

SINUMERIK 805 Software Version 4

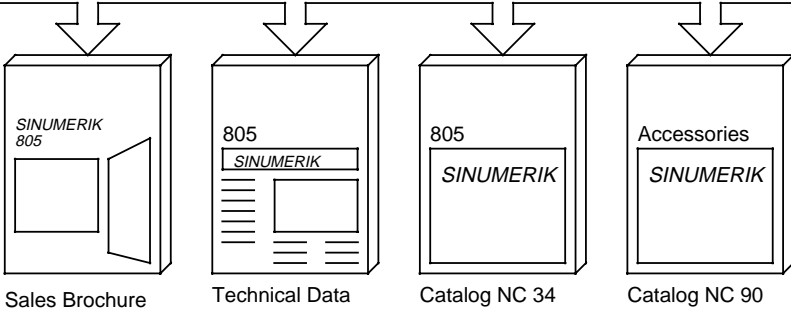
Programming Guide

05.93 Edition

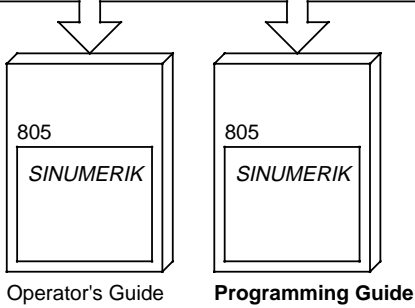
User Documentation

SINUMERIK 805

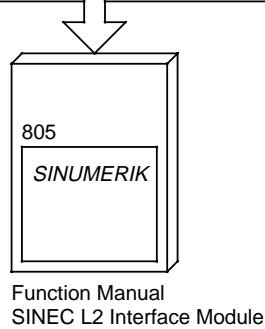
General Documentation



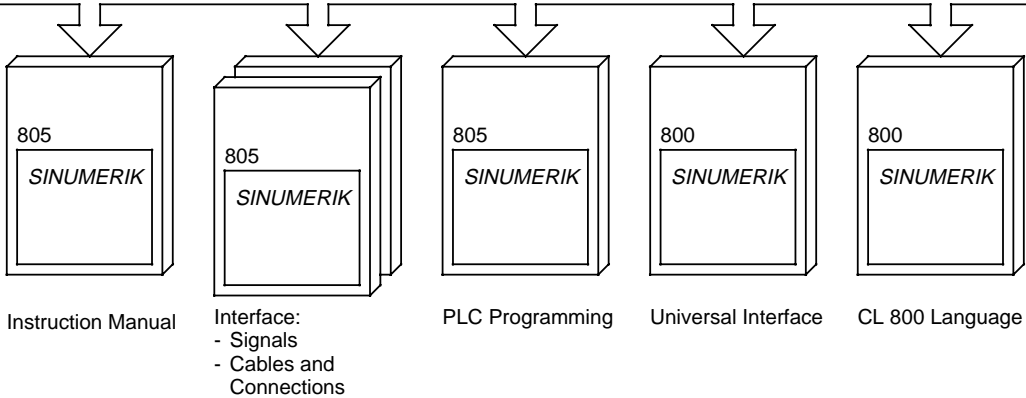
User Documentation



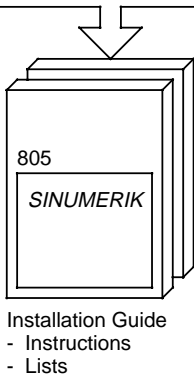
User/Manufacturer/Service Documentation



Manufacturer Documentation



Service Documentation



SINUMERIK 805 Software Version 4

Programming Guide

User Documentation

May 1993 Edition

SINUMERIK® documentation

Printing history

Brief details of this edition and previous editions are listed below.

The status of each edition is shown by the code in the "Remarks" column.

Status code in "Remarks" column:

A . . . New documentation

B . . . Unrevised reprint with new Order No.

C . . . Revised edition with new status. If factual changes have been made on the page since the last edition, this is indicated by a new edition coding in the header on that page.

Edition	Order No.	Remarks
06.90	6ZB5 410-0EK02-0BA0	A
01.91	6ZB5 410-0EK02-0BA1	C
11.91	6ZB5 410-0EK02-0BA2	C
05.93	6ZB5 410-0EK02-0BA3	C

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

This publication was produced on the Siemens 5800 Office System.
Subject to change without prior notice.

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

© Siemens AG 1990 All Rights Reserved

Preliminary Remarks

General Remarks

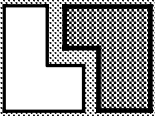
This Guide contains the information required to generate the NC part programs required by the user.

It explains how part programs are structured, their syntax and rules to be observed when programming.

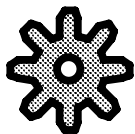
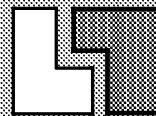
This documentation is directed at qualified technical personnel who have been specifically trained in or possess requisite knowledge of NC equipment and its operation.

Familiarity with and technically correct observation of the safety instructions and warnings are essential for safe installation and start-up as well as for safety during operation and maintenance of the product. Only qualified personnel are in a position to correctly interpret and implement the safety instructions and warnings described in general terms in the documentation.

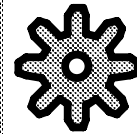
Technical Comments



A symbol like this in the documentation is a reference to an addition to the ordering data.



Certain programming commands only function in the described way when a suitable value is entered in a specific machine data.



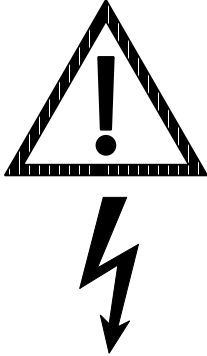
Other functions not described in this documentation might be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

This documentation is valid for software version 4.

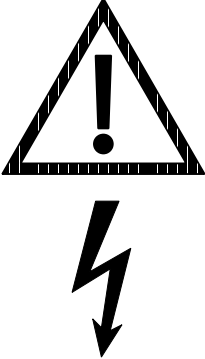
Proper usage

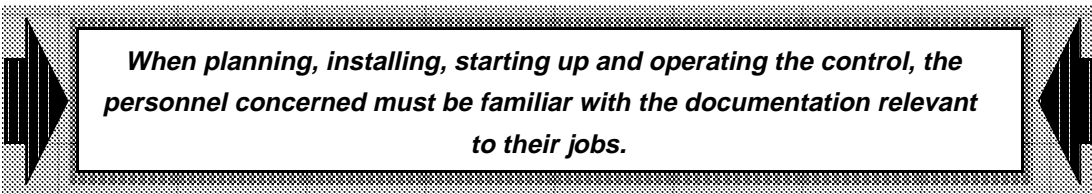
- The equipment/system or the system components may only be used for the applications described in the catalog or the technical description, and only in combination with the equipment, components and devices of other manufacturers as far as this is recommended or permitted by Siemens.
- The product described has been developed, manufactured, tested and the documentation compiled in keeping with the relevant safety standards. Consequently, if the described handling instructions and safety guidelines described for planning, installation, proper operation and maintenance are adhered to, the product, under normal conditions, will not be a source of danger to property or life.

Safety Guidelines

	WARNING
	<p>When electrical devices are in operation, certain parts of them are inevitably subjected to hazardous voltages.</p> <p>Improper interference with the device/system or failure to observe the warning advice can result in serious physical injury or material damage. Only appropriately trained personnel familiar with the assembly, installation, starting up or operation of the product are permitted to interfere with this device/system.</p>

Qualified Personnel

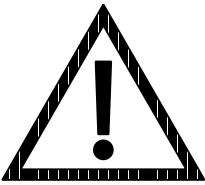
	WARNING
	<p>As far as the safety advice (contained in the documentation or as a sticker on the product) is concerned, "qualified personnel" refers to persons who, for instance:</p> <ul style="list-style-type: none">• have received training or instruction and authorization to energize and deenergize, earth and tag electric circuits and devices according to established safety practices.• have received training or instruction according to established safety practices in the care, use and repair of appropriate safety equipment.• have received training or instruction in working with electrostatically sensitive components or modules.• have been instructed as operators to work with automation technology equipment and are familiar with the contents in the Operator's and/or Programming Guide referring to operation.

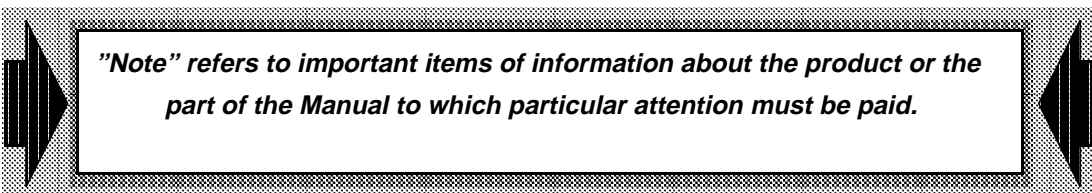


Notes on danger

The following notes are provided for your personal safety and to protect the product described here or connected devices and machines against damage.

Safety advice and warnings intended to avert danger to human life and health and to avoid material damage are highlighted in this Manual by the terms defined here. The terms have the following meanings in the context of this Manual and the remarks on the product itself:

	CAUTION
	<p>As far as this Manual and the warning advice on the products themselves are concerned, "caution" refers to instances where slight physical injury or material damage can result if proper precautions are not taken.</p>



Fundamentals of Programming	1
Directions of Movement, Dimensioning	2
Programming of Motion Blocks	3
Switching, Auxiliary and Miscellaneous Functions	4
Subroutines	5
Parameters	6
Contour Definition	7
Tool Offsets	8
Cutter Radius Compensation (CRC) Tool Nose Radius Compensation (TNRC)	9
Siemens Standard Cycles (Option)	10
Programming of Cycles	11
Extended Programming Functions	12
Program Key	13

Contents

	Page
1	Fundamentals of Programming 1-1
1.1	Program structure 1-1
1.2	Block elements 1-3
1.2.1	Main blocks and subblocks 1-3
1.2.2	Skippable blocks 1-3
1.2.3	Remarks 1-4
1.3	Word format 1-5
1.4	Character set 1-6
1.5	Input/output via V.24 (RS232C) interface 1-7
1.5.1	V.24 (RS232C) devices 1-7
1.5.2	Leader 1-7
1.5.3	Read-in stop 1-7
1.5.4	Code tables (example: punched tape) 1-8
1.5.5	Connecting external devices, settings 1-12
1.6	Program format for input and output via V.24 (RS232C) interface 1-13
1.7	Memory areas 1-16
1.8	Input/output areas 1-17
1.9	Feedrate per revolution threshold data 1-20
2	Directions of Movement, Dimensioning 2-1
2.1	Coordinate system 2-1
2.2	Position data, preparatory functions 2-2
2.3	Dimension systems: absolute position data G90, incremental position data G91 2-2
2.3.1	Linear axes 2-2
2.3.2	Rotary axes 2-4
2.4	Reference points 2-6
2.5	Zero offsets 2-8
2.6	Path calculation 2-14
2.7	Workpiece dimensioning, input system G70/G71 2-15
2.8	Mirroring 2-16
2.8.1	Mirroring one axis 2-16
2.8.2	Mirroring two axes 2-18
2.9	Programmable working area limitation G25, G26 2-19
3	Programming of Motion Blocks 3-1
3.1	Axis commands 3-1
3.1.1	Axis motion without machining (G00) 3-2
3.2	Axis motion with machining 3-2
3.2.1	Linear interpolation G01 3-3
3.2.2	Circular interpolation G02/G03 3-6
3.2.2.1	Interpolation parameters I, J, K 3-7
3.2.2.2	Radius programming 3-10

3.2.3	Helical interpolation	3-13
3.2.4	Polar coordinates G10, G11, G12, G13	3-14
3.2.5	Polar coordinates G110, G111	3-19
3.2.6	Feedrate F, G94, G95, G96, G97	3-21
3.2.7	Thread cutting G33, G34, G35	3-23
3.2.7.1	Thread with constant lead	3-24
3.2.7.2	Thread with variable lead	3-28
3.2.7.3	Infeed options	3-29
3.2.7.4	Multiple threads	3-31
3.2.8	Tapping without encoder G63	3-33
3.2.9	Exact positioning G09, G60, G00, continuous path operation G62, G64 ..	3-33
3.2.9.1	Fine and coarse exact stop tolerance ranges G09, G60, G00	3-33
3.2.9.2	Continuous path operation G62, G64	3-35
3.2.10	Dwell time G04	3-38
3.2.11	Plane selection G16, G17, G18, G19	3-38
3.2.12	Soft approach to and exit from the contour	3-44
3.2.13	Axes with standard motors	3-46
3.2.14	Linear path control / simultaneous axes	3-47
4	Switching, Auxiliary and Miscellaneous Functions	4-1
4.1	M, S, T, H	4-1
4.2	Miscellaneous function M	4-1
4.3	Spindle function S	4-3
4.4	Auxiliary function H	4-4
4.5	Tool number T	4-4
4.6	Special auxiliary functions	4-4
5	Subroutines	5-1
5.1	Application	5-1
5.2	Subroutine structure	5-1
5.3	Subroutine call	5-2
5.4	Subroutine nesting	5-3
6	Parameters	6-1
6.1	Parameter programming	6-1
6.2	Parameter definition	6-2
6.3	Parameter calculations	6-2
6.4	Parameter string	6-4
6.4.1	Indirect addressing of R parameters	6-4
6.5	Programming examples with parameters	6-5
7	Contour Definition	7-1
7.1	Blueprint programming	7-1
7.2	Contour definition programming	7-3

7.3	Operation of functions G09, F, S, T, H, M in contour definition	7-7
7.4	Chaining of blocks	7-7
7.5	Example milling machine	7-8
7.6	Example turning machine	7-9
7.7	Miscellaneous functions in chained blocks for turning and milling machines	7-11
8	Tool Offsets	8-1
8.1	Tool data	8-1
8.2	Turning machine: Tool offset without using tool nose radius compensation (TNRC)	8-9
8.3	Tool offset using tool nose radius compensation (TNRC)	8-10
8.4	Milling machine: Selection and cancellation of length compensation ...	8-11
8.5	G40, G41, G42 Intersection cutter radius compensation (CRC)	8-12
8.6	Tool length compensation (positive or negative)	8-16
8.7	Tool offsets for end mill	8-16
8.8	Tool offsets for angle cutter	8-17
8.9	Tool offset for drill	8-18
9	Cutter Radius Compensation (CRC), Tool Nose Radius Compensation (TNRC)	9-1
9.1	Selection of CRC/TNRC	9-1
9.2	CRC/TNRC in program	9-3
9.3	Cancellation of CRC/TNRC	9-6
9.4	Changing direction of compensation	9-8
9.5	Changing offset number (D..)	9-8
9.6	Changing compensation values	9-9
9.7	Repetition of selected G function (G41, G42) with same offset number	9-9
9.8	M00, M01, M02 and M30 with CRC/TNRC selected	9-10
9.9	CRC/TNRC with combination of various block types and in conjunction with contour errors	9-12
9.10	Special cases for CRC/TNRC	9-17
9.11	Effect with negative compensation values	9-20
10	Siemens Standard Cycles (Option)	10-1
10.1	Introduction	10-1
10.2	Machining cycles for drilling, boring and milling	10-4
10.2.1	Drilling and boring cycles G81 to G89	10-4
10.3	Machining cycles for turning	10-25
10.3.1	L93 Grooving cycle (precondition: Blueprint-programming)	10-25
10.3.2	L96 Stock removal cycle without relief cut elements	10-33
10.3.3	L97 Thread cutting cycle	10-37
10.3.4	L98 Deep hole boring cycle	10-44
10.4	Milling patterns	10-46
10.4.1	L903 Milling rectangular pockets	10-46

11	Programming of Cycles	11-1
11.1	Target code	11-1
11.1.1	Main groups	11-1
11.1.2	Identifiers	11-1
11.2	@ code description	11-2
11.2.1	Main group 0: general instructions for program structure	11-2
11.2.2	Main group 1: program branching	11-5
11.2.3	Main group 2: data transfer general	11-15
11.2.4	Main group 3: data transfer system memory to R parameter	11-16
11.2.5	Main group 4: data transfer R parameter to system memory	11-26
11.2.6	Main group 5: file handling general (available soon)	11-33
11.2.7	Main group 6: mathematical functions	11-33
11.2.8	Main group 7: NC specific functions	11-43
11.3	@ code table	11-46
12	Extended Programming Functions	12-1
12.1	Operator control	12-1
12.2	User image	12-4
13	Program Key	13-1
13.1	Internal breakdown of G groups with @36b	13-1
13.2	Program key SINUMERIK 805	13-2

1 Fundamentals of Programming

1.1 Program structure

The program structure is based on DIN 66025. A part program entered directly in the control comprises the following elements:

%44	L _F								Program header
N5	G90	X10	Y20	F500	L _F				} Machining blocks
N10	X20	Y0	L _F						
N15	X30	Y20	L _F						
N20	M02	L _F							Program end

Program header: % = Main program; L = Subroutine
 44 = Program number (range: 0 to 9999 for %;
 1 to 999 for L)
 L_F = End of line (line feed)

Subroutines and cycles can be components of the program. Cycles are subroutines created either by the machine manufacturer or by Siemens.

Machining blocks:

A single block contains all the data required to execute a work step.

Example:

```
N15 G91 G01 X100 F1500 LF
```

In this block the X axis is traversed by 100 mm in a positive direction at a feedrate of 1.5 m/min.

The block comprises the block number (N-), several words and the "L_F" end-of-block character.

The maximum block length is 120 characters. The entire block is displayed over several lines.

Any block number can be selected. To obtain a defined block search and defined jump functions, a block number can be used only once in a program.

A block number does not have to be programmed. In such instances, however, no block search and no block jump can be performed.

To obtain a clearly laid-out block structure, the words of a block should be arranged in the program key sequence.

Block example:

N45 G.. X.. Y.. Z.. F.. S.. T.. D.. H.. M.. LF

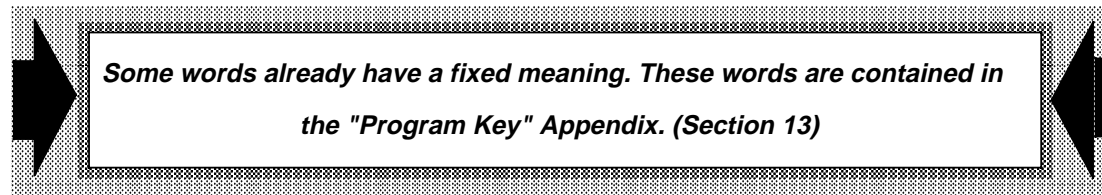
Word meanings:

- G.. This word selects the preparatory functions (e.g. valid dimension system, zero offset, traversing mode).
- X.., Y.., Z.. The desired traversing path for the axis concerned is programmed in conjunction with the axis name.
- F.. The feedrate (e.g. F500 = 0.5 m/min) valid from this block onwards is programmed under this word.
- S.. The spindle speed is programmed here.
- T.. The T number selects the tool with which the workpiece is to be machined.
- D.. The valid tool offsets are stated with the D number.
- H.. An auxiliary function specific to the manufacturer can be programmed under this word. The H words can be freely programmed by the manufacturer.
- M.. Machine functions (e.g. M06 = tool change) are programmed here.

Each block must be terminated with the "LF" end-of-block character. This character appears on the screen as the special character LF. It does not appear when the program is printed out.

Program end:

- N20 = Block number
- M02 or M30 = Program end
- LF = End of block



1.2 Block elements

1.2.1 Main blocks and subblocks

There are two types of block: 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 is identified by means of the character ":" instead of address character "N" (N denotes subblock).

Block example:

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

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

Block example:

```
N11 Z20 LF
```

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

Example:

```

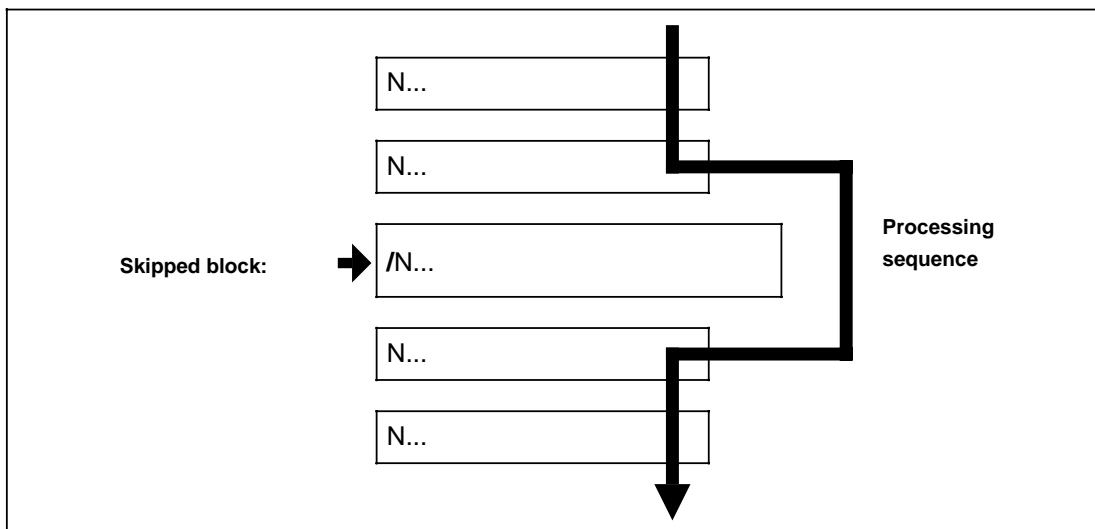
:10
N105
N110
N115

```

} Program section

1.2.2 Skippable blocks

Program blocks which must not be executed during every program run can be skipped by entering the slash character "/" at the beginning of the block. Block skip is activated in the "Program modification" softkey menu. A section can be skipped by skipping several consecutive blocks.



