

# **SINUMERIK 810M/820 M**

## **Basic Version 3, Software Version 3**

### **Part 2: Programming**

**User Documentation**

Fundamentals of Programming	1
Directions of Movement, Dimensional Notation	2
Programming of Motion Blocks	3
Miscellaneous, Switching and Auxiliary Functions	4
Subroutines	5
Parameters	6
Contour Definition	7
Tool Offsets	8
Cutter Radius Compensation (CRC)	9
Cycles	10
Programming of Cycles	11
SINUMERIK 810M/820M Program Key	12

# Contents

	Page
<b>1 Fundamentals of Programming</b>	<b>1-1</b>
1.1 Program structure	1-1
1.2 Block format	1-2
1.3 Block elements	1-3
1.3.1 Main blocks and subblocks	1-3
1.3.2 Skippable blocks	1-3
1.3.3 Remarks (Comments)	1-4
1.4 Word format	1-4
1.5 Character set	1-7
1.6 Tapes	1-7
1.6.1 Tape reader	1-7
1.6.2 Tape code	1-8
1.6.3 Leader	1-8
1.6.4 Read-in stop	1-8
1.7 Program format for input/output	1-9
1.8 Code table	1-13
1.9 Input/output formats	1-16
1.10 Revolutionary feedrate limit data	1-19
1.11 Channel structure	1-20
<b>2 Directions of Movement, Dimensional Notation</b>	<b>2-1</b>
2.1 Coordinate system	2-1
2.1.1 Flexible plane selection	2-2
2.2 Position data, preparatory functions	2-4
2.3 Dimension systems: absolute and incremental position data G90/G68/G91	2-4
2.4 Reference points	2-6
2.5 Zero offset	2-7
2.6 Path calculation	2-11
2.7 Workpiece dimensioning, input system G70/G71	2-12
2.8 Mirroring	2-13
2.9 Programmable working area limitation G25/G26	2-16
2.10 Software cam	2-18
2.11 Coordinate rotation	2-21
2.12 Scale modification: Selection G51, cancellation G50	2-23
<b>3 Programming of Motion Blocks</b>	<b>3-1</b>
3.1 Axis commands	3-1
3.1.1 Axis motion without machining G00	3-3
3.1.2 Axis synchronization	3-4
3.1.2.1 Function	3-4
3.1.2.2 Complete machining	3-7

3.2	Axis motions with machining .....	3-8
3.2.1	Linear interpolation G01 .....	3-8
3.2.2	Circular interpolation G02/G03 .....	3-12
3.2.2.1	Interpolation parameters I, J, K .....	3-13
3.2.2.2	Radius programming .....	3-15
3.2.3	Helical interpolation .....	3-16
3.2.4	Cylindrical interpolation .....	3-18
3.2.5	Polar coordinates G10/G11/G12/G13 .....	3-20
3.2.5.1	Polar coordinates G110/G111 .....	3-23
3.2.6	Feedrate F, G94/G95/G98 .....	3-24
3.2.7	Thread cutting G33/G34/G35 .....	3-25
3.2.7.1	Thread with constant lead .....	3-26
3.2.7.2	Thread with variable lead .....	3-26
3.2.7.3	Infeed options .....	3-26
3.2.7.4	Multiple threads .....	3-27
3.2.8	Tapping without encoder G63 .....	3-28
3.2.9	Tapping without G36 compensating chuck .....	3-28
3.2.10	Exact positioning G09/G60/G00, continuous path operation G62/G64 ..	3-30
3.2.10.1	Fine and coarse exact stop tolerance ranges G09/G60/G00 .....	3-30
3.2.10.2	Continuous path operation G62/G64 .....	3-32
3.2.11	Dwell G04 .....	3-33
3.2.12	Soft approach to and retraction from the contour .....	3-34
3.2.13	SPLINE interpolation G06 .....	3-37
3.2.14	Coordinate transformation TRANSMIT .....	3-38
3.2.14.1	TRANSMIT function .....	3-39
3.2.14.2	Block search with calculation and the TRANSMIT function .....	3-41
3.2.14.3	Principle of the TRANSMIT coordinate transformation .....	3-42
3.2.14.4	Machining accuracy with TRANSMIT .....	3-46
3.2.14.5	Velocity monitoring with TRANSMIT .....	3-47
3.3	Reference point approach in part program (G74) .....	3-49
3.3.1	Function description .....	3-49
3.3.2	Starting the function .....	3-49
3.4	Second spindle .....	3-50
3.5	On-the-fly synchronization of rotary axis .....	3-51
3.6	Overstore .....	3-52
<b>4</b>	<b>Miscellaneous, Switching and Auxiliary Functions .....</b>	<b>4-1</b>
4.1	M, S, T, H .....	4-1
4.2	Miscellaneous functions M .....	4-1
4.3	Spindle function S .....	4-3
4.4	Auxiliary functions H .....	4-4
4.5	Tool number T .....	4-4
<b>5</b>	<b>Subroutines .....</b>	<b>5-1</b>
5.1	Application .....	5-1
5.2	Subroutine structure .....	5-1
5.3	Subroutine call .....	5-2
5.4	Subroutine nesting .....	5-3

<b>6</b>	<b>Parameters</b>	<b>6-1</b>
6.1	Parameter programming	6-1
6.2	Parameter definition	6-2
6.3	Parameter calculation	6-3
6.4	Parameter string	6-4
6.5	Programming examples with R parameters	6-5
<b>7</b>	<b>Contour Definition</b>	<b>7-1</b>
7.1	Blueprint programming	7-1
7.2	Contour definition programming	7-2
7.3	Operation of function G09, F, S, T, H, M in contour definition	7-6
7.4	Linking of blocks	7-6
7.5	Programming examples: milling machine	7-6
7.6	Miscellaneous functions in linked blocks	7-8
<b>8</b>	<b>Tool Offsets</b>	<b>8-1</b>
8.1	Tool data	8-1
8.2	Selection and cancellation of length compensation	8-2
8.3	G40/G41/G42 intersection cutter radius compensation	8-2
8.4	Tool length compensation, positive or negative	8-8
8.5	Tool offsets for end mill	8-9
8.6	Tool offsets for angle head cutter	8-10
<b>9</b>	<b>Cutter Radius Compensation (CRC)</b>	<b>9-1</b>
9.1	Selection of CRC	9-1
9.2	CRC in the program	9-4
9.3	Cancellation of CRC (G40)	9-7
9.4	Changing direction of compensation (G41, G42)	9-9
9.5	Changing offset number (G41 D., G41 D.)	9-9
9.6	Changing compensation values (R1, R2)	9-10
9.7	Repetition of selected G function (G41, G42) with same offset number	9-10
9.8	M00, M01, M02 and M30 with CRC selected	9-11
9.9	CRC with combination of various block types and in conjunction with contour errors	9-13
9.10	Special cases for CRC	9-17
9.11	Effect with negative compensation values	9-21
<b>10</b>	<b>Cycles</b>	<b>10-1</b>
<b>11</b>	<b>Programming of Cycles</b>	<b>11-1</b>
11.1	General	11-1
11.2	Destination code	11-1
11.2.1	Main groups	11-1
11.2.2	Operands after the destination code	11-2
11.2.3	Notation	11-2

11.3	General statements for program structure .....	11-3
11.4	Program branchings .....	11-4
11.5	Data transfer, general .....	11-10
11.6	Data transfer, system memory to R parameters .....	11-11
11.7	Data transfer, R parameters to system memory .....	11-18
11.8	Mathematical functions .....	11-24
11.9	NC-specific functions .....	11-30
11.10	I/O functions .....	11-38
11.11	Operator guidance macro (OGM) .....	11-43
11.12	@ Code table .....	11-45
<b>12</b>	<b>SINUMERIK 810M/820M Program Key .....</b>	<b>12-1</b>
12.1	Internal G groups for @36b .....	12-1
12.2	Program key .....	12-2

# 1 Fundamentals of Programming

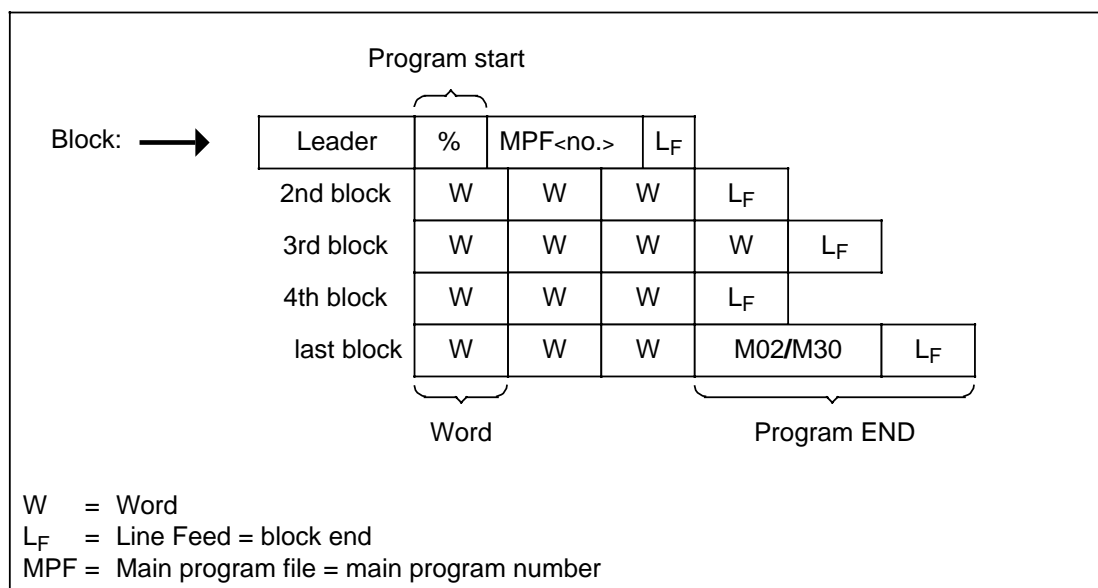
## 1.1 Program structure

The program structure is based on DIN 66025. A part program comprises a complete string of blocks which define the sequence of operations of a machining process on a numerically controlled machine tool.

A part program comprises:

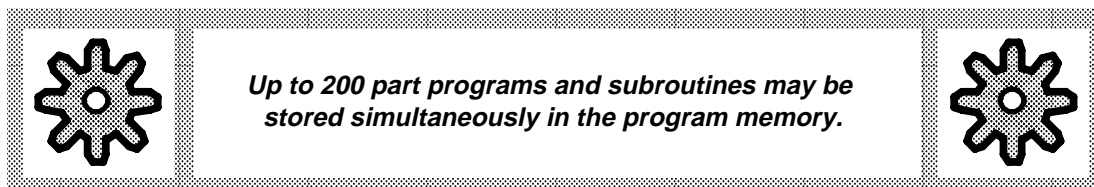
- The character for program start
- A number of blocks
- The character for program end.

The character for program start precedes the first block in the part program. The character for part program end is contained in the last block of the part program.



Program structure: Part program in input/output format

Subroutines and cycles may be components of the program. Cycles are subroutines which have been created either by the machine manufacturer or by Siemens. They can be specially protected against misuse.



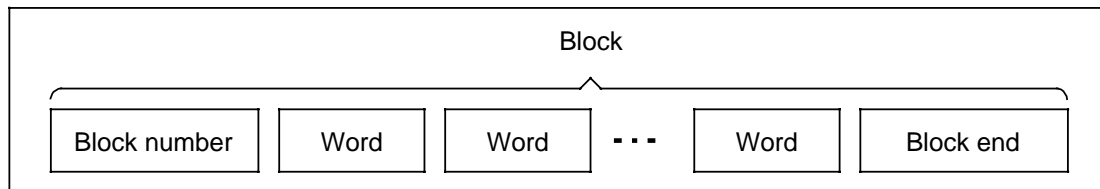
The input sequence is arbitrary. A total of:  
0 - 9999 machining programs and  
1 - 9999 subroutines  
are available for part programs.

If the program is entered by means of operator prompting via the operator panel, when the "BLOCK NUMBER" softkey is actuated, the block numbers are generated automatically in steps of five. The "Cancel" key can be used to delete the entered block number; the "Edit" key can be used to overwrite it.

## 1.2 Block format

A block contains all data required to implement an operating procedure. The block comprises several words and the "block end" character "LF".

The block length is max. 120 characters. The block is displayed in its entirety over several lines.



*Block structure: Block format*

The block number is entered under address N or with "N". Block numbers are freely selectable. A defined block search and defined jump functions can only be guaranteed if a block number is used no more than once in a program.

Programming without a block number is permissible. In this case, however, no block search or jump functions will be possible.

The block format should be made as simple as possible by arranging the words of a block in the program key sequence.

### Block example:

```
N9235 G.. X.. Z.. F.. S.. T.. M.. H.. LF
```

N	Address of block number
9235	Block number
G..	Preparatory function
X.. Z..	Position data
F..	Feedrate
S..	Spindle speed
T..	Tool number
M..	Auxiliary function
H..	Auxiliary function
LF	Block end

If the value for an address letter is programmed more than once, the last value to have been programmed applies.

Each block need to be terminated with the "LF" end-of-block character. This character appears on the screen as the special character "LF". When the program is printed out, this character does not appear.



## 1.3 Block elements

### 1.3.1 Main blocks and subblocks

There are two types of blocks: Main blocks and subblocks.

The **main block** must contain all **words** required to start the machining cycle in the program section beginning there. A main block may only be located in the part program (main program). A main block may be identified by means of the “:” **character** instead of address character “N” for the subblock. A main block and a subblock must **not** have the same block no. in one program.

#### Block example:

```
:10 G1 X10 Y-15 F200 S1000 M03 L_F
```

A subblock contains only those functions which differ from the functions in the previous block.

#### Block example:

```
N15 Y20 L_F
```

A main block and several subblocks together form a program section.

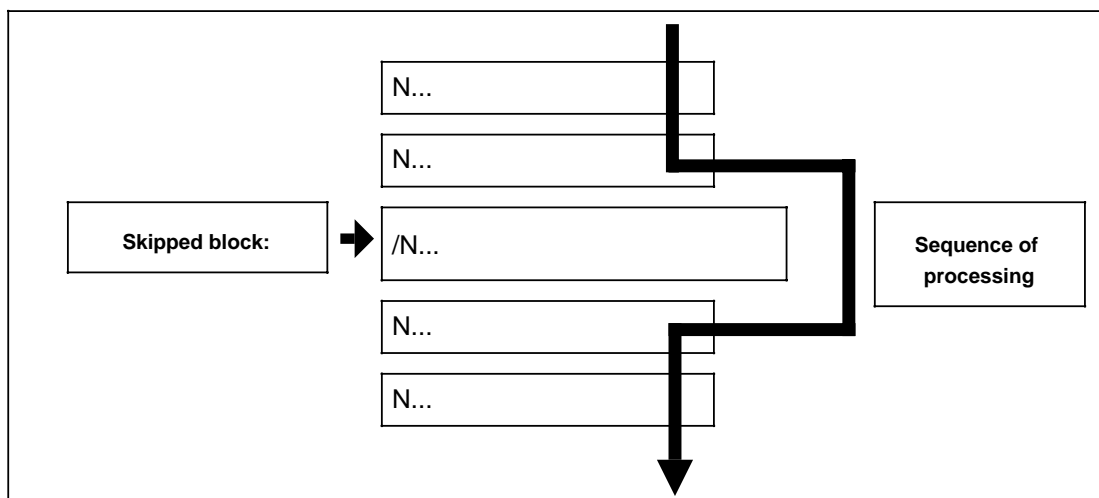
**Example:**

```
:10
N105
N110
N115
```

} Section

### 1.3.2 Skippable blocks

Program blocks which must not be executed during every program run can be skipped by entering the slash character “/” in front of the word with the block number. Skipping blocks is activated via the “SKIP YES-NO” softkey or via the interface controller. The skipped blocks must form a loop (with start and end at the same point) or the program may be executed incorrectly. A section can be skipped by skipping several consecutive blocks.



*Skipping blocks*

***In order to ensure faster block change times, several blocks must be buffered. If the machine stops on account of M00 (programmed stop), the next blocks will already have been read in advance. The "SKIP" function is only active on those blocks which have not been buffered. This buffering can be prevented by programming L999 (disable pre-reading @ 714) after the block containing M00.***

### 1.3.3 Remarks (Comments)

The blocks in a program can be explained by means of remarks. A remark permits instructions for the operator to be displayed on the screen. The text of a remark is enclosed between the start-of-remark character "(" and the end-of-remark character ")".

The remark must not contain the percent sign %, an end-of-block character **LF**, or bracket "(", ")".

A remark may be up to **120 characters in length**. Up to **41** of these may be **displayed** in the comment line of the **screen**.

***It is advisable to write the remark at the end of the block or in a separate line. The remark must never be located between the address and a digit or between a word and the corresponding parameter!***

#### Right:

```
N05      G00  X100 Y200 ( Position) LF
N10      G01  X100+R1 Y200 ( Machine ) LF
N15      ....
```

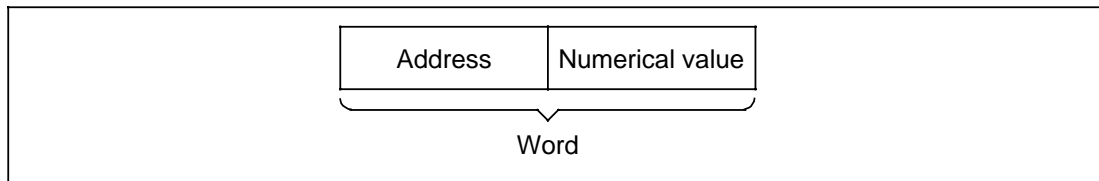
x	Address
100	Value
R1	R parameter
(	Start of remark (blank)
Machine	Remark (blank)
)	End of remark.

#### Wrong:

```
N05      X   ( Position ) 100 Y200 LF
N10      X100+   ( Machine ) R1 Y200 LF
```

## 1.4 Word format

A word is an element of a block. It comprises an address character and a string of digits. The address character is normally a letter. The string of digits may be specified with a sign and with decimal points. The sign is written between the address letters and the string of digits. A positive sign may be omitted.

*Word format***Examples:**

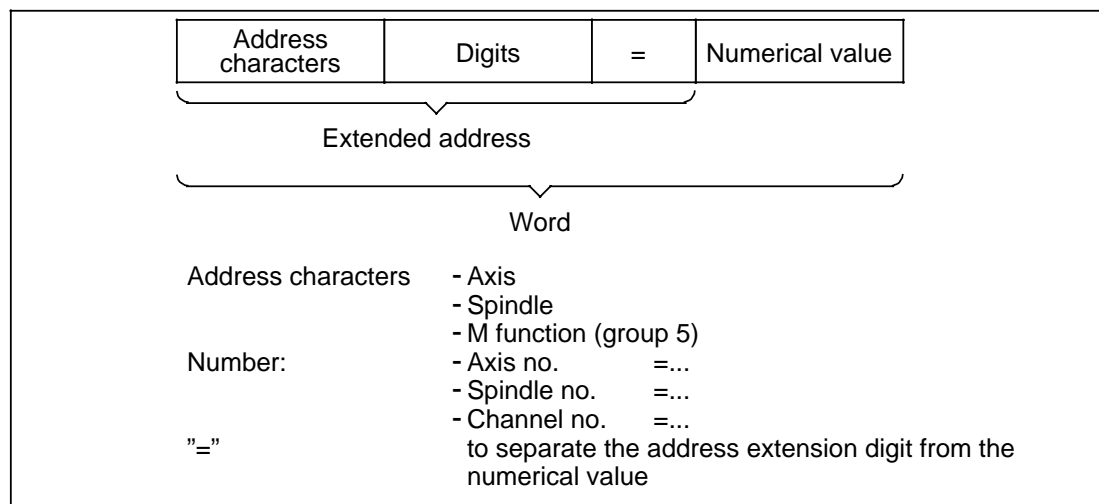
G91 OR M30

G Address

91 Numerical value, signifying: "Incremental dimensions"

M Address

30 Numerical value, signifying: "Program end".

**Extended address***Extended address***Example:**

Q1= 100

1st auxiliary axis

M1= 19

Oriented spindle stop of 1st spindle

M2= 100

M function 100 is output for channel 2

**Note:**G1 X1 = Z L<sub>F</sub>means: G1 X1=0 Z0 L<sub>F</sub>G16 X Y1 = Z ±L<sub>F</sub>means: G16 X0 Y1 = 0 Z0 ± L<sub>F</sub>

Plane selection with extended address (Y1)

"±" specifies the direction of the tool offset and must be placed after the axis.

"=-" must be written to define the figure after the address as an address extension.

The word format is based on DIN 66025.

Shorthand notation of words:

%4 N04 G02/G03 D03 XL+053 YL+053 ZL+053 AL053 ID053 JD053 KD053 F05  
L03/L04 S05 T08 R03 RL+053 BD033 M04 H08 P02 L<sub>F</sub>

Definitions:

First letter	Address	
Second letter	L	Absolute/incremental
Second letter	D	Incremental
Character	±	Absolute dimensions with positive or negative sign
First digit	0	Leading zeros may be omitted: Variable word length (G01 = G1)
Second digit	Decades	Decades Positions in digit string
Second/third digits	Decades	Digit string positions before and after decimal point (coordinate values X, Y, Z, I, J, K in mm)
Character	L <sub>F</sub>	Block end

**Example: XL+053**

<b>X</b>	Address
<b>L</b>	Absolute/incremental
<b>+</b>	Sign
<b>0</b>	Leading zeros may be omitted
<b>5</b>	Number of positions before decimal point
<b>3</b>	Number of positions after decimal point

**Word examples: X-12345.531**

**G9**

<b>X</b>	Address	<b>G</b>	Address
<b>-</b>	Sign	<b>9</b>	Digit
<b>12345</b>	Digits		
<b>.</b>	Decimal point		
<b>531</b>	Digits		

**Decimal point input:**

Value	Programmed value with decimal point
0.1 μm	X.0001
1 μm	X.001
10 μm	X.01
100 μm	X.1
1000 μm	X1 oder X1.
10200 μm	X10.2

Decimal point input is permissible for the following addresses:  
X, Y, Z, E, A, B, C, U, V, W, Q, I, J, K, R, F, S.

For address "R" only the notation with an extended address is valid: R10 = 50.0  
(see Section 12 Program key for restrictions on S).  
Leading and trailing zeros need not be written when decimal point notation is used.

## 1.5 Character set

It is always possible to choose between two codes for programming:

- DIN 66025 (**ISO**) or
- **EIA**-RS 244-B.

The **examples** used in these Instructions are based on the **ISO code**.

The following characters are available in ISO code for formulating program, geometric and process statements:

*Address letters:*

A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T, U, V, W, X, Y, Z

*Lower-case letters*

a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

*Digits*

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

The 5th axis must be entered with extended address, e. g. Q1 = 5.

*Hexadecimal digits with CL 800 machine code,*

a, b, c, d, e, f (CL 800 Configuring Instructions)

*Letter*

D (Input of tool offset [TO - TOOL OFFSET])

*Printable special characters*

%, (, ), +, -, /, :, ., =, \*, @

### Data input

The following characters are not processed or stored:

HT = Horizontal tabulator  
SP = Space  
DEL = Delete character  
CR = Carriage return

Other control characters are shown in the code table.

### Data output

The following characters are generated:

SP (after every word)

CR generated twice after L<sub>F</sub> or once before L<sub>F</sub> (setting data).

## 1.6 Tapes

### 1.6.1 Tape reader

The tape reader must be matched to the controller. The data transfer rate and the transfer format are defined via the setting data (see Universal Interface).

## 1.6.2 Tape code

The data on tape is coded according to fixed rules, i.e. each hole combination corresponds to a particular character. **Two tape codes** are used: ISO or EIA (see code table).

All characters of a code have a common identification:

- **ISO** always an **even number** of holes
- **EIA** always an **odd number** of holes.

The controller automatically recognizes the correct code as soon as it reads the first % (ISO) or EOR (EIA). The criterion relating to an odd or even number of holes is used - starting at the second character of the program - for a character parity check, which has an error detection rate for single errors of 100%. Each tape must be written in one of the permissible codes. It is not permissible to change the code within a tape or to splice tapes together; this will cause the character parity check to be initiated.

As a further check a complete program comparison is performed if a program already stored in the program memory is read in again. On detection of an error the read-in process is halted and the error displayed on the CRT display unit.

## 1.6.3 Leader

The leader is used to identify the programs. The tape leader may include all characters except the start-of-program character (% character). The leader is not stored, and is ignored by the controller during program processing.

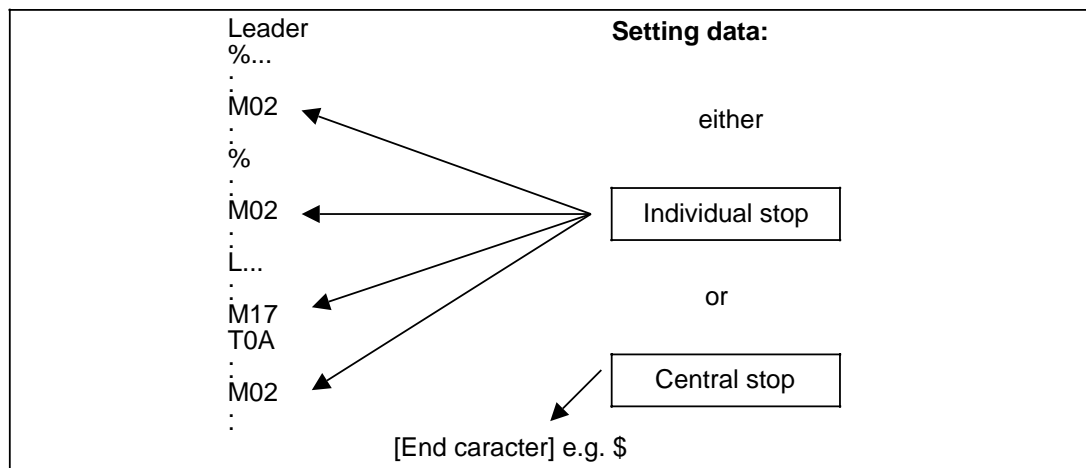
POCKET	%	MPF	1579	L <sub>F</sub>
--------	---	-----	------	----------------

## 1.6.4 Read-in stop

The read-in process is halted by M02, M30 or M17 if no central end-of-transmission character has been defined.

If an end-of-transmission character has been specified in the setting data, the program or data block end (M02, M17, M30) will not stop the reader during reading in of the tape.

The read-in process is not halted until the end-of-transmission character is reached.



**1.7 Program format for input/output**

<b>Program</b>	<b>Leader</b>
%MPF1235 L <sub>F</sub>	Part program 1235 ( <b>MAIN PROGRAM FILE</b> )
(Perform measurem.) N...L <sub>F</sub>	Remark Part program
N...L <sub>F</sub>	
M02 L <sub>F</sub> or M30 L <sub>F</sub>	Part program end

<b>Subroutines</b>	<b>Leader</b>
%SPF234 L <sub>F</sub>	Subroutine 234 ( <b>SUB PROGRAM FILE</b> )
N5...L <sub>F</sub>	Subroutine
N10...L <sub>F</sub>	
(Bore cycle)...L <sub>F</sub>	Remark
M17 L <sub>F</sub>	Subroutine end

<b>%ZOA L<sub>F</sub></b>	<b>Settable zero offsets (ZERO OFFSET ACTIVE)</b>
G154 X=... Y=... L <sub>F</sub> : G157 X=... Y=... L <sub>F</sub>	1st to 4th settable offset (coarse)
G254 X=... Y=... L <sub>F</sub> : G257 X=... Y=... L <sub>F</sub>	
M02 L <sub>F</sub> or M30 L <sub>F</sub>	Zero offset block data block end

<b>%TEA1 L<sub>F</sub></b>	<b>NC machine data (TESTING DATA ACTIVE 1)</b>
N...=...L <sub>F</sub> N...=...L <sub>F</sub>	Machine data
M02 L <sub>F</sub> or M30 L <sub>F</sub>	

<b>%TEA2 L<sub>F</sub></b>	<b>PLC machine data (TESTING DATA ACTIVE 2)</b>
N...=...L <sub>F</sub> N...=...L <sub>F</sub>	Machine data
M02 L <sub>F</sub> or M30 L <sub>F</sub>	Machine data block      data block end

<b>%SEA L<sub>F</sub></b>	<b>(SETTING DATA ACTIVE) (general setting data)</b>
N...=...L <sub>F</sub> N...=...L <sub>F</sub>	Address with value assignment      (0...9, 2000...2003, 3000...3171, 4000...4033, 5000...5771)
M02 L <sub>F</sub> or M30 L <sub>F</sub>	Setting data data block end

<b>%RPA0..2 L<sub>F</sub></b>	<b>(R PARAMETER ACTIVE ) Channel No. ( 0 = central R parameters)</b>
R...=...L <sub>F</sub> R...=...L <sub>F</sub>	Parameter numbers with value assignments (channel 1 and 2, channelspecific)
M02 L <sub>F</sub> or M30 L <sub>F</sub>	R parameter      data block end

<b>%TOA L<sub>F</sub></b>	<b>Tool offsets (TOOL OFFSET ACTIVE )</b> TO range (MD)
D1 P0=...P1=...P9=...L <sub>F</sub> D2 P0=...P1=...L <sub>F</sub>	Tool offsets (number of parameters in acc. with MD)
M02 L <sub>F</sub> or M30 L <sub>F</sub>	Tool offset block      data block end



<b>%PCA L<sub>F</sub></b>	<b>PLC alarm texts/operator messages</b> <b>(PROGRAMMABLE LOGIC CONTROL ALARM)</b>
N6000 (Text ...) L <sub>F</sub> : N6063 (Text ...) L <sub>F</sub>	PLC alarm texts (6000 - 6063) <div style="display: inline-block; vertical-align: middle; margin-left: 20px;">             } Text length: max. 36 ASCII characters (‘CR’ and ‘L<sub>F</sub>’ not allowed)           </div>
N7000 (Text ...) L <sub>F</sub> : N7063 (Text ...) L <sub>F</sub>	
M02 L <sub>F</sub> or M30 L <sub>F</sub>	PLC text data block end

<b>%PCP L<sub>F</sub></b>	<b>PLC program</b> <b>(PROGRAMMABLE LOGIC CONTROL PROGRAM)</b>
7070 8005. . .	Machine code
M02 L <sub>F</sub> or M30 L <sub>F</sub>	PLC program data block end

<b>%UMS L<sub>F</sub></b>	<b>User Memory Submodule</b>
:Hexcode	Configured data
:Hexcode	Configured data
M02 L <sub>F</sub> or M30 L <sub>F</sub>	User memory submodule data block end

## Memory areas:

The memory areas of the SINUMERIK 810M controller are addressed by means of the following **identifiers**:

Identifier	Meaning
MPF	Part program ( <b>M</b> ain <b>P</b> rogram <b>F</b> ile)
SPF	Subroutine ( <b>S</b> ub <b>P</b> rogram <b>F</b> ile)
TOA	Tool offsets ( <b>T</b> ool <b>O</b> ffset <b>A</b> ctive)
ZOA	Zero offsets ( <b>Z</b> ero <b>O</b> ffset <b>A</b> ctive)
TEA1	NC machine data ( <b>T</b> esting <b>D</b> ata <b>A</b> ctive <b>1</b> )
TEA2	PLC machine data ( <b>T</b> esting <b>D</b> ata <b>A</b> ctive <b>2</b> )
PCA	<b>PLC Alarm Texts</b>
PCP	PLC program (machine code) ( <b>P</b> rogrammable <b>C</b> ontrol <b>P</b> rogram)
RPA	R parameter numbers with value assignments ( <b>R</b> Parameter <b>A</b> ctive)
SEA	Addresses with value assignments ( <b>S</b> etting <b>D</b> ata <b>A</b> ctive)
CLF	Clear statement ( <b>C</b> lear <b>F</b> ile)
UMS	<b>U</b> ser <b>m</b> emory <b>s</b> ubmodule

- Deleting programs:**

These functions permit part programs and subroutines to be deleted in any sequence via the input/output interface.

DELETE PROGRAM	Leader
%CLF L <sub>F</sub>	<b>Delete program identifier (CLEAR FILE)</b> Setting data (SD for serial interface) can be used to determine whether automatic reorganization of the part program memory is to be prevented with %CLF.
MPF1234 L <sub>F</sub>	<b>Delete part program % 1234</b>
MPF 1, 1200 L <sub>F</sub>	Delete part program %1 to % 1200
MPF 0, 9999 L <sub>F</sub>	Delete all part programs
SPF 10 L <sub>F</sub>	<b>Delete subroutine L10</b>
SPF 11, 79 L <sub>F</sub>	Delete subroutines L11 to L79
SPF 1, 9999 L <sub>F</sub>	Delete all subroutines
M30, M02 or M17 L <sub>F</sub>	End identifier M30, M02 or M17

- Deleting text data:**

%PCA L<sub>F</sub>  
M02 or M30 L<sub>F</sub>

## 1.8 Code table

ISO/DIN 66024 extended										
Char-acter	Hole combination									Only leader and remark
	P	7	6	5	4	T	3	2	1	
NUL						.				Control characters not stored
SOH	•					.			•	
STX	•					.		•		
ETX						.		•	•	
EOT	•					.	•			
ENQ						.	•		•	
ACK						.	•	•		
BEL	•					.	•	•	•	
BS	•				•	.				
HT					•	.			•	
VT	•				•	.		•	•	
FF					•	.	•			
CR	•				•	.	•		•	
SO	•				•	.	•	•		
SI					•	.	•	•	•	
DLE	•			•		.				
DC1				•		.			•	
DC2				•		.		•		
DC3	•			•		.		•	•	
DC4				•		.	•			
NAK	•			•		.	•		•	
SYN	•			•		.	•	•		
ETB				•		.	•	•	•	
CAN				•	•	.				
EM	•			•	•	.			•	
SUB	•			•	•	.		•		
ESC				•	•	.		•	•	
FS	•			•	•	.	•			
GS				•	•	.	•		•	
RS				•	•	.	•	•		
US	•			•	•	.	•	•	•	
SP	•		•			.				
LF					•	.		•		
!			•			.			•	×
”			•			.		•		×
	•		•			.		•	•	×
\$			•			.	•			×
%	•		•			.	•		•	
&	•		•			.	•	•		×
,			•			.	•	•	•	×
(			•		•	.				
)	•		•		•	.			•	
*	•		•		•	.		•		×
+			•		•	.		•	•	
,	•		•		•	.	•			×
-			•		•	.	•		•	
•			•		•	.	•	•		
/	•		•		•	.	•	•	•	

ISO/DIN 66024 extended										
Char- acter	Hole combination									Only leader and remark
	P	7	6	5	4	T	3	2	1	
0			•	•		•				
1	•		•	•		•			•	
2	•		•	•		•		•		
3			•	•		•		•	•	
4	•		•	•		•	•			
5			•	•		•	•		•	
6			•	•		•	•	•		
7	•		•	•		•	•	•	•	
8	•		•	•	•	•				
9			•	•	•	•			•	
:			•	•	•	•		•		
;	•		•	•	•	•		•	•	x
<			•	•	•	•	•			x
=	•		•	•	•	•	•		•	
>	•		•	•	•	•	•	•		x
?			•	•	•	•	•	•	•	x
@	•	•				•				
A		•				•			•	
B		•				•		•		
C	•	•				•		•	•	
D		•				•	•			
E	•	•				•	•		•	
F	•	•				•	•	•		
G		•				•	•	•	•	
H		•			•	•				
I	•	•			•	•			•	
J	•	•			•	•		•		
K		•			•	•		•	•	
L	•	•				•	•			
M		•				•	•		•	
N		•			•	•	•	•		
O	•	•			•	•	•	•	•	
P		•		•		•				
Q	•	•		•		•			•	
R	•	•		•		•		•		
S		•		•		•		•	•	
T	•	•		•		•	•			
U		•		•		•	•		•	
V		•		•		•	•	•		
W	•	•		•		•	•	•	•	
X	•	•		•	•	•				
Y		•		•	•	•			•	
Z		•		•	•	•		•		
[	•	•		•	•	•		•	•	x
\		•		•	•	•	•			x
]	•	•		•	•	•	•		•	x
^	•	•		•	•	•	•	•		x
-		•		•	•	•	•	•	•	x

ISO/DIN 66024 extended										
Char- acter	Hole combination									Only leader and remark
	P	7	6	5	4	T	3	2	1	
`		•	•			•				x
a	•	•	•			•			•	
b	•	•	•			•		•		
c		•	•			•		•	•	
d	•	•	•			•	•			
e		•	•			•	•		•	
f		•	•			•	•	•		
g	•	•	•			•	•	•	•	x
h	•	•	•		•	•				x
i		•	•		•	•			•	x
j		•	•		•	•		•		x
k	•	•	•		•	•		•	•	x
l		•	•		•	•	•			x
m	•	•	•		•	•	•		•	x
n	•	•	•		•	•	•	•		x
o		•	•		•	•	•	•	•	x
p	•	•	•	•		•				x
q		•	•	•		•			•	x
r		•	•	•		•		•		x
s	•	•	•	•		•		•	•	x
t		•	•	•		•	•			x
u	•	•	•	•		•	•		•	x
v	•	•	•	•		•	•	•		x
w		•	•	•		•	•	•	•	x
x		•	•	•	•	•				x
y	•	•	•	•	•	•			•	x
z	•	•	•	•	•	•		•		x
{		•	•	•	•	•		•	•	x
:	•	•	•	•	•	•	•			x
}		•	•	•	•	•	•		•	x
		•	•	•	•	•	•	•		
DEL	•	•	•	•	•	•	•	•	•	
										% not allow- ed in tape leader

EIA/ 244B									
Char- acter	Hole combination								Only leader and remark
	P	7	6	5	4	T	3	2	1
no hole						•			x
RT			•		•	•		•	x
TAB			•	•	•	•	•	•	x
<=EOB	•					•			
LC)		•	•	•	•	•		•	
ZWR				•		•			
(				•	•	•		•	
)		•			•	•		•	
EOR					•	•		•	•
UC		•	•	•	•	•	•		
%					•	•		•	•
&					•	•	•	•	
>			•	•	•	•		•	•
@		•	•		•	•	•	•	
:		•				•	•	•	
•		•	•		•	•		•	•
/			•	•		•			•
+		•	•	•		•			
-		•				•			
0			•						
1						•			•
2						•		•	
3				•		•		•	•
4						•	•		
5				•		•	•		•
6				•		•	•	•	
7						•	•	•	•
8					•	•			
9				•	•	•			•
a		•	•			•			•
b		•	•			•		•	
c		•	•	•		•		•	•
d		•	•			•	•		
e		•	•	•		•	•		•
f		•	•	•		•	•	•	
g		•	•			•	•	•	•
h		•	•		•	•			
i		•	•	•	•	•			•
j		•		•		•			•
k		•		•		•		•	
l		•				•		•	•
m		•		•		•	•		
n		•				•	•		•
o		•				•	•	•	
p		•		•		•	•	•	•
q		•		•	•	•			
r		•			•	•			•
s			•	•		•		•	

EIA/ 244B									
Char- acter	Hole combination								Only leader and remark
	P	7	6	5	4	T	3	2	1
t			•			•		•	•
u			•	•		•	•		
v			•			•	•		•
w			•			•	•	•	
x			•	•		•	•	•	•
y			•	•	•	•			
z			•		•	•			•
IRR		•	•	•	•	•	•	•	

Not all ISO characters can be represented in EIA code. Consequently, discrepancies may occur when comparing a program generated in ISO code and stored in the NC with its equivalent program converted to EIA code.

The following functions are no longer capable of operating when read into the SINUMERIK controller once more:

- Parameter calculation
- Extended address
- @ commands with HEX digits (@ 36 a)
- Special characters
- Comments.

The EIA code for "@" and ":" can be set in setting data ( see Section 6.2 "Setting data for description of the interfaces" of Part 1 "Operating").



## 1.9 Input/output formats

The input/output formats depend on the machine manufacturer's machine data setting.

Input resolution: 0.01 mm or position control resolution 0.005 mm  
0.001 inch 0.0005 inch  
0.001 degrees 0.005 degrees

Significance Addresses		Metric		Inch		Degrees	
		Range	Unit	Range	Unit	Range	Unit
Position data (linear axes) Interpolation parameters		±0.01 to 99999.99	mm	±0.001 to 9999.999	inch	-	Degrees
Position data for G91 (rotary axes)		-		-		0.001 to 99999.999	
Position data for G90 (rotary axes)		-		-		±0.001 to 359.999	
Chamfer (U-); radius (U)		0.01 to 999999.99		0.001 to 99999.999		-	
Zero offset		±0.01 to 999999.99		±0.001 to 99999.999		±0.001 to 99999.999	
Thread lead <sup>1)</sup>		0.01 to 4000.00		0.001 to 160.000		-	
Spindle speed S <sup>1)</sup> (value determined via commissioning setting)		1 - 16000	1 rev/min	1 - 16000	1 rev/min		
		0.1 - 1600.0	0.1 rev/min	0.1 - 1600.0	0.1 rev/min		
Linear feedrate (F) (G94) <sup>2)</sup>		0.1 to 450000	mm/min	0.01 to 17700.00	inch/min	1 to 45000	Degrees/ min
Feedrate per revolution (F) (G95)		0.01 to 500.00 <sup>1)</sup>	mm/rev	0.001 to 20.000 <sup>1)</sup>	inch/rev		
Tool offset	Length	±0.01 to 99999.99	mm	±0.001 to 999.999	inch		
	Radius	±0.01 to 9999.99		±0.01 to 999.999			
Dwell	X	0.01 to 99999.999	sec	0.01 to 99999.999	sec		
	F	0.01 to 99999.999		0.01 to 99999.999			
	S	0.1 to 99.9	Revolu- tions	0.1 to 99.9	Revolu- tions		
Angle for polar coordinates		-		-		0 to 359.99999	Degrees
Angle with oriented spindle stop (M19)						0.1 to 359.9	Degrees
R parameters		Dimension depending on association (internal floating point) all combinations					

1) The maximum speed with linear feed (G94) must not be exceeded.

2) The limit values are valid for MD 155 = 2

Input resolution: 0.001 mm or position control resolution 0.0005 mm  
 0.0001 inch 0.00005 inch  
 0.001 degrees 0.0005 degrees

Significance Addresses		Metric		Inch		Degrees	
		Range	Unit	Range	Unit	Range	Unit
Position data (linear axes) Interpolation parameters		±0.001 to 99999.999	mm	±0.0001 to 9999.9999	inch	-	Degrees
Position data for G91 (rotary axes)		-		-		0.001 to 99999.999	
Position data for G90 (rotary axes)		-		-		±0.001 to 359.999	
Chamfer (U-); radius (U)		0.001 to 99999.999		0.0001 to 9999.9999		-	
Zero offset		±0.001 to 99999.999		±0.0001 to 9999.9999		±0.001 to 99999.999	
Thread lead		0.001 to 400.000		0.0001 to 16.0000		-	
Spindle speed S <sup>1)</sup> (value determined via commissioning setting)		1 - 16000	1 rev/min	1 - 16000	1 rev/min		
		0.1 - 1600.0	0.1 rev/min	0.1 to 1600.0	0.1 rev/min		
Linear feedrate (F) (G94) <sup>2)</sup>		0.01 to 45000	mm/min	0.001 to 1770.000	inch/min	1 to 45000	Degrees/ min
Feedrate per revolution (F)(G95)		0.001 to 50.000 <sup>1)</sup>	mm/rev	0.0001 to 2.0000 <sup>1)</sup>	inch/rev		
Tool offset	Length	±0.001 to 9999.999	mm	±0.0001 to 999.9999	inch		
	Radius	±0.001 to 999.999		±0.0001 to 99.9999			
Dwell	X	0.01 to 99999.999	s	0.01 to 99999.999	s		
	F	0.01 to 99999.999		0.01 to 99999.999			
	S	0.1 to 99.9	Revol- utions	0.1 to 99.9	Revol- utions		
Angle for contour definition A		-		-			
Angle with oriented spindle stop (M19)						0.1 to 359.9	Degrees
R parameters		Dimension depending on association (internal floating point) all combinations					

1) The maximum speed with linear feed (G94) must not be exceeded.

2) The limit values are valid for MD 155 = 2



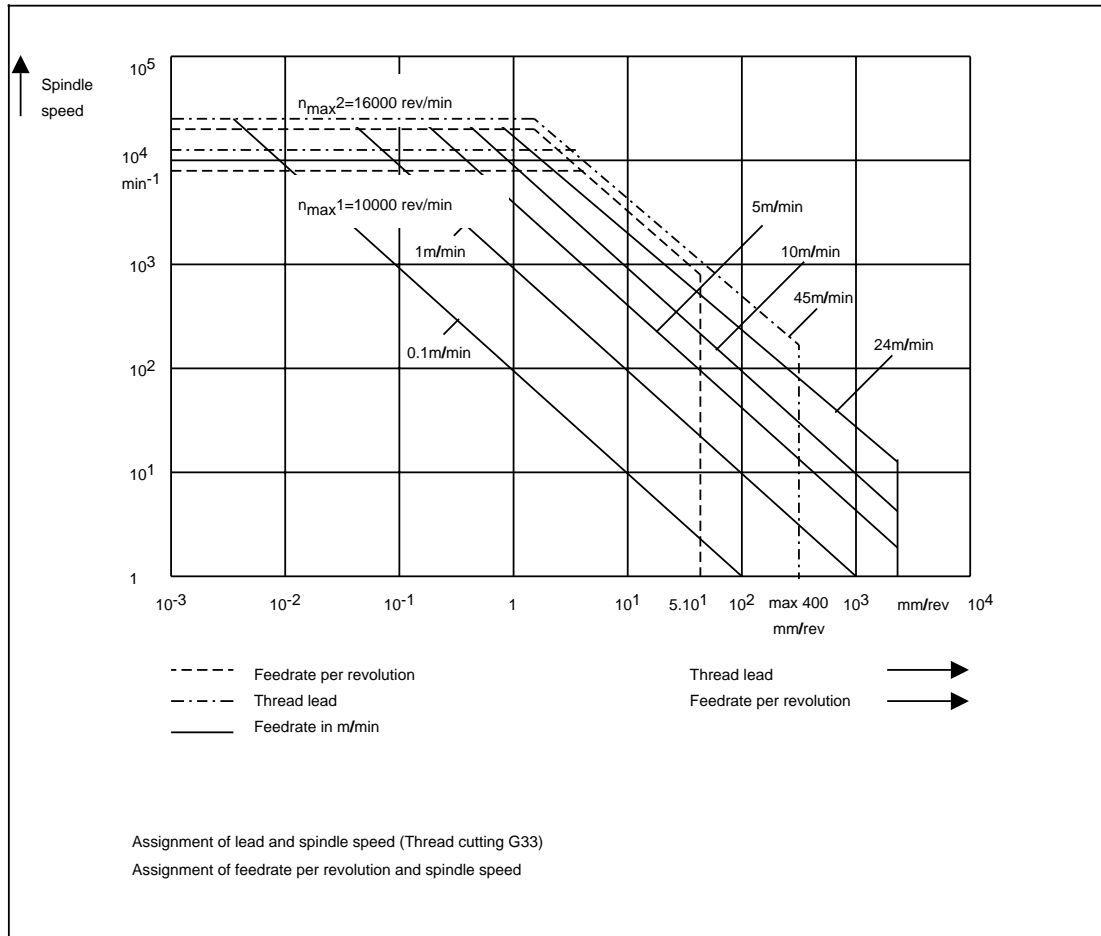
Input resolution: 0.0001 mm or position control resolution 0.00005 mm  
0.00001 inch 0.000005 inch  
0.001 degrees 0.000005 degrees

Significance Addresses		Metric		Inch		Degrees	
		Range	Unit	Range	Unit	Range	Unit
Position data (linear axes) Interpolation parameters		±0.0001 to 9999.9999	mm	0.00001 to 999.99999	inch	-	Degrees
Position data for G91 (rotary axes)		-		-		0.001 to 99999.999	
Position data for G90 (rotary axes)		-		-		±0.001 to 359.999	
Chamfer (U-); radius (U)		0.0001 to 9999.9999		0.00001 to 999.99999		-	
Zero offset		±0.0001 to 9999.9999		±0.00001 - 999.99999		±0.001 to 99999.999	
Thread lead		0.0001 to 40.0000		0.00001 to 1.60000		-	
Spindle speed S (value determined via commissioning setting)		1 - 16000	1 rev/min	1 - 16000	1 rev/min		
		0.1 - 1600.0	0.1 rev/min	0.1 - 1600.0	0.1 rev/min		
Linear feedrate (F) (G94) <sup>1)</sup>		0.001 to 4500.000	mm/min	0.0001 to 175.0000	inch/min	1 to 45000	Degrees/ min
Feedrate per revolution (F)(G95) <sup>2)</sup>		0.0001 to 5.0000 <sup>2)</sup>	mm/rev	0.00001 to 0.20000 <sup>2)</sup>	inch/rev		
Tool offset	Length	±0.0001 to 999.9999	mm	±0.00001 - 99.99999	inch		
	Radius	±0.0001 to 99.9999		±0.00001 - 9.99999			
Dwell	X	0.01 to 99999.999	s	0.01 to 99999.999	s		
	F	0.01 to 99999.999		0.01 to 99999.999			
	S	0.1 to 99.9	Revolu- tions	0.1 to 99.9	Revolu- tions		
Angle for contour definition A		-		-		0 to 359.99999	Degrees
Angle with oriented spindle stop (M19)						0.1 to 359.9	Degrees
R parameters		Dimension depending on association (internal floating point) all combinations					

1) The maximum speed with linear feed (G94) must not be exceeded

2) The limit values are valid for MD 155 = 2

## 1.10 Revolutional feedrate limit data



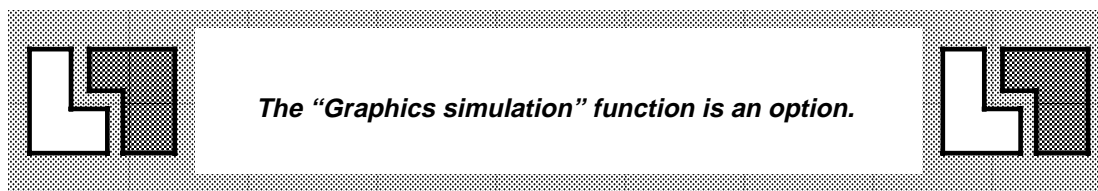
$n_{\text{max}1}$  Attainable with encoder 1024 pulses/revolution  
 $n_{\text{max}2}$  Attainable with encoder 512 pulses/revolution

## 1.11 Channel structure

The **SINUMERIK 810M/820M** is provided with 3 channels. These channels permit the **simultaneous** processing of two different programs in addition to other structural operations such as program editing and interface operation at the same time as processing in "AUTOMATIC" mode.

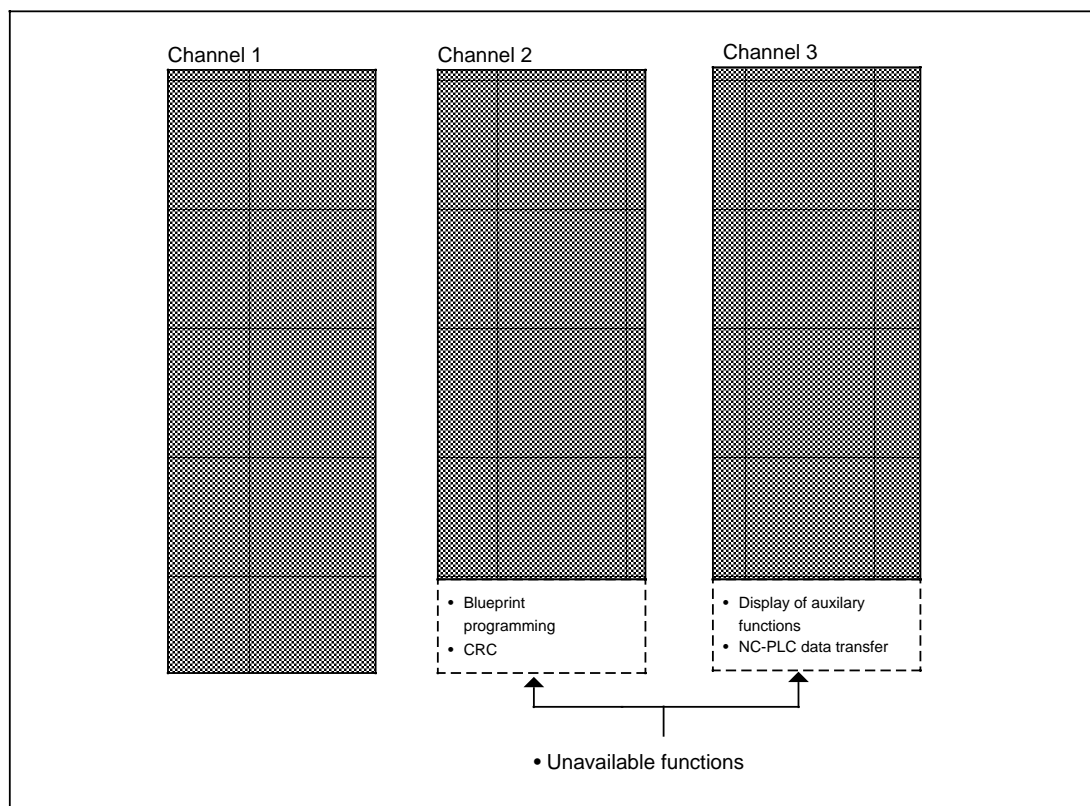
The three channels have the following significance:

- Channel 1:** Main channel for processing programs and spindle programming.
- Channel 2:** Auxiliary channel for processing programs for auxiliary axes or for mathematical functions in the background.
- Channel 3:** Graphic simulation for on-screen representation of programs.



In principle all **three channels** can be operated **simultaneously**. However, **problems of collision** arise with a small number of functions.

Functional scope of channels:



The **auxiliary channel (channel 2)** is a full-grade channel, with the exception of the unavailable functions. Its main function is to perform background calculations or auxiliary motions (e. g. tool changes).

The assignment of the axes (in automatic mode) to be traversed in each channel must be performed in the program. The same axis can be moved in channel 1 and channel 2 if the outputting of a travel command simultaneously from the 1st and 2nd channels is excluded (... alarm 180\*- axis programmed in both channels).

However, the main function of the **auxiliary channel** is to operate **loading axes** under PLC control at the same time as the main channel. Given the above-mentioned conditions, however, it is also possible to achieve other options using the auxiliary channel, giving rise to numerous potential applications.

However, since **only M functions** can be transferred from **channel 2** to the PLC, the opportunities for data transfer with the PLC are limited.

**Channel 3** is used exclusively for the graphic simulation of a part program. Another part program can be executed concurrently.

(See the notes in Part 1 "Operating", Section 3.1.13.5 "Shift program" and Section 3.1.14 "SIMULATION").

Spindle control can be made possible for the 2nd channel via machine data. For this purpose, all spindle functions are also available in the 2nd channel.

## 2 Directions of Movement, Dimensional Notation

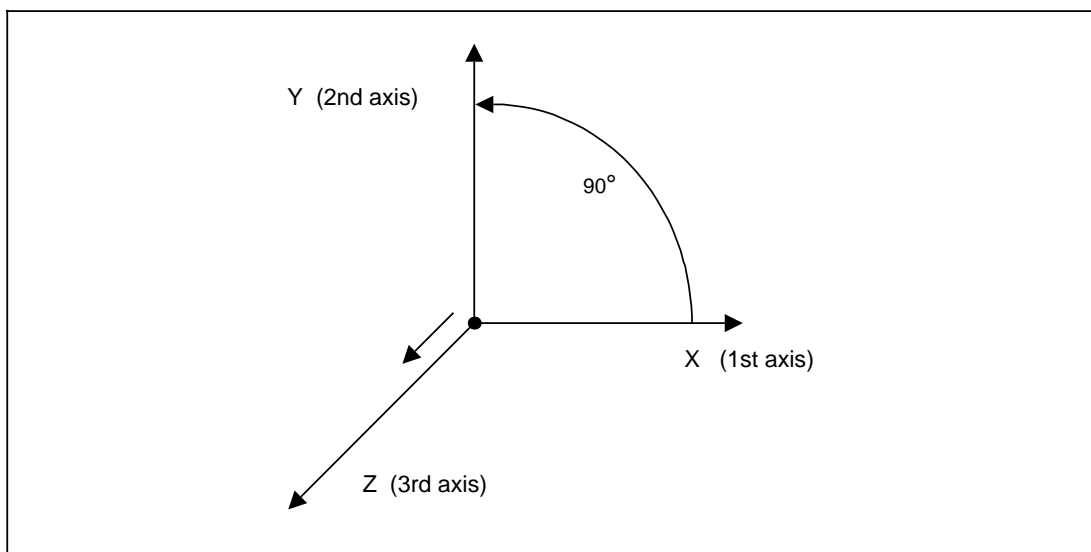
### 2.1 Coordinate system

The directions of movement of a machine tool are based on a coordinate system allocated to the axes of motion of the machine.

The **coordinate system used is clockwise and perpendicular**, and has X, Y and Z axes. The system is based on the main axes of the machine.

The coordinate system is defined as follows:

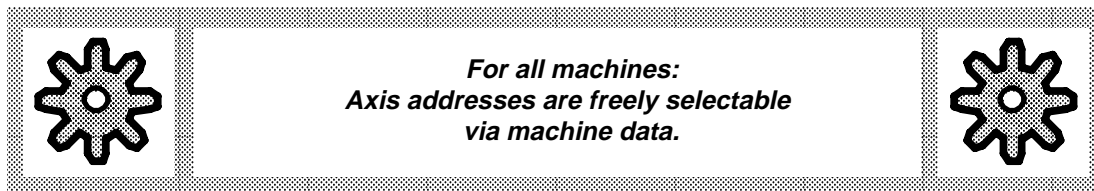
- The second axis is perpendicular to the first axis.
- If the first axis rotates over the shortest path ( $90^\circ$ ) towards the second axis, a connected screw with right-hand thread will move in the direction of the third axis.



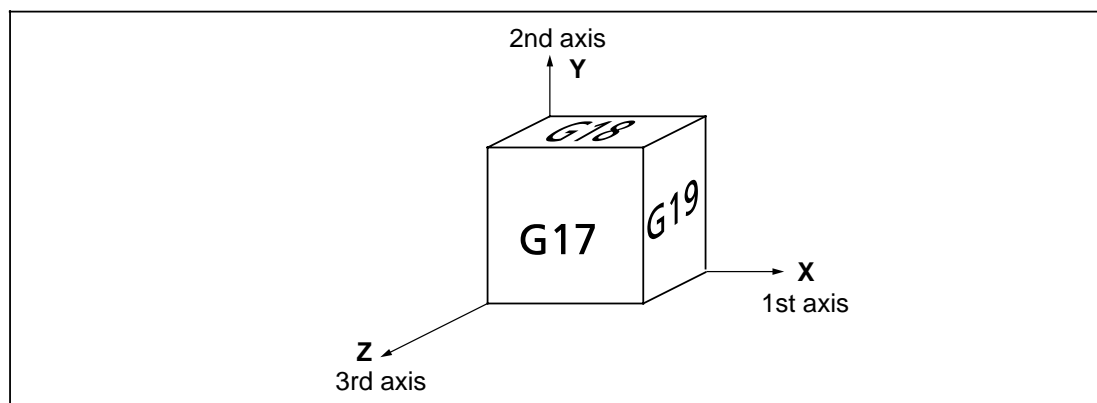
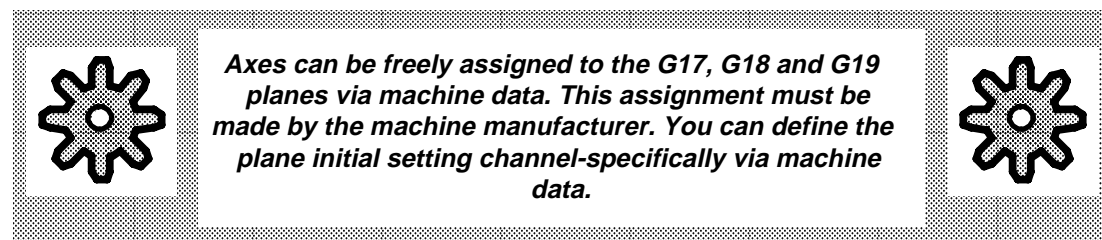
*Clockwise coordinate system*

The program is the same irrespective of whether the workpiece or the tool is moved during machining.

The default allocations for **milling machines** are as follows: **Main axes X, Y and Z**



## 2.1.1 Flexible plane selection



An example of axis assignment with flexible plane selection

### Definition of the planes in the initial setting to DIN:

M version	T version
G16	Plane selection with free axis selection
G17	Plane X – Y (1st axis – 2nd axis)
G18	Plane X – Z (1st axis – 3rd axis)
G19	Plane Y – Z (2nd axis – 3rd axis)

Find out the three axis names for the axes which form the G17, G18 or G19 planes from the machine manufacturer.

The assignment of the axes to the G17, G18 and G19 planes is also defined by the machine manufacturer. If the axis names are entered incorrectly alarm 3003 “invalid address” appears after NC start.

You can select the predefined planes G17, G18 and G19 by calling the corresponding G function in the program.

You can also set the initial setting for the plane selection via machine data (NC MD 110\*) separately for each channel (see Part 1 “Operating”, Section 7.3.4 “Definition of the initial setting of the G groups”).

**Plane selection via G16**

Program G16 if the required plane is no longer determined by the axes defined in G17, G18 or G19 or additional definitions are required. This applies to the following functions.

- Parallel axis assumes the function of a main axis
- Additional definition of the tool offset
- Specification of directions of angle heads

**Example 1:** Parallel axis

G16 X Y W

Let G17 plane be defined by axes X,Y,Z,  
let the W axis (parallel axis) be parallel to the Z axis.

**Example 2:** Angle head

G16 X Y Z Y

The first two axis addresses (X, Y) define the plane in which the cutter/tool nose radius compensation applies. The fourth axis address specifies the axis with the additional length correction in the previously defined plane.

The 3rd and 4th axis address can be given a negative sign to reverse the direction of the CRC/TNRC.

## 2.2 Position data, preparatory functions

The position data comprises an axis address and a numerical value, which describes the path on the addressed axis. If a sign is specified, it is written between the address and the numerical value.

In order to start the positioning procedure, the position data must be supplemented by the preparatory function (G function) and the feedrate (F) data. The preparatory functions describe the type of machine movement, the type of interpolation and the method of dimensioning.

The **G functions** are **subdivided into groups** (see Program key, Section 12). A program block may only contain one function from each group. The G functions are either **modal** or **non modal**:

- The G functions which remain active until they are replaced by a new G function in the same group are said to be modally active.
- The G functions which are only active in the block in which they are contained are said to be active block-by-block.

The resets take effect after powering up the controller, a reset or a program end. They need not be programmed.

## 2.3 Dimension systems: absolute and incremental position data G90/G68/G91

The traversing movement to a particular point in the coordinate system can be described by means of **ABSOLUTE** or **INCREMENTAL** position data input:

### Absolute position data input G90

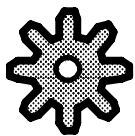
If absolute position data input is selected, all dimensional inputs refer to a fixed zero, which is normally the **workpiece zero**. The value of the associated position data specifies the target position in the coordinate system.

### Absolute dimensioning on shortest way (only with rotary axis) G68

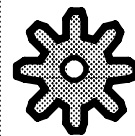
If G68 is selected the block end value is approached on the shortest way ( $< 180^\circ$ ). This function is modal. It is only active with axes which have been assigned the "Modulo programming" part function. Apart from the part function, G68 has an effect equivalent to G90.

### Incremental position data input G91

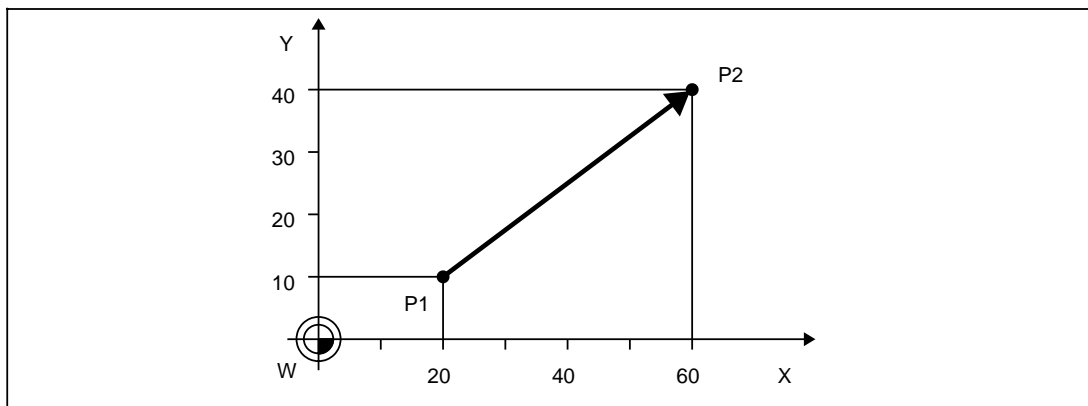
If incremental position data input is selected, the value of the position data corresponds to the path to be traversed. The direction of movement is specified by the sign. Depending on its current position the tool moves on by the programmed values. It is possible to switch between absolute and incremental position data input from block to block as desired, since the controller actual value is always referred to the zero point. A zero offset is calculated for both absolute and incremental programming.



*Depending on machine data, mixed programming using G90 and G91 in the same block is also possible.*





**Example:** Absolute and incremental position data input

Absolute position data input:

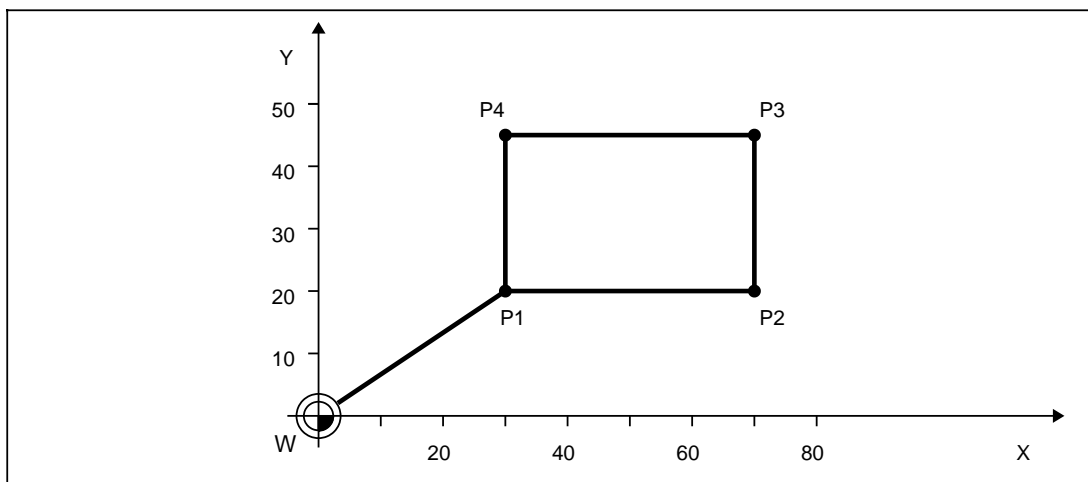
`N...G00 G90 X60 Y40 L_F`

The tool moves from any position to P2

Incremental position data input:

`N...G00 G91 X40 Y30 L_F`

The tool moves from P1 to P2

**Example:** Programming with absolute position and incremental position data**Changing between absolute dimensioning G90 and incremental dimensioning G91:**

You can change between absolute and incremental dimensioning from block to block.

