DEF ACT [<*CS_ID*>]

#ACS DEF ACT [<*ACS_ID*>]

As opposed to sequential deselection of the (A)CS by (A)CS OFF, the following NC commands can directly deselect the partial transformations formed from the linkage of CS or ACS.

Syntax:

#CS OFF ALL   Deselect all CS
#ACS OFF ALL   Deselect all ACS

BECKHOFF

Linkage of coordinate systems

```
N10 #CS ON[3]
N20 #CS ON[4]
N30 #CS DEF ACT[5]          (Store CS[3] o CS[4] in CS[5])
N31 #CS OFF ALL            (Deselect all CS)
N32 #ACS ON[3]
N33 #ACS ON[4]
N34 #ACS DEF ACT[5]        (Store ACS[3] o ACS[4] in CS[5])
N35 #ACS OFF ALL           (Deselect all ACS)
N36 X0 Y0 Z0
N360 #CS ON [5]
N370 #ACS ON[5]
N380 #CS DEF ACT[1]        (Store ACS[5] o CS[5] in CS[1])
N390 #ACS OFF ALL
N400 #CS OFF ALL
N500 #CS ON                (Select CS[1])
N510 #CS OFF
M30
```

Version: 1.238

TF5200 | TwinCAT 3 CNC
Programming manual

17.7.1.4 Define/activate a basic coordinate system (#BCS)

> **ⓘ** **This function is available as of CNC Build V3.1.3079.36.**

The purpose of the basic coordinate system (BCS) is to compensate for offsets from the basic system.

`Syntax of BCS programming`

Defining and storing a BCS:

#BCS DEF [**[**<*BCS-ID*>**]**] **[** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

Define and store with simultaneous activation:

#BCS ON [**[**<*BCS-ID*>**]**] **[** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

#BCS ON [<*BCS-ID*>**]**	Select a stored BCS
#BCS ON	Select the last BCS defined
#BCS OFF	Deselect the last BCS activated
	The BCS-ID parameter may not be programmed here since it is only permitted to deselect the last BCS activated.
<*BCS-ID*>	Coordinate system ID. The BCS-ID is assigned the default value 1 at program start. If the BCS-ID is not programmed with #BCS DEF or #BCS ON, the next free BCS-ID is calculated automatically. However, a BCS of this type is not longer available after it is deselected with #BCS OFF!
<*vi*>	Components of the translatory offset vector in [mm, inch]. (These refer to the main axes in the sequence contained in G17).
<*φi*>	Angle of rotation in [°].
	Note: Separate the translation and rotation components by commas. Gaps in the sequence can be marked by consecutive commas ", ,". However, to improve legibility, we recommend programming these components with 0.
	Example: [,,10,,,45] ↔ [0,0,10,0,0,45]
	Abbreviated programming of components is also permissible:
	Example: [0,0,10] translation only
	[0,0,10,0,30] rotation with 2 rotation angles

The BCS is modal and may be selected and deselected independently of a CS/ACS.

Zero offsets and reference point offsets may be programmed in the BCS. However, these values are only valid until the BCS is deselected and are not saved.

#BCS OFF ALL	Deselect all BCS

Fig. 183: Linkage with basic coordinate system #BCS

17.7.1.5   Effector coordinate system (#ECS ON/OFF)

ℹ This function is not included in the scope of the standard license!
The use of this feature requires a license for the "Transformations" extension package.

The effector coordinate system is mainly used to execute a withdrawal strategy after tool breakage, NC reset or program abort when machining takes place with a tool in any alignment. The ECS is determined by reversing the command TOOL ORI CS (section Orienting tool (#TOOL ORI CS) [▶ 681]).. Instead of aligning the tool on the machining plane is determined here perpendicular to the tool axis.

Syntax:

| **#ECS ON** | Select ECS |
| **#ECS OFF** | Deselect ECS |

No other coordinate system (CS) may be active when ECS is activated.

The ECS is then determined from the positions of the alignment axes so that its Z axis is in parallel to the current tool axis. The position of the X and Y axes are then undefined (arbitrary) and must therefore be predefined internally. The origin of the ECS is generally located outside of the tool tip or tool axis, i.e. a collision-free tool withdrawal is only guaranteed by relative path motions along the effector Z axis.

Fig. 184: Machining in a slanting hole

Effector coordinate system (ECS)

```
N01 #TRAFO ON                    (Select kinematic)
N05 #CS ON[1.5,0,32,14.5,0,45] (Select a BCS)
N10 #TOOL ORI CS
N15 X0 Y0 Z0
N20 LL TEILEPRG    (Subroutine call for contour machining)
...
(Tool breakage, NC reset)
(Withdrawal strategy)
N01 #TRAFO ON         (Select kinematic)
N05 #ECS ON           (Calculate the ECS)
                      (depending on position of the orientation)
                      (axes)
N10 G91 G01 F200
N20 Z62               (Withdrawal motion along the tool or ECS-Z axis)
:
N400 M30
```

```
N01 #TRAFO ON                    (Select kinematic)
N05 #CS ON[1.5,0,32,14.5,0,45] (Select a BCS)
N10 #TOOL ORI CS
N15 X0 Y0 Z0
```

17.7.1.6   Temporary transition to the machine coordinate system (#MCS ON/OFF)

ℹ️   Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

The MCS functionality temporarily deactivates the active kinematics and/or Cartesian transformation as well as all offsets included in the axes in order to position machine axes directly. After leaving the MCS, the state before selection is restored.

For example, a tool change often requires the approach to a defined tool change position with known machine origin coordinates. Approaching this machine position may pose a problem in the CS since the CS axes are positioned by the NC program.

Syntax:

| #MCS ON [EX TOOL] | Activate temporary transition to MCS |
| #MCS OFF | Deactivate temporary transition to MCS |

| EX TOOL | A tool change in the MCS does not include tool offsets in order to permit direct positioning of the machine axes. This only takes place with #MCS OFF. |

The MCS has no restrictions regarding use of NC functionality. However, it is not possible to select RTCP/ TLC , CS, ACS or ECS.

In addition, programmed offsets are only valid in the MCS until they are deselected and are not stored.

When the axis configuration is changed by external axis exchange (e.g. #CALL AX..) in the MCS, it must be noted that a definite axis configuration is required to reactivate kinematic and/or Cartesian transformation.

Temporary transition to the machine axis coordinate system (MCS)

```
N10 #TRAFO ON
N20 #CS ON[1.5,0,32,14.5,0,45]    (Select a BCS)
N30 G01 G90 F5000
N40 X0 Y0 Z0
N50 #MCS ON EX TOOL    (Transition to machine CS with the option)
                       ('EX TOOL' - tool is only included)
                       (with MCS OFF)
N60 T1 D1              ( Tool length is NOT included here,
                       ( but in line N70)
;...
N70 #MCS OFF           (Deactivate MCS; RTCP and CS are)
                       (reactivated)
N100 #TRAFO OFF
N110 #CS OFF
N400 M30
```

17.7.1.7 Auxiliary functions for coordinate transformation (#WCS TO MCS, #MCS TO WCS)

ⓘ Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

On 5-axis machines and machines with non-Cartesian axis structure (e.g. hexapods), two typical variants are used to define path motions.

1st case: The user programs the contour in space with circles, straight lines or polynomials and the tool tip (TCP) is moved along the path depending on the programmed contour.

2nd case: The user programs the target point in space or workpiece coordinates (WCS) which are mapped onto machine coordinates (MCS). Depending on the maximum possible velocities in the axes, the TCP moves on a non-predictable curve (PTP). Due to the loss of TCP motion along a curve in space, the PTP motion is generally faster than the TCP path motion.

The above mentioned mapping of programmed WCS target points onto MCS target points (backward transformation) can be executed in the NC program by the following NC commands. The user must then calculate the approach of the calculated MCS target points explicitly in absolute dimensions.

For example, this method can be used to move each single axis sequentially out of a collision area.

NOTICE
These commands only can be used with inactive transformation (#TRAFO OFF) and when the coordinate system is inactive (#CS OFF (ALL)).

```
Syntax of Calculating machine coordinates (MCS) from workpiece coordinates (WCS):
```
#WCS TO MCS [[**CS..**] [**KIN**] **]**

CS..   Calculate the target points taking into consideration an inactive Cartesian transformation with a specific valid ID.

KIN   Calculate the target points taking into consideration the currently valid kinematic transformation which is however inactive.

The following axis-specific variables are provided to declare the WCS target points and store the calculated MCS target points. These variables permit write and read access:

V.A.WCS.*   Access to the axis-specific auxiliary variable "workpiece coordinate" (WCS). The value does not correspond to the programmed workpiece coordinates when transformation is <u>active</u>.

V.A.MCS.*   Access to the axis-specific auxiliary variable machine coordinate (MCS). The value does not correspond to the programmed machine coordinates when transformation is <u>inactive</u>.

Auxiliary functions for coordinate transformations

The NC calculates the MCS target points using the Cartesian and kinematic backward transformation of specific WCS points. The user can access the calculated MCS positions using NC programming and reuse them in the NC program. The user defines the transformations which are to be executed by specifying the keywords and the associated CS IDs.

```
N02 #CS DEF[1][0,0,0,0,0,45]
N00 #KIN ID [1]
N10 V.A.WCS.X=10
N20 V.A.WCS.Y=10
N30 V.A.WCS.Z=100
N40 #WCS TO MCS[CS1, KIN] (Transformation of TCS in MCS)
N50 G00 Z=V.A.MCS.Z (Move single axes to MCS target points)
N60 G00 X=V.A.MCS.X
N70 G00 Y=V.A.MCS.Y
N…
:
```

Incorrect use:

```
N05 #CS DEF[1][0,0,0,0,0,45]
N10 #CS ON[1]
N20 V.A.WCS.X=100
N30 V.A.WCS.Y = 0
N40 V.A.WCS.Z = 0
N50 #WCS TO MCS [CS1] <- not permitted since a CS is active


N05 #KIN ID[12]
N10 #TRAFO ON
N20 V.A.WCS.X=100
N30 V.A.WCS.Y = 0
N40 V.A.WCS.Z = 0
N50 #WCS TO MCS [KIN] <- not permitted since a transformation is active
```

The inverse mapping of MCS target points to WCS target points (forward transformation) is executed with the following command. For example, this method can be used to map measured values in the WCS (normally determined in the MCS).

```
Syntax of Calculating workpiece coordinates (WCS) from machine coordinates (MCS):
```
#MCS TO WCS [[**CS..**] [**KIN**] **]**

CS..	Calculate the target points taking into consideration an inactive Cartesian transformation with a specific valid ID.
KIN	Calculate the target points taking into consideration the currently valid kinematic transformation which is however inactive.

17.7.1.8 Auxiliary function to calculate motion limits in the workpiece coordinate system (#GET WCS POSLIMIT)

The following command calculates the limits of a motion in the current workpiece coordinate system (WCS) in the direction of a programmed motion vector with the components (VC1, VC2, VC3). The vector components need not be specified in standardised form. The basis for path limiting are the axis-specific software limit switches. Based on these values, the controller calculates the motion limits in the current coordinate system.

```
Syntax:
```
#GET WCS POSLIMIT [VC1=.. **VC2=**.. **VC3=**.. **]**

VC1=..,	
VC2=..,	Components of direction vector, REAL number
VC3=..	

The following global variables read out the result of the calculation for the first three axes in the coordinate system.

V.G.WCS_POSLIMIT_1   Motion limit in the first main axis in WCS

V.G.WCS_POSLIMIT_2   Motion limit in the second main axis in WCS

V.G.WCS_POSLIMIT_3   Motion limit in the third main axis in WCS

NOTICE

The programmed motion direction must be absolutely retained for the real executed motion, otherwise the calculated motion limits are incorrect.

In the same way, no rotary axes may be programmed when kinematic transformation (#RTCP) is active.

Auxiliary function to calculate motion limits in the workpiece coordinate system

```
N05...
N10 G98 X-100 Y-100 Z-100        (Shift negative software limits)
N20 G99 X200 Y200 Z300         (Shift positive software limits)
N30 #ECS ON                       (Select effector coordinate system)
N40 # GET WCS POSLIMIT [VC1=0,VC2=0,VC3=1] (Calculate WCS motion limit)
N50 G01 G90 Z=V.G.WCS_POSLIMIT_3 F2000 (Approach WCS motion limit in Z)
N60 #ECS OFF                            (Deselect effector coordinate system)
N70...
```

Example of correct use of the command
```
N05...
N10 #CS ON[0,0,0,0,0,45]
N20 #GET WCS POSLIMIT [VC1=1 VC2=1 VC3=0]
N25 G01 G90 F2000
N30 X=V.G.WCS_POSLIMIT_1 Y=V.G.WCS_POSLIMIT_2 Z=V.G.WCS_POSLIMIT_3
N40 #CS OFF
N50...
```

Wrong, resulting motion direction does not correspond to setting:
```
N05...
N10 #CS ON[0,0,0,0,0,45]
N20 #GET WCS POSLIMIT [VC1=1 VC2=1 VC3=0]
N25 G01 G90 F2000
N30 X=V.G.WCS_POSLIMIT_1 Y=V.G.WCS_POSLIMIT_2
N40 Z=V.G.WCS_POSLIMIT_3
N50...
```

Fig. 185: Example of motion limits in the ZX plane in the current WCS

17.7.2   Extended programming

In this programming type, a coordinate system (CS) is addressed by a name instead of by an ID number. The stacking and formation of a CS stack is determined by the sequence of programmed definitions. Additional NC commands are provided to activate and adapt the CS stack later in the program sequence.

17.7.2.1   Defining and linking coordinate systems (#CS ADD)

Syntax of Defining and linking CS

#CS ADD [<*name*>**] [** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

<name>	Name of the CS with maximum of 8 characters This name can address the CS using all other appropriate NC commands.
	The names ACS, IMCS and MCS have a fixed present meaning and may not be assigned freely. (See #TRANSFORM command).
	A maximum of 5 CS definitions can be defined and linked.
<vi>	3 components of the translatory offset vector in [mm, inch]. (They refer to the main axes in the sequence contained in G17).
<φi>	3 angles of rotation in [°].

Linking to a CS stack is formed by programming a #CS ADD sequence. The new CS programmed last is always the top CS in the stack. The individual CS's are linked in the order of programming from bottom to top depending on the stack structure. The translatory and rotary shifts always refer to the origin of the previous CS. The same rules apply to the rotation sequence and rotation mode as for standard CS programming.

The CS stack can be used at interpolator level to display the currently transformed coordinates.

Fig. 186: Structure of a CS stack with #CS ADD

Link 3 CS's with #CS ADD to form a stack

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
Nxx M30
```

17.7.2.2   Selecting/activating a coordinate system (#CS SELECT)

The link must be activated so that newly programmed positions act on the correct CS of the CS stack. The following NC command is provided for this function:

```
Syntax of Selecting/activating CS:
```

#CS SELECT [<*Name*>**]**

<name>	Name of the CS to be activated or the linked section of a CS stack with maximum of 8 characters.

After selecting a CS, all the programmed positions apply only to this CS. All the higher Cs's in the CS stack are only used for display purposes. #CS SELECT cannot be used without at least defining a CS or CS stack in advance.

Fig. 187: Activating a CS stack with #CS SELECT

Activate a CS link based on a CS stack

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
N60 #CS SELECT [WCS]    ;Active CS link ICS-WCS
:
N80 #CS SELECT [WCS]    ;Active CS link ICS-WCS-PCS
:
Nxx M30
```

17.7.2.3   Selecting/activating the machine coordinate system (#CS SELECT[MCS])

A kinematic transformation when activated generates a predefined machine CS. The origin of this CS is at the machine zero point and it is assigned the fixed name of MCS. It may be used in combination with the selection command #CS SELECT. If a user-specific CS is programmed with this name via #CS ADDm an error message is output.