Block skip

If the user wants to activate the "Block skip" function in the middle of the program (e.g. after programmed stop M00), the following must be noted:
Several blocks can be buffered in order to obtain shorter block change times. When the machine stops owing to M00 (programmed stop), the next blocks will already have been input. "Block skip" acts only on blocks which have not yet been buffered. This buffering can be prevented by programming @ 714 (clear buffer) after the block containing M00.

Example:

```
N45 Z100 F500 L_F  
N50 M00 L_F           Unconditional stop  
N55 @714 L_F         Clear buffer  
/N60 X30 L_F        Skippable block  
. . .
```

1.2.3 Remarks

The blocks in a program can be explained by means of remarks. A remark also 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 ")".

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.

The remark must not contain the %, L_F, "("and")" characters.

A remark can be up to 120 characters in length. Up to 41 of these are displayed in the comments line on the screen.

```
X100      Y200 ( Pocket ) L_F  
X100+R1   Y200 ( Pocket ) L_F  
X         Address  
100      Numerical value  
R1       R parameter  
(       Start of remark, blank  
Pocket   Remark, blank  
)       End of remark
```


The remarks line written directly next to the program header without a block number is a special case. This line is displayed together with the program overview and thus contains the program name (max. number of displayed characters 49).

Example: % 100
 (Flange piece No. 37) L_F Remarks line contains program name
 N1 G0 X0 Y10 L_F
 .
 .
 M30 L_F

1.3 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 can include signs and decimal points. A sign is written between the address letters and the string of digits. A positive sign can be omitted.

Word		
Address	Numerical value	Meaning
M	03	Spindle direction of rotation clockwise
F	1000	Feedrate 1 m/min

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

Decimal point input is permissible for the following addresses:

X, Y, Z, E, A, B, C, U, W, Q, I, J, K, R, F.

Leading and trailing zeros need not be written when decimal point notation is used.

1.4 Character set

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

- DIN 66025 (ISO)
- EIA-RS 244-B

The examples used in this Programming Guide 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

Digits:

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

Hexadecimal digits with CL 800 machine code:

a, b, c, d, e, f (cf. CL 800 Planning Guide)

Printable special characters:

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

Data input

The following characters are not processed or stored:

HT = Horizontal tabulator
SP = Space
DEL = Delete
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 LF or once before LF (see data transfer parameters)

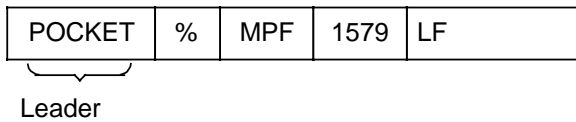
1.5 Input/output via V.24 (RS232C) interface

1.5.1 V.24 (RS232C) devices

The V.24 (RS232C) device must be matched to the control. The control can also be matched to the data input/output device by means of the parameters in the "Data transfer" softkey display.

1.5.2 Leader

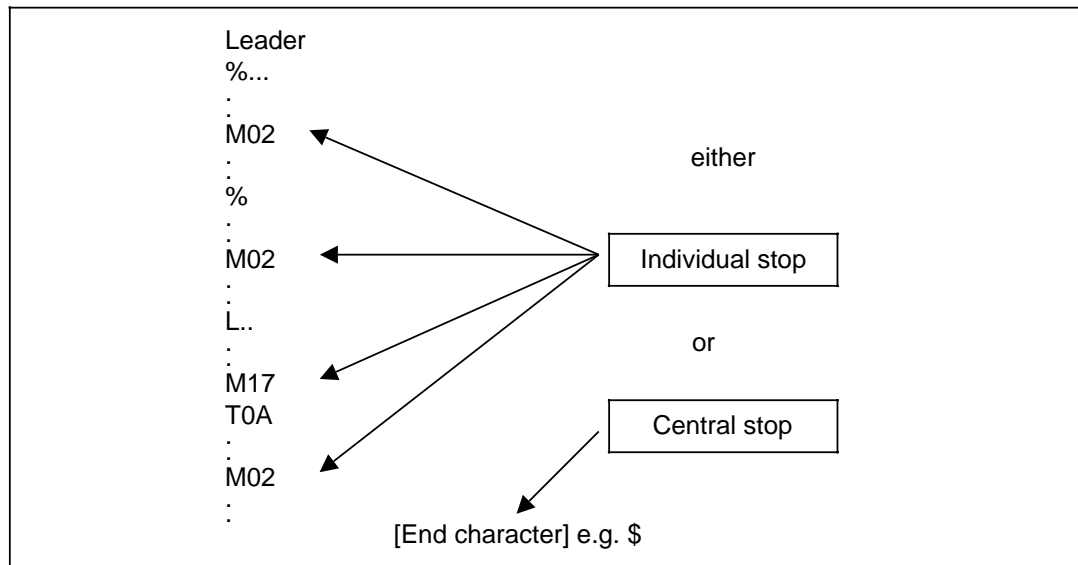
The leader is used to identify the programs. The leader can include all characters except the start-of-program character (%). The leader is not stored, and is ignored by the control during program processing.



1.5.3 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 and activated in the "data transfer, parameters" softkey display, the program or data block end (M02, M17, M30) will not stop the device while the data are being read in. The read-in process is not halted until the end-of-transmission character is reached (central stop).



1.5.4 Code tables (example: punched tape)

The data are encoded according to fixed rules whereby a bit combination (punched tape combination) corresponds to a specific character.

Two codes are used: ISO or EIA.

All characters of a code have a common identification:

ISO number of bits (holes) always even

EIA number of bits (holes) always odd.

The control automatically recognizes the correct code as soon as it reads the first % (ISO) or EOR (EIA).

The criterion of an even or odd number of bits (holes) is used for a character parity check starting with the second character of the program, with an error detection rate of 100 % for single errors. Each program must be written in one of the permissible codes.

The character parity check is triggered if programs with different codes are included in a single file (punched tape) and then transferred to the control.

As a further check, a complete program comparison is performed if a program that is already stored in the program memory is read in again.

If an error is detected, the read-in process is halted and the error is displayed on the control monitor.

Code table

ISO/DIN 66024 extended										
Character	Hole combination								Only leader and remark	
	P	7	6	5	4	T	3	2		1
NUL						.				
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				.	.			.		
!			.			.		.		x
"			.			.		.		x
'	x
\$.			.	.			x
%	
&	x
,			x
(.		.	.				
)		
*	x
+			
=	x
-			
.			
/	

Control characters are not stored

ISO/DIN 66024 extended										
Character	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	x
F		
G		
H		.			.	.				
I	
J		
K		
L		
M		
N		
O	
P		.		.		.				x
Q	
R	
S		
T		
U		
V		
W	
X				
Y		
Z			
[.	x
\			x
]	x
^	x
_		x

EIA/ 244B										
Charac- ter	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			.		.	.				
(.		
)		
ER				
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										
Charac- ter	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 can occur when comparing a program, generated in ISO code and stored in the NC, with its equivalent converted to EIA code.

The following functions are no longer executable when re-read into the SINUMERIK control:

- parameter calculation,
- extended address,
- @ commands with HEX digits (@ 36 a),
- special characters,
- comments.

1.5.5 Connecting external devices, settings

If external devices are connected to the V.24 (RS232C) interfaces of the NC, the following settings should be made at the devices:

When using ISO code (standard):

- 7 data bits, ASCII even parity
- 2 stop bits
- 1 start bit

When using EIA code:

- 7 data bits, ASCII odd parity
- 2 stop bits
- 1 start bit

The parity (even/odd) must not be selected at the NC.

Reason:

The NC always outputs 8 bits marked by either even (ISO code) or odd (EIA code) parity without parity selection.

1.6 Program format for input and output via V.24 (RS232C) interface

Main program	Leader
% MPF 1234 LF	Part program 1234 (MAIN PROGRAM FILE)
(PERFORM MEASUREM.)	Remark
N... LF	Part program
N... LF	
M02 LF	End of part program

Subroutines	Leader
% SPF 234 LF	Subroutine 234 (SUB PROGRAM FILE)
(DRILL CYCLE)... LF	Remark
N1... LF	Subroutine
N2... LF	
M17 LF	End of subroutine

%ZOA LF	Settable zero offsets (ZERO OFFSET ACTIVE)
G154 X=...Z=... LF	1st settable offset
:	
G157 X=...Z=... LF	2nd settable offset
G254 X=...Z=... LF	
:	
G257 X=...Z=... LF	Zero offset End of data block
M30 LF	

%TEA1 LF	NC machine data (TESTING DATA ACTIVE 1)
N...=... LF	Machine data
N...=... LF	
M30 LF	Machine data End of data block

%TEA2 LF	PLC machine data (TESTING DATA ACTIVE 2)
N...=... LF N...=... LF	Machine data
M30 LF	Machine data End of data block

%RPA1 LF	(R PARAMETERS ACTIVE)
R...=... LF R...=... LF	Parameter numbers with value assignments
M30 LF	R parameters End of data block

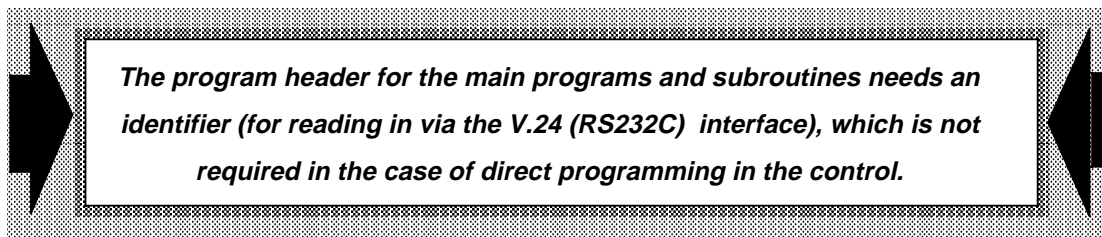
%TOA LF	Tool offsets (TOOL OFFSET ACTIVE)
D1 P0=...P1=...P7=... LF D2 P0=...P1=... LF	Tool offsets
M02 LF	Tool offsets End of data block

% SPFxxx LF xxx is determined by NC MD 30	PLC alarm texts/operational messages, user cycle alarm texts
(PLC-texts)	Remarks
N5000 (text ...) LF : N5099 (text...) LF	User cycle alarm texts (5000-5099)
N6000 (text ...) LF : N6063 (text ...) LF	PLC alarm texts (6000 - 6063)
N7000 (text ...) LF : N7063 (text ...) LF	PLC operational messages (7000 - 7063)
M17 LF	Alarm/message texts, end of text data block

Text length:
 Max. 50 ASCII
 characters
 ('CR' and 'LF'
 not allowed)

The control automatically generates the extended program header during read-out.

% SPFxxx LF xxx is determined by NC-MD 32	Operator guidance texts
(Operator prompting)	Remarks
//... LF : : LF	Start of declaration block : : End of declaration block
//... LF : : LF	Start of declaration block : : End of declaration block
: : :	
M17 LF	End of operator guidance texts



	Main program	Subroutine
Direct programming at the control	1234	123
Program input via V.24 (RS232C) interface	% MPF 1234 LF	% SPF 123 LF

1.7 Memory areas

The memory areas of the control are addressed by means of the following identifiers:

Identifier	Meaning
MPF	Part program (M ain P rogram F ile)
SPF	Subroutine (S ub P rogram F ile)
TOA	Tool offsets (T ool O ffset A ctive)
ZOA	Zero offsets (Z ero O ffset A ctive)
TEA1	NC machine data (T esting D ata A ctive)
TEA2	PLC machine data (T esting D ata A ctive)
RPA	R parameter numbers with value assignments (R Parameter A ctive)
SEA	Addresses with value assignments (S etting D ata A ctive)
CLF	Clear statement (C lear F ile)

Deleting programs:

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

DELETE PROGRAM	Leader
%CLF LF	Delete program identifier (CLEAR FILE) By means of setting data (SD for serial interface) it is possible to define whether automatic reorganization of the part program memory is to be prevented with % CLF.
MPF 1234 LF	Delete part program %1234
MPF 1, 1200 LF	Delete part program %1 to %1200
MPF 0, 9999 LF	Delete all part programs
SPF 10 LF	Delete subroutine L10
SPF 11, 79 LF	Delete subroutines L11 to L79
SPF 1, 999 LF	Delete all subroutines
M30, M02 or M17 LF	End identifier M30 or M02

1.8 Input/output formats

Input resolution: 0.01 mm
0.001 inch

Position control resolution: 0.005 mm
0.0005 inch

Function Addresses		Metric		Inch		Degrees	
		Range	Unit	Range	Unit	Range	Unit
Positional data (linear axes) Interpolation parameter		±0.01 to 99999.99	mm	±0.001 to 9999.999	inch	-	degrees
Positional data with G91 (rotary axes)		-		-		0.001 to 99999.999	
Positional data with 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 ²⁾		0.01 to 4000.00		0.001 to 160.000		-	
Spindle speed S ²⁾ (significance defined via startup setting)		1 to 12000	1 rev/min	1 to 12000	1 rev/min		
		0.1 to 1200.0	0.1 rev/min	0.1 to 1200.0	0.1 rev/min		
Linear feedrate (F) (G94) ²⁾		0.1 to 450000 600000 ³⁾	mm/min	0.01 to 17700.00 23600.00 ³⁾	inch/min	1 to 45000 60000 ³⁾	degrees/ min
Feedrate per revolution (F) (G95) ^{1) 2)}		0.01 to 500.00	mm/rev	0.001 to 20.000	inch/rev		
Constant cutting speed (S)(G96) ²⁾		1 to 1200.0	m/min	1 to 12000	ft/min		
		0.1 to 1200.0		0.1 to 1200.0			
Tool offset	Length	±0.01 to 99999.99	mm	±0.001 to 999.999	inch		
	Radius	±0.01 to 9999.99		±0.001 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	rev	0.1 to 99.9	rev		
Angle with polar coordinates		-		-		0 to 359.99999	degrees
Angle with oriented spindle stop (M19)						0.1 to 359.9	degrees
R parameter		Dimension depending on units (internal floating comma) all combinations					

1) The max. speed with linear feedrate (G94) may not be exceeded.

2) Determined by machine data.

3) As from software version 4.2

1.8 Input/output formats

Input resolution: 0.001 mm
0.0001 inch

Position control resolution: 0.0005 mm
0.00005 inch

Function Addresses		Metric		Inch		Degrees	
		Range	Unit	Range	Unit	Range	Unit
Positional data (linear axes) Interpolation parameter		±0.001 to 99999.999	mm	±0.0001 to 9999.9999	inch	-	degrees
Positional data (linear axes) Interpolation parameter		-		-		0.001 to 99999.999	
Positional data with 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 2)		0.001 to 400.000		0.0001 to 16.0000		-	
Spindle speed S 2) significance defined via startup setting)		1 to 12000	1 rev/min	1 to 12000	1 rev/min		
		0.1 to 1200.0	0.1 rev/min	0.1 to 1200.0	0.1 rev/min		
Linear feedrate (F) (G94) 2)		0.01 to 45000 60000 ³⁾	mm/min	0.001 to 1770.000 2360.000 ³⁾	inch/min	1 to 45000 60000 ³⁾	degrees/ min
Feedrate per revolution (F) (G95) 1) 2)		0.001 to 50.000	mm/rev	0.0001 to 2.0000	inch/rev		
Constant cutting speed G96 (S) 2)		1 to 12000	m/min	1 to 12000	ft/min		
		0.1 to 1200.0		0.1 to 1200.0			
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	sec	0.01 to 99999.999	sec		
	F	0.01 to 99999.999		0.01 to 99999.999			
	S	0.1 to 99.9	rev	0.1 to 99.9	rev		
Angle with polar coordinates		-		-		0 to 359.99999	degrees
Angle with oriented spindle stop (M19)						0.1 to 359.9	degrees
R parameter		Dimension depending on units (internal floating comma) all combinations					

1) The max. speed with linear feedrate (G94) may not be exceeded.
2) Determined by machine data.
3) As from software version 4.2

Input resolution: 0.0001 mm
0.00001 inch

Position control resolution: 0.00005 mm
0.000005 inch

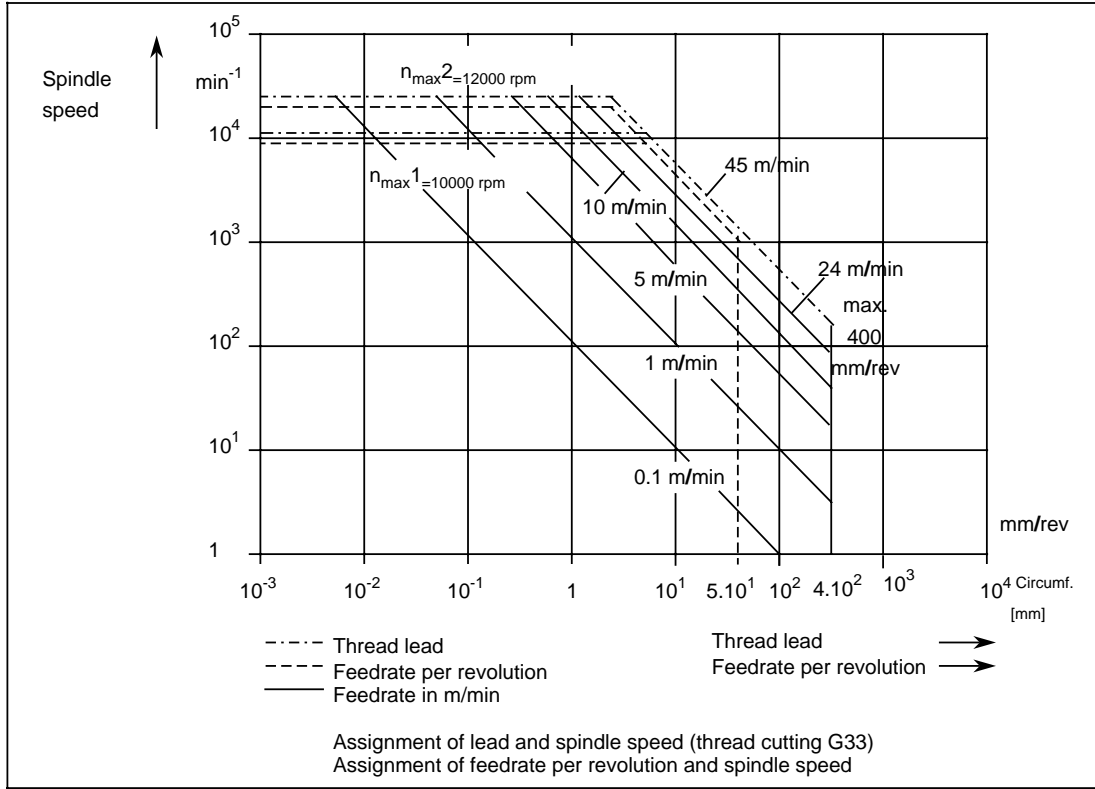
Function Addresses		Metric		Inch		Degrees		
		Range	Unit	Range	Unit	Range	Unit	
Positional data (linear axes) Interpolation parameter		±0.0001 to 9999.9999	mm	0.00001 to 999.99999	inch	-	degrees	
Positional data with G91 (rotary axes)		-		-		0.001 to 99999.999		
Positional data with G90 (rotary axes)		-		-		±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 ²⁾		0.0001 to 40.0000		0.00001 to 1.60000		-		
Spindle speed S ²⁾ (significance defined via startup setting)		1 to 12000	1 rev/min	1 to 12000	1 rev/min			
		0.1 to 1200.0	0.1 rev/min	0.1 to 1200.0	0.1 rev/min			
Linear feedrate (F) (G94) ²⁾		0.001 to 4500.000 6000.000 ³⁾	mm/min	0.0001 to 177.0000 236.0000 ³⁾	inch/min	1 to 45000 60000 ³⁾	degrees/ min	
Feedrate per revolution (F) (G95) ^{1) 2)}		0.0001 to 5.0000	mm/rev	0.00001 to 0.20000	inch/rev			
Constant cutting speed.G96 (S) ²⁾		1 to 12000	m/min	1 to 12000	ft/min			
		0.1 to 1200.0		0.1 to 1200.0				
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	sec	0.01 to 99999.999	sec			
	F	0.01 to 99999.999		0.01 to 99999.999				
	S	0.1 to 99.9	rev	0.1 to 99.9	rev			
Angle with polar coordinates		-		-		0 to 359.99999	degrees	
Angle with oriented spindle stop (M19)						0.1 to 359.9	degrees	
R parameter		Dimension depending on units (internal floating point) all combinations						

1) The max. speed with linear feedrate (G94) may not be exceeded.

2) Determined by machine data.

3) As from software version 4.2.