`%10 L_F`

`N05 G00 G90 G94 X30 Y20 L_F` (P1)

`N10 G01 G91 X40 F100 L_F` (P2)

`N15 Y25 L_F` (P3)

`N20 X-40 L_F` (P4)

`N25 Y-25 L_F` (P1)

`N30 M30 L_F`

## 2.4 Reference points

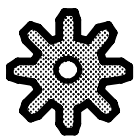
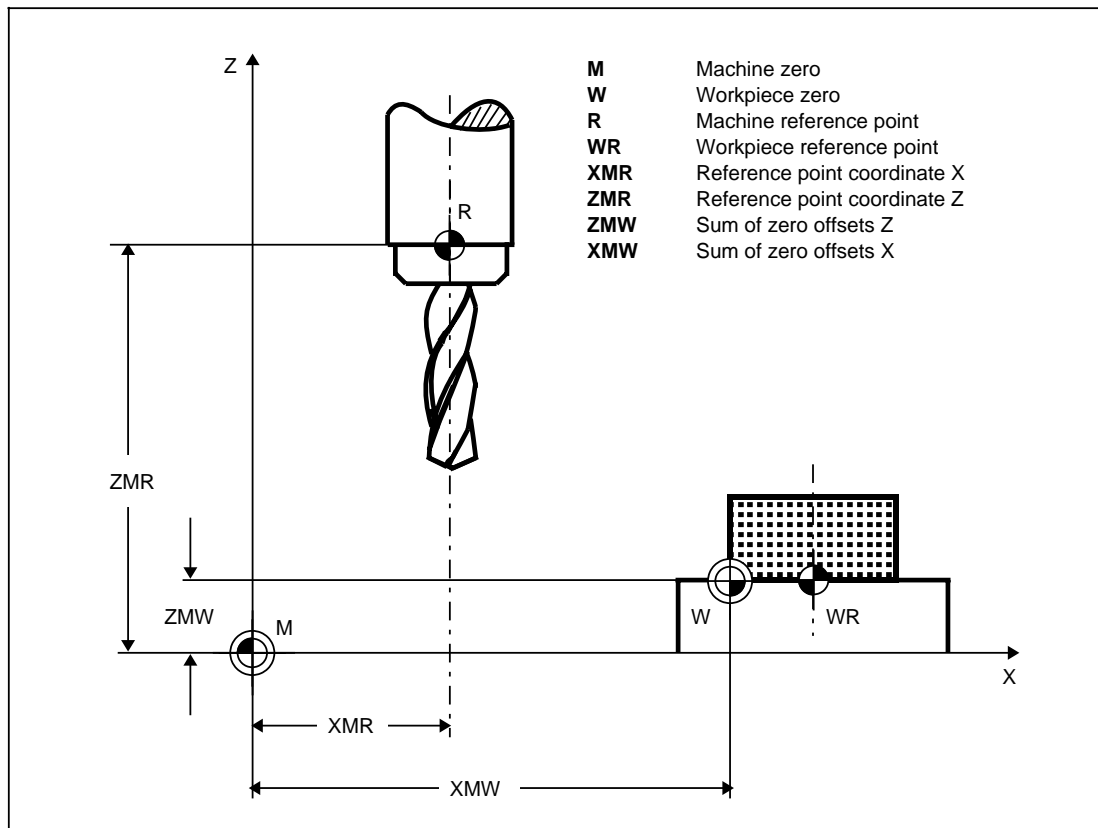
The zeros and various reference points are defined on all numerically controlled machine tools.

The machine zero **M** is the design zero of the machine coordinate system.

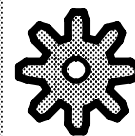
The workpiece zero **W** is the zero defined for programming the workpiece dimensions. It can be freely selected by the programmer. The relationship to the machine zero is defined by the zero offset.

Reference point **R** is a point defined by the machine manufacturer which is approached when the controller is powered up and which synchronizes the NC control with the machine tool.

**Example:** Reference points of drilling plane

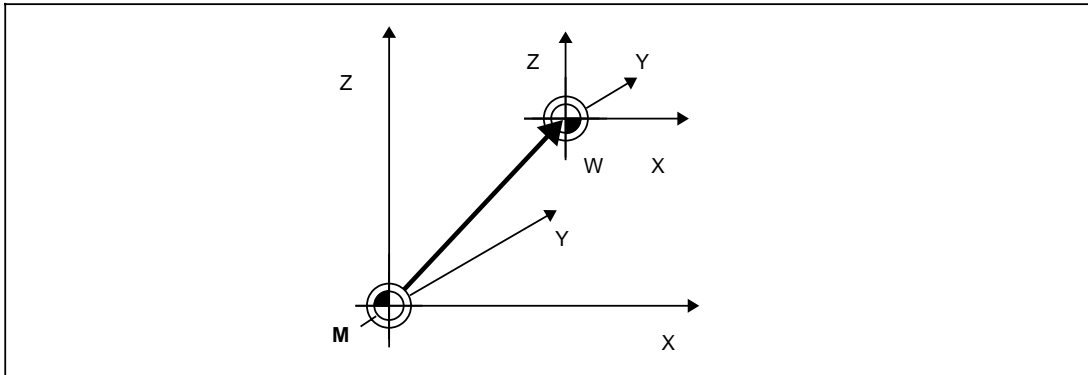


*The reference dimension is defined  
in the machine data.*



## 2.5 Zero offset

Zero offset is the distance between the workpiece zero **W** (on which the dimensions are based) and the machine **M** zero.

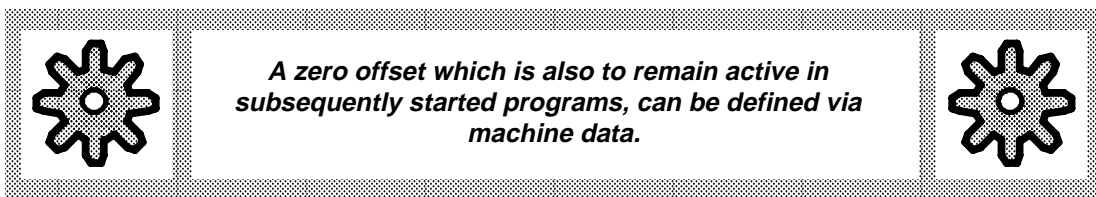


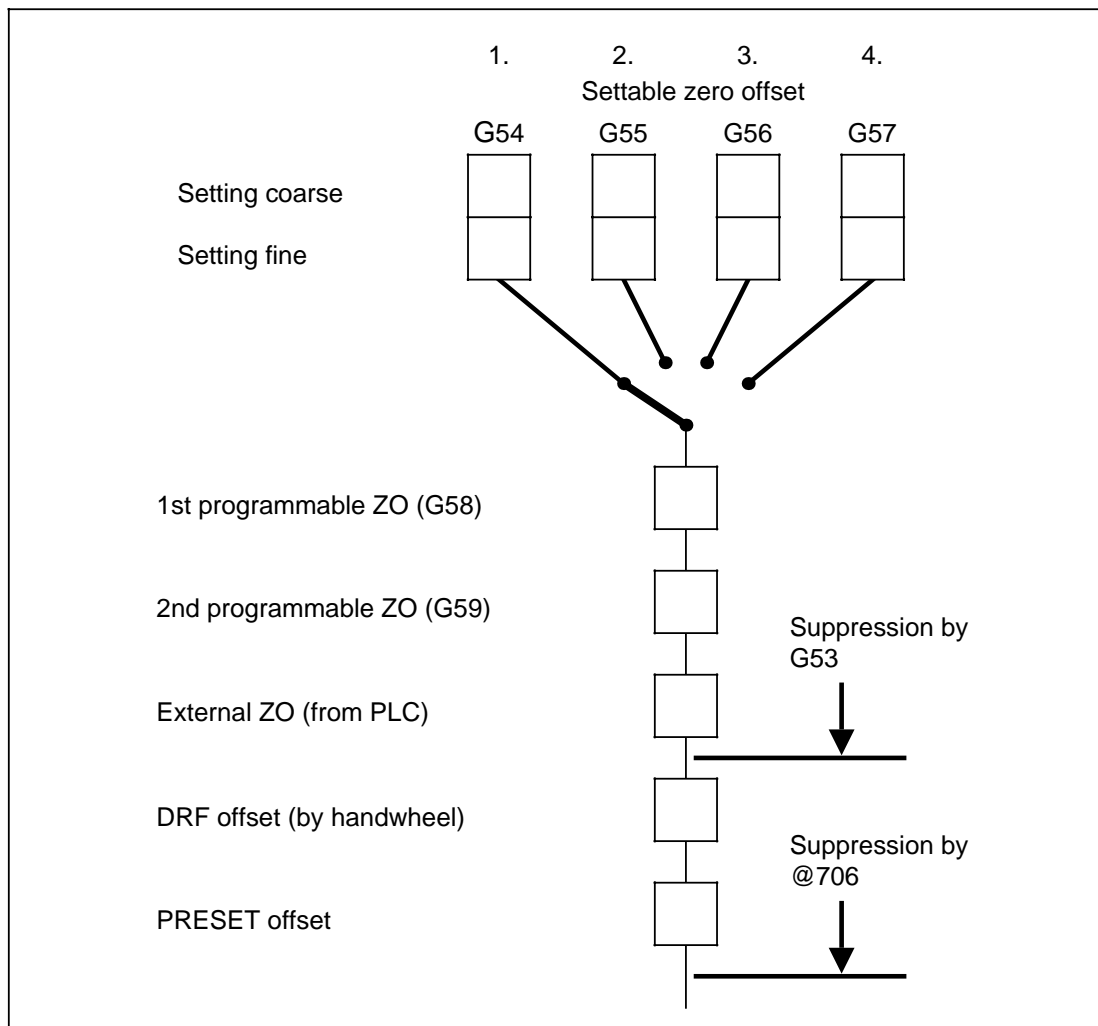
*Zero offset in the drilling plane*

The following types of zero offset can be activated:

- Settable zero offset (G54 to G57)
- Programmable zero offset (G58, G59)
- External zero offset (from PLC).

The zero offsets apply to the current program.





*Sum of zero offsets*

Sum of zero offsets =  
settable zero offset (G54 to G57) + programmable zero offset (G58, G59) + external zero offset (from PLC) .

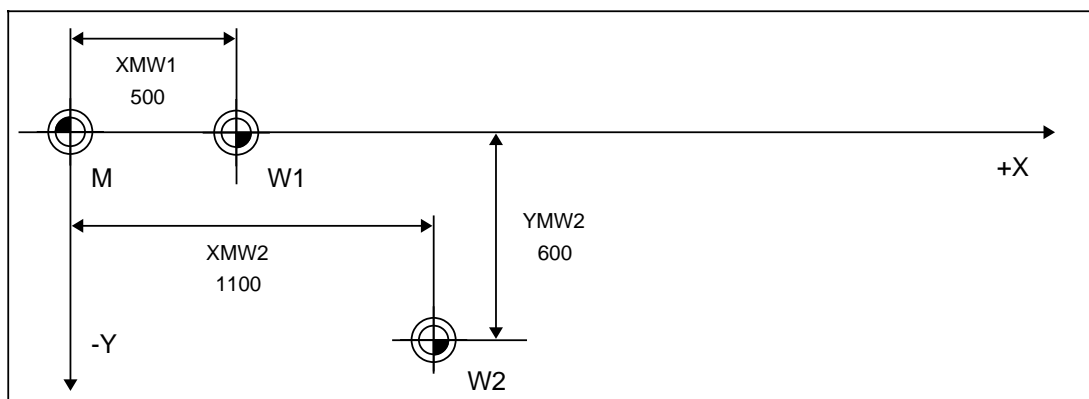
### **Settable zero offset G54, G55, G56, G57**

The settable zero offset values for each axis can be entered in the controller via the operator panel or via the universal interface.

The values are calculated in absolute and incremental position data blocks for the block end point if the relevant axis is programmed.

G54 to G57 permit 4 settable zero offsets each with two settings to be selected for the various axes. The various settable zero offsets subdivide into 2 ZOs each (coarse ZO and fine ZO), which are calculated additively.

The fine ZO is used as an additional fine offset (compensation) of the zero point.



Settable zero offset

The settable zero offset is entered via the universal interface:

%Z0A	L <sub>F</sub>				
G154	X = 250	Y = 280.1	L <sub>F</sub>	} Settable coarse ZO	
G155	X = 220.34	Y = 250.125	L <sub>F</sub>		
.					
G157	X = 320	Y = 350	L <sub>F</sub>	} Settable fine ZO	
G254	X = 0.1	Y = 0.3	L <sub>F</sub>		
.					
G257	X = 0.1	Y = 0.5	L <sub>F</sub>		
M02	L <sub>F</sub>				

For reasons of compatibility, the format **G54 Y = 250 L<sub>F</sub>**, for example, can be read in, the values then being entered in the settable coarse ZO.

### Programmable zero offset G58/G59

An additional zero offset can be programmed with G58 and G59 under the axis address for all existing axes. When calculating the path, the programmed values are added to the settable zero offset and external zero offset values.

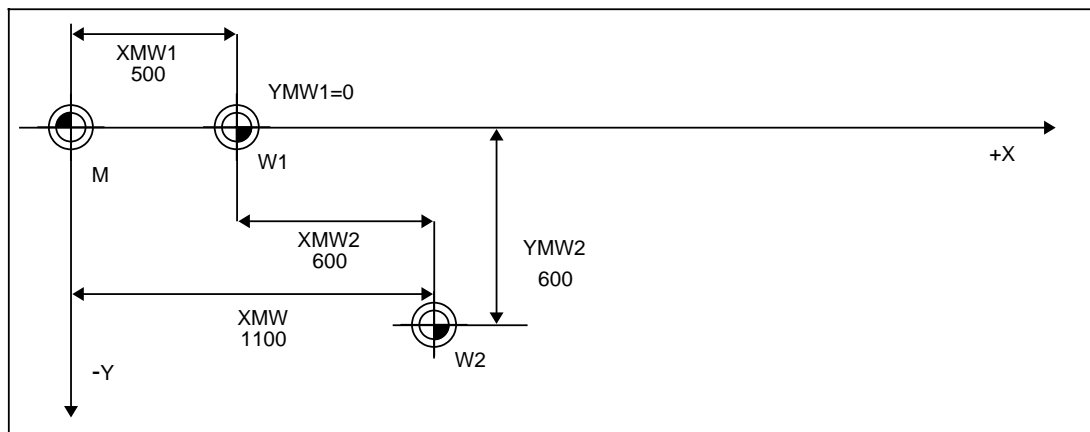
- **Settable** zero offset (coarse and fine)  
Input values, YMW1, XMW1
- **Programmable** zero offset  
Input values, YMW2, XMW2
- Total effective zero offset  
 $YMW = YMW1 + YMW2$   
 $XMW = XMW1 + XMW2$

Programming:

```

N30 ...
N35 G54 LF
N40 G59 X600 Y600 LF
N45 ...

```

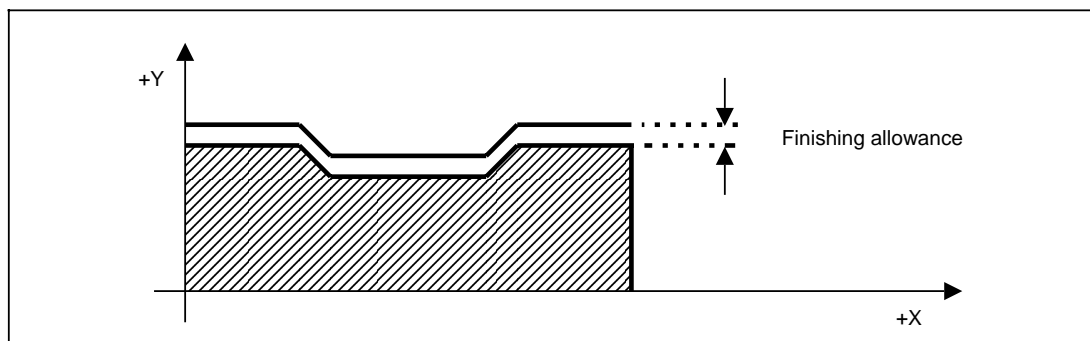


Settable and programmable zero offsets

A block containing G58 or G59 must not include any functions other than the zero offsets. The G58/G59 functions can apply to a maximum of five axes within one block.

### Application example with G59:

The contour has been exclusively programmed using absolute position data. In order to obtain a **finishing allowance**, the total contour can be offset in the Y coordinate by means of a **programmable zero offset**.

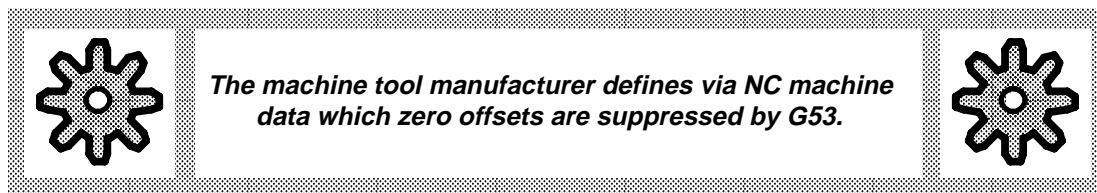


Zero offset with G59

To select: N..G59 Y... L<sub>F</sub>  
To cancel: N..G59 X0 L<sub>F</sub>

The programmable zero offset values set in this program are automatically reset each time the program is terminated with M02 or M30 or aborted. After RESET all programmable zero offsets are reset.

### G53 Cancelling zero offsets



The offset of the coordinates from the machine zero to the workpiece zero obtained by means of:

- settable zero offsets (G54 to G57)
- programmable zero offsets (G58, G59)
- external zero offsets (from PLC)

can be cancelled block by block using G53.

If the relevant machine data is set, you can also use G33 to cancel:

- DRF offset
- PRESET offset.

In this case G53 has the same effect as @706 (see Section 11.9).

The tool offset must be cancelled separately. In the block following G53 all zero offsets will be active again.

Reference to machine zero:

N30 D0 L<sub>F</sub>

Cancellation of tool offset

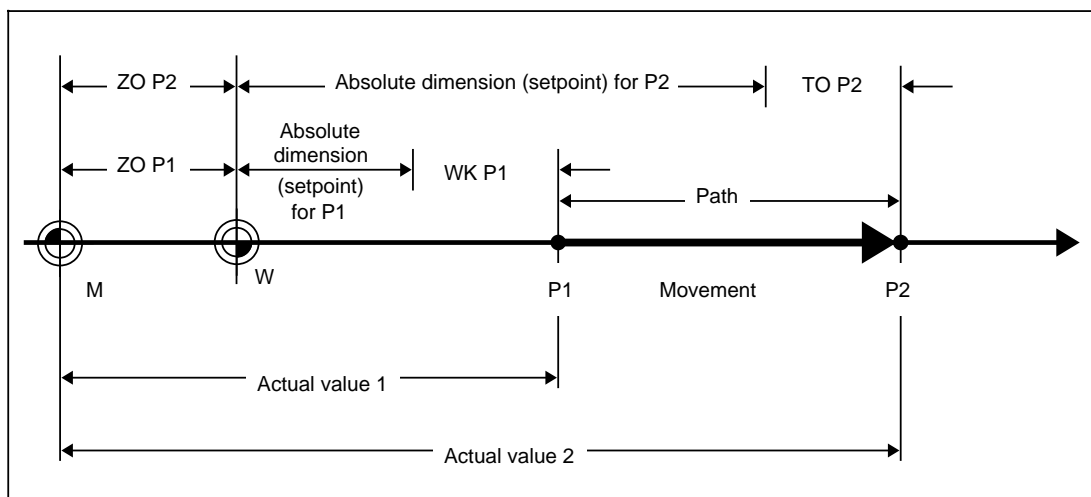
N35 G53 X... Y... L<sub>F</sub>

Cancellation of all zero offsets and return to position in machine system

## 2.6 Path calculation

The path calculation determines the distance to be travelled within a block, taking all offsets and compensations into consideration.

The formula is generally as follows:  $\text{Path} = \text{setpoint} - \text{actual value} + \text{zero offset (ZO)} + \text{tool offset (TO)}.$



Path calculation using absolute position data input

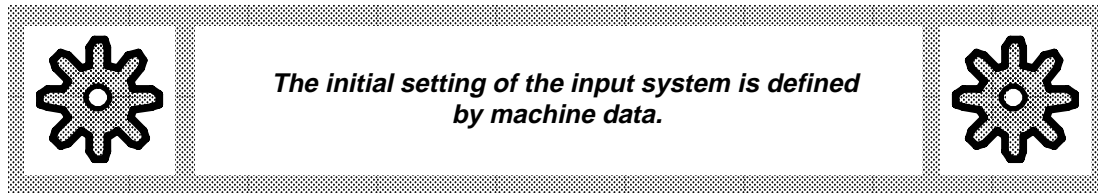
If **incremental position data input** is used, the zero offset is incorporated normally in the first block:  $\text{Path} = \text{incremental position data} + \text{zero offset} + \text{tool offset}$

If a new zero offset and a new tool offset are programmed in a new program block, the formula is as follows:

- With **absolute position data input**  
 $\text{Path} = \text{absolute position data P2} - \text{absolute position data P1} + \text{ZOP2} - \text{ZOP1} + \text{TOP2} - \text{TOP1}.$
- With **incremental position data input**  
 $\text{Path} = \text{incremental position data} + \text{ZOP2} - \text{ZOP1} + \text{TOP2} - \text{TOP1}.$

## 2.7 Workpiece dimensioning, input system G70/G71

The dimensions can be entered in the program in either mm or inches.



The input system can be changed by selecting the preparatory functions G70 or G71:

G70     input system:    inches  
G71     input system:    metric (mm)

The controller converts the entered value into the input system of the initial setting. When this type of block is processed, the value will be displayed already converted in the initial setting of the system.

It is essential to ensure that the **units of measurement** are the **same** before selecting subroutines or cycles.

The unit of measurement which is different from the initial setting can be fixed for one or more blocks or for an entire program.

The first block must then contain the necessary G function; the initial setting must be written again following the last block (the initial setting is written automatically following a program end with M02 or M30).

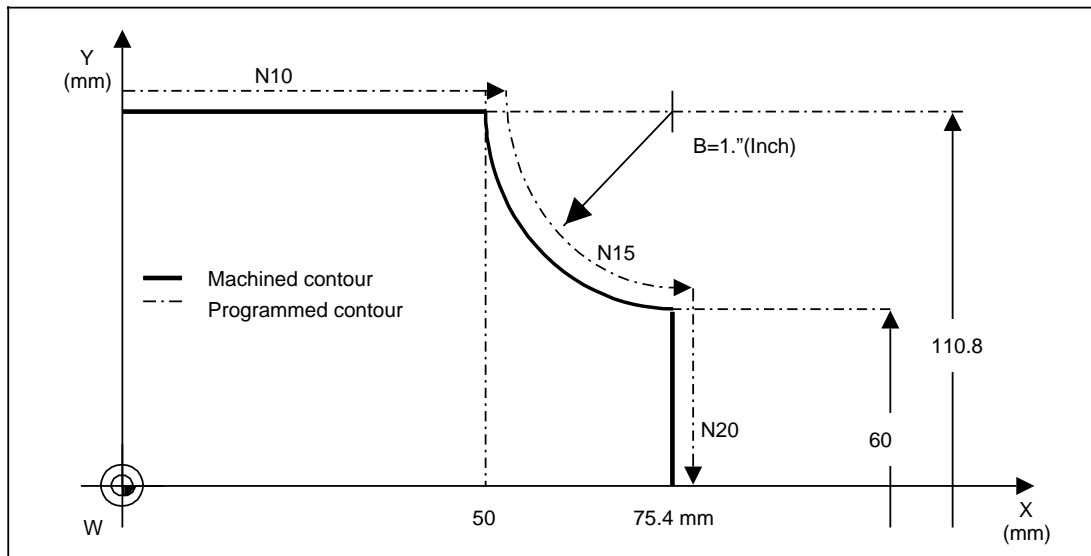
The following are dependent on the **initial setting** of the input system:

- Actual value display (including distance to go)
- Zero offset
- Feedrate/cutting speed G94/G95
- Tool offset

The following are dependent on the **programmed G70 or G71**:

- Position data X, Y, Z
- Interpolation parameters I, J, K
- Chamfers/radii U-/U
- Parameters related to position data, interpolation parameters and chamfers/radii.



**Example: G71 - Initial setting (metric)**

Input in inches for initial setting G71

N05 ...

N10 G91 X50 L<sub>F</sub>

N15 G03 G70 X1 Y-1 I1 J0 L<sub>F</sub>

N20 G01 G71 Y-30 L<sub>F</sub>

N25 ...

Circular arc programmed in inches

Straight line programmed in metric dimensions

## 2.8 Mirroring

### Mirroring of one axis

Mirroring of a coordinate axis permits machining of a contour

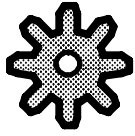
- with the same dimensions,
- at the same distance from the other axes,
- on the other side of the mirror axis and as a mirror image.

When mirroring an axis the controller inverts

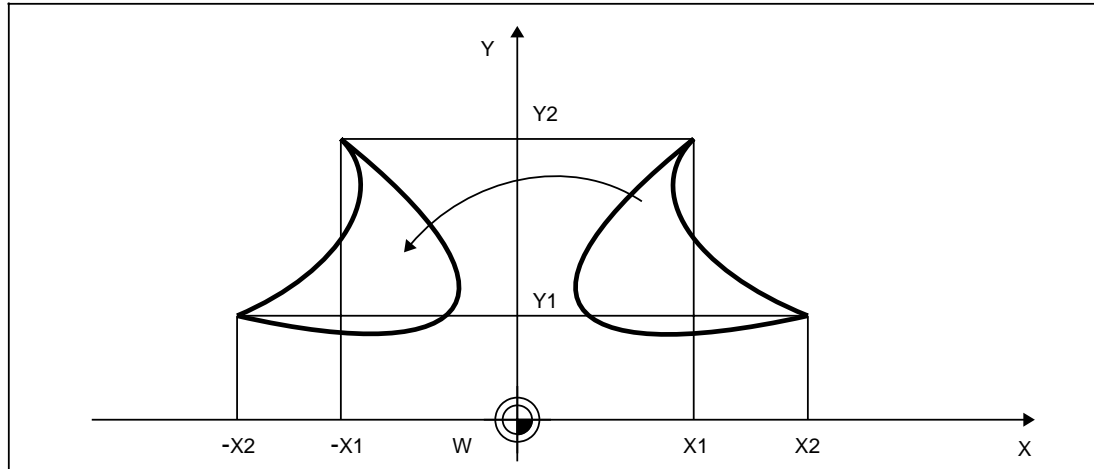
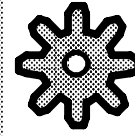
- the sign of the coordinates of the mirrored axis,
- the direction of rotation in the case of circular interpolation (G02 G03, G03 G02),
- the direction of machining (G41 G42, G42 G41).

There is no mirroring of:

- Tool length offsets
- Zero offsets.



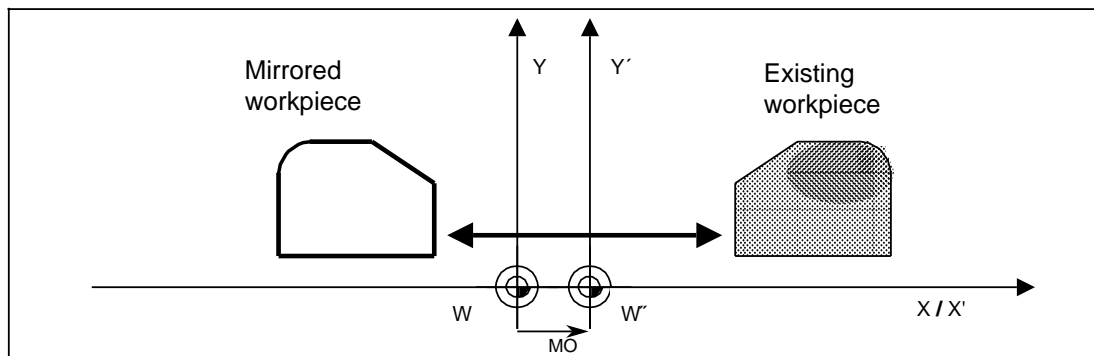
*If the bit of MD 572\*.3 has been set the tool offset for the facing axis is included in mirroring.*



*Mirroring of X axis*

Mirroring is always about the coordinate axis. In order for the contours to be mirrored to the exact position where they are to be machined, the position of the program start when mirroring is called must be such that the axes of the coordinate system are located exactly between the programmed contour and the mirrored contour.

If necessary, the zero of the coordinate system can be offset to the correct position before mirroring is called in the program (W to W prime). To mirror the workpiece onto the right position the zero must be offset by the value "MO". This ensures that the distance of both workpiece from the zero is equal. After mirroring the zero can be set back to its original position.



*Offset of workpiece zero*

## Mirroring of two axes

The mirroring process described is performed twice, e.g.:

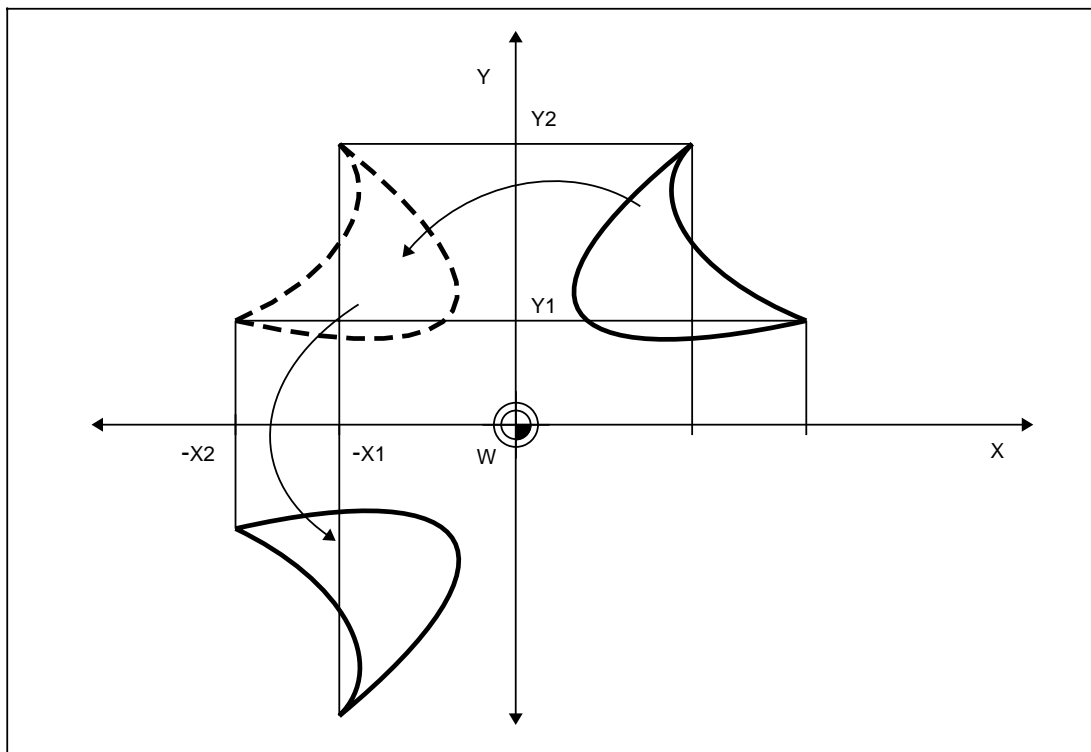
- X axis mirrored once and
- Y axis mirrored once.

The control inverts the signs of the two mirrored coordinates (XY).

The direction of machining and rotation of the circles remains identical since they are inverted twice.

No mirroring of:

- Tool length compensation
- Zero offset.

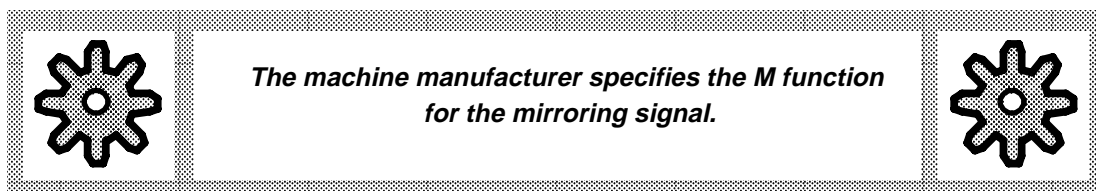


*Mirroring of X and Y axes*

The workpiece is always mirrored when the main axes are mirrored.

Program sections to be mirrored take the form of a subroutine in the program. This subroutine is then called by prefixing an appropriate mirroring function for the relevant contour.

The "mirroring" function is selected via the PLC.



### Example: Selection of mirroring

```

:
:
N10 G90 G54 G00 X0 Y0 L_F
N20 Z30 L_F
N30 G1 Z0 F500 L_F
N35 M... L_F           Mirror X axis
N36 G04 F... L_F       Dwell possibly on account of PLC cycle time
N37 L999 P1 L_F         Empty buffer
:
:
N40 X50 Y50 L_F
N45 M... L_F           Cancel axis mirroring (L999 L_F; @714 M17 L_F)
:
N55 M... L_F           Mirror Y axis
N60 G04 F... L_F       Dwell possibly on account of PLC cycle time
N65 L999 P1 L_F         Empty buffer
:
:
N80 G0 G53 Z0 L_F
N85 G53 X0 Y0 M30 L_F

```

Special function @ 714 (buffer empty) makes it possible to stop an additional block increment calculation until the buffer is empty.

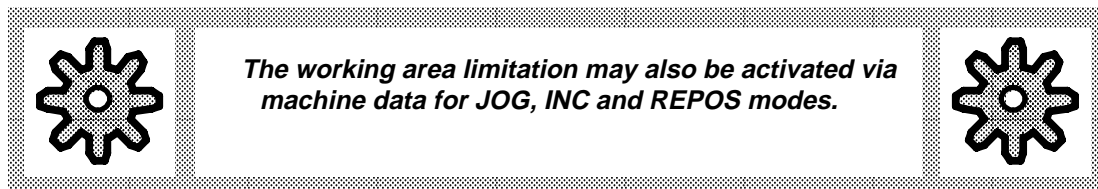
The use of @ 714 or L999 is required for all displacements requiring external influencing, e.g. mirroring.

## 2.9 Programmable working area limitation G25/G26

Programmable working area limitation provides **machine protection** in the event of programming and operating errors.

The slide reference point F must only move in the limited range. As soon as the tool leaves this limited area or is located outside this area on program start, or as soon as a position outside the working area limitation is programmed, the path setting is terminated or a travel command is not accepted (program stop, no program start, alarm).

The current following error is eliminated. Programmable working area limitation is active in automatic mode with the values in the setting data.



Programmable working area limitation is called using G25 and G26:

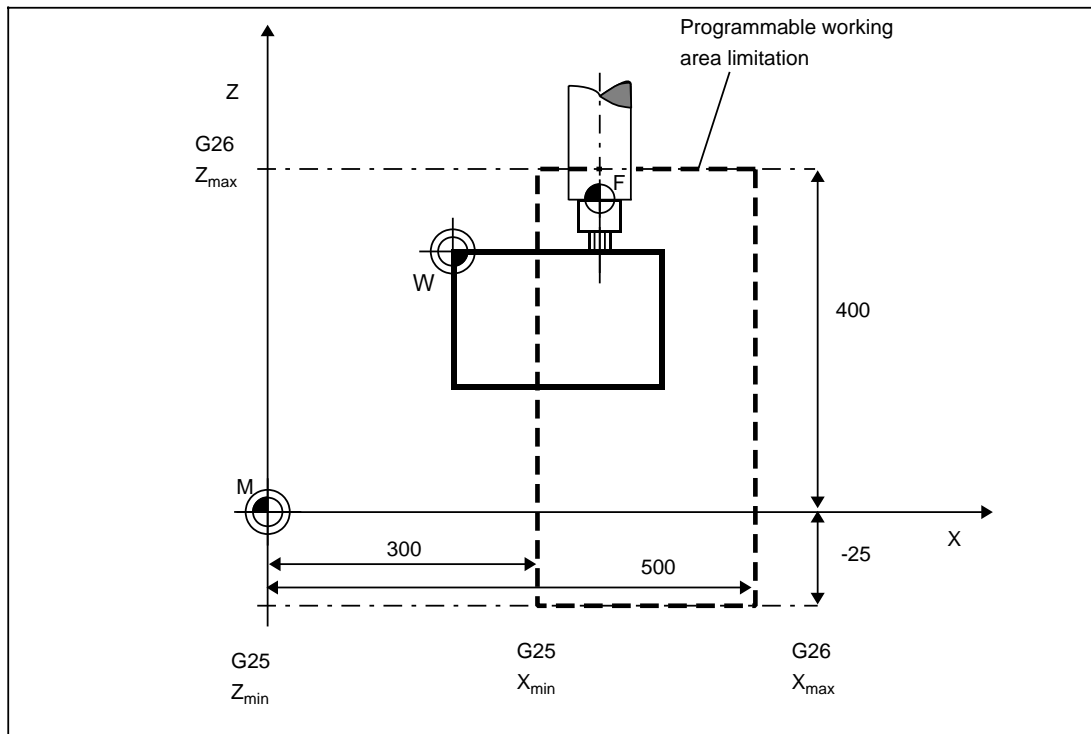
G25      minimum working area limitation  
G26      maximum working area limitation, e.g.

**Example:**

N10 G25 X300 Y-25 L\_F

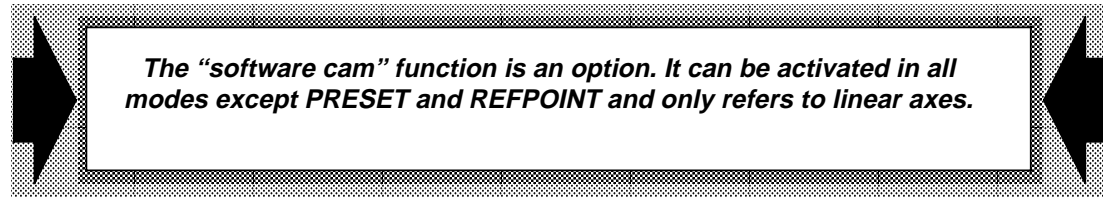
N20 G26 X500 Y400 L\_F

No more data is allowed in this block. Working area limitation is no longer active when -99999.999 and +99999.999 respectively are input for the minimum and maximum values per axis in the setting data.

**Example:** milling machine

## 2.10 Software cam

You can reduce the retooling times during a machining operation on a workpiece by dividing the machining table into two working areas. While in the first area a workpiece is being machined a tool can be changed in the second. You can use the “software cam” function to divide the machining table up in this way.



The “**software cam**” function generates **cam signals** and can be parameterized via R parameters and machine data. The R parameters contain the axis positions of the individual cams (**cam positions**) and are grouped into a cam parameter block. With five real axes, as with SINUMERIK 810T/M, 820T/M, up to ten cams can be set up. Two cams form a **cam pair**. Please consult the machine manufacturer for the numbers of the R parameters for the cams.

### Cam signals:

**Cam signals** are **control signals** from the NC. They emulate a cam of infinite length which is activated in direction of approach at the cam position. The cam signals are evaluated by the PLC program.

- Cam signals are only output after repositioning the axes.  
**Exception:** axes for which no start disable is programmed before reference point approach.
- The cam positions relate to the machine system.

### Cam pair and cam range:

A cam pair consists of a **plus cam** and a **minus cam**. The axis range of the plus cam is greater than its cam position and the axis range of the minus cam is less than its cam position. The cam positions must relate to the machine system (metric or inch). The machine related axis position can be read with @361.

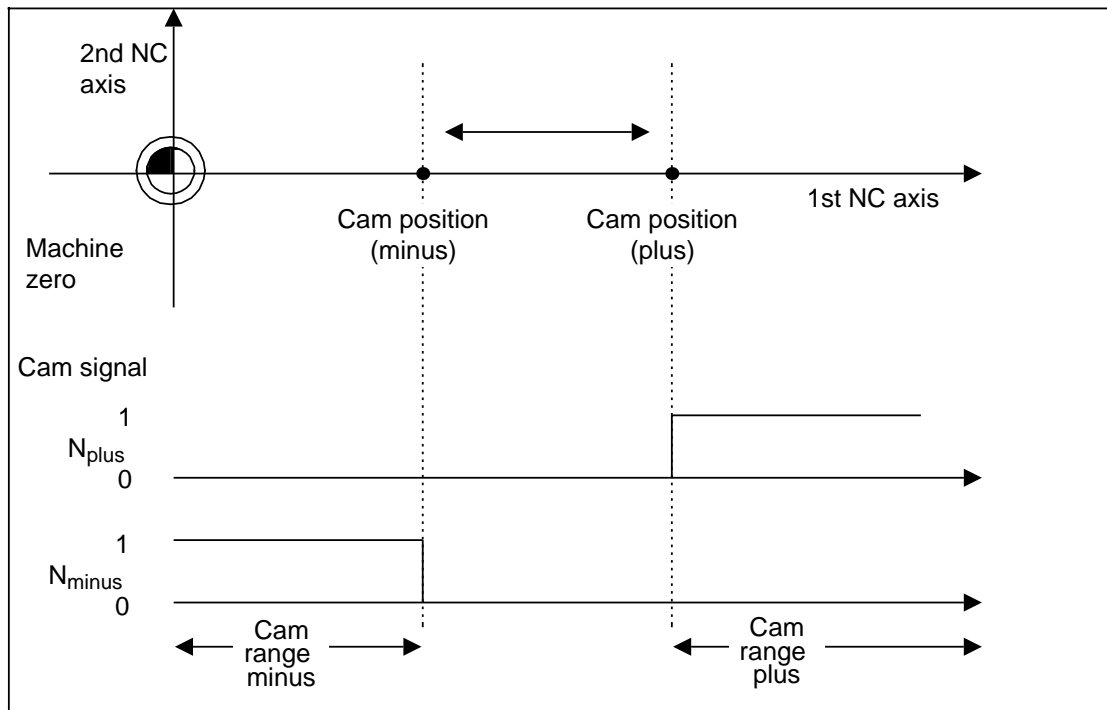
No check is made to whether the cam position lies within the maximum traverse range.

The axis range assigned to the cam is designated the **cam range**.

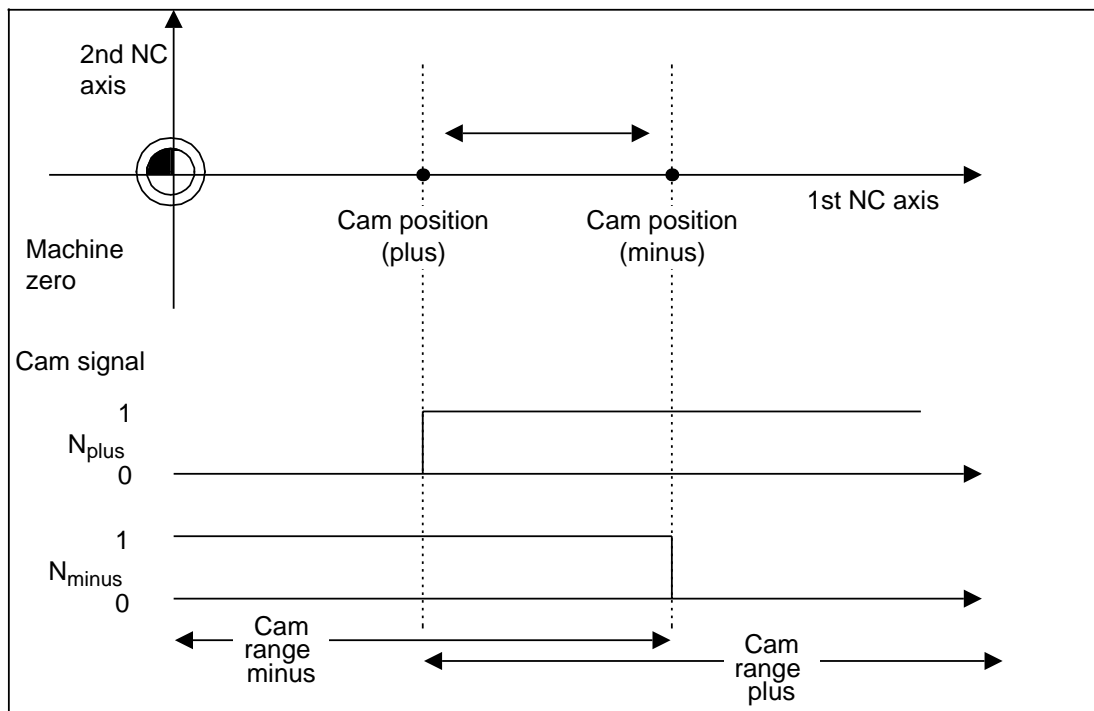
A cam pair can only ever be assigned to one NC axis,

**but:**

**Several** cam pairs can be activated for one axis.



*Minus cam < plus cam*



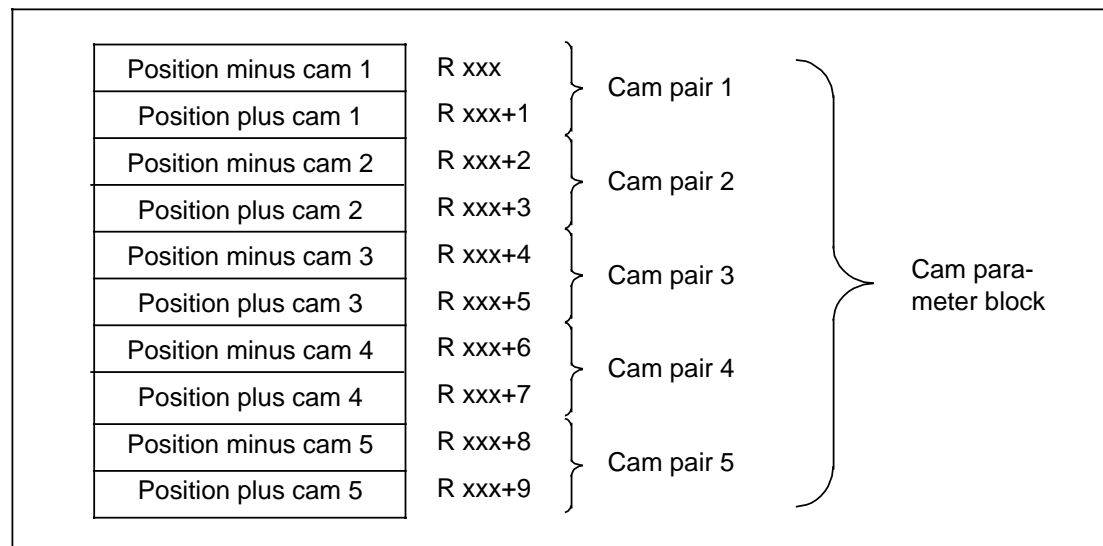
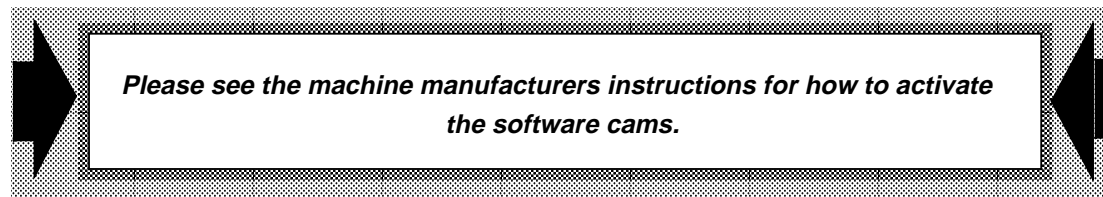
*Plus cam < minus cam*

### Cam parameters:

All cam parameters are grouped into an R parameter block. The R parameter block is designated cam parameter block.

In five consecutive R parameter pairs (**cam pairs**) the values of the axis positions for the ten software cams are stored. There is one parameter for the negative cam direction and one for the positive cam direction per cam pair. A cam parameter block can be formed from global or channel-specific R parameters. The beginning of the cam parameter block (Rxxx), i.e. the number xxx of the first R parameters, is set by the machine manufacturer.

The assignment of axes to cam pairs is also performed by and can be obtained from the machine manufacturer.



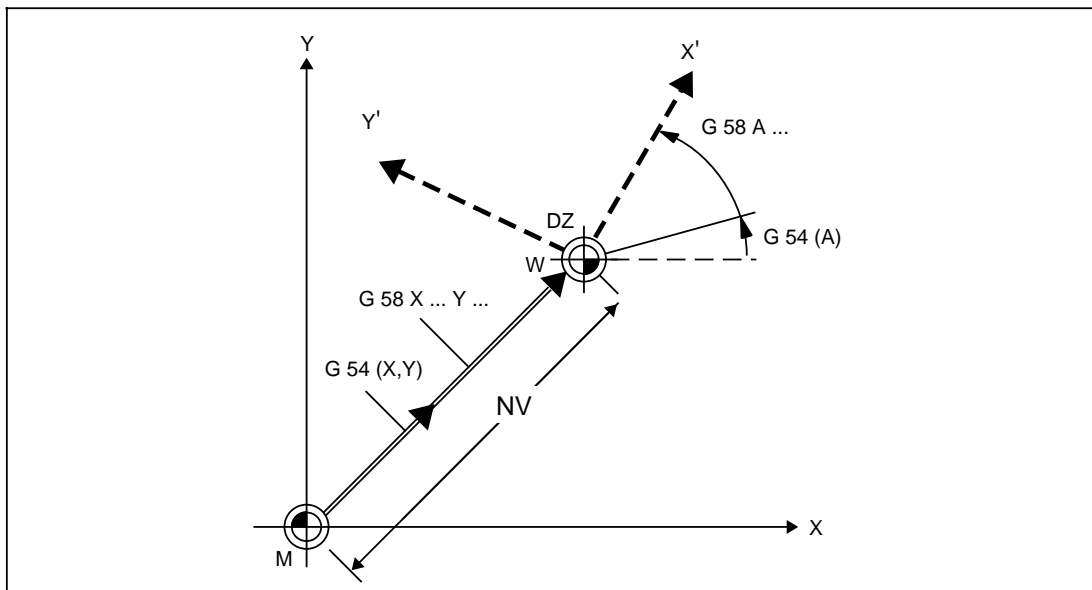
Definition of cam parameters



## 2.11 Coordinate rotation

By coordinate rotation the coordinate system of the workpiece can be adapted to the coordinate system of the machine. Execution of a part program (machining) takes place in the rotated coordinate system.

The centre of rotation (DZ) is the point defined by the sum of the zero offset (ZO).



The following coordinate rotations can be activated:

Settable coordinate rotation G54 to G57

Programmable coordinate rotation G58 A... and G59 A...

The sum of "settable coordinate rotation" and "programmable coordinate rotation" gives the "effective coordinate rotation".

### Settable coordinate rotation:

- There is no distinction between coarse and fine setting
- The adjustable coordinate rotation is defined using setting data
- Input of the value for angle of rotation "A" via keyboard into the input display coordinate rotation.
- Via the universal interface the angle of rotation "A" can be input for G54...G57 or for G154...G157 (input for G254...G257 is **not** permissible)

By G54 the Z0 entered in the setting data for G54 and the coordinate rotation G54A... entered in the setting data are activated.

### Example: Settable coordinate rotation

N...

N... G90 G54 L<sub>F</sub>

N...

**While coordinate rotation is active, the axes concerned may only be programmed with path feed.**

### Programmable coordinate rotation:

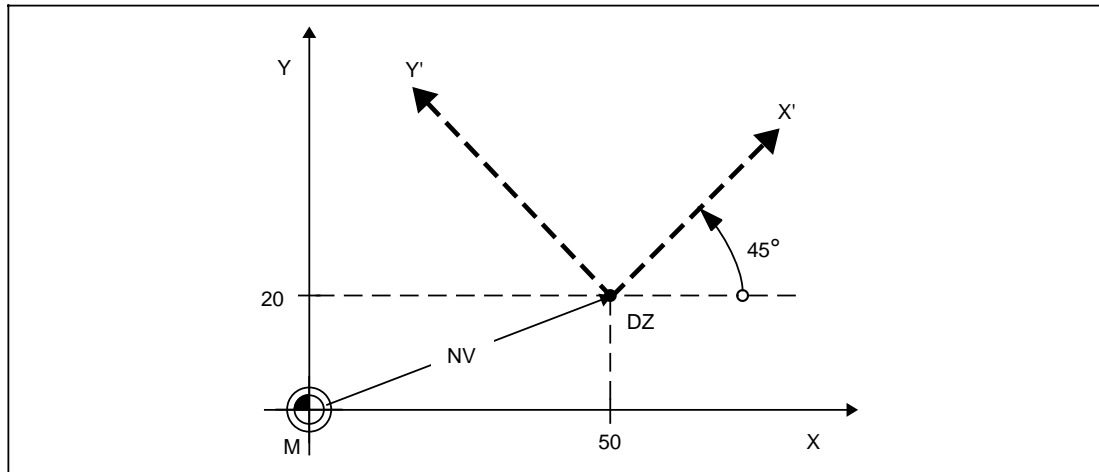
- Coordinate rotation can be programmed with G90 (absolute position data) or G91 (incremental position data)
- The value of the angle of rotation is programmed at address "A..."

### Example: Programmable coordinate rotation

N...

N... G90 G58 X50 Y20 A45 L<sub>P</sub>

N...

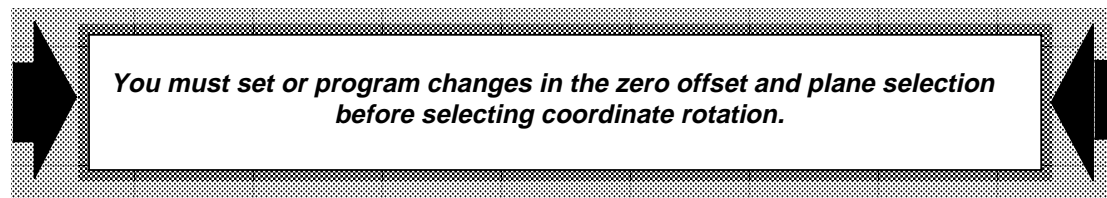


### Program control

- G53 or @706: Coordinate rotation and ZO can be suppressed block by block
- M02, M30: Programmable coordinate rotation and ZO are deleted, the adjustable coordinate rotation and ZO remain intact (setting data).

### Further characteristics:

- The coordinate rotation is executable in "AUTOMATIC" mode.
- The coordinate rotation is **channel specific**; in "SIMULATION" the coordinate rotations for **channel 3** (simulation channel) become effective.
- In "ACTUAL VALUE DISPLAY" the coordinate rotation is **not** taken into account.
- Circle interpolation may not be programmed immediately after a coordinate rotation
- When CRC is selected:
  - The centre of rotation (some of the ZO) must **not** be modified
  - The angle of rotation "A" of the adjustable and programmable coordinate rotation can be modified
- Coordinate rotation is associated with the function "Empty buffer until coordinate rotation" (@715). This function is initiated internally by block preparation and need not be programmed.



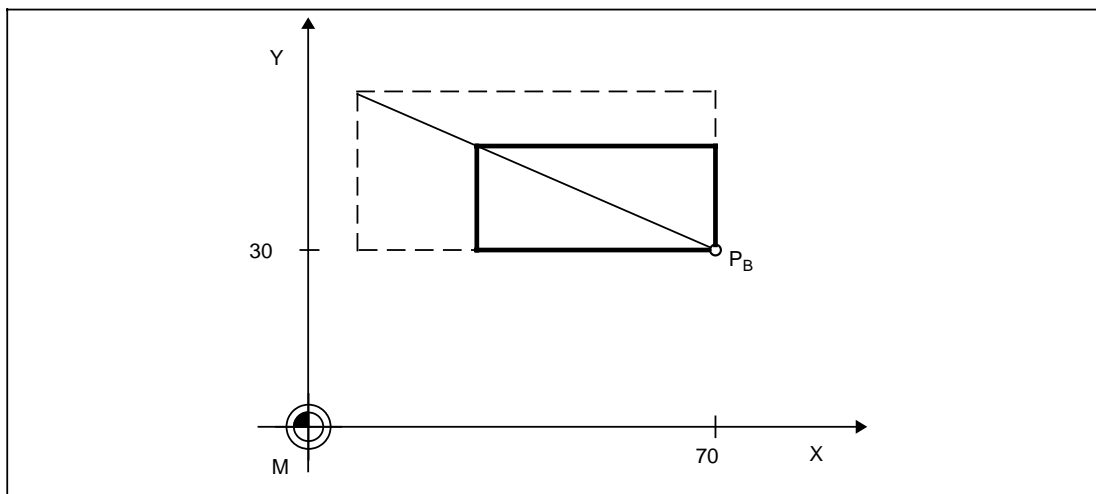
## 2.12 Scale modification: Selection G51, cancellation G50

With this function the programming effort for geometrically similar parts may be considerably reduced.

- A scale modification becomes effective with the programming of G51.
- G51 is modal and is only removed by G50 (cancellation of scale modification).
- The scale modification refers to a reference point  $P_B$  (scale centre). The coordinates (X, Y) of  $P_B$  are specified by programming. If X and Y are not specified, the control generates the values  $X=0$  and  $Y=0$  (reference point= workpiece zero).
- The value for the scale factor is specified at address "P..." in the range 0.00001 to 99.9999
- The following values are recalculated by scale modification:
  - Axis coordinates
  - Interpolation parameters
  - Radius
  - Programmable ZO
  - Thread gradient, gradient decrease or increase.
- Scale modification is connected with the function "Empty buffer" @ 714. This function is initiated internally.

**Example:** Possible representations in the program

- a) N...  
N... G51 X70 Y30 P1.5 L\_F  
N...  
OF
- b) N...  
N... G51 P1.5 L\_F  
N... G51 X70 Y30 L\_F  
N...



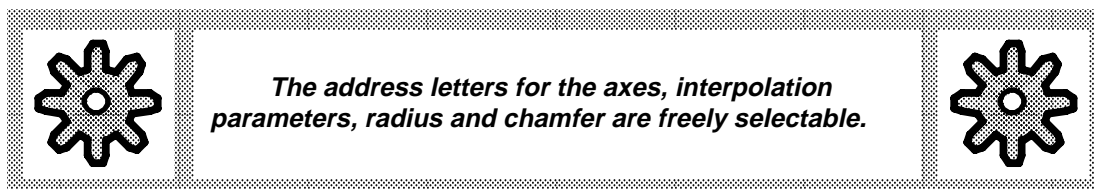
**Scale factor (channel-specific) and reference point are displayed on the screen under "SETTING DATA".**  
**The scale modification is enabled for the appropriate axis by setting data 560\*, bit 2.**

## 3 Programming of Motion Blocks

### 3.1 Axis commands

The address of the axis command determines the axis in which the following numeric value must be traversed, e. g. X, Y, Z.

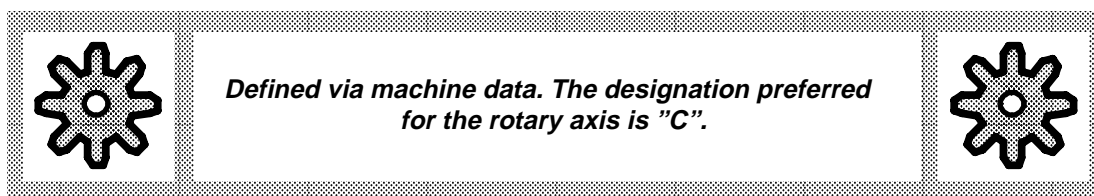
Addresses A, B, C, U, V, W and Q are optionally available for further axes. For these extended address notation is possible (e. g. C1 = ...).



Use is made below of the address letters used in preference for milling machines.

#### Rotary axis:

All axes can also be declared rotary axes.



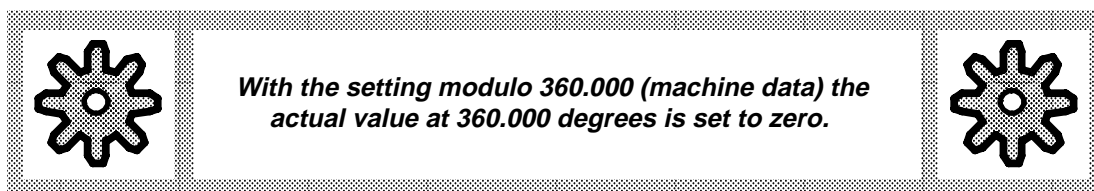
- The traversing range with absolute position data programming (G90) is  $\pm 360.000$  degrees.

The sign specifies the direction in which the axis is positioned at the absolute position within a revolution.

The traversing range by the shortest path (G68) with absolute position data programming is 0 to +360.000 degrees.

The control finds the shortest path from the current position to the programmed position so that the direction of travel is determined automatically. The first time the rotary axis is programmed with "G90" in the part program "G68" is automatically activated for this block. This automatic generation can be deselected with "G91 C0 L\_F".

The above notes on G68 are also valid when a "block search with calculation" is applied to a part program for the first time.

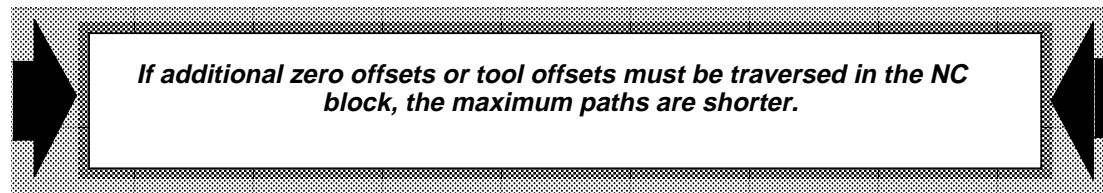


- The traversing range with incremental position data programming (G91) of  $\pm 99999.999$  degrees must be observed. The sign only specifies the direction of travel.

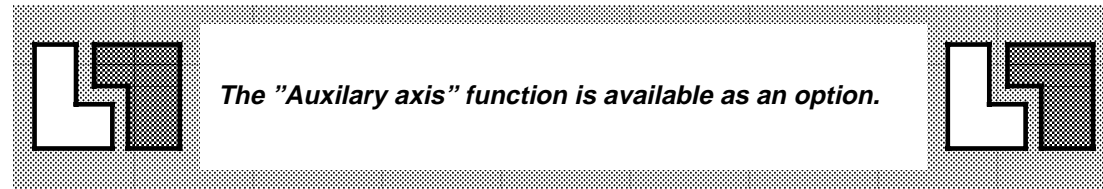
It is possible to define **several axes** as rotary axes **simultaneously**.  
The rotary axes can generally be rotated infinitely.

**Program section:**

```
.  
.   
.   
N5 G91 C 99999.999 L_F
```



**Auxiliary axes:**



The term "**auxiliary axis**" designates an axis used for workpiece or tool handling (loader, turret, magazine etc.) as opposed to actual workpiece machining.

Auxiliary axes may be located in the machining channel or in a separate NC channel. They have the same range of functions as an NC main axis (linear interpolation, circular interpolation, contour definition, tool offset, etc.).

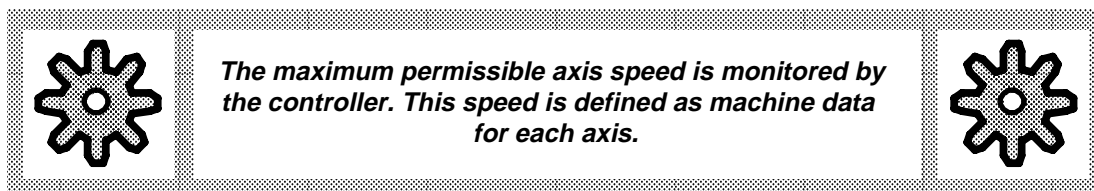
Provided the auxiliary axis is not programmed in the part program as the "**NC axis**", synchronization with the NC main axes will also be possible via the interface controller. The position measurements, traversing range, input resolution and geometrical and position control resolution correspond to those of an NC axis.

The axis addresses are freely selectable from the available addresses. They can be further distinguished using the default address Q and an extended address notation, e.g.: Q1, Q2 ...

If the auxiliary axis is located in a separate channel, it is not possible to program contour definition or tool offset. The NC axes and the auxiliary axes are then synchronized by the interface controller (application program).

### 3.1.1 Axis motion without machining G00

- Rapid traverse motions are programmed by means of the position data G00 and a target position specification. The target position can be reached by means of either an absolute position data input (G90) or an incremental position data input (G91).
- The path programmed using G00 is traversed at the **maximum possible speed** (rapid traverse) along a straight line **without machining** the workpiece (linear interpolation).



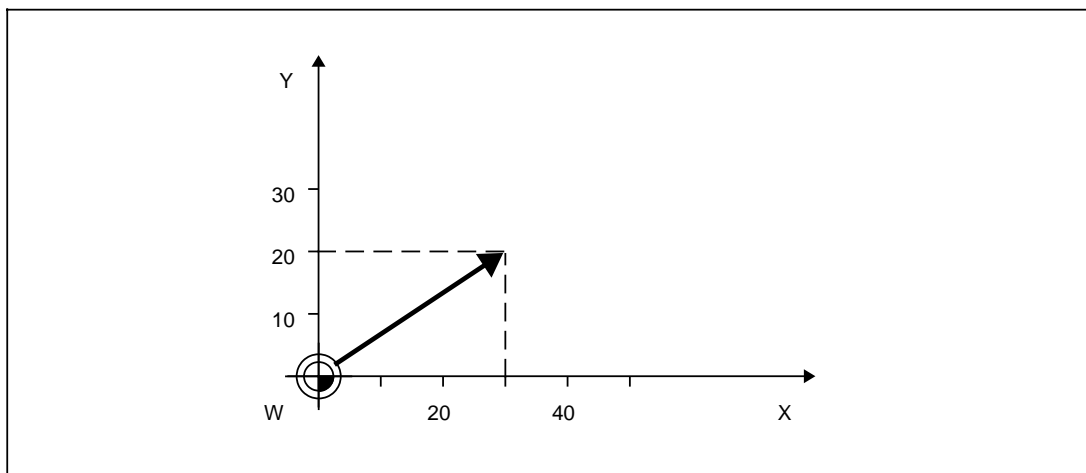
If the rapid traverse is executed simultaneously in several axes, the traversing speed is determined by the lowest axis speed specified in the MACHINE DATA. The preparatory function G00 automatically initiates "coarse exact positioning".

When G00 is programmed, the feedrate programmed under address F remains stored; it can be made active again, for example, by means of G01.

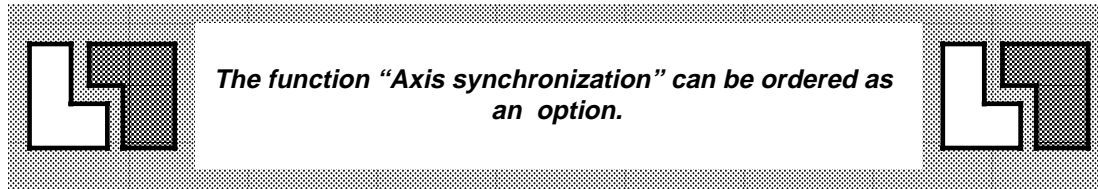
#### Example:

N5 G00 G90 X30 Y20 L<sub>F</sub>

Rapid traverse, absolute position specified



### 3.1.2 Axis synchronization



The "Axis synchronization" function duplicates the programmable main axes without exceeding the maximum number of axes (on the 810 T/M, 820 T/M: five axes).

In this way it is possible to machine two identical workpieces on a machine tool with two tool systems with one part program running on one channel of the NC.

The "Axis synchronization" function is only available in the "AUTOMATIC", "MDI AUTOMATIC" and "AUTOMATIC interrupted" modes.

With axis synchronization there is a measuring circuit for each axis, for the synchronized axes as well. For this reason, not more than two axes can be synchronized. The fifth axis is the available as an independent axis.

Please consult the machine manufacturer for the axis designations with the "Axis synchronization" function.