`Syntax of Selecting/activating a predefined MCS:`

#CS SELECT [MCS]

Fig. 188: Activate machine CS with #CS SELECT [MCS]

> **ℹ** **Limit #CS SELECT[MCS] and #MCS ON**
>
> The command #CS SELECT [MCS] activates the first Cartesian CS at the machine zero point when a kinematic transformation is active.
>
> By contrast, the command #MCS ON deactivates all active Cartesian and kinematic transformations and offsets to permit the machine axes to be positioned directly.

17.7.2.4 Changing the definition of a coordinate system (#CS SET)

Syntax of Changing the definition of a CS:

#CS SET [<*name*>**] [** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

<*name*>	Name of the CS to be changed with maximum of 8 characters
<*vi*>	3 components of the translatory offset vector in [mm, inch]. (They refer to the main axes in the sequence contained in G17).
<*φi*>	3 angles of rotation in [°].

The command #CS SET changes the definition of a CS already defined with #CS ADD. It is irrelevant whether the CS to be changed is an active CS (#CS SELECT) or a CS in the link of a CS stack. The new translatory components always refer to the next CS below.

After the CS is changed, all the positions are recalculated so that the changed CS data are also included in the display.

Fig. 189: Change a CS definition with #CS SET

Change a CS definition based on a CS stack

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
N80 #CS SET [ICS] [20,10,0,0,0,10]   ; Change definition of ICS
:
Nxx M30
```

17.7.2.5 Deleting and removing coordinate system links (#CS DEL)

Syntax of Deleting and removing a CS stack:

#CS DEL

When #CS DEL is programmed, the topmost CS in a CS stack is always deleted. When an active CS (#CS SELECT) is deleted, the new topmost CS automatically becomes the active CS.

Fig. 190: Delete a CS with #CS DEL

A CS stack can be deleted either steop by step with #CS DEL or in one step with:

#CS DEL ALL

This command deletes all CS's. Any MCS present becomes the active CS.

Fig. 191: Delete all CS's with # CS DEL ALL

Delete with #CS DEL and #CS DEL ALL based on a CS stack with subordinate MCS.

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
N60 #CS SELECT [WCS]    ;Active CS link MCS-ICS-WCS
:
N80 #CS DEL         ;Delete PCS; WCS becomes topmost CS
                    ;Active CS link MCS-ICS-WCS
N90 #CS DEL         ;Delete WCS; active CS link MCS-ICS
:
or alternatively
N80 #CS DEL ALL   ;Delete all CS's, MCS becomes topmost active CS
:
Nxx M30
```

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
```

17.7.2.6   Tracking a coordinate system (#CS TRACK)

A defined CS is shifted so that the current actual position corresponds to a specified tracking position. This command causes the physical axes not to move. The tracked CS need not be the active CS (#CS SELECT).

All CS's in a CS stack above the tracked CS then refer relatively to the new position of the tracked CS.

`Syntax of Defining a tracked CS:`

#CS TRACK [<*name*>**] [**<*POS_X*>**,** <*POS_Y*>**,** <*POS_Z*>**]**

<*name*>	Name of the tracked CS with maximum of 8 characters
<*POS_X, Y, Z*>	3 components of the new actual position in [mm, inch] in the tracked CS.

Fig. 192: Track a PCS in XY with #CS TRACK, P remains constant

Track a CS with #CS TRACK

```
:
N10 #CS ADD [PCS][...] [...]
N20 #CS SELECT [PCS]
N30 G0 X150 Y100 Z0 ; Move to PCS position P(150,100,0)
:
N50 #CS TRACK [PCS] [40,30,0] ;P(150,100,0) in tracked PCS: P(40,30,0)
:
M30
```

17.7.2.7   Reading the overall offset in the active coordinate system

In the CS activated by #CS SELECT, the total offset to the reference machine CS (MCS) formed by linking can be read using the following variables:

`Syntax of V.G. Variables for reading overall offset:`

V.G.SELECTED_CS.TRANS.X	Overall offset in 1st main axis in [mm, inch]
V.G.SELECTED_CS.TRANS.Y	Overall offset in 2nd main axis in [mm, inch]
V.G.SELECTED_CS.TRANS.Z	Overall offset in 3rd main axis in [mm, inch]

17.7.2.8   Coordinate transformation between coordinate systems (#TRANSFORM)

The #TRANSFORM command is used to convert any coordinates of a 3D point from one stack level to another stack level in the NC program based on the current CS stack. The transformation calculation required for this consists of a combination of kinematic and Cartesian forward or backward transformation, depending on the structure of the CS stack.

Auxiliary and additional axes are only taken into account in the transformation calculation with regard to their zero point displacements between the stack planes.

The #TRANSFORM command is also used for transformations that are already active (#CS SELECT) in the channel.

Syntax of Transformation command:

#TRANSFORM [<CS_source>**] [**<CS_dest> **] [**POS_X>**,** <POS_Y>**,** <POS_Z> **]** [**[** {<POS_aux>=..} **]]**

<CS_source>	Name of the tracked CS with maximum of 8 characters.
<CS_dest>	Name of the destination CS with maximum of 8 characters.
<POS_X, Y, Z>	3 components of the point to be transformed in [mm, inch] in the input CS.
<POS_aux>=..	Input coordinates of the auxiliary and additional axes

> **i** An error message is generated it:
> - the name of a coordinate system is not defined
> - 3 input coordinates are not programmed

The result of the transformation calculation is supplied in special axis-specific variables (V.A.). The axis can be programmed either by name or by axis index:

Syntax of Result variables

V.A.TRANSFORM.<axis_name>   Axis-specific coordinate after calculation in [mm, inch]

or

V.A.TRANSFORM[<axis_index>**]**   Axis-specific coordinate after calculation in [mm, inch]

The following examples show the schematic use of the #TRANSFORM command in a CS stack that consists of kinematic and Cartesian transformations. Specifying the input and target coordinate system determines whether the calculation requires a forward or backward transformation.

Example 1, basic configuration is a 5-axis kinematic with CS stack.

Transformation for the point (50,0,10) with tracking axis positions A10 and B20 from the ACS system to the WCS system. A kinematic Cartesian forward transformation is executed.

#TRANSFORM [ACS] [WCS] [50, 0, 10] [A10 B20]

Fig. 193: Example: forward transformation with #TRANSFORM

Example 2, basic configuration is a 5-axis kinematic with CS stack.

Transformation for the point (10,-10,15) with tracking axis positions A45 and B90 from the MCS system to the ACS system. A simple kinematic backward transformation is executed.

#TRANSFORM [MCS] [ACS] [10, -10, 15] [A45 B90]

Fig. 194: Example: backward transformation with #TRANSFORM

17.8 Orientation programming

Tool orientation can be programmed in a number of different ways. Basically, the representation of orientation depends on the basic kinematic (5-axis or robot kinematics) and on the settings of the CAD/CAM systems used.

In a conventional case, Cartesian coordinates are programmed in point/Euler angle representation for 5-axis machining, i.e. for a 5-axis machine with a CA tool head, i.e. the positions via X, Y and Z and machine angles for orientation via the configured axis identifiers C1 and A1.

Example:

X50 Y50 Z100 C1=45 A1=45

Since the contours of a workpiece are normally represented in vectors in CAD/CAM systems, it is also normal to generate NC programs in vector representation. This means that the contour description is independent of the machine or kinematic structure.

Tool orientation is defined by a vector which is aligned by the tool tip (TCP) in the direction of the tool clamp. Direction vector components are always programmed by A, B, C (or I, J, K). So, when vector programming is active, tracking axes configured with the same name cannot be programmed due to unambiguity reasons.

With 5-axis kinematics, vector components are defined by the "virtual" axis identifiers A, B and C.

Example:

X50 Y50 Z100 A-0.5 B0.5 C0.7071

Fig. 195: Orientation vector at 5-axis head

With 6-axis kinematics (e.g. robots), vector components are also defined by the "virtual" axis identifiers A, B and C or by the axis identifiers I, J and K which are also used in robotics.

BECKHOFF

Fig. 196: Orientation vector on robot

Example:

X-17.083 Y29.630 Z10 A-0.17083 B0.29630 C0.93969 or

X-17.083 Y29.630 Z10 I-0.17083 J0.29630 K0.93969

In order to correctly evaluate tool orientation, the appropriate orientation mode must be used and the corresponding transformation must be activated.

17.8.1 Programming and configuration of 5-axis kinematics (#ORI MODE)

Evaluation of point-vector programming after selecting the transformation is enabled by the NC command #ORI MODE[..]. In conventional point-vector representation, the VECTOR_2DOF (2 **D**egrees **O**f **F**reedom) mode is used. It remains valid until program end (M30) or until another programmed change occurs.

`Syntax of Point-vector representation`

#ORI MODE [VECTOR_2DOF]

VECTOR_2DOF	A, B and C are components of the direction vector. The address letters A, B, C must always be used; they have no reference to configured axis names in the channel list. The vector components need not be specified in standardised form.

#ORI MODE [VECTOR_2DOF] causes the preselection of orientation programming. Only when the transformation is active (#TRAFO ON) are point-vector representations detected and evaluated.

`Syntax of Switching over to conventional orientation programming`

#ORI MODE [ANGLE]

ANGLE	Angle values via configured axis names (default)

Special features relating to active coordinate systems (CS):

- With 5-axis RTCP transformations (incomplete) and an active CS, orientation is always represented independent of P-CHAN-00247.
- With complete 5-axis transformations and an active CS, orientation is represented dependent on P-CHAN-00247.
- Vector programming is also permitted with tube machining. Virtual axis programming may not be active (see [FCT-M5, Kin-ID 90 – HD14 = 0]).

Alternatively, point-vector programming can be preconfigured with P-CHAN-00177. With *ori.mode* the user defines whether the values programmed with A,B,C in the NC channel are read as normal coordinates or angle values with an active kinematic transformation or whether they are interpreted as corresponding vector components.

Alternatively, the following identifiers must be configured:

ori.mode   ANGLE   Angle values by configured axis names (default)

ori.mode   VECTOR_2DOF   Vector components by A, B, C

If *ori.mode* is unassigned, the default setting is active for orientation programming (orientation specified by rotation angle).

BECKHOFF

Switching over orientation programming to point-vector representation

```
%example_1
:
#KIN ID [9]
:
#ORI MODE [VECTOR_2DOF]
#TRAFO ON
G01 F1000
X79.993 Y57.197 Z-39.993 A0.67520 B0.29702 C-0.67520
X79.973 Y57.392 Z-39.973 A0.66945 B0.32198 C-0.66945
X79.941 Y57.586 Z-39.941 A0.66316 B0.34705 C-0.66316
:
X79.255 Y58.978 Z-39.255 A0.58988 B0.55144 C-0.58988
X79.121 Y59.121 Z-39.121 A0.57735 B0.57735 C-0.57735
X78.903 Y59.319 Z-38.903 A0.55691 B0.61620 C-0.55691
X78.666 Y59.493 Z-38.666 A0.53439 B0.65487 C-0.53439
X78.414 Y59.643 Z-38.414 A0.50964 B0.69321 C-0.50964
X75.000 Z-35.000 A0.00000 B1.00000 C0.00000
:
#TRAFO OFF
M30
```

Toggling between point-vector and point-angle representation

```
%example_2
:
#KIN ID [9]
:
#ORI MODE [VECTOR_2DOF]
#TRAFO ON
G01 F1000
X79.993 Y57.197 Z-39.993 A0.67520 B0.29702 C-0.67520
X79.973 Y57.392 Z-39.973 A0.66945 B0.32198 C-0.66945
X79.941 Y57.586 Z-39.941 A0.66316 B0.34705 C-0.66316
...
X79.255 Y58.978 Z-39.255 A0.58988 B0.55144 C-0.58988
X79.121 Y59.121 Z-39.121 A0.57735 B0.57735 C-0.57735
X78.903 Y59.319 Z-38.903 A0.55691 B0.61620 C-0.55691
X78.666 Y59.493 Z-38.666 A0.53439 B0.65487 C-0.53439
X78.414 Y59.643 Z-38.414 A0.50964 B0.69321 C-0.50964
X75.000 Z-35.000 A0.00000 B1.00000 C0.00000
:
#TRAFO OFF
:
#ORI MODE [ANGLE]
#TRAFO ON
G01 F1000
X10 Y10 Z10 C90 A15
X20 Y10 Z10 C90 A30
:
#TRAFO OFF
M30
```

17.8.2 Programming and configuration of 6-axis kinematics (robot) (#ORI MODE)

The evaluation of point-vector programming after selecting the transformation is activated by the NC command #ORI MODE[..]. In conventional point-vector representation, the VECTOR_ABC or VECTOR_IJK mode is used. It remains valid until program end (M30) or until another programmed change occurs. The behaviour of the fixed rotary axis is defined by 2 additional keywords.

`Syntax of Point-vector representation`

#ORI MODE [VECTOR_ABC | VECTOR_IJK FIXED_AX_IDX=.. **]**

or

#ORI MODE [VECTOR_ABC | VECTOR_IJK TOOL_AX_IN_PLANE=.. **]**

VECTOR_ABC	A, B and C are components of the direction vector. The address letters A, B, C must always be used; they have no reference to configured axis names in the channel list. Vector components need not be specified in standardised form.
VECTOR_IJK	I, J, K are components of the direction vector. The address letters I, J, K must always be used. Vector components need not be specified in standardised form. It is not permitted to use circular programming by I, J, K at the same time according to DIN 66025..

The 2 axes for the rotation angle to the tool orientation are obtained from the 3 vector components.. The angle setting of the third rotary angle is obtained from the joint angle settings at the time when the kinematic transformation is selected and remains unchanged during vector programming.

The axis index of the rotary axis not participating in orientation according to the Euler convention is obtained from considering the order of the axes that define the robot's position and hand orientation (see also description of P-CHAN-00178).

FIXED_AX_IDX=..	Axis index of fixed rotary axis.

Example: Rotary axis C angle setting 45° on selection, FIXED_AX_IDX = 5

Initial robot orientation   Orientation at target point

As an alternative to the fixed rotary axis, it is also possible to define the plane (YZ, ZX) containing either the Z or the Y tool axis. The third angle can then be determined so that the selected tool axis lies parallel to the defined plane at the target point (see also description of P-CHAN-00436).

TOOL_AX_IN_PLANE=. Plane parallel to a tool axis.

.

Example 1: Tool axis Z (red) parallel to the basic plane ZX, TOOL_AX_IN_PLANE = 1

Basic plane ZX: Initial orientation   Orientation at target point

Example 2: Tool axis Y (green) parallel to the basic plane YZ, TOOL_AX_IN_PLANE = 2

Basic plane YZ: Initial orientation   Orientation at target point

#ORI MODE [VECTOR_...] causes preselection of orientation programming. Only when the transformation is active (#TRAFO ON) are point-vector representations detected and evaluated