1.9 Feedrate per revolution threshold data



n_{max1} obtainable with encoder with 1024 pulses/revolution
 n_{max2} obtainable with encoder with 512 pulses/revolution

2 Directions of Movement, Dimensioning

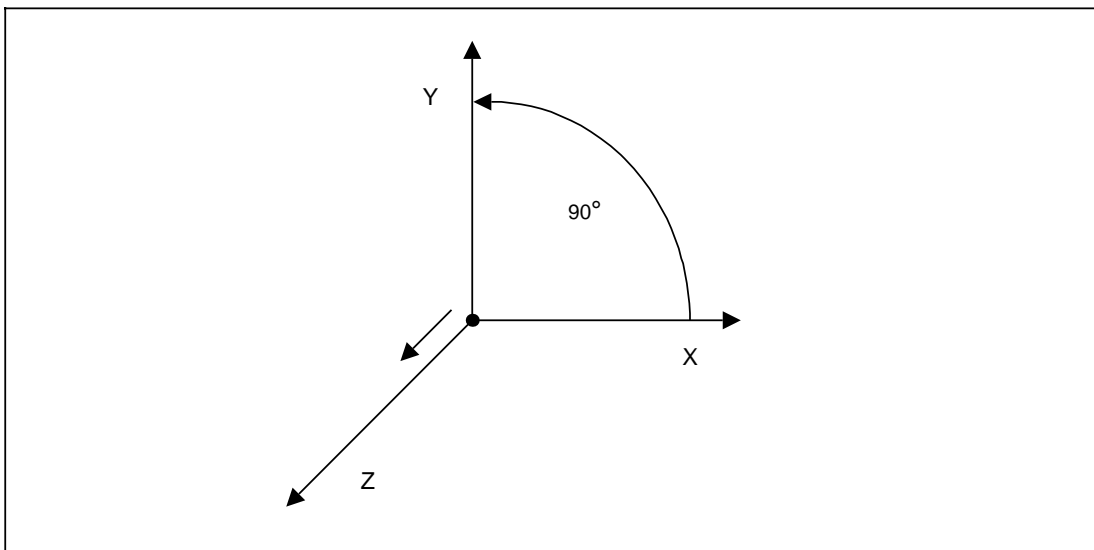
2.1 Coordinate system

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 rectangular, and has X, Y and Z axes. The system is based on the main axes of the machine.

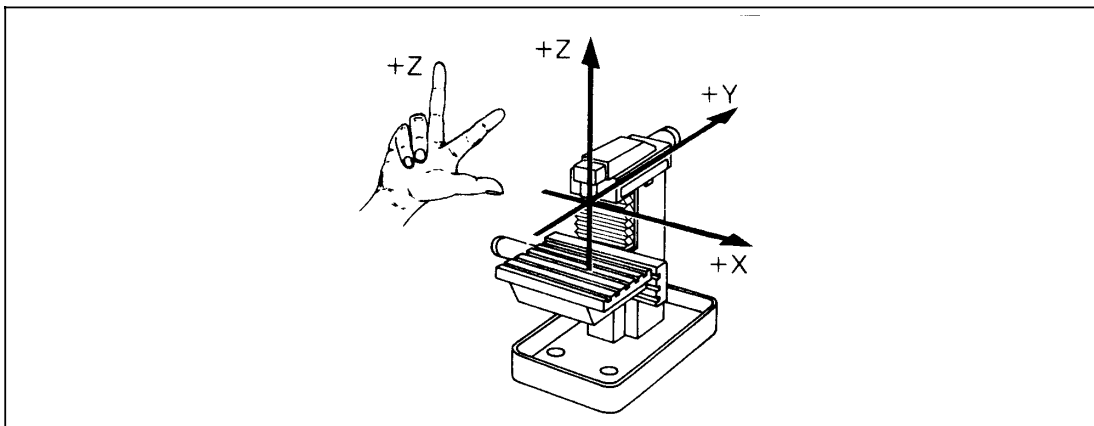
The coordinate system is defined as follows:

- The second axis is perpendicular to the first axis.
- If the first axis rotates via the shortest path (90°) towards the second axis, a right-hand threaded screw connected to it will move towards the third axis.



Clockwise coordinate system

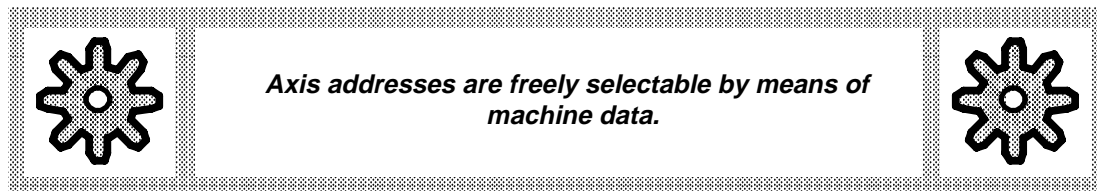
Example:



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

The default assignment for the SINUMERIK 805 is:

Main axes	X, Y, Z	(M selection)
	X, Z	(T selection)



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 movements, the type of interpolation and the method of dimensioning.

The G functions are divided into groups (see section 13 Program Key). A program block may only contain one function from each group. The G functions are either modal (stored) or effective for a single block.

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, and after reset or program end. They need not be programmed.

2.3 Dimension systems: absolute position data G90, incremental position data G91

2.3.1 Linear axes

The traversing movement to a particular point in the coordinate system can be described by means of absolute or incremental position data.

Absolute position data input G90

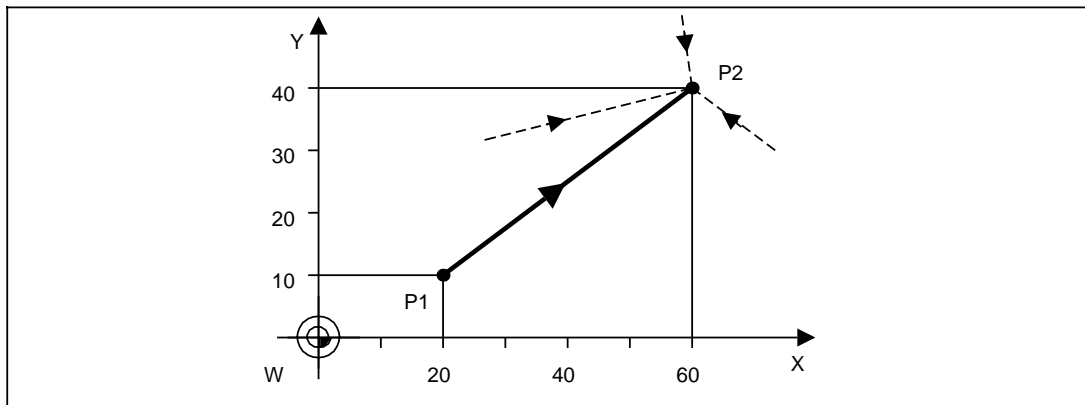
If absolute position data input is selected, all dimensional inputs refer to a fixed zero, which is normally the workpiece zero. The numerical value of the associated position data specifies the target position in the coordinate system.

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.

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 allowed for both in absolute and incremental programming.

Example: Absolute and incremental position data input



Absolute position data input -----

```
N5 G00 G90 X60 Y40 LF
```

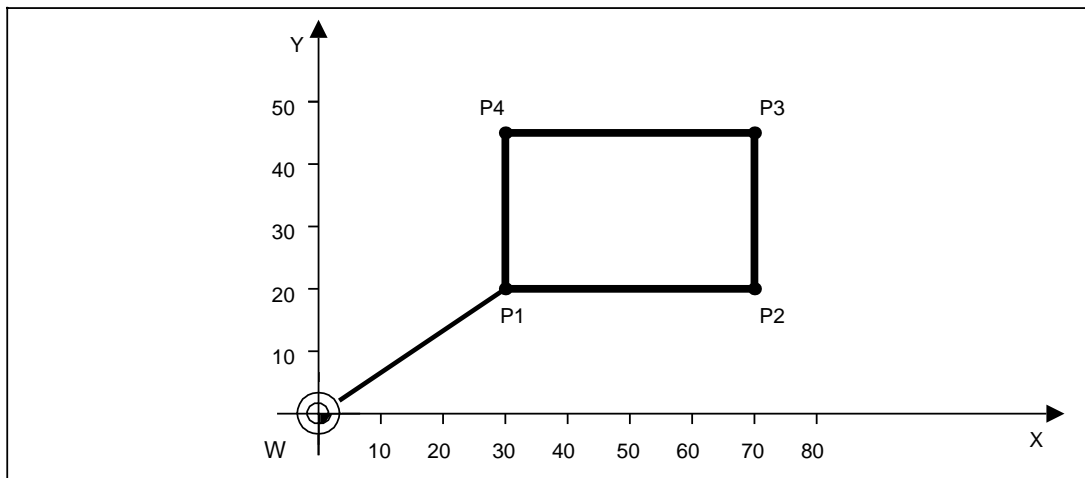
The tool moves from any position to P2.

Incremental position data input -----

```
N5 G00 G91 X40 Y30 LF
```

The tool moves from P1 to P2.

Example: Programming with absolute position and incremental position data

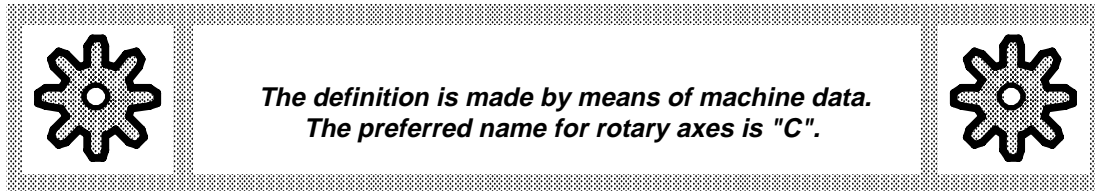


```
%2
N5 G00 G90 G94 X30 Y20 LF
N10 G01 G91 X40 F100 LF
N15 Y25 LF
N20 X-40 LF
N25 Y-25 LF
N30 M02 LF
```

Main program No. 2
Rapid traverse to P1 with absolute dimension
With progr. feedrate to P2 with incremental dimension
With progr. feedrate to P3 with incremental dimension
With progr. feedrate to P4 with incremental dimension
With progr. feedrate to P1 with incremental dimension
End of program

2.3.2 Rotary axes

Any axis can be declared a rotary axis.



- The traversing range with absolute dimension programming (G90) is ± 360.000 degrees.

The sign indicates the traversing direction for positioning to the absolute position within one revolution.

With absolute dimension programming with the shortest path (G68), the traversing range is 0 to + 360.000 degrees. With reference to the current position of the rotary axis, the control ascertains the shortest path to the programmed position and thus automatically determines the traversing direction.

When the rotary axis is programmed with "G90" for the first time in the part program, "G68" is automatically activated for this first block. This automatic generation can be deselected with "G91 C0 LF".

- The traversing range with incremental dimension programming (G91) per block is ± 99999.99 degrees. The sign indicates the traversing direction.

Several axes can be declared rotary axes **at the same time**.

Rotary axes can generally **rotate endlessly**.

Program extract:

·
·
·

```
N5 G91 C 99999.99 LF
```

Positioning rotary axes by the shortest path G68

Where G68 is specified in a traversing block, the rotary axis programmed in the block is traversed to the programmed position by the shortest path.

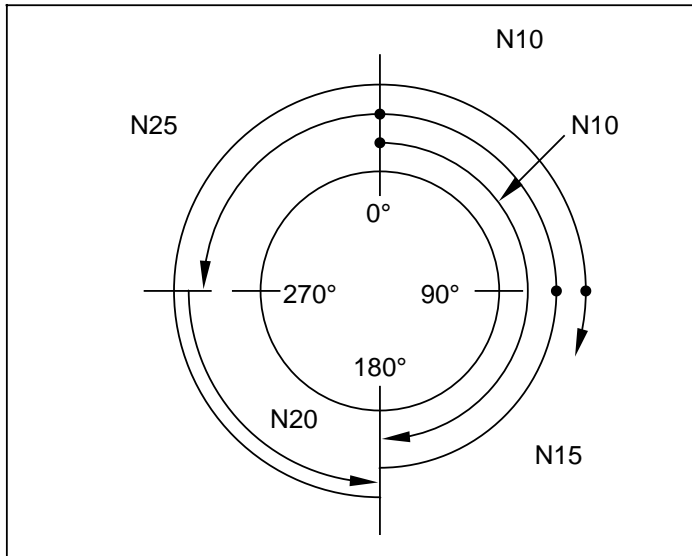
G68 is modal and like G90, selects the absolute dimension system.

Example:

```

N5   G0   C0
N10  G90  C180
N15  G90  C-270
N20  G68  C180
N25  C130

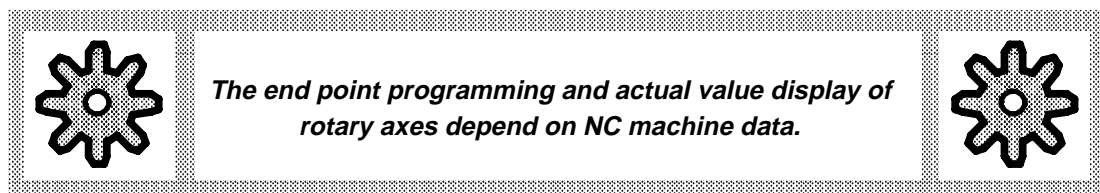
```

**Preconditions:**

```

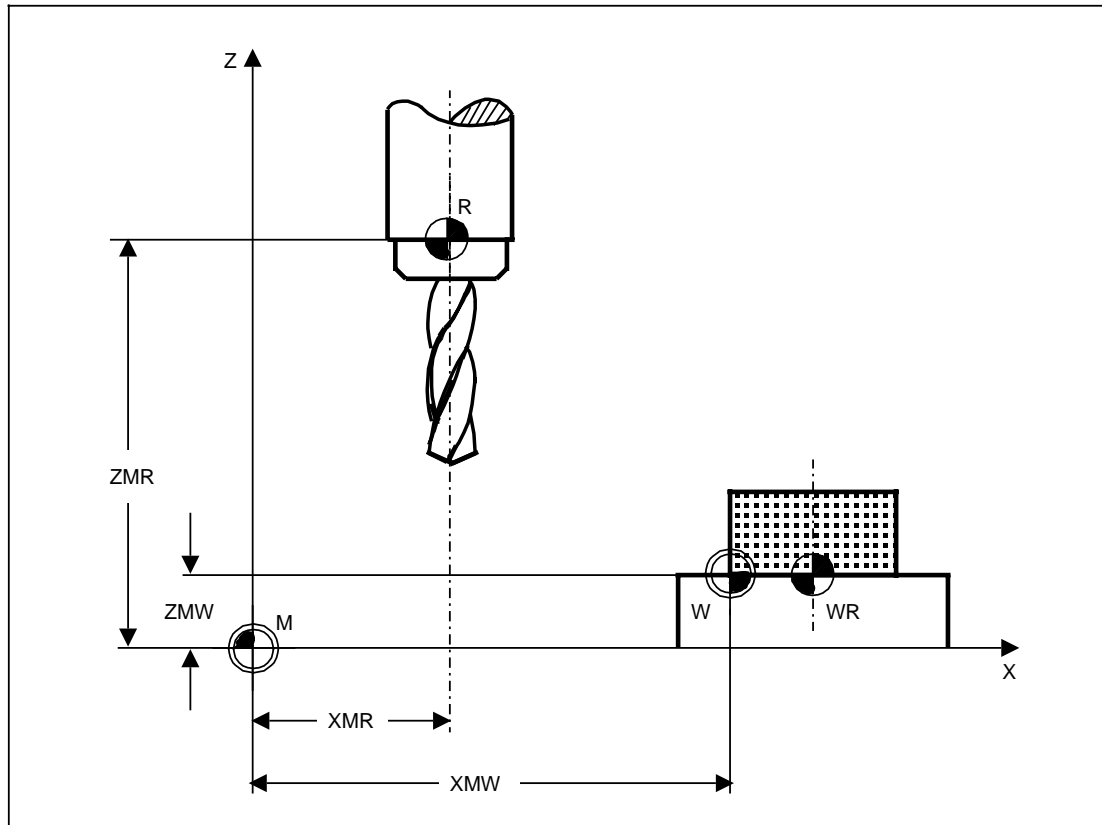
NC MD 564x  10100xxx
NC MD 568x  00000101
NC MD 572x  00000100

```



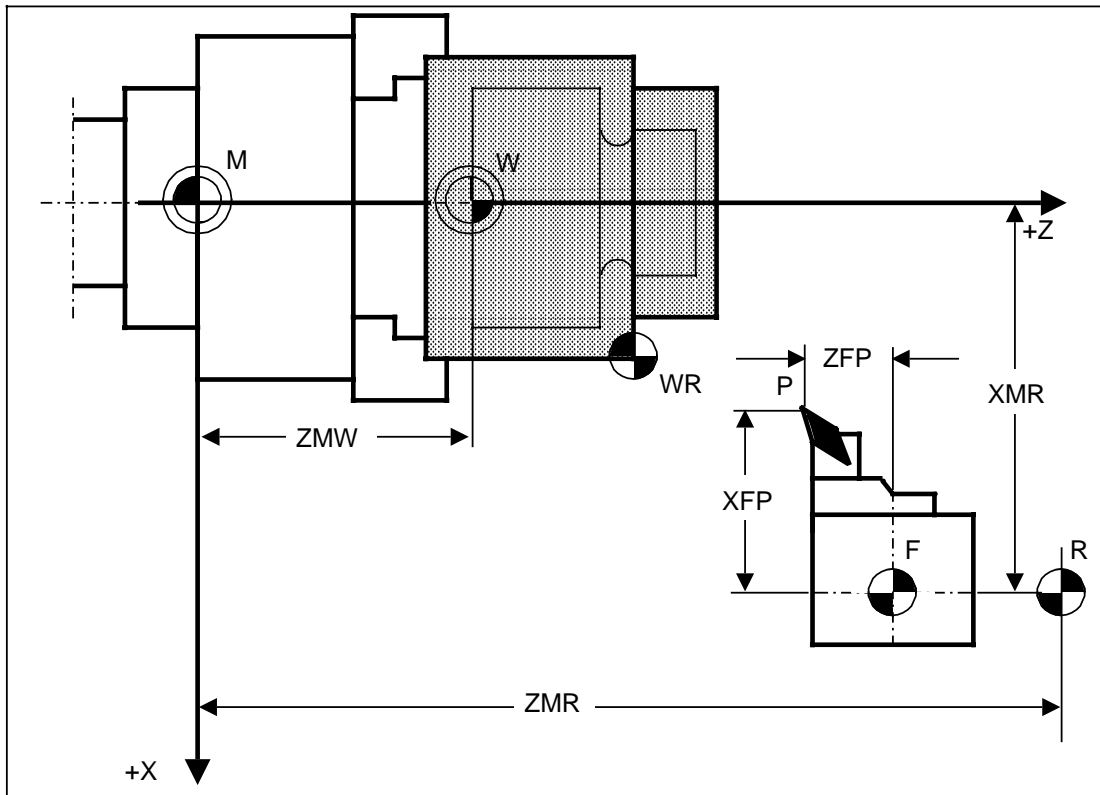
2.4 Reference points

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



Drilling and milling machine

R	Machine reference point
M	Machine zero
W	Workpiece zero
WR	Workpiece reference point
XMR, ZMR	Reference point coordinates
XMW, ZMW	Sum of zero offsets



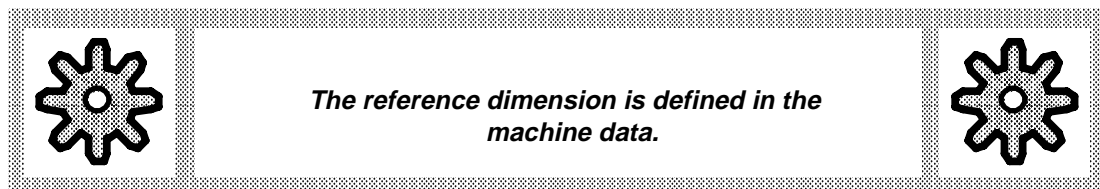
Turning machine (machining in front of the centre of rotation)

P	Tool setting point
M	Machine zero
W	Workpiece zero
R	Machine reference point
F	Slide reference point
WR	Workpiece reference point
XMR, ZMR	Reference point coordinates
XMW, ZMW	Sum of zero offsets per axis
XFP, ZFP	Tool geometry (L1, L2)

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 relation to the machine origin is defined by the zero offset.