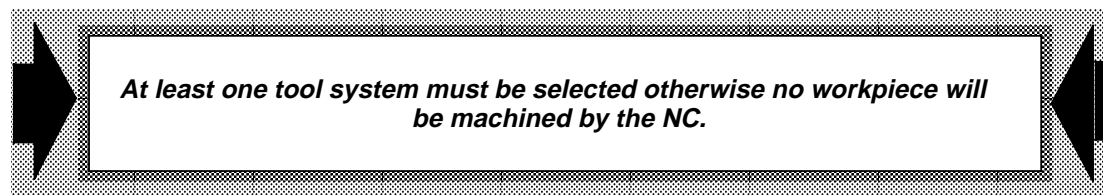
#### 3.1.2.1 Function

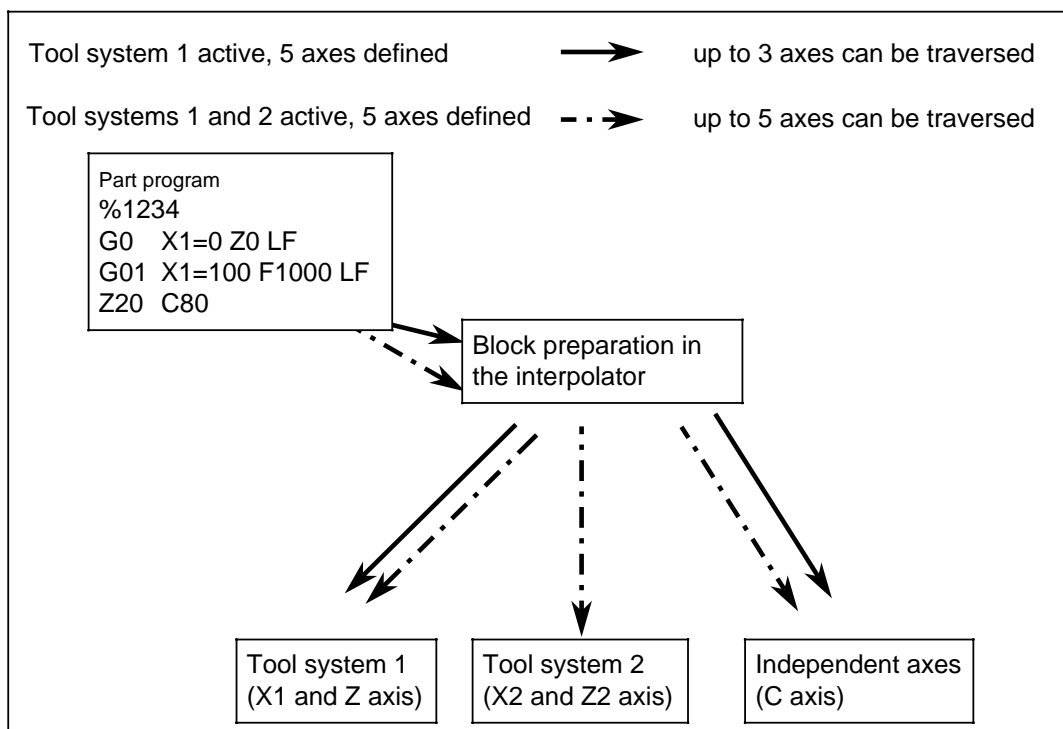
With the "Axis synchronization" function the NC executes one part program in **two separate tool systems** simultaneously. In the part program only one tool system is programmed (tool system 1).

For example, if on a turning machine the **leading (programmed) axes** are the X1 axis and the Z axis in **tool system 1**, the **following (synchronized) axes** are the X2 and Z2 axis in **tool system 2**.

The axes of tool system 2 must **on no account** be programmed in the part program. The names and traverse paths for the axes in tool system 2 are derived from the program for tool system 1. The same goes for the plane selection and the geometry assignment of the tools. Please see the machine manufacturers instructions for how to select tool system with "Axis synchronization".

After NC start the function selected via the PLC (tool system 1 or tool system 2 or both active) is activated and stored. The selected function is modal until the next end of program or RESET and cannot be changed during program execution (causes alarm).



**Part program with or without axis synchronization:**

What happens when tool system 1 is active:

**Part program**

```
N0010 G0 X1=0 Z D5 LF
N0020 G01 X1=100 F2000 LF
N0030 Z30 C50 LF
N0040 G33 X1=50 I2 LF
N0050...
```

**Result**

Axes X1 and Z move to 0 at rapid traverse rate, the selected tool is D5.  
Axis X1 moves to position 100 mm at feedrate 2000 mm/min.  
Axis Z and C (rotary axis) interpolate to position 30 mm and 50 degrees respectively at feedrate 2000 mm/min.  
Thread cutting with axis X1.

What happens when tool system 1 and 2 are active (axis synchronization):

**Part program**

```
N0010 G0 X1=0 Z D5 LF
N0020 G01 X1=100 F2000 LF
N0030 Z30 C50 LF
N0040 G33 X1= 50 I2 LF
N0050...
```

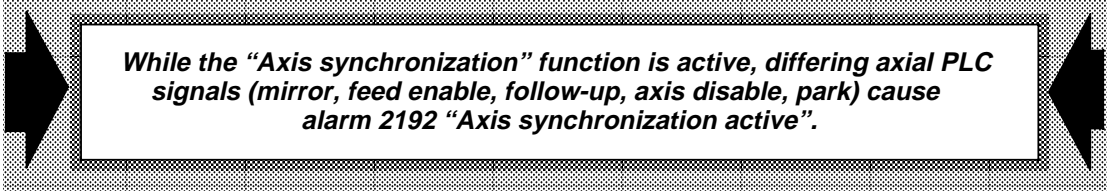
**Result**

Axis X1/X2 and Z/Z2 move to 0 at rapid traverse rate, the selected tool is D5 in tool system 2.  
Axis X1/X2 move to position 100 mm at feedrate 2000 mm/min.  
Axis Z/Z2 and C (rotary axis) interpolate to position 30 mm and 50 degrees respectively at feedrate 2000 mm/min.  
Thread cutting with axis X1/X2.



### Characteristics of the “Axis synchronization” function:

- The programmed set velocity applies to both tool systems.
- Handwheels and software limit switches are axial.
- The working area limitations G25/G26 are duplicated so that they apply in tool systems 1 and 2.
- The programmable zero offsets G58, G59 also apply to the duplicated axes. In the “Programmed zero offsets” display the value of the zero offset is always shown for **both** axes, the programmed and the duplicated axis.
- With the “Second reference point” and “Mirror zero offset” functions complete machining of a workpiece is also possible (see next page).
- A scale factor in tool system 1 is also valid in tool system 2.



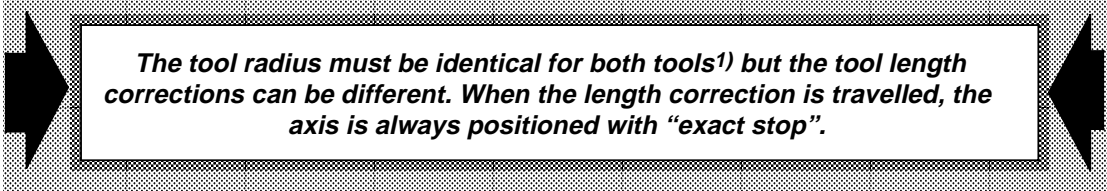
***While the “Axis synchronization” function is active, differing axial PLC signals (mirror, feed enable, follow-up, axis disable, park) cause alarm 2192 “Axis synchronization active”.***

### Tool management, tool offset:

When “Axis synchronization” is being used, the tool memory is divided into two. Tool numbers D1 to D49 apply to tool system 1 and tool numbers D51 to D99 apply to tool system 2. The offset between tool system 1 and tool system 2 is thus 50.

#### Example:

D3 is programmed for tool system 1. The control automatically selects D53 for tool system 2 when the command “D3” in the program is processed.



***The tool radius must be identical for both tools<sup>1)</sup> but the tool length corrections can be different. When the length correction is travelled, the axis is always positioned with “exact stop”.***

#### Note:

While axis synchronization is active and MD 5011 bit 3 = 1 or bit 4 = 1, length P2 and length correction P5 are calculated as diameters for P1 = 0.

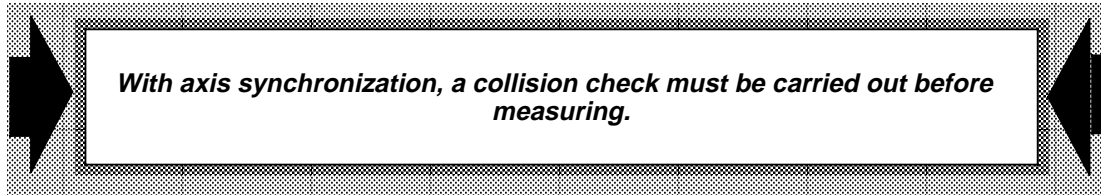
---

1) The tool radius of tool system 1 applies both to tool system 1 and tool system 2.

**@ commands:**

With the exception of @440 (programmed axis position) all axis-related @ commands function as previously.

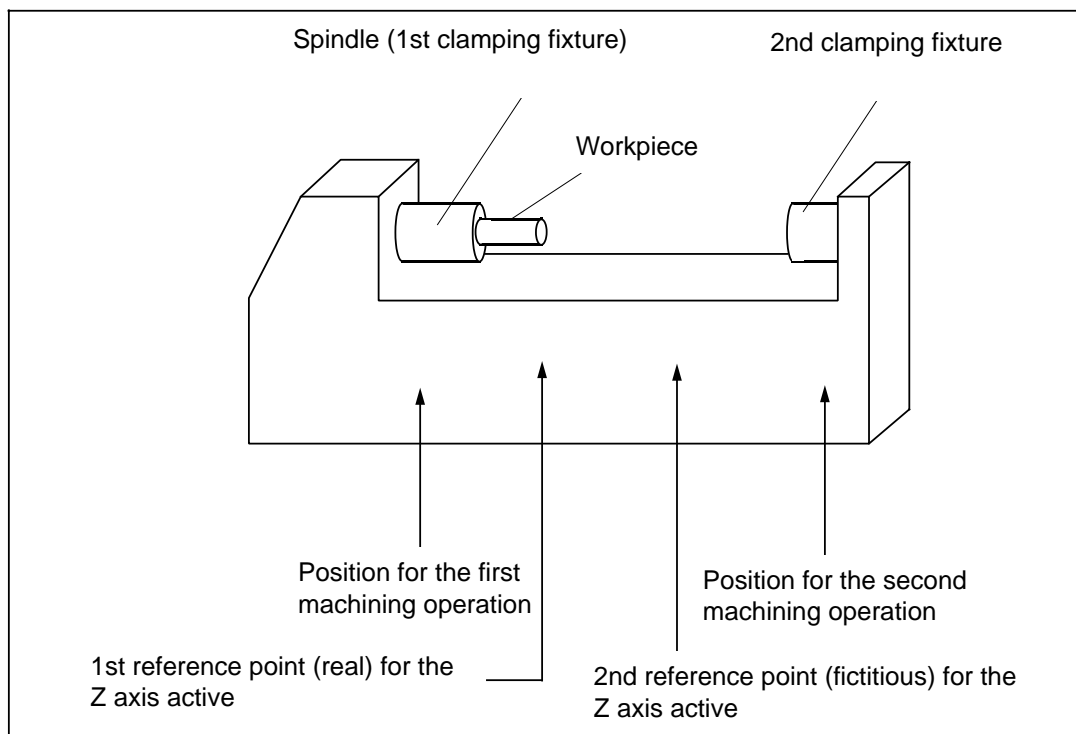
With @720 (in-process measurement) the programmed path is duplicated but only one actual value is measured. Both axes stop when the probe is triggered.

**Setting data (SD):**

The option bits for the scale factor applying to tool system 1 also apply to tool system 2. As for the zero offsets, the notes under “characteristics” apply.

**3.1.2.2 Complete machining**

For complete machining of a workpiece there are machine data not only for mirroring the zero offsets but also for a “second fictitious reference point value”. The difference between the “reference point value” and the “second fictitious reference point value” is set by the PLC as an external zero offset. The part of the NC program which follows must then refer to this reference point.



## 3.2 Axis motions with machining

The controller implements linear interpolation or circular interpolation, depending on the type of axis motion:

- **Linear interpolation:**  
Linear motion
  - paraxial
  - in two axes
  - in three axes
- **Circular interpolation:**  
Circular motion in 2 axes (plane).

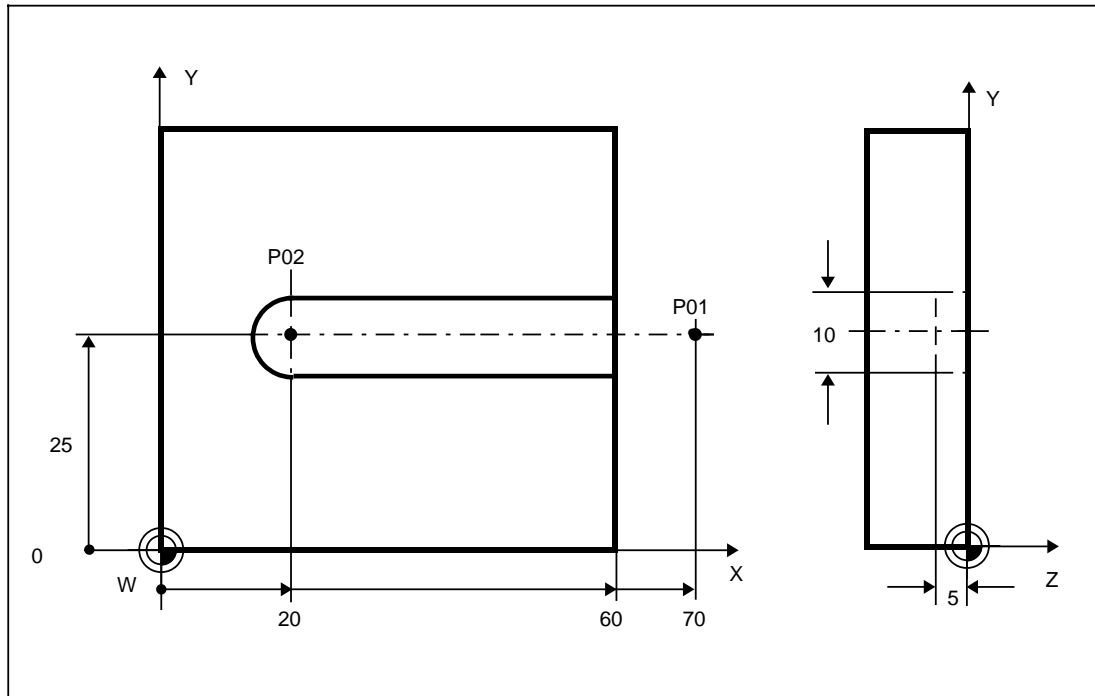
### 3.2.1 Linear interpolation G01

The tool must travel at a set feedrate along a straight line to the target position whilst machining the workpiece at the same time. The controller calculates the tool path by means of linear interpolation.

**Linear interpolation** effects motion

- in one axis direction (linear axis or rotary axis),
- from the starting position to the target position programmed using absolute or incremental position data,
- at the programmed feedrate,
- at the programmed spindle speed.

Paraxial movements and movements at any angle can be executed.

**Example:** Paraxial milling

```
%15 LF
```

```
N5 G00 G90 X70 Y25 Z1 S800 M3 LF
```

```
N10 Z-5 LF
```

```
N15 G01 X20 F150 LF
```

```
N20 G00 Z100 LF
```

```
N25 X-25 Y50 LF
```

```
N30 M30 LF
```

Spindle On, tool moves at rapid to P01,  
clockwise rotation 800 rev/min

Infeed in Z

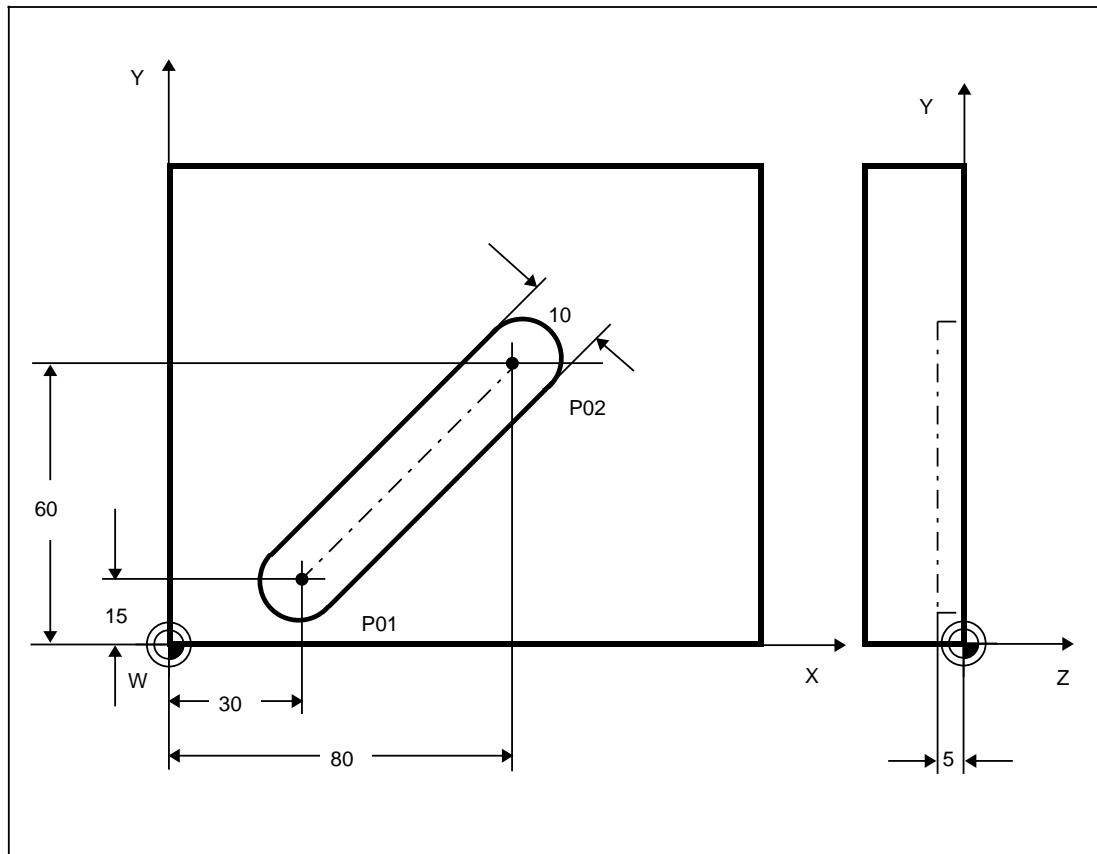
Tool moves from P01 to P02, feedrate 150 mm/min

Rapid traverse retraction

Rapid traverse retraction

End of program

**Example:** Linear interpolation in 2 axes



%25 L<sub>F</sub>

N5 G00 G90 X30 Y15 Z1 S500 M3 L<sub>F</sub>

N10 G01 Z-5 F100 L<sub>F</sub>

N15 X80 Y60 F200 L<sub>F</sub>

N20 G00 Z100 L<sub>F</sub>

N25 X-40 Y100 L<sub>F</sub>

N30 M30 L<sub>F</sub>

Tool rapid traverse to point P01 with coordinates (X30, Y15)

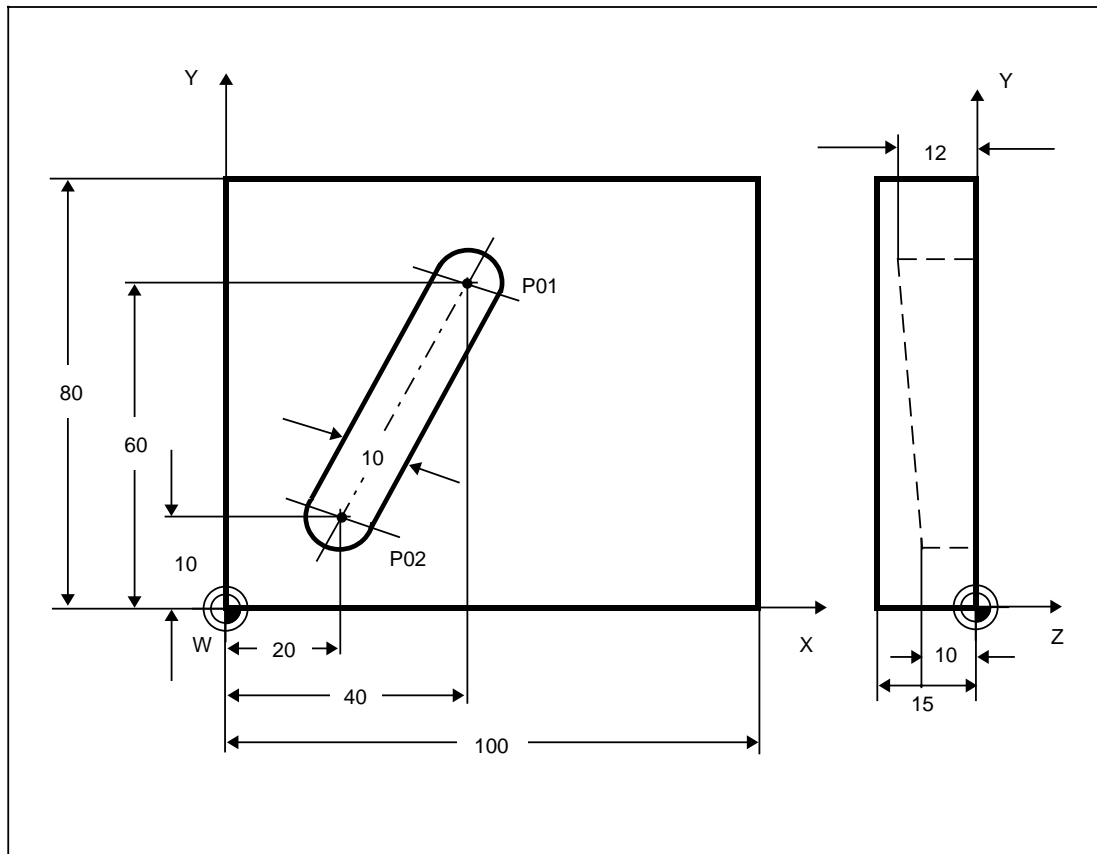
Infeed to depth Z-5, feedrate 100 mm/min

Tool traverse along a straight line from P01 to P02,  
feedrate 200 mm/min

Rapid traverse retraction

Rapid traverse retraction

End of program

**Example:** Linear interpolation in 3 axes

%30 L<sub>F</sub>

N5 G00 G90 X40 Y60 Z2 S500 M3 L<sub>F</sub>

N10 G01 Z-12 F100 L<sub>F</sub>

N15 X20 Y10 Z-10 L<sub>F</sub>

N20 G00 Z100 L<sub>F</sub>

N25 X-20 Y80 L<sub>F</sub>

N30 M30 L<sub>F</sub>

Tool rapid traverse to P01

Infeed to Z-12, feedrate 100 mm/min

Tool traverse along a straight line in space from P01 to P02

Rapid traverse retraction

Rapid traverse retraction

End of program

### 3.2.2 Circular interpolation G02/G03

The tool must traverse between two points on the contour in a circular arc, whilst simultaneously machining the workpiece.

The controller calculates the tool path by means of circular interpolation.

Circular interpolation effects tool motion:

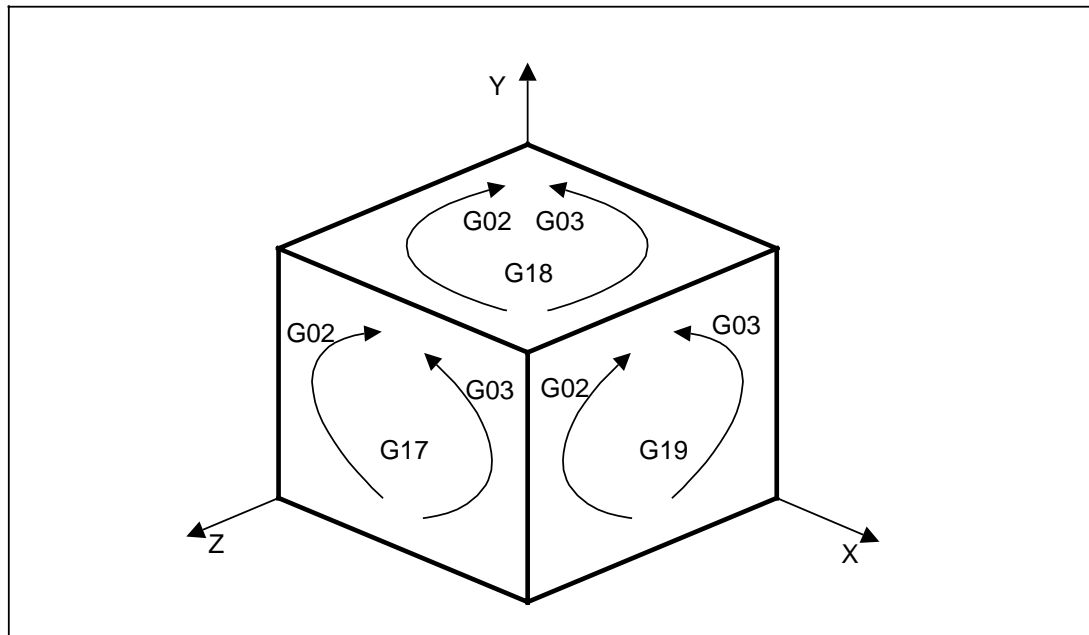
- along a circular arc  
in a **clockwise direction** with **G02**,  
in a **counterclockwise direction** with **G03**,
- about the programmed centre point of the circle,
- from the starting position on a circular path to the programmed end position.

The action of preparatory functions **G02** and **G03** is **modal (storing)**.

A circular motion can be executed in any plane if selected accordingly (XY, ZX or YZ).

The circular plane is determined by the axes first programmed in the circular block. If only one axis position is programmed (e.g. semi-circle) the circle plane is defined by the plane selected. If the plane and interpolation parameters defined in machine data are not identical, alarm 3006 "incorrect block structure" is triggered.

The direction of rotation in the various planes is defined as follows: Stand facing the axis perpendicular to the plane. The tool will move in a clockwise direction with G02 or in a counterclockwise direction with G03.



*Circular interpolation*

The interpolation parameters determine the circle or circular arc together with the axis commands:

- The starting position “**KA**” on the circle or circular arc is defined by the preceding block.
- The end position “**KE**” is defined by axis values X, Y and Z.
- The centre point of the circle “**KM**” is defined in one of the following ways:
  - By the interpolation parameters,
  - Directly using the radius (U).

### 3.2.2.1 Interpolation parameters I, J, K

The interpolation parameters are the paraxial coordinates of the distance vector from the starting position to the centre point of the circle.

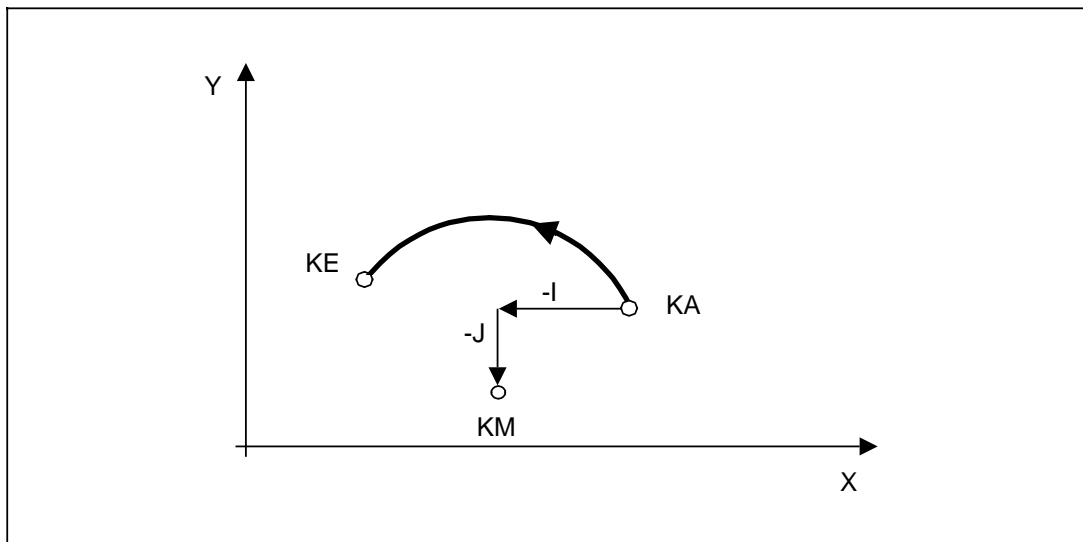
According to DIN 66025 interpolation parameters I, J and K are allocated to axes X, Y and Z. The interpolation parameters must always be entered as **incremental position data** and in the **correct sequence**, irrespective of whether X, Y and Z are programmed using absolute or incremental position data (selectable via MD).

The sign depends on the direction of the coordinates from the starting position to the centre point of the circle.

No programming is required:

- if the value of an interpolation parameter is 0,
- if the end position coordinates are the same as for the starting point of the circular path.

At least one axis must be programmed for a full circle (X0, Y0 or Z0).

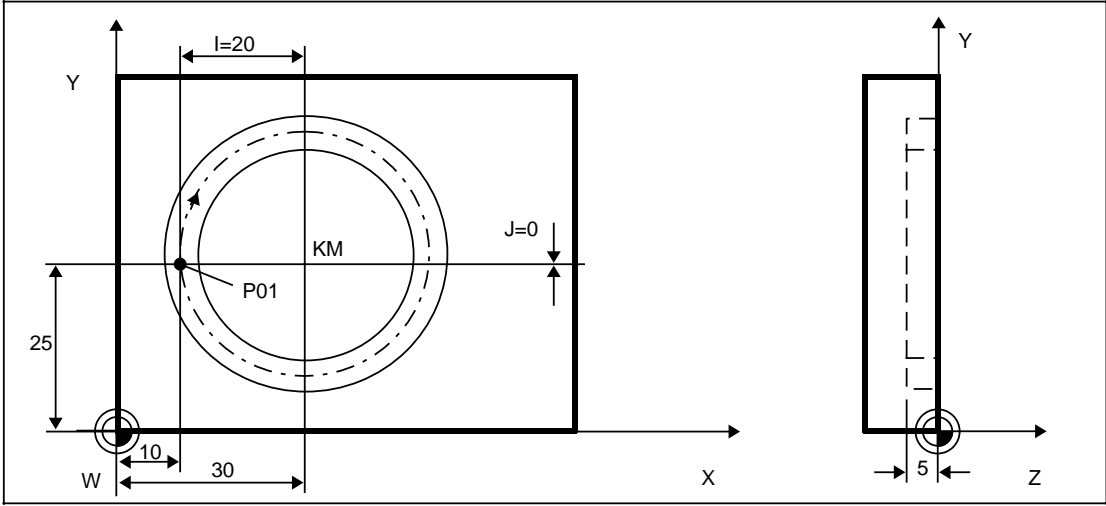


*Circular interpolation with interpolation parameters*



**Example:** Full circle in X-Y plane

The example below shows programming of a full circle. In this case the starting position must also be entered as the end position.



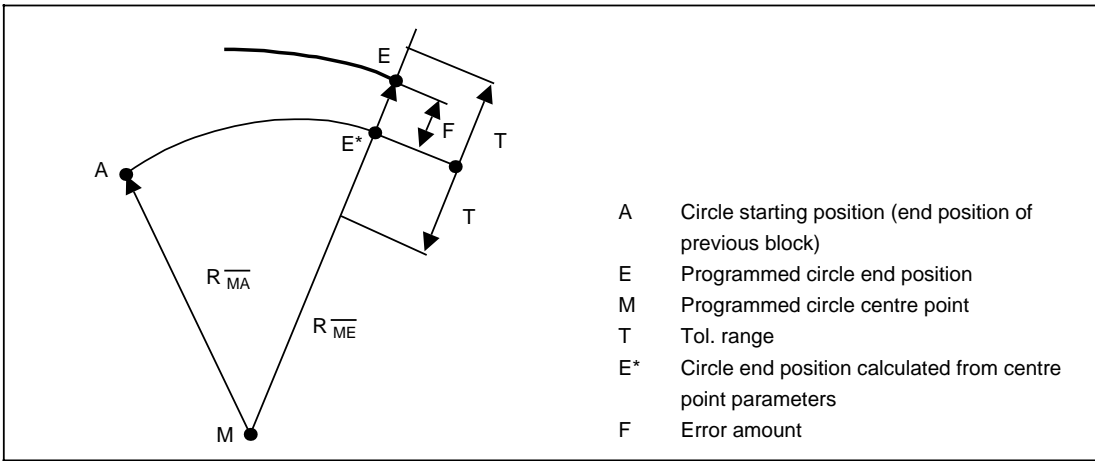
```
%40 L_F
N5 G00 G90 X10 Y25 Z1 S1250 M3 L_F
N10 G01 Z-5 F100 L_F
N15 G02 X10 Y25 I20 J0 F125 L_F

N20 G00 Z100 M5 L_F
N25 X-20 Y60 L_F
N30 M30 L_F
```

Tool rapid traverse to point P01  
Infeed to Z-5, feedrate 100 mm/min  
X-Y plane selected automatically (reset). Tool clockwise traverse around a full circle (G02)  
Rapid traverse clear  
Rapid traverse clear  
End of program

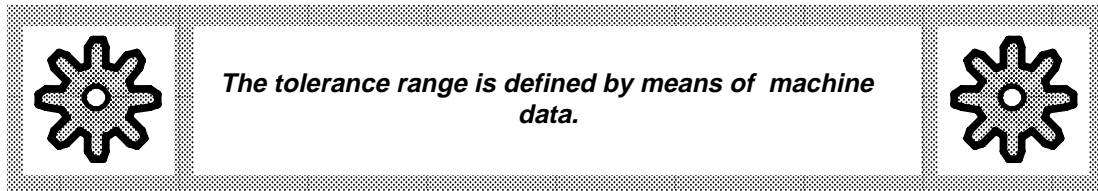
**Circle end position monitoring:**

Prior to processing a circle block, the NC checks to ensure that the programmed values are correct by determining the **difference in radii** for starting position **A** and end position **E**.



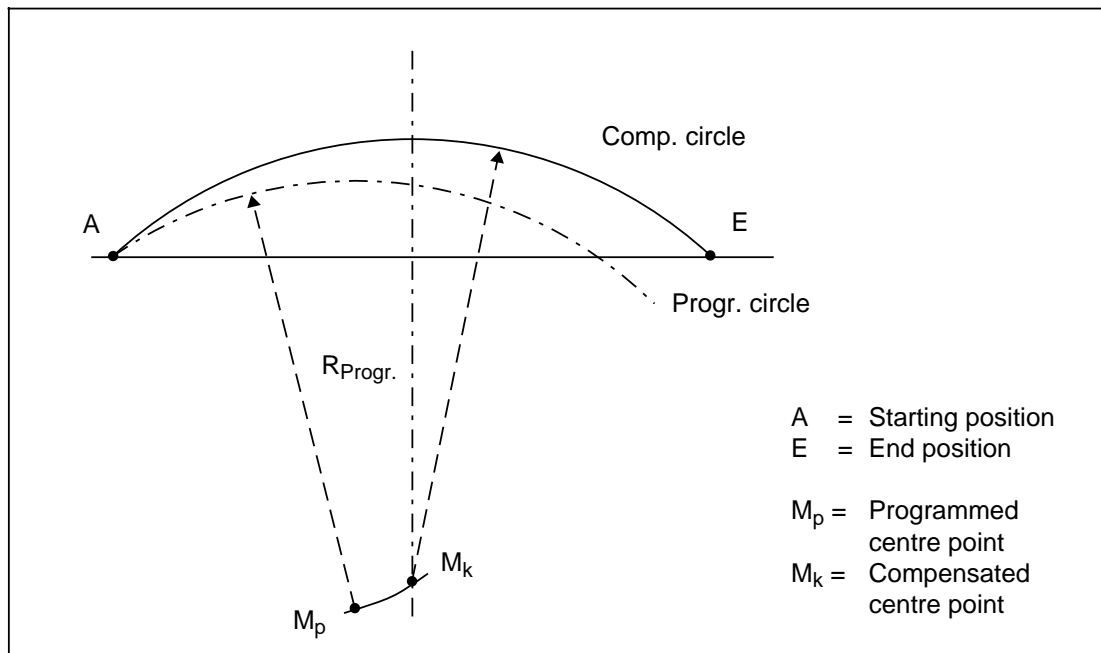
Definition of tolerance range of circle end point monitoring

If, due to imprecise input of the radius or the interpolation parameters, the difference (F) exceeds the tolerance range (T), then the circle block is not processed. The control outputs alarm "circle end position error" alarm is output.



If the difference in the radii is within the tolerance range, the centre point parameters are corrected since it is assumed that the circle end position has been "properly" programmed.

The circle block is then executed with the new, **compensated centre point**.



Principle of circle correction

### 3.2.2.2 Radius programming

In many cases the dimensions of a drawing are such that it is easier to specify radius B when defining the circular path. Extended address notation for the radius is not possible. The end point and radius B are programmed in radius programming.

G02 or G03 determines the direction of movement about the circle defined by means of the circle end position and the interpolation parameters or by means of radius U. Because the radius specified gives an unambiguous circular path only within a semicircle, it is necessary to specify whether the angle is less than or greater than 180 degrees.

Thus the radius is given the following sign:

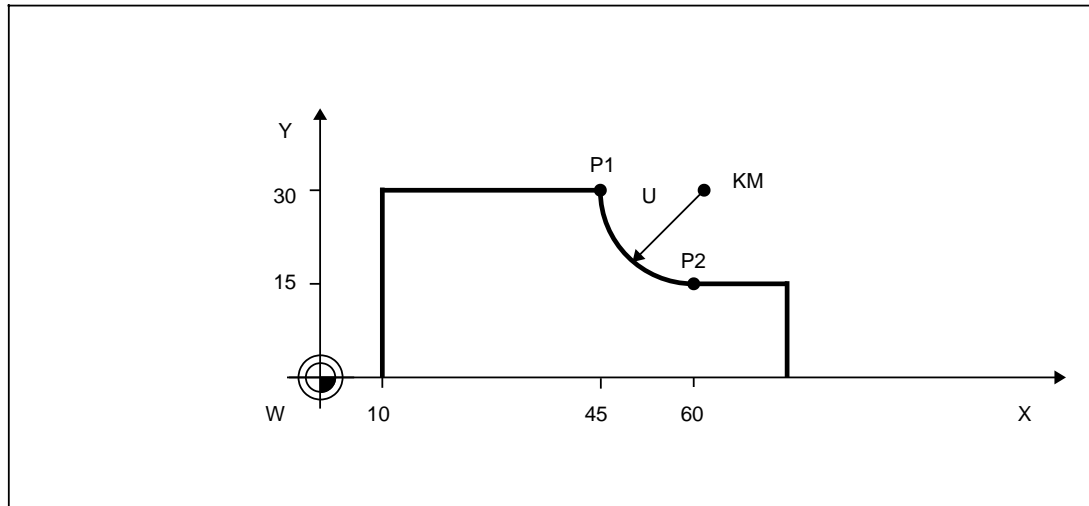
+U: Angle less than or equal to 180°

–U: Angle greater than 180°.



It is not permissible to program the radius if the traversing angle is 0° or 360°. Full circles must therefore be programmed using interpolation parameters.

#### Example: Radius programming



```

.
.
N5  G03 G90 X60 Y15 U15 F500 L_F    Tool machining from point 1 to point 2
N10 G02 X45 Y30 U15 L_F             Tool machining from point 2 to point 1
.
.

```

### 3.2.3 Helical interpolation

Helical interpolation is possible between 3 linear axes which are perpendicular to one another.

Helical interpolation involves programming a circular arc and a straight line perpendicular to the end position of this circular arc in a single block. When the program is processed the motions of the axis slides are coupled such that the tool describes a helix with a constant lead.

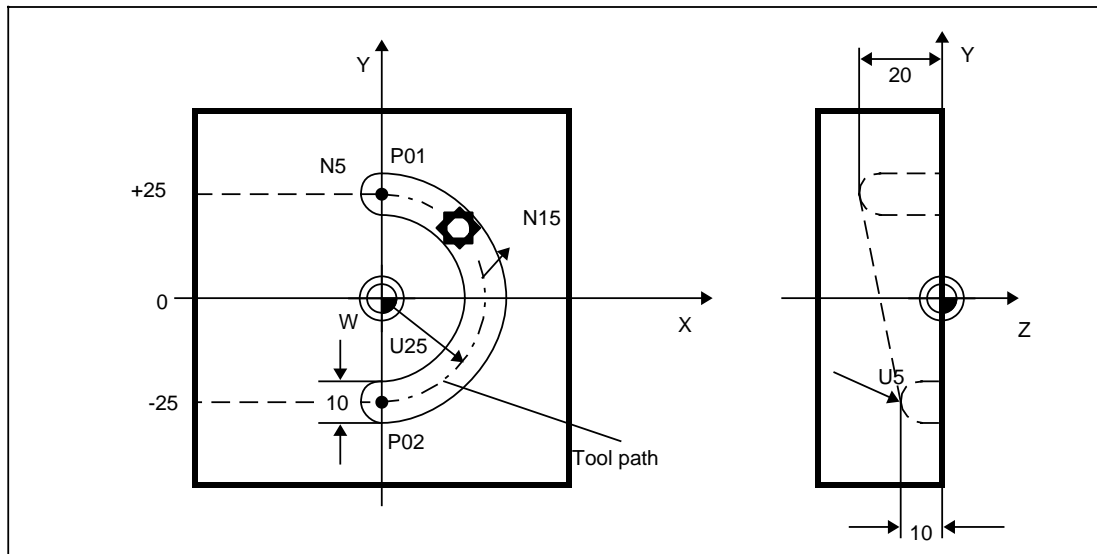
The circular plane is defined by the axes first programmed in the circular block. If the plane and interpolation parameters defined in MACHINE DATA are not identical, alarm 3006 "incorrect block structure" is triggered.

The axis coordinates X, Y and Z must always be programmed. The axis for linear interpolation can be programmed either before or after the interpolation parameters. If another axis is incorporated in the circular interpolation, the interpolation parameter specified via MACHINE DATA must be used for this axis.

The programmed feedrate is adhered to on the circular path rather than on the actual tool path. The required feedrate must be programmed before or in the helical block.

The programmed feedrate acts on the circular axes only, whilst control of the linear axes motions so that they reach their end point when the circular axes reach the circle end point.

The maximum permissible speed of the axes used for interpolation is monitored, and the feedrate is restricted when the speed limit is exceeded (i.e.  $V_{prog} > V_{max}$ ).

**Example:** Helical interpolation

```
%80 L_F
```

```
N5 G00 G90 X0 Y25 Z1 S800 M3 L_F
```

```
N10 G01 Z-20 F150 L_F
```

```
N15 G02 X0 Y-25 Z-10 I0 J-25 L_F
```

```
N20 G00 Z100 M5 L_F
```

```
N25 M30 L_F
```

XY plane selected automatically (reset)

Tool rapid traverse to point P01

Linear interpolation. Infeed to Z-20

Tool clockwise traverse on a helix (G02) from P01 to P02.

Rapid traverse clear

End of program

### 3.2.4 Cylindrical interpolation

*The "cylindrical interpolation" function is an ordering data option.*

Cylindrical interpolation permits machining of cylindrical paths with one **rotary** axis and one **linear** axis at a constant rotary table diameter. Both linear and circular contours may be programmed, but circles can only be programmed with circle radius programming (see Section 3.2.2.2). It is not possible to input the interpolation parameters I, J and K. SPLINE interpolation (see Section 3.2.11) is not possible either. The position of the rotary axis is entered in degrees. It is converted to the circumferential dimensions of the working diameter internally in the control. The ratio is programmed under G92 P ... for this purpose.

The control forms the ratio from the machining diameter and the unit diameter as follows:

$$P = \frac{\text{machining diameter}}{\text{unit diameter}}$$

#### Input system:

Unit diameter

- 114.592 mm in the metrical system with input resolution  $10^{-3}$  mm
- 114.592 inches with input resolution  $10^{-3}$  mm.

The unit diameter is derived from the relation  $\pi \cdot d = 360$ :  
( $\cdot$  = Multiplication character).

$$\text{Unit diameter} = \frac{360}{\pi} \text{ in mm or inches}$$

No characters other than the axis name must be written in a block containing G92P...

N.. G92 P.. C LF

P.. Factor for unit circle, **P =working diameter/unit diameter**

C Rotary axis.

The input resolution for P is  $10^{-5}$  and the value of P is  $< 100$ . The action of the factor for the unit circle is modal until reprogrammed or reset with M02/M30 or RESET.

The programmed feedrate is adhered to on the contour.

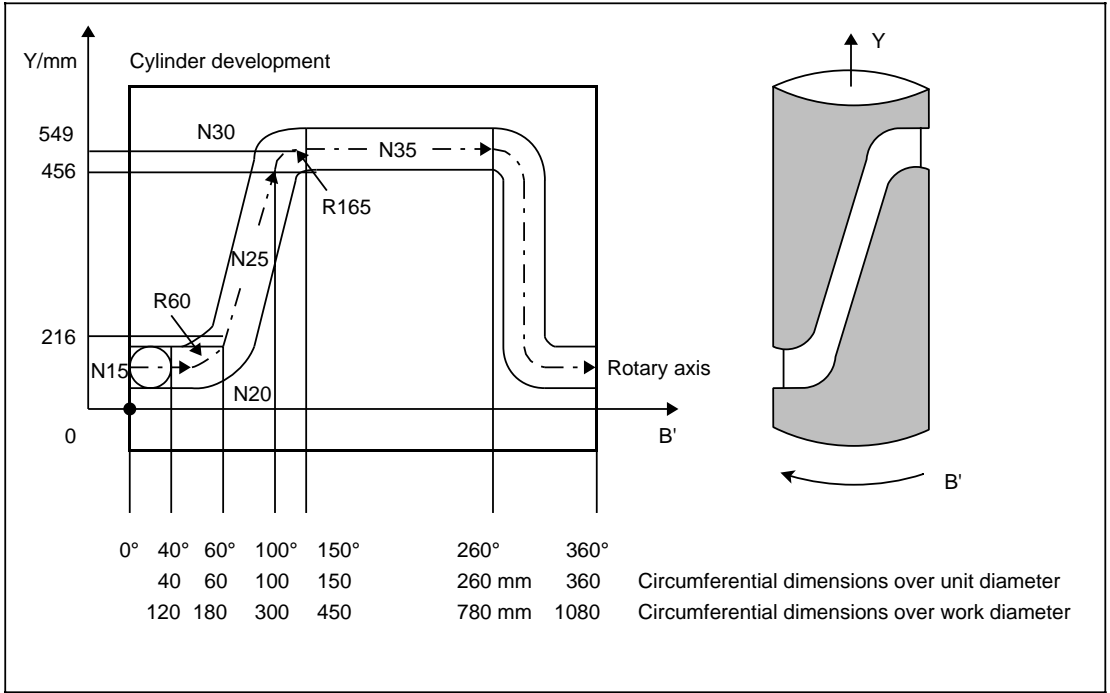
As long as the machining diameter is not equal to the unit diameter, this axis (e.g. C) can only be interpolated with one further axis, i.e. for interpolation with more than 2 axes, the machining diameter must be equal to the unit diameter (factor  $P = 1$ ).

For  $P < 1$ , it is important to note that the interpolation resolution of the rotary axis is  $1/p$  greater with cylindrical interpolation than without it.

While cylindrical interpolation is selected, the rotary axis must not be programmed with G68 (only G90,G91).

You can use cylindrical interpolation together with CRC/TNRC. It is possible to select and to deselect cylindrical interpolation while CRC/TNRC is selected. The function “soft approach/exit” cannot be used together with cylindrical interpolation.

Example: Cylindrical interpolation



Cylindrical interpolation

```
%10 LF
N10 G92 P3 C LF           Selection of cylindrical interpolation
N15 G01 B40 Y.. F1000 S800 M3 LF
N20 G03 B60 Y216 U+60 LF   Radius R60 (U60)
N25 G01 B100 Y456 LF
N30 G02 B150 Y549 U+165 LF Radius R165 (U165)
N35 G01 B260 LF
:
N60 G92 P1 C LF           Cancellation of cylindrical interpolation
N70 M02 LF
```

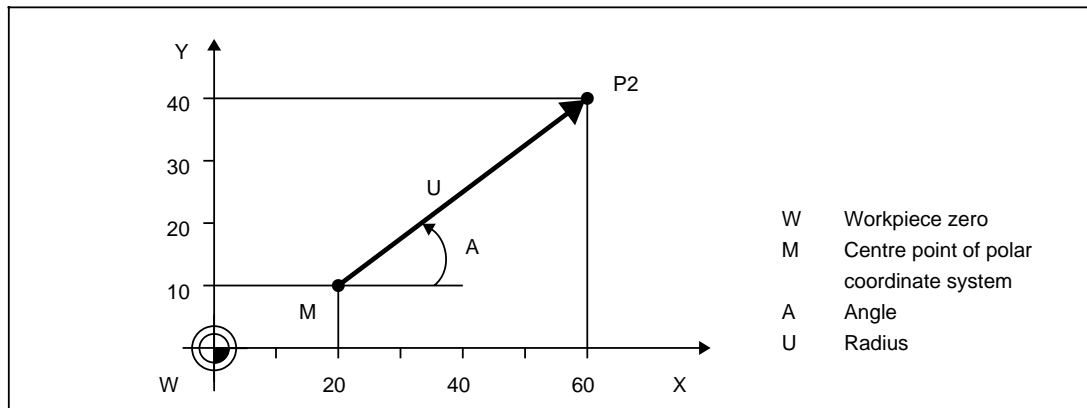
### 3.2.5 Polar coordinates G10/G11/G12/G13

Drawings dimensioned with an **angle** and a **radius** can be entered **directly** in the program with the aid of the polar coordinates.

The following preparatory functions are available for **programming with polar coordinates**:

- G10** Linear interpolation, rapid traverse
- G11** Linear interpolation, feedrate (F)
- G12** Circular interpolation, clockwise
- G13** Circular interpolation, counterclockwise

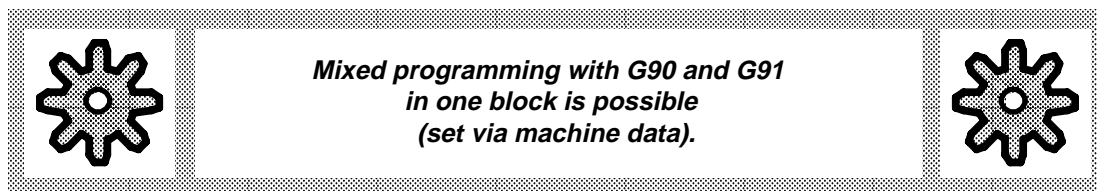
The action of the preparatory functions is **modal**.



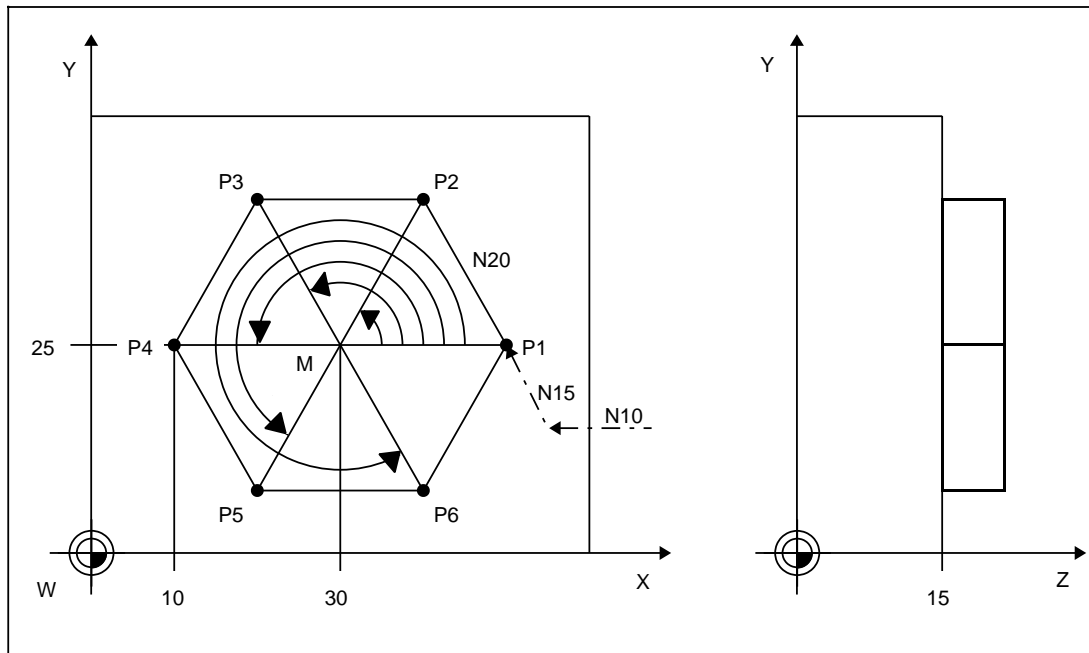
In order to determine the traverse path, the controller requires the **centre point**, the **radius** and the **angle**. The centre point is entered with perpendicular coordinates (X, Y, Z) and - on initial programming - using absolute position data. A subsequent incremental position data input (with G91) always refers to the last centre point programmed.

The centre point entry is modal and can be reset by means of M02/M30.

- The **radius** is programmed under the address **U** without a sign.
- The **angle** is entered under the address **A** without a sign (input resolution  $10^{-5}$  degrees). It always refers to the first positive axis of the centre point coordinates to be programmed (reference axis).
  - The positive direction of this axis corresponds to an angle of  $0^\circ$ .
  - Angles are specified as absolute or incremental and positive data.





**Example:** G11, milling machine

:

N10 G90 G0 Z15 L\_F

N15 G11 X30 Y25 U20 A0 F100 L\_F

N20 A60 L\_F

N25 A120 L\_F

N30 A180 L\_F

N35 A240 L\_F

N40 A300 L\_F

N45 A0 L\_F

N50 G0 Z100 L\_F

.

Infeed in Z

(P1) Approach milling position P1 (polar coordinates)

(P2)

(P3)

(P4)

(P5)

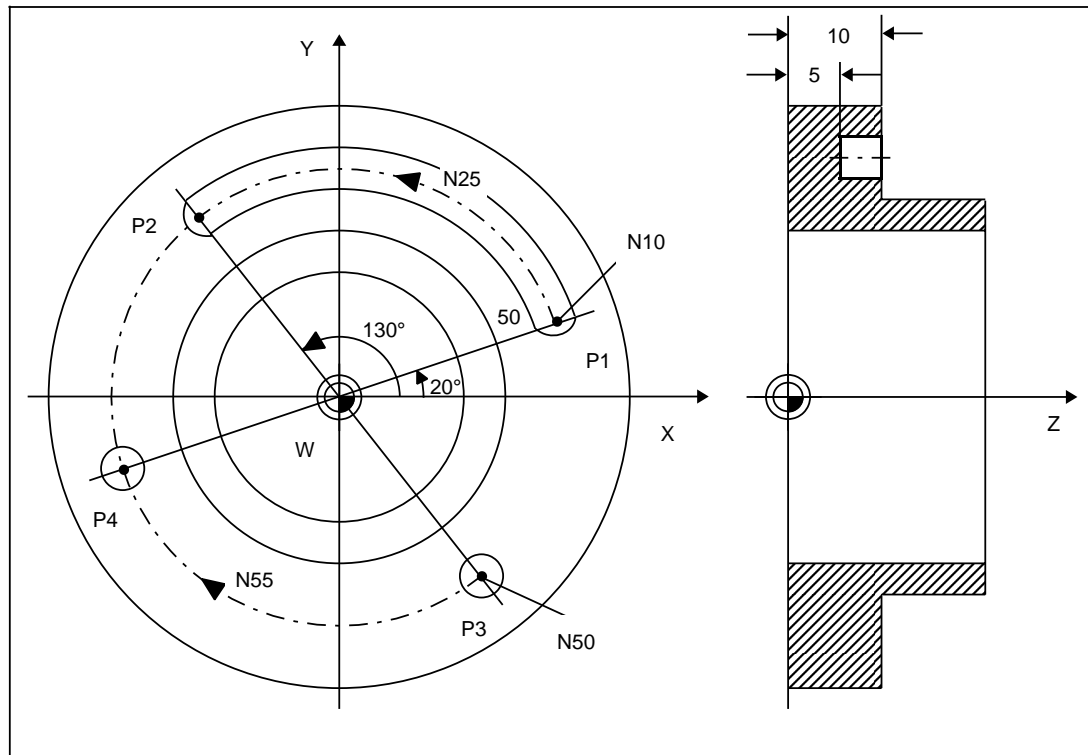
(P6)

(P1)

} Mill hexagon

Rapid traverse clear

**Example:** G10, G12, G13 milling machine



N05 G90 G0 X200 Y300 Z100 L <sub>F</sub>		
N10 G90 G10 X0 Y0 U50 A20 L <sub>F</sub>	(P1)	Approach milling position P1 (polar coordinates)
N15 G0 Z10 L <sub>F</sub>		
N20 G1 Z5 F1000 S800 M3 L <sub>F</sub>		Infeed in Z
N25 G13 A130 L <sub>F</sub>	(P2)	Slot cutting
N30 G0 Z100 L <sub>F</sub>		
N35 G0 X100 Y100 M5 L <sub>F</sub>		Approach tool change point
N40 T2 M6 L <sub>F</sub>		Change tool
N45 Z20 S800 M3 D2 L <sub>F</sub>		
N50 G81 G10 A310 R2=.. R3=.. L <sub>F</sub>	(P3)	Approach first drilling position and call drilling cycle
N55 G12 A200 R2=... R3=... L <sub>F</sub>	(P4)	Approach second drilling position, automatic call of drilling cycle
N60 G80 G0 Z100 L <sub>F</sub>		Cancel drilling cycle and withdraw drill
N65 M30 L <sub>F</sub>		

### 3.2.5.1 Polar coordinates G110/G111

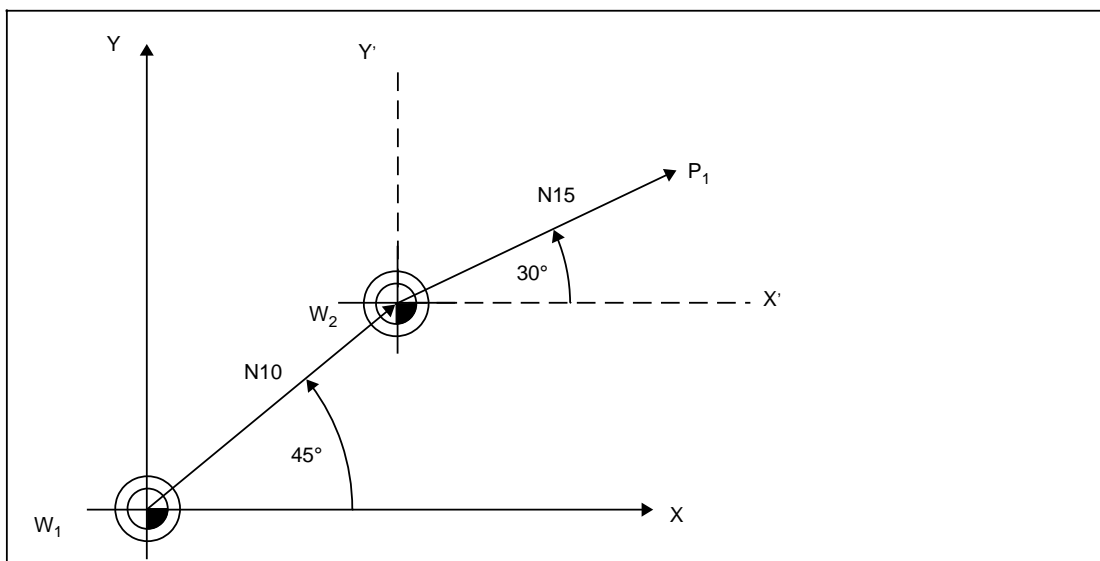
The functions G110 and G111 are used to adopt a new **centre point or zero point** when programming polar coordinates.

Using the new centre point, the angles are again taken from the horizontal and the radius is calculated from the new centre point. G110 and G111 have the following meanings:

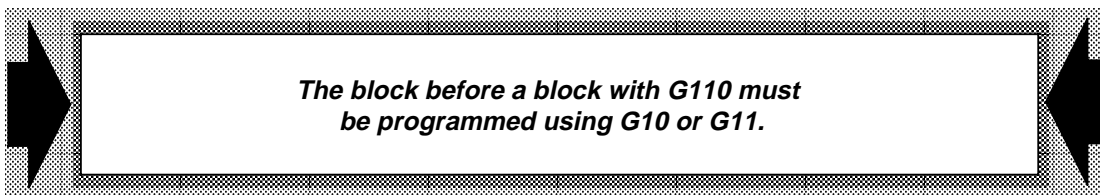
- G110** Adopt the setpoint reached as the new centre point  
**G111** Centre point programming with angle and radius without axis movement  
 (example: setting the arc center of a hole circle)  
 (the following traversing movement must be programmed using G110)

**Example:** Polar coordinates G110

```
%385
(G110 Polar coordinates) L_F
N5  G90 G10 X0 Y0 U0 A0 F1000 L_F
N10  G11 U30 A45 L_F
N15  G110 U20 A30 L_F
N20  M30 L_F
```



*Polar coordinates G110*



Functions G110/G111 are **non modal**. They only apply to **linear interpolation**. The feedrate is the last F value (G11) active or rapid traverse rate (G10) to have been programmed.



### 3.2.7 Thread cutting G33/G34/G35

Threads can be cut both on drilling/boring and milling machines with a boring tool or a facing tool. There are various types of thread which can be cut as follows:

- Threads with a constant lead
- Threads with a variable lead
- Single or multiple threads
- External or internal threads

The following preparatory functions are available for **machining threads**:

**G33** Thread cutting with constant lead

**G34** Thread cutting with linear lead increase

**G35** Thread cutting with linear lead decrease

The G functions G33, G34 and G35 are assigned to the leading spindle.

The **thread length** is entered under the corresponding path address; start-stop and overrun sections at which the feedrate is increased or reduced must be taken into consideration. The values can be entered using absolute or incremental position data.

The **thread lead** is entered under addresses I, J and K.

I, J and K must always be entered using incremental position data and without a sign.

The thread lead can be programmed between 0.001 mm and 2000.000 mm.

**Right** and **left-hand threads** are programmed by specifying the direction of spindle rotation **M03** and **M04**.

The direction of spindle rotation and the speed must be programmed in the block prior to the actual thread cutting operation to permit the spindle to run up to its nominal speed.

#### Example:

N10 S500 M03 L\_F

Spindle speed S = 500 rev/min, clockwise

N15 G33 Z... K... L\_F

Thread cutting with constant lead

The feed start does not begin until the zero mark is reached on the pulse encoder in order to permit threads to be cut in several steps. This ensures that the tool always enters the workpiece at the same point on the circumference of the workpiece. The cuts should be implemented at the same (spindle) speed to prevent discrepancies in the following error.

The feedrate override switch, the "FEED OFF" key, the spindle speed override switch and the "SINGLE BLOCK" mode have no effect during thread cutting.

The feedrate programmed under F remains stored however, and is effective again when, for example, G01 is next programmed.

The lead of the tapers at which the thread is cut can be altered in steps. This ensures that longitudinal threads run out gently.

### 3.2.7.1 Thread with constant lead

The feedrate **F** is not programmed here since the feedrate is linked directly to the spindle speed via a pulse encoder.

### 3.2.7.2 Thread with variable lead

The **thread lead per turn** is altered by the value programmed under **address F** until the maximum or minimum possible value is obtained.

**G34 Lead increase:**

**N25 G34 G90 Z217 K2 F0.1 L<sub>F</sub>**

K2 Initial lead 2 mm

F0.1 Lead change + 0.1 mm per turn, i.e. after 5 turns the lead is 2.5 mm

**G35 Lead decrease:**

**N45 G35 G90 Z417 K10 F0.5 L<sub>F</sub>**

K10 Initial lead 10 mm

F0.5 Lead change 0.5 mm per turn, i.e. after 10 turns the lead is 5 mm.

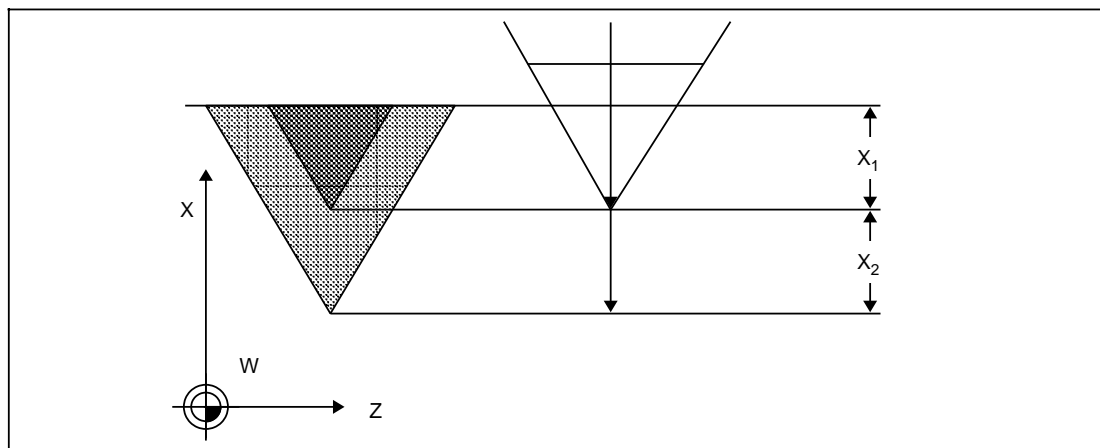
Value **F** is **calculated** from the initial and final leads:

$$F = \frac{\text{Initial lead}^2 - \text{Final lead}^2}{2 \cdot \text{Thread length}}$$

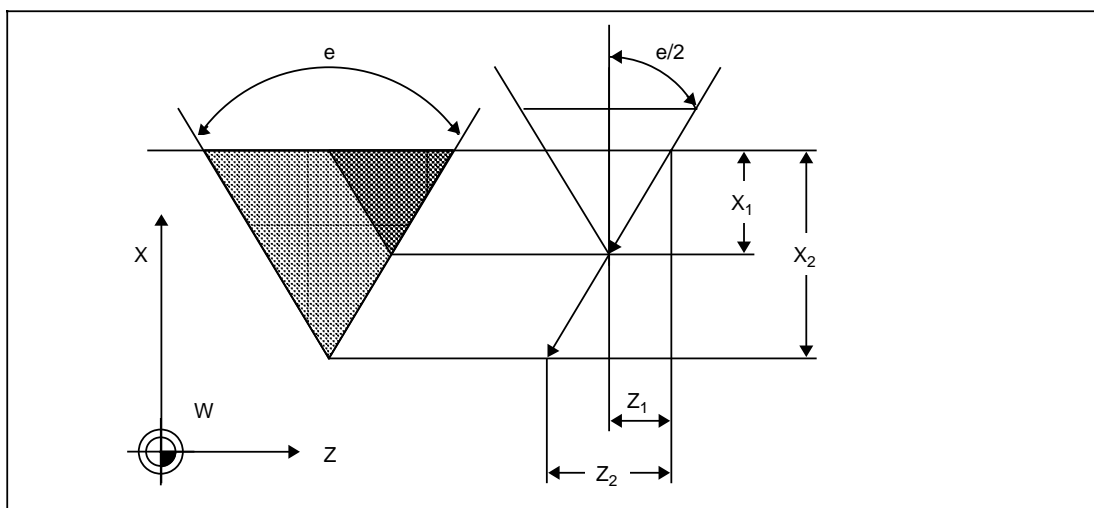
The value should be used without a sign.

### 3.2.7.3 Infeed options

The tool may be advanced perpendicular to the cutting direction or along the flank.



*Infeed perpendicular to cutting direction*



"Flank infeed"  $Z = X * \tan e/2$  - cut with one tool side

### 3.2.7.4 Multiple threads

Thread cutting always begins at the synchronization point of the zero mark of the pulse encoder. The feed is not enabled unless this signal is received from the digital rotary transducer. The starting position for cutting the thread can be offset with the aid of the program. This makes it possible to cut multiple threads. A turn of a multiple thread is programmed in the same manner as a **single thread**. When the first turn has been fully machined, the **starting position** is offset **by h'** and the next turn is machined.

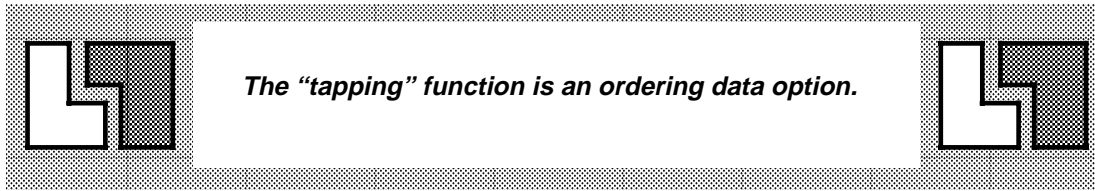
$h' = \frac{\text{Thread lead}}{\text{No. of turns}}$
---

The various turns must be executed at the **same spindle speed** in order to avoid discrepancies in the following error.





### 3.2.8 Tapping without encoder G63



Preparatory function G63 is used to tap threads using a **tap in the compensating chuck**. There is **no** functional relationship between the spindle speed and the feedrate.

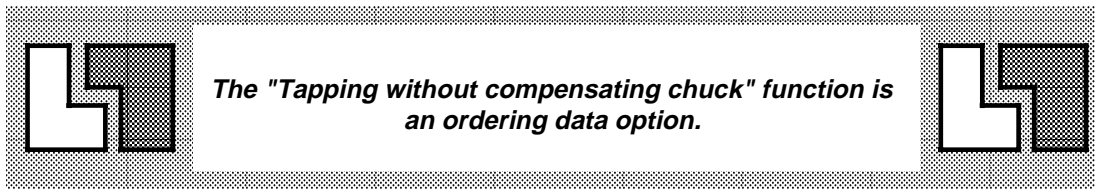
The spindle speed is programmed under address **S** and a suitable feedrate under address **F**.