```
Syntax of Switching over to conventional orientation programming
```
#ORI MODE [ANGLE]

ANGLE   Angle values by configured axis names (default).

Special features relating to active coordinate systems (CS):

- With complete 6-axis transformations and an active CS, orientation is always represented dependent on P-CHAN-00247.

Alternatively, point-vector programming can be preconfigured with P-CHAN-00177. The *ori.mode* allows the user to define whether values programmed with A, B, C or I, J, K are read in the NC program as normal coordinates or angle values when the kinematic transformation is active or whether the values are interpreted as vector components.

Alternatively, the following identifiers must be configured:

ori.mode	ANGLE	Angle values by configured axis names (default)
ori.mode	VECTOR_ABC	Vector components by A, B, C
ori.mode	VECTOR_IJK	Vector components by I, J, K

If *ori.mode* is unassigned, the default setting is active for orientation programming (orientation specified by rotation angle).

The axis index of the fixed rotary axis is specified in P-CHAN-00178:

| *ori.fixed_axis_index* | *<idx>* | Axis index of fixed rotary axis |

The plane parallel to the tool axis is specified in the channel parameter P-CHAN-00436:

| *ori.tool_ax_in_plane* | *<id>* | Plane parallel to tool axis plane |

Specifications relating to the fixed rotary axis P-CHAN-00178 and the tool axis plane P-CHAN-00436 are mutually exclusive. If the two parameters are assigned, error ID 22027 is output when the controller starts up and the two values are corrected to zero.

Switch over orientation programming to point-vector representation (ABC) and specify fixed rotary axis

```
%example_1
; …
#KIN ID [45]
; …
#ORI MODE [VECTOR_ABC FIXED_AX_IDX=5]
#TRAFO ON
X50        Y50                    A50       B0 C0
X75        Y150       Z180        A0        B0 C1
X149.316   Y150       Z180        A-0.0457 B0 C0.999
X149.316   Y150       Z165.012   A-0.0457 B0 C0.999
X150.0018 Y150        Z150.0279 A-0.0457 B0 C0.999
X162.1716 Y150        Z150.0621 A0.0349  B0 C0.9994
X172.1268 Y149.9997 Z149.3631 A0.1013  B0 C0.9949
X178.7241 Y149.9997 Z148.5459 A0.1454  B0 C0.9894
X188.532   Y149.9997 Z146.7645 A0.2111  B0 C0.9775
X198.2064 Y149.9997 Z144.3474 A0.2758  B0 C0.9612
X207.7002 Y149.9994 Z141.2733 A0.3393  B0 C0.9407
X216.978   Y149.9994 Z137.5713 A0.4012  B0 C0.916
; …
X150       Y150       Z180        A0.6111  B0.0014 C0.7916
X150       Y150       Z180        A0.0631  B0.0001 C0.998
X150       Y150       Z180        A0        B0      C1
; …
#TRAFO OFF
M30
```

Switch over orientation programming to point-vector representation (IJK) and specify the plane parallel to the tool axis

```
%example_2
;…
#KIN ID [45]
;…
#ORI MODE [VECTOR_IJK TOOL_AX_IN_PLANE=1]
#TRAFO ON
X75      Y150    Z180     I0      J0      K1
X10.874  Y0      Z-29.875 I-.099  J0      K.995
X10.846  Y.666   Z-29.872 I-.099  J-.006  K.995
X10.667  Y1.976  Z-29.854 I-.097  J-.018  K.995
X10.464  Y2.748  Z-29.792 I-.095  J-.025  K.995
X10.208  Y3.429  Z-29.668 I-.093  J-.031  K.995
X9.879   Y4.075  Z-29.46  I-.091  J-.037  K.995
X9.517   Y4.713  Z-29.296 I-.088  J-.043  K.995
X9.126   Y5.328  Z-29.166 I-.085  J-.049  K.995
X8.285   Y6.492  Z-29.086 I-.077  J-.06   K.995
X7.387   Y7.597  Z-29.317 I-.068  J-.07   K.995
X6.9     Y8.108  Z-29.472 I-.063  J-.075  K.995
X6.385   Y8.598  Z-29.664 I-.058  J-.079  K.995
X5.825   Y9.038  Z-29.8   I-.053  J-.082  K.995
X5.218   Y9.412  Z-29.841 I-.047  J-.086  K.995
X3.924   Y10.011 Z-29.852 I-.035  J-.091  K.995
X2.56    Y10.43  Z-29.849 I-.023  J-.095  K.995
X1.182   Y10.657 Z-29.835 I-.01   J-.097  K.995
X.461    Y10.682 Z-29.769 I-.004  J-.098  K.995
X-.257   Y10.636 Z-29.639 I.002   J-.098  K.995
X-1.027  Y10.509 Z-29.435 I.009   J-.097  K.995
X-1.696  Y10.366 Z-29.297 I.015   J-.096  K.995
X-3.083  Y9.956  Z-29.106 I.028   J-.093  K.995
X-4.428  Y9.482  Z-29.258 I.041   J-.088  K.995
X-5.462  Y9.007  Z-29.478 I.05    J-.083  K.995
X-6.068  Y8.681  Z-29.657 I.055   J-.08   K.995
X-6.642  Y8.299  Z-29.782 I.061   J-.076  K.995
X-7.696  Y7.337  Z-29.826 I.07    J-.067  K.995
X-8.601  Y6.233  Z-29.831 I.078   J-.057  K.995
;…
#TRAFO OFF
M30
```

17.9 Status & Turn (IS, IT)

An option exists to specify the robot pose for the corresponding Cartesian position as an alternative to axis-specific positioning and to obtain a more precise specified position of a #PTP movement for industrial robots.

The robot pose is described using 2 additional key figures (Status & Turn).

i Robot positioning with Status & Turn is currently only available for kinematic type 45.

Status bit

The robot pose is divided into 3 criteria. If a criterion applies, a corresponding numerical value is added to the status.

1st criterion If the wrist is behind axis A1, decimal 1 or binary 1 is added (yellow area in the figure on the left).

Bit 0 = 1   Bit 0 = 0

A1

Fig. 197: The intersection of the hand axes (arrowhead) is in the (blue) base area.

2nd criterion If the wrist is in front of the straight line through axes A2 and A3, decimal 2 or binary 10 is added (centre and right image).

Fig. 198: Status bit 1 for robots with an offset between axis A3 and axis A5

3rd criterion (legacy value, not normally used any more): Indicates the position of axis A5. If A5 > 0, decimal 4 or binary 100 is added.

Fig. 199: Status bit 2 for axis angle position A4=0° and A4=180°.

Turn bit

The turn value lists the negative signs of the axis angles.

When the turn value is considered in a binary representation, the sign of each axis angle is assigned to a bit. These are then added to a number, i.e. the turn.

If the axis angle of an axis is < 0°, the value is 1.

	A6 < 0°	A5 < 0°	A4 < 0°	A3 < 0°	A2 < 0°	A1 < 0°
Binary	100000	010000	001000	000100	000010	000001
Decimal	32	16	8	4	2	1

If all 6 axis angles are in the negative range, this results in a turn value of decimal 63 or binary 111111; accordingly, this results in decimal 0 and binary 000000 for 6 positive axis angles.

Description

The additional parameters Status & Turn are available for unambiguous programming of the robot poses (Kin_Typ_45) with Cartesian target coordinates of a PTP movement.

```
Syntax pf Programming Status & Turn with the prefixes "IS" and "IT":
```
#PTP ON

G.. X.. Y.. … **IS**.. **IT**..

#PTP OFF

Binary numbers can be programmed with the following syntax:

'B<0…1>', or '2#<0…1>', or '02#<0…1>'.

When binary numbers are used, the syntax is as follows:

Status: **IS**'Bxxx'

Turn: **IT**'Bxxxxxx'

When decimal numbers are used, the syntax is as follows:

Status: **IS**<expr>

Turn: **IT**<expr>

Display values

The following CNC objects are provided for this function:

- mc_st_valid_r: Validity of the status & turn value
  (Task COM index group 0x12010<C_{ID}> Index offset 0xB1)
- mc_st_status_r: Status value of kinematic 45
  (Task COM index group 0x12010<C_{ID}> Index offset 0xB2)
- mc_st_turn_r: Turn value of kinematic 45
  (Task COM index group 0x12010<C_{ID}> Index offset 0xB3)

If Status & Turn are not programmed, the target point is determined on the axis angle level using the shortest way strategy.

Status & Turn programmed with binary numbers

```
N010 #PTP ON
N020 G01 X1100 Y0 Z1400 A0 B90 C0 IS'B010' IT'B000010' F5000
N030 G01 X1200 ;target point is determined by shortest way
N040 #PTP OFF
```

Status & Turn programmed with decimal numbers

```
N010 #PTP ON
N020 G01 X1000 Y0 Z1400 A0 B90 C0 IS2 IT2 F5000
N030 #PTP OFF
N040 G01 X1500
N050 G01 Y1000
N060 G01 X-1000
N070 #PTP ON
N080 G01 X-1000 Y-1000 ;target point is determined by shortest way
N090 #PTP OFF
```

18   Programming modulo axes

The default mode for modulo programming supports the specific definition of the direction of rotation and position by programming 2 signs and limiting to maximum one revolution in absolute dimensions.

```
Syntax:
```

<axis_name> [**+** | **-**] *<pos>*

<axis_name>	Designation of modulo axis. Long axis designations are not supported (e.g. "C_MODULO").
+ | -	The 1st sign after the axis name <u>always</u> determines the direction of rotation:
	- means rotation clockwise (cc)
	+ means rotation counter-clockwise (ccw)
	no sign means rotation in the direction of the shortest way (optimised alignment)
<pos>	Axis position in [°]. The position value can be combined with an additional sign to specify an absolute dimension. The assignment of sign and position value can be forced to execute the shortest way by brackets [..].

NOTICE

The programming of 2 signs (direction and position) is only permitted if the axis has the "Modulo" axis mode P-AXIS-00015). Positioning is always executed on the shortest way if no sign is programmed directly after the axis name.

In addition, there is also an option to change to a mode that always positions on the shortest path (section <u>Positioning on the shortest way [▶ 745]</u>). In this mode, programming 2 signs is also permitted. However, evaluation is based on the following rule:

    -- => + (minus minus is plus)

    +- => - (plus minus is minus)

    -+ => - (minus plus is minus)

    ++ => + (plus plus is plus)

Programming in absolute dimensions (G90):

- The value assigned to the axes (target point) is shifted to the modulo range. Therefore, a maximum of one revolution can be moved.
- The value may be a numerical expression such as [3*2+5] , P1 , [P1+P2-3] , [-30].
- The first sign of the value after the axis name always defines the direction of rotation. Every further sign is evaluated as a part of the (absolute) position definition.

Example (assuming: 360° modulo)

```
G90 G1 C+560 * G90 G1 C+200 (Move to position 200 in + direction)
G90 G1 C-P1 (Go to position P1 (with implicit modulo) in – direction)
```

- If programmed position = current position, no motion.
- The motion path of a modulo axis is not limited by software limits.

Programming in incremental dimensions (G91)

- The value assigned to the axis indicates the amplitude of rotation of the axis with reference to the previous position. The first sign of the value after the axis name always defines the direction of rotation. Additional signs are not permitted in incremental programming.

Example (assuming: 360° modulo)

```
G91 G1 C+560 (Movement to "current position plus 560" in + direction)
```

- If the value is greater than the modulo value, the number of revolutions is taken into account. Therefore, a motion of more than one revolution is permitted.

The following V.A. variables permit read access to the current axis-specific modulo settings

V.A.MODE[i]	supplies the axis mode according to the axis table, e.g. 4 if is axis ia of the modulo type. -> is used to read modulo axes
V.A.MODULO_VALUE[i]	os used to read the modulo range e.g. 360 with a modulo range of 0-360°. (if the axis is a "non modulo axis", this value is not relevant).

Programming examples of modulo programming in absolute dimensions

G90 G1 C+350 <=> Go to position 350 in + direction

P1 =+10

G90 G1 C+P1 <=> G1 C+10 <=> Go to position 10 in +direction

P1= -350

G90 G1 C-P1 <=> G1 C-[-350] <=> G1 C-[10] <=> Go to position 10 in - direction

G90 G1 C+450 <=> G1 C+[450 mod 360] <=> Go to position 90 in + direction

Examples of correct programming:

```
C+200 Rotate in positive direction to position 200
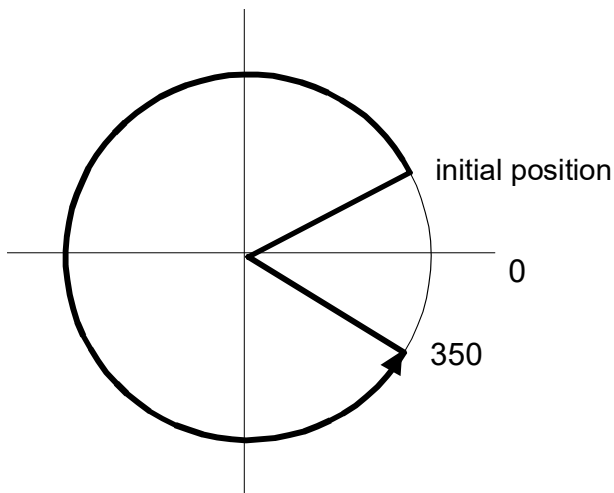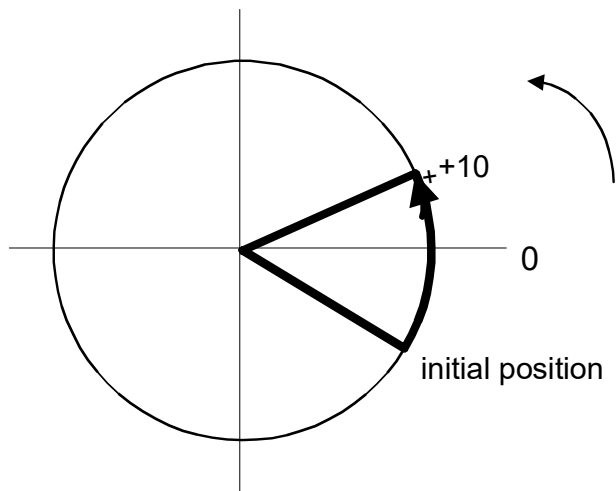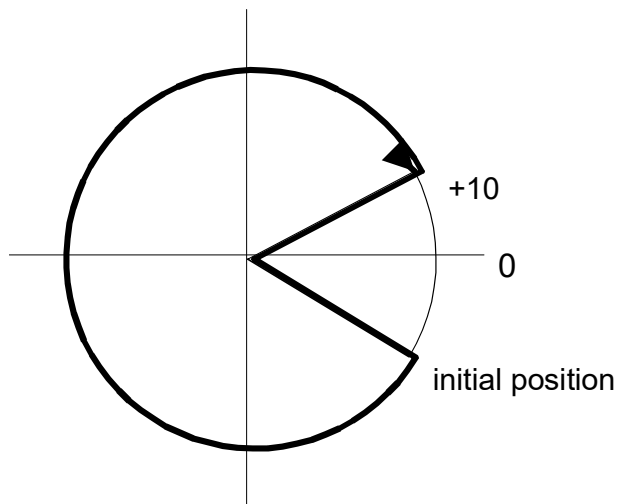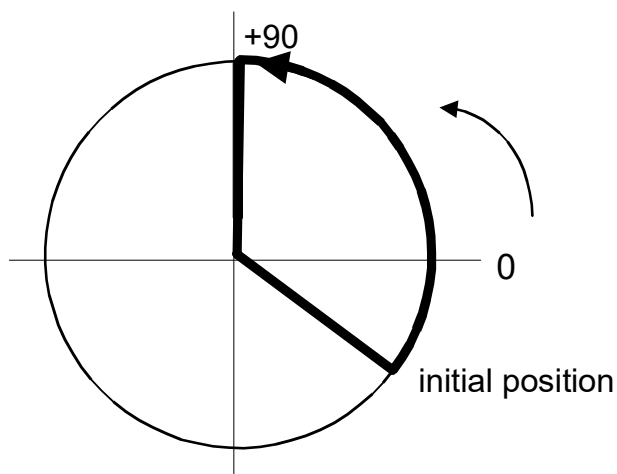
C-200 Rotate in negative direction to position 200