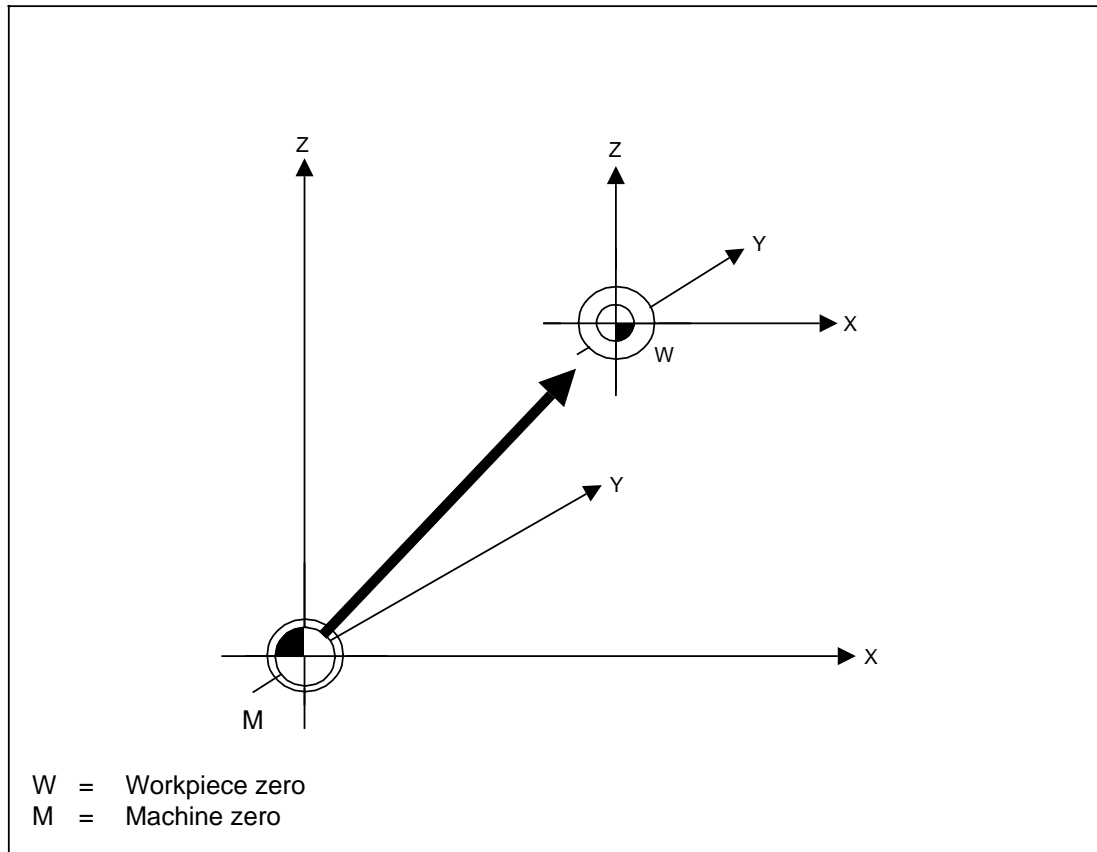
The machine reference point R is a point defined by the machine manufacturer which is approached when the control is powered up and which synchronizes the system.



P is the reference point for setting the tool.

2.5 Zero offset

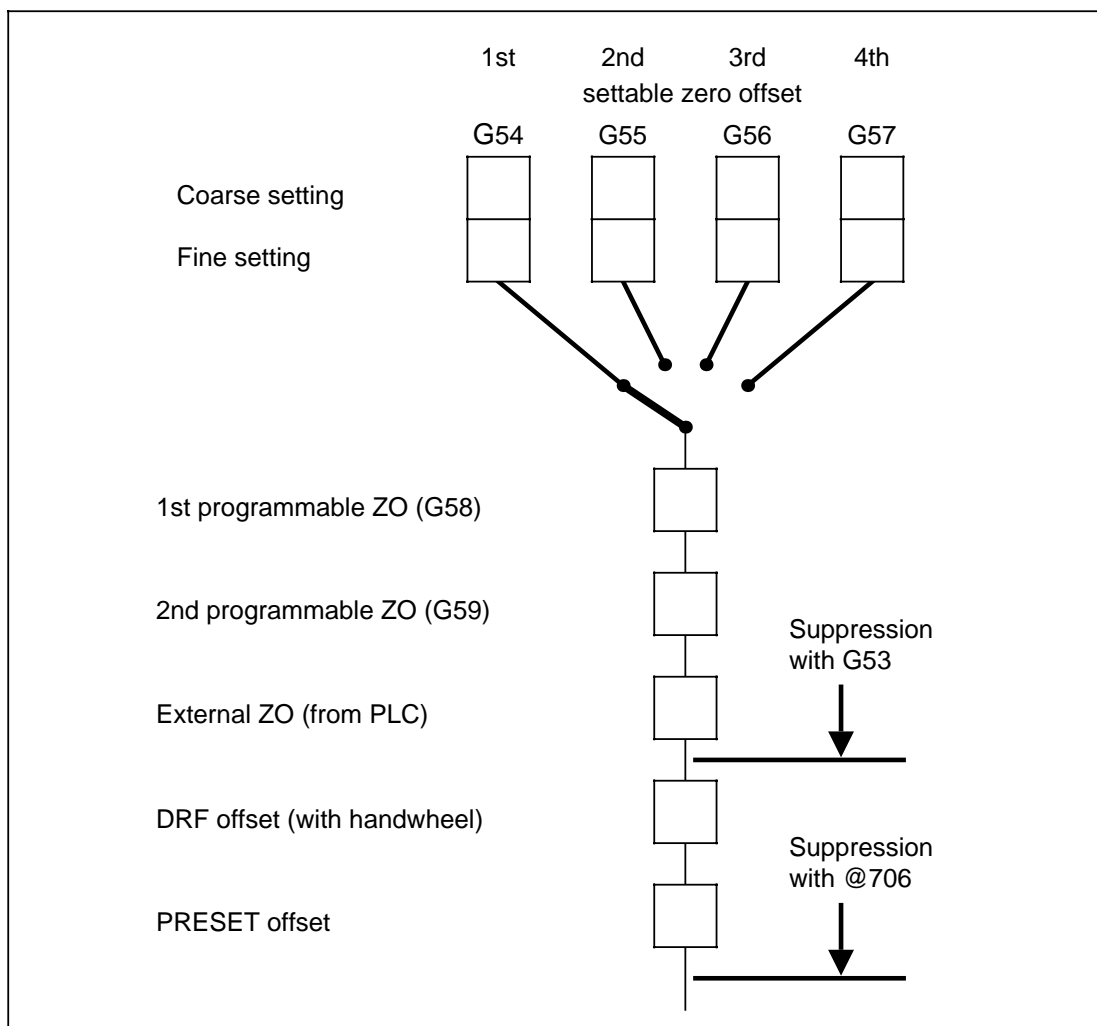
Zero offset is the distance between the workpiece zero (on which the dimensions are based) and the machine zero.



Zero offset

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).



Sum of zero offsets

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

Settable zero offset G54, G55, G56, G57

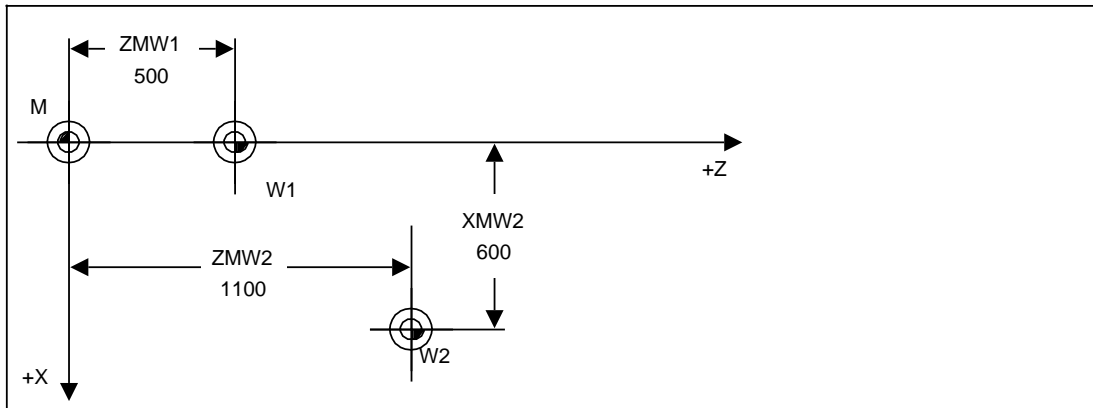
The settable zero offset values for each axis can be entered in the control via the operator panel or via the universal interface.

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

With G54 to G57, one of the four settable zero offsets can be selected for each axis. The settable ZOs are divided into coarse and fine ZOs which act additively.

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

Reading in the settable zero offsets via the V.24 (RS232C) interface:



Settable zero offset

"ZO coarse" and "ZO fine" settings are available in the control. In order for the control to recognize which values are "ZO coarse" and which "ZO fine", the appropriate format must be used:

- e.g.: G54 coarse has the format G154
- G54 fine has the format G254

Format via the interface:

```

% Z0A LF

G154 X = 250      Y = 280.1   LF
G155 X = 220.34  Y = 250.125 LF
.
.
} settable ZO coarse

G157 X = 320      Y = 350     LF
G254 X = 0.1      Y = 0.3     LF
.
.
} settable ZO fine

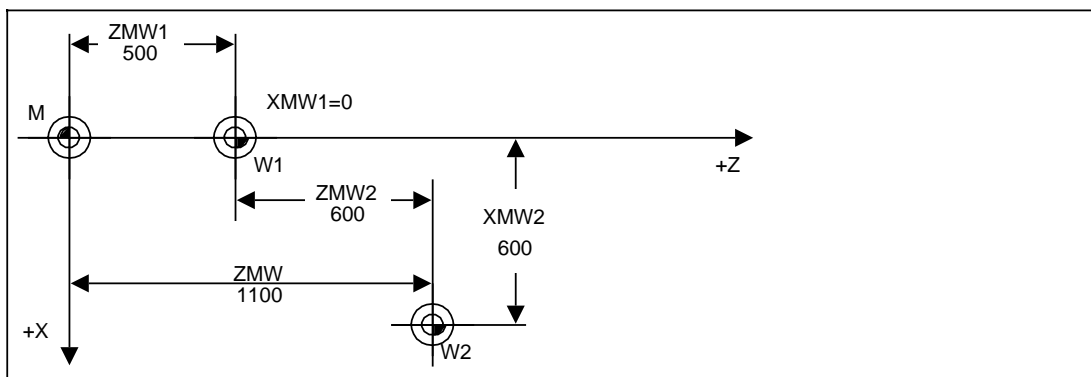
G257 X = 0.1      Y = 0.5     LF

M02 LF
    
```

If data and punched tape already exist with a format without any identification of the "ZO coarse" and "ZO fine" setting, e.g. G54 X = 250 L_F, this can also be read in via the V.24 (RS232C) interface, whereby the values are entered in the settable ZO coarse.

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 and programmable zero offsets

Settable zero offset (coarse and fine)

Input values XMW1, ZMW1

Programmable zero offset

Input values XMW2, ZMW2

Total effective zero offset

$XMW = XMW1 + XMW2$

$ZMW = ZMW1 + ZMW2$

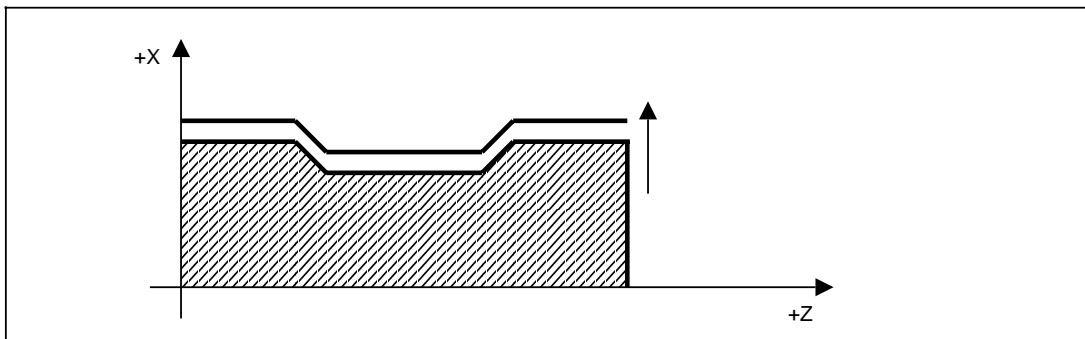
Programming:

```
N30 ...
N35 G54 L_F
N40 G59 X600 Z600 L_F
N45 ...
```

A block containing G58 or G59 must not include any functions other than the zero offsets. Up to 4 axes can be written in a block with G58/G59.

Application example with G59:

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



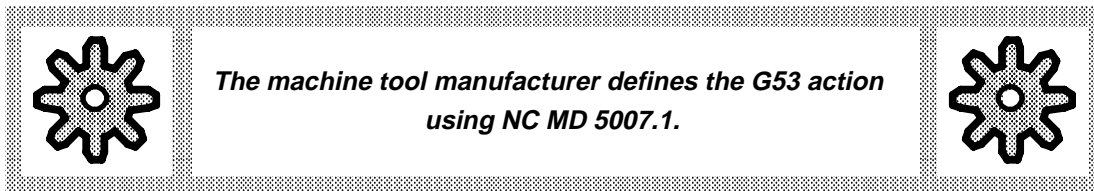
Zero offset with G59

Select: N.. G59 X... L_F

Cancel: N.. G59 X0 L_F

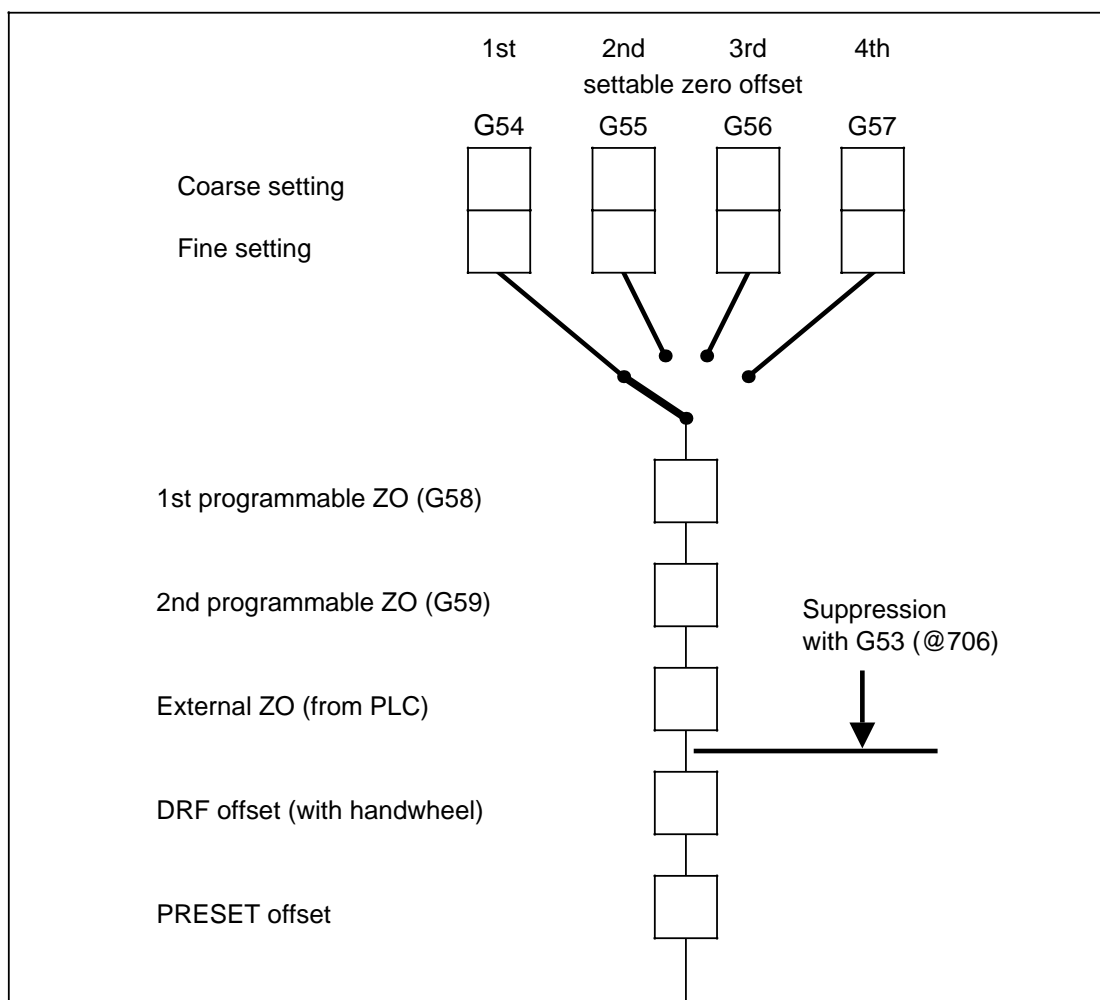
The programmable zero offset values set in this program are automatically deleted each time the program is terminated with M02 or M30 or aborted. All programmable ZOs are deleted with RESET.

G53 Cancelling zero offsets



With G53 the following zero offsets are cancelled block-by-block:

- Action a (MD5007.1=0)
 - Block-by-block cancellation of:
 - settable zero offset coarse and fine (G54 to G57)
 - programmable zero offset (G58 and G59)
 - external zero offset (from PLC)

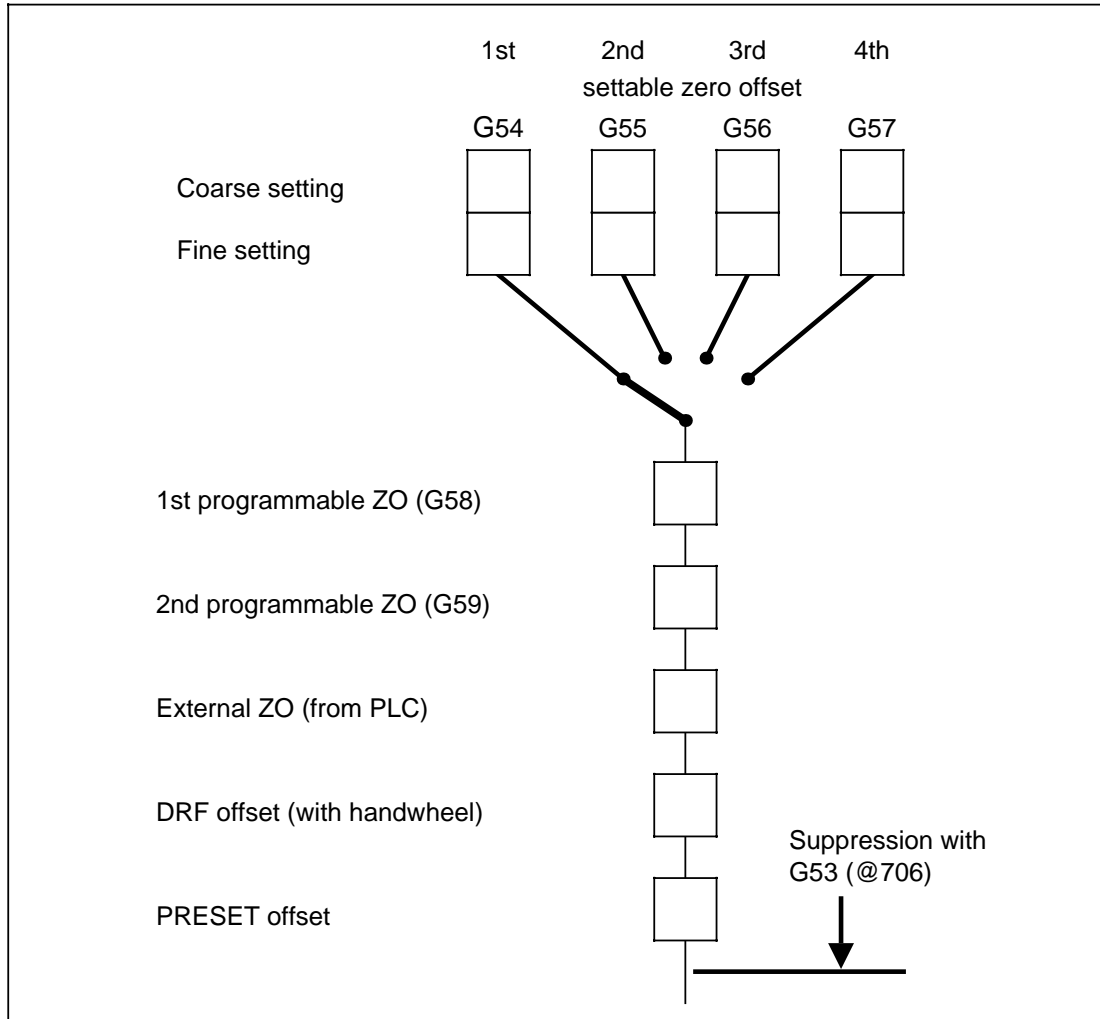


Action a: suppression of zero offset

- Action b (MD5007.1=1)

Block-by-block cancellation of:

- settable zero offset coarse and fine (G54 to G57)
- programmable zero offset (G58 and G59)
- external zero offset (from PLC)
- DRF offset
- PRESET offset



Action b: suppression of zero offset

Reference to machine zero

N30 D0 L_F

Cancellation of tool offset

N35 G53 X... Y... L_F

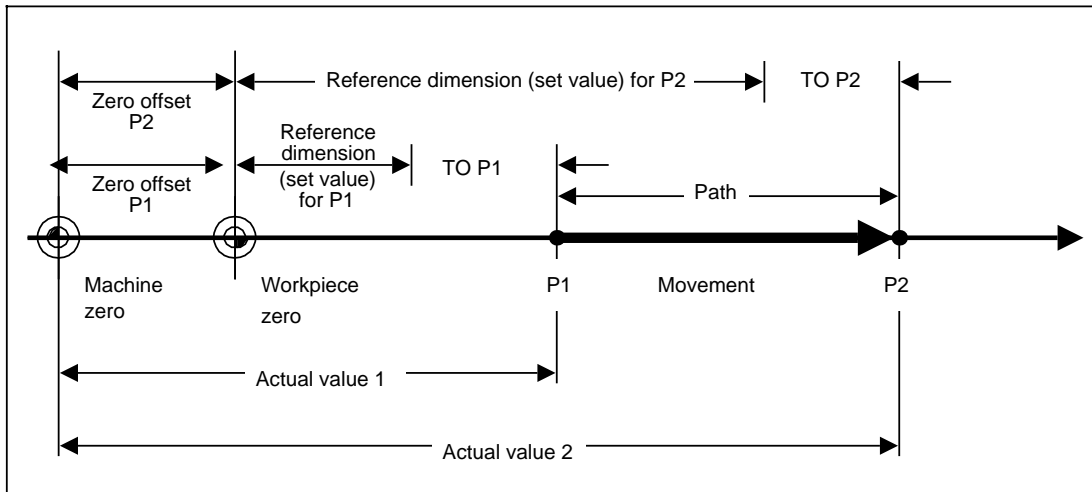
Cancellation of all zero offsets and travel to position in machine system

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:

Path = setpoint - actual value + zero offset (ZO) + tool offset (TO).