The length compensating chuck must allow for the tolerances between the feedrate and the speed as well as the spindle overrun when the position is reached.

With G63 the **feedrate override switch** is **set to 100%**. The spindle may also be stopped in conjunction with "Feed hold" depending on the design of the interface controller. The spindle speed override switch is active.

**G63** can only be used in blocks **with** linear interpolation **G01**.

### 3.2.9 Tapping without G36 compensating chuck



Precondition for this function is rotary axis operation of the spindle.

Preparatory function G63 is used to tap threads (or to cut in the case of turning machines) using a tap without compensating chuck. The spindle in rotary axis operation and the infeed axis/axes interpolate. Thus, using a compensating chuck is not required for tapping or thread cutting on turning machines.

G36 can be used for the following types of thread:

- Cylindrical threads (spindle in rotary axis operation and **one** linear axis)
- Tapered threads (spindle in rotary axis operation with **two** linear axes)

Other programmings with G36 are not permitted.

Threads with lead increase and lead decrease (as in the case of G34, G35) cannot be implemented with G36.

On writing G36, the feed for the rotary axis in rev/min (G98) is automatically activated. No other feedrate for the rotary axis may be selected in a program part when G36 is active.

G36 is not deactivated until another G function of the G group 0 (e.g. G0) is selected. When deselecting G36, G98 is also deselected. The feed active beforehand becomes active again.

Programming conditions:

- Rotary axis operation of the spindle must be selected.
- In every block where G36 is active, only the rotary axis and the relevant linear axes may be written.  
A numerical value behind the rotary axis is ignored. The feed value F in rev/min for the rotary axis must be programmed in the first G36 active block.
- The thread lead is entered under addresses I, J, or K (please observe the machine manufacturer's specifications). The sign of the values for I, J, or K indicate the sense of rotation of the rotary axis, plus meaning right-hand thread and minus left-hand thread.
- One rotary axis and **one** linear axis must be programmed for cylindrical threads **and one** rotary axis and **two** linear axes for tapered threads.  
Other axis programmings generate alarm 3006. In the case of a tapered thread the lead increase parameter I, J or K is assigned to the linear axis with the longer traversing path. This enables programming of tapered threads with an inclination angle of max. 45°C.

#### Example 1: Cylindrical thread assigning K for the infeed axis Z

N10 M1=70 L_F	Selecting rotary axis operation for the spindle
N15 G0 C30 Z2 L_F	Position rotary axis C on 30 degrees and feed axis Z to thread insert position
N20 G36 C Z-30 K5 F10 L_F	Start tapping Lead increase 5 mm/rev right-handed thread and C axis feed 10 rev/min
N25 C Z2 K-5 L_F	Tapping Lead increase 5 mm/rev left-handed thread and C axis feed 10 rev/min (back to thread insert position)

#### Example 2: Tapered thread assigning K for the infeed axis

N10 M1=70 L_F	Selecting rotary axis operation for spindle
N15 G0 C30 X10 Z2 L_F	Position C axis to 30 degrees and infeed axis X, Z to thread insert position
N20 G36 C Z-30 X20 K5 F10 L_F	C axis feed 10 rev/min right-handed thread, K acts on Z, the path of axis Z being longer than that of axis X

Behavior on reset:

In the case of M02/M30 or operator panel reset, the delete position stored in the machine data becomes active (G36 or G98).

Display:

- G36 and G98 are displayed in the "G functions" display field.
- While G98 is active, the feedrate is displayed related to the rotary axis in rev/min (identifier U).
- The "rev/min" feedrate is displayed in the feed display with identifier U.

#### **Behavior after block search with calculation**

When starting a G36 block following block search with calculation, the spindle must be in rotary axis operation. Otherwise, the axis-specific reset alarm 196\* "Follow-up/Park for axis" is output. If no error occurs, the programmed G98 value (rev/min) is used as feed value.

From a technological point of view, a positioning block in front of the tapping block should be selected or all accumulated paths and offsets should be traversed in "REPOS" mode.

#### **Note:**

G36 enforces block end velocity 0. Thus a dwell time is no longer required from technological point of view.

### **3.2.10 Exact positioning G09/G60/G00, continuous path operation G62/G64**

#### **3.2.10.1 Fine and coarse exact stop tolerance ranges G09/G60/G00**

G09/G60	Fine exact stop tolerance range
G00	Coarse exact stop tolerance range

G09, G60 and G00 can be used to approach a target position within a specified **exact stop tolerance range**. When the **exact positioning window** is reached, the feedrate of the traversed axis (from P1 to P2, see next page) is reduced to 0. The following error is eliminated. At the same time a block change is initiated and the axis motion programmed in the next block (from P2 to P3) begins.

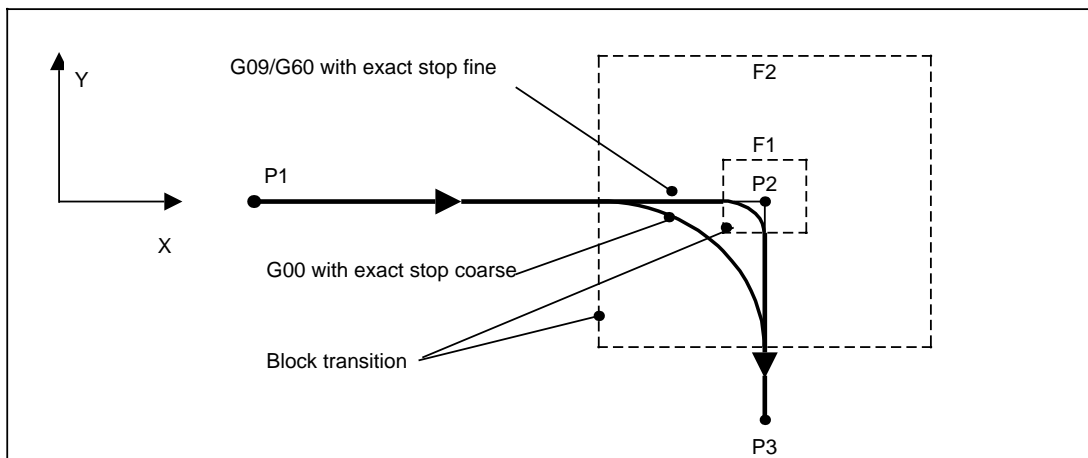
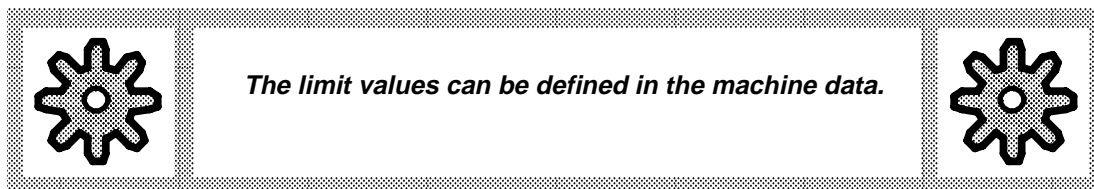
The action of G09 is block by block, whilst that of G60 is modal.

Functions G09 and G60 can be used, for example, for machining sharp corners, or when reversing the direction of movement.

## Fine and coarse exact stop tolerance ranges

Fine exact positioning:  $10\mu\text{m}$  (window F1)

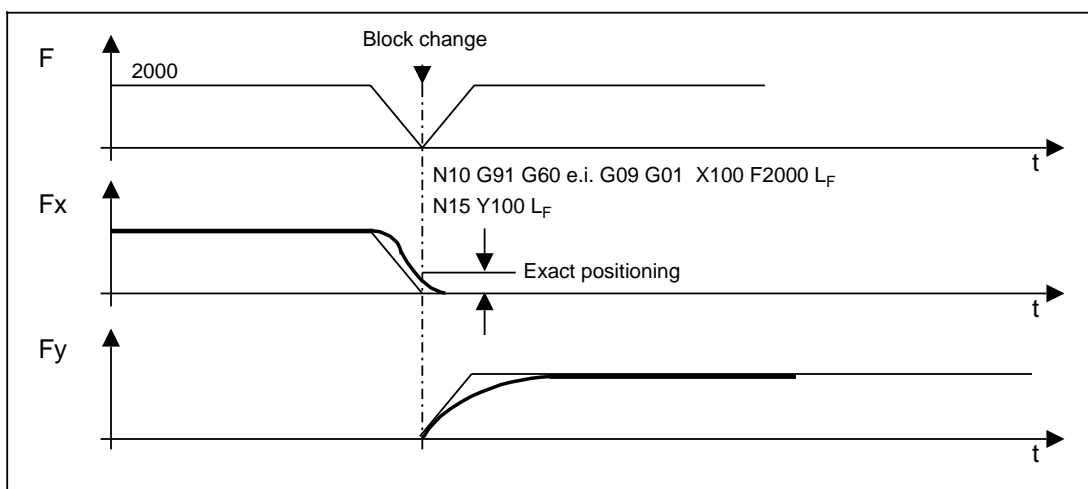
Coarse exact positioning:  $250\mu\text{m}$  (window F2).



Exact positioning window

If the two exact stop tolerance ranges are identical (window F1 = window F2), the effect of G00 will be the same as that of G09/G60. The **rapid traverse motion** generally has a **larger exact positioning window**. This saves time during rapid traversing (earlier block change).

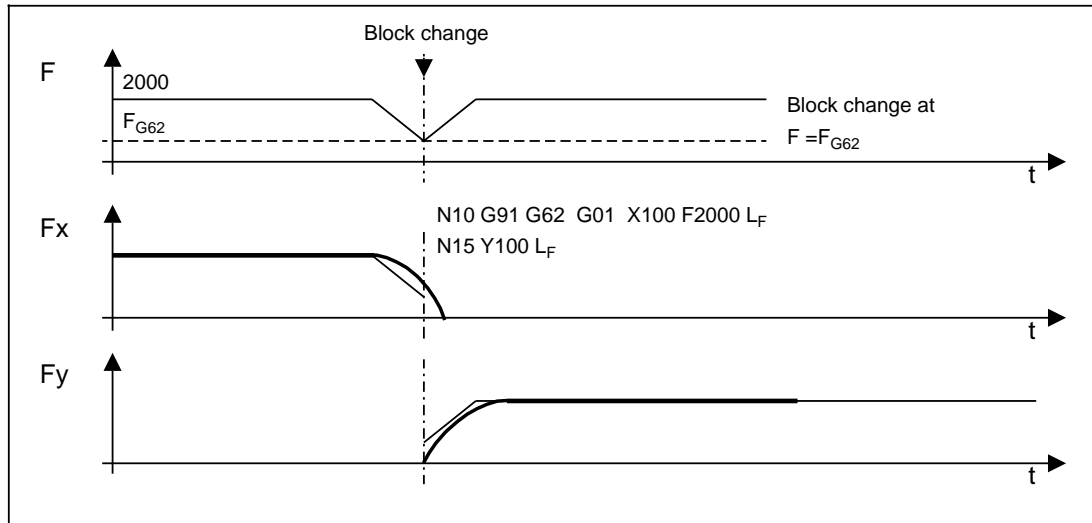
The thin line shows the velocity control of the controller. The position controller in the NC ensures a smooth characteristic (thick line).



Exact positioning G60/G09

### 3.2.10.2 Continuous path operation G62/G64

Function G62 **reduces the feedrate  $F$**  to a rate defined in **NC MD 3** towards the end of the **block**. With G64 the feedrate is not reduced (initial setting for continuous-path operation). The action of G62 and G64 is modal.



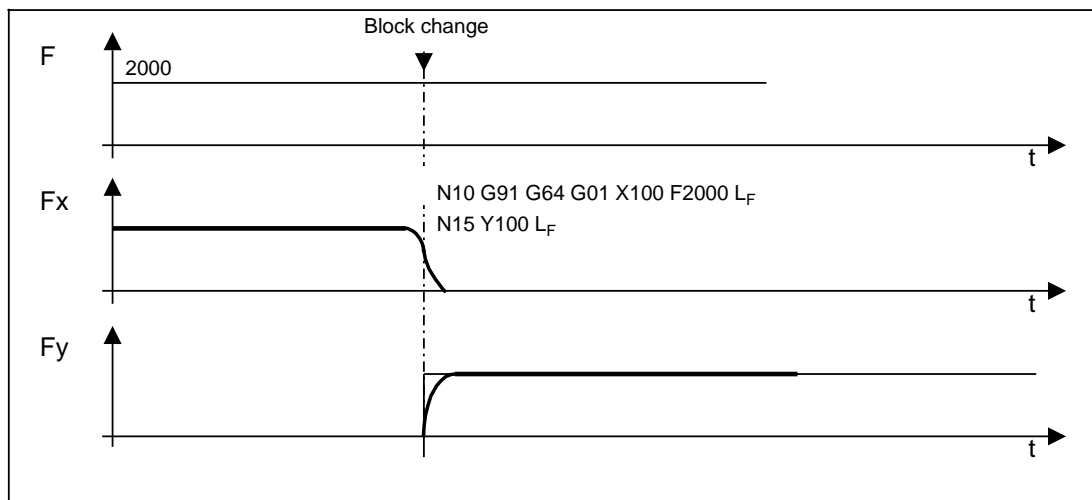
Continuous path operation with G62

#### Application:

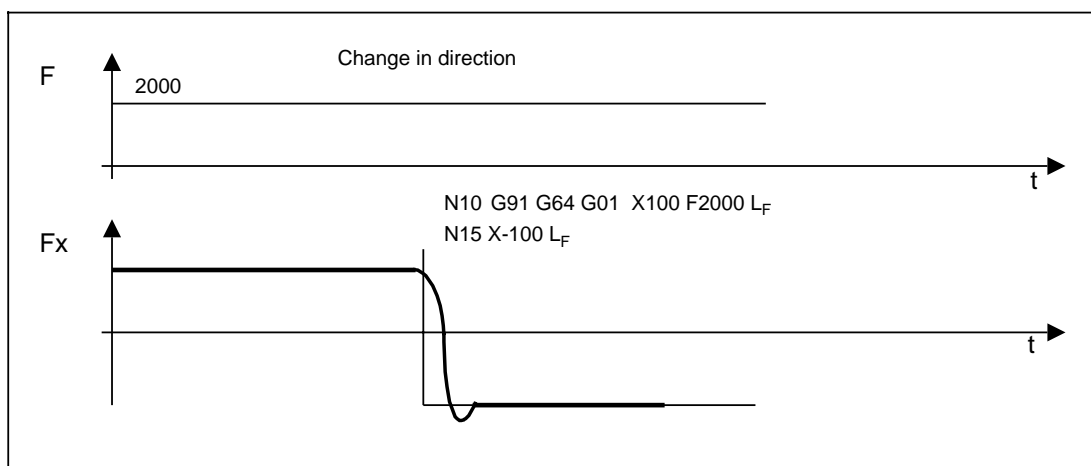
In woodworking operations the feedrate must not be allowed to become zero during a block transition, or scorch marks will result on the workpiece.

#### Block transition without feedrate reduction G64:

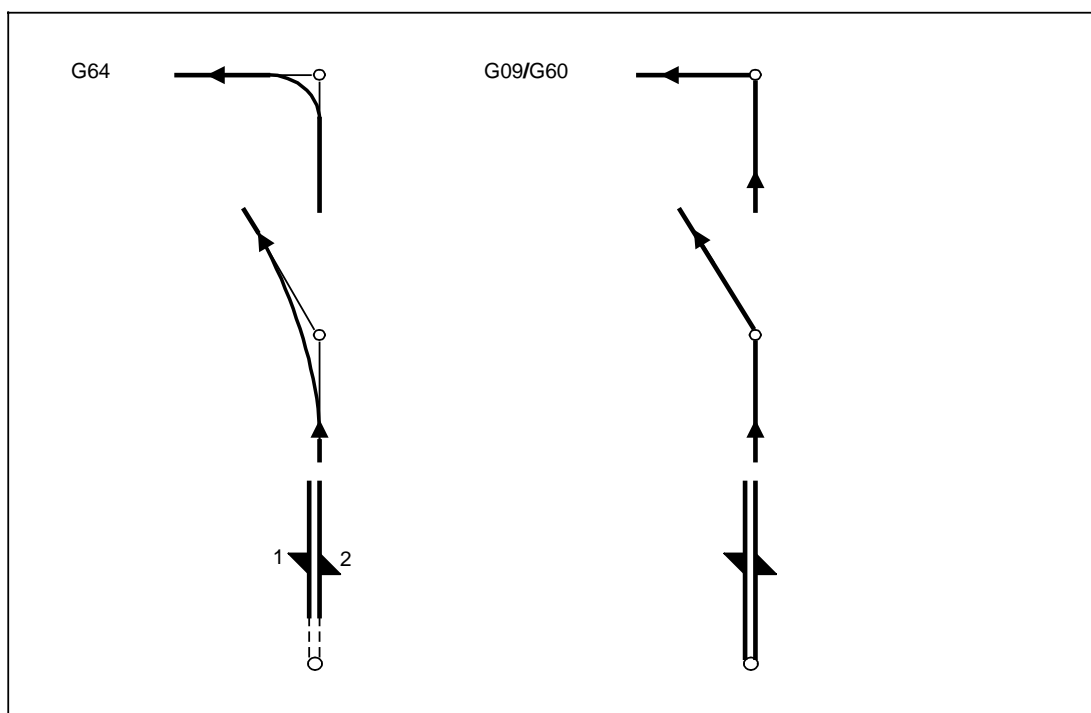
Preparatory function G64 is used **if relief cutting must be avoided during transitions** from one block to the next. It also permits smooth transitions when changing the direction of movement.



Continuous path operation with G64 without reduction in feedrate and with different axes



Continuous path operation with G64 and with a change in direction in one axis



Change in direction with (G09/G60) and without a (G64) reduction in feedrate

### 3.2.11 Dwell G04

Dwells are required on relief cutting, possibly on change in speed and machine switching operations (steadyrest, tailstock etc.).

The dwell is always entered without a sign.

The dwell is entered under address **X** or **F**. The **time range** is as follows:

0.001 to	99999.999 sec	for X and
0.001 to	99.999 sec	for F and
0.1 to	99.9 sec	spindle revolutions.

The action of G04 is **block by block**. **No other functions** may be written in a block containing a dwell time. G04 S... is assigned to the leading spindle.

**Example:**

```
N10 G04 X11.5 LF
```

└─ Dwell 11.5 s (always without a sign)

A further option is to program the dwell in spindle revolutions. The dwell is then programmed under address **S** in the range between **0.1 and 99.9 revolutions**.

**Example:**

```
N10 G04 S5 LF
```

Dwell 5 spindle revolutions

### 3.2.12 Soft approach to and retraction from contour

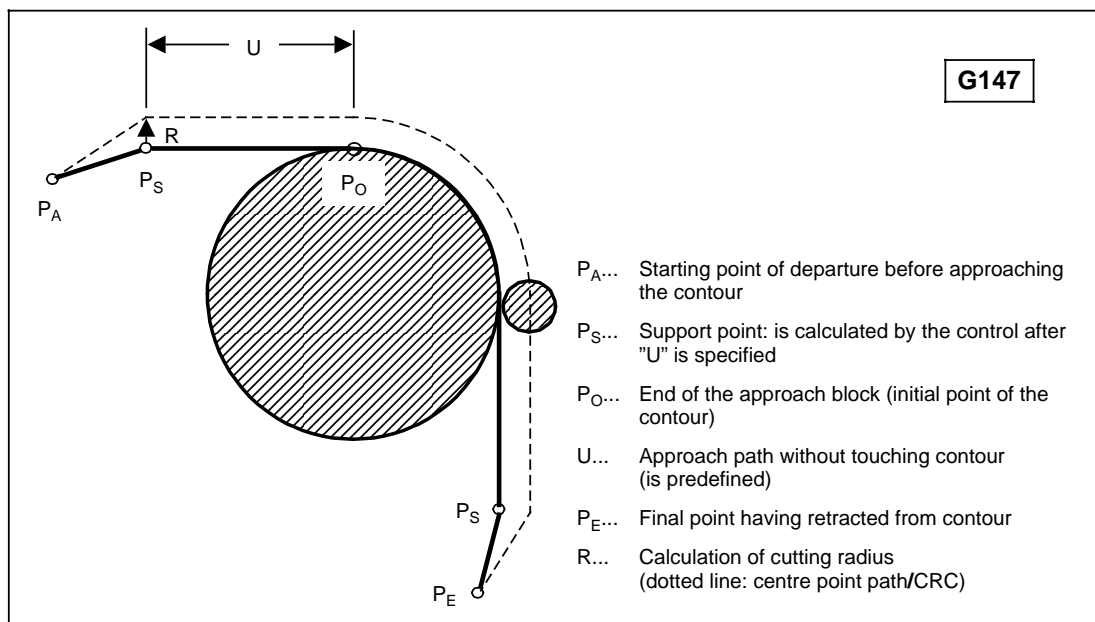
To avoid cutting marks a contour is approached to and retracted from **tangentially**.

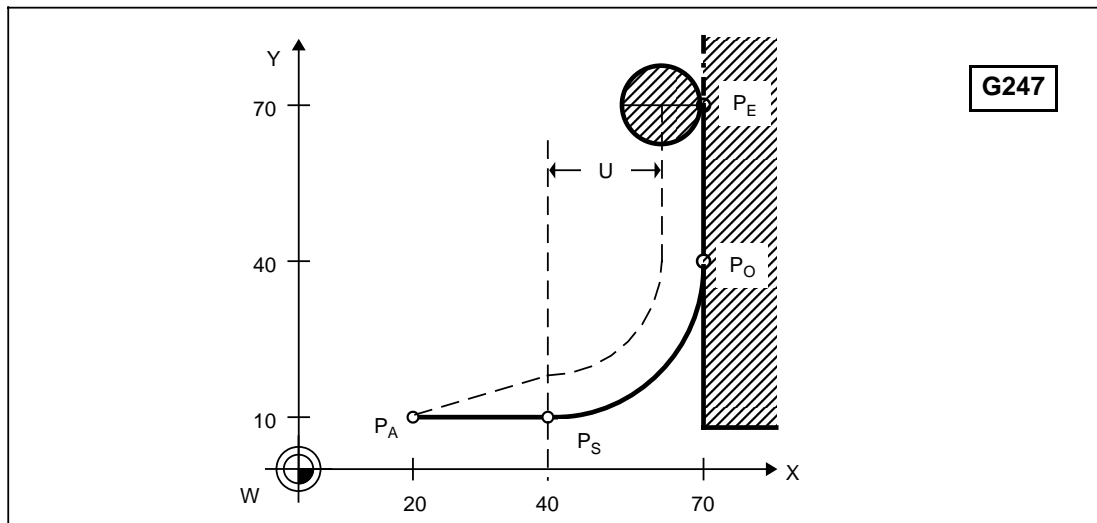
Soft approach to and retraction from a contour can be used **for all types of tools**. "Radius" and "Wear" are calculated.

**Approach to and retraction from** the contour can be programmed with the following functions:

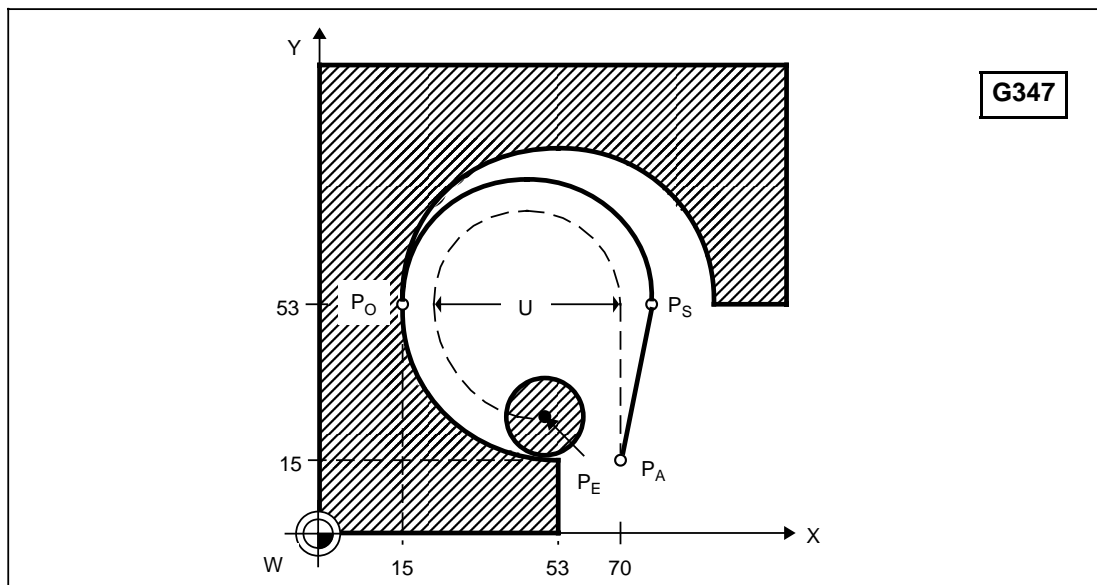
- G147** Approach linear
- G247** Approach in quarter circle
- G347** Approach in semi circle
- G148** Retraction linear
- G248** Retraction in quarter circle
- G348** Retraction in semi circle
- G48** Retraction from the contour as it was approached.

**Example:** Straight line to a circle



**Example:** Approaching a linear contour in a quarter circle

N05 G00 X20 Y10 L_F	Rapid traverse to P <sub>A</sub>
N10 G01 G41 X20 Y10 D1 F1000 L_F	Selection of CRC counter-clockwise (G41), selection of tool offset (D1)
N15 G247 X70 Y40 U25 L_F	Selection "Soft approach to contour with quarter circle" (G247), P <sub>S</sub> P <sub>O</sub>
N20 G01 Y70 L_F	Travelling along the linear contour up to P <sub>E</sub>
N25 M30 L_F	End of program

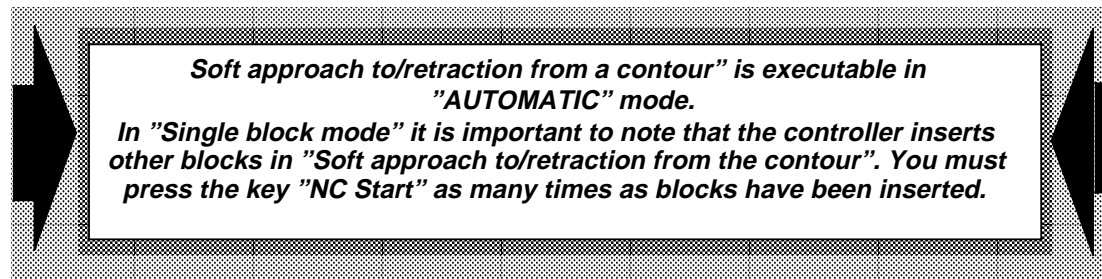
**Example:** Approaching an inside circle in a semicircle

N05 G00 X70 Y15 L_F	Rapid traverse to P <sub>A</sub>
N10 G01 G41 X70 Y15 D2 F1000 L_F	Selection of CRC counter-clockwise (G41), selection of tool offset (D2)
N15 G347 X15 Y53 U50 L_F	Selection "Soft approach to contour with semicircle" (G347), P <sub>S</sub> P <sub>O</sub>
N20 G03 X53 Y15 I38 J0 L_F	Travelling along the contour up to P <sub>E</sub>
N25 G0 X70 Y15 L_F	Rapid traverse to P <sub>A</sub>
N30 M30 L_F	End of program

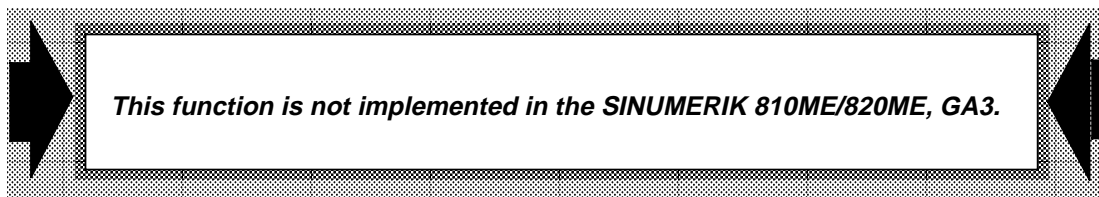


### Programming features

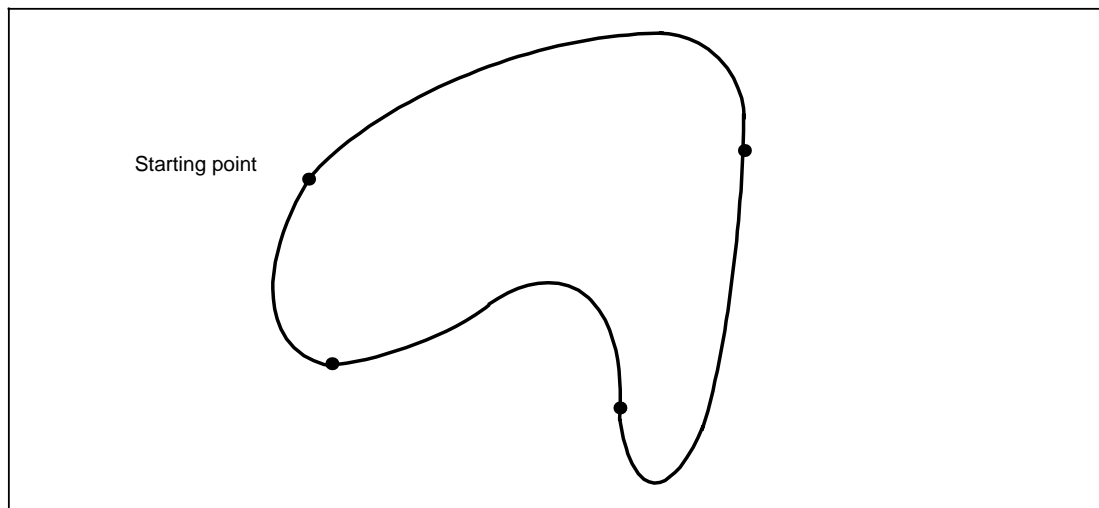
- The functions for approaching to and retracting from the contour are only effective in the block.
- The following must be specified in the approach block:
  - The coordinates of the initial point  $P_0$  of the contour
  - The value of U (approach path without touching contour)
- The following must be specified in the exit block:
  - The coordinates of the final point  $P_E$  after exiting the contour
  - The value of U (exit path without touching contour)
- No other traverse movements must be programmed in an approach and retraction block.
- Auxiliary functions can be programmed both in the approach block and in the exit block.
- A pure auxiliary function block may not be programmed. After an approach block or before an exit block.
- For the combination of approach and exit blocks it is important to note that "G40" (cancellation of the CRC) is generated in the exit block by the controller: program G41 or G42 before every approach block!
- For parts programs to be developed block by block (TEACH IN/PLAYBACK) the complete parts program can be completed subsequently.



### 3.2.13 SPLINE interpolation G06



A spline is the **linking of curves** which at the points where they join have the same function value, identical gradient and identical curvature.



*SPLINE interpolation G06*

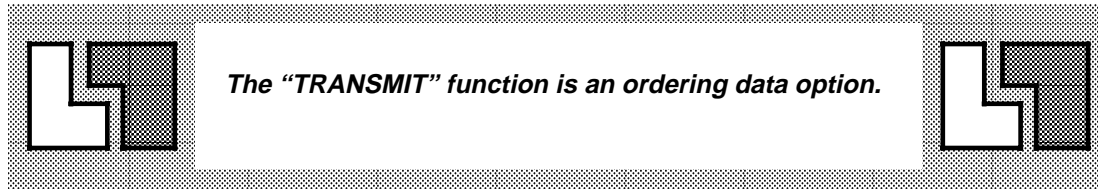
SPLINE interpolation reduces the programming overhead for machining **complex** contours and permits shapes to be machined which cannot be described with standard geometrics.

Programming of SPLINE interpolation is described in a separate **publication** "Programming Guide, SINUMERIK System 800, Spline Interpolation" (Order No. 6ZB5 410-7BA02-0BA1).

G06 is modal and is cancelled by another preparatory function of G group 0.



### 3.2.14 Coordinate transformation TRANSMIT

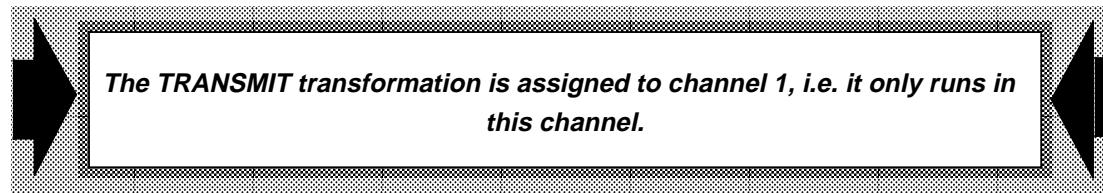


Definition: TRANSMIT = **TRANS**formation **M**illing Into **T**urning.

The TRANSMIT coordinate transformation function enables the face milling of turning parts on turning machines (810/820T). The program is written in a **fictitious** (Cartesian) coordinate system. Machine motions are performed in a **real** machine coordinate system. The axes for the fictitious coordinate system are defined in machine data by the machine manufacturer. A fictitious axis can only be moved when the transformation is selected. The real axes which form the transformation grouping are moved by the NC as the fictitious axes are controlled. In addition to the five real axes you can have two fictitious axes. The fictitious axes are handled by the NC in the same way as real axes. The only exception is the position control. Only the five real axes are processed connected to the measuring circuits. The TRANSMIT function is defined by a transformation record, which contains the following information:

- G function for TRANSMIT: G131
- the names of the fictitious axes involved in the transformation (programmed axes)
- The names of the axes which are generated and traversed by the transformation (real axes)
- the name of the infeed axis.

This transformation record is defined by machine tool manufacturer and must be known before programming.

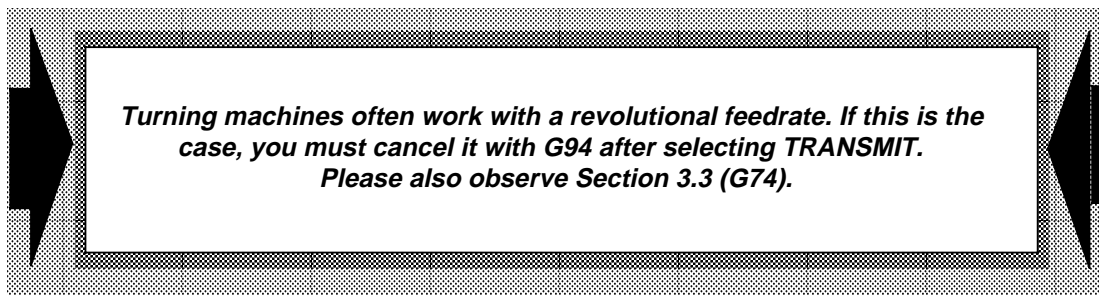


### 3.2.14.1 TRANSMIT function

The plane definition defined in the channel-specific machine data applies to the real system. For TRANSMIT, G17 is defined as the basic plane (fictitious), i.e. G17 is automatically set after transformation selection. When transformation is deselected the control **automatically** returns to the plane which was active before transformation was selected.

The **fictitious plane** is defined in the transformation data by assignment of the fictitious axes. You can program deviations from the basic planes with G16. The axis designations for fictitious and real axes must be unambiguous, i.e. the multiple use of an axis name is not permissible.

You select the TRANSMIT function with G function G141. This function must be in a block of its own, i.e. no other motions or other functions can be programmed in the same block. Otherwise the control outputs the alarm 2043 "programming error during transformation".



You can deselect TRANSMIT with a block containing the function G130. Fictitious axes must not be programmed in the **initial setting** (after G130). The real axes involved in the transformation, on the other hand, can only be programmed in the initial setting (after G130). If you do not observe this condition the alarm 2043 will be output.

#### When the TRANSMIT function is selected

The real axes of the transformation grouping are disabled. The machining plane is now defined by fictitious system. Linear and circular motions are combined to form a defined path.

The working area limitations (G25/G26) can only be programmed in the fictitious coordinate (Cartesian) system.

Tool offsets and zero offsets also relate to the fictitious coordinate system.

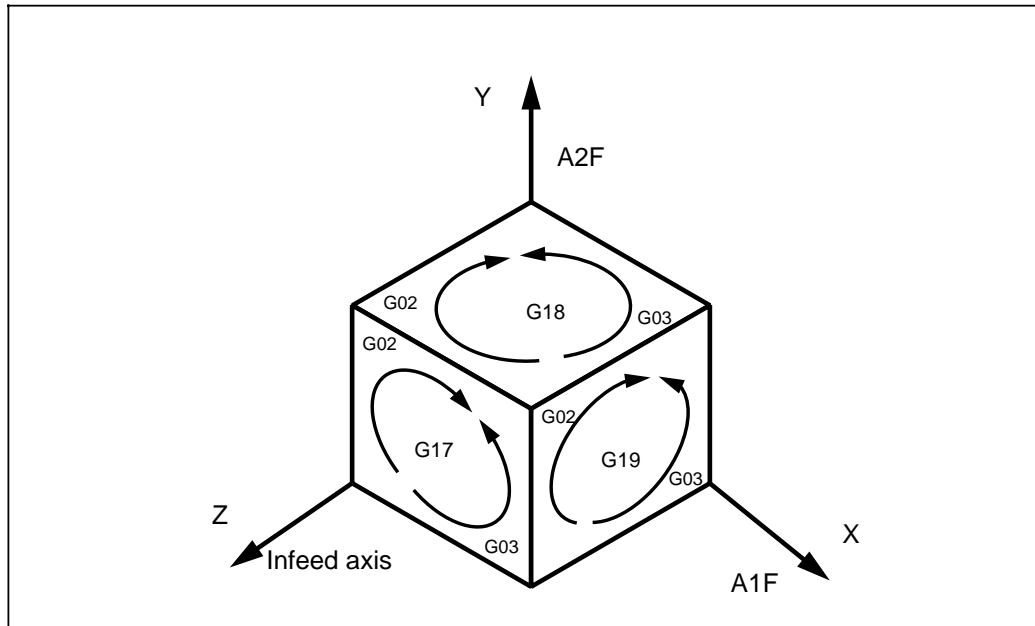
If you use the TRANSMIT tools with the tool length correction L1, you must specify the axis in which the correction is to be active.

You can either:

- set it up using free plane selection with the command:  
G16<1st fictitious axis><2nd fictitious axis> <infeed axis>  
or
- specify it directly in MD 5064 (name of the infeed axis).

In both cases the tool length correction L1 is on a real axis.

### Example: plane definition



M05	G17	Machining plane X-Y
M10	G00 X Y	Plane X-Y
M15	G16 Q1 B	Plane selection Q1-B
M20	G01 Q1=20 B=30	Plane Q1-B
M25	G131	Selection block for TRANSMIT transformation
.		Machining plane A1F,A2F
M40	G130	Deselection block for TRANSMIT transformation
.		Machining plane Q1-B
.		
M60	G17	Machining plane X-Y

### Explanation:

A1F ... A2F: 1st ... 2nd fictitious axis (A1F and A2F must be replaced by meaningful axis names).

Every selection/deselection of TRANSMIT performs the function “clear buffers” (@714). This function is activated internally by block preparation and need not be programmed. You must cancel the cutter/tool nose radius compensation (CRC/TNRC) before selecting TRANSMIT (because of @714).

Otherwise the alarm 2081 “CRC/TNRC not allowed” is output

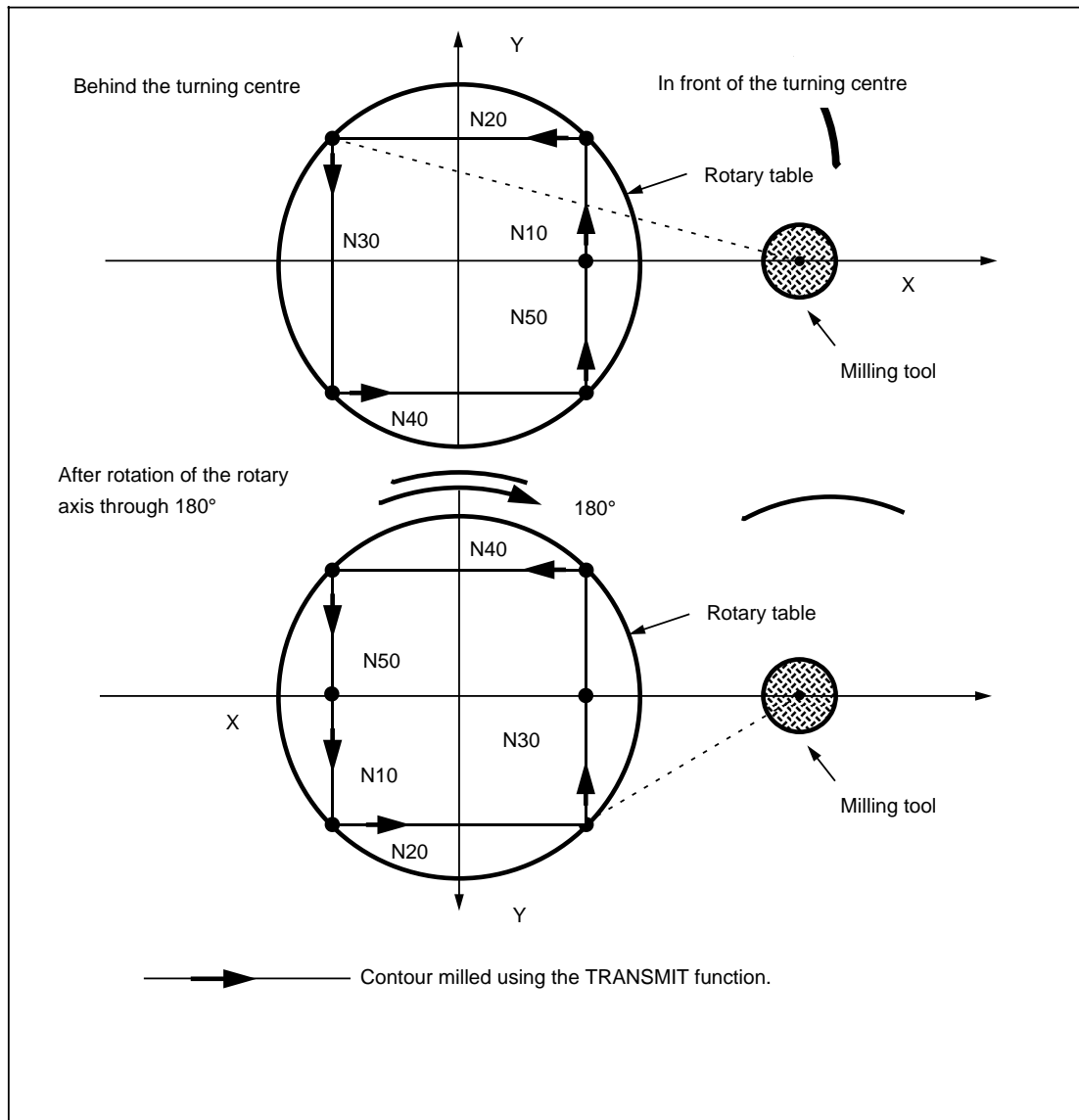
You must cancel CRC/TNRC again before deselecting TRANSMIT because when TRANSMIT is deselected the machining plane is changed. After deselection TRANSMIT the control returns to the plane which was active before TRANSMIT was selected. Only after selection or deselection of TRANSMIT can you reselect the CRC/TNRC.

TRANSMIT cannot be selected or deselected within a series of contour blocks. Block search with calculation to a part of program in which TRANSMIT is active is permitted (see Section 3.2.14.2).

### 3.2.14.2 Block search with calculation and the TRANSMIT function

The function "block search with calculation" can be used without restriction when TRANSMIT is active. The control generates a linear block from the current position to the target position after "block search with calculation" and "NC start" (**selected block**).

If the target position is behind the turning centre from the point of view of the tool, there is a risk of collision between the tool and the workpiece when the selected block is executed. In this case the rotary axis must be traversed in the REPOS mode until the target position is in front of the turning centre. If the tool is in the turning centre, the alarm 2191 "transformation in zero" is output when TRANSMIT is selected. Then you must move the tool out of the turning centre in JOG mode.



*Manual approach to the target position*

### 3.2.14.3 Principle of the TRANSMIT coordinate transformation

Two types of coordinate system are used in NC controls:

a) **Fictitious coordinate system:**

Cartesian coordinate system for producing e.g. part programs for milling on turning machines.

(Workpiece oriented coordinate system, control coordinate system, interpolation coordinate system).

b) **Real coordinate system:**

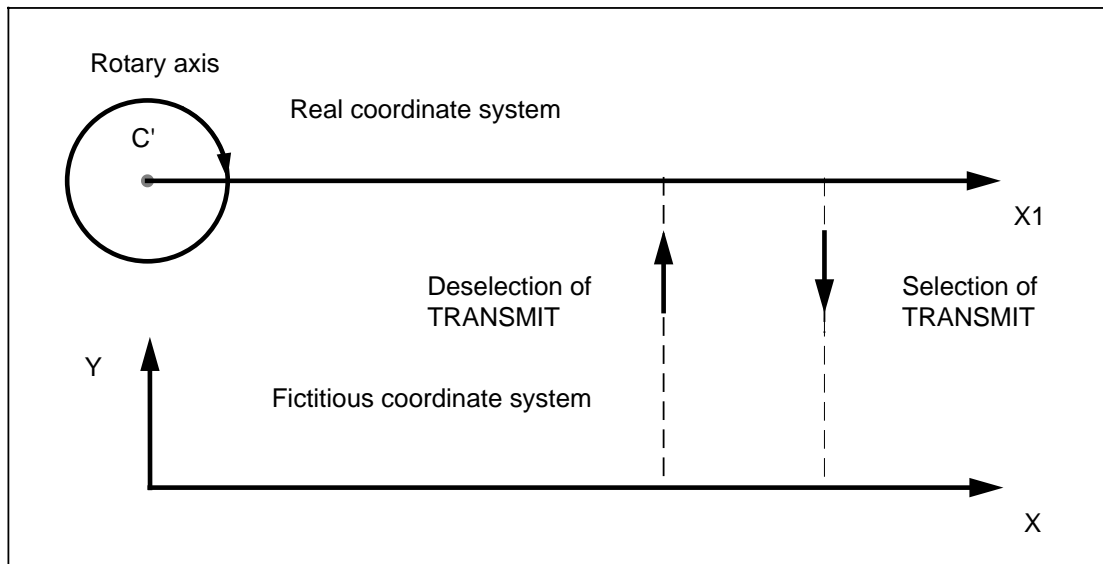
Coordinate system based on the real arrangement of machine axes (machine coordinate system).

A coordinate transformation describes the mathematical relation between two different coordinate systems by system of transformation equations. This system of equations must be valid for both directions of transformation. The coordinates are entered and the interpolation is performed in the Cartesian (fictitious) coordinate system. The setpoints are output to the position controller in the machine (real) coordinate system. The transformation converts between these two coordinate systems.

Using the TRANSMIT function you can perform milling operations on turning machines. Motions which would be difficult to program in the real coordinate system can be implemented with relatively simple motion blocks in the fictitious coordinate system.

#### Initialization of the coordinate systems

The fictitious and real coordinate systems have the same zero.



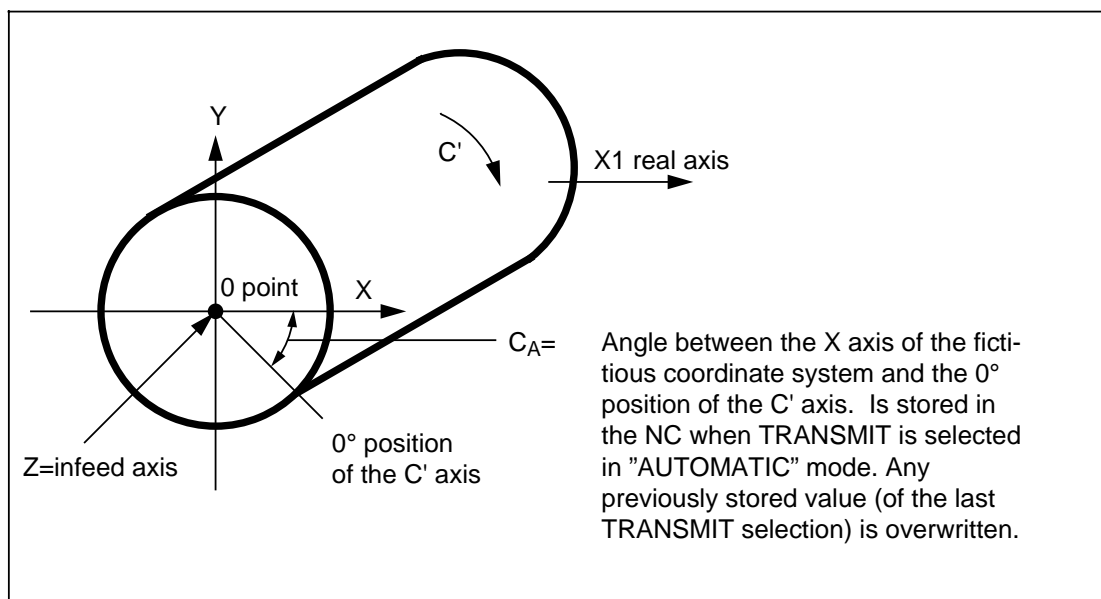
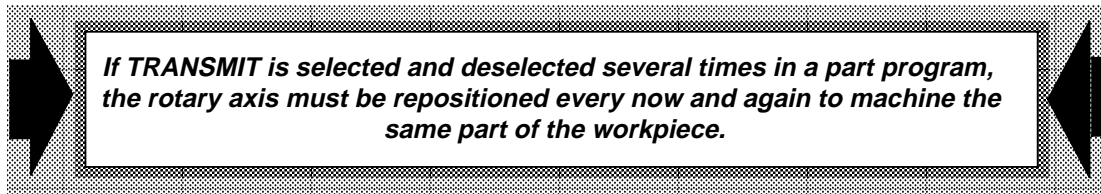
*Selection and deselection of TRANSMIT*



**Selection/deselection of TRANSMIT in "AUTOMATIC" or "MDI AUTOMATIC" mode:****AUTOMATIC mode:**

Selection of TRANSMIT (G131)

Deselection of TRANSMIT (G130)

 $X = X1$   
 $Y = 0$ 
 $X = 0$   
 $Y = 0$ 
**MDI AUTOMATIC mode:**

Selection of TRANSMIT (G131)

Deselection of TRANSMIT (G130)

 $X = X1 \cos (C_M - C_A)$   
 $Y = X1 \sin (C_M - C_A)$ 
 $X = 0$   
 $Y = 0$   
 $C_M$ : Current angle of the rotary axis  
 $C_A$ : Angle of the rotary axis on the last selection of the transformation in AUTOMATIC mode.

When TRANSMIT selected in "MDI AUTOMATIC" mode the angle  $C_A$  of the rotary from the last use of TRANSMIT is used in "AUTOMATIC" mode.

**Example:** Axis combinations for milling

Fictitious coordinate system: X,Y  
Real coordinate system: X1,C',Z

Progr. axis	Controlled axis	
	after selection	after deselection from TRANSMIT
1 dimensional		
X	X1 + C'	-
Y	X1 + C'	-
X1	-	X1
C'	-	C'
Z	Z	Z
2 dimensional		
X,Y	X1 + C'	-
X,Z	X1 + C',Z	-
Y,Z	X1 + C',Z	-
C',X1	-	C',X1
C',Z	-	C',Z
X1,Z	-	X1,Z
3 dimensional		
X,Y,Z	X1 + C',Z	-
X1,C',Z	-	X1,C',Z

The combinations marked "-" cause a machine stop and the alarm 2043 "programming error during transformation" is output.



### 3.2.14.4 Machining accuracy with TRANSMIT

The machining accuracy with TRANSMIT is determined by

- the feed programmed for the fictitious axes
- the velocity of the real axes, i.e. the machining rate
- the measuring system used (encoder).

#### Programmed feed, machining rate:

The size of the feedrate is the path velocity of the programmed motion in the fictitious plane X,Y. However, the velocity of the real axes, i.e. the machining rate, is constantly changing (see Section 3.2.14.5). This entails accelerations which make it impossible to guarantee the accuracy of a pure milling machine.

#### Measuring system used:

The machining accuracy in the linear axis (X1 axis) depends on the resolution of the encoder used. The influence of the rotary axis (C' axis) on the machining accuracy depends on the resolution of the angle position encoder by the following "rule of thumb":

The real linear deviation  $s$  therefore is up to:

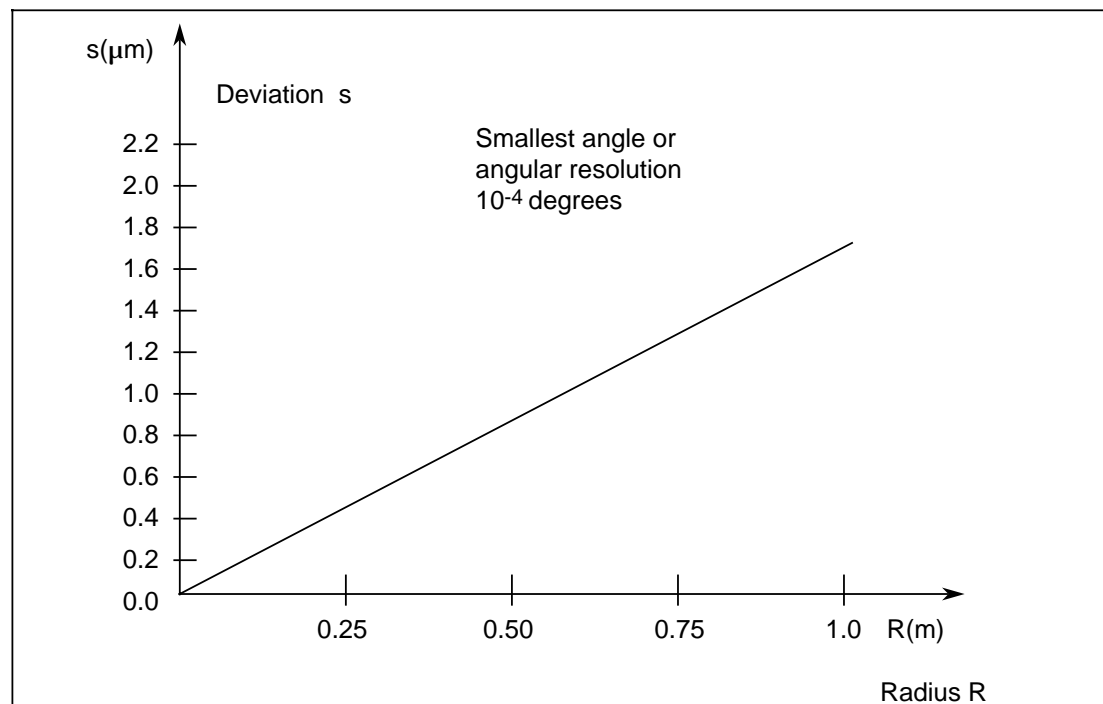
$$s = R / 180$$

where  $R = \sqrt{X^2 + Y^2}$ : distance of the programmed position (radius) for the fictitious coordinate system zero.

: smallest, settable angle value in degrees, i.e. the resolution of the angle position encoder.

The influence of the angular resolution on the deviations  $s$  to be expected with various radii  $R$  is shown by following diagram and table.

However, the real deviations are usually smaller than those calculated.



Deviation  $s$  on the surface of a rotary axis at an angular resolution of  $10^{-4}$  degrees

Deviation /10 <sup>-4</sup> mm	Radius R /10 <sup>-3</sup> m	Deviation /10 <sup>-4</sup> mm	Radius R /10 <sup>-3</sup> m
1	57.29747	10	572.9747
2	114.5949	11	630.2721
3	171.8924	12	687.5695
4	229.1899	13	744.8670
5	286.4873	14	802.1645
6	343.7848	15	859.4618
7	401.0827	16	916.7592
8	458.3798	17	974.0567
9	515.6772	18	1031.354

*Deviation s at an angular resolution of 10<sup>-4</sup> degrees*

### 3.2.14.5 Velocity monitoring with TRANSMIT

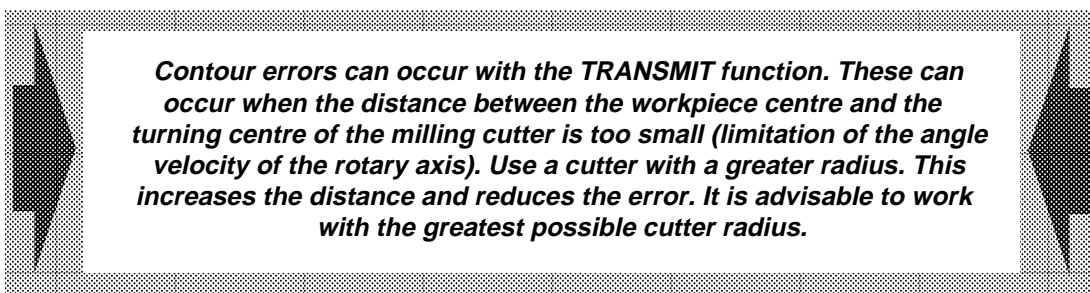
In "AUTOMATIC" mode the programmed velocity, i.e. the velocity in the fictitious coordinate system, is monitored.

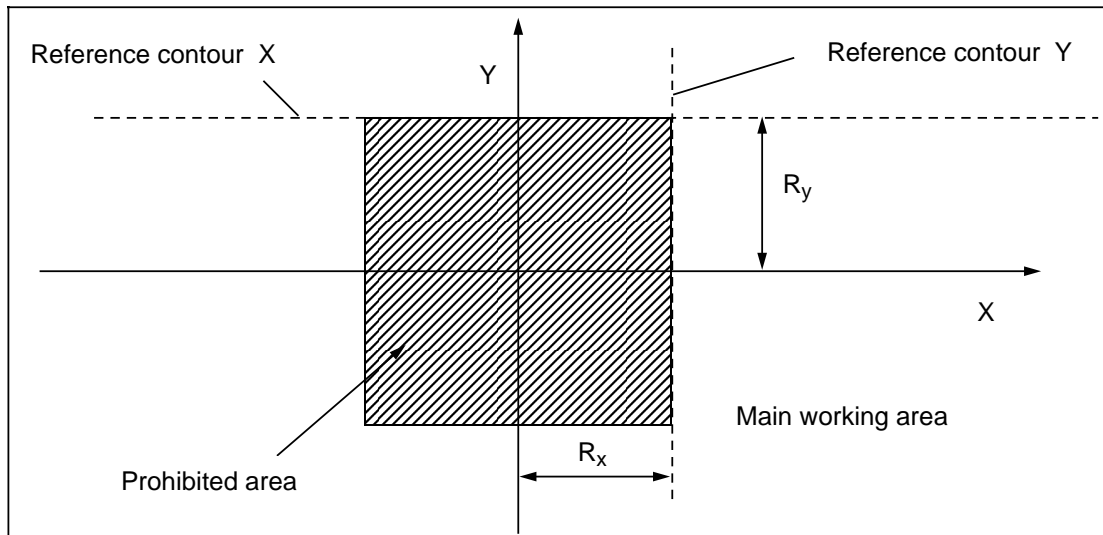
To prevent the rotary axis turning faster than is defined in MD 280\*, the feedrate might have to be reduced. If this is the case the block related alarm 3083 "feedrate limitation fictitious axis" is output. The velocity of the other real axis is also a function of the programmed contour and is therefore not constant.

#### Working area for the fictitious coordinate system:

The fictitious coordinate system is divided into a **main working area** and a **prohibited area**. If the fictitious axes enter the **prohibited area** the monitoring system responds. Controlled movements of the fictitious axes are possible in the prohibited area if the override switch is used. The response of the monitoring system is affected by the override switch.

The angular velocity of the rotary axis is monitored and if the maximum velocity (defined in a machine data) is violated the velocity is limited to this maximum value. At the same time the alarm 2035 "feedrate limitation" is output and except in JOG, INC and REPOS modes the alarm 3083 "feedrate limitation fictitious". These alarms must be acknowledged, alarm 2035 with a RESET and alarm 3083 with the acknowledgement key. An NC stop is not triggered. Because of the limitation, form errors occur on the contour when the alarm 2035 occurs.





*Devision of the coordinate system*

The boundary between the **main working area** and the **prohibited area** is the reference contour.

Please consult the machine manufacturer for the data of the areas.

Also consult the machine manufacturer for the maximum axis velocities.

A path which runs through the centre cannot be traversed. Angular velocities of up to 180° C sampling cycle would occur and the maximum rotary axis velocity would be exceeded. The rotary axis would have to be rotated through 180° at the turning centre. To avoid this, contours of this kind must be spread over at least three blocks (2nd to 4th block):

- |            |                                |
|------------|--------------------------------|
| 1st block: | Contour to the turning centre. |
| 2nd block: | Retraction of cutter.          |
| 3rd block: | Rotate of cutter 180°.         |
| 4th block: | Infeed the cutter.             |
| 5th block: | Contour to the turning centre. |

### 3.3 Reference point approach in part program (G74)

#### 3.3.1 Function description

The function "reference point synchronization in the part program" enables the user to approach the reference point of a real programmed NC axis from the part program using a G function. The machine control (servo and traverse behaviour) and the communication interface of the CNC (actual value, service display) are used unchanged. The function "REF in PP" is a standard function which is applicable to both linear and rotary axes.

#### 3.3.2 Starting the function

The function "REF in PP" can be activated in the "AUTOMATIC" and "MDI AUTOMATIC" modes. Syntax of the function in the part program:

N <No.> G74 < axis name> LF

##### Explanations:

<No.>                =    Block number  
< Axis name>       =    Address of a generated NC axis

##### Example:

N1000 G74 C L\_F

With an extended address name of an NC axis, e.g. Y1 the following applies:

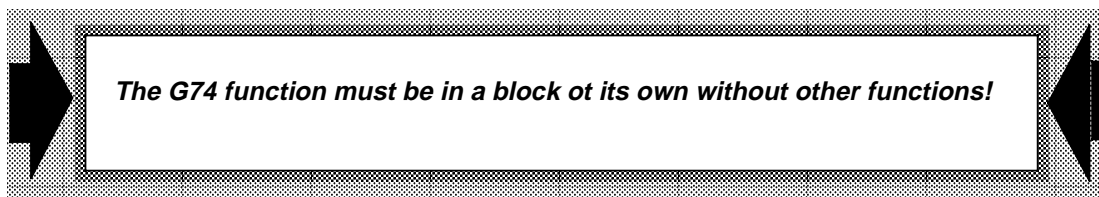
N <No.> G74 Y1=LF

##### Example:

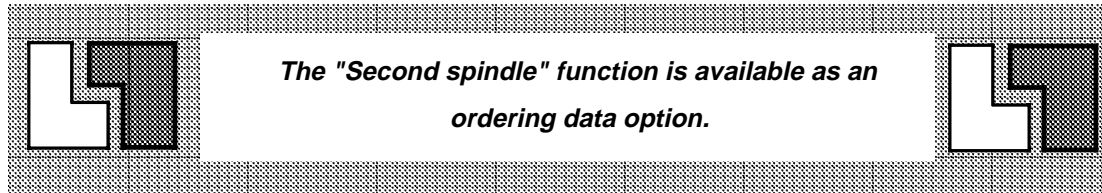
N1000 G74 Y1= L\_F

##### Remarks:

1. For the 810/820 T/M a restriction to one NC axis per G74 block applies.
2. Rotary and linear axes can be specified.



## 3.4 Second spindle



The second spindle can be used as main spindle or auxiliary spindle, e.g. as driven tool for the TRANSMIT function.

In the case of two spindles, one spindle must be defined as leading spindle for each channel. For the exact specification refer to the machine manufacturer. Setpoints and actual values for the valid leading spindle are displayed in the basic display operating modes.

The following G functions are assigned to the leading spindle:

- G33, G34, G35 Tapping, thread cutting
- G04 Sn= Dwell time related to spindle revolutions, n being the number of the leading spindle
- G95 Feedrate per revolution
- G96 Sn= Constant cutting velocity, n being the number of the leading spindle
- G97 Extended spindle speed from G96
- G63 Thread without encoder

M and S functions can be programmed with or without address extension (with the exception of G04, G96). The programmed M and S functions refer to the leading spindle when programming without address extension. The address extension is programmed directly behind the address character S or M, then "=" followed by the auxiliary function value. Spindle functions may be programmed only for one spindle in a part program block.

### Programming setting data for the spindle

SD 4010	(spindle 1)	Spindle speed limitation for G92
SD 4011	(spindle 2)	Spindle speed limitation for G 92
SD 4020	(spindle 1)	Oriented spindle stop (M19)
SD 4021	(spindle 2)	Oriented spindle stop (M19)
SD 4030	(spindle 1)	Spindle speed limitation
SD 4031	(spindle 2)	Spindle speed limitation
SD 4040	(spindle 1)	Smoothing constant for thread
SD 4041	(spindle 2)	Smoothing constant for thread

With G26 S1=500, for example, the spindle speed limitation is entered in setting data 4030.  
With G92 S1=500, for example, the spindle speed limitation is entered in setting data 4010.  
With M1 = 19 S1 = 180, the setpoint position of the spindle in degrees is entered in setting data 4010 in addition to the oriented spindle stop.  
Setting data 4040 and 4041 replace setting data SD1.



### 3.5 On-the-fly synchronization of rotary axis

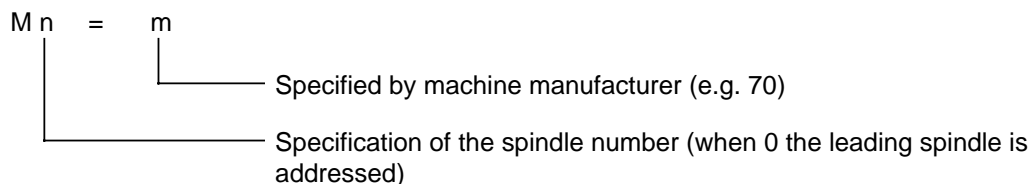
The functions

- Transmit for turning machines
- Tapping without compensating chuck (interpolating)
- Thread cutting without compensating chuck (interpolating)
- Cylindrical interpolation for turning machines

cause switchover from spindle operation to rotary axis operation and vice versa.

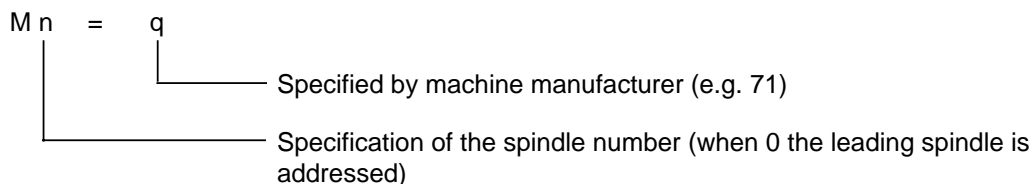
By means of the "On-the-fly synchronization of rotary axis" function, synchronization of the rotary axis is performed already when braking the spindle.

Switching from spindle operation to rotary axis operation is performed by means of the auxiliary function.



**Example:** M1=70

Switching from spindle operation to rotary axis operation is performed by means of the auxiliary function.



**Example:** M1=71

Internally, command @714 is executed.

The switching commands must be programmed in an NC block. In the NC reset state, switching can be effected by means of overstore. Before selecting block search, select or deselect rotary axis operation by means of overstore.

#### Notes:

- Spindle operation  
Position display for the rotary axis is "0". Rotary axis operation may be selected repeatedly.  
The rotary axis must neither be programmed nor traversed manually (otherwise alarm 1961 is generated).
- Rotary axis operation  
The spindle speed remains "0". Repeated selection of the rotary axis operation may be programmed.  
The spindle must not be programmed.
- Overstore in an interrupted program is not possible.
- Depending on the machine manufacturer's specifications, rotary axis operation remains active or is deselected with RESET.
- Selection and deselection of the rotary axis operation must be performed in the same channel.

## 3.6 Overstore

Speed values, miscellaneous functions M, tool numbers T, auxiliary functions H and the positions for a spindle can be overstored. The values can be entered via the address extension.

Please note:

- If values other than the existing spindles are specified for the address extension of S, these values are transmitted to the PLC without a spindle function being executed.