C+-200  Rotate in positive direction to position -200 (= +160)

C+[-200]  Rotate in positive direction to position -200 (= +160)

C—200 Rotate in negative direction to position -200 (= +160)

C-[-200]  Rotate in negative direction to position -200 (= +160)

C200 Rotate on shortest way to position 200

C[+200] Rotate on shortest way to position 200

C[-200] Rotate on shortest way to position -200
```

Examples of incorrect programming:

None because the first sign after the axis name determines the direction of rotation and every additional sign belongs to the position expression.

Programming examples of modulo programming in relative dimensions

G91 G1 C+30

final position

+30 °

0

initial position

G91 G1 C-30

-360

0

-30 °

initial position

final position

Examples of correct programming:

```
C+200 Rotate in positive direction to "current position plus 200"
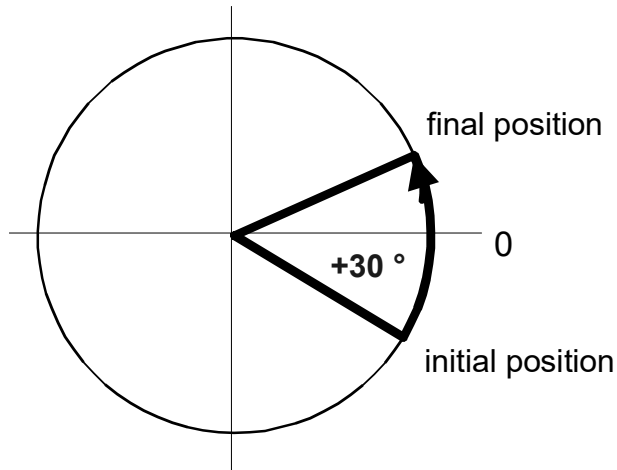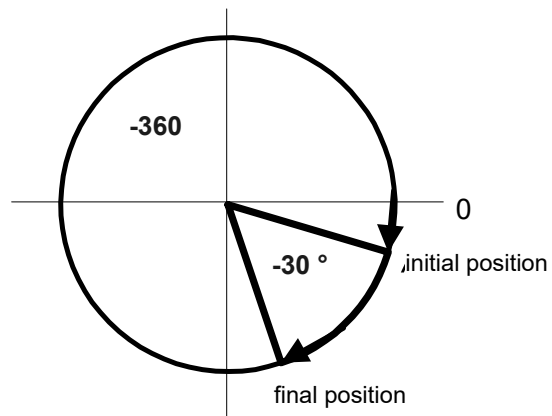
C-200 Rotate in negative direction to "current position minus 200"

C200 Rotate in positive direction to "current position plus 200"
```

Examples of incorrect programming:

```
C+-200    Error: Negative motion path during rel. programming not allowed.

C—200     Error: Negative motion path during rel. programming not allowed.

P1=-1
C-[P1]    Error: Negative motion path during rel. programming not allowed.
```

```
C+-200    Error: Negative motion path during rel. programming not allowed.
```

18.1 Positioning on the shortest way

This mode always permits the positioning of modulo rotary axes on the shortest way. This obviates the need for programming signs.

This mode is activated by the parameter P-CHAN-00346. However, if a pre-definition of rotation direction is desired, this requires relative programming in this mode.

Programming examples of modulo programming in absolute dimensions

Assuming: Modulo range 0-360°, G90, current axis position 0°

```
C200 Rotate on shortest way (-160) to position 200

C+200  Rotate on shortest way (-160) to position 200

C-200 Rotate on shortest way (+160) to position -200 (= +160)
```

Programming examples of modulo programming in relative dimensions

Assuming: Modulo range 0-360°, G91, current axis position +60°

```
C+200  Rotate in positive direction to position 260 (60+200)

C-200  Rotate in negative direction to position 220 (60-200)
```

19 Extended tool programming

19.1 Description of function

19.1.1 Tool ID

Within the scope of extended tool programming, the CNC provides tool management tasks (WZM) with new tool-specific communication objects. This refers to the enhanced tool (WZ) descriptions and tool service life variables.

In the present standard programming, single-element numbers are used to identify tools in the NC program. According to DIN 66025, this numerical ID is programmed together with the D word that is used to include new data (computational tool change). In conjunction with the T language command, the ID defines the next tool that is to be physically changed.

To include new data, this data must be requested from the present external tool management. The tool management function has special manufacturer-specific algorithms that are used to determine the tool that must be changed based on tool identification. It must be taken into account that the transferred tool ID only defines the tool type and that the tool magazine may contain several tools of the same type (alternate tools) that are ready to be used. Therefore, a specific tool cannot be clearly identified in the NC program.

In the first instance, the T number is used as technological information. This means that it reaches the PLC over the NC channel. According to DIN 66025, M06 is used to trigger the physical insertion of the new tool into the working spindle. By specifying the two commands "T with tool number" and M06 separately, preparatory measures can be taken (in the tool magazine, for example) after the T command and before M06 actually inserts the tool into the working spindle.

In the extended tool management function, tool data is identified by a three-digit tool number. Tool ID number:

Tool ID = base tool number + alternate tool number + modification number

The base tool number describes the tool type and the alternate tool number describes a tool unit of this type. The modification number has a purely data-related significance. It permits the use of various data records or a tool.

19.1.2 Tool life data recording

When tool life values are recorded, the contact time (service life) and the distance covered by the tool during contact (service distance) must be calculated. These channel parameters are required for the configuration of the tool life data recording function.

Only travel by motion blocks is considered in the basic setting. Positioning by rapid traversing has no impact on the tool life quantities.

The interpolator displays the data (tool ID, contact time, contact distance) after the tool is replaced. The trigger point for activating tool life data recording can be linked with by P-CHAN-00482 using the T or the D word.

The service life is recorded in ms; the service distance is recorded in mm.

Tool life data recording can be adapted to tool use by means of weighting factors [▶ 751].

19.2 Using tool ID (V.TOOL.) (#TOOL DATA, #TOOL PREP)

The tool ID is programmed with plain text commands. A request for new tool data with #TOOL DATA corresponds to the D command; the preparatory technology command for physical tool change with #TOOL PREP corresponds to the T command.

```
Syntax:
```
#TOOL DATA [<*basic*> [**,** <*sister*> [**,** <*variant*>]] **]**   Request new tool data
#TOOL PREP [<*basic*> [**,** <*sister*> [**,** <*variant*>]] **]**   Announce a tool change

The number of parameters is fixed [6] [▶ 786]-9.18. Expedient values are between 1 and 3. If two parameters are expected, they are the base tool number and the sister tool number.

The base tool number must be specified in all cases (**basic**). In addition, if the number of parameters is assigned the value 3, **sister** can be programmed for the sister tool number and optionally **variant** for the variant. If sister or variant are not programmed (a comma follows a comma or a closing bracket ')' follows a comma), 0 is inserted instead.

Programming commands and variables with 'sister' and 'variant'

```
#TOOL DATA [ P10, "SISTER", 0 ] <=> #TOOL DATA [ P10, "SISTER", ]
and
#TOOL PREP [P10, 0, "VARIANT"]  < = > #TOOL PREP [P10, , "VARIANT"]
```

The mathematical expressions used to specify the D and T commands are to be interpreted as a base tool number. Therefore, the tool management function retains the same degree of freedom as before to select the tool data block.

```
Syntax:
```
T<*basic*> or **D**<*basic*>

Decoder variables are used to implement access to the elements of tool identification. The current software version already provides the option to get the number of the last tool requested from the external tool management function via V.G.T_AKT. For compatibility reasons, this variable is retained. It always represents the base tool number in parallel to the new syntax introduced below.

```
Syntax:
```
V.TOOL.BASIC   Read access to the last base tool number programmed
V.TOOL.SISTER   Read access to the last sister tool number programmed
V.TOOL.VARIANT   Read access to the last variant number programmed

Programming commands and variables with 'sister'

```
#TOOL DATA [ P10, "SISTER", 3 ]
.....
#TOOL PREP [V.TOOL.BASIC, V.TOOL.SISTER, V.TOOL.VARIANT]
```

```
#TOOL DATA [ P10, "SISTER", 3 ]
.....
#TOOL PREP [V.TOOL.BASIC, V.TOOL.SISTER, V.TOOL.VARIANT]
```

19.3 Refreshing tool data (#TOOL REFRESH)

Following command is provided for the repeated inclusion of tool data of the active tool, e.g. in additional axes after an axis exchange:

Syntax:

#TOOL REFRESH   Refresh data of active tool

The command results in the immediate inclusion of the already available data of the active tool. No more tool data is adopted from an internal tool list [5] [▶ 786] or requested from an external tool management function. The D word reprogramming function can be deselected.

19.4 Reading/removing tool life values (#TOOL LIFE READ/ REMOVE)

The following commands are provided for access to tool life data before the next tool change. For example, they are required to determine tool life data for tools in slave axes in synchronous mode since this data is not acquired, as described in section Weighting factors for tool life and tool life distance ((V.TLM) [▶ 751]. The following command reads the tool life data of the active tool in the current channel and outputs this data to the tool management function for any tool (e.g. tool ID of a slave axis tool).

Syntax:

#TOOL LIFE READ [<*basic*> [**,** <*sister*> [**,** <*variant*>]] **]**   Read current tool life data and assign a tool ID in the tool management system

The same rules apply to programming tool ID as for #TOOL PREP (section Programming commands and variables [▶ 747]), for example. According to the HÜEMNOS convention, a tool ID may include 1, 2 or 3 tool parameters. It is mandatory to specify at least one tool parameter that is always interpreted as the base tool number (basic). Tool parameters may be any mathematical expression; e.g. V.G.T_AKT, V.TOOL.BASIC, Pxx, V.L.xxx etc.

After sending the current tool life data to the tool management system, the internal tool life acquisition is not reset to 0. Instead, the tool life data is aggregated.

The following command resets to 0 the previously acquired tool life data of the tool currently active in the current channel without sending the data to the tool management system.

Syntax:

#TOOL LIFE REMOVE   Delete current tool life data

Reading and deleting tool life data

```
....
:
....   Normal mode
:
#TOOL LIFE READ [V.G.T_AKT]    (Read tool life data of current tool)
#TOOL LIFE REMOVE              (Reset tool life data for separate)
                              (acquisition of tool life data during)
                              (synchronous operation)
Select synchronous mode
:
....   Synchronous mode
:
Deselect synchronous mode
#TOOL LIFE READ [T10]          (Assuming: T10 is also moved in synchronous)
                              (mode: Send tool life data)
                              (to tool management for T10)
:
....   Normal mode
:
```

19.5 Weighting factors for tool life and tool life distance ((V.TLM)

The NC program can change the weighting the tool life values. Changeable factors ares used to adapt tool life acquisition to tool use.

In the case of every tool change initiated by the T or D command (see P-CHAN-00482), the complete tool ID, the tool life time and the tool life distance are sent to the tool management system. All parameters are then zeroed and tool life acquisition is enabled for the new tool substituted.

The following two decoder variables are used to program the weighting factors of service life and service distance (access is not synchronous with real time):

```
Syntax:
```

| **V.TLM.TIME_FACT** | Weighting factor of tool life time ≥ 0.0 |
| **V.TLM.DIST_FACT** | Weighting factor of tool life distance ≥ 0.0 |

The variables can be read and written. Both factors are 1.0 (100% at program start). They can also be programmed with values > 1.0 if this is necessary. Both variables may be written in an NC block.

Iif a tool is always in contact, it should be weighted with a factor of 1.0 (100%). However, if material is only removed along half the motion path, a weighting factor of 0.5 can be included in the calculation. The default value of the weighting factors of service time and service distance is 1.0.

Acquisition conditions:

- Rapid traverse blocks are not included in tool life acquisition.
- Tool life acquisition is stopped at a feedrate of zero.
- With the exception of rapid traverse interpolation, all motion types are included in tool life acquisition. For example, G01, G02, G03, spline interpolation and G63 are included.
- The weighting factors are included in the calculation.
- A distinction is not made between main and tracking axes for axes participating in the motion. The path feedrate is always used to add up the distance. In the case of tracking axes programmed on their own in the block, the path covered by the tracking axis is added to the service distance. If this is not desired, the programmer can correct it by specifying the weighting factors V.TLM.TIME_FACT/DIST_FACT = 0.0.
- Active master/slave arrangements are not taken into account.
- In the case of a reset or program abort, the last current values are also saved to the database of the tool management system.
- If only substitution occurs, i.e. no tool was previously in the work spindle and the current T number is zero, no data is sent.
- Tool data is only sent if a tool management system is actually present (P-CHAN-00016).

19.6 Setting tool life parameters (#TOOL LIFE DEF)

The (relative) tool life parameters of time and distance of the current tool are acquired as soon as the tool executes movements in the NC channel that are relevant for calculating tool life parameters (e,g, G01, G02, G03). The calculation normally starts here at zero. The incremental values acquired are then transferred to an external tool management system at the next tool change. The absolute values for tool life distance and tool life time are then obtains by aggregation and weighting.

In certain situations, it may be necessary to start the incremental calculation of tool life parameters not at zero but with specific values for a tool in the NC channel. The following NC command is therefore provided to initialise the tool life parameters of distance and time for the current tool in the NC channel:

```
Syntax:
```

#TOOL LIFE DEF [DIST=.. **TIME**=..**]**	Set tool life parameters of the active tool
DIST=..	Tool life distance in [mm]
TIME=..	Tool life time in [s]

20 Positioning axes

A complete list of axis-specific additional functions is contained in the overview of commands in the Appendix under Additional axis-specific functions (<X>[..]) [▶ 800].

Positioning axes are translatory or rotary axes which can be interpolated in the same NC channel independent of the path axis compound. Each positioning axis has its own axis interpolator and can be commanded at its own feedrate. Rotary positioning axes in modulo mode always move on shortest way.

Restrictions:

An axis cannot be programmed as a positioning axis if:

- This axis is currently moving in synchronous mode
- A kinematic or Cartesian transformation is active and certain conditions are not fulfilled (see Section Cartesian / kinematic transformation and positioning axes [▶ 762]).
- Block search is active
- Simulation modes (Online-Simulation, Contour visualisation, Machining time calculation) are active
- Spline interpolation is active
- Polynomial contouring is active
- Turning functions are active

20.1 Independent axes (INDP_SYN, INDP_ASYN) (#WAIT INDP, #WAIT INDP ALL)

Two different operation modes are provided to program independent axes:

- Command value based synchronisation of path axes and independent axes at block end.
- Command value based synchronisation of path axes and independent axes over several blocks.

Fig. 200: Motion diagram of path axis compound/independent axes

> **i** No offsets are considered for independent axes.

Additive manual mode (G201/G202) with an independent axis is possible.