Path calculation using absolute position data input

If blocks with **incremental position data** input are used, the zero offset is incorporated normally in the first block only.

Path = incremental position data + ZO + TO

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**

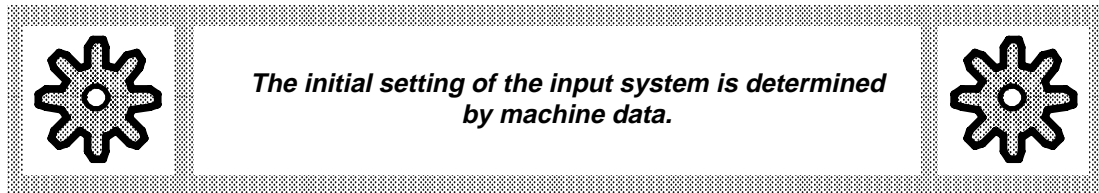
Path = absolute position data P2 - absolute position data P1 + ZOP2 - ZOP1 + TOP2 - TOP1

With **incremental position data input**

Path = incremental position data + ZOP2 - ZOP1 + TOP2 - TOP1

2.7 Workpiece dimensioning, input system G70/G71

The units of measurement can be entered in either mm or inches when programming.



The input system can be changed by means of the preparatory function G70 or G71:

G70 input system: inches
G71 input system: metric

The control 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 system of the initial setting.

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

The unit of measurement differing 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 after the last block (the initial setting comes automatically after M02, M30 end of program).

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

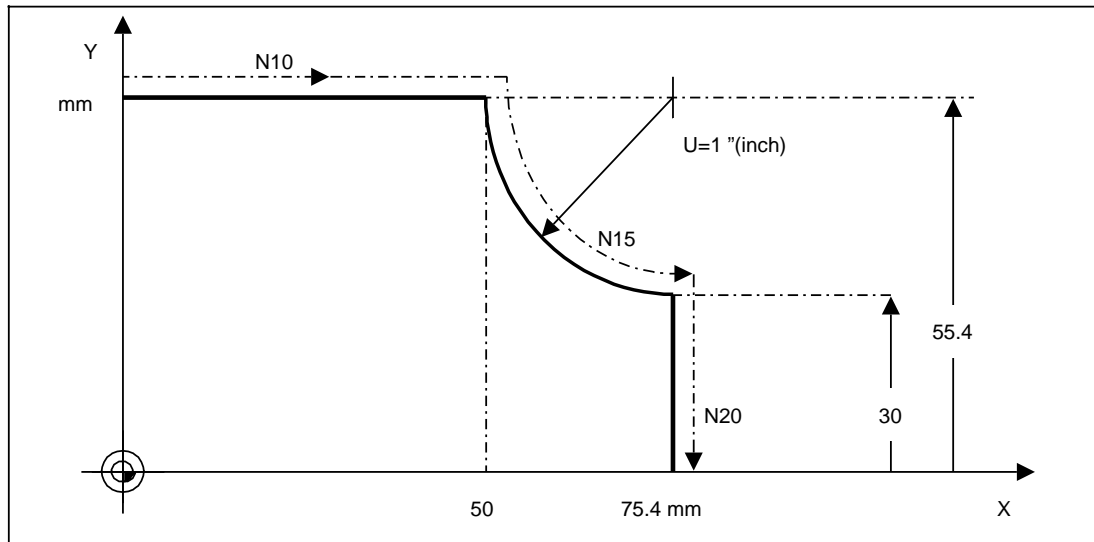
- Actual-value display (including setpoint/actual value difference)
- Zero offset
- Feedrate/cutting speed G94/G95
- Tool offset

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 with initial setting G71 (metric)

```

N09 ...
N10 G91 X50 LF
N11 G03 G70 X1 Y-1 I1 LF
N12 G01 G71 Y-30 LF
N13 ...
    
```

2.8 Mirroring

2.8.1 Mirroring one axis

By mirroring one coordinate axis, a contour can be machined:

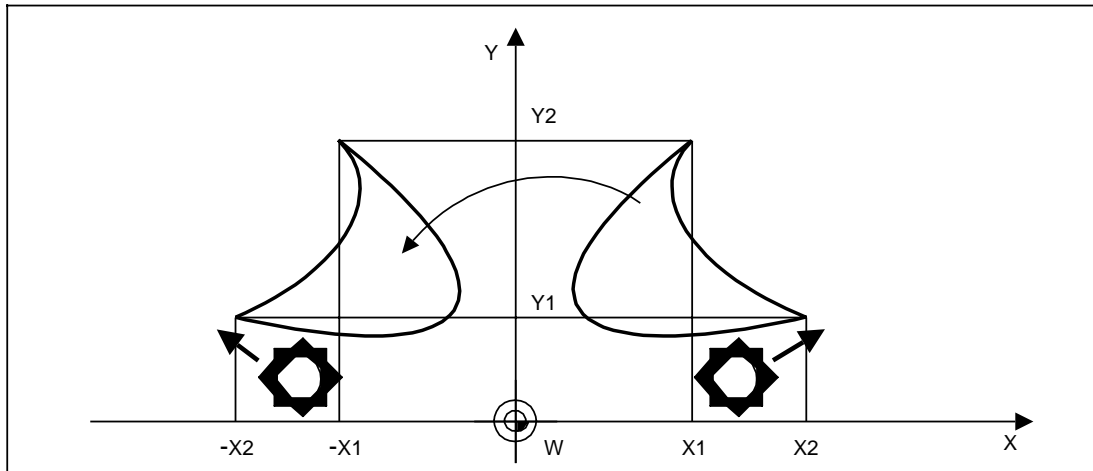
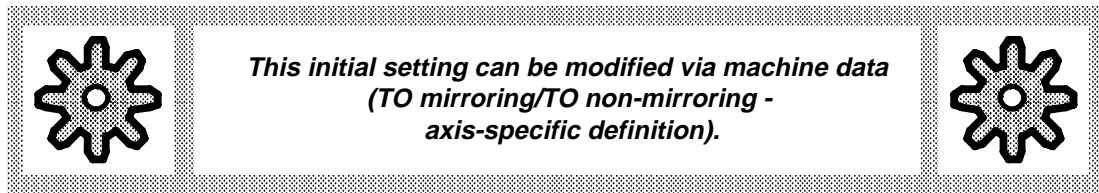
- with the same dimensions,
- at the same distance from the other axes,
- on the other side of the mirror axis and as a mirror image.

When mirroring an axis the control changes

- 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:

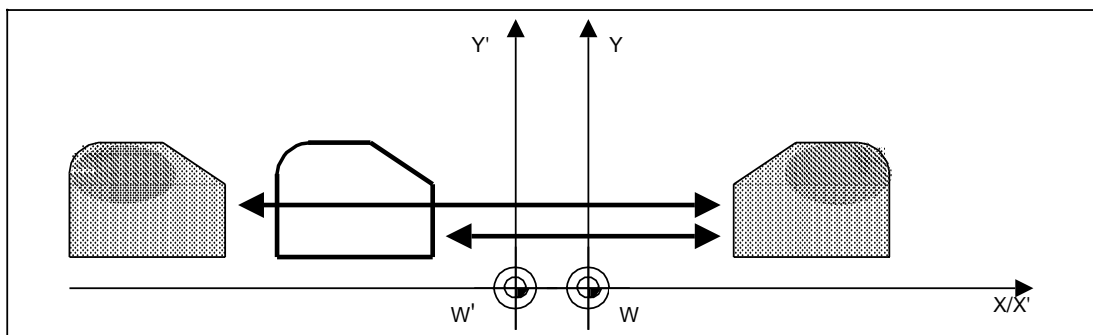
- tool length offsets
- zero offsets



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 must 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').



Offsetting workpiece zero

2.8.2 Mirroring two axes

The mirroring procedure described above must be performed twice, e.g.:

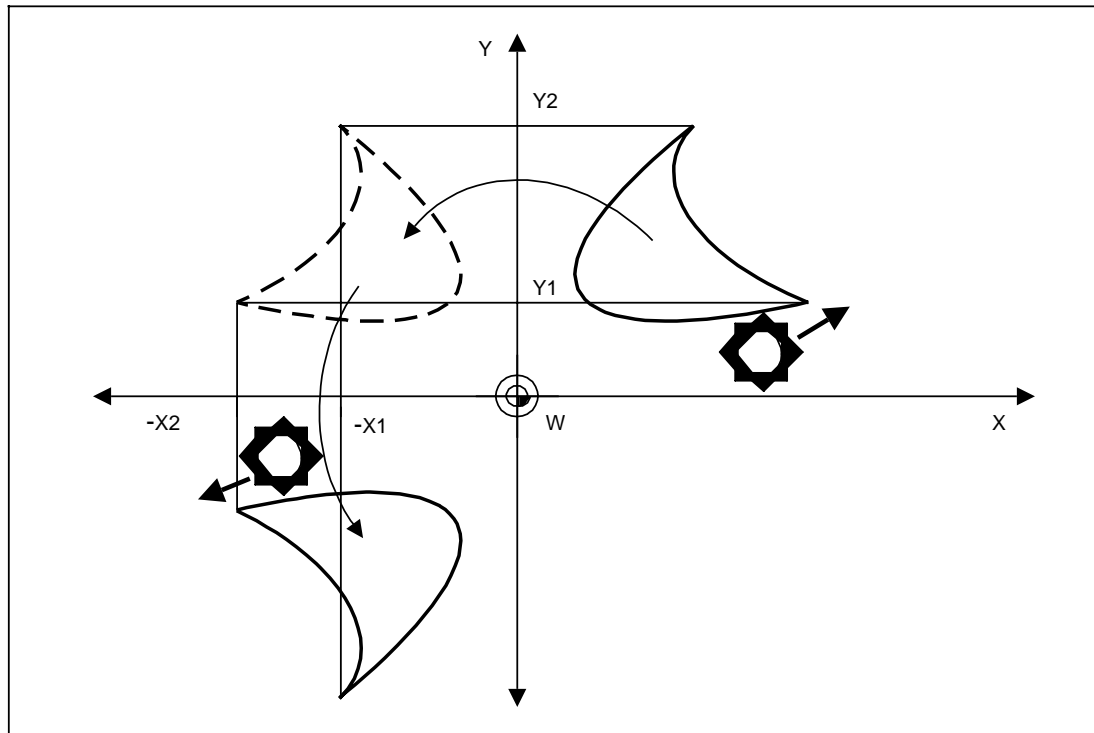
- the X axis is mirrored once,
- the Y axis is mirrored once.

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

The direction of machining and the direction of rotation for circular interpolation remain the same since they have been inverted twice.

There is no mirroring of:

- tool length offsets,
- zero offsets.

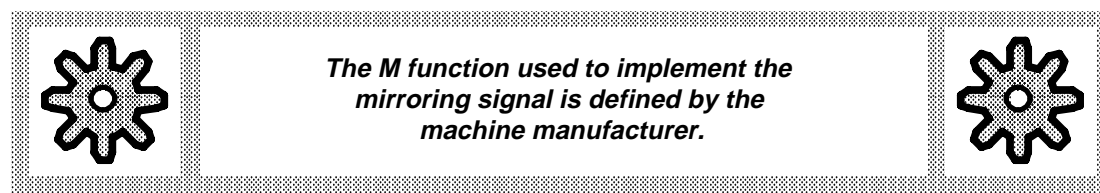


Mirroring of X and Y axes

When mirroring the main axes, the workpiece is always mirrored.

Program sections which must be mirrored take the form of self-contained subroutines in the program. The relevant mirroring function for the contour must then be selected before calling the subroutine.

The “mirroring” function is selected via the PLC.



Example: To select mirroring

N10	G90	G54	G00	X0	Y0	L _F	Absolute dimension, select 1st ZO
N20	Z30	L _F					
N30	G1	Z0	F500	L _F			
N35	M..	L _F					Call "Mirror X axis"
N36	G04	F...	L _F				Dwell time
N37	L999	P1	L _F				Subroutine L999, 1 pass (clear buffer)
.							
.							
.							
N40	X50	Y50	L _F				
N45	M..	L _F					Cancel "Mirror X axis"
.							
.							
.							
N46	G04	F...	L _F				
N47	L999	P1	L _F				
.							
.							
.							
N50	G0	G53	Z0	L _F			Select rapid traverse, deselect all ZO block by block
N60	G53	X0	Y0	M30	L _F		Deselect all ZO block by block, program end

Subroutine L999:

```
L999
@714 M17 LF
```

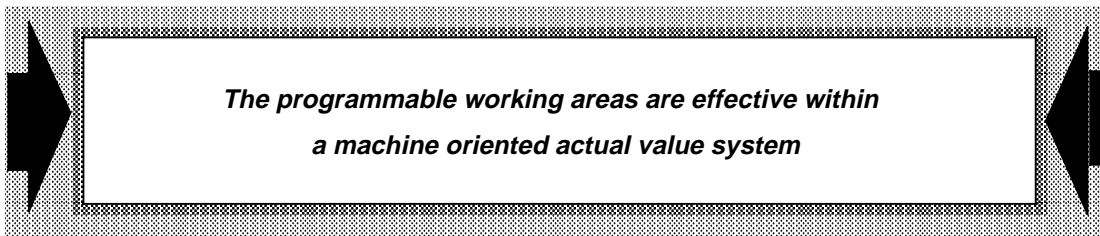
Special function @714 (buffer empty) makes it possible to stop any additional block increment calculation until the buffer is empty.

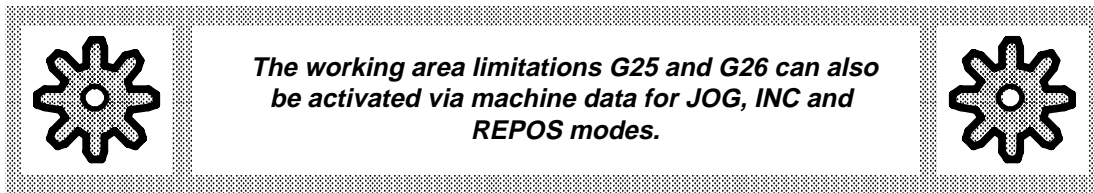
The use of @714 is required for all externally induced offsets, e.g. mirroring.

2.9 Programmable working area limitation G25/G26

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

The tool reference point F (tool holder) must only move in the limited range (area indicated by broken lines). 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.





The programmable working area limitation is called using G25 and G26:

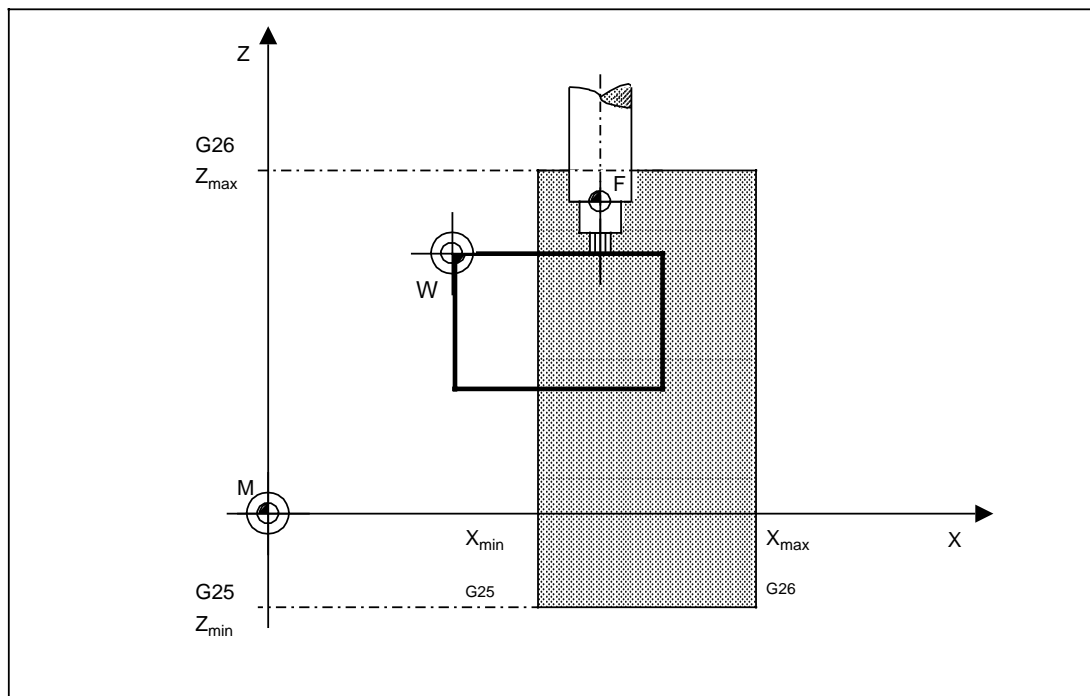
- G25 minimum working area limitation
- G26 maximum working area limitation

Example:

```
N10 G25 X200 Z-25 L_F  
N20 G26 X500 Z200 L_F
```

No more data are allowed in this block. The values in the setting data are overwritten with G25/G26. Working area limitation is no longer active when -99999999 and +99999999 respectively are input for the minimum and maximum values per axis in the setting data.

Example: Milling machine



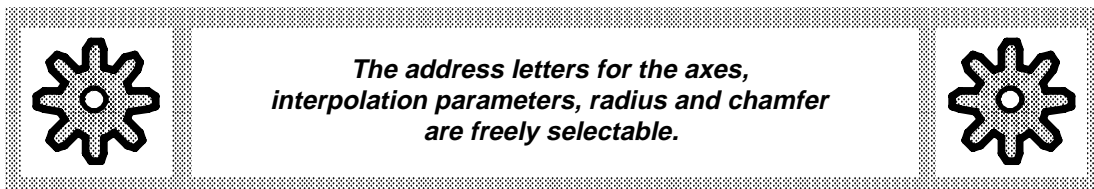
Example for milling machine

3 Programming of Motion Blocks

3.1 Axis commands

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

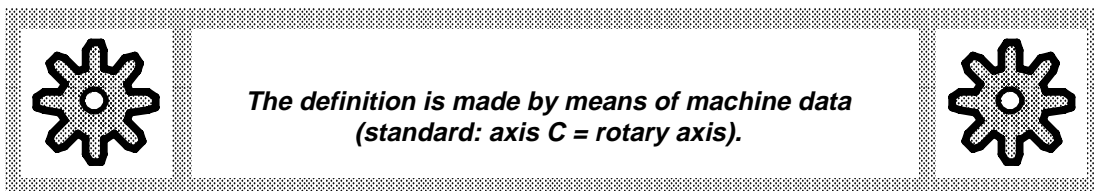
Addresses A, B, C, U, V, and W are optionally available for further axes.



The address letters generally used with milling machines are used in the following.

Rotary axis:

Any axis can be operated as a rotary axis.



The traversing range with absolute dimension programming (G90) is ± 360.000 degrees.

The traversing range with incremental dimension programming (G91) is ± 99999.999 degrees.

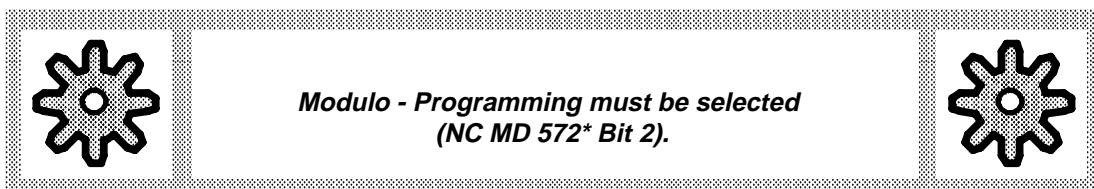
```

.
.
N5 G91 C 99999.999 L_F
.
.

```

Several axes can be declared rotary axes **at the same time**.

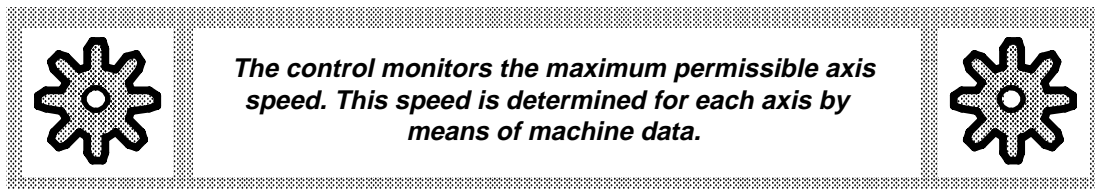
Rotary axes can generally **rotate endlessly**. The actual value is reset to zero degrees after *2982 times 360 degrees.