Possible values:

With one spindle S0=..., S2=... to S9=...

With two spindles S0=..., S3=... to S9=...

- Position values can only be overstored in full degrees.
- When overstoring M19 without specifying a position setpoint, the axis is positioned to 0°. The setting data "Oriented spindle stop" is not considered with M19 overstoring. This also means that this data is not overwritten during overstoring.
- When overstoring M and S values for the spindle without specifying the address extension, the number of the selected leading spindle is extended (see also display OVERSTORE).



### M01 or M\*=1 Programmed stop (conditional)

Function M01 is the same as M00 but is active only if the "Conditional stop (M01) active" function has been activated by means of a softkey or at the interface. Text in reverse video cannot be displayed with M01.

### M02 or M\*=2 Program end

M02 signals the end of the program and resets the program to the beginning. It is written in the last block of the program. The controller is reset. M02 may be written in a separate block or in a block containing other functions. The read-in process can be stopped with M02 (setting data).

### M17 or M\*=17 Subroutine end

M17 is written in the last block of a subroutine. It may be written in a separate block or in a block containing other functions. M17 must **not** be written in the same block as a subroutine (nesting), otherwise the subroutine will not be executed.

### M30 or M\*=30 Program end

Function M30 is the same as M02.

### M36, M37 or M\*=36, M\*=37 feed reduction ratio 1:100

**M36** The programmed feedrate is valid again  
Deselect M37

**M37** The programmed feedrate is reduced by 1:100. Is active for G94, G95 and G96.  
Applies for path feed and all simulation feedrates.

#### Example:

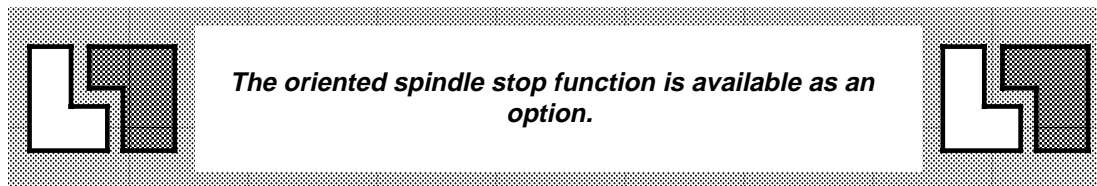
N10	G94	X30	F1000 L <sub>F</sub>	Feedrate 1000 mm/min active
:				
N30	M37	L <sub>F</sub>		Feedrate 10 mm/min active
:				
N50	M36	L <sub>F</sub>		Feedrate 1000 mm/min active

### M1=3, M2=3, M1=4, M2=4, M1=5, M2=5, M1=19, M2=19 Main spindle control

(M19 only with pulse encoder at main spindle)

The extended address notation establishes the relation to spindle 1 or spindle 2.

When 0 is specified for the address extension, the function is valid for the leading spindle.



In the version with analog spindle speed output the following M words are fixed for spindle control:

M1=3, M2=3	Clockwise spindle direction of rotation	spindle 1, spindle 2
M1=4, M2=4	Counter-clockwise spindle direction of rotation	spindle 1, spindle 2
M1=5, M2=5	Non-oriented spindle stop	spindle 1, spindle 2
M1=19, M2=19	Oriented spindle stop	spindle 1, spindle 2
M0=...	Applies to the leading spindle	

M19 S.. can be used to perform an oriented main spindle stop. The relevant angle is programmed under S in degrees. The angle is measured from the zero mark in the clockwise direction of rotation. M19 must be in a block of its own.

The action of the angle programmed under address S is modal. If M19 is programmed without S, the value stored under S is valid for the angle, i.e. a repeated stop can be effected simply by programming M19.

The angle may also be input via the operator panel under "Spindle setting data". A "MACHINE DATUM" can specify whether the spindle has to be at rest before the axis motion programmed in the next block is started or whether the next block is enabled during spindle positioning.

M19 does not cancel M03 or M04.

### Freely assignable miscellaneous functions

All miscellaneous functions **with the exception of** M00, M01, M02, M03, M04, M05, M17, M19, M30, M36 and M37 are freely assignable.

The extended address notation must be used and the channel number specified for all freely assignable miscellaneous functions:

#### Example:

**M3=124**

**3...** NC channel number

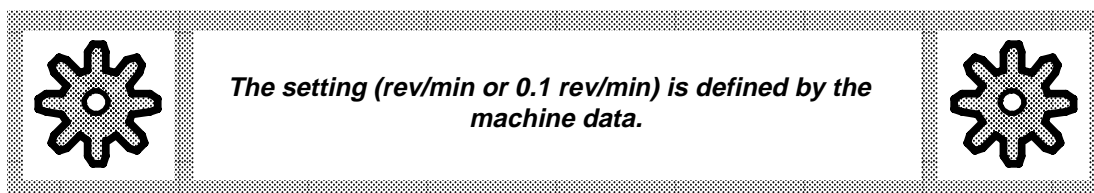
**124...** M function 124.

Further details on the use of the various functions can be found in the specific machine **program key** (see Section 12). The meaning of some of these functions is defined in DIN 66025.

## 4.3 Spindle function S

The data below can optionally be entered under address S:

- Spindle speed in rev/min or 0.1 rev/min \*)
- Spindle speed limitation in rev/min or 0.1 rev/min \*)
- Spindle stop in degrees (M19)
- Spindle dwell in revolutions (see G04).



\*) The speed and cutting speed must be programmed in the same input format.

For the **S** word the **extended address representation** with the spindle number applies.

**Example:**

**S2=1000**

**2 ...** Spindle number

**1000 ...** Spindle speed (rev/min).

S3=... S9=12345 (max. 5 digits for the value)



With one spindle: 2, 3, ..., 9

With two spindles: 3, 4, ..., 9

## 4.4 Auxiliary functions H

One auxiliary function per block can be entered under address H for machine switching functions or movements not covered by numerical control. H can be programmed with 4 decades. The meaning of the functions is described in the programming instructions issued by the machine-tool manufacturer.

## 4.5 Tool number T

The tool number determines the tool required for a machining operation:

**Example:**

**T 1234 ...**

**T** Address

**1234 ...** Tool number ( max. 4 decades).

## 5 Subroutines

### 5.1 Application

If the same machining operation must be performed repeatedly when a workpiece is machined, it can be entered as a subroutine and called as often as desired in the part program or by means of manual data input.

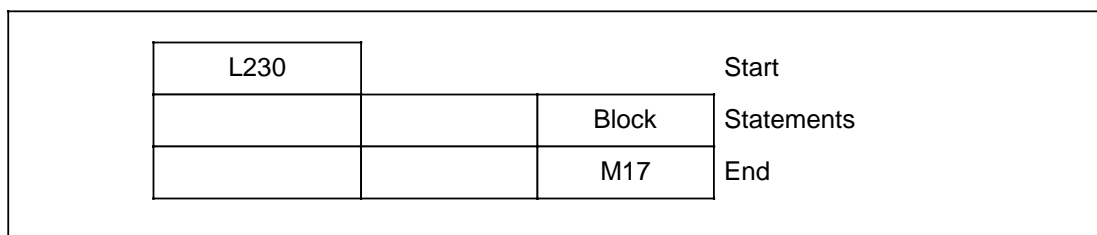
The **program memory** can be used to store **200** part programs and subroutines simultaneously.

Subroutines should preferably be programmed using **incremental position data**. The tool is set to the start position in the part program before the subroutine call. The machining sequence at the workpiece can then be repeated at various points on the workpiece without modifying the dimensions in the subroutine.

### 5.2 Subroutine structure

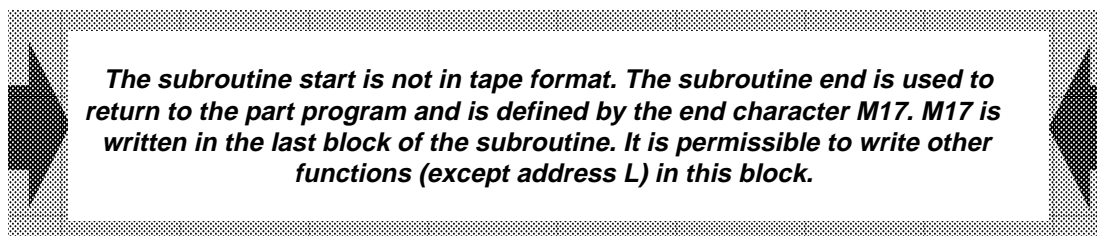
A subroutine comprises:

- the subroutine start (header),
- the subroutine blocks,
- the subroutine end.



*Subroutine structure*

The **subroutine start** comprises **address L** and the max. four-digit subroutine number (see program key, Section 12).



## 5.3 Subroutine call

The subroutine is called in a part program via address **L** with the subroutine number and the **number of passes** with address **P**. If a subroutine number is programmed without address **P**, a number of passes of **P1** (1 pass) is automatically assumed.

### Example:

<b>L123 P1 L_F</b>	
<b>L123</b>	Subroutine number (1...9999)
<b>P1</b>	Number of passes (1...99)

The following should be noted during programming:

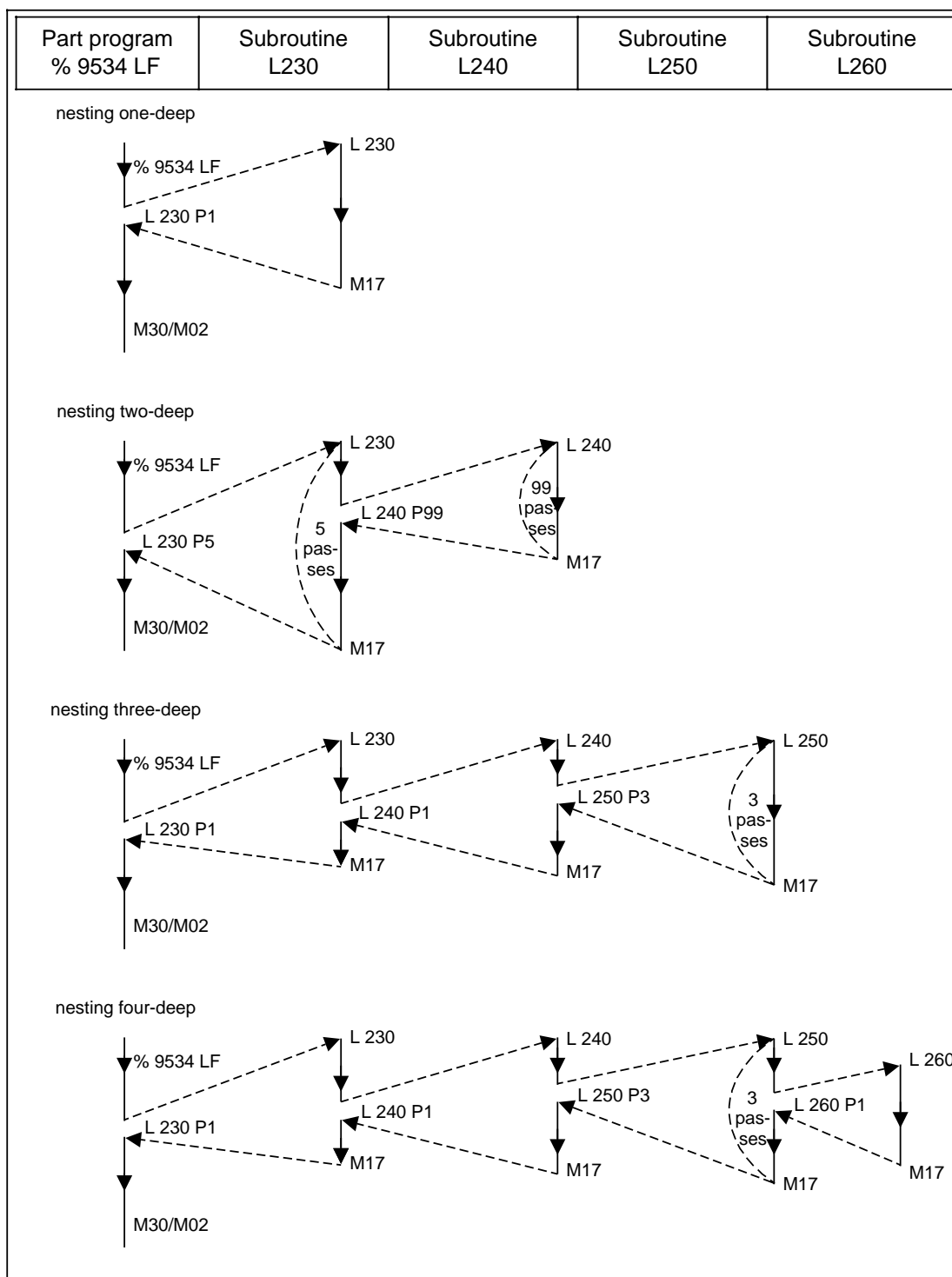
- The subroutine call must not be written in a block together with M02, M30 or M17, otherwise the subroutine is not executed.
- If the subroutine is called whilst the cutter radius compensation (CRC) function is selected, the section on special cases for CRC ("Blocks without path addresses") should be referred to.
- If the subroutine call is written in a block containing other functions, the subroutine is called at the end of the block.
- The special functions L81 to L89 can be called with G functions G81 to G89. G80 cancels G81 to G89 (initial setting).  
G81 to G89 and L81 to L89 in in a block trigger alarm 3006 "wrong block structure".
- G81 to G89 are modal.



## 5.4 Subroutine nesting

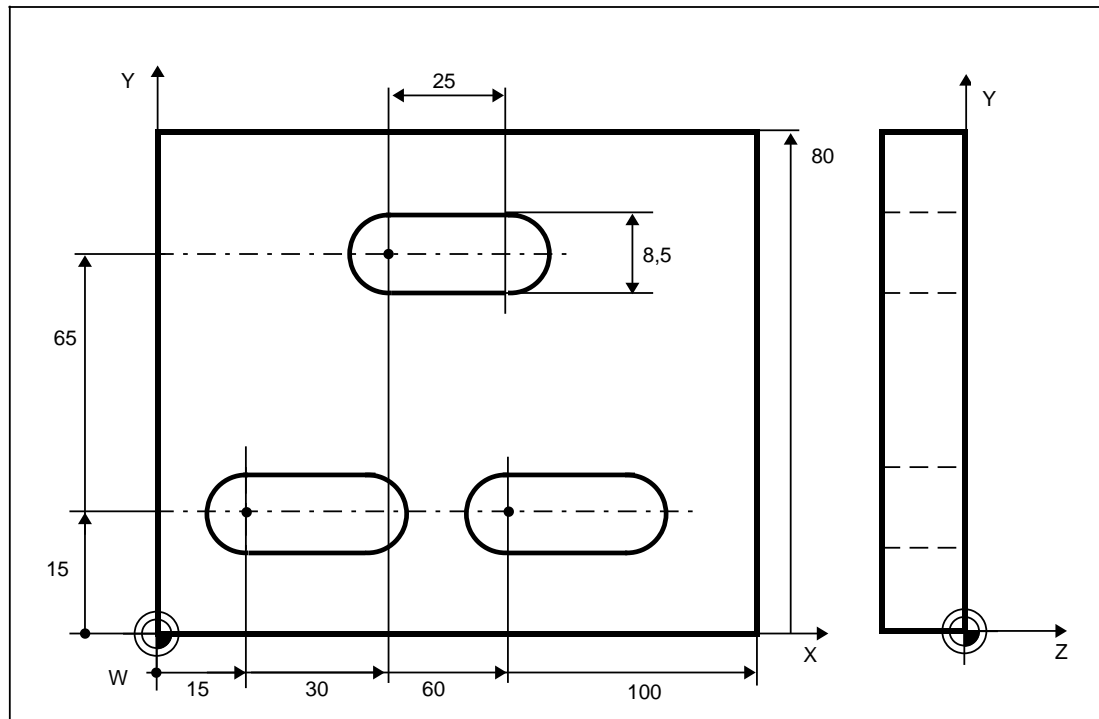
Subroutines can be called not only from a part program, but also from other subroutines. This process is referred to as subroutine nesting.

Nesting of the subroutine to a **depth of four** is possible.



*Subroutine call and subroutine nesting*

### Example of subroutine nesting: slot cutting



```
%4011 LF
N5 G00 Z100 LF
N10 T1 D2 LF
N15 L20 P1 X15 Y15 D1 Z2 S2000 M03 LF
```

```
N20 L20 P1 X60 LF
N25 L20 P1 X30 Y65 LF
N30 D00 Z100 M5 LF
N35 M30 LF
```

```
L10 LF
N5 G91 Z-2 F80 LF
N10 X25 F125 LF
N15 Z-2 F80 LF
N20 X-25 F125 LF
N25 G90 M17 LF
```

```
L20 LF
N5 G01 Z0 F80 LF
N10 L10 P4 LF
N15 G91 G64 G41 Y4.25 LF
N20 G03 Y-8.5 J-4.25 LF
N25 G01 X25 LF
N30 G03 Y8.5 J4.25 LF
N35 G01 X-25 LF
N40 G00 G40 Y-4.25 LF
N45 G90 Z2 M17 LF
```

Selection of tool offset (D2)

Tool no. T1; call subroutine L20, starting position,  
tool offset (D1)

Movement, call L20

Movement, call L20

Cancellation of tool offset, traverse clear

Cutting of slot, 2 mm deep

Cutting of slot of another 2 mm depth

Call subroutine L10

Continuous path operation, CRC counter-clockwise in the  
contour

Semicircle (counter-clockwise)

Semicircle (clockwise)

Cancellation of CRC

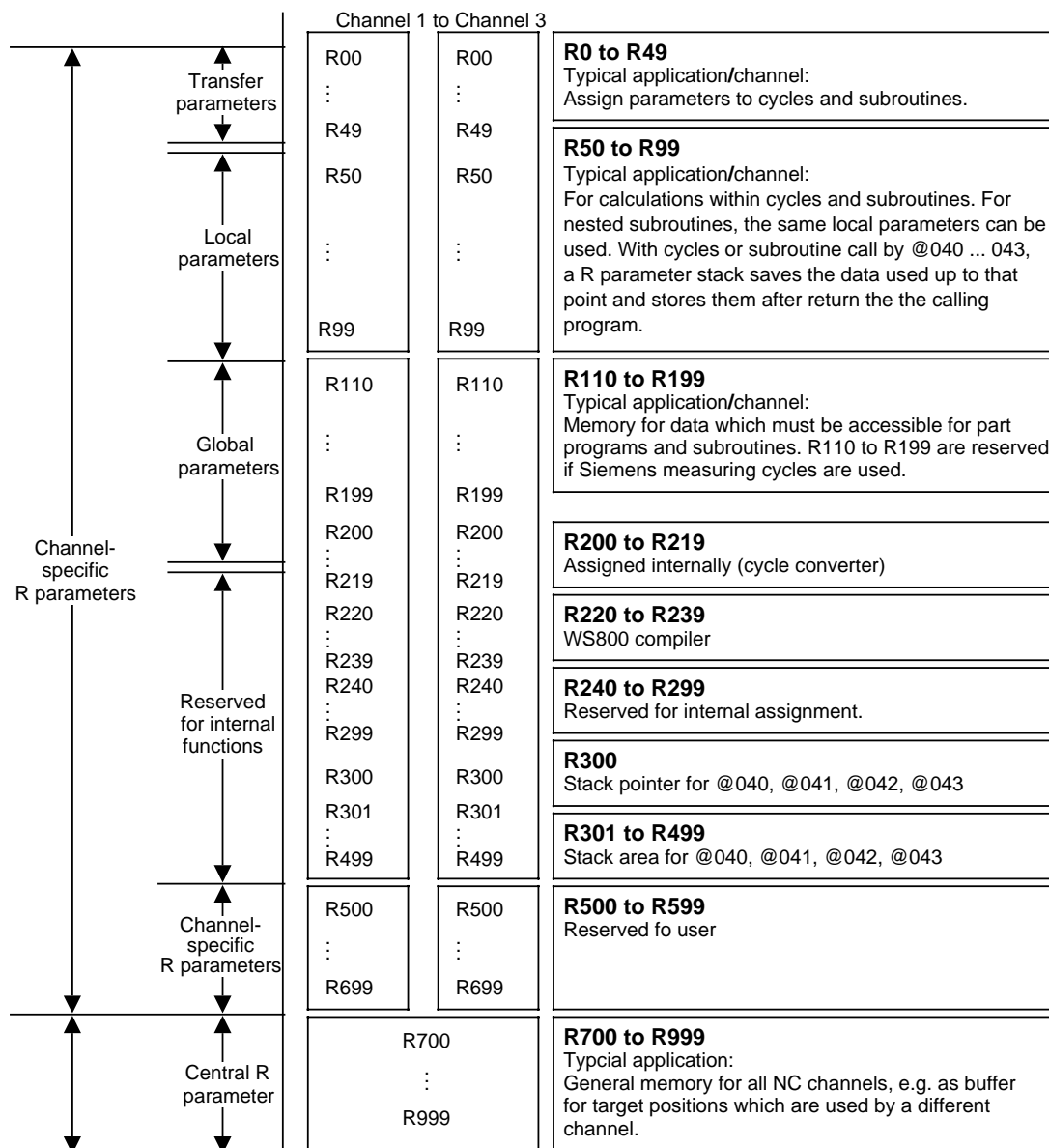
## 6 Parameters

### 6.1 Parameter programming

Parameters are used in a program to represent the numeric value of an address.

They are assigned values within the program, and can thus be used to adapt a program to several similar applications (e.g. different feedrates, different spindle speeds for various materials, different operating cycles).

A parameter comprises address **R** and a **number with up to 3 digits**. In the basic configuration **700 channel-specific and a total of 300 parameters** are available to the control; they are subdivided into transfer parameters, computing parameters, parameters defined as being channel-dependent or channel-independent and central parameters (see CL800 language description).



Structuring and application of R parameters

A parameter can be assigned instead of a value to all addresses with the exception of N.

N5 Z = R5 L<sub>F</sub>

The word structure of the various address must be observed (decimal notation or use of sign does not apply to all addresses).

Incorrect:

R1= 51120. 98  
H = -R1

This numerical value is not possible at address H (Alarm 3000 general programming error).

## 6.2 Parameter definition

Parameter definition is used to assign certain numeric values **with signs** to the various parameters.

The parameters can be defined either in **part programs** or in **subroutines**.

R1 = 10 L<sub>F</sub>

Parameter definition, subroutine call and switching functions may be written in a single block. The value defined for a parameter is assigned direct to the address.

### Example:

```
%5772 LF
N5...
:
N35 R1=10 R29=-20.05 R5=50 LF      Parameter definition
N40 L51 P2 LF                        Call of subroutine 51, 2 passes
N45 M02 LF

L51 LF
N5 X=-R5 B=-R1 LF
N10 Y=-R29 LF
:
N50 M17 LF
```

## 6.3 Parameter calculation

### Parameter linking

All **four basic arithmetic operations** are possible with parameters. The linking **sequence** is however crucial to the result of the calculation.

The rule whereby multiplication and division are performed before addition and subtraction does not apply here.

Arithmetic operation	Programmed arithmetic operation
Definition	$R1 = 100$
Assignment	$R1 = R2$
Negation	$R1 = - R2$
Addition	$R1 = R2 + R3$
Subtraction	$R1 = R2 - R3$
Multiplication	$R1 = R2 \cdot R3$
Division	$R1 = R2 / R3$

The result of an arithmetic operation is written in the first parameter of a link; its initial value is thus overwritten and lost when the logic operation is performed.

The values of the second and/or third parameters are retained.

### Value assignment amongst parameters

If the value of one parameter is to be assigned to another parameter, the following is valid:

$R1 = R3 \text{ L}_F$
-----------------------

### Calculation using numbers and parameters

- Addition and subtraction of numbers and parameters in conjunction with addresses

It is possible to add a parameter to the value of an address or to subtract it from it. Calculation signs must be written. No sign signifies a positive number.

The "+" sign must always be entered.

$X = 10 + R100 \text{ L}_F$
-----------------------------

or

$X = R100 + 10 \text{ L}_F$
-----------------------------

**Example:**

N35 R1=9.7 R2=-2.1 L\_F

N40 X=20.3+R1 L\_F

N45 Y=32.9-R2 L\_F

N50 Z=19.7-R1 L\_F

Result: X=30

Y=35

Z=10

- Calculations using numbers and parameters

It is possible to multiply, divide, add and subtract absolute numbers and R parameters.

R10=15+R11 L\_F

- Several separate equations can be programmed in one block.

R1 = R2+ 23 R50 = R37·3 R99 = R27 / R13 L\_F

## 6.4 Parameter string

R1=R2+R3-R4·R5/R6.. L\_F

All 4 basic arithmetic operations are permissible in any sequence. A parameter string is limited by the **block length** of **120 characters maximum**.

The calculations are performed as follows:

Step 1 R1 = R2

Step 2 R1 = R1 + R3

Step 3 R1 = R1 - R4

Step 4 R1 = R1 · R5

Step 5 R1 = R1 / R6

Step 1 R1 = R2

Step 2 R1 = R2 + R3

Step 3 R1 - R4

Step 4 R1·R5

Step 5 R1 / R6

R1

Instead of a link R parameter (not a result parameter), constants and pointers (pointers to R parameters) are allowed with address P in the parameter string.

**Example:**

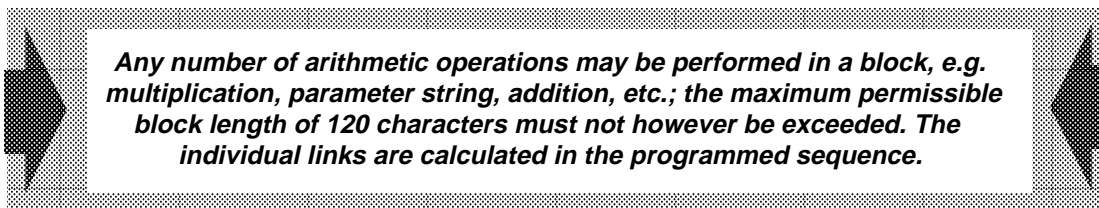
$R1=R2+10.5 - P3 \cdot R5/R6 \dots$

The result parameter must be an R parameter.

**P3:**

**P** Address of pointer

**3** Pointer to R parameter R3, i.e. the contents of R3 are the address of an R parameter whose value is included in the parameter string.



Value range: Minimum value:  $1 \cdot 10^{-8}$

Maximum value: 99999999.

Display: Floating point ( $\pm 12345678$ ) to ( $\pm 12345678.$ )

## 6.5 Programming examples with R parameters

### Parameters for subroutines without values

In the case of subroutines without values the current data are transferred by means of parameters **R00 to R99**.

These parameters are used in subroutines instead of numerical values.

**Up to 10 parameters** may be programmed for each block.

When a subroutine is called, it must be ensured that the parameters used exhibit the correct values. The values are assigned to the parameters in the main program.

**Example:** Paraxial cutting in the X-Y plane

The subroutine below permits a rectangle, whose sides are parallel to the machine axes, to be machined in the X-Y plane with a variable aspect ratio.

```

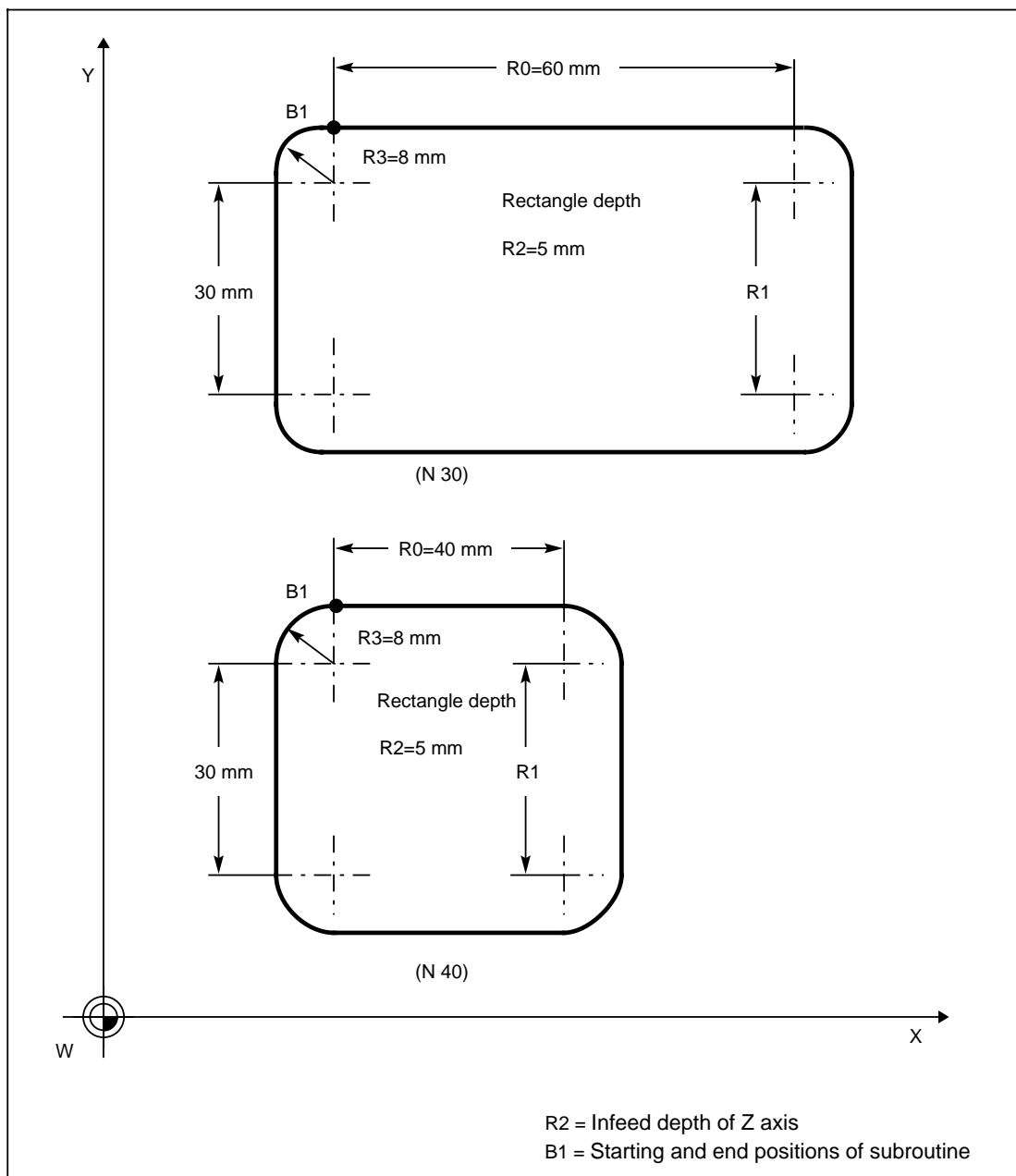
L46 L_F
N05 G01 G64 G91 Z=-R2 F500 L_F
N10 X=R0 L_F
N15 G02 X=R3 Y=-R3 I0 J=-R3 L_F
N20 G01 Y=-R1 L_F
N25 G02 X=-R3 Y=-R3 I=-R3 J0 L_F
N30 G01 X=-R0 L_F
N35 G02 X=-R3 Y=R3 I0 J=R3 L_F
N40 G01 Y=R1 L_F
N45 G02 G60 X=R3 Y=R3 I=R3 J0 L_F
N50 G00 Z=R2 L_F
N55 M17 L_F

```

Subroutine call:

.	
N25 G90 X... Y... Z... L_F	First starting position of current program
N30 L46 P1 R0=60 R1=30 R2=5 R3=8 L_F	
N35 G90 X... Y... Z... L_F	Second starting position
N40 L46 P1 R0=40 L_F	
.	





*Paraxial cutting in the X-Y plane*

### Example: Machining of internal semicircle

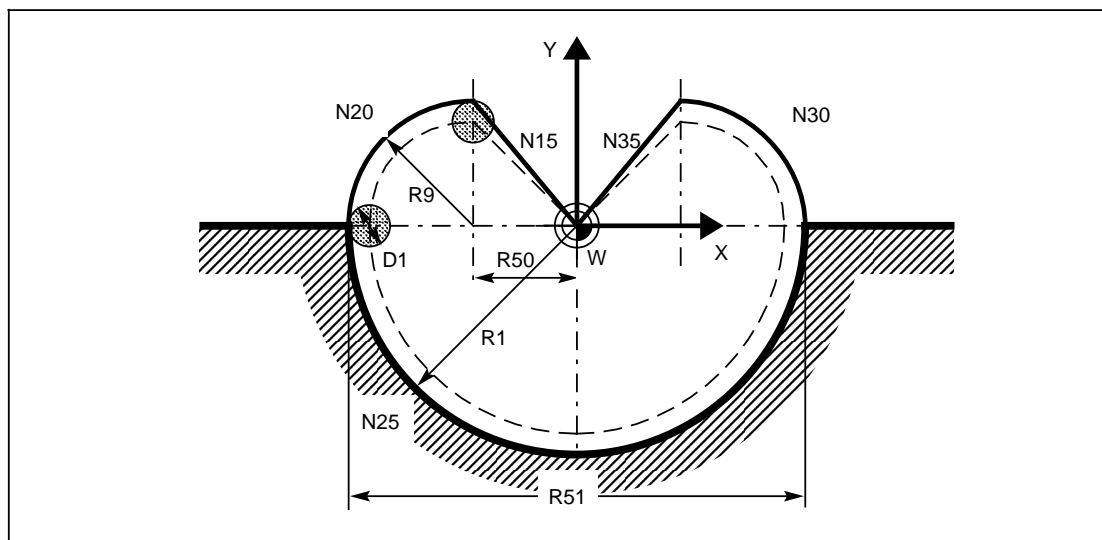
The subroutine below is used for roughing and finishing of a semicircle. The contour radius and the approach circle radius can be varied by means of parameters. The difference between the actual and specified workpiece sizes can be checked after each pass. This difference is then entered as additive tool wear.

Subroutine:

L11 L_F	
N5 R50=R1-R9 L_F	Calculation of approach circle
N10 R51=2·R1 L_F	Definition of semicircle
N15 G00 G64 G91 G41 X=-R50 Y=R9 F500 D1 L_F	Position approach circle
N20 G03 X=-R9 Y=-R9 U=R9 L_F	Contour approach
N25 X=R51 U=R1 L_F	Machining
N30 X=-R9 Y=R9 U=R9 L_F	Departure from contour
N35 G00 G40 X=-R50 Y=-R9 L_F	Positioning
N40 M17 L_F	Subroutine end

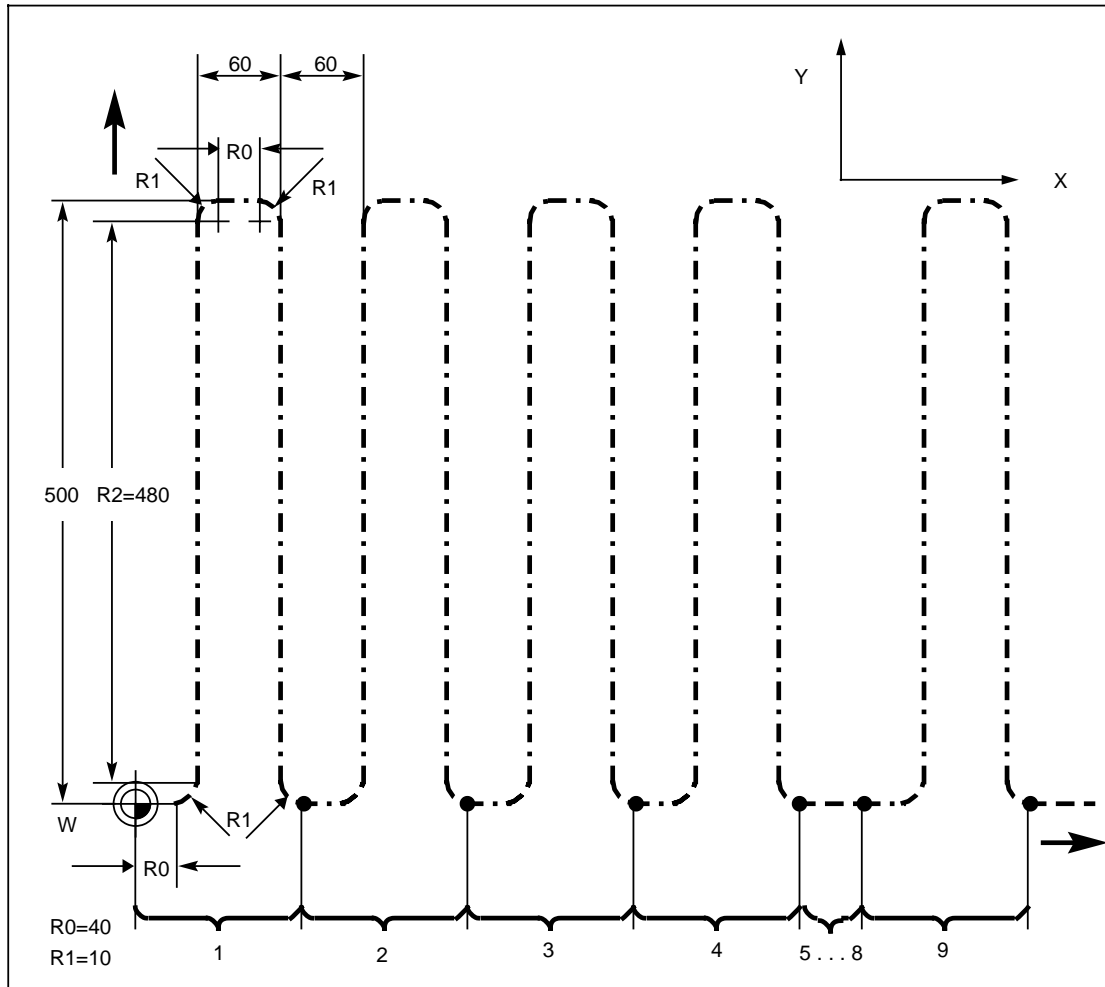
Subroutine call:

```
%5873 L_F
N5 ..... L_F
N10 L11 P1 R1=50 R9=10 L_F
N15 ..... L_F
:
```



**Example:** Traverse milling

The transitions are programmed with radii, in order to prevent any reduction in the feedrate and hence relief-cutting marks when changing the direction of movement.



Subroutine:

```

L34 LF
N5 G01 G64 G91 X=R0 LF
N10 G03 X=R1 Y=R1 I0 J=R1 LF
N15 G01 Y=R2 LF
N20 G02 X=R1 Y=R1 I=R1 J0 LF
N25 G01 X=R0 LF
N30 G02 X=R1 Y=-R1 I0 J=-R1 LF
N35 G01 Y=-R2 LF
N40 G03 X=R1 Y=-R1 I=R1 J0 LF
N45 M17 LF

```

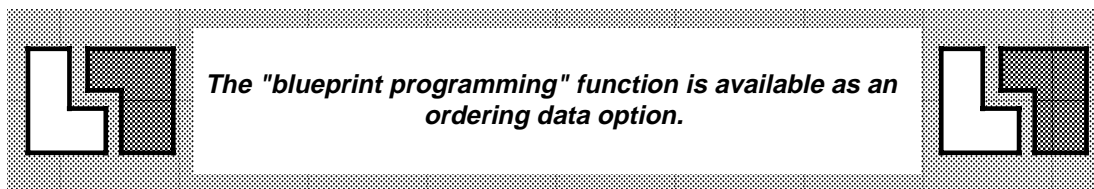
Calling L34 in higher-ranking program:

```

:
:
N15 L34 P9 R0=40 R1=10 R2=480 F200 LF
:

```

## 7 Contour Definition



### 7.1 Blueprint programming

Multi-point cycles for direct **programming** in accordance with the workpiece drawing are provided for **blueprint programming**. The points of intersection of the straight lines are entered as coordinate values or via angles.

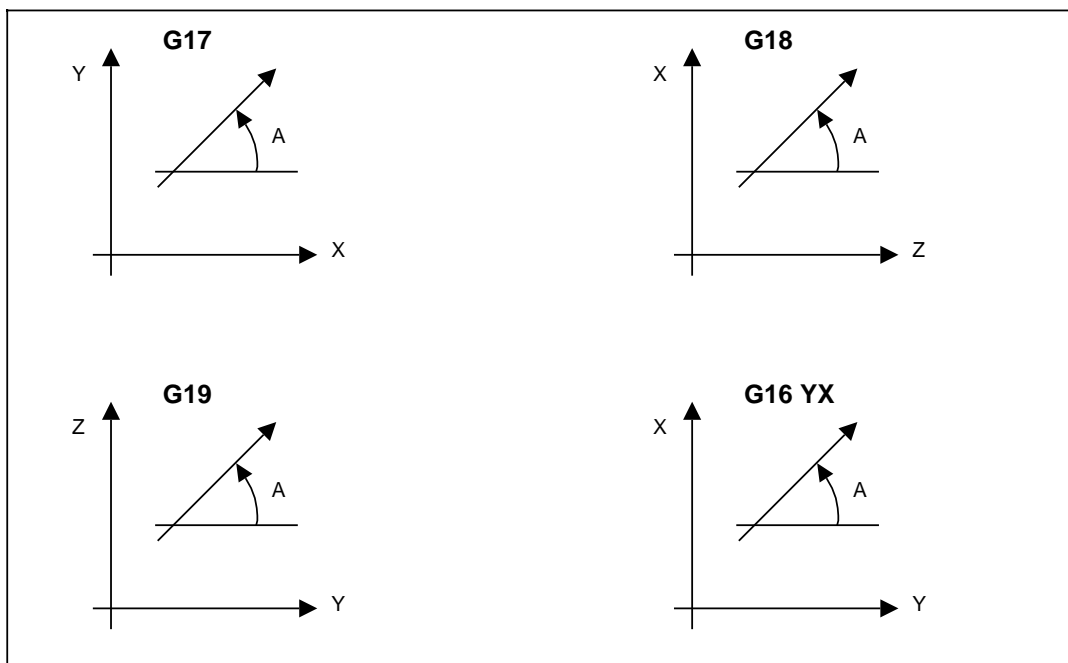
The various straight lines may be joined together directly in the form of a corner, rounded via radii or chamfered. Chamfer and transition radii are specified only by means of their size. The **geometrical calculation** is performed by the controller. The end position coordinates may be programmed using either absolute or incremental position data.

#### Angle (A):

Input resolution 0.00001 corresponds to  $10^{-5}$  degrees.

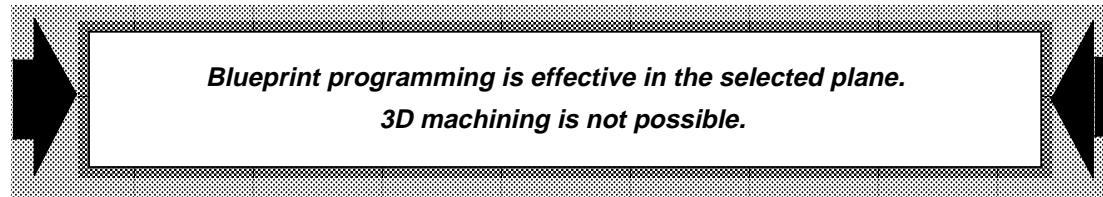
In the clockwise coordinate system the angle (max.  $359.99999^\circ$ ) is always measured from the horizontal axis direction to the vertical axis direction.

Plane selection: The required plane is selected with G17, G18 or G19.  
In the example the axis addresses are defined with X, Y, Z.



Plane selection

If the plane is freely selected (G16), it is specified by means of the programmed axes. The first axis programmed is the reference axis. The angle in the clockwise coordinate system is always referenced to the reference axis.



## 7.2 Contour definition programming

The elements described are valid for a milling machine in the X-Y plane (G17).

Examples 1 to 8 represent the **basic elements** of contour definition programming. These basic elements can be **combined** in a number of ways. The addresses for the angle (in this case A) and the radius (in this case U) are freely selectable in the controller via machine data. One and the same address must not be assigned more than once. Extended addresses are not possible for angle (A) and radius/chamfer (U, U-).

Function	Programming	Example
(1) 2-point cycle	N... A... X <sub>2</sub> ... (or Y <sub>2</sub> ) L <sub>F</sub>  The second end coordinate is calculated by the controller.	
(2) Circular arc	N...G02 (or G03) I.. J.. U.. X <sub>2</sub> .. (or Y <sub>2</sub> ) L <sub>F</sub>  The circular arc is limited to one quadrant. The second end position coordinate is calculated by the controller. In the contour definition parameters I and J must both be programmed, even if one of the values is zero.	
(3) 3-point cycle	N.. A <sub>1</sub> .. A <sub>2</sub> .. X <sub>3</sub> .. Y <sub>3</sub> .. L <sub>F</sub>  The controller calculates the coordinates of the vertex and generates 2 blocks. Angle A <sub>2</sub> is referred to the second straight line.	

Function	Programming	Example
(4) Chamfer	$N... X_2... Y_2... U-... L_F$ $N... X_3... Y_3... L_F 1)$  U-... means insert a chamfer U ... means insert a radius (The "minus" character is not a sign here; instead it is a special identifier for U = chamfer)	
(5) Radius	$N... X_2... Y_2... U... L_F$ $N... X_3... Y_3... L_F 1)$  The inserted radius must not be larger than the smaller of the two paths.	
(6) Straight line - circular arc (tangent)	$N.. G02 (or G03) A.. U.. X_3.. Y_3.. L_F$  The circular arc must not exceed 180°. The sequence A (angle) followed by U (radius) must be followed.	
(7) Circular arc - straight line (tangent)	$N.. G02 (or G03) U.. A.. X_3.. Y_3.. L_F$  The circular arc must not exceed 180°. The sequence U, A must be used. No radius may be inserted in $X_3, Y_3$ .	
(8) Circular arc - circular arc (tangent)	$N.. G02 (or G03) I_1.. J_1.. I_2.. J_2.. X_3.. Y_3.. L_F$ The preparatory function is programmed for the first circular arc. The second preparatory function is always the opposite of the first one and is not programmed. The interpolation parameters of the second circle are referred to the end position of this circle. Both interpolation parameters must be programmed, even if one value is zero.	

1) Second block may also be a contour definition.

Function	Programming	Example
(1) + (4) 2-point cycle + chamfer	N... A... X <sub>2</sub> ... (or Y <sub>2</sub> ...) U... L <sub>F</sub> N... X <sub>3</sub> ... Y <sub>3</sub> ... L <sub>F</sub> 1)	
(1) + (5) 2-point cycle + radius	N... A... X <sub>2</sub> ... (or Y <sub>2</sub> ...) U... L <sub>F</sub> N... X <sub>3</sub> ... Y <sub>3</sub> ... L <sub>F</sub> 1)  The inserted radius must not be larger than the smaller of the two paths.	
(3) + (4) 3-point cycle + chamfer	N... A <sub>1</sub> ... A <sub>2</sub> ... X <sub>3</sub> ... Y <sub>3</sub> ... U... L <sub>F</sub>	
(3) + (5) 3-point cycle + radius	N... A <sub>1</sub> ... A <sub>2</sub> ... X <sub>3</sub> ... Y <sub>3</sub> ... U... L <sub>F</sub>	
(3) + (4) + (4) 3-point cycle + chamfer + chamfer	N... A <sub>1</sub> ... A <sub>2</sub> ... X <sub>3</sub> ... Y <sub>3</sub> ... U <sub>1</sub> ... U <sub>2</sub> ... L <sub>F</sub> 1) N... X <sub>4</sub> ... Y <sub>4</sub> ... L <sub>F</sub> 1)  Addition of a second chamfer at end position (X <sub>3</sub> , Y <sub>3</sub> ).	

1) Second block may also be a contour definition.

Function	Programming	Example
(3) + (5) + (5) 3-point cycle + radius + radius	N... A <sub>1</sub> .. A <sub>2</sub> .. X <sub>3</sub> .. Y <sub>3</sub> .. U <sub>1</sub> ... U <sub>2</sub> .. L <sub>F</sub> N... X <sub>4</sub> .. Y <sub>4</sub> .. L <sub>F</sub> 1)  Addition of a second radius at end position (X <sub>3</sub> , Y <sub>3</sub> )	
(3) + (4) + (5) 3-point cycle + chamfer + radius	N... A <sub>1</sub> .. A <sub>2</sub> .. X <sub>3</sub> .. Y <sub>3</sub> .. U <sub>1</sub> -... U.. L <sub>F</sub> N... X <sub>4</sub> .. Y <sub>4</sub> .. L <sub>F</sub> 1)  Addition of a radius at end position X <sub>3</sub> , Y <sub>3</sub> . The next block is always taken into consideration automatically.	
(3) + (5) + (4) 3-point cycle + radius + chamfer	N... A <sub>1</sub> .. A <sub>2</sub> .. X <sub>3</sub> .. Y <sub>3</sub> .. U.. U-.. L <sub>F</sub> N... X <sub>4</sub> .. Y <sub>4</sub> .. L <sub>F</sub> 1)  Addition of a chamfer U- at end position.	

**U0** must be programmed for corners where **no chamfer or radius** must be inserted if a further **radius or chamfer follows** in the contour definition.

**When this is programmed, the controller generates a block with a distance 0.**

**This must be noted in relation to the action of the CRC.**

**U-0 is interpreted as U0.**

**A radius or chamfer can only be inserted between two linear blocks.**

The sequence of addresses A, X, Y, U, F, etc. is freely selectable; angles and radii must however be entered in the **sequence** described above (first angle before second angle, first radius before second radius in **machining direction**).

1) Second block may also be a contour definition.



### 7.3 Operation of function G09, F, S, T, H, M in contour definition

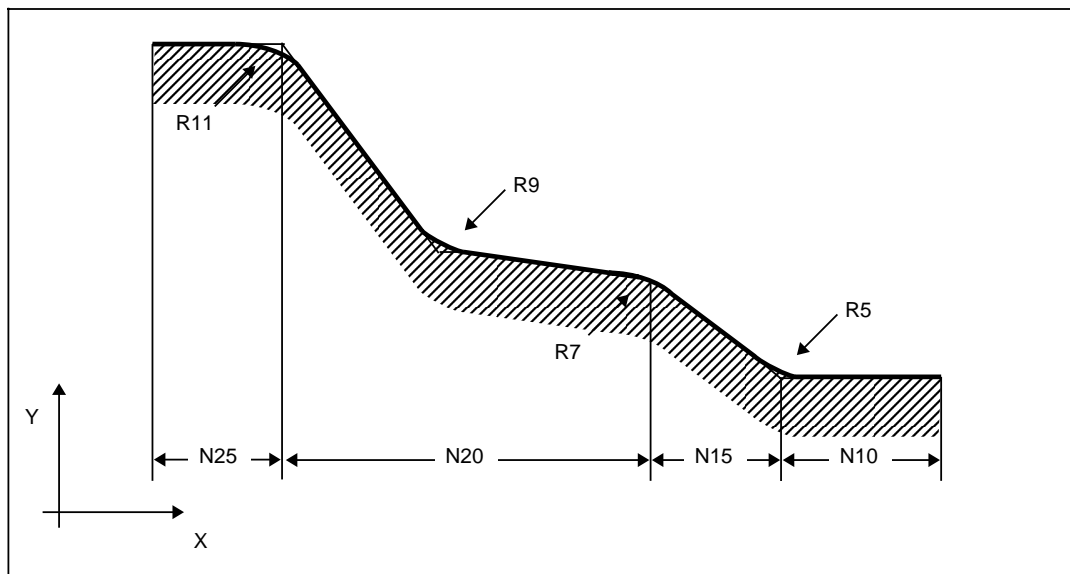
If **G09** is programmed in a contour definition block, it is not active until the **end of the block**, in other words until the **end position** is reached. G09 is automatically generated by the controller at irregular points (corners, edges) in the contour definition.

- If **F, S, T, H or M** is programmed in a contour definition, it is active at the **start of the block**;
- **M00, M01, M02, M17 and M30** are active at the **end of the block**.

### 7.4 Linking of blocks

It is possible to link blocks with or without angle inputs and with inserted radii or chamfers in any sequence: smallest filleting radius R.. cutter.

**Example:** Linking of blocks



.		
N10	X... U5 L <sub>F</sub>	Straight line with radius
N15	A... Y... U7 L <sub>F</sub>	Straight line with radius
N20	A1 A2 X... Y... U9 U11 L <sub>F</sub>	3-point definition with radius at either end
N25	X... L <sub>F</sub>	Straight line
.		
.		

### 7.5 Programming examples: milling machine

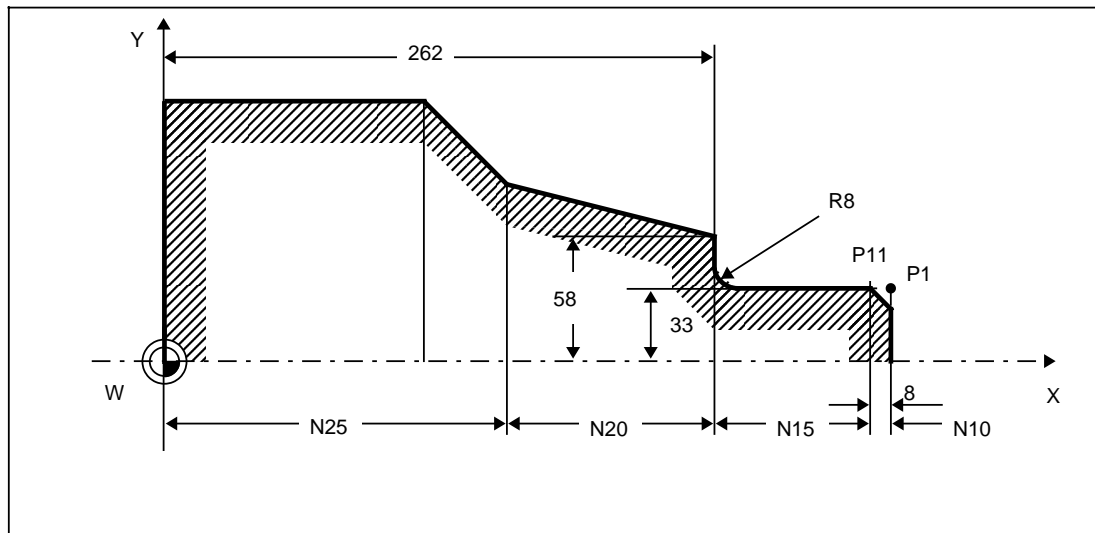
Angle A1 refers to the starting position; angle A2 refers to the missing vertex. The end position can be programmed using absolute position data G90 or incremental position data G91. Both end position coordinates must be specified. The controller determines the vertex from the known starting position, the two angles and the end position.



## 7.6 Miscellaneous functions in linked blocks

Blocks are said to be linked whenever they are joined together by means of radii or chamfers. A block with miscellaneous functions may be located between two linked block. Miscellaneous functions are active at the end of the chamfer and radius. Relief-cutting is also performed at this point.

**Example:** Miscellaneous functions in linked blocks



There may be a block with miscellaneous functions in between linked blocks:

N10	G01	G09	A90	Y33	U-8	F0.2	L <sub>F</sub>	(P1)	2-point definition + chamfer (U-8) *)
N11	M...	H...	...						
N15	A180	A90	Y58	X262	U8	L <sub>F</sub>			3-point definition + chamfer (U8)

Miscellaneous functions become effective in position P11 (see above). Relief cutting will thus be performed in position P11. The F value programmed in block N10 will become active at the beginning of block N10.

\*) Linking by U-

## 8 Tool Offsets

### 8.1 Tool data

The **geometrical** tool data for the tool are stored under **tool offset number D**:

Length  $\pm 9999.999$  mm, Radius  $\pm 999.999$  mm; T number 4 decades (Input resolution  $1\mu\text{m}$ ).

The tool number, tool type, geometry, basic dimension and wear of all active tools are stored in the TO area of the NC.

- The TO area is subdivided into 99 tool offset blocks (D1 to D99).
- Each block is subdivided into 10 columns and/or **10 tool parameters (P0 to P9)** and is set up as follows:

D1 ... D99	P0 :	Tool number	} Wear data
	P1 :	Type	
	P2 :	L1 Geometry	
	P3 :	L2 Geometry	
	P4 :	Diameter/Radius	
	P5 :	L1 Wear	
	P6 :	L2 Wear	
	P7 :	Diameter/Radius	
	P8 :	L1 Base	
	P9 :	L2 Base	

- The format of the tool offset block is identified by the **tool type (P1)**.

#### Breakdown of tool type P1:

Type 0 Tool not defined

Type 1 . . . 9 Lathe tools, position of tool tip

Type 10 . . . 19 Tools with active length compensation only (e.g. drills)

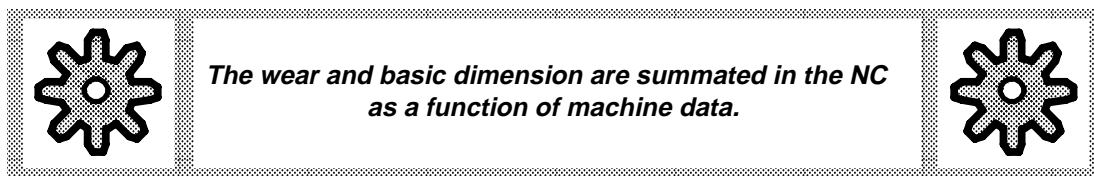
Type 20 . . . 29 Tools with radius compensation and one length compensation (e.g. cutters)

Type 30 . . . 39 Tools with radius compensation and two length compensations (e.g. angle cutters)

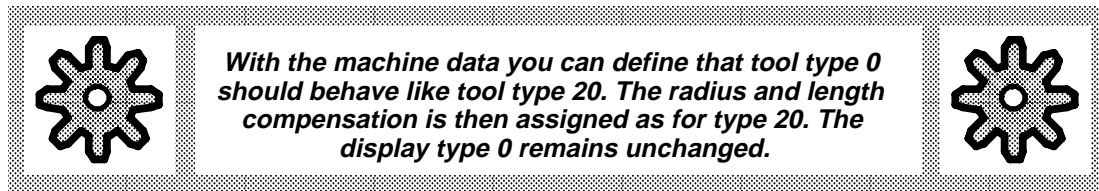
- The tool offset is **called** in up to 2 decades via **D1 to D99**
- It is **cancelled** with **D0**.

The offset is not executed until the corresponding **axis is programmed**.

The geometry and wear are updated, for example by measuring cycles in the NC.



Tool offset may be entered via both the operator panel and the **data input interface**. **No** block numbers may be programmed (see tape formats). The machine manufacturer can limit the maximum wear specification to  $\pm 0.999$  mm instead of  $\pm 9.999$  mm.



## 8.2 Selection and cancellation of length compensation

This function can only be **selected** if **G00** or **G01** is active. It is necessary to select the **plane** perpendicular to the length compensation direction:

```
.
N5 G00 G17 D.. Z.. LF
.
```

Only the length compensation is used from **compensation memory D...** . The offset value contained in the D-word is always combined with the sign entered for the corresponding axis. Length compensation can be **cancelled** with **D0**. It is not executed unless the corresponding **axis** has been **programmed** (standard MACHINE DATA).

### Length compensation without cutter radius compensation (CRC)

```
.
:
N5 G90 G00 G17 D1 Z.. LF           Selection of length compensation (e.g. drill)
:
:
N50 D0 Z.. LF                     Cancellation of length compensation
.
```

### Length compensation with cutter radius compensation

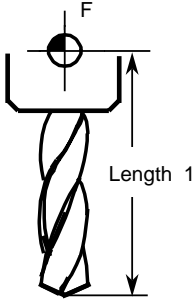
```
.
:
N5 G90 G00 G17 G41 D2 X ... Y... LF Automatic selection of cutter radius compensation
N10 Z... LF                       With length compensation
:
:
N50 G40 X... LF                   Cancellation of cutter radius compensation
N55 D0 Z... LF                   Cancellation of length compensation
.
```

## 8.3 G40/G41/G42 intersection cutter radius compensation

**G40** No intersection cutter radius compensation  
**G41** Direction of tool travel left from workpiece  
**G42** Direction of tool travel right from workpiece

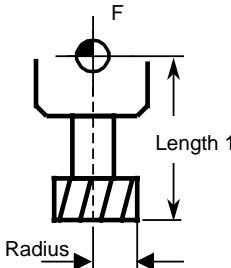
The compensation of the cutter radius is effective in the selected plane (**G16** to **G19**). The **length compensation of the cutter** is always **perpendicular** to the selected **plane**.

## Structure of tool offset memory

Dn	Offset memory structure					Programming
	T No.	Type	Geometry	Wear	Basic (supp. TO)	Tool call
	P0	P1	P2	P5	P8	
	123...8	10...19	Length 1	Length 1	Length 1	T100 . . M06 G17 (D50) *) or M06 G16 UV +/- W (D50) *)
<b>1. Drill</b> 						

Dn	Offset memory structure						Programming	
	T No.	Type	Geometry		Wear		Basic (supp. TO)	Tool call
	P0	P1	P2	P4	P5	P7	P8	
	123...8	20...29	Length 1	Radius or diameter	Length 1	Radius or diameter	Length 1	T100 : : M06 G17 (D50) Length 1 in Z, Radius active in the X/Y- plane or M06 G16 UV +/- W (D50) Length active in W Radius active in U/V plane *)

### 2. End mill



The diagram illustrates an end mill tool. A vertical dashed line represents the tool's axis. At the top, a circle with a dot indicates the tool tip, with a downward arrow labeled 'F' representing the feed force. The tool body is shown with a central cutting edge. A vertical double-headed arrow on the right indicates the 'Length 1' of the tool. At the bottom, a cross-section of the tool is shown with diagonal hatching, and a horizontal double-headed arrow indicates the 'Radius' of the tool.

\*) G16 is a strict setting function: travel is not possible in these blocks.  
D is not linked to plane selection.

	Offset memory structure										Programming
	T No.	Type	Geometry			Wear			Basic (supp. TO)		Tool call
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	
Dn	123...8	30...39	Length 1	Length 2	Radius or diameter	Length 1	Length 2	Radius or diameter	Length 1	Length 2	T100 : :  M06 G16 UV +/- Z +/- U (D50) Radius in U/V plane, Length 1 in Z, Length 2 active in U *)
<div>3. Angle head</div> <div></div>											

When **mirroring** is used, the path travelled by the tool is as follows (taking the **sign of radius compensation** into consideration):

Sign of cutter radius compensation value	G41		G42	
	Both axes mirrored or both axes not mirrored	One axis mirrored	Both axes mirrored or both axes not mirrored	One axis mirrored
+	Left of cutter	Right of cutter	Right of cutter	Left of cutter
-	Right of cutter	Left of cutter	Left of cutter	Right of cutter

Selection and cancellation of intersection cutter radius compensation

This function can only be selected if G00 or G01 is active. G40, G41 and G42 can be programmed in a block without paths. They are not active however unless a movement has been programmed in at least one axis.

\*) G16 is a strict setting function: travel is not possible in these blocks.  
D is not linked to plane selection.

**Example: Selection**

```

:
:
N10 G01 G17 G41 D7 X... Y... F... LF

```

At the end of this block the compensated path will be reached in the selected plane. Only the radius compensation value is incorporated.

```

N15 Z... LF

```

The tool length compensation is incorporated.

OR

```

:
:
N10 G17 LF
N15 G41 D7 LF
N20 G01 X... Y... F... LF

```

Selection of plane

Selection of compensation

At the end of this block the compensated path will be reached in the selected plane. Only the radius compensation value is incorporated.

```

N25 Z... LF

```

The tool length compensation is incorporated.

```

:

```

The cutter radius compensation (G41 or G42) can be cancelled with G40 if function G00 or G01 is active. In order to activate the correct compensations, at least one axis in the selected plane must be programmed.

The length compensation is cancelled with D0 and is active if the length compensation axis is programmed.

**Example: Cancellation**

```

:
:
N30 G40 X...LF

```

Cancellation of tool compensation. Only the radius compensation value is not active.

```

N35 D0 Z... LF

```

Length compensation value = 0 is active.

```

:

```

OR

```

:
:
N30 G41 D0 X... LF

```

Cancellation of all compensations with compensation values = 0.

Only the radius compensation value is not active.

```

N35 Z... LF

```

Length compensation value = 0 is active.

```

:

```

Prior to selecting another plane, CRC must be cancelled.

**Change from G41 to G42**

```

:
:
N10 G01 G17 G41 D12 X... Y... F.. LF
N15 Z... LF
N20 G42 X... Y... LF

```

Incorporation of radius compensation

Incorporation of length compensation

Radius compensation changed (e.g. change in direction of movement to workpiece; traverse milling)

```

N25 Z... LF

```

No change in length compensation

```

:

```



## Change in tool offset number

The G function for selecting the CRC need not be re-entered.

:

N10 G01 G17 G41 D12 X... Y... F... L<sub>F</sub>

N15 Z... L<sub>F</sub>

N20 D10 Z... L<sub>F</sub>

Change in length compensation

N25 X... Y... L<sub>F</sub>

Change in cutter radius compensation

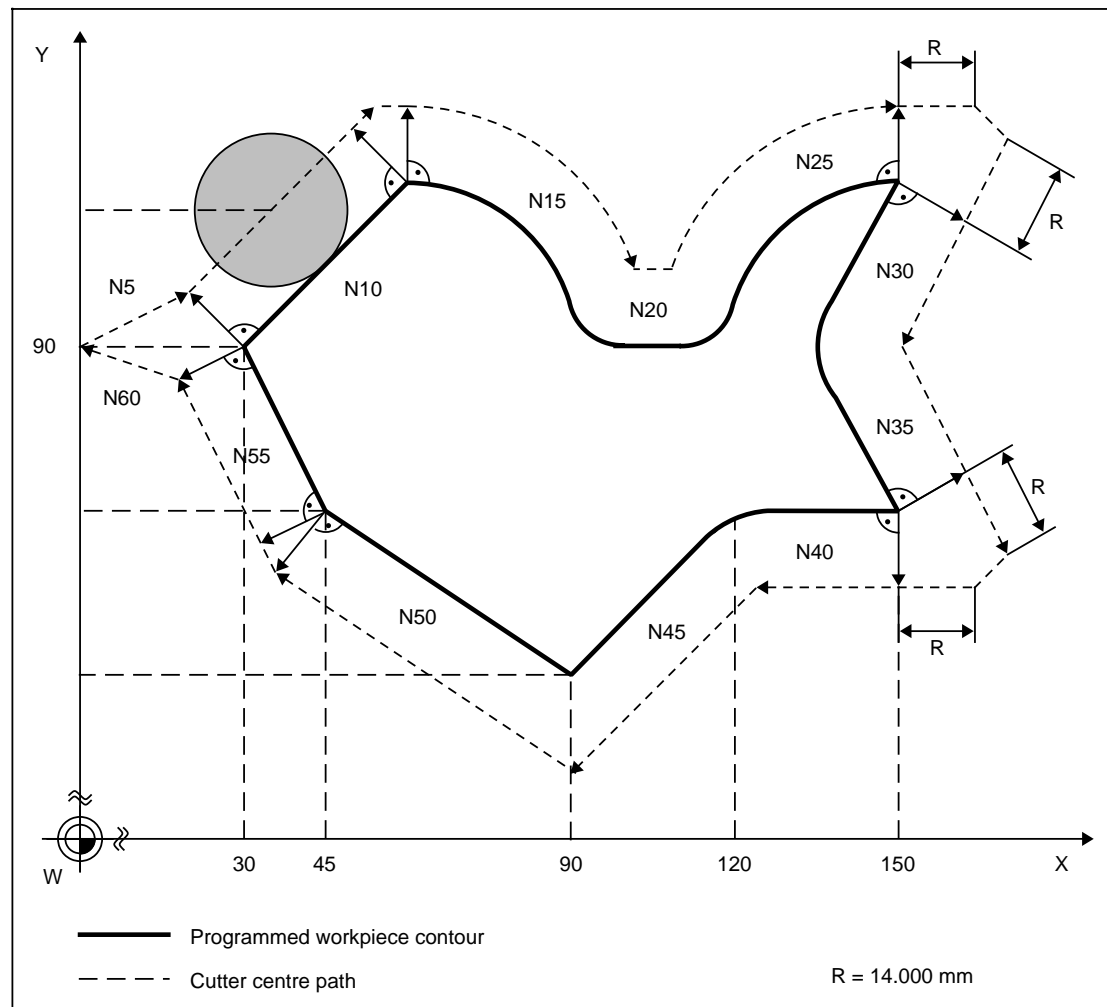
.

If CRC is selected, it is not permissible to program G58, G59, G33, G34 or G35.