```
Programming syntax for independent axes:
```

<axis_name> **[INDP_SYN** | **INDP_ASYN G90** | **G91 G00** | [**G01** | **G100 FEED**=.. |

TIME=.. | **FEED_MAX_WEIGHT**=..] POS=.. [**SLOPE_TYPE**=*<ident>*]

{**M..**} {**H..**} [**DRY_RUN**] [**ACC_WEIGHT**=..] [**RAPID_ACC_WEIGHT**=..] { **** } **]**

<axis_name>	Name of independent axis
INDP_SYN	Identifier for synchronous (blockwise) independent axis motion. The transition to the next block is only executed if all axes have reached their target point. Must always be programmed as the <u>first</u> keyword.
INDP_ASYN	Identifier for asynchronous (cross-block) independent axis motion. There is no synchronisation at target points. Command value based synchronisation is executed by a special NC command (#WAIT INDP) or by programming the independent axis as a conventional path axis. Must always be programmed as the <u>first</u> keyword.
G90 / G91	Absolute/relative dimension
G00 / G01	Rapid traverse/linear interpolation
FEED=..	Axis-specific feedrate in [mm/min, m/min, inch/min]
TIME=..	Axis-specific motion time in [s]
FEED_MAX_WEIGHT=..	Weighting factor in [%] referred to axis-specific maximum feed P-AXIS-00212. Only weighting factors less than 100% are permitted. (according to G194, Section <u>Machining time/feedrate (G93/G94/G95/G194) [▶ 149]</u>)
POS=..	Axis position in [mm, inch]
SLOPE_TYPE=*<ident>*	Slope profile types according to #SLOPE [TYPE=...], Section <u>Independent axes [▶ 357]</u>). If no slope type is programmed, the slope type taken from the channel parameter list P-CHAN-00071 is set by default.
Old syntax: *SLOPE_PROFIL=..*	*Slope profile types (0, 1, 2, 3). If no slope type is programmed, the slope type taken from the channel parameter list P-CHAN-00071 is sued by default.*

G100 **[as of Build V2.11.2801.05]**	If measurement types 1, 2 or 7 (Section <u>Measuring functions [▶ 87]</u>) are used, a measurement run can be executed with independent axes. The measuring point is latched for each axis involved. An independent measurement run is also possible in parallel to a path motion of a G100 measurement run. For more information see [FCT-C4//Measurement].
DRY_RUN	Dry run of axis motion. The motion is only executed in the NC channel but the axis is not really moved. This offsets the axis coordinate within the channel relative to the physical axis. This offset is automatically cancelled at every program start or by an explicitly programmed #CHANNEL INIT [CMDPOS] (see Programming example 3).
M..	Axis-specific M functions (*)
H..	Axis-specific M/H-functions (*)
ACC_WEIGHT=.. **[as of V3.1.3079.06]**	Axis-specific weighting factor in [%] referred to acceleration with G01, G02, G03 (see section <u>Acceleration weighting (G130/G131/G231/G333/G334) [▶ 144]</u>)
RAPID_ACC_WEIGHT=.. **[as of V3.1.3079.06]**	Axis-specific weighting factor in [%] referred to acceleration with G00 (see section <u>Acceleration weighting (G130/G131/G231/G333/G334) [▶ 144]</u>)
\	Separator ("backslash") for clear programming of the command over multiple lines.

(*) only possible with synchronisation modes MOS, MVS_SVS, MVS_SNS, MNS_SNS.

Axis-specific M/H functions can also be output to an independent axis without programming a motion. This only requires the identifier INDP_SYN or INDP_ASYN.

```
Syntax:
```
<axis_name> **[INDP_SYN | INDP_ASYN M..** { **M**..} **H**.. {**H**..} { \ } **]**

<axis_name>	Name of independent axis
INDP_SYN/ INDP_ASYN	Identifier for an independent axis
M..	Axis-specific M function
H..	Axis-specific H functions
\	Separator ("backslash") for clear programming of the command over multiple lines.

Command value based synchronisation of specific asynchronous axis motions (INDP_ASYN) can be forced by the NC command:

`Syntax:`

#WAIT INDP [*<axis_name>* { ,*<axis_name>* } **]**

<axis_name>   Name of asynchronous axis

> **i** If an asynchronous axis is again programmed with a motion before or without a corresponding #WAIT INDP [ ], command value based synchronisation is implicitly executed in the interpolator.

Command value based synchronisation of all currently active asynchronous axis motions (INDP_ASYN) is forced by the NC command:

`Syntax:`

#WAIT INDP ALL

> **i** If the axis of a pre-assigned axis-specific M/H function (P-CHAN-00039, P-CHAN-00025) is programmed in the same NC block as an independent axis, an error message is output.
>
> **Example:** M10 is pre-assigned for a specific X axis (m_default_outp_ax_name[10] x):
>
> N10   **M10**   **X** [INDP_SYN G01 G90 POS10 FEED1000 M7]
>
> |_____| **<- Error!**

Independent axes

Example 1:
```
;N10 ends when X,Y and the indep. sync. Z axis
;end their motions
N10 X10 Y11 Z[INDP_SYN POS50 G01 FEED100 G90]

;N20 is executed after all motions in N10 end
N20 X20

;N30 ends when X and Y end their motions;
;independent asynchronous axis continues its motion
N30 X5 Y10 Z[INDP_ASYN POS500 G01 FEED200 G90]

;N40 is interpolated, asynchronous independent Z axis continues its motion
N40 X20 Y30

;Forced synchronisation of the Z axis: wait until target position
;Z500 is reached from N30
N50 #WAIT INDP[Z]

;Interpolation in N60 with X, Y, Z in coordinated motion
;starts after synchronisation takes place in N50
N60 X30 Y40 Z60
N70 Z[INDP_SYN M50]    ;Output of M50 via independent Z axis
N80 …
```

Example 2:
```
;N10 is interpolated, the independent asynchronous Z axis
;continues its motion
N10 X10 Y11 Z[INDP_ASYN POS500 G01 FEED200 G90]

;N20 is interpolated, the independent asynchronous Z axis
;continues its motion
N20 X20 Y22

;Implicit synchronisation of Z motion of N10 before motion
;Z550 starts
N30 Z550

N40 X20 Y30 Z60  ;N40 is interpolated
N50 …
```

Example 3:
```
%dry_run
N100 X1 Y2 Z3                              ;IPO=3, LR=3, offset=0

N200 G01 X10 F100 Z[INDP_SYN POS=4 G01 G90 \
                  FEED=120 DRY_RUN]        ;IPO=4, LR=3, offset=1
N300 Y20 F1000
N350 Z[INDP_SYN POS=7 G00 G90]            ;IPO=7, LR=6, offset=1
N360 Z[INDP_SYN POS=4 G01 G91 \
      FEED=100 DRY_RUN]                    ;IPO=11, LR=6, offset=5

;Remove DRY_RUN offset
N001 #TIME 2
N111 #CHANNEL INIT[CMDPOS]                 ;IPO=6, LR=6, offset=0
N222 #TIME 2
N400 Y10 Z5
M30
```

20.2 Oscillating axes (OSC)

i The availability of this function depends on the configuration and on the version scope.

An oscillating axis motion is required in certain machining technologies, e.g. grinding, and this is executed mainly independently of a path motion.

This motion referred to below as an "oscillating motion" is executed by the tool with periodic reversal across the workpiece.

An oscillating axis in grinding is presented below as an example. The workpiece is machined by superimposing the oscillating X motion on positioning motions in the Y and Z axes.

Fig. 201: Grinding with an oscillating axis

The essential characteristics of the oscillating motion result from the oscillating motion between two absolute positions as well as the feedrate

The oscillating motion start and stop and its parameters are defined in the NC program

Any axis can be defined as an oscillating axis within the scope of the configured axes. The oscillating motion is asynchronous to the path motion.

The oscillating motion is deactivated either:

- directly by an NC command
- or implicitly by programming a path motion for the oscillating axis
- or implicitly by requesting axis positions to synchronise decoding and interpolation
- or implicitly at the end of the NC program

The type of velocity profile can be defined in the dynamic phase by specifying the slope type in the channel parameters P-CHAN-00071 (linear/non-linear slope) for the oscillating motion.

The programming syntax is based on the axis-specific programming of independent axes. After the axis identifier, the parameters of the oscillating motion are defined via keywords and, if applicable, an associated value:

```
Syntax for programming an oscillation motion:
```
<axis:name> **[OSC ON** | [**OFF | OFF FEED=.. | OFF INSTANT**]
          FEED=.. | FREQ=.. | TIME=.. [1ST_POS=.. 2ND_POS=..]
              | [ZERO_POS=.. EXCUR=..] [1ST_DELT=.. 2ND_DELT=..] [NBR_OSC=..] { \ }]

<axis_name>	Name of the oscillating axis
OSC	Identifier for "Oscillating" function. Must always be programmed as the <u>first</u> keyword.
ON	Oscillation on. The motion is stopped at block end when a path motion is active and the oscillating motion is then commanded.
OFF	Oscillation off. Current oscillation cycle is finished. The oscillating axis can then move again in the coordinated motion. If the oscillating motion is stopped implicitly if it is not previously deselected when a new axis motion is programmed.
OFF FEED=..	Fast oscillation stop. The current oscillation cycle is aborted and the axis moves at the specified feedrate to target position (2ND POS). The oscillating axis can then move again in the coordinated motion.
OFF INSTANT	Immediate oscillation stop. Axis stops immediately and can be moved again within the path compound immediately. Available as of V3.1.3107.38
FEED=..	Feedrate of the oscillating motion in [mm/min, m/min, inch/min]
FREQ=..	Frequency of the oscillating motion in [Hz]
TIME=..	Period of the oscillating motion in [s]
1ST_POS=..	First reversal position in [mm, inch]
2ND_POS=..	Second reversal position in [mm, inch]
ZERO_POS=..	Zero point or zero crossing of the oscillating motion in [mm, inch]
EXCUR=..	Excursion in [mm, inch]
1ST_DELT=..	Wait time at first reversal position in [s]
2ND_DELT**=..**	Wait time at second reversal position in [s]
NBR_OSC=..	Number of oscillations
\	Separator ("backslash") for clear programming of the command over multiple lines.

The characteristic of the oscillating motion is determined by the locations of the reversal positions and the axis feedrate. The reversal positions can be either specified directly or alternatively they are determined automatically via the zero position and the excursion.

Oscillating positions are always **absolute positions**.

After an oscillating motion is deselected, the tool always stops at oscillating position 2.

Alternatively, the oscillating velocity can be determined by feedrate, frequency or period.

If no restriction occurs due to the dynamic axis characteristics, the frequency and the period are maintained exactly when the linear slope is used and maintained approximately when the non-linear slope is used.

Fig. 202: Positioning procedure with pendulum movement

Specifying the oscillating motion path via reversal positions
N10 X[OSC ON 1ST_POS=-100 2ND_POS=100 FEED=1000]

Specify the oscillation travel distance via the zero position and the excursion,
10 oscillations
N20 X[OSC ON ZERO_POS=0 EXCUR=100 FEED=1000 NBR_OSC=10]

Specify 1 Hertz oscillation frequency
N30 X[OSC ON ZERO_POS=0 EXCUR=100 FREQ=1]

Specify a 4s oscillating period
N40 X[OSC ON ZERO_POS=0 EXCUR=100 TIME=4]

Oscillating motion with feed motion of a path axis
N50 X[OSC ON 1ST_POS=111 2ND_POS=222 FREQ=1]
N60 G01 G90 Y500 F200

Oscillate with wait times of 0.5 s each at reversal positions
N70 X[OSC ON 1ST_POS-100 1ST_DELT0.5 2ND_POS200 2ND_DELT0.5 FEED1000]

Deselect oscillation
Oscillation is stopped when reversal position 2 is reached:
N80 X[OSC OFF]

Fast oscillation stop
If a feedrate is programmed with FEED in combination with OFF, the oscillating motion is stopped
immediately (feedhold of oscillating axis) and the reversal position 2 is directly approached at the
new feed rate.
N90 X[OSC OFF FEED=5000]

Immediate oscillation stop
The current oscillation motion is aborted (feedhold of oscillation axis). After standstill, the
current axis position is synchronised with the decoder. The axis is again available for coordinated
motion.
N90 X[OSC OFF INSTANT]

20.3 Cartesian/kinematic transformation and positioning axes

20.3.1 Positioning and shifts

The user must program absolute positions for this axes in the case of conventional operation and in CS mode (#CS, #ACS) with active independent axis or oscillating axis . This means that, if a tool change was executed, the tool length my have to be taken into consideration when programming the axes. Zero offsets (G54...G59) or reference point offsets (G92) previously programmed are not effective.

When kinematic transformation (#TRAFO) is active, tool offsets are considered directly in the transformation. This means that they are also considered for active independent axes or oscillating axes. In this case too, previously programmed zero offsets (G54...G59) or reference point offsets (G92) are not effective.

20.3.2 Restrictions

An oscillating motion or independent axis motion must be deselected before selecting a new Cartesian or kinematic transformation.

A positioning axis may only be programmed:

- with Cartesian kinematics and
- only in the 3rd axis (generally A. Z axis) for tools perpendicular to the XY machine base (e.g. A axis to 0° with CA head kinematic).

Programming independent axes:

```
N10 #KIN ID[9]
N20 #TRAFO ON
N30 Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_TYPE=STEP]
N40 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N50 G01 G90 X100 F0.1
N60 #TRAFO OFF
N100 M30


N10 G00 X0 Y0 Z0 C0
N20 #CS ON[0,0,0,0,0,45]
N30 Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_TYPE=STEP]
N40 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N50 G01 G90 X100 F0.1
N60 #CS OFF
N100 M30


N10 #KIN ID[9]
N20 #TRAFO ON
N30 #CS ON[0,0,0,0,0,45]
N40 Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_TYPE=STEP]
N50 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N60 G01 G90 X100 F0.1
N70 #CS OFF
N80 #TRAFO OFF
N100 M30
```

Programming oscillating axes:

```
N10 G00 X0 Y0 Z0 C0
N20 #KIN ID[9]
N30 #TRAFO ON
N40 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N50 G01 G90 X100 Y100 F0.1
N60 Z[OSC OFF FEED=2.00]
N70 #TRAFO OFF
N100 M30


N10 G00 X0 Y0 Z0 C0
```

```
N20 #CS ON[0,0,0,0,0,45]
N30 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N40 G01 G90 X100 Y100 F0.1
N50 Z[OSC OFF FEED=2.00]
N60 #CS OFF
N100 M30


N10 G00 X0 Y0 Z0 C0
N20 #KIN ID[9]
N30 #TRAFO ON
N40 #CS ON[0,0,60,0,0,45]
N50 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N60 G01 G90 X100 Y100 F0.1
N70 Z[OSC OFF FEED=2.00]
N80 #CS OFF
N90 #TRAFO OFF
N100 M30
```

Impermissible nested programming

The following program extract shows an impermissible nesting of CS with kinematic transformation and oscillation:

```
N10 #KIN ID[9]
N20 #TRAFO ON
N30 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N40 G01 G90 X100 Y100 F0.1
N50 #CS ON[0,0,0,0,0,45]
N60 G01 G90 X100 F0.1
N70 #CS OFF
N80 #TRAFO OFF
N90 Z[OSC OFF FEED=2.00]
N100 M30
```

21   Axis-specific programming

A complete list of axis-specific additional functions is contained in the overview of commands in the Appendix under Additional axis-specific functions (<X>[..]) [▶ 800].

The programming syntax of the following NC commands is based on the axis-specific programming of positioning axes. After the axis identifier, parameterisation takes place by keywords and, if applicable, the associated values.

21.1   Selecting/deselecting axis compensations in the NC program (COMP)

i   **This function is available as of CNC Build V2.10.1501.00.**

The various axis compensations [FCT-C5] can also be selected and deselected directly in the NC program in addition to the option of using the corresponding axis parameters. Axis-specific axis compensations for several axes in an NC block can be activated or deactivated simultaneously.

i   Axis compensations switched off by the COMP command has a global NC program effect, i.e. compensations are <u>not</u> automatically activated at program end. They must be switched back on explicitly using the COMP command in the subsequent NC program.

Syntax:

<axis_name> **[COMP** [[**ON** | **OFF** [**CROSS PLANE LEAD TEMP FRICT**]] | **OFF_ALL**] [**NO_MOVE**] {****}**]**

<axis_name>	Name of the axis
COMP	Identifier to select/deselect axis-specific compensation. Must always be programmed as the <u>first</u> keyword.
ON	Activates programmed compensation(s)
OFF	Deactivates programmed compensation(s)
CROSS	Keyword for cross compensation
PLANE	Keyword for plane compensation
LEAD	Keyword for spindle leadscrew error compensation
TEMP	Keyword for temperature compensation
FRICT	Keyword for friction compensation [as of Build V2.11.2022.05]
CROSSTALK	Keyword for crosstalk compensation [as of Build V3.1.3079.32]
OFF_ALL	Switch off all active compensations. No further compensation keywords may be programmed after the keyword.

NO_MOVE	By default the position offset occurring when axis compensations are switched on/off is driven out before the NC program processing is continued. The keyword NO_MOVE suppresses this motion. The channel is initialised with the changed axis position. The position offset is only deactivated at the next axis motion programmed in the NC program.
\	Separator ("backslash") for clear programming of the command over multiple lines.

Axis-specific programming

```
;Deactivate cross and plane compensation in the X axis
N10 X[COMP OFF CROSS PLANE

;Compensation programming of multiple axes in an NC block
N50 X[COMP OFF CROSS] Y[COMP ON LEAD TEMP]

;Deactivate all compensations in the Z axis
N100 Z[COMP OFF_ALL]

;Deactivate all compensations of the Y axis without axis motion
N200 Y[COMP OFF_ALL NO_MOVE]
```

21.2   Distance control (sensed spindles) (DIST_CTRL)

ℹ **The availability of this function depends on the configuration and on the version scope.**

If the tool-supporting axis (spindle axis) is equipped with the necessary hardware, this function specifies the distance of the tool to an uneven workpiece surface. The distance is detected by a measuring system and is continuously tracked by the NC on the uneven surface.

The distance control for a sensed spindle is enabled by the parameter P-AXIS-00328. It is activated by the following NC command. For more information please refer to the functional description "Distance control" [FCT-M3]

```
Syntax:
```
<axis_name> **[DIST_CTRL** [**ON** [**DRYRUN**] [**CONST_DIST**]] | [**OFF** | **CHECK_POS** | **FREEZE** | **REF**]

        SET_POS=.. **SET_DIST**=.. [**KP**=..] [**I_TN**=..] [**D_TV**=..] [**FILTER_TYPE**=..]

        [**N_CYCLES**=..] [**FG_F0**=..] [**ORDER**=..] [**SMOOTH_FACT**=..] [**KALMAN_SIGMA**=..]

        [**NO_MOVE**] [**VAL1**=.. - **VAL5**=..] { **** } **]**

<axis_name>	Name of the axis supporting the tool.
DIST_CTRL	Identifier for the "Sensed spindles" function. Must always be programmed as the <u>first</u> keyword.
ON	Distance control on when workpiece surface is specified. A set position (SET_POS) must be set at switch-on.
CONST_DIST	Activate distance control (ON) when a constant distance to the workpiece surface is specified. A distance must be set with SET_DIST at switch-on. [as of Build V2.11.2804.03]
OFF	Deactivate distance control.
CHECK_POS	Check whether position is within the tolerance window.
FREEZE	Freeze the control distance across the workpiece. The axis position or the output correction value is maintained. Axis tracking is interrupted.
DRYRUN	In combination with ON, the axis is not tracked in DRYRUN mode when there are changes in the workpiece surface. This allows data to be evaluated without feedback from the controller (e.g. filter effect). **[as of V3.1.3079.23]**
	When distance control is activated, a set position must be set with SET_POS if the workpiece surface is specified.
	When distance control is activated, a set distance must be set with SET_DIST if a constant distance from the workpiece surface is specified.
REF	Reference measuring system (sensor) (only if there is no absolute measuring system).

SET_POS=..	Specify the workpiece surface in [mm, inch] (absolute position). In the event of reset or program end, the set position is reset, i.e. a new set position must be specified before distance control is reactivated.
SET_DIST=..	Specify the constant distance to the workpiece surface in [mm, inch]. In the event of reset or program end, the distance is reset, i.e. a new distance must be specified before distance control is reactivated.
KP=..	Weighting the distance control output values. Parameterisation can be executed analogous to P-AXIS-00759. The value range is limited to 0.0 < KP <= 2.0. For KP values less than 1.0, the distance control dynamics are reduced; for KP values greater than 1.0, the dynamics are increased. A KP factor less than 1.0 reduces 1 possible distance control oscillation and steadies control in the event of minor distance errors. **[as of V2.11.2809.06 or V3.1.3079.06]**
I_TN=..	Integral action time of the PID controller in [s]. The integral action time defines the time after which the P and I components of the manipulated variable are equal. Parameterisation can be executed analogous to P-AXIS-00764. The value range is limited to 0.0 <= I_TN <= 50.0. A large integral action time produces greater control stability. The shorter the integration action time, the greater the I component and the faster the control. A short integral action time excites oscillations more strongly. **[as of V2.11.2809.06 or V3.1.3079.06]**
D_TV=..	Derivative action time of the PID controller in [s]. The derivative action time defines the time after which the P and D components of the manipulated variable are equal. Parameterisation can be executed analogous to P-AXIS-00765. The value range is limited to 0.0 <= D_TV <= 2.0. The larger the derivative action time, the stronger the D component. **[as of V2.11.2809.06 or V3.1.3079.06]**
FILTER_TYPE=..	Filter type to filter sensor values according to P-AXIS-00782. **[as of V3.1.3079.23]**
N_CYCLES=..	Number the measured values used for filtering according to P-AXIS-00413. **[as of V3.1.3079.23]**
FG_F0=..	Cut-off frequency for the low-pass filter in [Hz] according to P-AXIS-00508. **[as of V3.1.3079.23]**
ORDER=..	Order of the low-pass filter according to P-AXIS-00507. **[as of V3.1.3079.23]**
SMOOTH_FACT=..	Smoothing factor of the exponential averaging filter according to P-AXIS-00784. Specifies the weighting of the current measured value.
KALMAN_SIGMA=..	Uncertainty of the included measured values according to P-AXIS-00783. **[as of V3.1.3079.23]**
NO_MOVE	By default, the resulting correction offset is deactivated when distance control is switched off. This motion can be suppressed by specifying NO_MOVE in combination with OFF. The channel is initialised with the changed axis position. The position offset is only deactivated at the next axis motion programmed in the NC program.
VAL1=..-VAL5=..	Five freely assignable values in real format.
\	Separator ("backslash") for clear programming of the command over multiple lines.

The keywords ON/OFF, FREEZE, DRYRUN, CHECK_POS and REF in the same command sequence mutually cancel each other out.

The keywords SET_POS, SET_DIST and VAL1-VAL5 can always be programmed in combination with each other.

---

i If distance control is still active at program end, it is not automatically deselected.

When a reset or axis error occurs, active distance control is always deselected automatically.

---

i Parameters of the PID controller are not reset at program end.

BECKHOFF

Programming examples for distance control

```
%DIST_1
; Set expected position of the workpiece surface
N10 Z[DIST_CTRL SET_POS=30]

N20 Z[DIST_CTRL ON]          ;Select
; …
Nxx Z[DIST_CTRL OFF]         ;Deselect
N999 M30
```

```
%DIST_2
; Select + set expected position of the workpiece surface
N10 Z[DIST_CTRL ON SET_POS=30]

; …
Nxx Z[DIST_CTRL FREEZE]      ;Hold position
; …
Nxx Z[DIST_CTRL OFF]         ;Deselect
N999 M30
```

```
%DIST_3
; Select + set expected position of the workpiece surface
N10 Z[DIST_CTRL ON SET_POS=50]

; Deactivate distance control; Z axis does not move
Nxx Z[DIST_CTRL OFF NO_MOVE]
; The generated compensation offset is included for motion to the target point
; 100.
Nxx G0 Z100
N999 M30
```

```
%DIST_4
; Set distance parameters
N10 Z[DIST_CTRL SET_POS=30 SET_DIST=10]
; Select with specified workpiece surface (SET_POS)
N20 Z[DIST_CTRL ON]
; …
Nxx Z[DIST_CTRL OFF]         ;Deselect
; …
; Select with specified workpiece surface (SET_DIST)
Nxx Z[DIST_CTRL ON CONST_DIST]
; …
Nxx Z[DIST_CTRL OFF          ;Deselect
N999 M30
```

```
%DIST_5
N10 Z[DIST_CTRL FILTER_TYPE=KALMAN_MA] ;Select filter type
; Parametrise filter
N20 Z[DIST_CTRL N_CYCLES=30 KALMAN_SIGMA=1000]
; Activate distance control
N30 Z[DIST_CTRL ON CONST_DIST SET_DIST=1].
; …
; Change filter
Nxx Z[DIST_CTRL FILTER_TYPE=KALMAN_EXPO SMOOTH_FACT=0.3]
.
.
Nxx Z[DIST_CTRL OFF]; Deselect
N999 M30
```

21.3   Programmable axis override (OVERRIDE)

This command allows for the axis feed, if required the different influencing of feed and rapid feed blocks in the NC program. The axis-specific programmed override is active during path motions if the axis is moving. This does not affect the mode of operation of real-time influencing of feed by the PLC.

In addition a programmable path override [▶ 436] function is also provided.

When several axes are moved in the same NC block with different axis-specific override values, the smallest override always takes effect. If an additional path override is also defined, the effective override results from multiplying the two override values.

ℹ️   The G166 function [▶ 177] suppresses the programmed override values.

Syntax:

<axis_name> **[OVERRIDE FEED_FACT**=.. **RAPID_FACT**=.. **{ \ }]**

<axis_name>	Name of the axis
OVERRIDE	Identifier for axis-specific override programming. Must always be programmed as the <u>first</u> keyword.
FEED_FACT=..	Override factor for feed blocks in [0.1%-200%]
RAPID_FACT=..	Override factor for rapid traverse blocks in [0.1%-200%]
\	Separator ("backslash") for clear programming of the command over multiple lines.

Programmable axis override

```
%ax_override

N10  G01 X100 Y100 Z100 F1000
N40  X[OVERRIDE FEED_FACT=20 RAPID_FACT=60] Axis override X G01 20%, G00 60%
N50  Y[OVERRIDE FEED_FACT=30 RAPID_FACT=70] Axis override Y G01 30%, G00 70%
N60  Z[OVERRIDE FEED_FACT=40 RAPID_FACT=80] Axis override Z G01 40%, G00 80%
N50  G00 X0                 G00 motion with 60% override
G00 Y0                      G00 motion with 70% override
G00 Z0                      G00 motion with 80% override
N80 G01 X100 F2000          G01 motion with 20% override
N90  Y100                   G01 motion with 30% override
N100 Z100                   G01 motion with 40% override
N110 X200 Y200              G01 motion with 20% override
N120 X300 Y300 Z200         G01 motion with 20% override
M30
```

21.4 Programmable acceleration overload (DYNAMIC)

For technological reasons it may be necessary to exceed the specified dynamic limit values of the drive in connection with contour affecting influencing processes e.g. to ensure a constant path velocity on polynomial contours.

The following axis-specific command combined with the assigned parameter P-AXIS-00394 permits a weighting of the axis dynamic in percent % in excess of the permissible maximum acceleration P-AXIS-00008. P-AXIS-00394 represents the permissible upper limit for the acceleration weighting factor of the axis in per mill ‰. The weighting factor refers to the feed dynamic limit values of the corresponding active slope profile.

Currently the acceleration weighting function can be used in conjunction with Contouring mode 6.

```
Syntax:
```
<axis_name> **[DYNAMIC DIST_SOFT** | **ACC_FACT**=.. { \ } **]**

<axis_name>	Name of the axis
DYNAMIC	Identifier for axis dynamic weighting. Must always be programmed as the <u>first</u> keyword.
DIST_SOFT	Identifier for the polynomial contouring mode 6
ACC_FACT=..	Axis-specific weighting factor in [%]
\	Separator ("backslash") for clear programming of the command over multiple lines.

---

i The minimum weighting value is 100%.

The maximum weighting value is limited to P-AXIS-00394.

Programmable acceleration overload

```
%dynamic
N10 #SLOPE[TYPE=STEP]
N20 #CONTOUR MODE[DIST_SOFT PATH_DIST=35 ACC_MAX=100 ]
N30 C[DYNAMIC DIST_SOFT ACC_FACT=200]
(* Acceleration overload factor for C axis 200% *)
N30 G1 G91 G261
N40 X59.485 F10000
N50 X105.172 C26.992
N60 X113.189 C46.171
N70 X100.348 C-46.171
N80 X99.179 C-26.992
N90 G260 X138.799
N100 G261
M30
```

21.5 Synchronising an axis in coordinated motion (SYNC IN / OUT)

> ℹ **This function is available as of CNC Build V2.11.2013.22**

Some specific processes require a synchronised motion of a single axis (slave axis) in combination with a coordinated motion. At certain programmed positions it is required that the slave axis is located at a specific position and moves at a specific velocity. The slave axis then moves at the synchronised velocity until synchronisation is cancelled.

Typical application examples include machines with the continuous throughput of endless material. The material must then be cut at a specific place during the coordinated motion. At a specific master position (workpiece length) the rotating knife must be placed in cutting position. The knife then moves at constant velocity until the cut is finished.

Fig. 203: Synchronised cutting

Restrictions:

An axis cannot be synchronized if:

- the axis is moving in coordinated motion at the time.

Configuration:

To use this function, the following setting must be made in the start-up list ([STUP]):

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_SYNC**

configuration.channel[0].interpolator.function FCT_IPO_DEFAULT | **FCT_SYNC**

Syntax of Programming synchronous motions:

<axis_name> [SYNC IN | OUT G90 | G91 G00 | G01 FEED=.. **FEED_MAX_WEIGHT**=.. POS=.. DIST=.. **{\}]**

<axis_name>	Name of the axis to be synchronised
SYNC	Identifier for synchronised axis motion. Must always be programmed as the <u>first</u> keyword.
IN	Identifier to mark the start of synchronised motion.
OUT	Identification to mark the end of synchronised motion.
G90 / G91	Absolute/relative dimension
G00 / G01	Rapid traverse/linear interpolation
FEED=..	Axis-specific feedrate in [mm/min, m/min, inch/min]
FEED_MAX_WEIGHT =..	Weighting factor in [%], referred to axis-specific maximum feed P-AXIS-00212. Only weighting values less than 100% are permitted (<u>as per G194 [</u>▶ <u>149]</u>).
POS=..	Axis position in [mm, inch] at which the synchronous velocity is reached.
DIST=..	Distance in [mm, inch] at which the synchronous velocity is moved..
\	Separator ("backslash") for clear programming of the command over multiple lines.

Synchronising an axis in coordinated motion