3.1.1 Axis motion without machining (G00)

Rapid traverse motions are programmed by means of the position data G00 and a **specified target position**. The target position can be reached by either an absolute (**G90**) or incremental (**G91**) position data input.

The path programmed with G00 is traversed at the **maximum possible speed**, rapid traverse, along a straight line (linear interpolation). **No** machining should be performed during the movement.

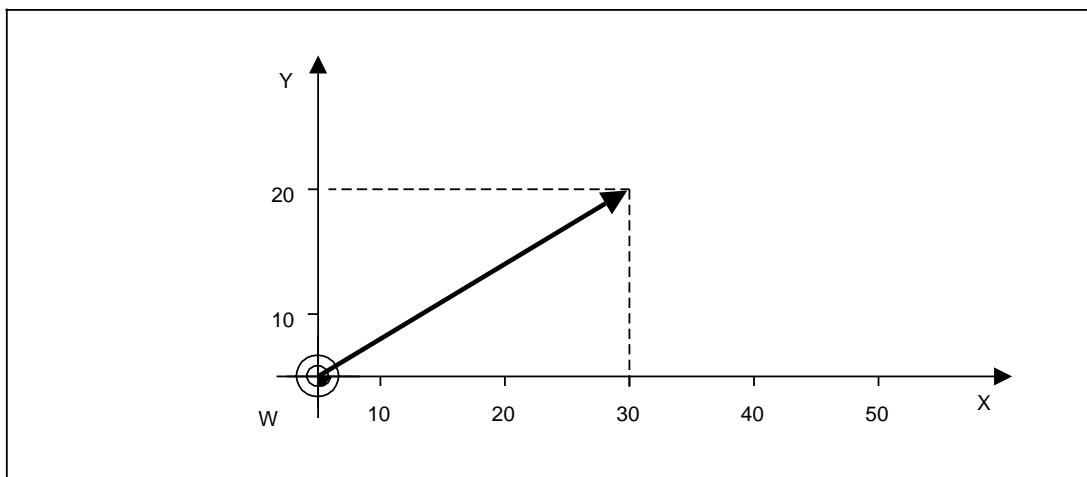


If the rapid traverse movement is executed in several axes at the same time, the traversing speed is determined by the lowest axis speed value specified by machine data.

The preparatory function G00 automatically initiates **exact positioning coarse**. When G00 is programmed, the **feedrate** programmed under the address F remains **stored**; it is reactivated again, for example, by G01.

Example:

N1 G00 G90 X30 Y20 L_F Rapid traverse, absolute dimension programming



3.2 Axis motion with machining

The **control** implements linear 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)

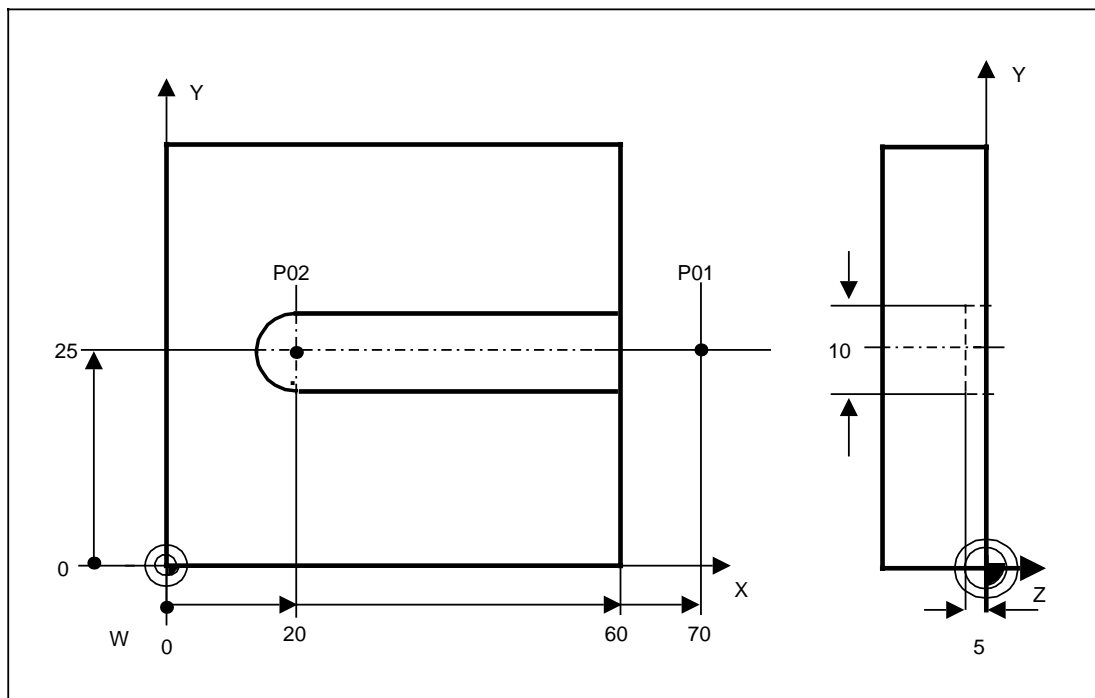
3.2.1 Linear interpolation G01

In this case, the tool travels at a set **feedrate** along a straight line to the target position while **machining** the workpiece at the same time. The **control calculates** the tool path by linear interpolation.

The **linear interpolation** effects the movement:

- in one axis direction (**linear or rotary axis**)
- from the starting position to the target position programmed with absolute or incremental dimensions
- at the programmed feedrate
- at the programmed spindle speed

Example: Paraxial milling



```

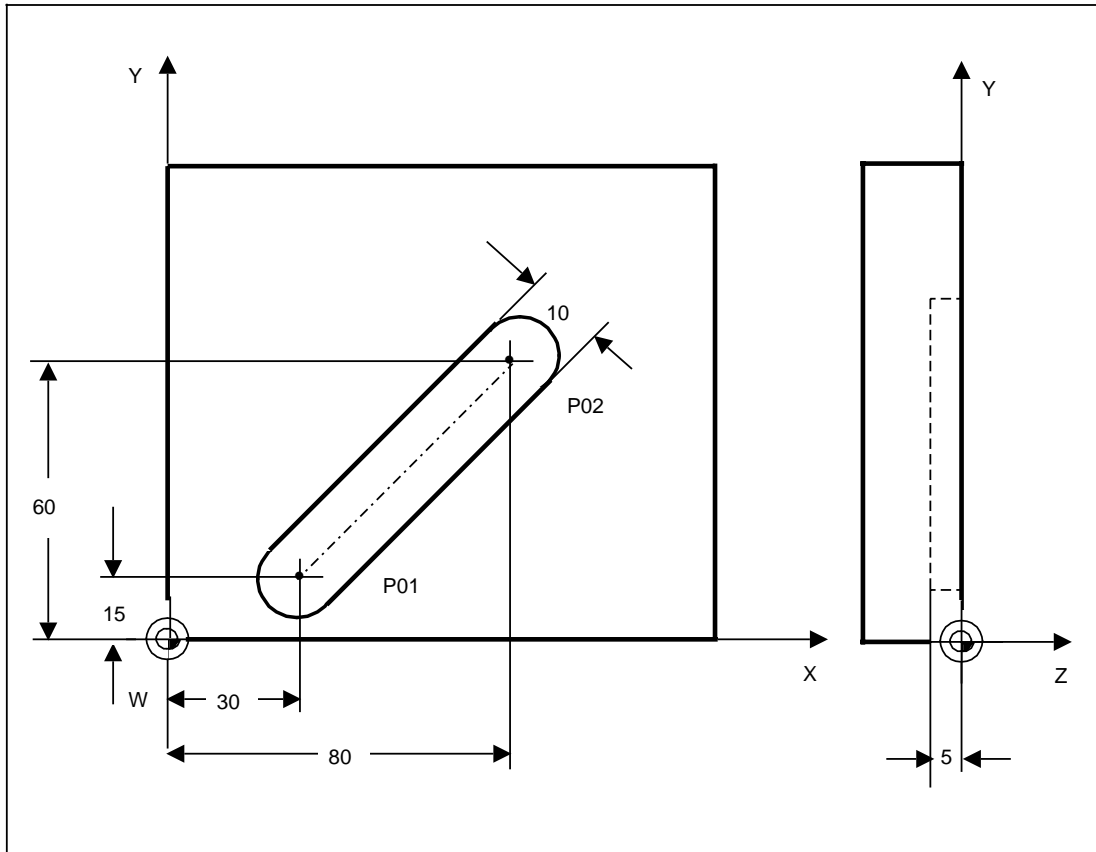
%315
(G01) L_F
N5 G00 G90 X70 Y25 Z1 S800 M3 L_F

N10 Z-5 L_F
N15 G01 X20 F150 L_F
N20 G00 Z100 L_F
N25 X-25 Y50 L_F
N30 M30 L_F

```

Spindle on, tool rapid traverse to P01, clockwise rotation 800 Rev./min.
Infeed to Z
Tool traverse from P01 to P02, feedrate 150 mm/min
Rapid traverse retraction
Rapid traverse retraction
End of program

Example: Linear interpolation in 2 axes



%325

(G01 in 2 axes, 2D interpolation) LF

N5 G00 G90 X30 Y15 Z1 S5000 M3 LF

N10 G01 Z-5 F100 LF

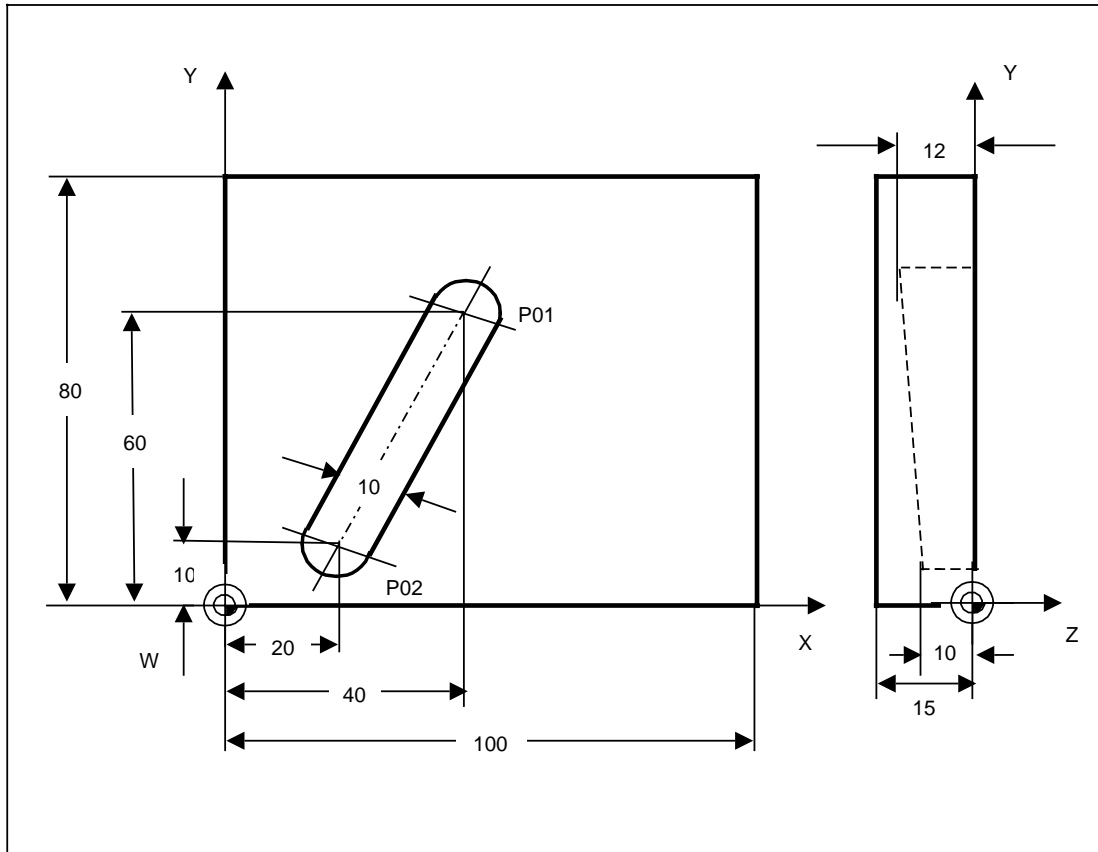
N15 X80 Y60 F200 LF

N20 G00 Z100 LF

N25 X-40 Y100 LF

N30 M30 LF

Tool rapid traverse to 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

```
%330
```

```
(G01 in 3 axes, 3D interpolation) LF
```

```
N5 G00 G90 X40 Y60 Z2 S5000 M3 LF
```

```
N10 G01 Z-12 F100 LF
```

```
N15 X20 Y10 Z-10 LF
```

```
N20 G00 Z100 LF
```

```
N25 X-20 Y80 LF
```

```
N30 M30 LF
```

Tool rapid traverse to P01

Infeed to Z-12, feedrate 100 mm/min

Tool traverse along a straight line in space 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 simultaneously machining the workpiece.

The control calculates the **tool traversing path** by means of circular interpolation.

Circular interpolation effects **tool** motion:

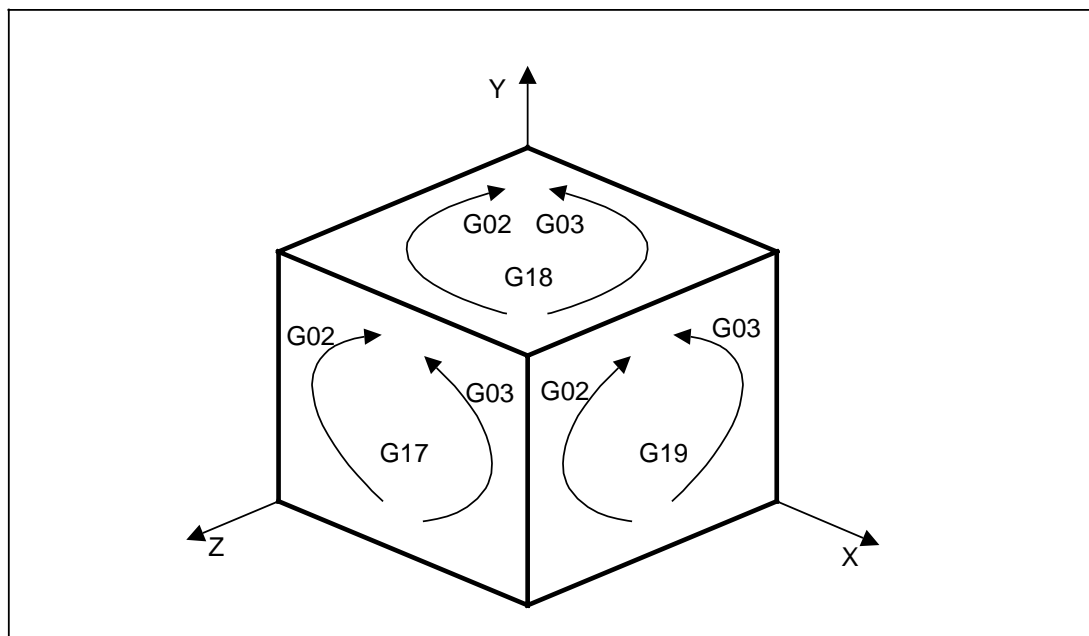
- along a circular arc
 - in a clockwise direction with **G02**
 - in a counterclockwise direction with **G03**
- in a specific plane via Function G16, G17, G18, and G19
- from the starting position on a circular path to the programmed end position.

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

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

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 counter-clockwise direction with G03.



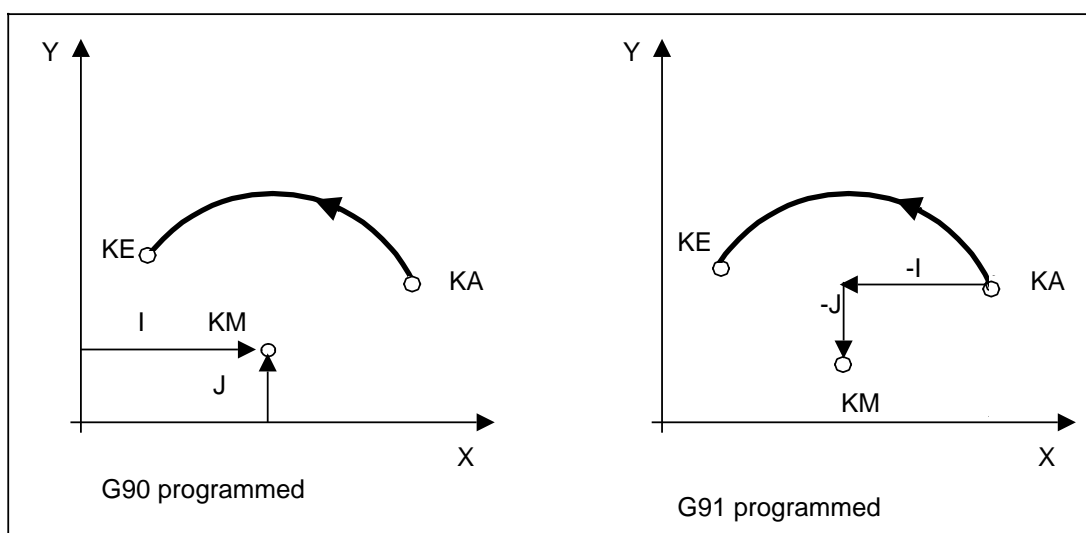
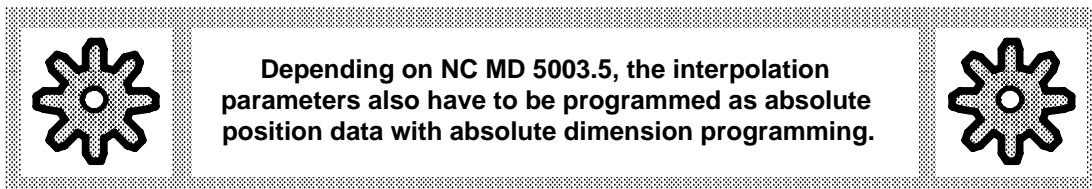
Circular interpolation

The interpolation parameters together with the axis commands determine the circle or circular arc. The starting point “**KA**” on the circle or circular arc is defined by the preceding block. The end point “**KE**” is defined by axis values X,Y and Z. The centre point of the circle “**KM**” is defined either:

- by the interpolation parameters (I,J,K), **or**
- 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**. As a standard, the interpolation parameters must always be entered **as incremental position**. The **sign** is based on the direction of the coordinates from the starting position to the centre point of the circle. If the value of an interpolation parameter is 0, this parameter need not be programmed.

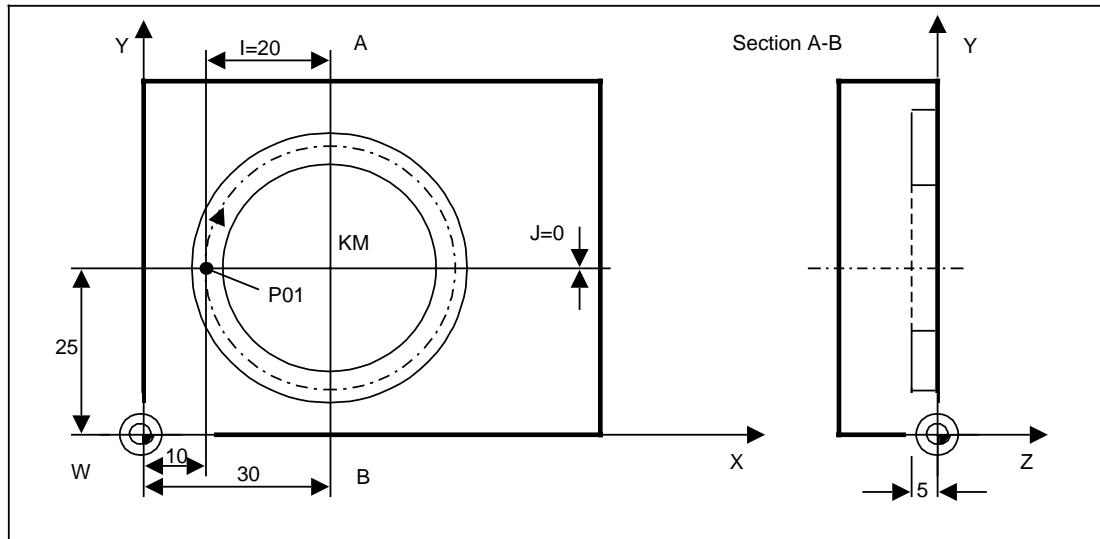


Circular interpolation with interpolation parameters (incremental dimension) depending on NC MD 5003.5.

Similarly any end point coordinates which are the same as for the starting point of the circular path need not be programmed. At least one axis must be programmed for a full circle (X0, Y0 or Z0).