### Remedy:

Program the above functions before selecting the cutter radius compensation, or cancel the cutter radius compensation select G58, G59, G33, G34 or G35 and then reselect CRC. When CRC has been selected, including the G40 block, the effective zero offset value must not be changed.

### Example: Milling with cutter radius compensation (CRC)



```

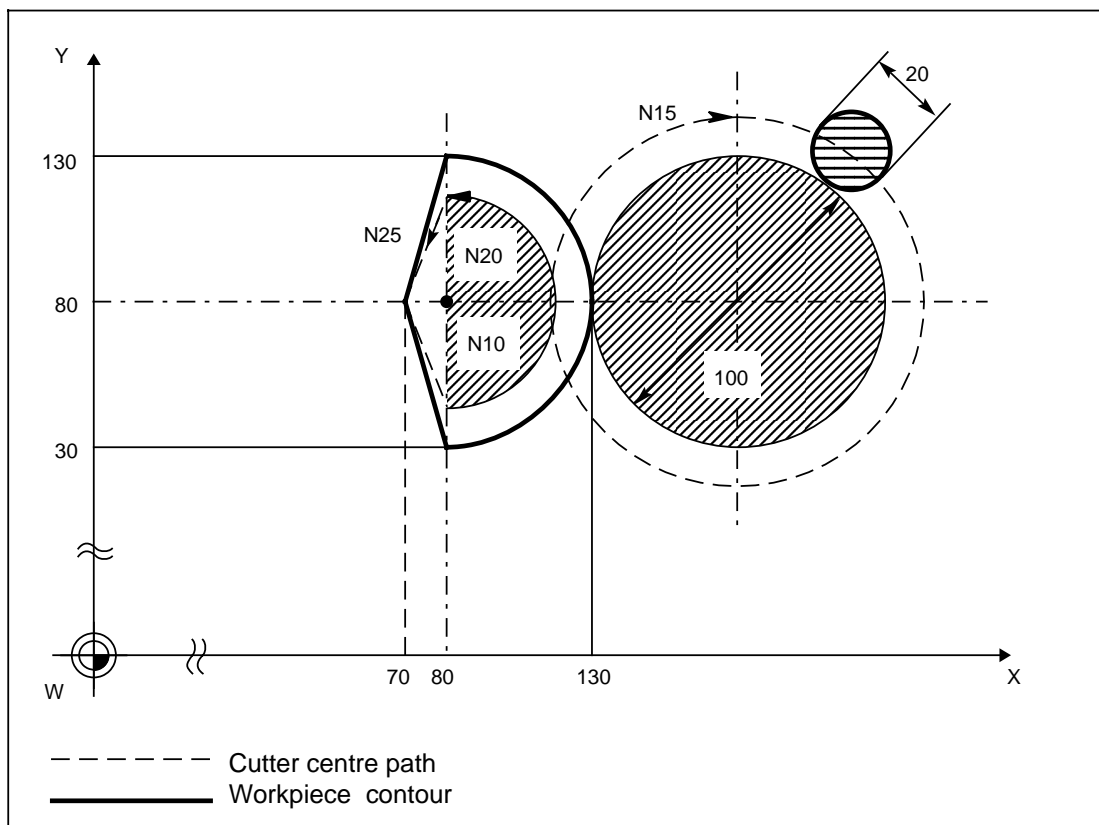
:
:
N5  G01 G41 D1 G90 G17 X30 Y90 F500 S56 M03 L_F
N10 G91 X30 Y30 L_F
N15 G02 X30 Y-30 I0 J-30 L_F
N20 G01 X30 L_F
N25 G02 X30 Y30 I30 J0 L_F
N30 G01 X-15 Y-30 L_F
N35 X15 Y-30 L_F
N40 X-30 L_F
N45 X-30 Y-30 L_F
N50 X-45 Y30 L_F
N55 X-15 Y30 L_F
N60 G40 G90 X0 Y90 L_F
N65...
:
:

```

The milling cutter used has a radius of 14.000 mm.

The cutter radius must be entered under tool offset number D1.

**Example:** Full circle programming using cutter radius compensation



```

:
:
N5  G90 G00 G17 G41 D1 X80 Y30 F500 S56 M03 L_F
N10 G03 X130 Y80 I0 J50 L_F
N15 G91 G02 X0 Y0 I50 J0 L_F
N20 G90 G03 X80 Y130 I-50 J0 L_F
N25 G00 G40 X70 Y80 L_F
:
:

```

## 8.4 Tool length compensation, positive or negative

When the length compensation is selected, positive compensation takes effect. A **negative** tool length compensation must be programmed **with G16**. The tool length compensation is effective in the axis perpendicular to the CRC plane.

### Example:

G16 U V±W

**U,V** Plane selection (CRC plane)

**±W** The sign specifies whether the tool length compensation for axis W is positive or negative.

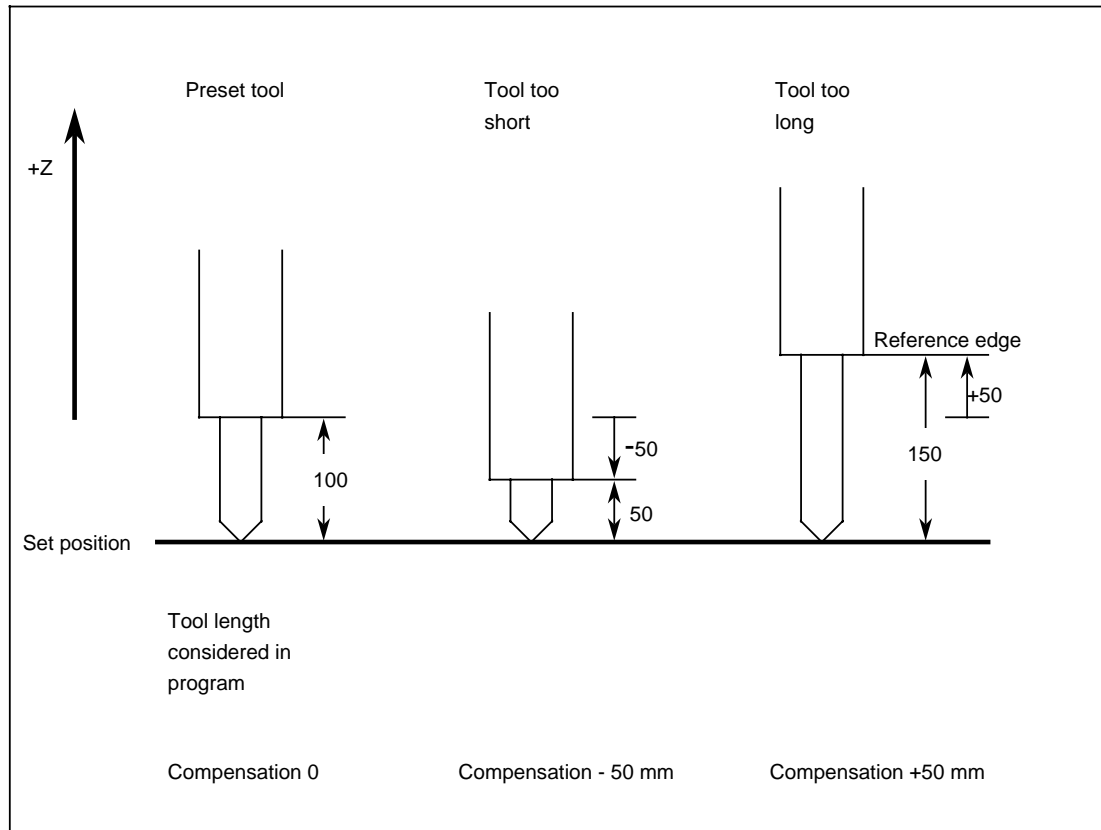
### Determining the sign

A **positive** sign is entered by the operator if the **actual value** of the tool is **higher** than the value used by the programmer. A minus sign is entered if the actual value of the tool is less than the value used by the programmer.

### Example:

Drill used is **longer** than the programmed drill : + compensation value

Drill used is **shorter** than the programmed drill : - compensation value



## 8.5 Tool offsets for end mill

In the case of an end mill, the cutter radius compensation is active in the plane specified with the corresponding preparatory function (geometry = P4, wear = P7) while length compensation (geometry = P2, wear = P5) is effective in the third axis.

The following applies:

**G17** Cutter radius XY cutter length Z

**G18** Cutter radius ZY cutter length Y

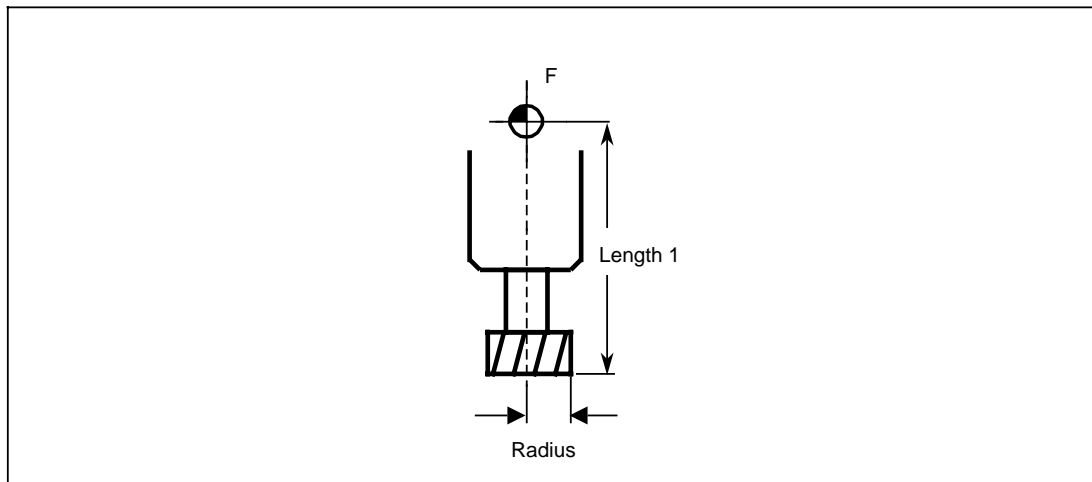
**G19** Cutter radius YZ cutter length X

It is also possible to incorporate the 4th axis using G16.

G16 X Y U Cutter radius XY cutter length U and to reverse the direction of the length compensation

G16 X Y Z Cutter radius XY cutter length Z

The number 20 must be entered under P1 (tool type)



### Tool offsets of the end mill:

P1 : Tool type (enter 20)

P2 : Geometry, length 1

P4 : Geometry, cutter radius

P5 : Wear, length 1

P7 : Wear, cutter radius.

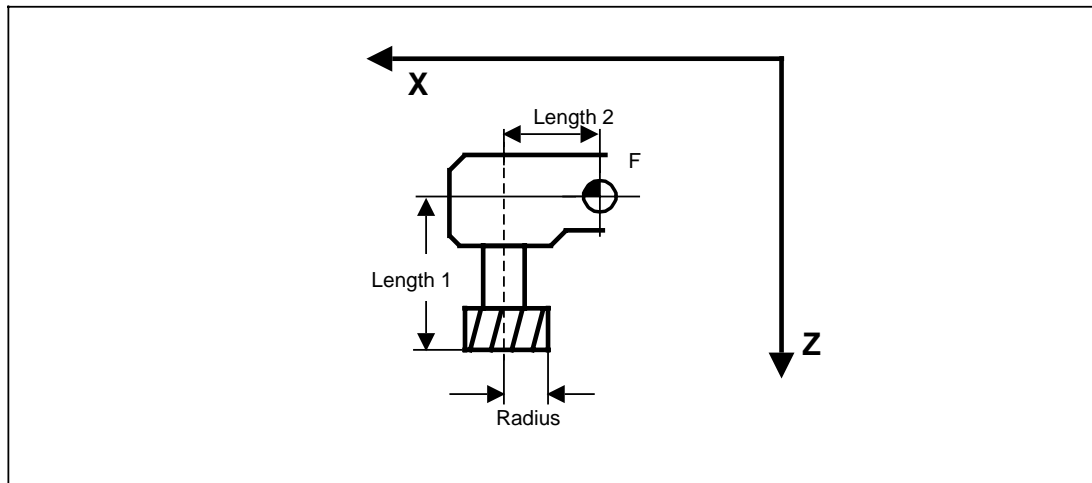
## 8.6 Tool offsets for angle head cutter

As opposed to the end mill cutter, an angle head cutter requires an additional length compensation located in the plane of the cutter radius compensation (length 2). Free axis selection must always be used with preparatory function G16.

Function G16 is followed in this instance by **four** axis addresses, the first two of which specify the plane in which the **cutter radius compensation** is to be active; the third axis address indicates the **standard length compensation** (as in the case of the end mill cutter), while the fourth address specifies the **additional length compensation (length 2)** in the cutter radius plane.

The third and fourth axis addresses can of course be given a negative sign to reverse the direction of compensation.

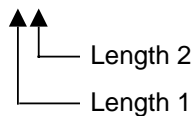
The number 30 must be entered under P1 (tool type).



### Tool offsets of the angle head cutter:

- P1 : Tool type (enter 30)
- P2 : Geometry, length 1
- P3 : Geometry, length 2
- P4 : Geometry, cutter radius
- P5 : Wear, length 1
- P6 : Wear, length 2
- P7 : Wear, cutter radius.

Compensation assignment: G16 X Y Z X



## 9 Cutter Radius Compensation (CRC)

All **stop positions for single blocks** are marked **S**.

The pertinent block numbers are shown in parentheses.

In the block following the selection block, a block start vector (**length R**) is created **perpendicular** to the programmed path.

The diagrams are shown with G42. In the case of programs with G41, the angle transition is  $\beta=360^\circ$ .

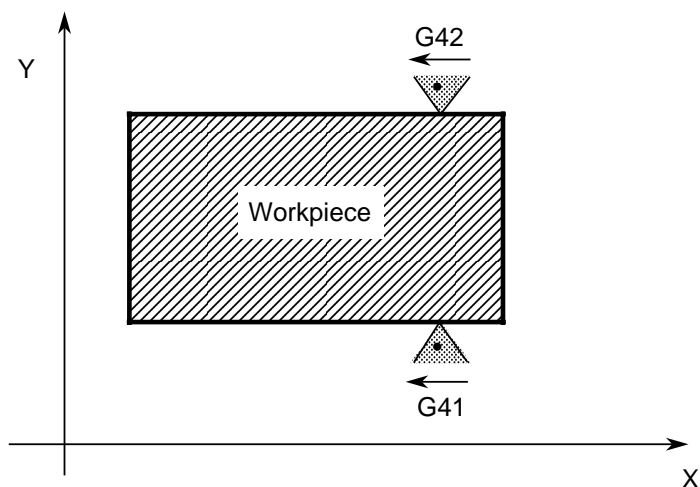
**Contour elements** are marked with a thick line (—)

### 9.1 Selection of CRC

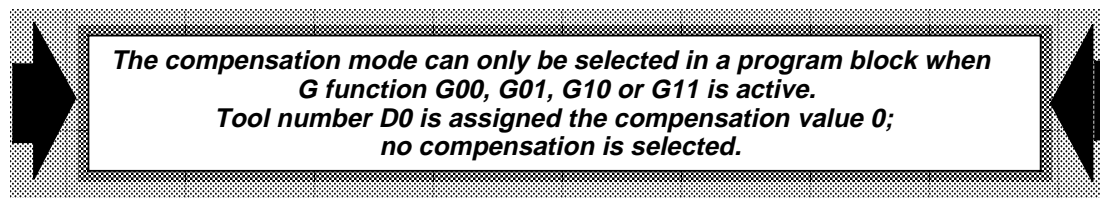
The **compensation mode is selected** in the fixed plane with preparatory functions **G41/G42** and offset number **D**.

The compensation is effected **to the left** of the workpiece contour (in the traversing direction) **with G41** and **to the right** of the workpiece contour **with G42**.

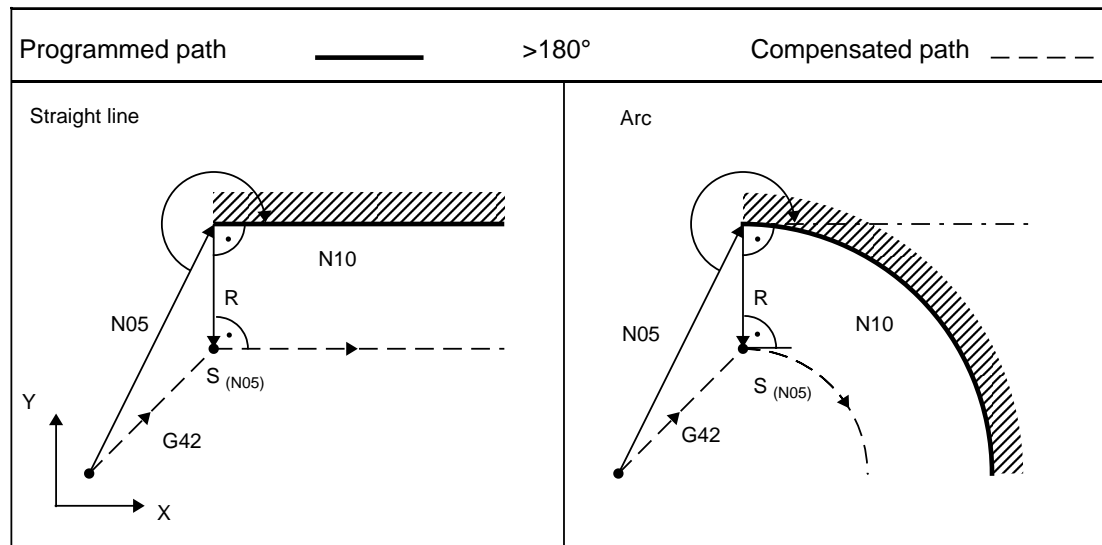
**Example of a milling tool:**



When selecting CRC, two or three program blocks are always read in for calculating the point of intersection.

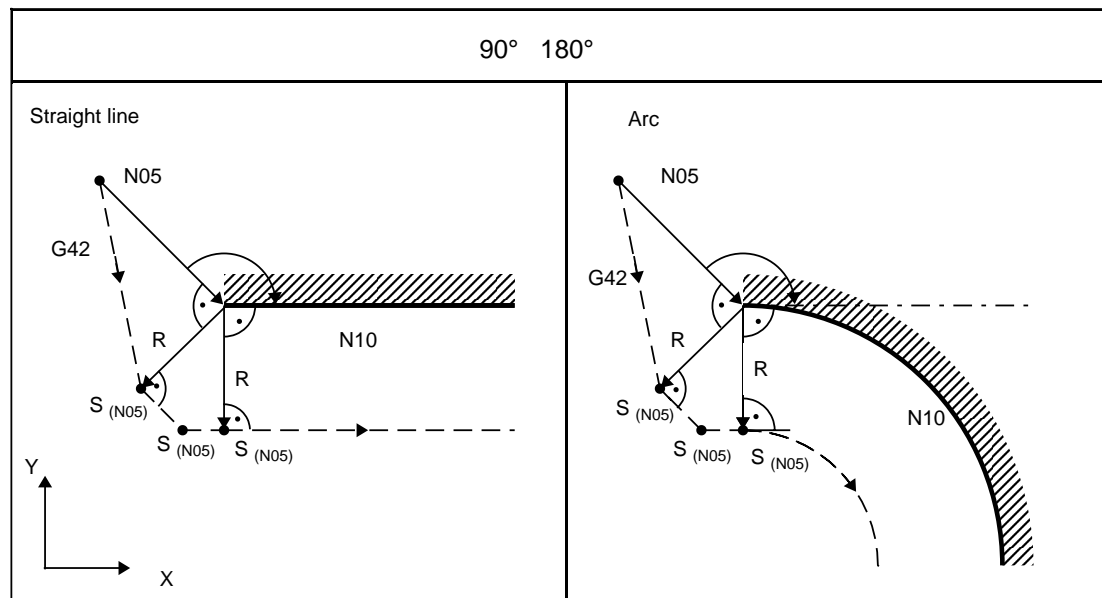


The diagrams below show the **compensation selected for various approach angles**.

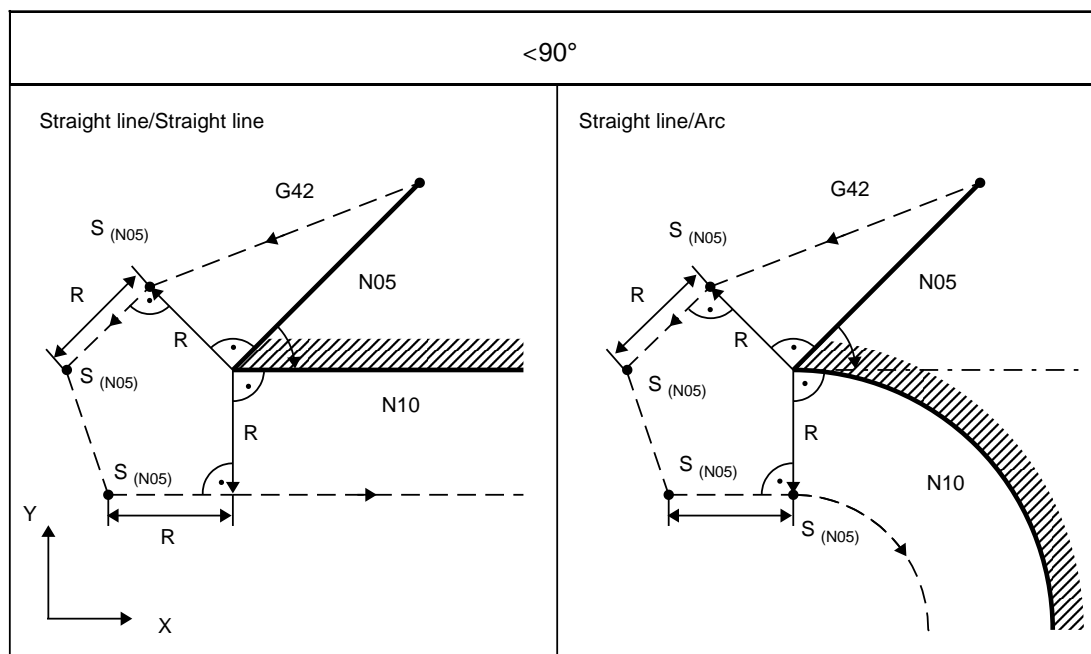


Selection of compensation mode at  $>180^\circ$

### Selection of compensation mode



Selection of compensation mode at  $90^\circ$   $180^\circ$

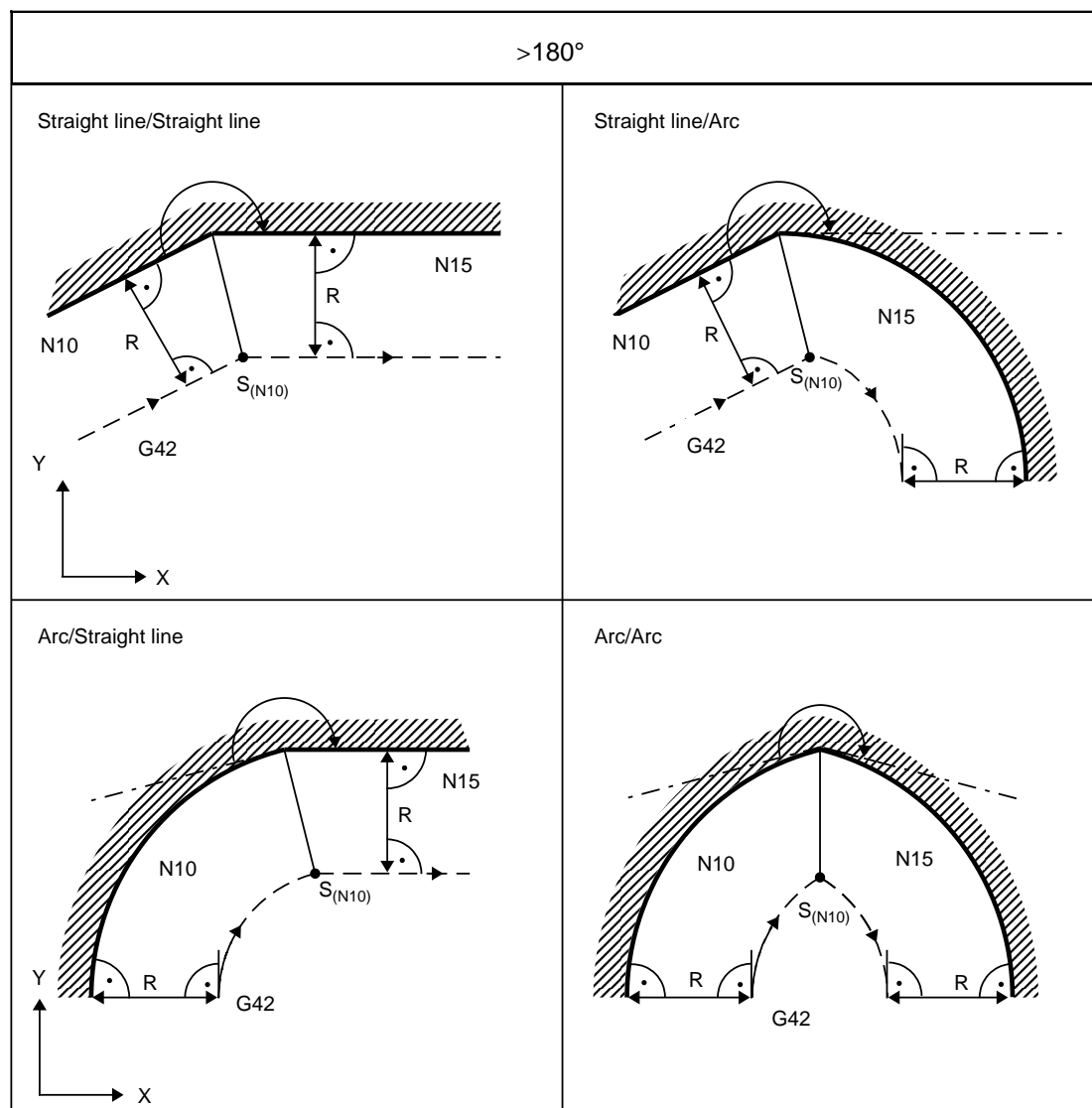


*Selection of compensation mode at  $<90^\circ$*

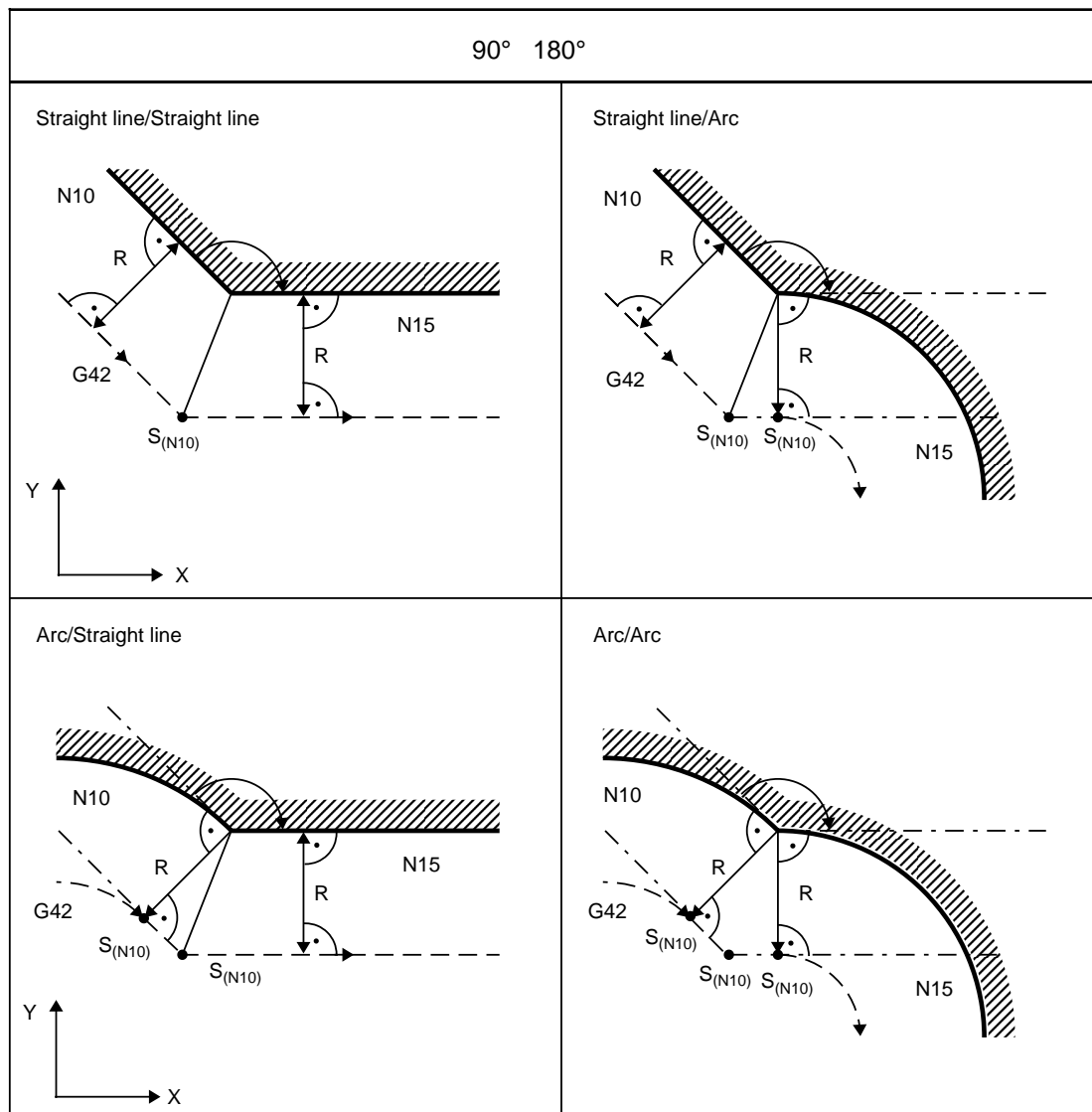


## 9.2 CRC in the program

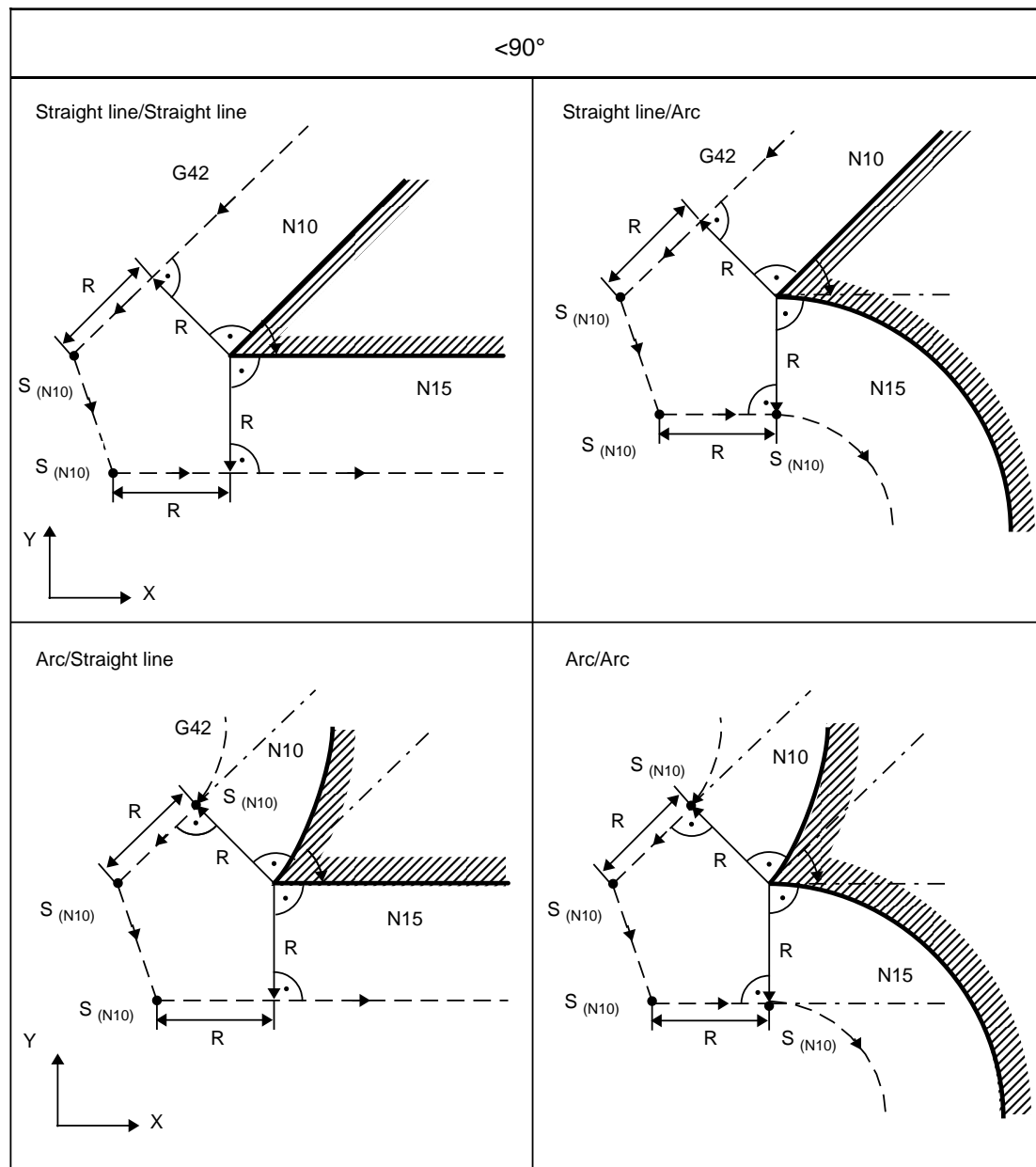
When CRC is selected, the controller reads in two further blocks in advance during **processing of the current block** and calculates the intersection point of the **compensated paths**. The diagrams below shows compensation for **various transitions**.



CRC for various transitions at  $>180^\circ$



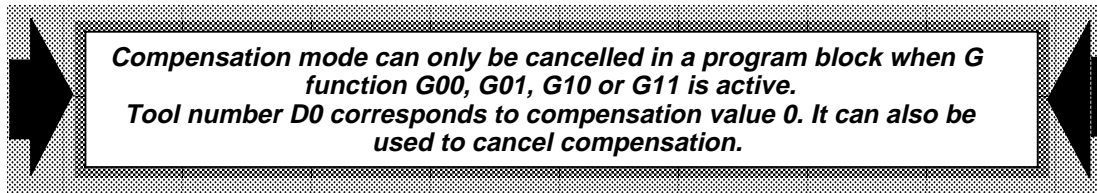
CRC for various transitions at 90° 180°



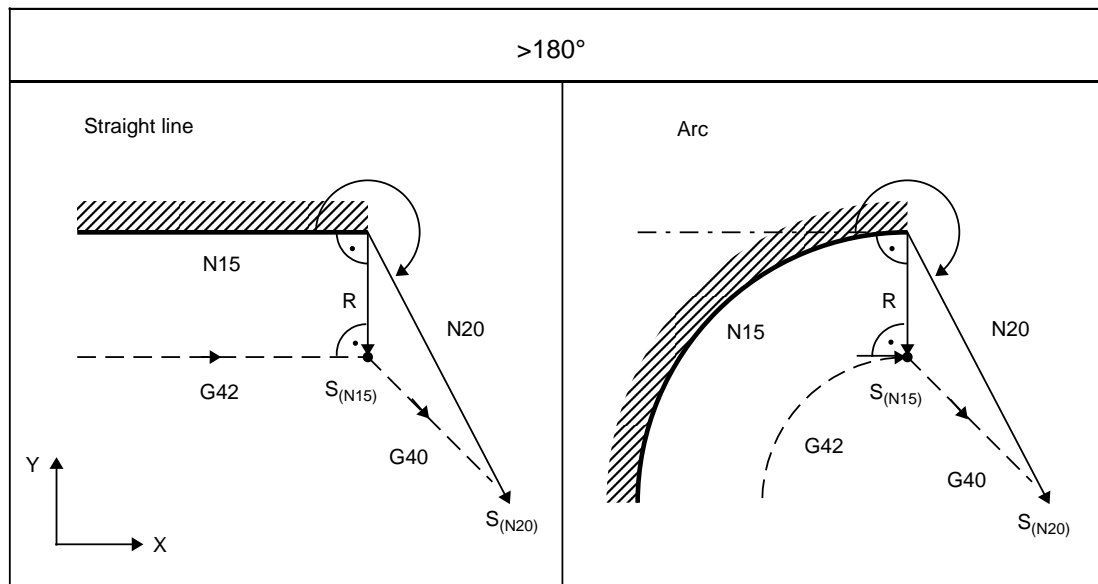
CRC for various transitions at  $<90^\circ$

### 9.3 Cancellation of CRC (G40)

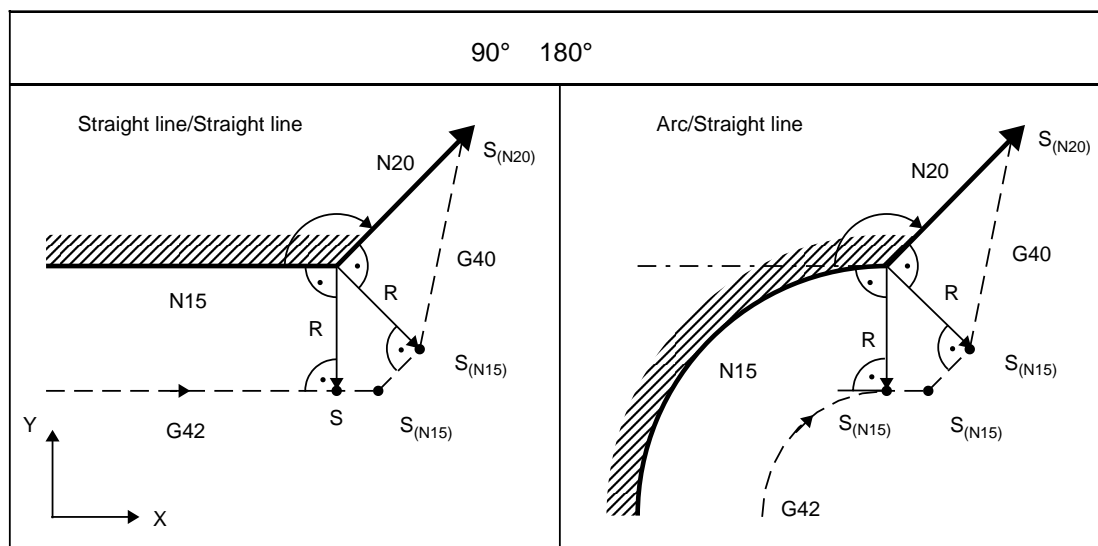
Compensation mode is **cancelled** using preparatory function **G40**.



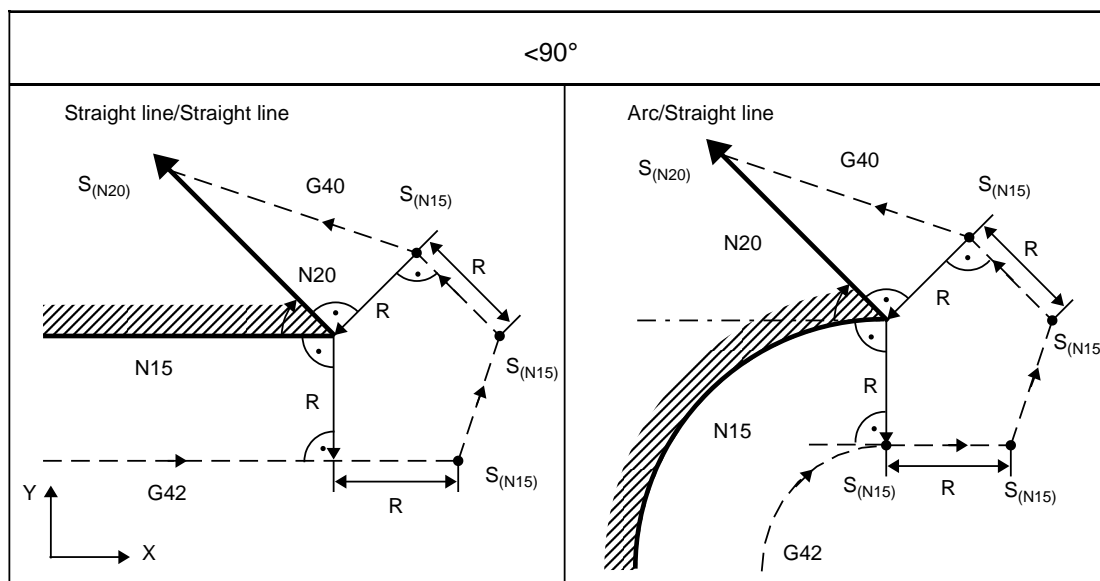
#### Cancellation of compensation mode



Cancellation of compensation mode at  $>180^\circ$

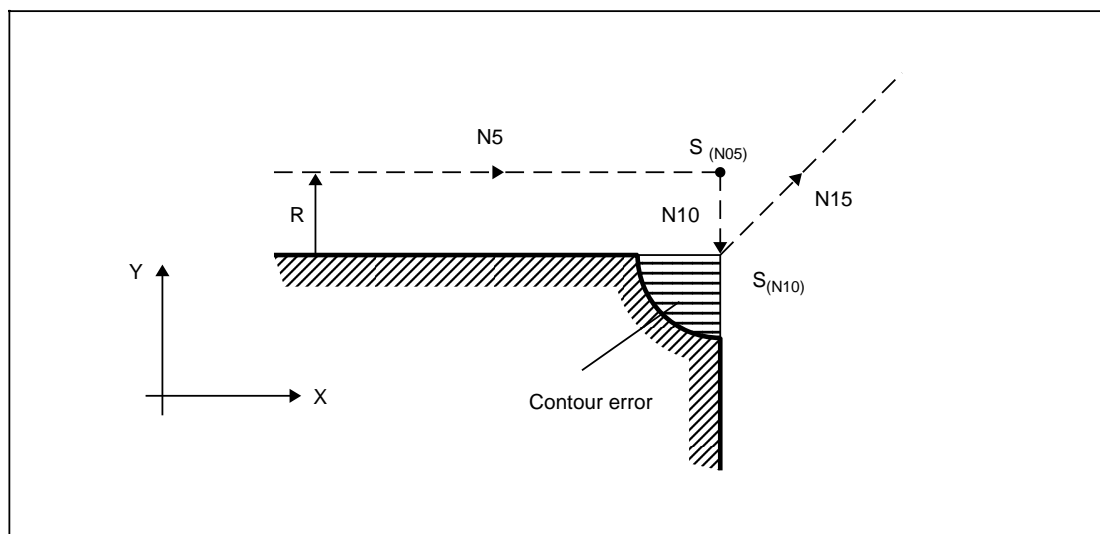


Cancellation of compensation mode  $90^\circ \quad 180^\circ$



Cancellation of compensation mode at  $<90^\circ$

### Cancellation of compensation mode (special case)



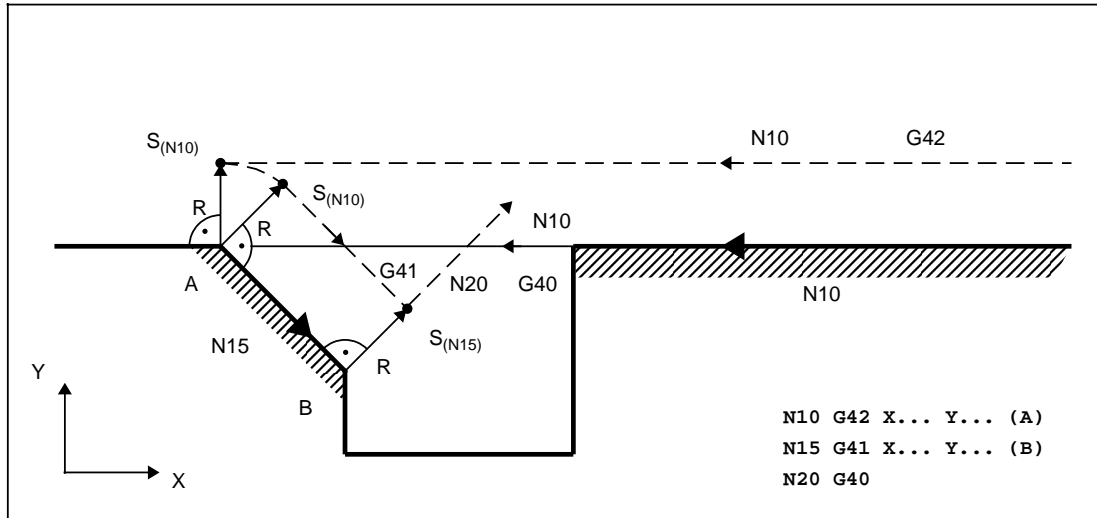
Cancellation of CRC in one block "Distance = 0"

```
N5 G91 X100 L_F
N10 G40 X0 L_F
N15 X100 Y+100 L_F
```

Contour error (hatched range).

## 9.4 Changing direction of compensation (G41, G42)

A perpendicular **vector** of length **R** is created in the appropriate **direction of compensation** at the end position of the block with the **old G function** (e.g. G42) and at the starting position (A) of the block with the **new G function** (e.g. G41).

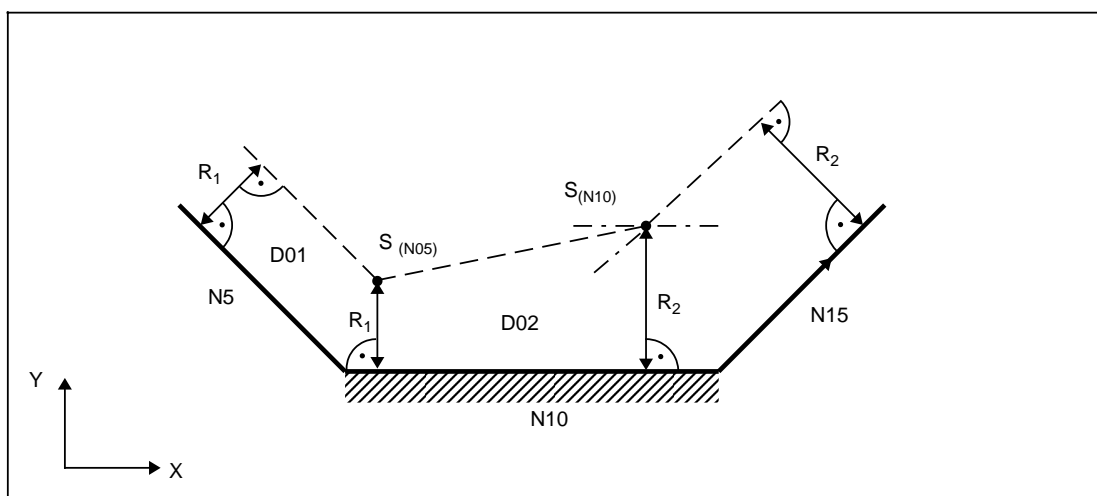


Changing direction of compensation

## 9.5 Changing compensation number (G41 D.. , G41 D.. )

When changing the compensation number (e.g. G41 D.., G41 D..), the following applies:

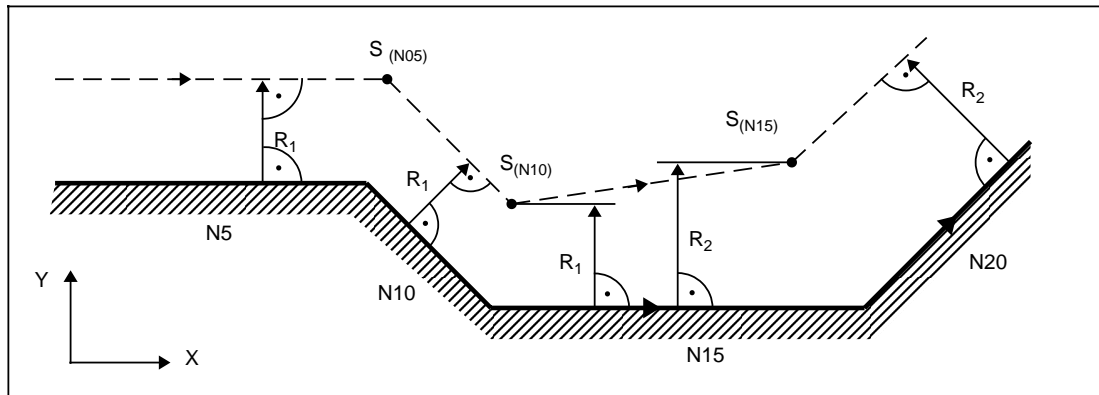
**No** block start intersection is calculated with the old compensation; a vector perpendicular to the contour with length **R1** is created at the end position of the block with the old offset number D01; the **block end intersection** is **calculated** with the new compensation D02.



Changing compensation number

## 9.6 Changing compensation values (R1, R2)

The **compensation values** can be **changed** via the operator panel, via tape input, via the external tool offset or **in the part program**. The new compensation value takes effect in the **next** block (N20 in the example).



Changing compensation values

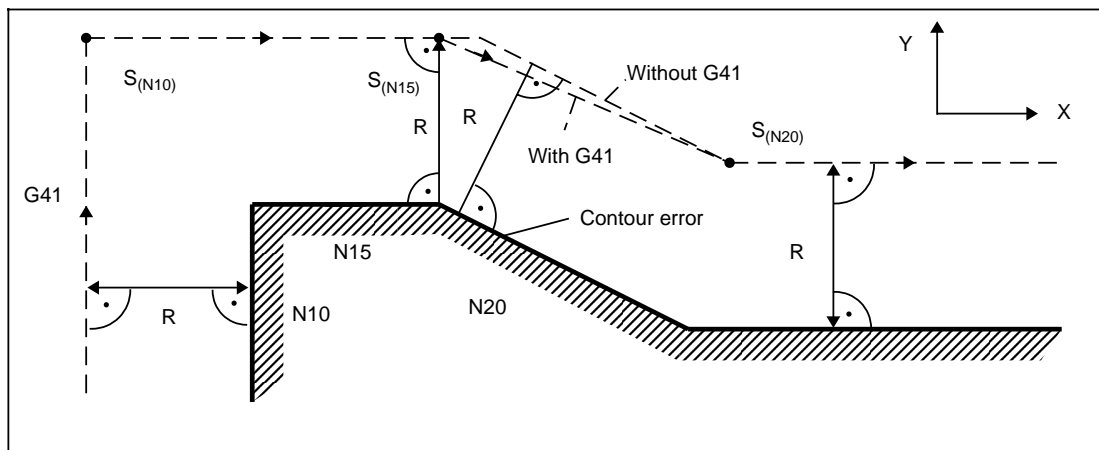
## 9.7 Repetition of selected G function (G41, G42) with same offset number

If a **G41** or **G42** function which has already been programmed is **repeated**, a **vector** of length **R** and perpendicular to the programmed path is created in the preceding block at the **block end position**.

The block start intersection S1 is **calculated** for the following block:

```
N5 G91 D10 G41 X... Y... L_F
N10 Y... L_F
N15 X... L_F
N20 G41 X... Y... L_F
N25 X... L_F
```

**Error:** G41 repeated in N20. This causes a contour error.

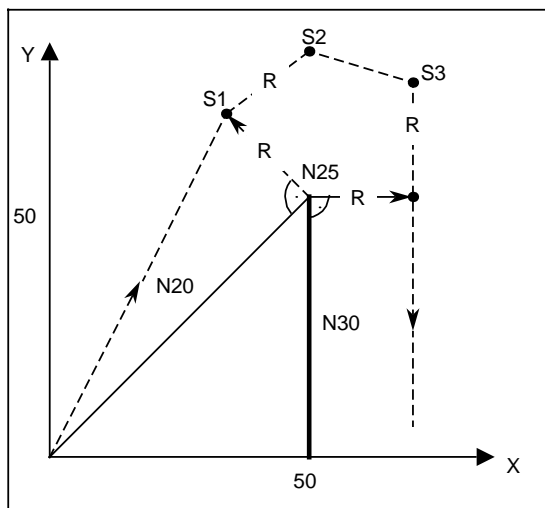


Repeated selection

## 9.8 M00, M01, M02 and M30 with CRC selected

### M00, M01

The NC **stops** at stop position **S1**, **S2** and **S3** for a **single block** (the positions are shown in the diagrams).



M00, M01 with CRC selected.

**Two cases** are to be distinguished:

Case 1: N10 G01 F100 X0 Y0 L<sub>F</sub>  
 N20 G41 X50 Y50 L<sub>F</sub>  
 N25 H111 M00 L<sub>F</sub>  
 N30 Y0 L<sub>F</sub>

H and M functions will be implemented at point S3.

Case 2: N10 G01 F100 X0 Y0 L<sub>F</sub>  
 N20 G41 X50 Y50 L<sub>F</sub>  
 N25 H111 M00 Q-5 L<sub>F</sub>  
 N30 Y0 L<sub>F</sub>

H/M functions and Q movement will be implemented at point S1.

### M02, M30

- The **compensation is traversed** one block ahead of the block in which it was cancelled via G40 and programmed via at least one axis **address** (N150 in this example).

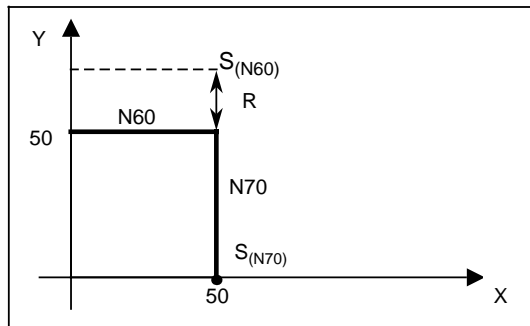
```
N150 X.. Y.. LF
N200 G40 Y.. M30 LF
```

- The compensation is **not** traversed if **no path** is programmed with G40 and if G40 is followed by **M30**.

```
N150 X.. Y.. LF
N200 G40 LF
N250 M30 LF
```



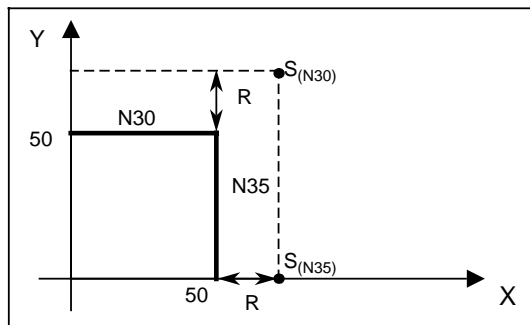
Cancellation with G40:



N60 X50 Y50 L\_F  
 N70 G40 X50 Y0 M30 L\_F

The compensation is traversed one block ahead of the block in which it was cancelled with G40 (N60 in this example).

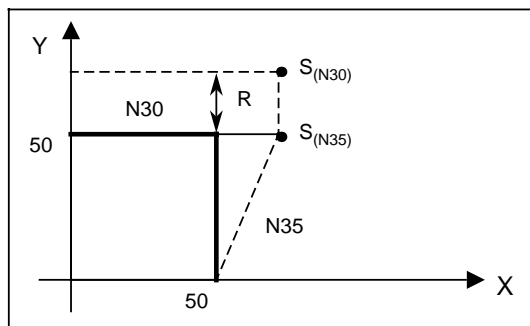
No path with G40 or programming of an axis in the last block:



N30 X50 Y50 L\_F  
 N35 Y0 L\_F  
 N40 G40 L\_F  
 N45 M30 L\_F

Compensation is not traversed.

G40 in the last-but-one block or programming of two axes in the last block:



N30 X50 Y50 L\_F  
 N35 G40 Y0 L\_F  
 N40 M30 L\_F

Compensation is traversed in N35.

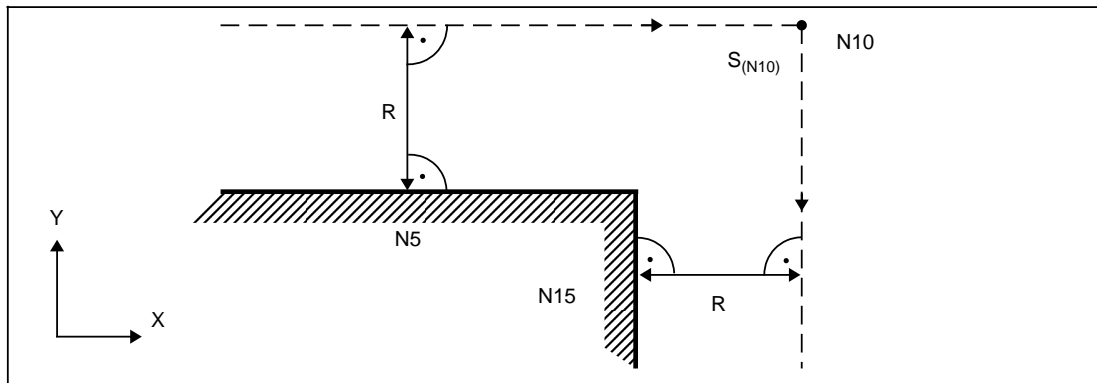
## 9.9 CRC with combination of various block types and in conjunction with contour errors

When programming in **compensation mode**, special attention must be paid to the blocks **without tool movements** in order to prevent **contour errors**. There are 3 different types of blocks:

- Block with distance 0:  
**Path addresses** are programmed, but there is no movement since the **distance is 0**.  
N5 G91 X0 L\_F
- Block "without path":  
Auxiliary functions or dwells are programmed in the compensation plane **instead of path addresses**.  
N5 M05 L\_F  
N10 G04 Y100 L\_F
- Blocks "Not in compensation plane":  
Axis addresses outside the compensation plane have been programmed.

### Examples:

- One** "auxiliary function block" between movements in the compensation plane.



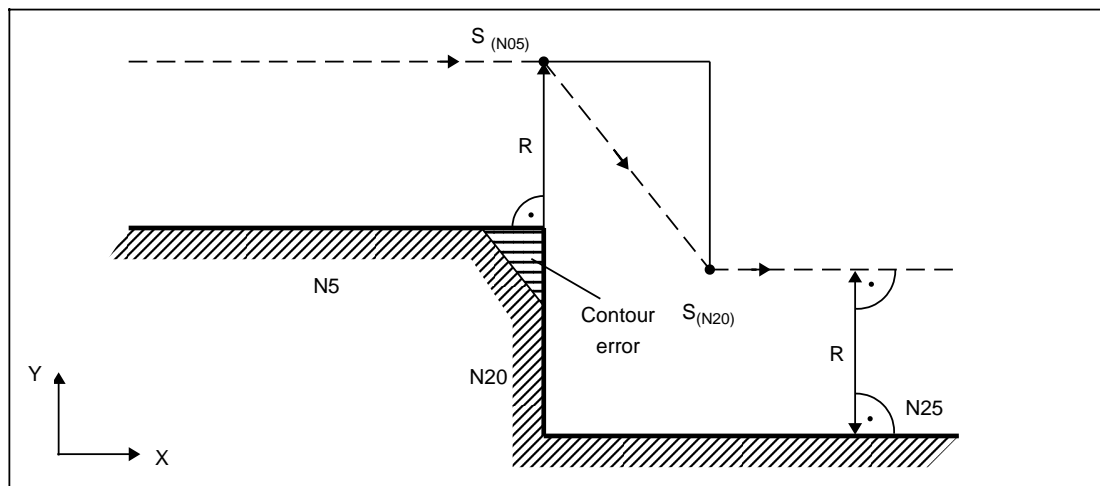
CRC: One „auxiliary function block“ between two motion blocks

```
N5 G91 X100 L_F
N10 M08 L_F
N15 Y-100 L_F
```

Block **N10** is executed at point **S<sub>(N10)</sub>**.



- **Two** "auxiliary function blocks" between movements in the compensation plane.



CRC: Two "auxiliary function blocks" between two motion blocks

```

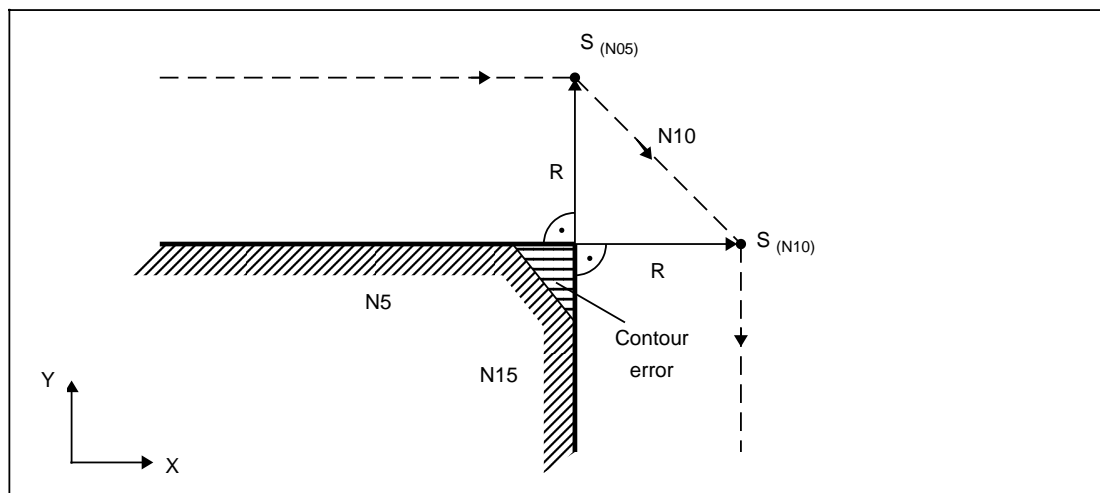
N5   G91  X100  L_F
N10  M08  L_F
N15  M09  L_F
N20  Y-100 L_F
N25  X100  L_F

```

Blocks **N10** and **N15** are executed at point **S (N5)**.

**Contour error** (hatched range)

- **One block** for "movement = 0" between two motion blocks.



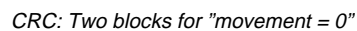
CRC: One block for „movement = 0“

```

N5   G91  X100  L_F
N10  X0    L_F
N15  Y-100 L_F

```

**Contour error** (hatched range)

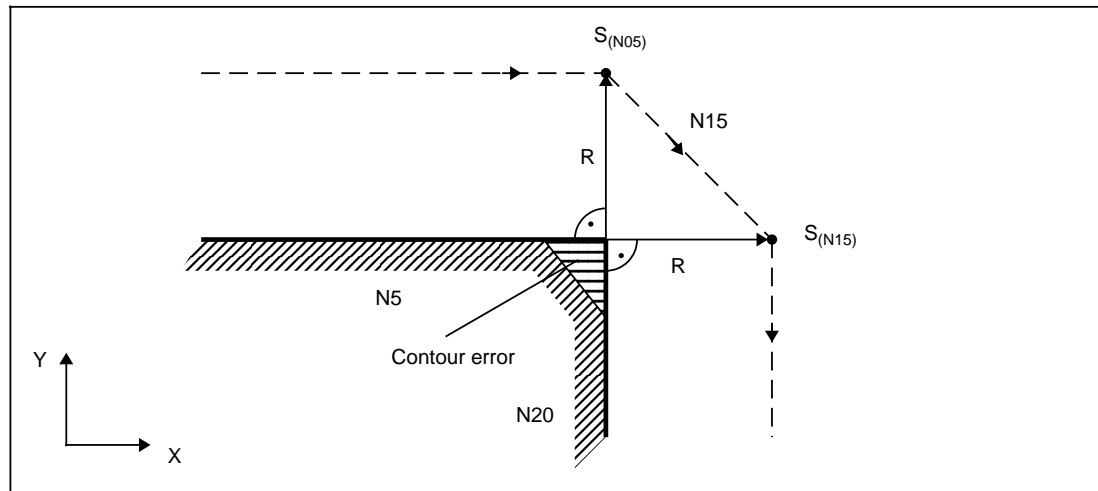


**Contour error** (hatched range)

- CRC: One block for “movement = 0” and one “auxiliary function block”*

Block **N15** is executed at point **S<sub>(N10)</sub>**.  
**Contour error** (hatched range)

- **One** “auxiliary function block” **and one block** for “movement = 0” between movements in the compensation plane.



CRC: One “auxiliary function block” and one block for “movement = 0”

```

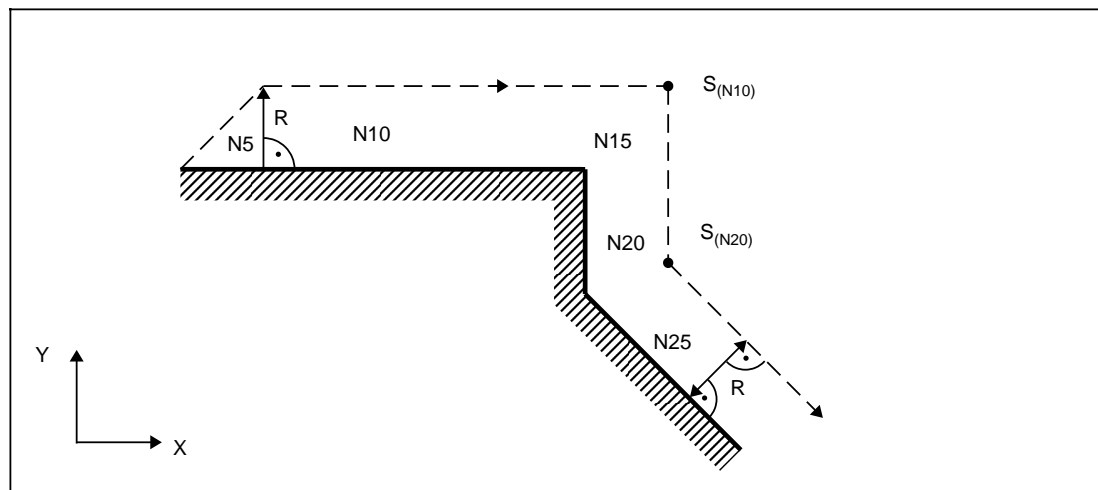
N5   G91  X100  L_F
N10  M08   L_F
N15  X0    L_F
N20  Y-100 L_F

```

Block **N10** is executed at point **S (N10)**.

**Contour error** (hatched range)

- **One** block „Not in compensation plane” between movements in the compensation plane.



CRC: One block „Not in compensation plane”

```

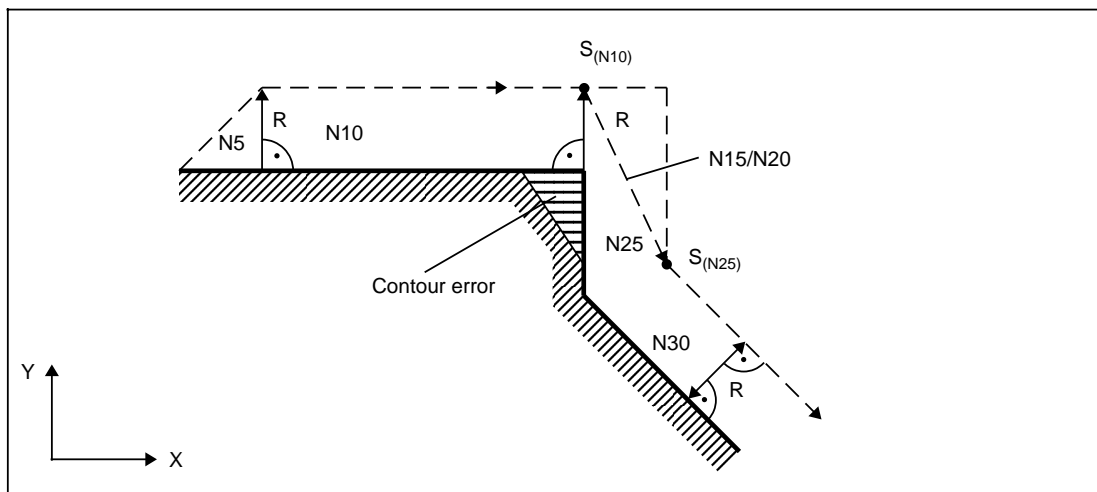
N5   G91  G41  G01  G17  X100 F100 D01  L_F
N10  X500  L_F
N15  Q500  L_F1)
N20  Y-500 L_F
N25  X1000 Y-600 L_F

```

1) Block not in compensation plane

**No violation of contour**

- **Two** blocks „Not in compensation plane” between movements in the compensation plane.