```
%sync
N010 G90 X0 Y0 Z0 A0
N020 G91 F5000
N030 X=67.913 A[SYNC IN G01 FEED_MAX_WEIGHT=100 G91 POS=130 DIST=70]
               ;A axis reaches maximum velocity on axis position 130,
               ;while X axis is reaching position 67.913 at this point

N040 X=1.5 ;A and X axes move synchronously, in doing so the velocity
               ;of the X axis is then defined so that the X axis covers a distance of 3mm
N050 X=1.5     ;while the A axis moves 70° at maximum velocity

N060 X=14.541 A[SYNC OUT G91 G0 POS160] G261
               ;At beginning of this block synchronous motion is cancelled
               Path axes move again at the programmed feed rate; the A axis
               moves independently to the specified position

N070 X=15.862 Z=1.248 Y=0.185
N080 X=15.992 Z=1.889 Y=0.213
N090 X=32.243 Z=3.306 Y=0.482
N100 X=22.186
N110 X=31.696 Z=-2.597 Y=-0.389
N120 X=25.297 Z=-3.846 Y=-0.491
N130 X=39.819 A[SYNC IN G01 FEED_MAX_WEIGHT=100 G91 POS=130 DIST=70]
N140 X=1.257
N150 X=1.257
N160 X=200     A[SYNC OUT G91 G0 POS160]
N180 M30
```

21.6 Programming an axis polynomial (POLY)

ℹ **This function is available as of CNC Build V2.11.2016.08**

Characteristics of axis-specific polynomials

The motion rules for an axis can be programmed by specifying axis-specific polynomials.

This axis-specific polynomial motion is programmable for linear motions (G00, G01). The dynamic parameters of the currently active G function (G00 or G01) are used.

An upper limit of the polynomial parameter is specified for each axis polynomial up to which the polynomial parameter is interpolated. If the upper limit is not programmed, the value 1.0 is assigned.

The polynomial coefficients of an axis polynomial are defined in square brackets after the axis identifier in ascending order. The keyword POLY is always specified first as identification. Higher polynomial coefficients which are not required can be omitted. Coefficients which are not programmed are assigned the value 0. At least the first coefficient 'A0' must be set.

The maximum possible degree of the polynomial is 5.

Evaluation

The polynomial coefficients refer to the specification of <u>absolute</u> axis positions and are used in a 5th degree polynomial:

$p(s) = A0 + A1 * s + A2 * s^2 + A3 * s^3 + A4 * s^4 + A5\ s^5$

The polynomial parameter is interpolated for the polynomial from zero up to the programmed upper limit simultaneously with the executed motion path.

This applies to the absolute position of the polynomial axis in mm or degrees:

At motion start (s = 0):
$$p(0) = A0$$

At motion end (s = L):

$$p(L) = A0 + A1 * L + A2 * L^2 + A3 * L^3 + A4 * L^4 + A5\ L^5$$

Axis-specific polynomial programming is only effective blockwise. Therefore, if required, it must be reused in the next motion block for each axis.

ℹ When determining the polynomial coefficients, it is important to ensure that the axis position is continuous, i.e. the polynomial value at 0 position must correspond to the axis position of the previous motion block.

The repeated programming of axis polynomials in sequenced NC blocks requires that the end position of a polynomial corresponds to the start position of the next polynomial.

Since the value at position 0 is only defined by the coefficient A0 for a polynomial, the following applies: A0 is equal to the axis position from the previous motion block.

Programming

Scheme:

Axis [POLY L<*Maximum value of polynomial parameter*> A0 A1 A2 A3 A4 A5]

Example:

X [POLY L=1.0 A0=0.1 A1=0.2 A2=0.3 A3=0.4 A4=0.5 A5=0.6]

The same NC block may contain a mixed programming of linear motions and <u>one</u>* axis-specific polynomial. Active offsets (G54, G92, #PSET...) may be included in the programmed polynomial positions.

No calculation or monitoring of the dynamic is executed for the programmed polynomial axis. Similarly, no command value monitoring of software limits takes place (only actual value specific monitoring of limits).

Syntax:

<axis_name> **[POLY** [**L**=..] **A0**=.. **[A1**=.. **A2**=.. **A3**=.. **A4**=.. **A5**=..] **]**

<axis_name>	Name of the polynomial axis
POLY	Identifier for the polynomial programming of an axis. Must always be programmed as the <u>first</u> keyword.
L=..	Upper limit of the polynomial parameter of the axis to be moved without unit (optional: if not programmed, L has value 1.0, programmed value must be > 0)
A0=..	First polynomial coefficient, mandatory (start value of polynomial)
A1=.. - A5=..	Second to sixth polynomial coefficient, x, x^2, x^3, x^4, x^5 (optional: default value of non-programmed coefficients is 0)

Programming an axis polynomial

```
;C axis, polynomial parameter L = 0.7 A0 = 0.1, A1 = 0.3, A2 = 0.5
Nxx C0.1
Nxx C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]

;X axis, polynomial parameter L = 0.3, A0 = 0.2, A1= 0.5
Nxx X0.2
Nxx X[POLY L=0.3 A0=0.2 A1=0.5]

;Simple programming without polynomial parameter (default L1), only A0 coefficient
Nxx X0.2
Nxx X[POLY A0=0.2]

;Mixed programming of linear motion and axis polynomial
Nxx C0.1
Nxx G01 F1000 X100 Y150 C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]

;Note: The equals signs between the keyword and the value are optional
```

BECKHOFF

21.7 Setting an axis position in the channel (SET_POSITION)

ℹ **This function is available as of CNC Build V2.11.2808.**

This command sets the current position of an axis in the NC channel to a defined value. This value acts on the ACS level (in the position controller). It does not initiate a motion. Instead, it marks the axis as referenced after the axis position is repositioned. The NC channel is then initialised with the new axis positions, taking into consideration any active offsets.

The new axis position is specified with as an absolute value (POS) or a relative value to the current position (OFFSET).

```
Syntax:
```
<axis_name> **[SET_POSITION POS**=.. | OFFSET=.. {\}**]**

<axis_name>	Name of the axis
SET_POSITION	Identifier for the function of setting an axis position. Must always be programmed as the <u>first</u> keyword.
POS=..	New defined absolute axis position in [mm, inch]
OFFSET=..	Relative offset to the current axis position in [mm, inch]
\	Separator ("backslash") for clear programming of the command over multiple lines.

Set an axis position

```
%set_pos.nc
N010 G01 F2000 X0 Y0 Z0 A0 B0 C0
N020 $FOR P1=0,100,1
N030 G91 X100 ;Axis X moves to 10000mm
N040 $ENDFOR
N050 X[SET_POSITION POS=100] ;Set X axis position to 100
;..
;..
N999 M30
```

21.8 Lifting/lowering an axis (LIFT)

For more information please refer to the functional description "Collision detection by lift function" [FCT-A11].

Cross-block lifting/lowering

Programming is based on the syntax for independent axes. The corresponding parameters can be programmed at the start of lifting/lowering. These are non-modal parameters, i.e. if required they are reset for every start.

```
Syntax:
```
<axis_name> **[LIFT_START** [**DOWN**] [**G90** | **G91**] [**POS**=..] **POS_LIMIT**=.. **]**

<axis_name>	Lift axis name
LIFT_START	Identifier for the start of the (cross-block) independent lifting motion of the axis. Must always be programmed as the <u>first</u> keyword.
DOWN	The axis motion direction can be inverted via DOWN, i.e. the motion is in the direction of the negative software limit switch. If nothing is specified, the default direction is in the direction of the positive software limit switch.
G90 / G91	Absolute/relative dimension; the default dimension is G90. G91 is non-modal and is only active for the lifting/lowering motion.
POS=..	Target position of the lift axis after the lifting motion in [mm, inch]. The current command position of the axis (see V.A.ABS.*<axis name>*) is the default.
POS_LIMIT=..	Maximum lifting height or lowering depth in [mm, inch]

```
Syntax:
```
<axis_name> **[LIFT_END]**

| *<axis_name>* | Lift axis name |
| LIFT_END | Identifier for the end of the (cross-block) independent lifting motion of the axis. |

Cross-block lifting/lowering

```
N10 X10 Y20 Z30     ;Cut with laser
N20 M5              ;Laser off
N30 Z[LIFT_START POS=12 POS_LIMIT=100]    ;Lift Z axis
N30 G01 X.. Y..
N40 G02 X.. Y..
N50 G03 X.. Y..
N60 G01 X.. Y..
N70 Z[LIFT_END]    ;Absolutely lower Z axis to target 12 mm
N80 M4              ;Laser on
N90 X20 Y20 ...

N10 X10 Y20 Z30
N30 Z[LIFT_START POS=12 POS_LIMIT=100] ;Lift Z axis
N40 G01 X.. Y..
N50 G01 X.. Y..
N60 Z[LIFT_END]    ;Absolutely lower Z axis to target 12 mm
N70 X100

alternative programming
N110 X10 Y20 Z30
N140 G01 X.. Y.. Z[LIFT_START POS=12 POS_LIMIT=100]
N150 G01 X.. Y.. Z[LIFT_END]
N170 X100
```

Lifting/lowering in an NC block

Programming is based on the syntax for independent axes. The corresponding parameters can be programmed at the start of lifting/lowering. These are non-modal parameters, i.e. if required they are reset for every start.

Syntax:

<axis:name> **[LIFT** [**DOWN**] [**G90** | **G91**] [**POS**=..] **POS_LIMIT**=.. **]**

<axis_name>	Lift axis name
LIFT	Identifier for the start and end of the independent lifting motion of the axis in the current NC block Must always be programmed as the <u>first</u> keyword.
DOWN	The axis motion direction can be inverted via DOWN, i.e. the motion is in the direction of the negative software limit switch. If nothing is specified, the default direction is in the direction of the positive software limit switch (option not available as at 10/2011).
G90 / G91	Absolute/relative dimension. The default dimension is G90. G91 is non-modal and is only active for the lifting/lowering motion.
POS=..	Target position of the lift axis after the lifting motion in [mm, inch]. The current command position of the axis (see V.A.ABS.*<axis name>*) is the default.
POS_LIMIT=..	Maximum lifting height or lowering depth in [mm, inch]

Lifting/lowering in an NC block

```
; single-row programming
N200 Z40
N240 X10 Y.. Z[LIFT POS=30 POS_LIMIT=300]
N250 X20 Y.. Z[LIFT POS=20 POS_LIMIT=300]
N260 X30 Y.. Z[LIFT POS=25 POS_LIMIT=300]
N270 X.. Y.. Z[LIFT POS=30 POS_LIMIT=300]
N280 X.. Y.. Z[LIFT POS=30 POS_LIMIT=300]
```


Fig. 204: Single-row lifting

21.9   Moving to fixed stop (FIXED_STOP)

For more information please refer to the functional description "Moving to fixed stop" [FCT-M8]

```
Syntax:
```
<axis_name> **[FIXED_STOP [ON | OFF] [TORQUE_LIMIT**=..] [**POS_LAG_LIMIT**=..] [**CYCLES**=..]

[**WINDOW**=..] [**START**=..] [**END**=..] [**ERR_NOT_DETECTED**=..] { \ } **]**

<axis_name>	Name of the axis to be used with the "Move to fixed stop" function.
FIXED_STOP	Identifier for position lag during "Move to fixed stop". Must always be programmed as the <u>first</u> keyword.
ON	Activate the "Move to fixed stop" function for this axis. Motion information must also be specified for the axis.
OFF	Deactivate the "Move to fixed stop" function. In addition a motion should be programmed for the axis to move away from the fixed stop.
TORQUE_LIMIT=..	Specifying the torque limit with "Move to fixed stop". Scaling is determined by parameterising the "Move to fixed stop" function from the axis parameters (see P-AXIS-00724). Normally this is given as a percentage [%] of the drive nominal torque. If no torque is specified for the axis in the NC program, the default value in the axis parameter P-AXIS-00729 is used for the torque limit.
POS_LAG_LIMIT=..	Limit for position lag in [mm, inch, °]. If this limit is overshot, the CNC reverts to the "Fixed stop reached" state after the LR cycles specified in CYCLES. If no position lag limit is specified for the axis in the NC program, the default value in the axis parameter P-AXIS-00712 is used.
CYCLES=..	Number of position controller cycles in which the position lag must be above the specified POS_LAG_LIMIT limit before the controller reverts to the "Fixed stop reached" state. If the number of position controller cycles is not specified for the axis in the NC program, the default value from the axis parameter P-AXIS-00714 is used.
WINDOW=..	Tolerance window for fixed stop position in [mm, inch, °]. After the fixed stop is reached, the controller checks whether the actual position of the drive leaves the specified tolerance window to detect a breakaway of the fixed stop. If no tolerance window is specified for the axis in the NC program, the default value value in the axis parameter P-AXIS-00713 is used. The monitor is disabled with a value of 0.
START=..	This parameter can delay the monitoring function for when the fixed stop is reached by one percent [%] referred to the path distance in order to prevent the incorrect detection of a fixed stop due to friction etc. when the axis starts off. If this parameter is not specified in the NC program, the start of the motion is always monitored (START = 0%).
END=..	This parameter can prematurely end the monitoring function for when the fixed stop is reached by one percent [%] referred to the path distance in order to prevent the incorrect detection of a fixed stop when the axis is decelerated on approaching the target point. If this parameter is not specified in the NC program, monitoring always takes place up to the target point of the motion block (END = 100%).

ERR_NOT_DETECTED=..	This parameter suppresses the output of error message P-ERR-50886 if the fixed stop is not detected in the approach motion. This permits the execution of simple measurement processes, e.g. with "Move to fixed stop".
	0: Output error message P-ERR-50886.
	1: Do <u>not</u> output error message P-ERR-50886.
\	Separator ("backslash") for clear programming of the command over multiple lines.

21.10   Rotary axes

i   **These functions are available as of CNC Build V3.1.3079.40.**

The properties of rotary axes with regards to monitoring and defining software limit switches and modulo limits are defined in the axis-specific configuration lists. These properties can no longer be changed after controller start-up or program start. However, certain applications require these settings to be changed in the NC program for technology reasons.

For example, wire erosion requires rotary axes that can be operated both as modulo axes and as endless rotary axes. In addition, monitoring and parameterising the software limit switches also require a change in the NC program. The following transformation functions for this are available:

21.10.1   Programming software limit switch monitoring (POS_LIMIT)

The default setting of the software limit switches (SLS) is configured in the axis-specific lists by the parameters P-AXIS-00177 and P-AXIS-00178. After controller start-up, these limits are monitored for translatory axes and for every rotary **non-modulo** axis. The effectiveness of the monitoring function can be switched on and off for each axis by setting P-AXIS-00705 irrespective of axis type and axis mode. With modulo axes, the limits are only efrfective if they are located within the modulo range.

These default settings can be changed by NC command.