Example: Full circle in the X-Y plane

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



Full circle in the X-Y plane

```

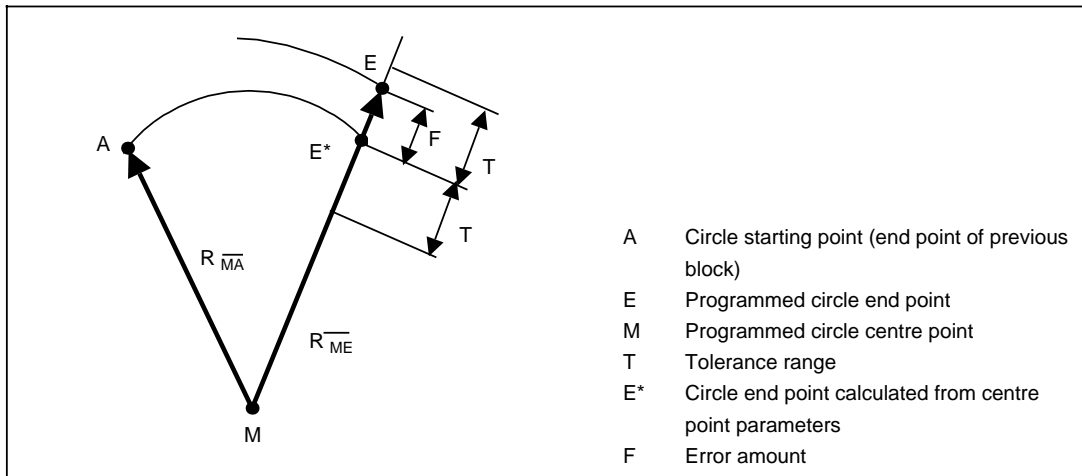
%340
(G02, CIRCULAR INTERPOLATION) LF
N5 G00 X10 Y25 Z1 S1250 M3 LF
N10 G01 Z-5 F100 LF
N15 G02 X10 Y25 I20 J0 F125 LF

N20 G00 Z100 M5 LF
N25 X-20 LF
N30 M30 LF
    
```

Tool rapid traverse to P01
 Infeed to Z-5
 X-Y plane selected automatically (reset). Tool
 clockwise traverse around a full circle (G02)
 Rapid traverse retraction
 Rapid traverse retraction
 End of program

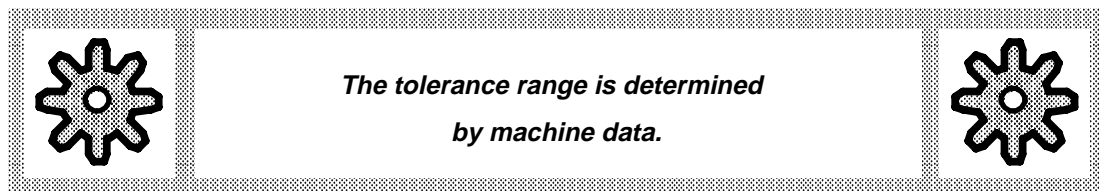
Circle end point monitoring:

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 point **A** and end point **E**.



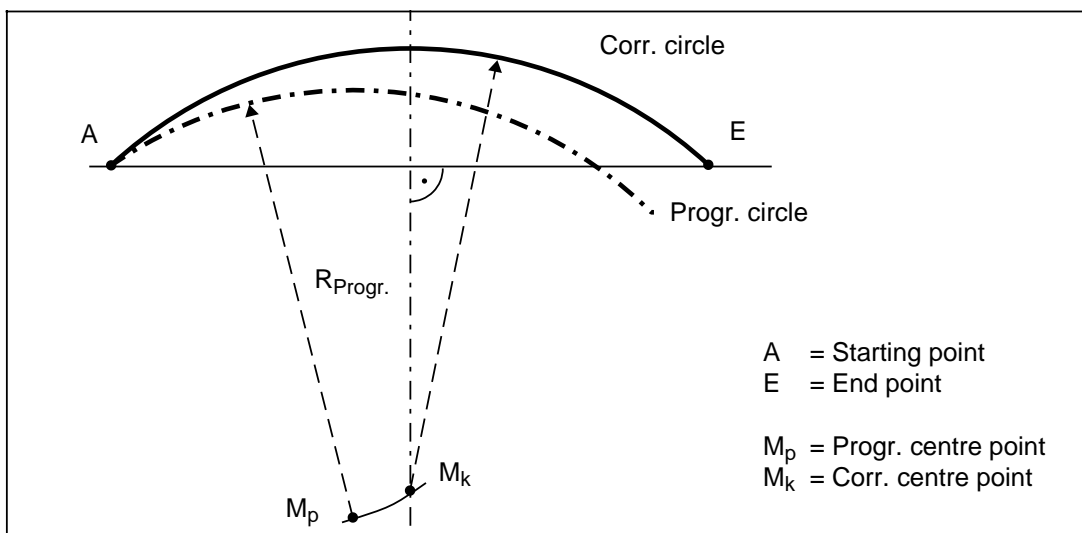
Definition of tolerance range of circle end point monitoring

If the radius or interpolation parameters are not input accurately, the difference (F) oversteps the tolerance range (T) and the circle block is not processed; the **"circle end point error"** alarm is issued.



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 point has been "properly" programmed.

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



Correction of the programmed circle

3.2.2.2 Radius programming

In many cases the dimensions of a drawing are such that it is easier to specify **radius U** when defining the **circular path**.

Since the radius in conjunction with G02 or G03 can only specify a definite circular path within a semicircle, it is necessary to specify in addition whether the traversing angle is to be greater or less than 180°.

Thus the radius is given the following sign:

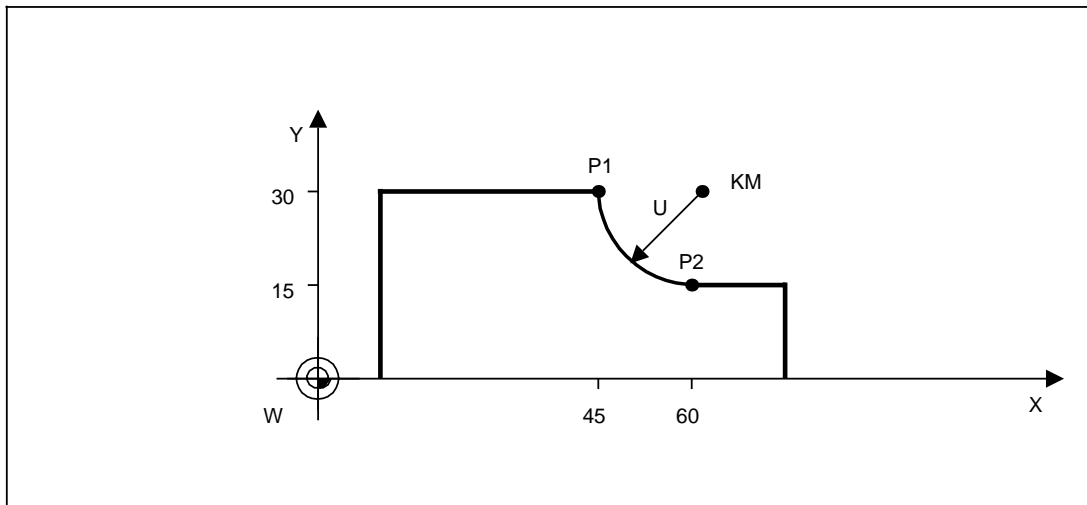
+U: traversing angle less than or equal to 180°

–U: traversing 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**.

G02 or G03 determines the direction of movement about the circle defined by means of the circle end point and the interpolation parameters or by means of radius U.

Example: Radius programming



Radius programming

%341

N5 G00 G90 Y30 F500 LF

N10 X45 LF

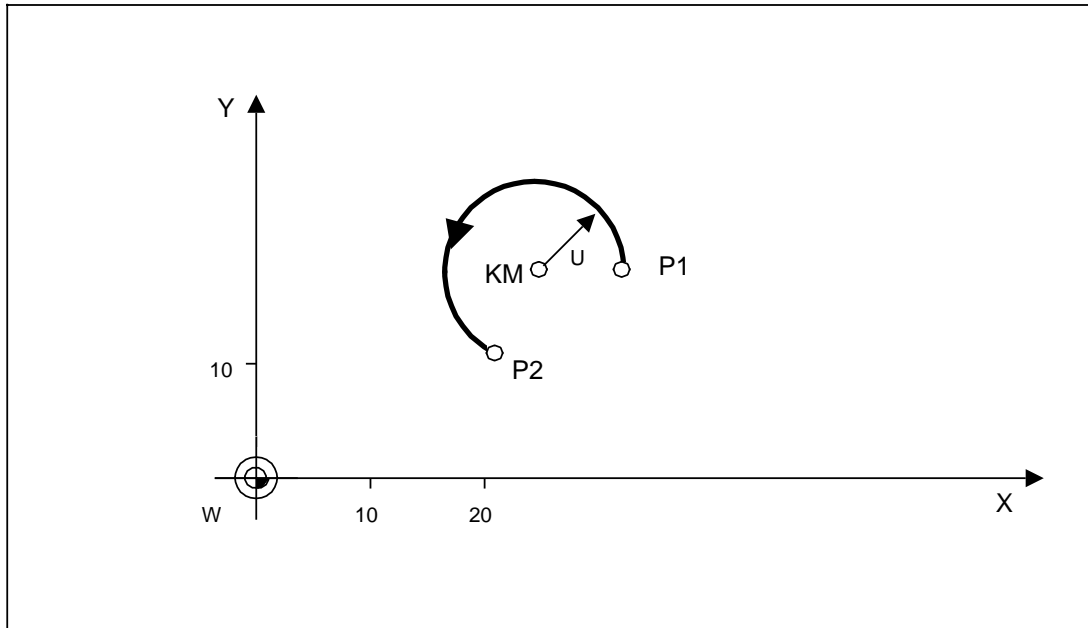
N15 G03 G90 X60 Y15 U15 LF

Tool machines from point 1 to point 2

N20 G02 X45 Y30 LF

Tool machines from point 2 to point 1

N25 M02 LF

Example: Radius programming*Radius programming*

```
%342
```

```
(G03, RADIUS PROGRAMMING) LF
```

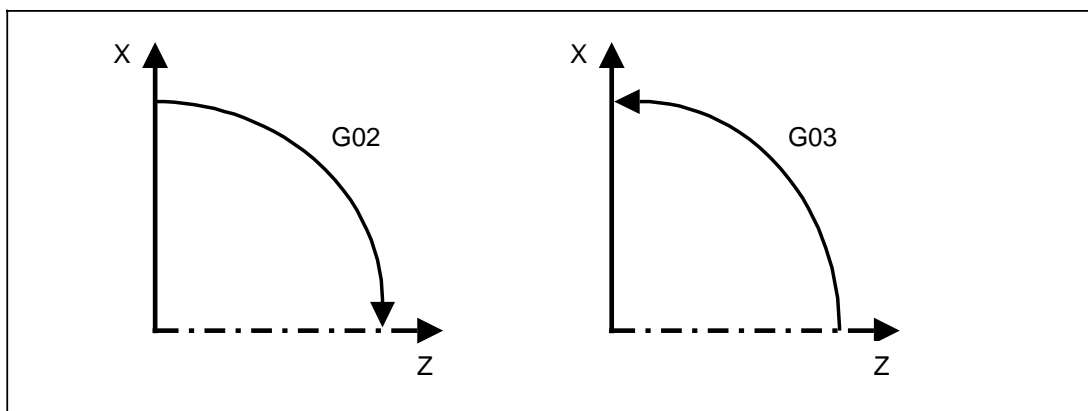
```
N10 G01 G90 X30 Y20 F500
```

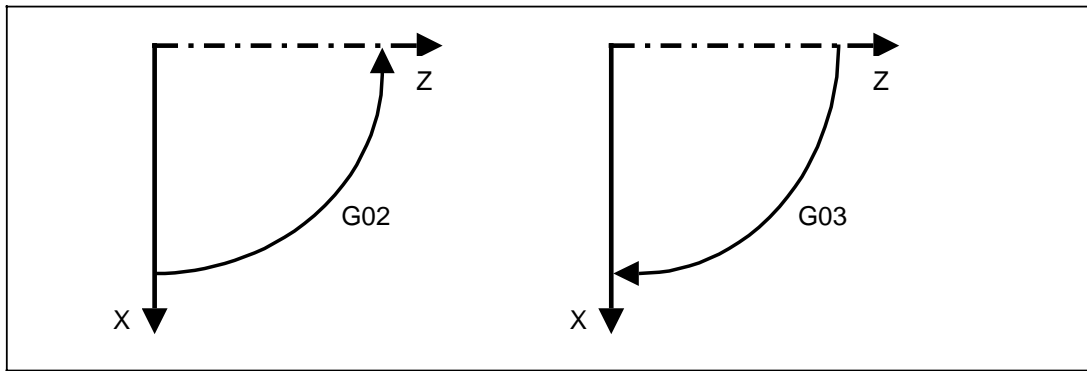
Tool traverse to point 1

```
N15 G03 X20 Y10 U-10 LF
```

Counter-clockwise tool machining from point 1 to point 2

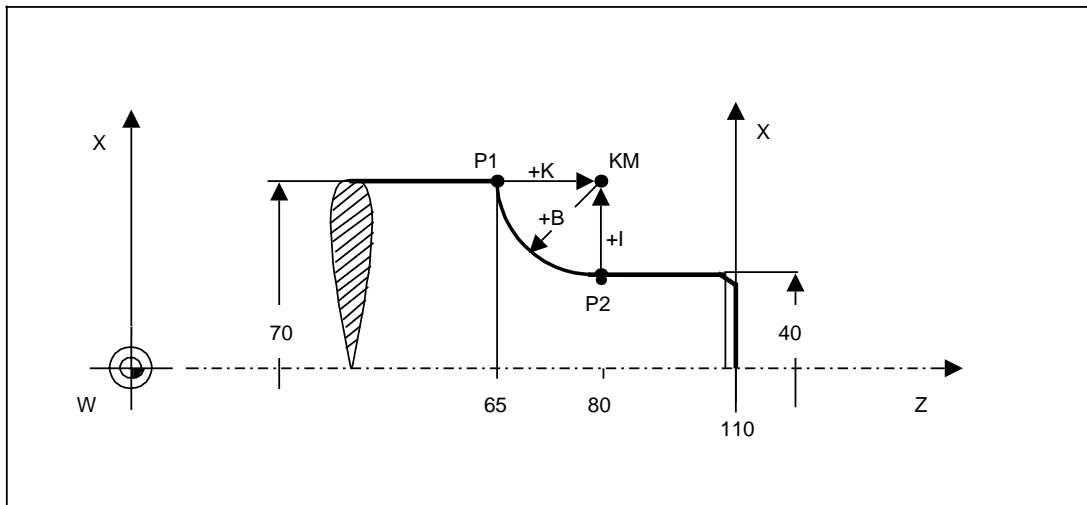
```
N20 M30 LF
```

Turning machine: effect of G02 and G03 after and before the turning centre*After turning centre*



Before turning centre

Programming example: Turning machine



Circular interpolation

Interpolation parameters:

```

.
.
.
N2  G01  X70  Z65  F500  LF
N5  G03  G90  X40  Z80  K15  I0  LF           Tool machines from P1 to P2
N10 G02  X70  Z65  K0  I15  LF           Tool machines from P2 to P1
    
```

or radius programming:

```

.
.
.
N12 G01  X70  Z65  F500  LF
N15 G03  G90  X40  Z80  B+15  LF           Tool machines from P1 to P2
N20 G02  X70  Z65  B+15  LF           Tool machines from P2 to P1
.
.
.
    
```


3.2.3 Helical interpolation

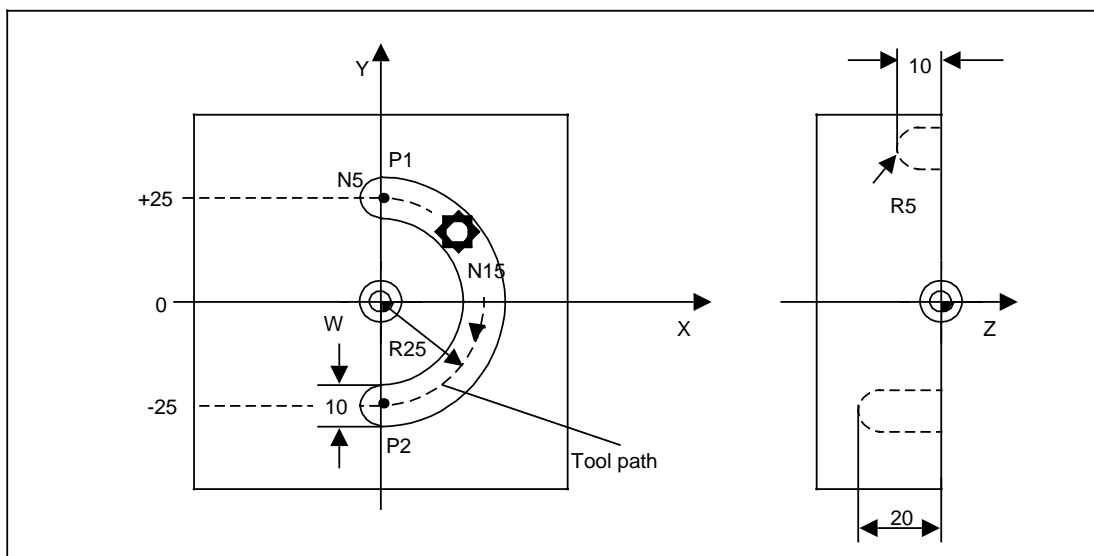
Helical interpolation is possible between **3 linear axes** which are perpendicular to one another.

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 constant lead.

The circular plane is **selected** with G16, G17, G18, or G19. 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.

Example: Helical interpolation



Helical interpolation

%380

(HELICAL INTERPOLATION) L_F

N5 G00 X0 Y25 Z1 S800 M3 L_F

N10 G01 Z-10 F150 L_F

N15 G02 X0 Y-25 Z-20 I0 J-25 L_F

N20 G00 Z100 M5 L_F

N25 M30 L_F

X-Y plane selected automatically (reset).

Tool rapid traverse to point P1

Linear interpolation; infeed to Z-10

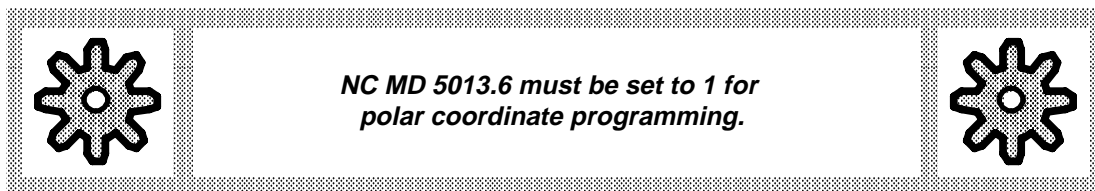
Tool clockwise traverse (G02) on a helix to P2

Rapid traverse retraction

End of program

3.2.4 Polar coordinates G10, G11, G12, G13

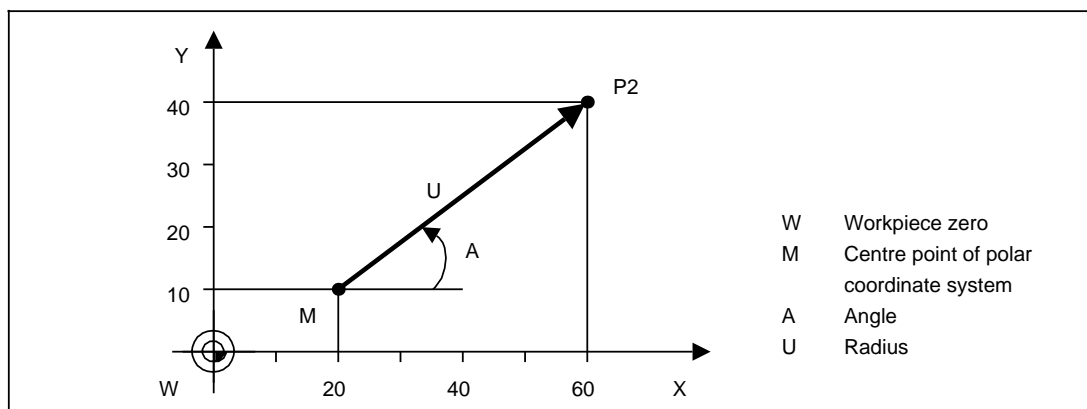
Drawings dimensioned with angles and radii can be entered **directly** in the program with the aid of polar coordinates.