CRC: Two blocks „Not in compensation plane”

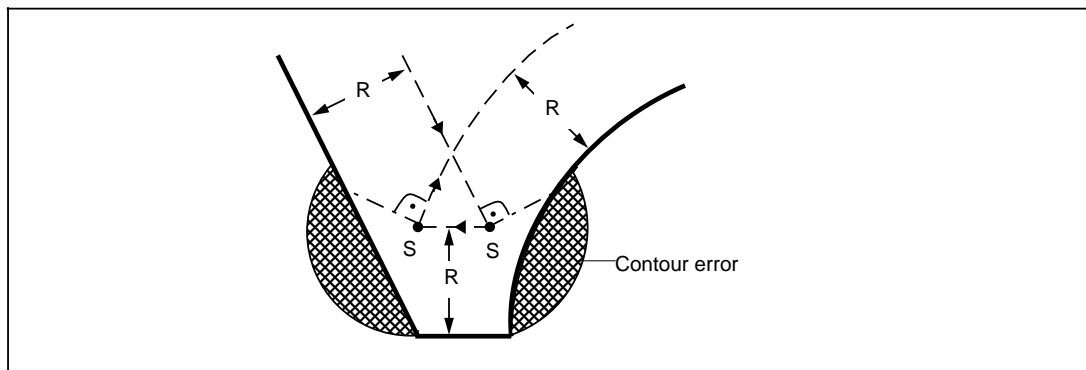
N5	G91	G41	G01	G17	X100	F100	D01	L <sub>F</sub>	
N10	X500	L <sub>F</sub>							
N15	Q500	L <sub>F</sub>							Block not in compensation plane
									} contour violation (hatched range)
N20	Q-500	L <sub>F</sub>						Block not in compensation plane	
N25	Y-500	L <sub>F</sub>							
N30	X1000	Y-600	L <sub>F</sub>						

**Contour error** (hatched range)

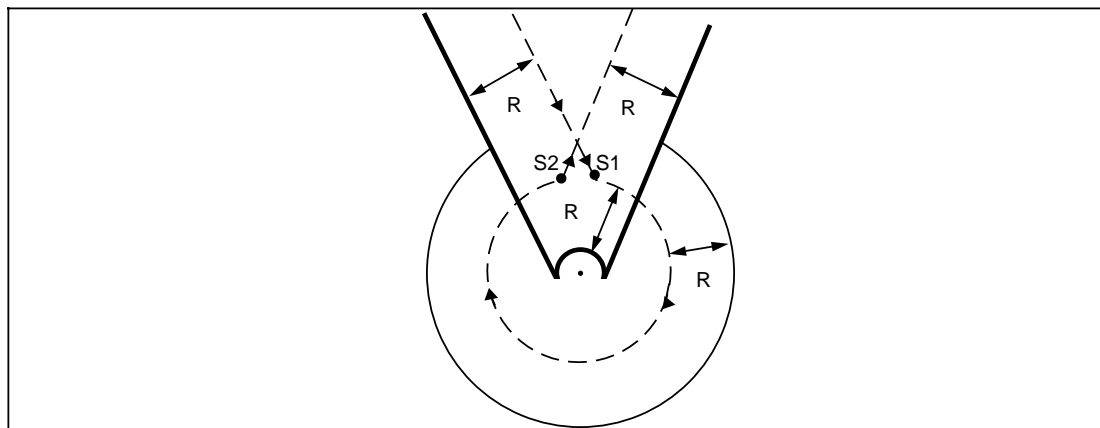
## 9.10 Special cases for CRC

The controller always uses the next block to **calculate the point of intersection of the compensated paths**. If no axes in the compensation plane are programmed in the next block, the controller uses the **next block but one**. **Contour errors** may occur if the **intermediate block** is **smaller** than the selected compensation value.

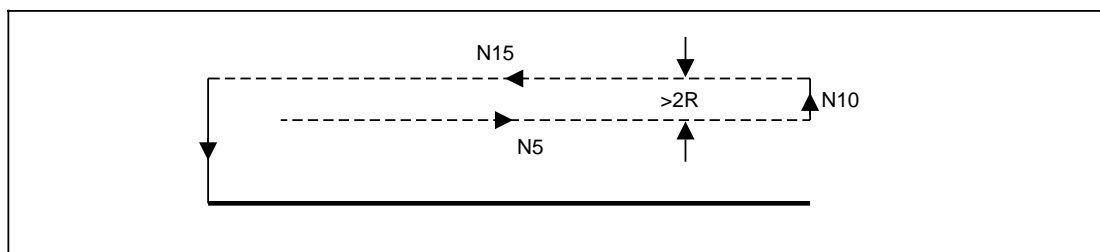
Machining is not interrupted, but an **alarm is indicated**.



CRC special case: Intermediate block < compensation value



*Special case of TNRC: Intermediate block too small for compensation*

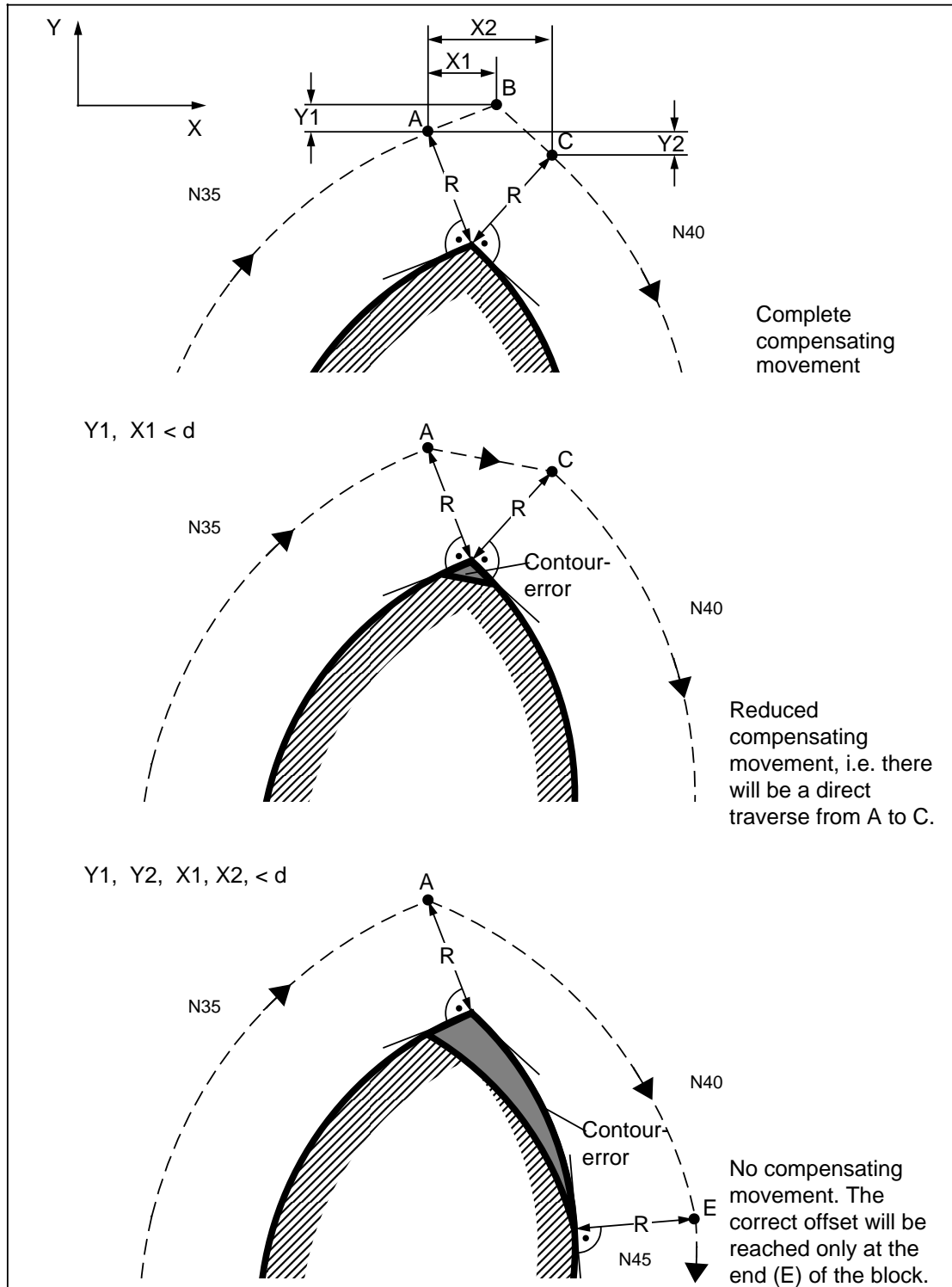


*CRC special case: Direction of compensation retained and traversing direction reversed*

The **direction of compensation** for CRC is retained and the **traversing direction** is reversed.  
 The return path in N10 must **exceed twice** the cutter radius, or the **tool** will move in the **wrong direction**.



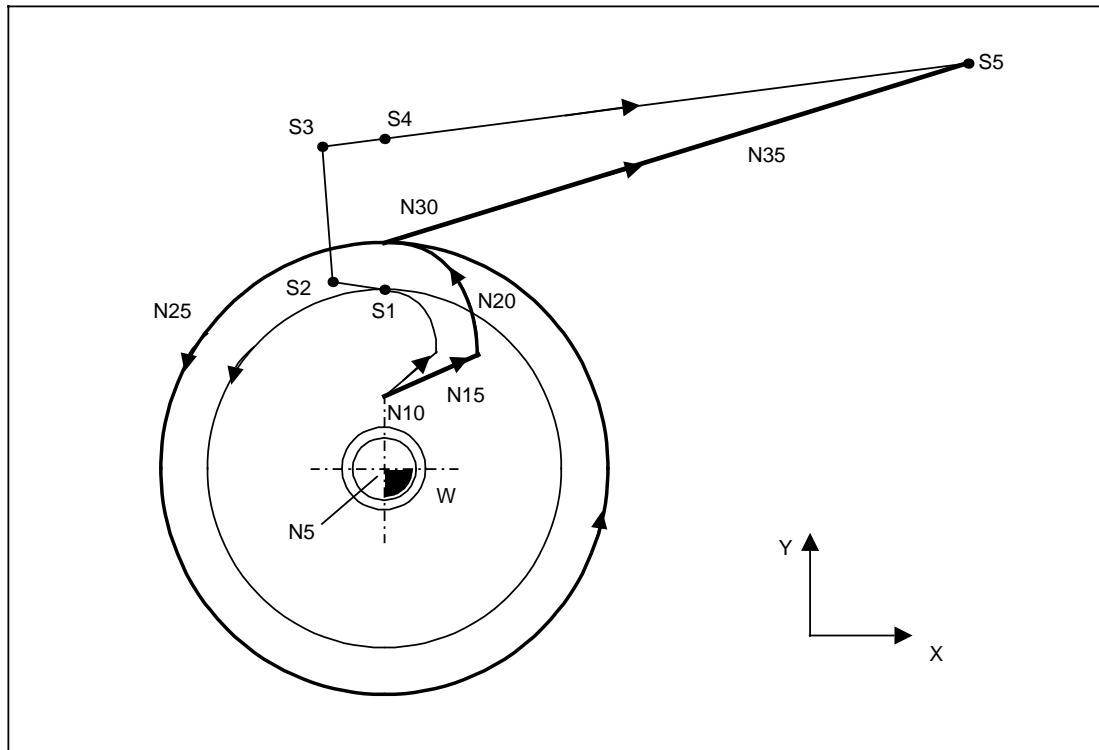
The following applies to external contours with circle transitions and obtuse angles:



In order to prevent a conditional stop in continuous path mode due to intermediate blocks which are too small, paths AB and BC can be omitted by the NC. The precise path depends on tolerance  $d$  defined on start-up (max. 32000  $\mu\text{m}$ ).

The block numbers are exchanged if the CRC generates intermediate blocks (also in case of selection/cancellation) and if an axis motion outside the compensation plane has been programmed in between.

The control generates three intermediate blocks between points S1-S2-S3-S4; only when the cutter (S1) is lifted they are traversed.



```

N5   G00  G90  Z100  L_F
N10  X0   Y10  L_F
N15  G41  D01  X20   Y20  L_F
N20  G03  X0   Y40  I-20 J0  F300  L_F
N25  X0   Y40  I0   J-40  L_F
N30  G0   Z160  L_F
N35  G40  X80  Y60  M5   L_F
N40  M30  L_F

```

Points S1, S2, S3, S4 are the auxiliary points in the CRC plane between N25 and N35. The processing sequence (as shown by single block) is as follows:

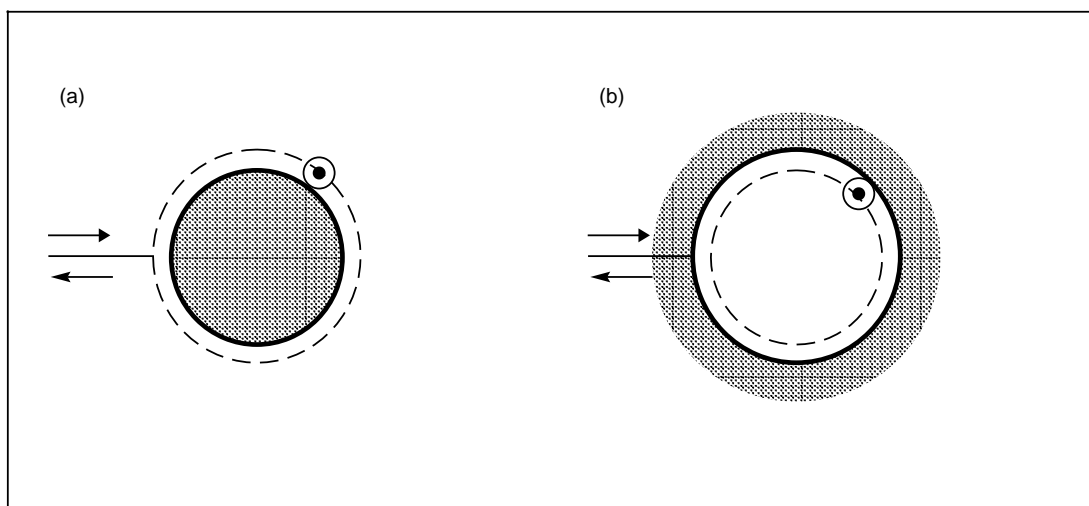
..., N20, N25 (S1), N30 (removal of tool from workpiece, retraction), N25 (S2), N25 (S3), N25 (S4), N35... . The same procedure applies with N25 being a linear block.

## 9.11 Effect with negative compensation values

If the compensation value is negative (cutter radius P4), a compensated path corresponding to G42 with a positive compensation value is implemented with G41, i.e. an analog internal contour is followed instead of the programmed external contour, and vice versa.

For the cutter centre path (  $\text{---}\bullet\text{---}$  ) shown in the diagram (a) below a positive compensation value has been entered.

A negative compensation value in conjunction with the same machining program will effect the machining shown in diagram (b).



If the program is generated as shown in b) with a positive compensation value, a negative compensation value effects machining as shown in diagram (a). The two operations are distinguished by entering a positive or negative compensation value.

## 10 Cycles

The following machining cycles are available as fixed subroutines stored in the user memory submodule (UMS):

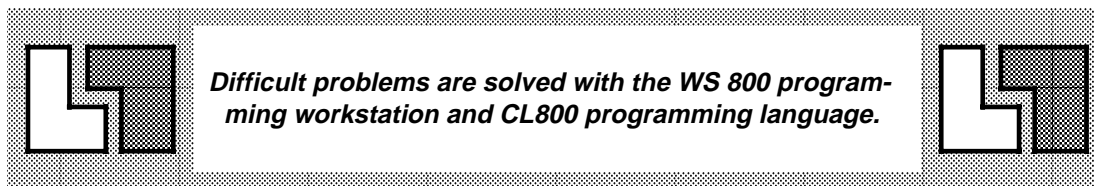
- Drilling/boring cycles G81 to G89 (L81 to L89)  
Note:  
While cycle inhibit is active, the program blocks between activation and deactivation of a drilling cycle are not displayed in the "Actual block" menu.
- Retract cycles for tool change L91/L92
- Drilling/boring and milling patterns (prerequisite: polar coordinate programming)
- L900 Drilling/boring patterns
- L901 "SLOT" milling pattern
- L902 "ELONGATED HOLE" milling pattern
- L903 Milling rectangular pocket
- L930 Milling circular pocket

These cycles are described in a separate publication "Programming Guide, SINUMERIK System 800, Cycles".

# 11 Programming of Cycles

## 11.1 General

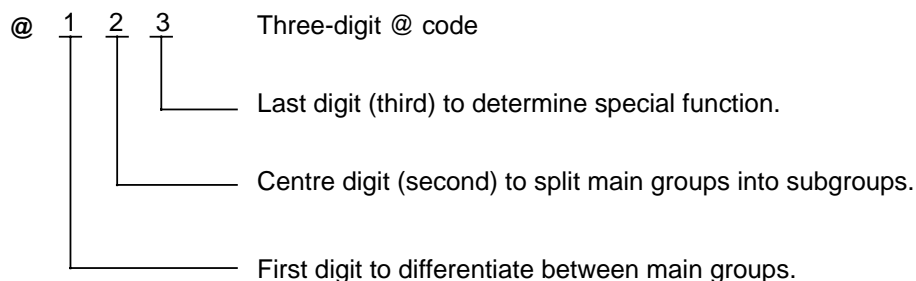
Using the @ codes, **cycles** etc. can be generated directly on the control.



The sections below are designed to provide you with an introduction to **programming** with the @ code.

## 11.2 Destination code

The destination code (@ code) consists of **three digits**, which can be interpreted as follows:



### 11.2.1 Main groups

The @ code is broken down into the following **eight groups**:

- @0 . . General statements for program structure
- @1 . . Program branchings
- @2 . . Data transfer, general
- @3 . . Data transfer, system memory to R parameters
- @4 . . Data transfer, R parameters to system memory
- @6 . . Mathematical functions
- @7 . . NC-specific functions
- @a . . I/O functions

## 11.2.2 Operands after the destination code

The individual @ commands require more detailed information (operands) for their function. These operands are defined by means of the following **letters**:

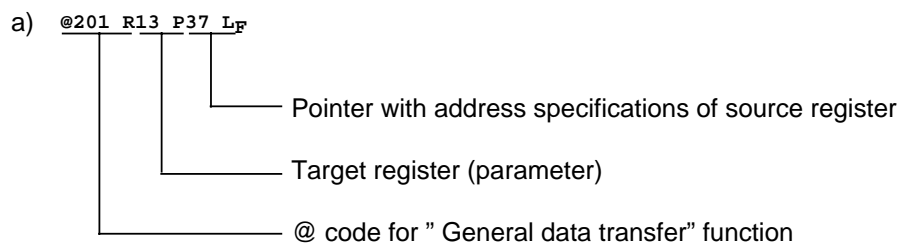
- C** Constant
- R** R parameter
- P** Pointer

The value defined with constant C is fixed in the program and cannot be changed (direct value assignment).

The value in an R parameter can be modified by the program (indirect value assignment).

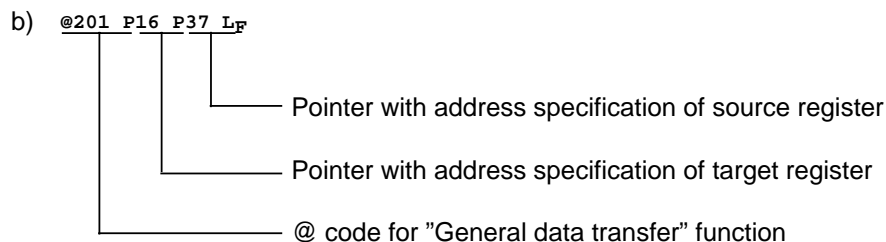
The pointer points to a parameter containing the address of the parameter the contents of which are to be used for the function (indirect value assignment).

**Examples** for @ code with operands:



Explanation of function in Example a):

Load the contents of the source register, the address of which is in register R37, into target register R13.



Explanation of function in Example b):

Load the contents of the source register, the address of which is in register R37 into the target register, the address of which is in register R16.

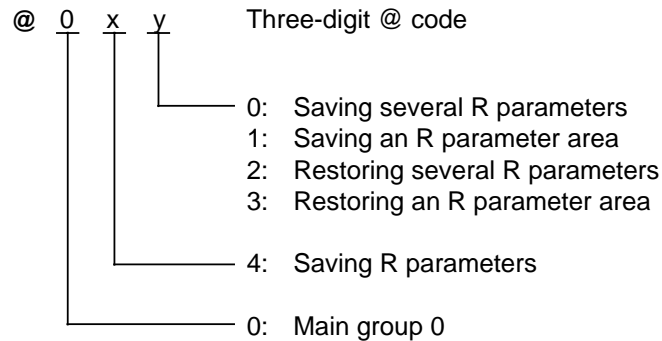
## 11.2.3 Notation

The @ code requires a strict notation. In the list of the individual commands below, the three-digit @ command is followed by a series of notations each in angle brackets. The individual notations have the following significance:

- <Const>** Direct value specification (constant K)
- <R par>** Indirect value specification (R parameter)
- <Var>** Indirect value specification (R parameter or pointer)
- <Value>** Mixed value specification (constant, R parameter or pointer)

## 11.3 General statements for program structure

Main group 0 is broken down as follows:



Main group 0/subgroup 0: **General program statements**

### @00f

#### (Enable for softkey start)

If this is the only command programmed in the first block of a subroutine, this means that the subroutine is enabled for start via softkey. The command should be at the beginning of the program to save time for searching. Refer to the machine tool manufacturer for information on where the softkey is configured.

Main group 0 / Subgroup 4: **Saving of R parameters**

### @040 <Const> <R par 1> ... <R par n>

#### Saving the R parameters onto stack

The constant <Const> indicates the number of the subsequent R parameters belonging to this function. The contents of the R parameters are saved in a stack register from R300 onwards.

### @041 <R par 1> ... <R par 2>

#### Saving a group of R parameters onto stack

The contents of the R parameters from <R par 1> to <R par 2> are transferred to the stack register from R300 onwards.

### @042 <Const> <R par n> ... <R par 1>

#### Fetch R parameters from stack

This command takes the saved values from the stack register and loads them into the stated R parameters. The R parameters must be listed in reverse sequence as compared with @040.

### @043 <R par 1> ... <R par 2>

#### Fetch a group of R parameters from stack

The values saved with @041 are reloaded into the R parameters.

These commands of the main group 0 and subgroup 4 are used when working in a subroutine with R parameters which may already have been used at a higher level.

A push command (@040 or @041) is written at the start of the subroutine, being used to save the values and to occupy the specified R parameters with the value 0.

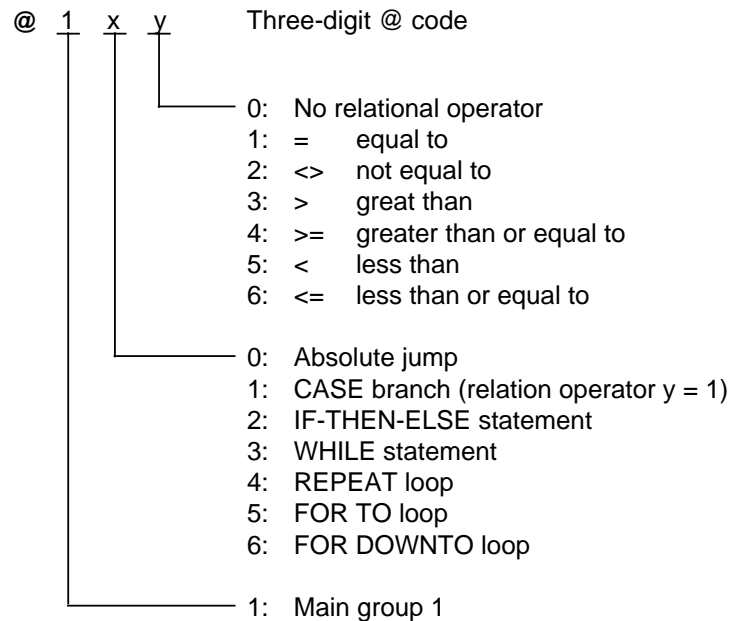
At the end of the subroutine a pop command (@042 or @043) is used to restore the original status.

#### Example of notation in program:

L100 LF	<b>Call subroutine:</b>
@041 R61 R69 LF	The contents of the R parameters R61 to R69 are transferred to the stack register and set to "0".
:	
:	
@043 R61 R69 LF	The saved values are reloaded into parameters R61 to R69.
:	
:	
M17 LF	<b>Subroutine end</b>

## 11.4 Program branchings

Main group 1 is broken down as follows:





Main group 1 / Subgroup 0: **Absolute jump**

**@100 <Const> or @100 <R par>**

## Absolute jump

The constant indicates the jump destination (block number) and jump direction. A positive block number means that the block to which the jump is to be performed is located towards the end of the program, while a negative block number means that it is near the start of the program.

If the sign points in the wrong direction, the control does not find the block, even if it is in the program (Alarm 3012: "block not in memory").

### Examples:

@100 K375 L<sub>F</sub>

Absolute jump to block N375 towards end of program

@100 K-150 L<sub>F</sub>

Absolute jump to block N150 towards start of program.

Main group 1 / Subgroup 1: **CASE** branch

```
@111  <Var>    <Value 1><Const 1>
          <Value 2> <Const 2>
          :      :
          :      :
          <Value n> <Const n>
```

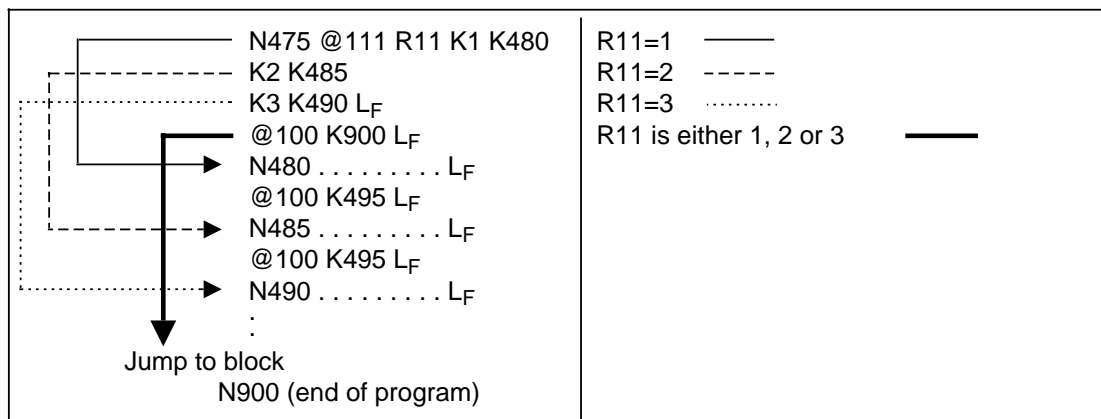
## CASE branch

The numerical value defined with the notation <Var> is compared consecutively with those of notations <Value... >.

If they tally, a jump is performed to the block determined with the corresponding constant <Const...>.

If the comparison is not fulfilled, the program is continued with the next block in which the block number for continuing the program is programmed (e.g. with an absolute jump with @100).

**Example of CASE branch:**



*CASE branch used in a cycle with axis switching*

### Explanations regarding the above example:

The diagram shows how a CASE branch is used in a cycle with axis switching. Depending on whether the programmer has defined the machining plane by parameterizing R11=1, 2 or 3, the program branches to the blocks N480 (R11=1), N485 (R11=2) or N490 (R11=3). If there is none of those three values in R11 there obviously exists a parameterizing error. The program branches to block N900 (=end of program).

Main group 1 / Subgroup 2: **IF-THEN-ELSE branch**

An IF-THEN-ELSE branch means:

If (IF) the condition (3rd digit of the @ code) is fulfilled, then (THEN) execute the statements in the next blocks. Otherwise (ELSE) a jump is made to the block whose number was determined with the last constant.

Here too, the sign of the block number determines the search direction.

#### @121 <Var> <Value> <Const>

#### Equal to

If (IF) the numerical value defined with notation <Var> is equal to the value defined with <Value>, then <THEN> the program is continued with the next block. Otherwise <ELSE> a jump is made to the block determined with the constant.

#### Example:

@121 R13 R27 K375 L<sub>F</sub> Continuation of program if R13=R27; otherwise conditional jump to block N375 in direction towards program end.

#### @122 <Var> <Value> <Const>

#### Not equal to

If (IF) the numerical value defined with notation <Var> is not equal to the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

#### @123 <Var> <Value> <Const>

#### ">"

If (IF) the numerical value defined with notation <Var> is greater than the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

#### Example:

@123 R13 R27 K-150 L<sub>F</sub> Continuation of program if R13>R27; otherwise conditional jump to block N150 in direction towards program start.

#### @124 <Var> <Value> <Const>

#### ">="

If (IF) the numerical value defined with notation <Var> is greater than or equal to the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

#### @125 <Var> <Value> <Const>

#### "<"

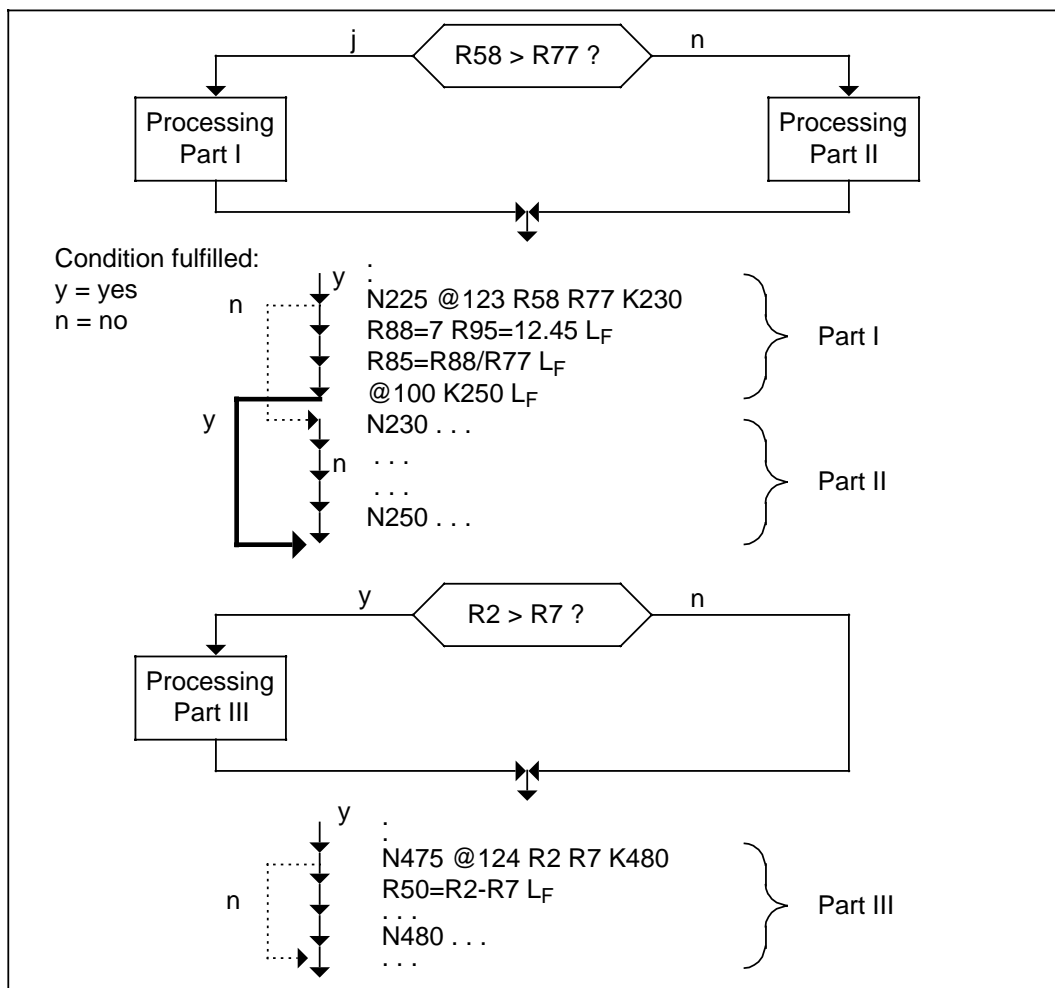
If (IF) the numerical value defined with notation <Var> is less than the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

#### @126 <Var> <Value> <Const>

#### "<="

If (IF) the numerical value defined with notation <Var> is less than or equal to the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

**Example:** Program run with IF-THEN-ELSE branches



#### Explanations regarding the above example:

The shown program section from a cycle illustrates how you can create program branches by using IF-THEN-ELSE branches. If the contents of the register R58 in block N225 is greater than the contents of the register R77 the statements in the next line are executed.

The register R88 is loaded with 7 and the register R95 with 12.45. If, however, R58 is less than or equal to R77 the program branches to block N230. But in the line preceding block N230 there is an absolute jump to block N250.

The IF-THEN-ELSE branch in block N225 thus either causes block N225 or the program section from block N230 to N250 to be processed.

In the second example the unconditioned jump does not exist, so that the statements in block N475 are obeyed or not. This program section can thus be skipped by means of the IF-THEN-ELSE branch @124.

### Main group 1 / Subgroup 3: **WHILE loop**

#### @13y <Var> <Value> <Const>

The WHILE loop is a repeat statement with sampling of the repeat conditions at the start of the loop. The relation operators correspond to those of the IF-THEN-ELSE branch.

Provided the comparison is fulfilled, the next block is processed.. At the end of the block an absolute jump must be programmed with @100 <Const> which leads back to sampling.

If the comparison is not fulfilled, a jump is made to the block defined under <Const> and which is generally located after the block with the absolute jump.

The setting of y shall be in accordance with commands @121...@126.

#### Examples:

N300 @131 R13 R27 K375 L\_F      Continuation of loop as long as loop condition  
R13=R27 is fulfilled.

:

@100 K-300 L\_F

N375 . . L\_F

N300 @133 R13 R27 K375 L\_F      Continuation of loop as long as loop condition  
R13>R27 is fulfilled.

:

@100 K-300 L\_F

N375 . . L\_F

:

### Main group 1 / Subgroup 4: **REPEAT loop**

#### @14y <Var> <Value> <Const>

The REPEAT loop is a repeat statement with sampling of the repeat conditions at the start of the loop. The relation operators correspond to those of the IF-THEN-ELSE branch. If the comparison is not fulfilled, a jump is made back to the block defined under <Const>. If the condition is fulfilled, the loop is exited and the program is resumed.

The setting of y shall be in accordance with commands @ 121 ...@ 126.

#### Examples:

N400 . . L\_F      Repeat following instructions until condition  
R13=R27 is fulfilled.

:

@141 R13 R27 K-400 L\_F

N400 . . L\_F      Repeat following instructions until condition  
R13>R27 is fulfilled.

:

@143 R13 R27 K-400 L\_F

:

Main group 1 / Subgroup 5: **FOR TO loop****@151 <Var> <Value 2> <Const>**

The FOR TO loop is a counting loop in which the contents of the R parameter defined under <Var> are incremented with each pass. Sampling for "equal to" is performed at the beginning of the loop. In the event of inequality, the loop is processed, otherwise a jump is made to the block defined under <Const>. At the end of the loop the variable <Var> must be incremented (@620) with an absolute jump back to the start of the loop.

**Example:**

R50=1 R51=5 R52=10 L <sub>F</sub>	Value assignment for R5, R51, R52
@201 R50 R51 L <sub>F</sub>	Data transfer from R5 to R50
N500 @151 R50 R52 K505 L <sub>F</sub>	Beginning of FOR-TO loop
:	
:	
:	
@620 R50 L <sub>F</sub>	
@100 K-500 L <sub>F</sub>	
N505	

Main group 1 / Subgroup 6: **FOR DOWNT0 loop****@161 <Var> <Value 2> <Const>**

The FOR DOWNT0 loop is a counting loop in which the contents of the R parameter defined under <Var> are decremented with each pass. Sampling for "equal to" is performed at the beginning of the loop. In the event of inequality, the loop is processed, otherwise a jump is made to the block defined under <Const>.

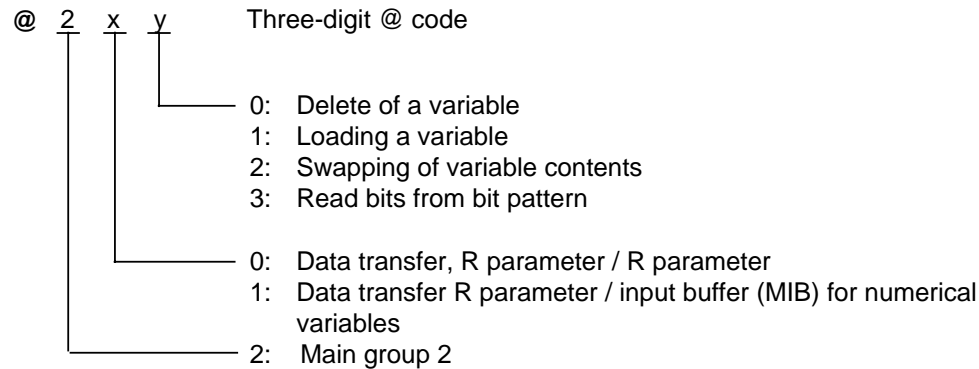
At the end of the loop the variable <Var> must be decremented (@621) with an absolute jump back to the start of the loop.

**Example:**

R50=10 R51=5 R52=1 L <sub>F</sub>	Value assignment for parameters R5, R51, R52
@201 R50 R51 L <sub>F</sub>	Data transfer from R5 to R50
N600 @161 R50 R52 K605 L <sub>F</sub>	Beginning of FOR DOWNT0 loop
:	
:	
:	
@621 R50 L <sub>F</sub>	
@100 K-600 L <sub>F</sub>	
N605	

## 11.5 Data transfer, general

Main group 2 is broken down as follows:



Main group 2 / Subgroup 0: **Data transfer R parameter/R parameter**

**@200 <Var> Delete variable**  
The value of the R parameter defined with the notation <Var> is deleted.

**@201 <Var> <Value> Load variable with value**  
The numerical value defined under <Value> is transferred to the R parameter defined under <Var>.

**@202 <Var 1> <Var 2> Swap the contents of the variables**  
The contents of the two R parameters defined under <Var 1> and <Var 2> are swapped.

**@203 <Var 1> <Var 2> <Const> Read a bit from a bit pattern**  
The bit of the bit pattern contained in <Var 2> and defined by <Const> is read into <Var 1>.

Main group 2 / Subgroup 1: **Data transfer R parameter/machine input buffer (MIB) for numerical variables**

**@210 <Value 3> <Value 4> Delete input buffer**  
Delete input buffer.  
<Value 3> Starting address  
<Value 4> End address

**@211 <Var> <Value 1> Read input buffer**  
Der R parameter <Var> is loaded with the content of the input buffer cell <Value 1>.

### Example:

@211 R50 K101 LF      The value 5 is written in the input buffer cell 101.

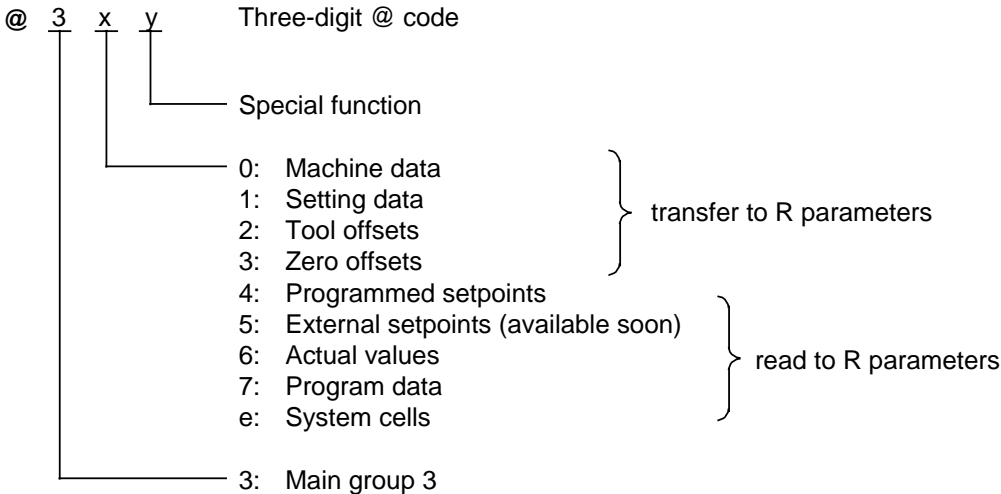
**@212 <Value 1> <Value>**  
The input buffer cell <value 1> is loaded with the numeric quantity <value>.

**Write machine input buffer**

**Example:**  
`@212 K102 K5 LF`                      The value 5 is written in the input buffer cell 102.

11.6 Data transfer, system memory to R parameters

Main group 3 is broken down as follows:



All @ commands in this main group feature <Var> as their first notation. Consequently, an R parameter is defined, directly or via a pointer, to which the contents of the system memory cell addressed are to be transferred.

Main group 3 / Subgroup 0: **Transfer machine data to R parameters**

**@300 <Var> <Value 1>**  
The address of NC machine data is defined under <Value 1>.  
Address range: 0 to 4999.

**Machine data NC**

**Example:**  
`@300 R50 K2240 LF`                      Parameter R50 contains the value of the 1st software limit switch in plus direction for the 1st axis.

**@301 <Var> <Value 1>**  
The byte address of NC machine data is defined under <Value 1>.  
Address range: 5000 to 6999.

**Machine data NC byte**

**@302 <Var> <Value 1> <Value 2>****Machine data NC bit**

The byte address of an NC machine data bit is defined under <Value 1>.

Address range: 5000 to 6999. The bit address (0 to 7) is under <Value 2>.

**@306 <Var> <Value 1>****Machine data PLC**

The byte address of PLC machine data is defined under <Value 1>.

Address range: 0 to 1999.

**@307 <Var> <Value 1>****Machine data PLC byte**

The byte address of PLC machine data is defined under <Value 1>.

Address range: 2000 to 3999.

**@308 <Var> <Value 1> <Value 2>****Machine data PLC bit**

The byte address of PLC machine data bit is defined under <Value 1>.

Address range: 2000 to 3999. The bit address (0 to 7) is under <Value 2>.

Main group 3 / Subgroup 1: **Transfer setting data to R parameters**

**@310 <Var> <Value 1>****Setting data NC**

The address of a setting data is defined under <Value 1>.

Address range: 0 to 4999.

**@311 <Var> <Value 1>****Setting data NC byte**

The byte address of a setting data is defined under <Value 1>.

Address range: 5000 to 9999.

**@312 <Var> <Value 1> <Value 2>****Setting data NC bit**

The byte address of a setting data bit is defined under <Value 1>.

Address range: 5000 to 9999. The bit address (0 to 7) is under <Value 2>.

Main group 3 / Subgroup 2: **Transfer tool offsets to R parameters**

**@320 <Var> <Value 1> <Value 2> <Value 3>****Tool offset**

With this command the individual compensation values can be read out of the tool offset memory into the parameter under the notation <Var>.

The notations <Value 1> to <Value 3> should be as follows:

<Value 1> TO range

Range: 0

<Value 2> Tool offset number (D number)

Range: 1 to 99

<Value 3> Number of tool offset memory (P number)

Range: 0 to 9

**Example:**

@320 R67 K0 K14 K2 L\_F

Read the offset value P2 (geometry, length 1) of tool offset number D14 for TO range 0 into parameter R67.



Main group 3 / Subgroup 3:      **Transfer zero offsets to R parameters**

**@330 <Var> <Value 1> <Value 2> <Value 3>      Settable zero offset**

The notations <Value 1> to <Value 3> should be as follows:

- <Value 1>    Group of settable zero offsets  
                  (G54 = 1 to G57 = 4)
- <Value 2>    Axis number
- <Value 3>    Coarse or fine value (0 or 1)

**Example:**

@330 R81 K1 K2 K0 L\_F

Parameter R81 is loaded with the coarse value of the 1st settable zero offset (G54) of the 2nd axis.

**@331 <Var> <Value 1> <Value 2>      Programmable zero offset**

Significance:

- <Value 1>    Group of programmable additive zero offsets  
                  (G58=1 to G59=2)
- <Value 2>    Axis number

**@332 <Var> <Value 2>      External zero offset from PLC**

<Value 2> defines the number of the axis of the external zero offset entered from the PLC.

**@333 <Var> <Value 2>      DRF offset**

<Value 2> defines the number of the axis of the DRF offset.

**@334 <Var> <Value 2>      PRESET offset**

<Value 2> defines the number of the axis of the PRESET offset.

**@336 <Var> <Value 2>      Total offset**

<Value 2> defines the number of the axis of the total offset. The total offset comprises:

- the selected settable zero offset
- the programmable zero offset
- the external zero offset
- the selected tool offset

PRESET and DRF offsets are not taken into account.

**@337 <Var> <Value 1> <Value 2> <Value 3>      Settable coordinate rotation**

Parameter <Var> may be loaded with the angle of rotation of the settable coordinate rotation.

The notations <Value 1> to <Value 3> should be as follows:

- <Value 1>    Number of channel (0 = own channel)
- <Value 2>    Group of settable coordinate rotations (G54=1 to G57=4)
- <Value 3>    Number of angle (currently = 1).

**@338 <Var> <Value 1> <Value 2> <Value 3>****Programmable coordinate rotation**

The angle of rotation of the programmable coordinate rotation can be loaded into the parameter <Var>.

Significance:

- <Value 1> Number of channel (0 = own channel)
- <Value 2> Group of programmable coordinate rotations  
(G58=1 and G59=2)
- <Value 3> Number of angle (currently = 1).

Main group 3 / Subgroup 4: **Read programmed setpoints into R parameter**

**@342 <Var> <Value 1> <Value 3>****Programmed spindle speed**

Parameter <Var> is loaded with the programmed spindle speed from one channel.

Notations will be set as follows:

- <Value 1> Number of channel (0= own channel)
- <Value 3> Spindle number (0= number of leadscrew)

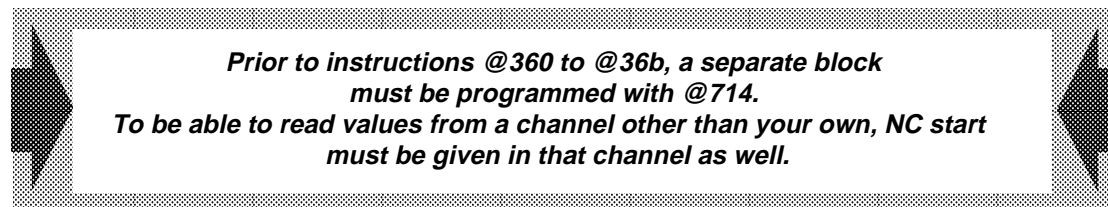
**@345 <Var> <Value 1> <Value 2>****Programmed cutting rate**

With this command, the cutting speed programmed under G96 may be transferred into an R parameter.

The channel number is entered under <Value 1> (own channel=„0“).

<Value 2> is set to „0“.

Main group 3 / Subgroup 6: **Read actual values to R parameters**

**@360 <Var> <Value 2>****Actual axis position workpiece-related**

<Value 2> defines the axis whose workpiece-related actual value is to be transferred to the R parameter.

**Example:**

@360 R54 K2 L\_F

.Read actual position value of the 2nd axis related to workpiece zero and enter it into register R54.

**@361 <Var> <Value 2>****Actual axis position machine-related**

<Value 2> defines the axis whose machine-related actual value is to be transferred to the R parameter.

*In simulation, values output by @360 and @361 are identical.*

**@363 <Var> <Value 2>****Actual spindle position**

<Value 2> defines the spindle whose actual position is to be transferred to the R parameter.

**@364 <Var> <Value 2>****Actual spindle speed**

<Value 2> defines the spindle whose actual speed is to be transferred to the R parameter.

**@367 <Var> <Value 1>****Axis number of current plane/leadscrew number**

<Var> is to define the R parameter from which onwards the axis number of the plane selected with G16 (4 values) shall be entered. The number of the leadscrew is stored in the 5th R parameter. The channel number is defined under <Value 1> (0= own channel).

The following R parameters are loaded:

Rn	Number of the horizontal axis
Rn+1	Number of the vertical axis
Rn+2	Number of the axis vertical to the plane
Rn+3	Number of the axis in which length 2 acts (tool type 30 on the M version)
Rn+4	Number of the leading spindle

If a tool length compensation in negative direction is programmed with G16 XYZ, the value 128 is added for the Z axis by means of @367 in order for the minus sign to be recognized.

**Example:**

@367 R50 K1 L\_F

Data from the current plane and spindle number are read in channel 1 and stored from R50 onwards.

**@36a <Var> <Value 1>****Actual D function**

This command is used to transfer the number of the selected tool offset (D number) to the R parameter. The channel (own channel = 0) must be entered under <Value 1>. "0" must be entered in the case of the SINUMERIK 810M/820M.

Note: "a" in the @ code stands for hexadecimal "A" (=10).

**@36b <Var> <Value 1> <Value 3>****Actual G function**

The G function of the part program block which is being processed is read from the main memory into parameter <Var>. The channel number is defined in <Value 1>. If it is defined as 0 the same channel is read. <Value 3> defines the **internal** G group to which the current G function belongs. See the appendix for a table with the **internal** G group division (overview for @36b).

**Example:**

@36b R50 K0 K0 L\_F

The current G function of the first internal G group (G0) is read into the parameter R50 in the same channel.

Main group 3 / Subgroup 7: **Read program data to R parameters****@371 <Var> <Value 1> <Value 3>****Special bits**

This command is used to read out special bits for detecting different active signals.

**Channel-dependent bits:** Bit 0 = Block search active  
 Bit 1 = Dry run feed active  
 Bit 2 = Simulation active

The channel number (own channel = 0) should be entered in <Value 1>·<Value 3> contains the bit number.

**Channel-independent bits:** Bit 0 = Measuring input 1 active  
 Bit 1 = Measuring input 2 active

99 must be entered in <Value 1>·<Value 3> contains the bit number.

**Example:**

```
@371 R81 K0 K1 L_F
```

The state of the special bits for "Dry run feed" in the own channel is read to parameter R81.

Main group 3 / subgroup 8: **Read PLC signal bits to R parameters****@380 <Var> <Value 1> <Value 2> Value 3>**

The state of an input bit in the PLC is read into the parameter defined with <Var>. The PLC number is defined by <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 must be entered for the PLC number.

**Example:**

```
@380 R50 K1 K2 K0
```

The state of the defined input bit is read into R50.

**@381 <Var> <Value 1><Value 2><Value 3>**

The state of an output bit in the PLC is read into the parameter defined with <Var>. The PLC number is defined by <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 must be entered for the PLC number.

**@382 <Var> <Value 1><Value 2><Value 3>**

The state of a flag bit in the PLC is read into the parameter defined with <Var>. The PLC number is defined by the <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 must be entered for the PLC number.

**@383 <Var> <Value 1><Value 2><Value 3><Value 4>**

The state of an output bit is read into the parameter defined with <Var>. The PLC number is defined by <Value1>, the number of the DB or DX by <Value2>, the data word number by <Value3> and the bit address by <Value4>. K1 must be entered for the PLC number.

**Example:**

@383 R51 K1 K2 K4 K2

The state of the defined PLC data word bit is read into R51.

**Note:**

In the 3rd channel (simulation) 380 to 383 are overread. This can cause the program run to be changed.

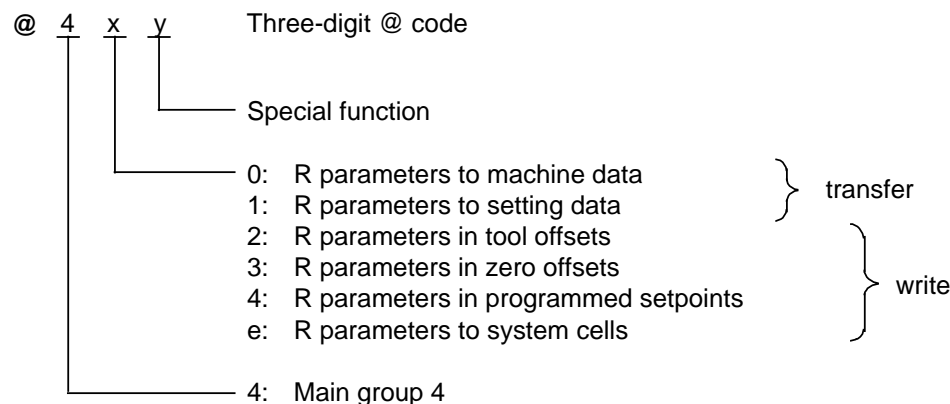
Main group 3 / Subgroup e: **Read system cells to R parameters**

**@3e4 <Var> <Value 1>****Read active gear stage**

Dependent upon spindle number <Value 1>, the active gear stage is read in the parameter <Var>. If the spindle number is 0, the number of the leadscrew will be used as spindle number.

## 11.7 Data transfer, R parameters to system memory

Main group 4 is broken down as follows:



All @ commands in this main group feature <Value> as their last notation. Consequently, the numerical value to be transferred is defined either directly using a constant or indirectly via an R parameter or a pointer.

### Main group 4 / Subgroup 0: **Transfer R parameters to machine data**

#### @400 <Value 1> <Value> **Machine data NC**

The address of NC machine data is defined under <Value 1>.

Address range: 0 to 4999.

##### Example:

@400 K2241 R90 L\_F

The machine data of the 1st software limit switch for the 2nd axis in positive direction is loaded via parameter R90.

#### @401 <Value 1> <Value> **Machine data NC byte**

The byte address of an NC machine data bit is defined under <Value 1>.

Address range: 5000 to 6999.

#### @402 <Value 1> <Value 2> <Value> **Machine data NC bit**

The byte address of an NC machine data bit is defined under <Value 1>.

Address range: 5000 to 6999. The bit address (0 to 7) is under <Value 2>.

#### @406 <Value 1> <Value> **Machine data PLC**

The address of PLC machine data is defined under <Value 1>.

Address range: 0 to 1999.

#### @407 <Value 1> <Value> **Machine data PLC byte**

The byte address of PLC machine data is defined under <Value 1>.

Address range: 2000 to 3999.

#### @408 <Value 1> <Value 2> <Value> **Machine data PLC bit**

The byte address of PLC machine data bit is defined under <Value 1>.

Address range: 2000 to 3999. The bit address (0 to 7) is under <Value 2>.

Main group 4 / Subgroup 1: **Transfer R parameters to setting data**

**@410 <Value 1> <Value>**

**Setting data NC**

The address of setting data is defined under <Value 1>.  
Address range: 0 to 4999.

**@411 <Value 1> <Value>**

**Setting data NC byte**

The byte address of setting data is defined under <Value 1>.  
Address range: 5000 to 9999.

**@412 <Value 1> <Value 2> <Value>**

**Setting data NC bit**

The byte address of setting data bit is defined under <Value 1>.  
Address range: 5000 to 9999. The bit address (0 to 7) is under <Value 2>.

Main group 4 / Subgroup 2: **Write R parameter in tool offsets**

**@420 <Value 1> <Value 2> <Value 3> <Value>**

**Tool offset**

The numerical value is entered in the tool offset memory. The contents of the memory are overwritten.

The notations <Value 1> to <Value 3> should be as follows:

<Value 1> TO range; Range: 0  
<Value 2> Tool offset number (D number)  
Range: 1 to 99  
<Value 3> Number of tool offset value (P number)  
Range: 0 to 9

**Example:**

**@420 K0 K2 K3 R80 L\_F**

In TO range 0, offset memory D2 under tool offset value P3 is loaded with the contents of R80.

**@423 <Value 1> <Value 2> <Value 3> <Value>**

**Tool offset additive**

The numerical value is added to the value in the tool offset memory.  
The notations <Value 1> to <Value 3> should be as follows:

<Value 1> TO range; Range: 0  
<Value 2> Tool offset number (D number)  
Range: 1 to 99  
<Value 3> Number of tool offset value (P number)  
Range: 0 to 9.

Main group 4 / Subgroup 3: **Write R parameters to zero offsets****@430 <Value 1> <Value 2> <Value 3> <Value>      Settable zero offset**

The numerical value is entered in the zero offset memory. The contents of the memory are overwritten.

The notations <Value 1> to <Value 3> should be as follows:

- <Value 1>    Group of settable zero offsets (G54=1 to G57=4)
- <Value 2>    Axis number
- <Value 3>    Coarse or fine value (0 or 1)

**Example:**

@430 K1 K2 K0 K500 Lp

The coarse value of the first settable zero offset is loaded into the second axis with the constant 500.

**@431 <Value 1> <Value 2> <Value 3> <Value>      Additive settable zero offset**

The numerical value is added to the value in the zero offset memory.

The notations <Value 1> to <Value 3> should be as follows:

- <Value 1>    Group of settable zero offsets (G54=1 to G57=4)
- <Value 2>    Axis number
- <Value 3>    Coarse or fine value (0 or 1).

**@432 <Value 1> <Value 2> <Value>      Programmable zero offset**

Significance:

- <Value 1>    Group of programmable additive zero offsets  
(G58=1 to G59=2)
- <Value 2>    Axis number

**@434 <Value 2> <Value>      DRF offset**

<Value 2> defines the number of the axis of the DRF offset.

**@435 <Value 2> <Value>      PRESET offset**

<Value 2> defines the number of the axis of the PRESET offset. If the @435 is used to describe the PRESET offset in the AUTOMATIC or MDA mode, the offset is only active after M2/M30/PRESET.

If "TRANSMIT" function is active, command @435 also applies to fictitious axes.

@435 is chiefly used to deliberately **reset** the PRESET offset at the end of program.

**@437 <Value 1> <Value 2> <Value 3> <Value>      Settable coordinate rotation**

Write coordinate rotation.

The notations <Value 1> to <Value 3> should be as follows:

- <Value 1>    Number of channel (0 = own channel)
- <Value 2>    Group of settable coordinate rotations (G54=1 to G57=4)
- <Value 3>    Number of angle (currently = 1)



**@438 <Value 1> <Value 2> <Value 3> <Value>****Additive settable coordinate rotation**

Write additive, settable coordinate rotation (G54 to G57).

Significance:

<Value 1> Number of channel (0 = own channel)

<Value 2> Group 1 to 4 (G54 to G57)

<Value 3> Number of angle (currently = 1).

**@439 <Value 1> <Value 2> <Value 3> <Value>****Programmable coordinate rotation**

Write programmable coordinate rotation.

Significance:

<Value 1> Number of channel (0 = own channel)

<Value 2> Group of programmable additive coordinate rotations  
(G58=1 to G59=2)

<Value 3> Number of angle (currently = 1).

**@43a <Value 1> <Value 2> <Value 3> <Value>****Additive programmable coordinate rotation**

Write additive programmable coordinate rotation (G58 and G59).

Significance:

<Value 1> Number of channel (0 = own channel)

<Value 2> Group 1 to 2  
(G58 and G59)

<Value 3> Number of angle (currently = 1).

Main group 4 / Subgroup 4: **Write R parameters to programmed setpoints**

**@440 <Value 3> <Value>****Programmed axis position**

This command makes it possible to program axes independently of axis names.

The number of the axis to be traversed is entered under <Value 1> and the position to be approached or the distance to be traversed is entered under <Value>.

If function "Axis duplication" is active, the value of the programmed axis position is assigned to both the leading and the following axis.

**Example:**

@440 K2 K100 L\_F

Value 100 is assigned to the traversing path of the second axis via a constant.

**@442 <Value 3> <Value>****Programmed spindle speed**

This command permits the spindle speed to be programmed.

The number of the spindle is entered under <Value 3> and the spindle speed under <Value>.

**@446 <Value>****Programmed radius**

This command makes it possible to program the radius independently of the address specified in the machine data. The numerical value is defined under<Value>.

**@447 <Value>****Programmed angle**

This command makes it possible to program the angle independently of the address specified in the machine data. The numerical value is defined under<Value>.

**@448 <Value 3> <Value>****Programmed interpolation parameter**

This command permits the interpolation parameters for circle and thread to be programmed.

The number of the axis is entered under <Value 3> and the interpolation parameter under <Value>.

Main group 4 / subgroup 8: **Write R-parameters in PLC signal bits**

**@482 <Value 1><Value 2><Value 3><Value>**

The state of the flag bits in the PLC is loaded via a parameter, a pointer or a constant. The PLC number is defined by <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 must be entered for the PLC number.

**Example:**

@482 K1 K2 K0 K0

The state of the defined PLC flag bit is loaded with 0.

**@483 <Value 1><Value 2><Value 3><Value 4><Value>**

The state of a data word bit in the PLC is loaded via a parameter, a pointer or a constant. The PLC number is defined by <Value1>, the DB number by <Value2>, the data word number by <Value3> and the bit address by <Value4>. K1 must be entered for the PLC number.

**Example:**

R60=1 R62=2 R63=4

@483 R60 R62 R63 K2 K1

The state of the defined data word bit is loaded with 1.

**Note:**

In the 3rd channel (simulation) 482 and 483 are overread. This can cause the program run to be changed.

Main group 4 / Subgroup e: <b>Write R parameter to system cells</b>
---

**@4e1 <Value 1> <Value 2> <Value>****Program spindle acceleration time  
constant**

This command permits the spindle acceleration time constant to be programmed.

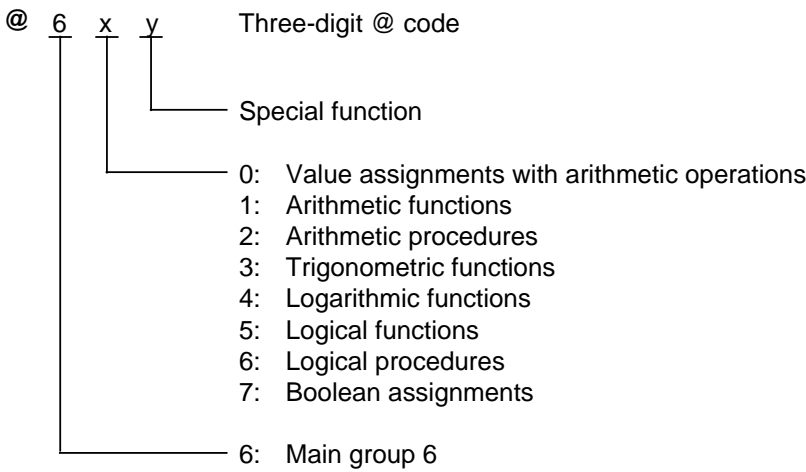
The spindle acceleration time constant defined under <Value> does not overwrite the values entered in the machine data, but an internal data cell. The machine data become active again after cancellation of thread machining (e.g. G00), after a change of machine data, after RESET and after POWER ON.

The spindle number is defined by <Value 1> and the gear stage by <Value 2>.



### 11.8 Mathematical functions

Main group 6 is broken down as follows:



Main group 6 / Subgroup 0: **Value assignments with arithmetic operations**

In this subgroup @ is not required. An incremental calculation with several notations on the right-hand side of the equation is allowed.

<Var>	=	<Value 1>	+	<Value 2>	Addition
<Var>	=	<Value 1>	-	<Value 2>	Subtraction
<Var>	=	<Value 1>	.	<Value 2>	Multiplication
<Var>	=	<Value 1>	/	<Value 2>	Division

Main group 6 / Subgroup 1: **Arithmetic functions**

@610 <Var> <Value>

Absolute value generation

The absolute-value part of the numerical value defined under <Value> is stored after <Var>.

Example:

R12=-34 L\_F

@610 R76 R12 L\_F

:

The R parameter R76 contains the absolute value (=34) from R12.

@613 <Var> <Value>

Square root

The square root is formed from the numerical value defined under <Value> and is stored after <Var>.

Example:

@613 R13 K64 L\_F

The square root is formed from the constant (=64) and the result (=8) is entered/stored in R13.

**@614 <Var> <Value 1> <Value 2>****Root from sum of squares**

The sum of the squares is formed from the numerical value defined under <Value 1> and <Value 2>, the square root is taken and is stored in <Var>.

**Example:**

```
R25=15 R26=20 L_F
```

```
@614 R77 R25 R26 L_F
```

The square root is formed from the sum of squares of the R parameters R25 (=225) and R26 (=400). The result (=25) is entered/stored in R77.

Main group 6 / Subgroup 2: **Arithmetic procedures**

**@620 <Var>****Incrementing**

The contents of the R parameter defined under <Var> are incremented.

**Example:**

```
R70=1 L_F
```

```
@620 R70 L_F
```

The content of parameter R70 is incremented; the new content is 2.

**@621 <Var>****Decrementing**

The contents of the R parameter defined under <Var> are decremented.

**Example:**

```
R70=1 L_F
```

```
@621 R70 L_F
```

The content of parameter R70 is decremented; the new content is 0.

**@622 <Var>****Integer component**

The integer component is formed from the numerical value defined by means of an R parameter or a pointer. The result is then in the same R parameter or pointer.

**Example:**

```
R60=2,9 L_F
```

```
@622 R60 L_F
```

The integer content of R60 is formed; the new content is 2.

Main group 6 / Subgroup 3: **Trigonometric functions**

**@630 <Var> <Value>**

**Sine**

The sine is formed from the angle value defined under <Value> and is stored after <Var>.

**Example:**

R27=30 L<sub>F</sub>

@630 R15 R27 L<sub>F</sub>

The sine of the content of R27 (=0.5) is entered/stored in R15.

**@631 <Var> <Value>**

**Cosine**

The cosine is formed from the angle value defined under <Value> and is stored after <Var>.

**@632 <Var> <Value>**

**Tangent**

The tangent is formed from the angle value defined under <Value> and is stored after <Var>.

**@634 <Var> <Value>**

**Arc sine**

The arc sine is formed from the numerical value defined under <Value> and is stored as an angle value after <Var>.

**Example:**

R35=0,70710678 L<sub>F</sub>

@634 R17 R35 L<sub>F</sub>

The arc sine is formed from the content of R35 and the result (=45) entered/stored in R17.

**@637 <Var> <Value 1> <Value 2>**

**Angle from 2 vector components**

The numerical values defined under <Value 1> and <Value 2> are viewed as vectors. The result is the angle between the component under <Value 2> and the sum vector.



***Only one constant <Const> is allowed as an operand for <Value 1> and <Value 2>. The other operand must be a variable <Var> (R parameter or pointer).***

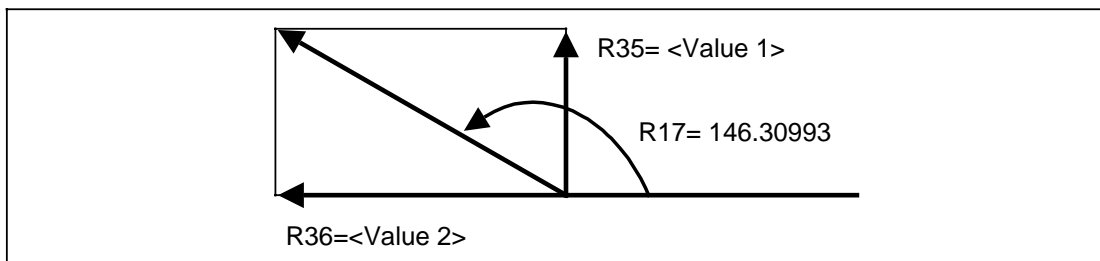


**Example:**

R35=20 R36=30 L<sub>F</sub>

@637 R17 R35 R36 L<sub>F</sub>

The angle is formed from the vector components of the contents of R parameters R35 and R36 and the result (=146.30993) is entered/stored in R17.



Example of @637

#### Main group 6 / Subgroup 4: **Logarithmic functions**

##### **@640 <Var> <Value>**

##### **Natural logarithm**

The natural logarithm is formed from the numerical value defined under <Value> and is stored in <Var>.

##### **Example:**

@640 R80 K10 L<sub>F</sub>

The natural logarithm is formed from the constant 10. The result (=2.3025846) is entered/stored in R80.

##### **@641 <Var> <Value>**

##### **Exponential function**

The exponential function  $e^x$  is formed from the numerical value defined under <Value> and is stored in <Var>.

##### **Example:**

@641 R80 K2.5 L<sub>F</sub>

The exponential function is calculated for the exponent defined by the constant. The result (=12.182496) is entered/stored in R80.

#### Main group 6 / Subgroup 5: **Logical functions**

##### **@650 <Var> <Var 1> <Value>**

##### **OR**

The bit patterns under <Var 1> and <Value> undergo a logical OR operation. The result is stored in <Var>.

##### **Example:**

R50=00101100 L<sub>F</sub>

R51=10110011 L<sub>F</sub>

@650 R52 R50 R51 L<sub>F</sub>

The pattern variables R50 and R51 undergo a logical OR operation and the result is stored in R52.

The content of R52 is 10111111.



**@651 <Var> <Var 1> <Value>**

**EXCLUSIVE OR**

The bit patterns under <Var 1> and <Value> undergo a logical exclusive OR operation. The result is stored in <Var>.