```
Syntax:
```
<axis_name> **[POS_LIMIT ON** | **OFF** | **DEFAULT** [**MIN**=.. **MAX**=..] [**BEHAVIOUR**=*<error_mode>*] { \ } **]**

<axis_name>	Name of the axis
POS_LIMIT	Identifier for the axis-specific programming of the software limit switch monitoring function. Must always be programmed as the <u>first</u> keyword.
ON	Select software limit switch monitoring with new limits. MIN, MAX are optional; if not programmed, the previous limits remain valid.
OFF	Deselect software limit switch monitoring
DEFAULT	Reset the configured default values as per P-AXIS-00177, P-AXIS-00178 and P-AXIS-00705
MIN=..	Lower limit switch limit in [mm, inch, °]
MAX=..	Upper limit switch limit in [mm, inch, °]
BEHAVIOUR =*<error_mode>*	Define the error response if the software limit switch is crossed:
	ERROR: Crossing results with an error already in path preparation (command value monitoring)
	ERROR_LR: Crossing results in a warning during path preparation. If crossing takes place, an error is output in the position controller (actual value monitoring).
	WARNING: If crossing takes place, only warnings are output in path preparation and in the position controller.
\	Separator ("backslash") for clear programming of the command over multiple lines.

ℹ The software limit switches are checked in the axis coordinate system. The CNC channel checks the command positions, while the position controller checks the actual positions.

ℹ The configured limits P-AXIS-00177 and P-AXIS-00178 are not changed by this command. The limit switch limit values are reset to the configured default values at CNC reset, program start and axis exchange (e.g. #CALL AX). The change to the limit switch limit values when axis lists are re-read is considered at the next program start.

Programming of software limit switch monitoring


```
%pos_limit
:
;Select SLS monitoring:
N100 A[POS_LIMIT ON MIN=-135 MAX=135 BEHAVIOUR=WARNING]
:
;Select configured default values:
N200 A[POS_LIMIT DEFAULT]
:
;Deselect SLS monitoring:
N300 A[POS_LIMIT OFF]
:
M30
```

21.10.2 Programming the modulo range (MODULO)

The default setting of the modulo limits is configured in the axis-specific lists by the parameters P-AXIS-00126 and P-AXIS-00127. After controller start-up, the limits are active for rotary axes with MODULO axis mode and spindles. With linear axes, modulo calculation is switched off by default in the default setting. The effectiveness of the modulo calculation can be switched on and off for each axis by setting P-AXIS-00557 irrespective of axis type and axis mode.

These default settings can be changed by NC command.

```
Syntax:
```
<axis_name> **[MODULO ON | OFF | OFF_POS_INIT | DEFAULT | SHIFT** [**MIN**=.. **MAX**=..] { **** } **]**

<axis_name>	Name of the axis
MODULO	Identifier for the axis-specific programming of the modulo function. Must always be programmed as the <u>first</u> keyword.
ON	Select the modulo function with new limits. MIN, MAX are optional; if not programmed, the previous limits remain valid.
OFF	Deselect the modulo function.
OFF_POS_INIT	Deselect the modulo function with an implicit position initialisation of the NC channel (see #CHANNEL INIT [CMDPOS] [▶ 172]).
DEFAULT	Reset the configured default values as per P-AXIS-00126, P-AXIS-00127 and P-AXIS-00557.
SHIFT	Convert the current nominal position in the module range. After conversion, the previous modulo settings are again valid.
MIN=..	Lower modulo limit in [°]
MAX=..	Upper modulo limit in [°]
\	Separator ("backslash") for clear programming of the command over multiple lines.

If kinematic and/or Cartesian transformations are active in the channel, the programmed modulo limits refer to the programming coordinate system (PCS) or the machine coordinate system (MCS). If no transformations are active, the modulo range is defined in the axis coordinate system (ACS).

In the position controller, the modulo property of the axis is fixed after start-up and cannot be changed by programming.

If the modulo range is changed or activated, the current axis position is converted to the new modulo range.

Programming the modulo function

```
%modulo_calc_1
:
N100 A[MODULO ON MIN=0 MAX=360] ;Select modulo monitoring
N110 A-+700                      Rotate in negative direction to position 340°
N120 A[MODULO DEFAULT]          ;Select configured default values
:
N200 A[MODULO OFF]
N210 X100 A2000                 ;Motion without modulo calculation
N220 A[MODULO ON MIN=0 MAX=360] ;Position 2000 to mod(360)=200
N230 A-+700                      Rotate in negative direction to position 340°
:
M30

;Programming example for modulo conversion (SHIFT):
%modulo_calc_2
:
N10 A[MODULO=OFF]                   ;Enable multiple rotation
N20 X100 A2000                      ;Rotate A axis multiple times without modulo
N30 A[MODULO=SHIFT MIN=0 MAX= 360] ;Position 2000 once to
                                    ;mod(360)=200
N40 X500 A2000                      ;Rotate A axis multiple times without modulo
:
M30
```

22 References

[1] Documentation/General description of channel parameters [CHAN]

No.	Description
1	Elements of the structure makro_def[i].*
2	Elements of the structure synchro_data.koppel_gruppe[0].*
3	Elements of the structure spindel[i].*
4	Elements of the structure spindel[i].range_table[i].*
5	Elements of the structure gruppe[j].achse[i].*
6	Elements of the structure speed_limit_look_ahead.*
7	Elements of the structure dynamic_weighting[i].*

[2] Documentation/General description of axis parameters [AXIS]

No.	Description
1	Elements of the structure getriebe[i].slope_profil.*
2	Elements of the structure getriebe[i].lslope_profil.*
3	Elements of the structure filter[i].*

[3] Documentation/General description of zero offset data [ZERO]

[4] Specifications of SERCOS Interface, IEC 61491

[5] Documentation/General description of tool data [TOOL]

[6] Documentation of control/manufacturer specific settings of system parameters [SYSP]

[7] Documentation/General description of start-up list [STUP]

[8] Documentation/General description of external variables [EXTV]

[9] Motion Control Platform for PLCopen [MCP-P1]

23   Appendix

23.1   Overview of commands

23.1.1   G functions (G..)

23.1.2   M functions (M..)

M00	Programmed stop	p [▶ 185]
M01	Optional stop	p [▶ 185]
M02	Program end	p [▶ 185]
M03	Spindle rotation cw	
	• in DIN syntax	p. [▶ 590]
	• in spindle-specific syntax	p. [▶ 634]
M04	Spindle rotation ccw	
	• in DIN syntax	p. [▶ 590]
	• in spindle-specific syntax	p. [▶ 634]
M05	Stop spindle	
	• in DIN syntax	p. [▶ 590]
	• in spindle-specific syntax	p. [▶ 634]
M06	Calling a tool change program	p [▶ 186]
M17	Subroutine end	p [▶ 185]
M19	Positioning spindle	
	• in DIN syntax	p. [▶ 591]
	• in spindle-specific syntax	p. [▶ 636]
M29	Subroutine end	p [▶ 185]
M30	Program end	p [▶ 185]
M40 - 45	Selecting spindle gear stages	p [▶ 595]

23.1.3 Functions reserved according to DIN and ISG extensions

D	Tool geometry compensation (tool compensation data)	p. [▶ 470]
E	Feedrate at block end	p [▶ 192]
F	Feedrate in the block	p [▶ 192]
M/H	User-specific technology functions (M/H)	
	• in DIN syntax	p. [▶ 184]
	• in spindle-specific syntax	p. [▶ 639]
L	Global subroutines	p. [▶ 198]
LL	Local subroutines	p. [▶ 197]
L , LL CYCLE	Cycles as global or local subroutines	p. [▶ 203]
L SEQUENCE	Calling block sequences	p [▶ 210]
N	Block number	p [▶ 195]
P	Parameters and parameter calculation	p. [▶ 219]
R	Radius programming	p [▶ 134]
S	Spindle speed	
	• in DIN syntax (S word)	p. [▶ 593]
	• in spindle-specific syntax (REV)	p. [▶ 638]
S.POS	Position spindle	
	• in DIN syntax	p. [▶ 591]
	• in spindle-specific syntax (POS)	p. [▶ 636]
S.OFFSET	Angle offset for multi-turn threads	p [▶ 604]
T	Selecting tool position	p [▶ 191]
/	Skipping NC blocks	p [▶ 27]
****	Line break in NC block	p [▶ 29]
"..."	Macro programming	p [▶ 672]

23.1.4   Control block statements ($..)

$BREAK	BREAK statement	S. [▶ 239]
$CONTINUE	CONTINUE statement	S. [▶ 240]
$DO **$ENDDO**	DO loop	S. [▶ 237]
$REPEAT **$UNTIL**	REPEAT loop	S. [▶ 237]
$FOR **$ENDFOR**	FOR loop	S. [▶ 235]
$GOTO	GOTO statement	S. [▶ 230]
$IF **$ELSE** **$ELSEIF** **$ENDIF**	The IF-ELSE branching	S. [▶ 227]
$SWITCH **$CASE** **$DEFAULT** **$ENDSWITCH**	Switch branching	S. [▶ 230]
$WHILE **$ENDWHILE**	WHILE loop	S. [▶ 237]

23.1.5   Additional functions (#..)

A

#ACS ON/OFF	Defining/activating a clamping position compensation coordinate system	p. [▶ 697]
#ADD	Additional information at block end	p. [▶ 355]
#AKIMA STARTVECTOR	Defining start tangent	p. [▶ 286]
#AKIMA ENDVECTOR	Defining end tangent	p. [▶ 287]
#AKIMA TRANS	Specifying spline curve motion block	p. [▶ 286]
#ANG	Contour line programming	p. [▶ 69]
#AX DEF	Defining an axis configuration (extended syntax)	p. [▶ 318]
#AX DEF DEFAULT	Loading default axis configuration (extended syntax)	p. [▶ 318]
#AX LINK ON/OFF/OFF ALL	Programming axis couplings	
	• extended syntax	p. [▶ 339]
	• Extension "SOFT-GANTRY"	p. [▶ 341]
#AX LOCK/UNLOCK ALL	Locking an axis motion during PTP	p. [▶ 686]
#AX REQUEST	Requesting axes (extended syntax)	p. [▶ 310]
#AX RELEASE	Releasing axes (extended syntax)	p. [▶ 316]
#AX RELEASE ALL	Releasing all axes (extended syntax)	p. [▶ 316]

B

#BACKWARD STORAGE CLEAR	Clearing backward storage	p. [▶ 432]
#BCS DEF	Defining and storing a BCS	p. [▶ 705]
#BCS ON/OFF	Defining, storing and activating a BCS	p. [▶ 705]
#BLOCKSEARCH LOCKED/ RELEASED	Lock program areas for block search	[FCT-C6]

C

#CACHE LOAD/CLEAR/ALL	Load NC programs to local cache	[FCT-C23]
#CALL AX	Requesting axes	p. [▶ 302]
#CAX	Requesting a spindle axis for C axis machining	p. [▶ 617]
#CAX OFF	Returning C axis to the spindle	p. [▶ 617]
#CAXTRACK ON/OFF	Automatic axis tracking	p. [▶ 393]
#CHANNEL INIT	Initialising channel with current command/actual positions	
	• Initialising channel with command positions	p. [▶ 172]
	• Initialising channel with actual positions	p. [▶ 173]
#CHANNEL INTERFACE ON/ OFF	Dynamic CS via channel interface	[FCT-C30]
#CHANNEL SET	Setting function-specific parameters in the channel	
	• Feed programming for microjoints	[FCT-C1]
	• Time offset with estimation	[FCT-C34]
#CHF	Inserting chamfers and roundings: chamfer length	p. [▶ 150]
#CHR	Inserting chamfers and roundings: chamfer width	p. [▶ 150]
#CLEAR CONFIG	Deleting a saved configuration	p [▶ 300].
#COMMAND WR	Non-synchronised write of SERCOS commands	p. [▶ 362]
#COMMAND WR SYN	Synchronised write of SERCOS commands	p. [▶ 363]
#COMMAND WAIT	Non-synchronised wait for SERCOS commands	p. [▶ 364]

S

23.1.6   Additional axis-specific functions (<X>[..])

23.1.7   PLC-Open functions (<X>[MC_..])

23.1.8   Variable programming (V.)

23.1.9   Miscellaneous functions

Various calculation operations and functions are provided to program mathematic expressions [▶ 32] (e.g. SIN, COS, MOD, ABS, OR..) and to process strings [▶ 38] (e.g. LEFT, MID, INSERT..).

23.1.10 Migrated NC commands

ℹ The following table lists commands which were transferred to a new NC syntax as a result of functional advanced developments or for syntax reasons.

Commands previously used may still be used for programming (downwards compatibility) but they should no longer be used to generate new NC programs.

Old syntax:	New syntax	from version
#SET DEC LR SOLL	#CHANNEL INIT [...] [▶ 172]	V2.10.1504.00
#VECTORVEL ON / OFF	#VECTOR LIMIT ON / OFF [...] [▶ 406]	V2.10.1507.02
#VECTORACC ON / OFF	#VECTOR LIMIT ON / OFF [...] [▶ 406]	V2.10.1507.02
#INIT MAKRO TAB	#INIT MACRO TAB [▶ 672]	V2.11.2010.02
G200 #ACHSE [...]	G200 X.. Y.. [▶ 162]	V2.11.2010.02
G201 #ACHSE [...]	G201 X.. Y.. [▶ 160]	V2.11.2010.02
G202 #ACHSE [...]	G202 X.. Y.. [▶ 160]	V2.11.2010.02
#GET IPO OFFSET	#GET MANUAL OFFSETS [▶ 171]	V2.11.2010.02
#SET OFFSET [...] X	#MANUAL LIMITS [...] [▶ 166]	V2.11.2010.02
#SET HR [...] X	#HANDWHEEL [...] [▶ 163]	V2.11.2010.02
#SET TIP [...] X	#JOG CONT [...] [▶ 164]	V2.11.2010.02
#SET JOG [...] X	#JOG INCR [...] [▶ 165]	V2.11.2010.02
#SET IPO SOLLPOS [...]	#GET CMDPOS [...] [▶ 174]	V2.11.2010.02
#SET SLOPE PROFIL [...]	#SLOPE ... [...] [▶ 357]	V2.11.2010.02
#SET ASPLINE STARTTANG X.. Y..	#AKIMA STARTVECTOR X.. Y.. [▶ 286]	V2.11.2010.02
#SET ASPLINE ZIELTANG X.. Y..	#AKIMA ENDVECTOR X.. Y.. [▶ 287]	V2.11.2010.02
#SET ASPLINE MODE [...]	#AKIMA TRANS [...] [▶ 286]	V2.11.2010.02
#SET CORNER PARAM [...]	#CORNER PARAM [...] [▶ 124]	V2.11.2010.02
#SET TANGFEED RMIN [...]	#TANGFEED [...] [▶ 328]	V2.11.2010.02
#SET SPLINE ON / OFF	#SPLINE ON [▶ 284] / OFF [▶ 284]	V2.11.2010.02
#SET SPLINETYPE AKIMA	#SPLINE TYPE AKIMA [▶ 284]	V2.11.2010.02
#SET SPLINETYPE BSPLINE	#SPLINE TYPE BSPLINE [▶ 290]	V2.11.2010.02
#RTCP ON / OFF	#TRAFO ON / OFF [▶ 677]	all versions

24   Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

  • support
  • design, programming and commissioning of complex automation systems
  • and extensive training program for Beckhoff system components

Hotline: +49 5246 963-157
e-mail: support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

  • on-site service
  • repair service
  • spare parts service
  • hotline service

Hotline: +49 5246 963-460
e-mail: service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963-0
e-mail: info@beckhoff.com
web: www.beckhoff.com

Index

More Information:
www.beckhoff.com/TF5200