**@652 <Var> <Var 1> <Value>**

**AND**

The bit patterns under <Var 1> and <Value> undergo a logical AND operation. The result is stored in <Var>.

**@653 <Var> <Var 1> <Value>**

**NAND**

The bit patterns under <Var 1> and <Value> undergo a logical AND operation. The result is negated and stored in <Var>.

**@654 <Var> <Value>**

**NOT**

The bit pattern under <Value> is logically negated. The result is stored in <Var>.

**Example:**

R50=00101100 L<sub>F</sub>

@654 R52 R50 L<sub>F</sub>

The contents of the pattern variable R50 is negated and the result stored in R52. The content of R52 is 11010011.

**@655 <Var> <Var 1> <Value>**

**OR bit**

The bits under <Var 1> and <Value> undergo a logical OR operation. The result is stored in <Var>.

**@656 <Var> <Var 1> <Value>**

**EXCLUSIVE OR bit**

The bits under <Var 1> and <Value> undergo a logical EXOR operation. The result is stored in <Var>.

**@657 <Var> <Var 1> <Value>**

**AND bit**

The bits under <Var 1> and <Value> undergo a logical AND operation. The result is stored in <Var>.

**Example:**

R50=1 L<sub>F</sub>

R51=0 L<sub>F</sub>

@657 R52 R50 R51 L<sub>F</sub>

The Boolean variables R50 and R51 undergo a logical AND operation and the result is stored in R52. The content of R52 is 0.

**@658 <Var> <Var 1> <Value>**

**NAND bit**

The bits under <Var 1> and <Value> undergo a logical NAND operation. The result is stored in <Var>.

**Example:**

R50=1 L<sub>F</sub>

R51=0 L<sub>F</sub>

@658 R52 R50 R51 L<sub>F</sub>

The Boolean variables R50 and R51 undergo a logical AND operation and the result is negated and stored in R52. The content of R52 is 1.

**@659 <Var> <Value>**

**NOT bit**

The bit under <Value> is logically negated. The result is stored in <Var>.

Main group 6 / Subgroup 6: **Logical procedures****@660 <Var> <Const>****Delete bit in PATTERN**

The constant <Const> is used to define a bit (0 to 7) which is to be deleted in the bit pattern specified by means of <Var>.

**Example:**

R60=01100111 L<sub>F</sub>

@660 R60 K6 L<sub>F</sub>

Bit no. 6 of the pattern variable is deleted. The content of R60 is 00100111.

**@661 <Var> <Const>****Set bit**

The constant <Const> is used to define a bit (0 to 7) which is to be set to "1" in the bit pattern specified by means of <Var>.

**Example:**

R70=00000000 L<sub>F</sub>

@661 R70 K2 L<sub>F</sub>

Bit no. 2 of the pattern variable is set. The content of R70 is 00000100.

Main group 6 / Subgroup 7: **Boolean assignments****@671 <Var 1> <Var 2> <Value>****Equal to**

If the numerical values defined under <Var 2> and <Value> are equal, the Boolean variable <Var 1> is set to "1".

**Example:**

R50=11001100 L<sub>F</sub>

@671 R51 R50 K11001100 L<sub>F</sub>

As R50 equals the bit pattern of constant K, R51 is set to "1".

**@672 <Var 1> <Var 2> <Value>****Not equal to**

If the numerical values defined under <Var 2> and <Value> are not equal, the Boolean variable <Var 1> is set to "1".

**@673 <Var 1> <Var 2> <Value>****">"**

If the numerical value defined under <Var 2> is greater than the value under <Value>, the Boolean variable <Var 1> is set to "1".

**@674 <Var 1> <Var 2> <Value>****">="**

If the numerical value defined under <Var 2> is greater than or equal to the value under <Value>, the Boolean variable <Var 1> is set to "1".

**@675 <Var 1> <Var 2> <Value>****"<"**

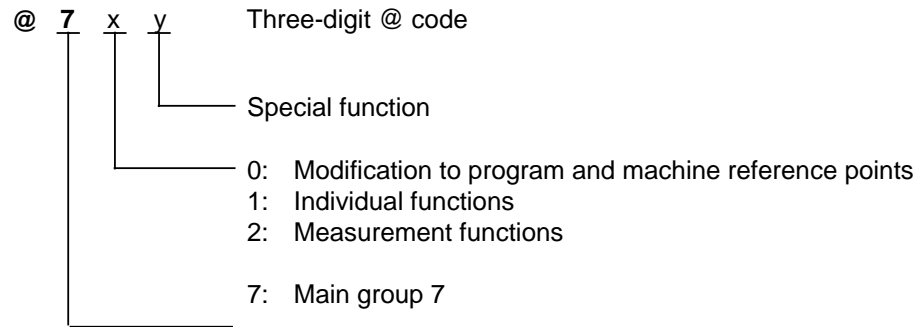
If the numerical value defined under <Var 2> is less than the value under <Value>, the Boolean variable <Var 1> is set to "1".

**@676 <Var 1> <Var 2> <Value>****"<="**

If the numerical value defined under <Var 2> is less than or equal to the value under <Value>, the Boolean variable <Var 1> is set to "1".

## 11.9 NC-specific functions

Main group 7 is broken down as follows:



Main group 7 / Subgroup 0:

**Modification to program and  
machine reference points**

### @706

**Position in the block referred to  
the machine actual value system**

The @706 command defaults a position with reference to machine zero which is then approached by the specified axes. The command is only effective for one block.

As many axes can be defaulted as can be traversed simultaneously by the NC. The axis positions to be approached are either programmed in DIN code or with the command @440 ....

Command @706 suppresses all zero offsets (settable, settable additive, programmable and external) and PRESET and DRF offsets.

In order to approach a position in the machine actual value system, the tool offsets must also be cancelled.

If machine data 5007.1 is set, G53 has the same effect as @706.

If the machine data is not set, G53 does not suppress PRESET and DRF offsets.

### Example:

@706 X1000 Z500 L\_F

The programmed paths in X and Y are approached with reference to machine zero.

Main group 7 / Subgroup 1 : **Individual functions****@710 <Var 1> <Var 2>****Reference preparation**

This command is required for reference preparation.

This command splits a contour programmed in a subroutine into individual blocks with the data being stored in R parameters.

The contour element is stored in a total of 8 R parameters starting from R parameter <Var 1> ( $R_n$ ).

Reference preparation requires a total of 4 R parameters as input data. The first of these R parameters is defined by <Var 2> ( $R_m$ ).

The input control parameter ( $R_m+3$ ) must be set to "1" before the first call of @710. The first contour element of the subroutine is stored in R parameters starting with the starting point of the contour ( $R_m+1$ ,  $R_m+2$ ). This point is not programmed in the subroutine.

Command @710 sets the control parameter to "0", so that with each further call the values of the next contour element are loaded. If the command recognizes "subroutine end" (M17), the output control parameter ( $R_n+7$ ) is automatically set to "1" or "2".

**Prerequisite**

1. Contour in subroutine
2. Starting point
3. Control parameter ( $R_m + 3$ ) = 1

**Parameter**

$R_m$	Subroutine number
$R_m + 1$	Starting point Y
$R_m + 2$	Starting point X
$R_m + 3$	Control parameter

&lt; Var 2 &gt;

**@710 loads into:**

$R_n$	Starting point Y
$R_n + 1$	Starting point X
$R_n + 2$	End point Y
$R_n + 3$	End point X
$R_n + 4$	Interpolation parameter J
$R_n + 5$	Interpolation parameter I
$R_n + 6$	G function
$R_n + 7$	Control parameter

&lt; Var 1 &gt;

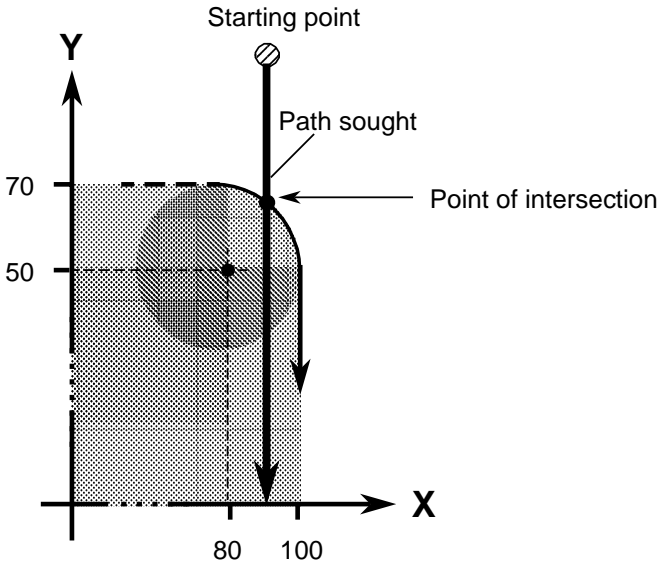
$R_n + 7 = 0$ :	Block without M17
$R_n + 7 = 1$ :	Block with M17
$R_n + 7 = 2$ :	M17 alone in block

@711 <Var 1> <Var 2> <Var 3>

Intersection calculation

This command is required for intersection point calculation.

@711 is used to find the point of intersection of a path sought with a contour element. The calculation of the intersection requires a total of 8 parameters as input data for the 1st contour element. The first R parameter number is defined in <Var 2> (Rn). The second contour starting from < Var 3> (Rr) is not implemented at the moment, i.e. this notation is not needed at the moment. Nevertheless some R parameter must be specified when programming. The output data of the intersection calculation are stored in a total of 3 R parameters starting with < Var 1 > (Rm). The search direction is programmed after the command as normal traversing movement. Both axis values are required to determine the point of intersection. The output data of the reference processing are used as input data for the intersection point calculation.



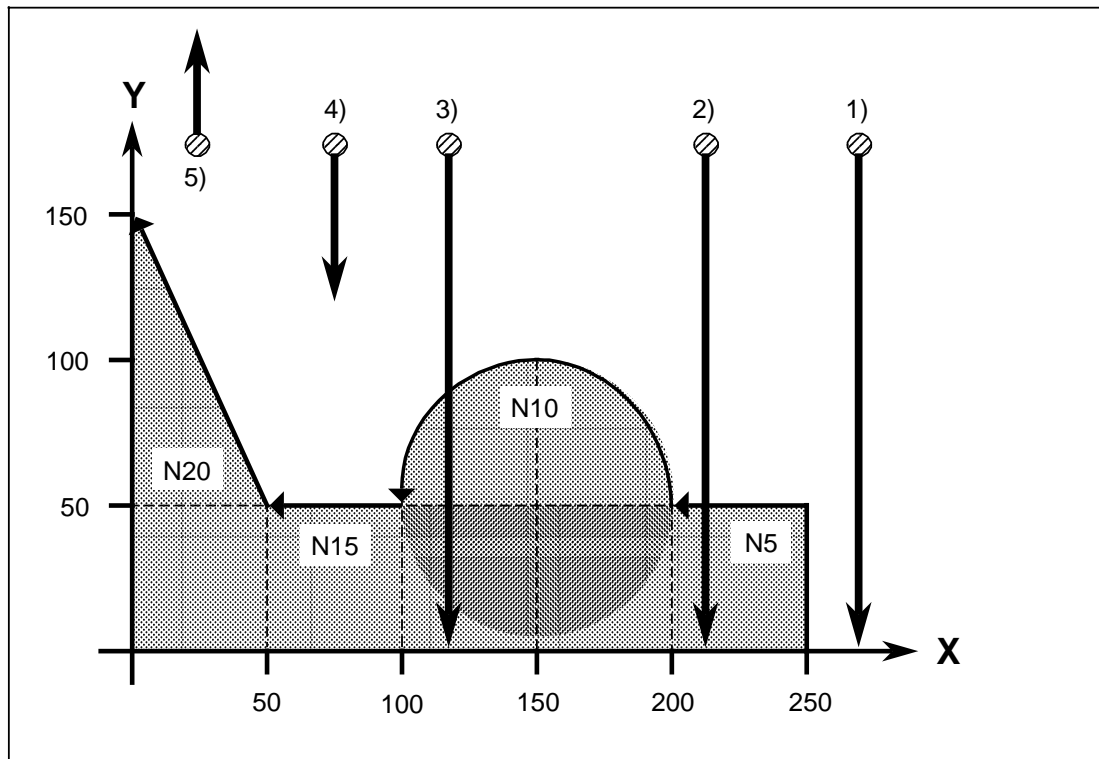
		Register contents
R <sub>n</sub>	Start of block Y	70
R <sub>n</sub> +1	Start of block X	80
R <sub>n</sub> +2	End of block Y	50
R <sub>n</sub> +3	End of block X	100
R <sub>n</sub> +4	Interpolation parameter J	- 20
R <sub>n</sub> +5	Interpolation parameter I	0
R <sub>n</sub> +6	G function	2

@711 R<sub>m</sub> R<sub>n</sub> R<sub>r</sub> G01 G90 Y0 X90 L<sub>F</sub>

Search direction,  
loads to:

R <sub>m</sub>	Result	1*)
R <sub>m</sub> +1	Point of intersection Y	67.320
R <sub>m</sub> +2	Point of intersection X	90.000

\*) 1=found, 0=not found

**Evaluation of intersection calculation with @711:**

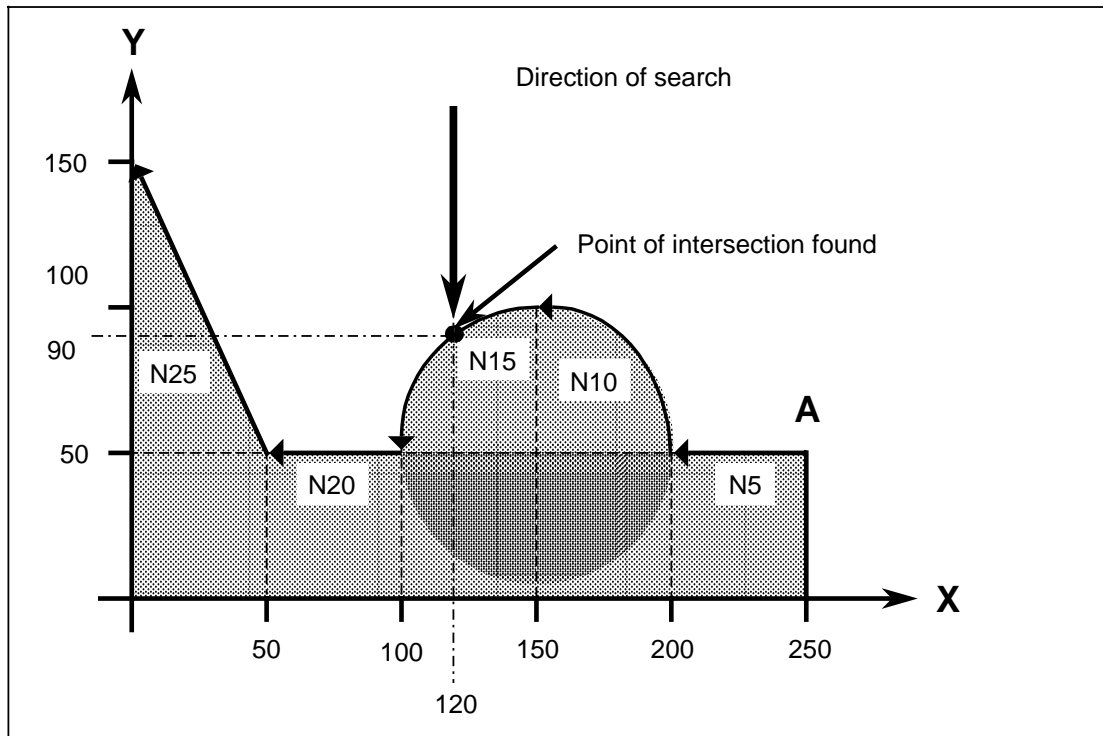
- 1) Intersection point not found
- 2) Intersection point found
- 3) Arc goes outside quadrant limits (!):  
Intersection point not found
- 4) Path sought was too short:  
Intersection point nevertheless found
- 5) Wrong search direction:  
Intersection point nevertheless found

The search direction defines a path somewhere along which the intersection point is expected. An intersection point is always found if this path or its prolongation meets the contour stored in  $R_n$  to  $(R_n+6)$ . Therefore an intersection is always found, even if the path programmed after @711 is in the wrong direction or is too short.

Function @711 now enters value 0 into register  $R_m$  if no intersection point was found, or it enters the value 1 if the search was successful. If an intersection point was found, the parameters  $(R_m+1)$  and  $(R_m+2)$  contain the intersection coordinates, otherwise, value 0. If a traversing movement is to be programmed with the values from  $(R_m+1)$  and  $(R_m+2)$ , a scan should be made in a loop whether an intersection point has been found or not.

### Example : Intersection point calculation with @711

The intersection point with the programmed contour at X=120 is searched for.



Subroutine of the contour:

```
L20 L_F
N5  X200 Y50L_F
N10 G03 X150 Y100 I-50 J0 L_F
N15 G03 X100 Y50 I0 J-50 L_F
N20 X50 Y50 L_F
N25 X0 Y150 L_F
N30 M17 L_F
```

Main program:

```
%30 L_F
G0 X120 Y150 L_F
R50=20 R51=0 R52=250 R53=1 R70=0 L_F

N300 @131 R70 K0 K305 L_F
@710 R54 R50 L_F
@711 R70 R54 R62 G01 G90 Y0 X120 L_F
@100 J-300 L_F
N305 G00 Y=R71 X=R72 L_F
:
:
M30 L_F
```

R50 to R54, input data; R70: flag for intersection calculation

While loop, as long as R 70=0

Reference preparation


Intersection point calculation

End of loop

Approach point of intersection in rapid traverse

The program loop is run through until an intersection point is found.

	$R_n$	$R_{n+1}$	$R_{n+2}$	$R_{n+3}$	$R_{n+4}$	$R_{n+5}$	$R_{n+6}$	$R_{n+7}$
1st call @710	50	250	50	200	0	0	1	0
@711	INTERSECTION POINT FOUND? No: $R_m = 0$							
2nd call @710	50	200	100	150	0	-50	3	0
@711	INTERSECTION POINT FOUND? No: $R_m = 0$							
3rd call @710	100	150	50	100	-50	0	3	0
@711	INTERSECTION POINT FOUND? Yes: $R_m = 1$							



In this example the intersection was found at Y=90 (R71) and X=120 (R72) after the 3rd call.

#### @713 <Var >

#### Start preparation cycles

This command is used to transfer the numerical value corresponding to the clearance of 1 mm in the current input format (with G70 = 0.03937 and with G71 =1) to the R parameter defined with <Var>. The numerical value "1" in the case of radius programming and "2" in the case of diameter programming is transferred to the next R parameter. The allowable parameter range for <Var> is R0 to R98 and R900 to R998.

#### @714

#### Stop decoding until buffer is empty

With this command "STOP DEC", the block preparation (decoding) is stopped until the buffer is empty.

In program processing, several program blocks are decoded in the control in advance and loaded into the NC buffer. This speeds up program processing, however in combination with certain NC commands (read actual value, measuring, data transmission NC-PLC) it can lead to erroneous program sequences. With the STOP DEC command (Stop decoding) the decoding of NC blocks in advance which stand after this command is stopped until the block (@714) with the STOP DEC command is processed. This ensures that the buffer is empty and information required in the next NC blocks is available.

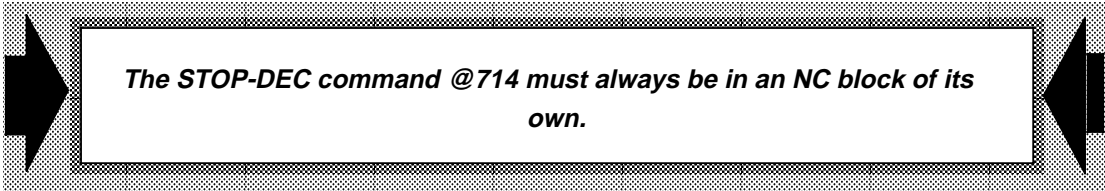
The STOP DEC statement must be programmed for the following information from the interface control, provided they are required for the next NC blocks:

- Machine data
- Setting data
- Tool offsets
- Zero offsets
- R parameters
- "Mirroring" signal

The STOP DEC command must be programmed before each reading of actual values in the own channel and after each measurement.

It is also necessary to program this command before writing zero offsets and tool offsets if the new values are intended to be effective only from this block onward.





***The STOP-DEC command @714 must always be in an NC block of its own.***

**Beispiel:**

M94 L<sub>F</sub>

@714 L<sub>F</sub>

@123 R60 K100 K5 L<sub>F</sub>

The part number is loaded to R60 by the PLC.

Stop decoding in order to allow the current part number to be taken into consideration in the next NC block.

Branch corresponding to part number

**@715**

**Stop decoding until buffer is empty for coordinate rotation**

If the angle of rotation of the settable or programmable coordinate rotation is loaded by means of the CL 800 language, this angle value is immediately used for calculation. This can cause the angle to be already calculated into preceding traversing blocks. If the angle is not to be valid until the next block, all previous blocks must have been processed (empty buffers).

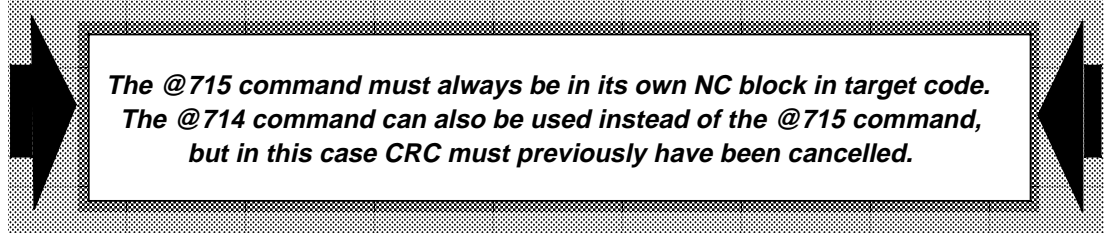
This can be achieved by programming STOP DEC 1 even if CRC has been selected. Decoding is then only reactivated when the buffer is empty before coordinate rotation.

**Example:**

@715

@437 K0 K1 K1 K30 L<sub>F</sub>

Coordinate rotation G54 with angle of rotation 30° in channel 0 (own channel)



***The @715 command must always be in its own NC block in target code. The @714 command can also be used instead of the @715 command, but in this case CRC must previously have been cancelled.***

Main group 7/subgroup 2 : **Measurement functions**

**@720 <Var> <Value>****Inprocess measurement referred to machine zero**

This value is used in measuring cycles.

The measurement function is used to determine the actual values of the moving axes at the moment of an input signal from the probe. The actual values are acquired directly by the measuring cycle of the control when the edge of the switching signal from the probe is recognized. The values are stored after the <Var> parameter with axis numbers in ascending order and they are referred to machine zero. Then the control generates a "Delete distance to go", i.e. the distances to go of all axes are deleted. The setpoint 0 is predefined by the control as jump function. The axes' remaining deceleration distances are travelled, i.e. the following errors are eliminated. Consequently the next traversing blocks have to be programmed in absolute dimensions (G90).

The acquired actual values of the axes traversed at the moment of measuring are stored after the parameter defined under <Var> and with axis numbers in ascending order.

The number of the measurement input (1 or 2) is defined by <Value>.

The traversing paths of the axes (setpoints) are programmed in the same NC block using the DIN code or the @440 command. The setpoints of the axes are referred to machine zero.

When the function "axis duplication" is active, command @440 applies both to the leading, programmed axes and to the duplicated axes.

**Example:**

```

.
.
.
@720 R93 K1
@440 K1 R70 L_F

```

The actual value of the first axis is measured and loaded to R93.  
The value from R70 is assigned to the traversing path of the first axis via a constant.

```

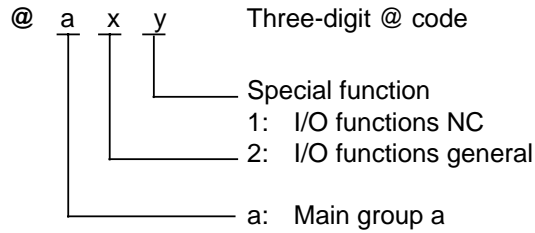
.
.
.

```

***If the function "axis duplication" is active, the programmed path is duplicated on the following axis but only the actual value of the leading axis is measured. Both axes stop when the measuring probe is triggered.***



## 11.10 I/O functions



### Main group a/subgroup 1: I/O functions NC

#### @a15 <Value 4> <Value 5>

(Select menu from NC program)

This command selects menus (displays and menu texts) in the user area and the standard area from the NC program.

<Value 4>= 0: User area  
          1: Standard area  
<Value 5>= 1: to 254: Menu number

#### Example:

```
.
N..  @a15    K1    K20    LF
.
```

Calls the standard display for data input/output.

#### Application:

- Calling configured displays which provide the user with information on the current machining cycle.
- User screenforms that must be acknowledged by the user.

#### @a1b

(Return to the initial menu)

After selecting any menu with @a15, with this command you can return to the menu you had selected before the first @a15 call.

#### Example:

At present the PLC status display is selected.

```
.
N..  @a15    K1    K20          Calling the I/O display
N..  @a1b                                Returning to the PLC status display
.
```

#### Application:

A display selected and overwritten by the user can be called again.

## Main group a/Subgroup 2: I/O functions general

### @a20 <Value> Select RS232C (V.24) interface

This command selects the interface through which data input/output is to be handled.

<Value> = Number of interface (1 or 2).

#### Example:

```
N. . @a20 K2 L_F
```

The second interface is selected.

#### Note:

The interface (1 or 2) selected by the operator is not affected by this command.

**Before data is input or output with the commands  
@a25, @a26, @a27, @a28 and @a29,  
the interface must be selected with @a20 in the program.**

### @a25 <Value 1>

### Output of zero offsets via RS232C (V.24)

With this command, zero offsets and channel-specific angles of rotation can be output via RS232C (V.24).

<Value 1> = 0	Zero offsets G54-G57	
1	Angle of rotation for coordinate rotation	channel 1
2	"	channel 2
3	"	channel 3.

#### Example:

```
N. . @a20 K2 L_F
```

Settable zero offset

```
N. . @a25 K0 L_F
```

G54-G57 are output via the second RS232C (V.24) interface.

#### Note :

- Selection of the interface with @a20 must be programmed before calling @a25.
- A machine data bit can be used to select whether data is to be output simultaneously with the running program or whether the block change should be disabled for the time of transmission.

### **@a26 <Value 2> <Value 3> <Value 4>                      Output of data via RS232C (V.24)**

This command permits particular data to be output via RS232C (V.24). Parameters must be assigned in <Value 2> to designate the data.

<Value 2> = 1: Main program  
2: Subroutines  
5: NC machine data  
6: Tool offset  
8: Setting data  
9: PLC machine data.

The start address is defined with <Value 3>, the end address of the data block with <Value 4>.

#### **Example:**

```
N.. @a20 K1 L_F                      The available part programs with program identifiers %1.
N.. @a26 K1 K1 K10 L_F              %1 to %10 are output via the first RS232C (V.24) interface.
```

#### **Note :**

- Before data output, the interface must be selected with command @a2.
- A machine data bit can be used to select whether data is to be output simultaneously with the running program or whether the block change should be disabled for the time of transmission.
- If the data to be output are changed in the subsequent program part, a @714 (STOP DEC) must be programmed after the command @a26.

### **@a27 <Value 1> <Value 3> <Value 4>                      Parameter output via RS232C (V.24)**

With this block individual R parameter blocks can be output via the RS232C (V.24). The channel number is defined via <Value 1>. If a zero is specified, the R parameters are always output from the own channel. For central variables zero is always specified. The start address of the block is defined with <Value 3>, the end address with <Value 4>. With machine data 5147.1= "1", the block change can be disabled in the current program at the same time as R parameters are output.

<Value 1> = Channel number  
0: Central R parameter or own channel  
1: R parameters channel 1  
2: R parameters channel 2  
3: R parameters channel 3.

<Value 3> = Start address  
000 to 699      for global parameters  
700 to 999      for central parameters.

<Value 4> = End address  
000 to 699      for global parameters  
700 to 999      for central parameters.

#### **Example:**

```
N.. @a20 K1 L_F                      Select first RS232C (V.24) interface
N.. @a27 K1 K0 K699 L_F              All global parameters of channel 1 are output
N.. @a27 K0 K700 K999 L_F            All central R parameters are output
N.. R1=20 R11=40 L_F
N.. @a27 K1 R1 R11 L_F                Global R parameters 20 to 40 from channel 1 are output.
```

**Application:**

- Output of measured values to a printer
- Output of parameters to a linked computer
- Output of data to another NC control.

**Note:**

Selection of the interface with @a20 must be programmed before calling @a27. If the R parameters to be output are changed in the subsequent program part, a @714 (STOP DEC) must be programmed after the command.

A machine data bit can be used to select whether output is to occur simultaneously with the running program or whether the block change should be disabled for the time of transmission.

**@a28 <Value 2>****Reading in data via RS232C (V.24)**

With this command reading in data via the RS232 C interface can be started. <Value 2> specifies what type of data may be read in.

<Value 2> =

- 0: The data type is not checked
- 1: Main programs
- 2: Subroutines
- 3: Clear programs
- 5: NC machine data
- 6: Tool offsets
- 7: Zero offsets
- 8: Setting data
- 9: PLC machine data
- 10: R parameters.

**Example:**

```

.
N.. @a20 K1 L_F      Selection of first RS232C (V.24) interface.
N.. @a28 K10 L_F     The control expects R parameters in the input. If a different type of
                      data is received, an error message is output.

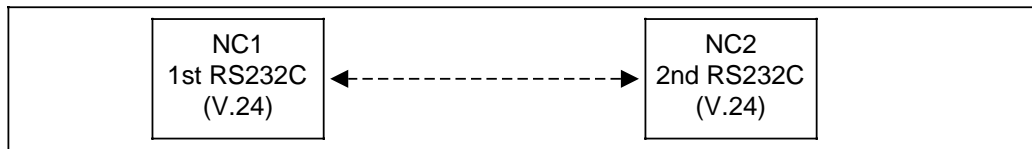
```

**Note:**

Selection of the interface with @a20 must be programmed before calling @a28.

A machine data bit can be used to select whether the data are to be read in simultaneously with the running program or whether the block change should be disabled for the time of data input.

If in the subsequent program part processing is to involve, the R parameters read in, the command @714 (STOP DEC) must be programmed after transmission. Commands @a27 and @a28 are a simple way of transferring data between two SINUMERIK controls.



### Prerequisites for NC-NC link:

Interface-specific setting data:

- No header or trailer when reading out.

If the same interface is also required for linking with other devices, the respective setting data can be converted before transmission and subsequently reset.

### Example:

N.. @311 R700 K5016 L <sub>F</sub>	SD 5016 is saved into R700
N.. @411 K5016 K00000010 L <sub>F</sub>	Convert SD 5016
N.. @a20 K1 L <sub>F</sub>	Select 1st RS232C (V.24)
N.. @a27 K1 K50 K50 L <sub>F</sub>	R50 is output
N.. @714 L <sub>F</sub>	STOP DEC
N.. @411 K5016 R700 L <sub>F</sub>	Reset SD 5016.

### @a29

### Output of ETX via RS232C (V.24)

With this command, the end of transmission character defined in the setting data can be output via the RS232C (V.24) interface.

### Example:

N.. @a20 K1 L <sub>F</sub>	Via the first RS232C (V.24) interface, R parameters
N.. @a27 K1 K0 K100 L <sub>F</sub>	R0 to R100 are output from channel 1 followed
N.. @a29 L <sub>F</sub>	by the end of transmission character.

### Application:

- After a sequence of outputs without end of transmission character the receiver is informed of the end of the data package with @a29.



## 11.11 Operator guidance macro (OGM)

Main group c/subgroup c: **Back translation**

### @ccc (Identifier for the beginning of the OGM data block)

Parameterization of machining cycles can be assisted graphically by means of configured input displays. If the user supplies one of these input displays with new machining parameters, these values are first stored in machine input buffers (MIB). For further processing of the input values press a softkey to start an operator guidance macro (OGM). This OGM assigns the values from the MIB to specified addresses such as G, N, R, X, Y, Z, L, U, I, K, etc. By combining the addresses and the machining input buffer in an operator guidance macro the following data block can be generated:

```
G01 G90 R10=50 R11=20 R12=15 L901 L_F
```

The OGM enters this data block in a part program as program block. For this purpose, the program must be selected in editing mode beforehand and the insert position must be marked with the cursor.

"Back translation" means that a program block generated with the aid of an operator guidance macro is put back in its input display. This makes graphically supported editing possible.

A data block prepared for back translation can be recognized by two additional elements:

- 1: @ccc (x:y:z) identifies the beginning of the data block. The axis names specify the level plane for which the data block has been programmed.
- 2: (OGM: GROOVING) identifies the end of the data block.  
"GROOVING" is the name of the OGM.

When such a data block is marked with the cursor in the part program (editing mode), the associated input display is called after pressing the "INPUT DISPLAY" softkey.

The information contained in the data block is copied in the MIB and displayed. The complete data block is then:

```
@ccc (x:y:z) G01 G90 R10=50 R11=20 R12=15 L901 (OGM:GROOVING) L_F
```

Back translation is no longer possible if the order of the parameters is changed. Message: "=GM cannot be back translated".

### Configuration of the back translation function

Every OGM which is to generate a data block which can be back translated, contains an unambiguous name in the first block. The name is programmed like a comment and must have the following format:

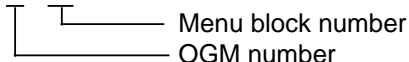
```
(OGM:GROOVING1) . "GROOVING1" is the name.
```

This name is entered in an assignment list. OGM 999 has the function of the assignment list. It assigns the OGM number and the menu block number (number of the configured input display) to the name.

The name may have a length of 12 characters. Blanks are not permitted. If the name contains more than 12 characters or if it is not entered in the assignment list the message "OGM not found" is output.

#### Example: Structure of the assignment list in OGM 999

```
% OGM 999
GROOVING1 = 7, 11
GROOVING2 = 9, 12
MACHINING = 12, 13
TAPER = 4, 25
```



The @ccc identifier is generated when after the OGM number the OGM name is configured : (OGM:<name>). A data block configured like this can be edited later graphically by means of the "Back translation" function.

#### Example: Structure of OGM 7

Configured data block

```
% OGM 7
(OGM: GROOVING1)
N 150 G 100 G 101 R10= 102 R11= 103 R12= 104 R13= 105 L_F
R140= 106 R15= 107 L901 L_F
```

This OGM 7 would insert the following data block in the part program at the position marked with the cursor:

```
@ccc (X:Y:Z) N2 G01 G90 R10=50 R11=20 R12=15 R13=25 L_F
R14=27 R15=39 L901 (OGM:GROOVING) L_F
```

#### Note:

- The maximum block length of 120 characters and the maximum OGM name length of 12 characters should be observed.
- The first instruction should be a block number ("N ..."). Thus the data block can be found quicker when editing.
- In the part program the OGM may not generate a block number or a comment at the beginning of the block.
- To delete the data block position the cursor in the data block to be deleted, enter "N+" and press the "CANCEL" key or start the procedure with the configured softkey "Delete block" after having positioned the cursor.

## 11.12 @ code table

Key:

y Relational operator rop

0: . . . . No condition

1:= . . . . Equal to

2: . . . . Not equal to

3: . . . . Greater than

4: = . . . . Greater than or equal to

5: . . . . Less than

6: = . . . . Less than or equal to

7: . . . . True

8: . . . . not

1) Not at CL800 level

2) " Condition"

a) Var =Boolean variable

b) Var . Const =Bit from pattern

c) Var "rop" Value

d) Extended condition

3) No pointers possible,

only Const definable on CL 800 level

@ code	CL 800 statement	Function
@00f		Enable for softkey start
@041 R Par 1 R Par 2	(Push Block) <sup>1)</sup>	Saving of a group of R parameters to stack
@042 Const R Par n . . . R Par 1	(Pop) <sup>1)</sup>	Fetching saved R parameters from stack
@043 R Par 1 R Par 2	(Pop Block) <sup>1)</sup>	Fetching group of saved R parameters from stack
@100 Const	GOTO Label ;	Absolute jump to NC block
@100 R-Par <sup>3)</sup>		
@111 Var Value1 Const 1 Value2 Const 2 . . Value n Const n	CASE Var = Value 1 : Statement 1 ; . . = Value n : Statement n ;	CASE branching
@12y Var Value Const	IF condition <sup>2)</sup> THEN Statement 1 ; [ELSE Statement 2 ;] END IF;	IF-THEN-ELSE statement y relational operator rop Var R parameter or pointer
@13y Var Value Const	WHILE condition <sup>2)</sup> DO Statement ;	Repeat statement with sampling of repeat condition at start. y relational operator rop
@14y Var Value Const	REPEAT Statement ; UNTIL condition; <sup>2)</sup>	Repeat statement with sampling of repeat condition at end. y relational operator rop

@ code	CL 800 statement	Function
@151 <i>Var Value 2 Const</i>	FOR <i>Var</i> = Value 1 TO Value 2 DO Statement ;	Repeat statement with repetitions until <i>Var</i> has incrementally reached Value 2
@161 <i>Var Value 2 Const</i>	FOR <i>Var</i> = Value 1 DOWNT0 Value 2 DO Statement ;	Repeat statement with repetitions until <i>Var</i> has decrementally reached Value 2
@200 <i>Var</i>	CLEAR( <i>Var</i> );	Delete variable
@201 <i>Var Value</i>	<i>Var</i> = Value	Load variable with value
@202 <i>Var 1 Var 2</i>	XCHG ( <i>Var 1</i> , <i>Var 2</i> );	Swapping of variable contents
@203 <i>Var 1 Var 2 Const</i>		Reading of a bit from bit pattern
@210 <i>Value 3 Value 4</i>	CLEAR MIB (<Value 3>, <Value 4>);	Clear machine input buffer Value 3: MIB start address 0 . . .499 Value 4: MIB end address 0 . . .499
@211 <i>Var Value 1</i>	<i>Var</i> = MIB ( <i>Value</i> );	Load numeric variable <i>Var</i> with the content of the input buffer cell Value 1 Value 1: buffer no. 0 . . . 499
@212 <i>Value 1 Value</i>	MIB ( <i>Value1</i> )= <i>Value</i> ;	Load MIB (machine input buffer) cell Value 1 with the numerical number Value Value 1: MIB No. 0 . . . 499
@300 <i>Var Value 1</i>	<i>Var</i> =MDN ( <i>Value 1</i> );	Machine data NC Value 1: Addr. 0 . . . 4999
@301 <i>Var Value 1</i>	<i>Var</i> =MDNBY ( <i>Value 1</i> );	Machine data NC byte Value 1: Byte addresses 5000 . . . 6999
@302 <i>Var Value 1 Value 2</i>	<i>Var</i> =MDNBI ( <i>Value 1</i> , <i>Value 2</i> );	Machine data-NC bit Value 1: Byte addresses 5000 . . . 6999 Value 2: Bit addr. 0 . . . 7
@306 <i>Var Value 1</i>	<i>Var</i> =MDP ( <i>Value 1</i> );	Machine data PLC Value 1: Addr. 0 . . . 1999
@307 <i>Var Value 1</i>	<i>Var</i> =MDPBY ( <i>Value 1</i> );	Machine data PLC bytes Value 1: Byte addresses 2000 . . . 3999

@ code	CL 800 statement	Function
@308 Var Value 1 Value 2	Var =MDPBI ( Value 1 , Value 2 );	Machine data PLC bit Value 1: Byte addresses 2000 . . . 3999 Value 2: Bit addr. 0 . . . 7
@310 Var Value 1	Var =SEN ( Value 1 );	Setting data NC Value 1: Addr. 0 . . . 4999
@311 Var Value 1	Var =SENB ( Value 1 );	Setting data NC byte Value 1: Byte addresses 5000 . . . 9999
@312 Var Value 1 Value 2	Var =SENB ( Value 1 , Value 2 );	Setting data NC bit Value 1: Byte addresses 5000 . . . 9999 Value 2: Bit addr. 0 . . . 7
@320 Var Value 1 Value 2 Value 3	Var =TOS ( Value 1 , Value 2 , Value 3 );	Tool offset Value 1: 0 Value 2: D no. 1 . . 99 Value 3: P no. 0 . . (9)
@330 Var Value 1 Value 2 Value 3	Var =ZOA ( Value 1 , Value 2 , Value 3 );	Settable zero offset (G54-G57) Value 1: Group 1 . . . 4 (G54-G57) Value 2: Axis no. 1, 2 . . . Value 3: (0/1) coarse/fine
@331 Var Value 1 Value 2	Var =ZOPR ( Value 1 , Value 2 );	Programmable zero offset (G58,G59) Value 1: Group 1 or 2 (G58 or G59) Value 2: Axis no. 1, 2 . . .
@332 Var Value 2	Var =ZOE ( Value 2 );	External zero offset from PLC Value 2: Axis no. 1, 2 . . .
@333 Var Value 2	Var =ZOD ( Value 2 );	DRF offset Value 2: Axis no. 1, 2 . . .
@334 Var Value 2	Var =ZOPS ( Value 2 );	PRESET offset Value 2: Axis no. 1, 2 . . .
@336 Var Value 2	Var =ZOS ( Value 2 );	Total offset Value 2: Axis no. 1, 2 . . .
@337 Var Value 1 Value 2 Value 3	Var =ZOADW ( Value 1 , Value 2 , Value 3 );	Settable coordinate rotation (G54-G57) Value 1: Channel no. 0 . . . 2 Value 2: Group 1 . . . 4 (G54- G57) Value 3: = Angle no. (=1)

@ code	CL 800 statement	Function
<b>@338</b> <i>Var Value 1 Value 2 Value 3</i>	Var =ZOPRDW ( Value 1 , Value 2 , Value 3 );	Programmable coordinate rotation (G58-G59) Value 1: Channel no. 0 . . . 2 Value 2: Group 1 or 2 (G58 or G59) Value 3: = Angle no. (=1)
<b>@342</b> <i>Var Value 1 Value 3</i>	Var =PRSS ( Value 1 , Value 3 );	Read programmed spindle speed Value 1: Channel no. 0 . . . 2 Value 3: Spindle no. 0 . . 6
<b>@345</b> <i>Var Value 1 Value 2</i>	Var =PRVC ( Value 1 , Value 2 );	Programmed cutting speed Value 1: Channel no. 0 ... 2 Value 2: 0=G 96
<b>@360</b> <i>Var Value 2</i>	Var =ACPW ( Value 2 );	Actual axis position workpiece-related Value 2: Axis no. 1, 2 . . .
<b>@361</b> <i>Var Value 2</i>	Var =ACPM ( Value 2 );	Actual axis position machine-related Value 2: Axis no. 1, 2 . . .
<b>@363</b> <i>Var Value 2</i>	Var = ACSP ( Value 2 );	Actual spindle position Value 2: Spindle no. 0 . . 6
<b>@364</b> <i>Var Value 2</i>	Var =ACSS ( Value 2 );	Actual spindle speed Value 2: Spindle no. 0 . . 6
<b>@367</b> <i>Var Value 1</i>	Var =ACAS ( Value 1 );	Read axis no. of the current plane/leading spindle no. into R parameter Var : Var+0: no. of horizontal axis Var+1: no. of vertical axis Var+2: no. of axis vertical to plane Var+3: no. of axis in which length 2 acts (tool type 30) Var+4: no. of control spindle Value 1: Channel no. 0 ... 2
<b>@36a</b> <i>Var Value 1</i>	Var =ACD ( Value 1 );	Actual D function Value 1=0

@ code	CL 800 statement	Function
@36b Var Value 1 Value 3	Var =ACG ( Value 1 , Value 3 );	Read the G function from the main memory of the current block Value 1: Channel no. 0 ... 2 Value 3: internal G group to which G function belongs 0 ... 15
@371 Var Value 1 Value 3	Var =SOB ( Value 1 , Value 3 );	Special bit Value 1: Channel no. 0 ... 2 =Channel-dependent 99=Channel-independent Value 3: Bit no. 0 ... 7
@380/ @381 Var Value 1 Value 2 Value 3	VAR )= PLCI/PLCQ ( Value 1 , Value 2 , Value 3 );	PLC input/output bit Value 1: PLC No. 1 Value 2: Byte address 0..127 Value 3: Bit No. 0...7
@382 Var Value 1 Value 2 Value 3	VAR )=PLCF ( Value 1 , Value 2 , Value 3 );	PLC flag bit Value 1: PLC No. 1 Value 2: Byte address 0..255 Value 3: Bit No. 0...7
@383 Var Value 1 Value 2 Value 3	VAR )=PLCW ( Value 1 , Value 2 , Value 3 , Value 4 );	PLC data word bit Value 1: PLC No. 1 Value 2: DB No. 0...255 Value 3: DW No. 0...255 Value 4: Bit No. 0...15
@3e4 Var Value 1	Var =AGS ( Value 1 );	Read active gear stage Value 1: Spindle no. 0 to 6
@400 Value 1 Value	MDN ( Value 1 )= Value ;	Machine data NC Value 1: Adr. 0 ... 4999
@401 Value 1 Value	MDNBY ( Value 1 ) = Value ;	Machine data NC byte Value 1: Byte addresses 5000 ... 6999
@402 Value 1 Value 2 Value	MDNBI ( Value 1 , Value 2 ) = Value ;	Machine data NC bit Value 1: Byte addresses 5000 ... 6999 Value 2: Bit addr. 0 ... 7
@406 Value 1 Value	MDP ( Value 1 )= Value ;	Machine data PLC Value 1: Addr. 0 ... 1999
@407 Value 1 Value	MDPBY ( Value 1 ) = Value ;	Machine data PLC byte Value 1: Byte addresses 2000 ... 3999

@ code	CL 800 statement	Function
<b>@408</b> Value 1 Value 2 Value	MDNBI ( Value 1 , Value 2 ) = Value ;	Machine data PLC bit Value 1: Byte addresses 2000 . . . 3999 Value 2: Bit addr. 0 . . . 7
<b>@410</b> Value 1 Value	SEN ( Value 1 )= Value ;	Setting data NC Value 1: Addr. 0 . . 4999
<b>@411</b> Value 1 Value	SENB ( Value 1 )= Value ;	Setting data NC byte Value 1: Byte addresses 5000 . . . 9999
<b>@412</b> Value 1 Value 2 Value	SENB Value 1 , Value 2 = Value ;	Setting data NC bit Value 1: Byte addresses 5000 . . . 9999 Value 2: Bit addr. 0 . . . 7
<b>@420</b> Value 1 Value 2 Value 3 Value	TOS ( Value 1 , Value 2 , Value 3 )= Value ;	Tool offset Value 1: 0 Value 2: D no. 1 . . 99 Value 3: P no. 0 . . 7 (9)
<b>@423</b> Value 1 Value 2 Value 3 Value	TOAD ( Value 1 , Value 2 , Value 3 )= Value ;	Tool offset additive Value 1: 0 Value 2: D no. 1 . . 99 Value 3: P no. 0 . . 7 (9)
<b>@430</b> Value 1 Value 2 Value 3 Value	ZOA ( Value 1 , Value 2 , Value 3 )= Value ;	Settable zero offset (G54-G57) Value 1: Group 1 . . . 4 (G54-G57) Value 2: Axis no. 1, 2 . . . Value 3: (0 / 1) coarse/fine
<b>@431</b> Value 1 Value 2 Value 3 Value	ZOFA ( Value 1 , Value 2 , Value 3 )= Value ;	Settable zero offset additive Value 1: Group 1 . . . 4 (G54-G57) Value 2: Axis no. 1, 2 . . . Value 3: (0 / 1) coarse/fine
<b>@432</b> Value 1 Value 2 Value	ZOPR ( Value 1 , Value 2 )= Value ;	Programmable zero offset (G58,G59) Value 1: Group1 or 2 (G58 or G59) Value 2: Axis no. 1, 2 . . .
<b>@434</b> Value 2 Value	ZOD ( Value 2 )= Value ;	DRF offset Value 2: Axis no. 1, 2 . . .
<b>@435</b> Value 2 Value	ZOPS ( Value 2 )= Value ;	PRESET offset Value 2: Axis no. 1, 2 . . .



@ code	CL 800 statement	Function
<b>@437</b> <i>Value 1 Value 2 Value 3 Value</i>	ZOADW ( Value 1 , Value 2 , Value 3 )= Value ;	Settable coordinate rotation absolute Value 1: Channel no. 0 . . . 2 Value 2: Group 1 . . . 4 (G54-G57) Value 3: Angle no. (= 1)
<b>@438</b> <i>Value 1 Value 2 Value 3 Value</i>	ZOFADW ( Value 1 , Value 2 , Value 3 ) = Value ;	Programmable coordinate rotation additive Value 1: Channel no. 0 . . . 2 Value 2: Group 1 . . . 4 Value 3: Angle no. (= 1)
<b>@439</b> <i>Value 1 Value 2 Value 3 Value</i>	ZOPRDW ( Value 1 , Value 2 , Value 3 )= Value ;	Programmable coordinate rotation Value 1: Channel no. 0 . . . 2 Value 2: Group 1 or 2 (G58-G59) Value 3: Angle no. (= 1)
<b>@43a</b> <i>Value 1 Value 2 Value 3 Value</i>	ZOFPROW ( Value 1 , Value 2 , Value 3 )= Value ;	Programmable coordinate rotation additive Value 1: Channel no. 0 . . . 2 Value 2: Group 1 or 2 (G58 or G59) Value 3: Angle no. (= 1)
<b>@440</b> <i>Value 3 Value</i>	PRAP ( Value 3 )= Value ;	Programmed axis position Value 3: Axis no. 1, 2 . . .
<b>@442</b> <i>Value 3 Value</i>	PRSS ( Value 3 )= Value ;	Programmed spindle speed Value 3: Spindle no. 0 . . . 6
<b>@446</b> <i>Value</i>	PRAD= Value ;	Programmed radius
<b>@447</b> <i>Value</i>	PANG= Value ;	Programmed angle
<b>@448</b> <i>Value 3 Value</i>	PRIP ( Value 3 )= Value ;	Programmed interpolation parameter for circle and thread Value 3: Axis no. 1,2,...
<b>@482</b> <i>Value 1 Value 2 Value 3 Value</i>	PLCF ( Value 1 , Value 2 , Value 3 ))= Value ;	PLC flag bit Value 1: PLC No. 1 Value 2: Byte adr. 100...115 136...199 224...255 Value 3: Bit No. 0...7

@ code	CL 800 statement	Function
<b>@483</b> <i>Value 1 Value 2 Value 3 Value 4 Value</i>	PLCW ( Value 1 , Value 2 , Value 3 )= Value ;	PLC data word bit Value 1: PLC No. 1 Value 2: DB No. 1...255 Value 3: DW No. 0...255 Value 4: Bit No. 0...15
<b>@4e1</b> <i>Value 1 Value 2 Value</i>	SATC ( Value 1 , Value 2 )= Value ;	Write spindle acceleration time constant Value 1: Spindle no. 0 to 6 Value 2: Gear stage 1 to 8 Value : Spindle acceleration time constant 0 to 16000
<i>Var = Value 1 + Value 2 ; Var = Value 1 - Value 2 ; Var = Value 1 · Value 2 ; Var = Value 1 / Value 2 ;</i>	<i>Var = Value 1 + Value 2 ; Var = Value 1 - Value 2 ; Var = Value 1 · Value 2 ; Var = Value 1 / Value 2 ;</i>	Addition Subtraction Multiplication Division
<b>@610</b> <i>Var Value</i>	<i>Var=ABS ( Value );</i>	Absolute value generation
<b>@613</b> <i>Var Value</i>	<i>Var=SQRT ( Value );</i>	Square root
<b>@614</b> <i>Var Value 1 Value 2</i>	<i>Var=SQRTS ( Value 1 , Value 2 );</i>	Root from sum of squares
<b>@620</b> <i>Var</i>	<i>INC ( Var );</i>	Incrementing of "Var" by 1
<b>@621</b> <i>Var</i>	<i>DEC ( Var );</i>	Decrementing of "Var" by 1
<b>@622</b> <i>Var</i>	<i>TRUNC ( Var );</i>	Integer
<b>@630</b> <i>Var Value</i>	<i>Var=SIN ( Value );</i>	Sine
<b>@631</b> <i>Var Value</i>	<i>Var=COS ( Value );</i>	Cosine
<b>@632</b> <i>Var Value</i>	<i>Var=TAN ( Value );</i>	Tangent
<b>@634</b> <i>Var Value</i>	<i>Var=ARC SIN ( Value );</i>	Arc sine
<b>@637</b> <i>Var Value 1 Value 2</i>	<i>Var=ANGLE ( Value 1 , Value 2 );</i>	Angle from two vector components
<b>@640</b> <i>Var Value</i>	<i>Var=LN ( Value );</i>	Natural logarithm
<b>@641</b> <i>Var Value</i>	<i>Var=INV LN ( Value );</i>	e <sup>x</sup> exponential function
<b>@650</b> <i>Var Var 1 Value</i>	<i>Var = Var 1 OR Value ;</i>	OR
<b>@651</b> <i>Var Var 1 Value</i>	<i>Var = Var 1 XOR Value ;</i>	EXKLUSIVE OR
<b>@652</b> <i>Var Var 1 Value</i>	<i>Var = Var 1 AND Value ;</i>	AND

@ code	CL 800 statement	Function
@653 Var Var 1 Value	Var = Var 1 NAND Value ;	NAND
@654 Var Value	Var =NOT Value ;	NOT
@655 Var Var 1 Value	Var = Var 1 ORB Value ;	OR bit
@656 Var Var 1 Value	Var = Var1 XORB Value ;	EXKLUSIVE OR bit
@657 Var Var 1 Value	Var = Var1 ANDB Value ;	AND bit
@658 Var Var 1 Value	Var = Var 1 NANDB Value ;	NAND bit
@659 Var Value	Var =NOTB Value	NOT bit
@660 Var Const	CLEAR BIT ( Var . Const );	Delete bit in pattern Const=Bit no. 0 . . . 7
@661 Var Const	SET BIT ( Var . Const );	Set bit; Const=bit no.0..7
@67y Var 1 Var 2 Value		If the relation of Var 2 and Value is fulfilled, Boolean variable Var 1 is set to "1"
@706	POS MSYS;	Specification of a position in relation to the actual-value system of the machine
@710 Var 1 Var 2	Var 1 =PREP REF ( Var 2 );	Reference preparation Var 1: Outp. data from Var 1 Var 2: Inp. data from Var 2
@711 Var 1 Var 2 Var 3	Var 1 =INT SEC ( Var 2 Var 3 );	Intersection calculation Var 1: Outp. data from Var 1 Var 2: 1st contour from Var2 Var 3: Preset with 0
@713 Var	Var =PREP CYC;	Start preparation for cycles Var: Output data from Var
@714	STOP DEC;	Decoding stop; until buffer empty
@715	STOP DEC 1;	Stop of decoding until buffer is empty (applies to coordinate rotation)
@720 Var Value	Var =MEAS M Value ;	Inprocess measurement Var: Data stored from Var Value: No. of measurement input; 1 or 2

@ code	CL 800 statement	Function
@a15 Value 4 Value 5	WRT PIC ( Value 4 Value 5 );	Selection of displays and menu texts Value 4: 0=User area 1=Standard area Value 5: 1...254 Menu No.
@a1b	RECALL PIC	Return to the initial menu
@a20 ValueValue	PORT Value	Selection of interface Value: Number of interface 1or 2
@a25 Value 1	OUTP ZOA ( Value 1 );	Output zero offsets via RS232 Value 1: Chan. No. 0 . . . 3 0= ZO G54 - G57 1= Angle of rotation Channel 1 2= Angle of rotation Channel 2 3= Angle of rotation Channel 3
@a26 Value 2 Value 3 Value 4	OUTP DATA ( Value 2 Value 3 Value 4 );	Output data via RS232 C Value 2: Data type identifier 1= Main program 2= Subroutine 5= NC machine data 6= Tool offsets 8= Setting data 9= PLC machine data Value 3: Start address Value 4: End address
@a27 Value 1 Value 3 Value 4	OUTP PARA ( Value 1 Value 3 Value 4 );	Parameter output via RS232 C Value 1: Channel no. Value 2: Start address Value 3: End address
@a28 Value 2	INP ( Value 2 );	Read-in via RS232 Value 2: 0= No data type check . . . . . . 10=R parameter

@ code	CL 800 statement	Function
@a29	OUTP ETX	Output of the end-of-transmission character via RS232 C
@ccc		Back translation to the input screenform. Identifier for the beginning of the "OGM data block"

# 12 SINUMERIK 810M/820M Program Key

## 12.1 Internal G groups for @36b

Group G intern	G Functions														
	Hex Code														
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01 <sup>M</sup>	10	11	02	03	33	34	35	06	12	13	36		
1	09														
2	17 <sup>M</sup>	18	19	16											
3	40 <sup>M</sup>	41	42												
4	53														
5	54 <sup>M</sup>	55	56	57											
6	04	25	26	58	59	92	74								
7	60 <sup>M</sup>	63	64	62											
8	70	71													
9	80 <sup>M</sup>	81	82	83	84	85	86	87	88	89					
10	90 <sup>M</sup>	91	68												
11	94 <sup>M</sup>	95	96	97	98										
12	147	247	347	148	248	348	48	110	111						
13	50 <sup>M</sup>	51													

M: Initial setting of the G group

## 12.2 Program key

## 12.2 Program key

Group	EIA	ISO	Value range/code	Function and significance
	EOR	%	0 to 9999	Program no.
	mpf ... spf ... ... EOB	MPF ... SPF ... ...LF	0 to 9999 1 to 9999	Main programm Soubroutine
	o n /o /n	: N /: /N	1 to 9999 . . .	Main block Subblock Skippable main block Skippable subblock
G0	g	G	00 01 * 02 03 06 10 11 12 13 33 34 35 36	Rapid traverse, coarse exact positioning Linear interpolation Circular interpolation, clockwise Circular interpolation, counter-clockwise Spline Interpolation Polar coordinate programming, rapid traverse Polar coordinate programming, linear interpolation Polar coordinate programming, clockwise Polar coordinate programming, counter-clockwise Thread cutting, constant lead Thread cutting, linearly progressive lead Thread cutting, linearly degressive lead Tapping without compensating chuck
G1	g	G	09 #	Speed reduction, fine exact positioning
G2	g	G	16 17 * 18 19	Plane selection with freeaxis selection Plane selection X - Y (depending on machine data) Plane selection Z - X (depending on machine data) Plane selection Y - Z (depending on machine data)
G3	g	G	40 * 41 42	No tool nose radius compensation Tool nose radius compensation, counter-clockwise Tool nose radius compensation, clockwise
G4	g	G	53 #	Suppression of zero offset
G5	g	G	54 * 55 56 57	Zero offset 1 Zero offset 2 Zero offset 3 Zero offset 4
G6	g	G	04 # 1)   25 # 1) 26 # 1)  58 # 1) 59 # 1)  74 # 1)  92 1)	Dwell, duration predetermined in seconds under address X or F and revolutions under address S. The time range is between. 0.001 to 99999.999 s with X 0.001 to 99.999 s with F 0.1 to 99.9 Umdreh. with S  Minimum working area limitation Maximum working area limitation  Programmable additive zero offset 1 Programmable additive zero offset 2  Reference point approach via part program (PP)  Limitation of set spindle speed under address S
G7	g	G	60 62 63 64 *	Speed reduction, fine exact positioning Continuous path operation, block transition with speed reduction Tapping without encoder, 100% feedrate override Continuous path operation, block transition with speed reduction
G8	g	G	70 71	Inch input system Metric input system
				Reset via machine data Reset via machine data

1) No other functions may be written in this block

3) Other axes selectable (A, B, C, E, U, V, W)

# Active block-by-block, all others modal

\* Initial setting (after Reset, M02/M30, after switching on the control)

Group	EIA	ISO	Value range/code	Function and significance
G9	g	G	<b>80 *</b> 81 82 83 84 85 86 87 88 89	Delete G81 to G89 Call cycle L81 - drilling, centering axis switching Call cycle L82 - drilling, spot facing axis switching Call cycle L83 - deep-hole drilling axis switching Call cycle L84 - thread tapping with encoder axis switching Call cycle L85 - boring 1 axis switching Call cycle L86 - boring 2 axis switching Call cycle L87 - boring 3 axis switching Call cycle L88 - boring 4 axis switching Call cycle L89 - boring 5 axis switching
G10	g	G	<b>90 *</b> 91 68	Absolute dimensioning Incremental dimensioning Absolute dimensioning on shortest way (only with rotary axis)
G11	g	G	<b>94 *</b> 95 96 97 98	Feedrate under address F in mm/min or inches/min Feedrate under address F in mm/rev or inches/rev Feedrate under address F in mm/min or inches/min and constant cutting speed under address S in m/min or ft/min Cancellation G96, storing last setpoint speed of G96 Feedrate for rotary axis in rev/min
G12	g	G	110 111 147 # 1) 247 # 1) 347 # 1) 148 # 1) 248 # 1) 348 # 1) 48 # 1)	Polar coordinate programming, take setpoint position reached as new centre Polar coordinate programming, centre programming with angle and radius Soft approach to contour with linear Soft approach to contour with quarter circle Soft approach to contour with semicircle Soft leaving with linear Soft leaving with quarter circle Soft leaving with semicircle Leave as approached
G13			<b>50 *</b> 51	Cancellation of scale modification Scale modification
	x 4)	X 4)	± 0.001 to± 99999.999 ± 0.0001 to± 3999.9999 0.001 to 99999.999 ± 0.001° to± 99999.999°	Position data in mm Position data in inches Dwell in sec Position data in degrees
	y 4)	Y 4)	± 0.001 to± 99999.999 ± 0.0001 to± 3999.9999 0.001 to 99999.999 ± 0.001° to± 99999.999°	Position data in mm Position data in inches Dwell in sec. Position data in degrees
	z 4)	Z 4)	± 0.001 to± 99999.999 ± 0.0001 to± 3999.9999 ± 0.001° to± 99999.999°	Position data in mm Position data in inches Position data in degrees
	q 4)	Q 4)	± 0.001 to± 99999.999 ± 0.0001 to± 3999.9999 ± 0.001° to± 99999.999°	Auxiliary axes, position data in mm Auxiliary axes, position data in inches Position data in degrees

1) No other functions may be written in this block

3) Other axes selectable (A, B, C, E, U, V, W)

# Active block-by-block, all others modal

\* Initial setting (after Reset, M02/M30, after switching on the control)

4) The range is set via machine data. Refer to the machine manufacturer for the valid range of values.



## 12.2 Program key

Group	EIA	ISO	Value range	Function and significance
	a 3)	A 3)	0 to 359.99999°	Angle in degrees with contour definition
	c 4)	C 4)	± 0.001 to ± 99999.999 ± 0.0001 to ± 3999.9999 ± 0.001° to ± 99999.999°	Auxiliary/rotary position information in mm Auxiliary/rotary position information in inch Positon information in degrees
	d	D	1 to 99 0	Tool offset selection Tool offset cancellation
	f 4)	F 4)	0.01 to 45000 0.1 to 1770.000 0.001 to 50.000 0.0001 to 2.0000 0.001 to 16.000 0.0001 to 6.0000 0.001 to 99999.999	Feedrate in mm/min Feedrate in inches/min Feedrate in mm/rev Feedrate in inches/rev Thread lead increase or decrease in mm/rev Thread lead increase or decrease in inches/rev Dwell in sec
	h	H	1 to 9999	Auxiliary functions
	i 4)	I 4)	± 0.001 to ± 99999.999 ± 0.0001 to ± 3999.9999 0.001 to 400.000 0.0001 to 16.000	Interpolation parameters for X axis in mm Interpolation parameters for X axis in inches Thread lead in mm Thread lead in inches
	j 4)	J 4)	± 0.001 to ± 99999.999 ± 0.0001 to ± 3999.9999 0.001 to 400.000 0.0001 to 16.000	Interpolation parameters for Y axis in mm Interpolation parameters for Y axis in inches Thread lead in mm Thread lead in inches
	k 4)	K 4)	± 0.001 to ± 99999.999 ± 0.0001 to ± 3999.9999 0.001 to 400.000 0.0001 to 16.000	Interpolation parameters for Y axis in mm Interpolation parameters for Y axis in inches Thread lead in mm Thread lead in inches
	l	L	1 to 9999	Subroutine number
	p	P	1 to 99	Number of subroutine passes Ratio P = working diameter/unit diameter with cylindrical interpolation
	r	R	0 to 49 50 to 99 110 to 699 700 to 999	Transfer parameters Arithmetic parameters Channel-dependent declared parameters Central parameters
	s	S	1 to 16000 1 to 16000 0.5 to 359.5 0.1 to 99.9	Spindle speed in rev/min <sup>-1</sup> or 0.1 rev/min <sup>-1</sup> and constant cutting speed in m/min or 0.1 m/min or ft/min or 0.1 ft/min Spindle speed limitation in rev/min <sup>-1</sup> or 0.1 rev/min <sup>-1</sup> Spindle stop in degrees (M19), distance from zero mark of encoder Dwell in revolutions/min
	t	T	1 to 9999	Tool number
	u 3) 4)	U 3) 4)	± 0.001 to + 99999.999 ± 0.0001 to + 3999.9999 + 0 or - 0 - 0.001 to - 99999.999 - 0.0001 to - 3999.9999 + 0.001 to + 99999.999 + 0.0001 to + 3999.9999	Radius with circular interpolation and polar coordinates in mm Radius with circular interpolation and polar coordinates in inches Corner with contour definition Chamfer with contour definition in mm Chamfer with contour definition in inches Radius with contour definition in mm Radius with contour definition in inches

1) No other functions may be written in this block

3) Other axes selectable (A, B, C, E, U, V, W)

# Active block-by-block, all others modal

\* Initial setting (after Reset, M02/M30, after switching on the control)

4) The range is set via machine data. Refer to the machine manufacturer for the valid range of values.

Group	EIA	ISO	Code	Function and significance
M1	m	M	00 # 01	Programmed stop, unconditional Programmed stop, conditional
M2	m	M	02 17  30	Program end, in last program block Subroutine end, in last subroutine block, without stop in repeat passes Program end, in last program block
M3	m	M	03 04 05 * 19	Direction of spindle rotation clockwise Direction of spindle rotation counter-clockwise Spindle stop, non-oriented Oriented spindle stop, angle in degrees under address S
M4	m	M	36 37	Feedrate as programmed under F Feedrate in mm/min or mm/rev, 1:100 reduction ratio Also active with G33
M5	m	M	0 to 99	Miscellaneous functions, freely assignable except for groups M1 to M4
	I	L	70...	Process-oriented (flying) measuring cycle
	I	L	81 .. 89	Drilling cycles
	I	L	91/92	Retract cycles for tool change
	I	L	900	Drilling patterns
	I	L	901	"SLOT" milling pattern
	I	L	902	"ELONGATED HOLE" milling pattern
	I	L	903	Milling rectangular pocket
	I	L	930	Milling circular pocket
	I	L	999	Empty buffers
	@	@		
	=	=		Separator, mandatory in address extensions, e.g. R35 = 123.5
	+	+		Addition for parameters
	-	-		Subtraction for parameters
	*	*		Multiplication for parameters
	/	/		Division for parameters
	5-4-2 2) 7-4-2 2)	( )		Control-in Control-out
	EOB	LF		Block end

2) Punched tracks

# Active block-by-block, all others modal

\* Initial setting (after Reset, M0/M30, after switching on the control)

Siemens AG

AUT V250  
P.O. Box 31 80  
D-91050 Erlangen  
Federal Republic of Germany

**Suggestions**

**Corrections**

For Publication/Manual:  
SINUMERIK 810M, Basic Version 3  
Software Version 3  
Operating and Programming  
User's Guide

User Documentation

Order No.: 6ZB5 410-0EQ02-0BA2  
Edition: 01.93

**From:**

Name \_\_\_\_\_

Company/Dept. \_\_\_\_\_

Address \_\_\_\_\_

Telephone \_\_\_\_\_ / \_\_\_\_\_

Should you come across any printing errors when reading this publication, please notify us on this sheet. Suggestions for improvement are also welcome.

**Suggestions and/or corrections**

Siemens AG  
Automation Group  
Automation Systems  
for Machine Tools, Robots  
and Special Purpose Machines  
P.O Box 31 80, D-91050 Erlangen  
Federal Republic of Germany

Siemens Aktiengesellschaft

© Siemens AG 1990 All Rights Reserved  
Subject to change without prior notice

Order No. 6ZB5 410-0EQ02-0BA2  
Printed in the Fed. Rep. of Germany