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

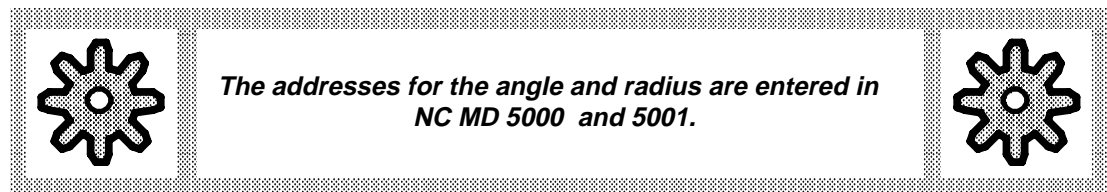
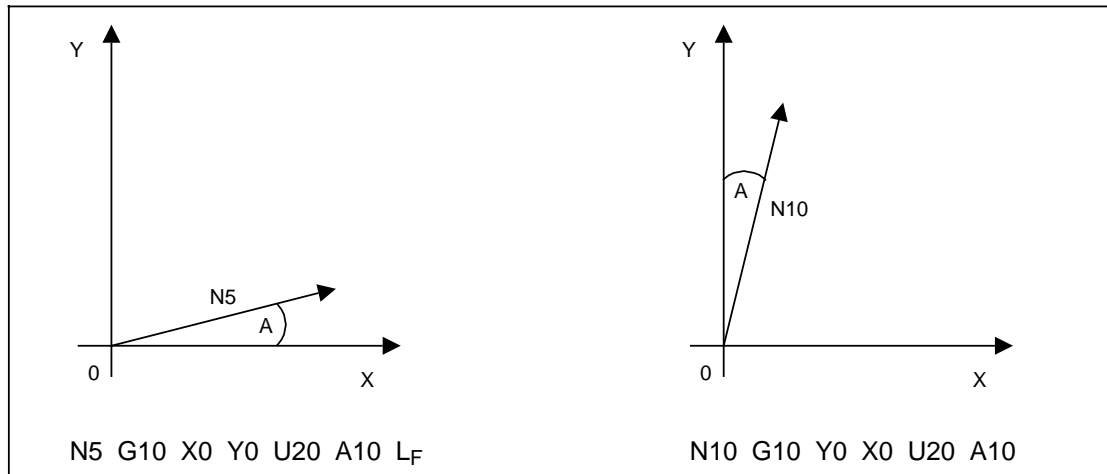
The action of the preparatory functions is **modal**.



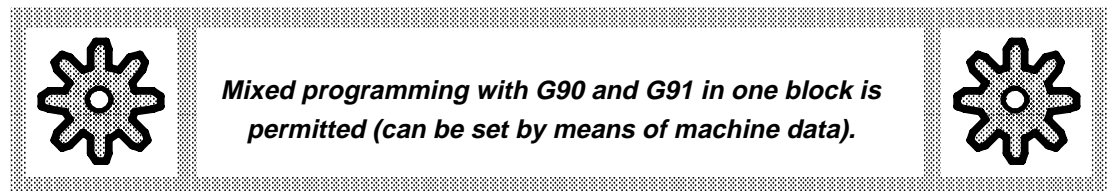
In order to determine the traverse path, the control requires the centre point, the radius and the angle. The centre point is entered with perpendicular coordinates (X, Y, Z) and - on initial programming - using absolute position data. A subsequent incremental position data input (with G91) always refers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

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

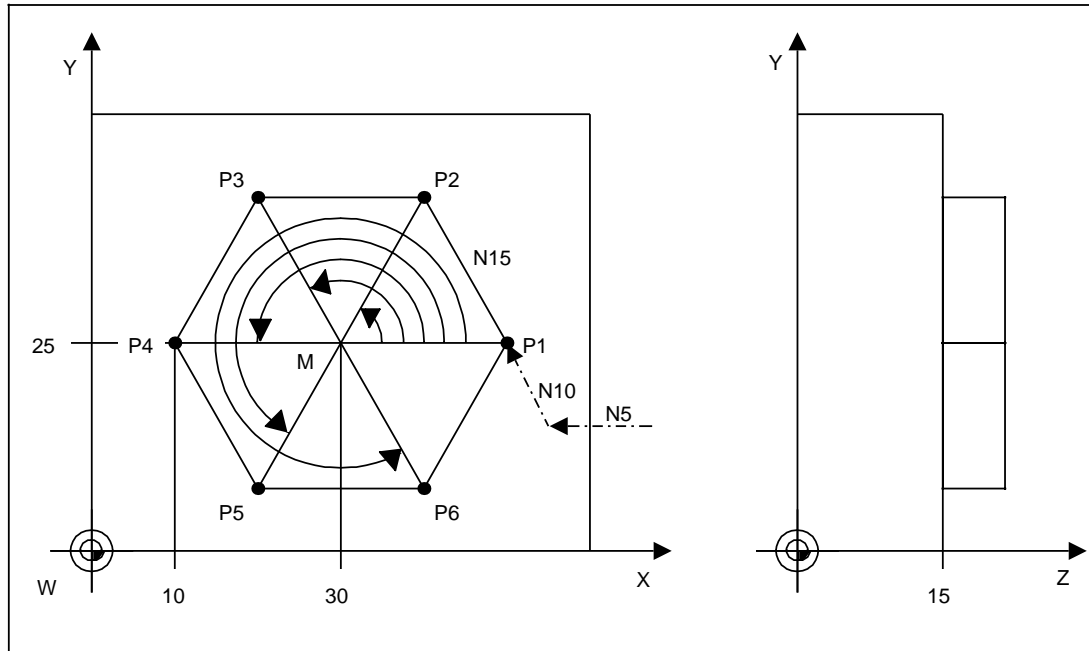
The radius is programmed under the address **U without sign**. The **angle** is entered under the address **A** without sign. It always refers to the **first positive axis** of the centre point coordinates to be programmed (reference axis).



Angles are specified as absolute or incremental and **positive** data.



Example: G11, milling machine



G11, milling machine

%382

N5 G90 G0 Z15 X50 Y20 L_F
 N10 G11 X30 Y25 U20 A0 F100 L_F (P1)

Infeed in Z, absolute dimension input

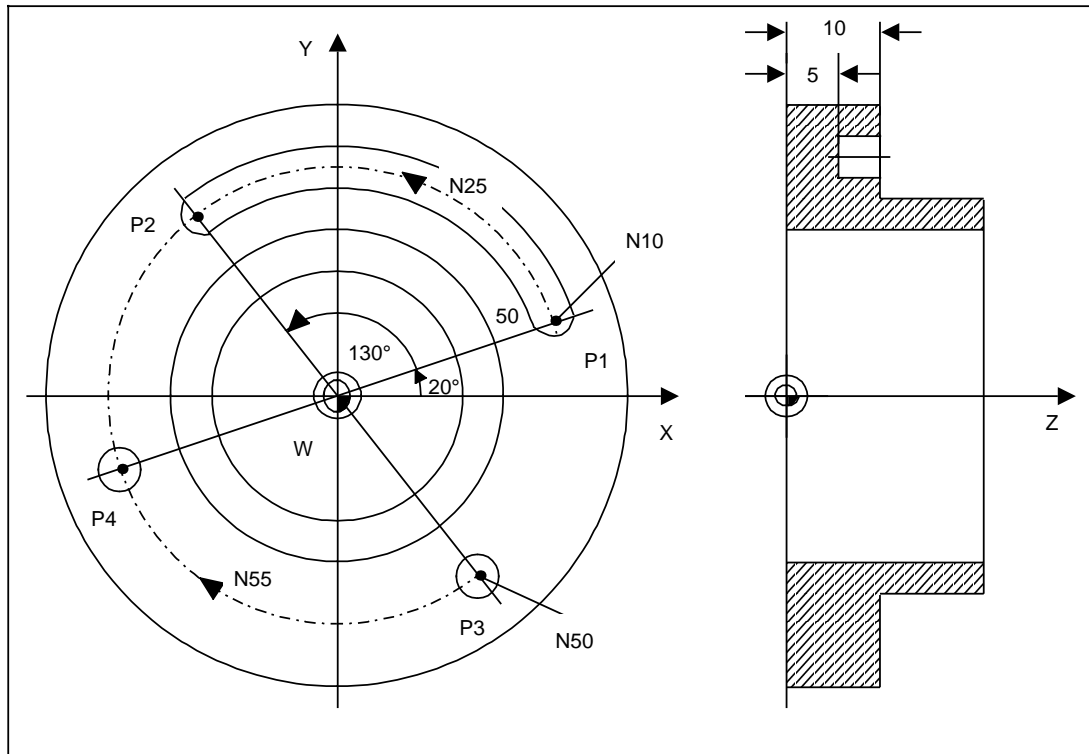
X..., Y..., = Centre point input

U..., A..., = Radius, angle

N15 A60 L_F
 N20 A120 L_F
 N25 A180 L_F
 N30 A240 L_F
 N35 A300 L_F
 N40 A0 L_F
 N45 G0 Z100 L_F
 N50 M30 L_F

(P2)
 (P3)
 (P4)
 (P5)
 (P6)
 (P1)

} Mill hexagon

Example: G10, G12, G13, milling machine

G10, G12, G13, milling machine

```
% 383
```

```
(G10, G12, G13, POLAR COORDINATES) L_F
```

```
N5 G90 G0 X200 Y300 Z100 L_F
```

```
N10 G90 G10 X0 Y0 U50 A20 L_F
```

```
N15 G0 Z10 L_F
```

```
N20 G1 Z5 F1000 S800 M3 L_F
```

```
N25 G13 A130 L_F
```

```
N30 G0 Z100 L_F
```

```
N35 G0 X100 Y100 M5 L_F
```

```
N40 T2 M6 L_F
```

```
N45 Z20 S800 M3 D2 L_F
```

```
N50 G81 G10 A310 R2=20 R3=5 R10=25 L_F
```

```
N55 G12 A200 R2=20 R3=5 R10=25 L_F
```

```
N60 G80 G0 Z100 L_F
```

```
N65 M30 L_F
```

(P1) Approach P1 (polar coordinates)

Infeed in Z

(P2) Mill slot
Approach tool change point

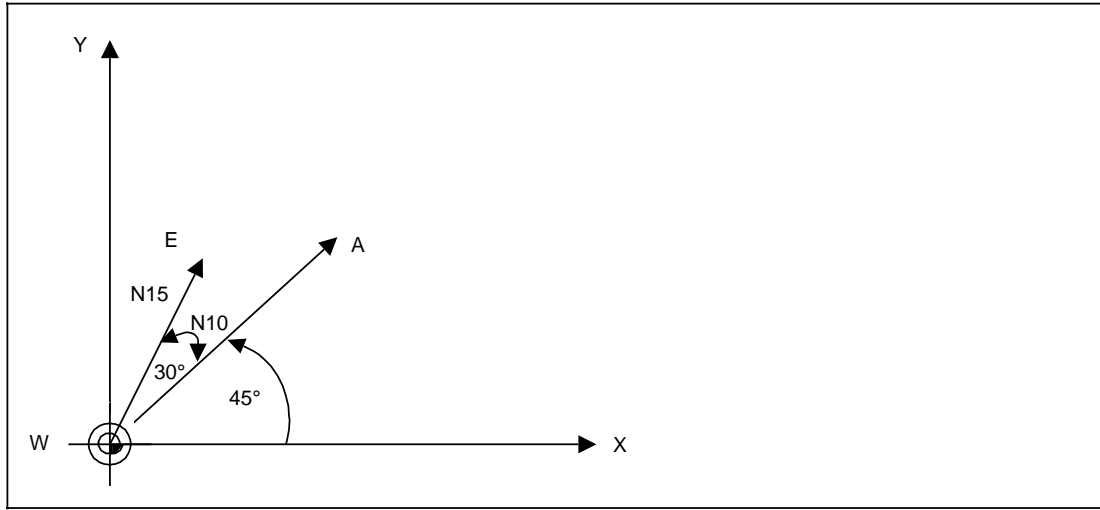
Change tool

(P3) Approach first drilling position and call
drilling cycle

(P4) Approach second drilling position and
automatic call of drilling cycle
Cancel drilling cycle and clear

Example: Incremental polar coordinate programming (G91)

With incremental polar coordinate programming the angles are added up but the radii are always calculated from the last centre point.



Incremental polar coordinates

```
%384  
(Incremental polar coordinates) LF  
N5 G90 G10 X0 Y0 U0 A0 F1000 LF  
N10 G90 G11 U30 A45 LF  
N15 G91 G11 U20 A30 LF  
N20 M30 LF
```

NC MD 5003.4 must be set to 1 for incremental polar coordinate programming.

3.2.5 Polar coordinates G110, G111

The functions G110 and G111 are used to adopt a new centre point or zero point.

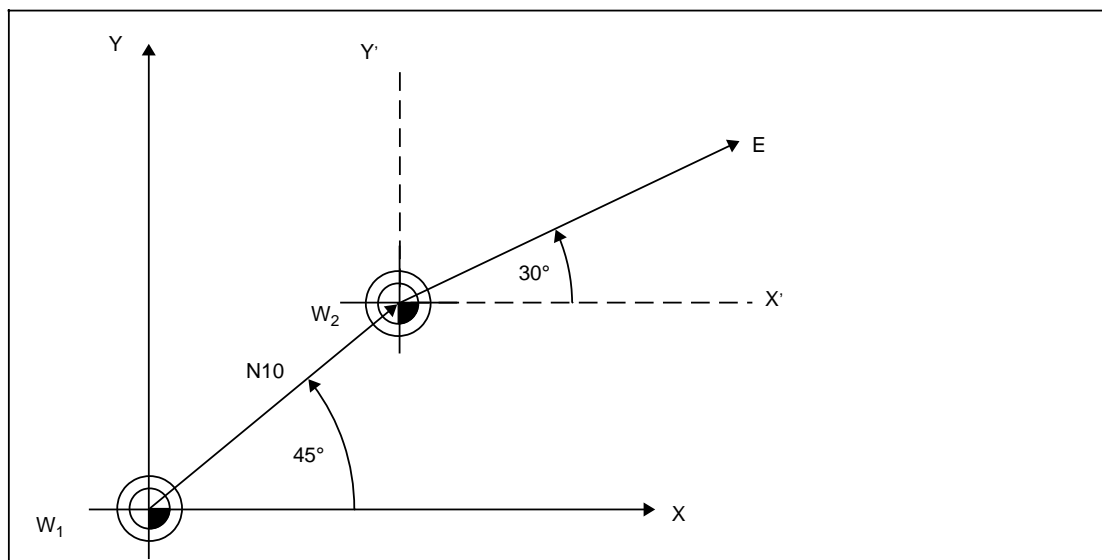
Using the new centre point, the angles are again taken from the horizontal and the radius is calculated from the new centre point.

G110: Adopt the setpoint reached as the new centre point

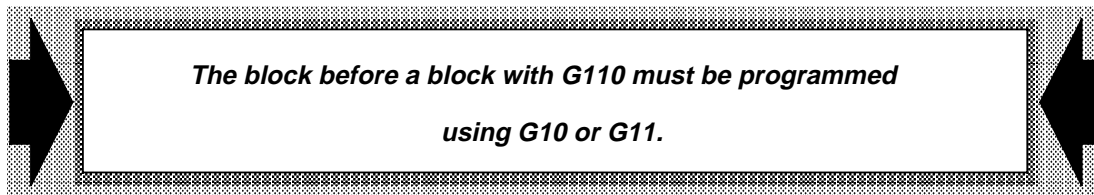
G111: Centre point programming with angle and radius without axis movement
(example: setting the arc center of a hole circle)
(the following traversing movement must be programmed using G110)

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
```

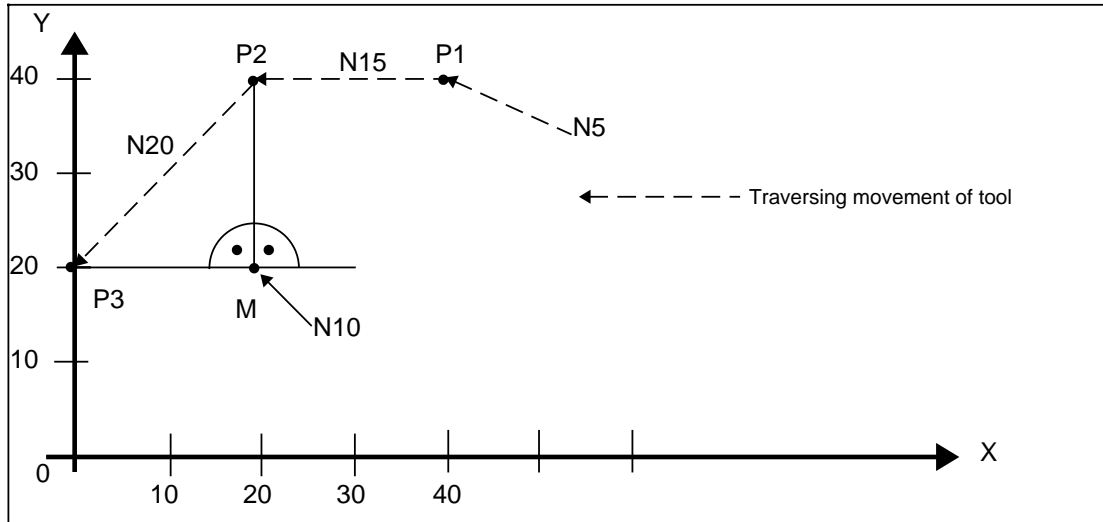


Polar coordinates G110



Example: Polar coordinates G111

```
%386  
(G111 Polarkoordinaten) L_F  
N5 G11 X40 Y40 A U F1000 L_F P1  
N10 G111 A225 U28 L_F Set centre point without traversing  
N15 G110 A90 U20 L_F P2  
N20 A180 L_F P3  
N25 M30 L_F
```



Polar coordinates G111

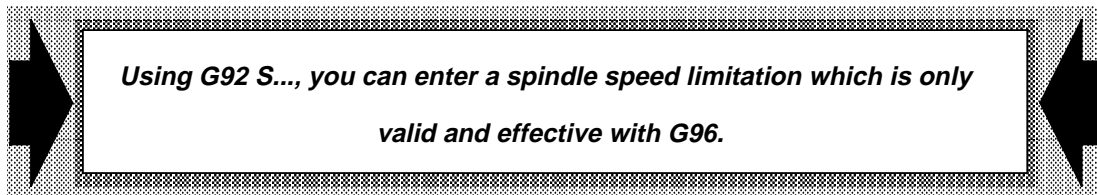
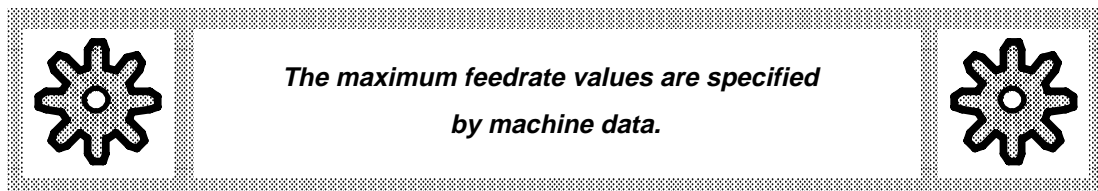
3.2.6 Feedrate F, G94, G95, G96, G97

The feedrate **F** is programmed in mm/min (m/min) or **mm/rev**:

- G94 F..** Feedrate in mm/min
- G95 F..** Feedrate in mm/rev
- G96 F..** Feedrate in mm/rev
- S..** Constant cutting speed (S in m/min)

The feedrate determines the **machining speed** (tool path feedrate) and is adhered to with all types of interpolation, even when tool offsets on the contour are taken into consideration. The value programmed under address **F** **remains** in the program until a **new F value is programmed**. The F value is **deleted at the end of the program or on a reset**. An F value must therefore be programmed in the **first program block**.

The **programmed** feedrate F can be modified between 1% and 120% by means of a feedrate **override switch** at the operator panel. The **100% setting** corresponds to the programmed value.



Constant cutting speed G96 S..

A constant cutting speed can be entered under address **S** with **G96**:
G96 F.. S.. Constant cutting speed in m/min

Example: Turning machine

The **control determines the spindle speed** for the current turning diameter in accordance with the programmed cutting speed.

N5.. G96 F.. S..

The relationship between the turning diameter, the spindle speed and the feed motion permits **optimum** matching of the **program** to the workpiece and the tool.

The zero of the **X axis must be the turning centre**. This is ensured by the reference point approach.

When calculating the spindle speed for the constant cutting speed, the following values are taken into consideration:

- Actual value
- Tool length compensation
- Zero offset in the X direction
 - Settable zero offset G54, G55, G56, G57
 - Programmable zero offset G59, G58
 - External zero offset

The workpiece must not be displaced from the turning centre as a result of zero offset in the X axis. Zero offset in the X axis can be used to displace the tool carrier.

Reference is always made to the workpiece zero when determining the spindle speed.

One gear speed only is used for a constant cutting speed. It is not permissible to change the gear speed. The relevant gear speed must be selected in advance.

Constant speed G97

The constant cutting speed is cancelled with G97. The final speed is retained as the constant speed. An undesirable speed change can be avoided with G97 for motions in the X direction without machining.

G97 Cancellation of constant cutting speed and storage of last set speed of G96.

Feedrate reduction ratio M37

The programmed feedrate can be reduced by 1:100 with M37. M37 is cancelled with M36 (M 36 = initial setting).

3.2.7 Thread cutting G33,G34, G35

There are various types of thread which can be cut as follows:

- Threads with constant lead
- Single or multiple threads
- Threads on cylindrical or tapered blocks
- External or internal threads
- Transversal threads

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

G33 Thread cutting with constant lead

G34 Thread cutting with linear lead increase

G35 Thread cutting with linear lead decrease

The **thread length** is entered under the corresponding path address; starting, stopping 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, K, depending on the axis address.

Example: Turning machine

The lead is entered under K for longitudinal threads, under I for transversal threads and under I or K for tapered threads. I and K must always be entered using incremental position data and without sign. The standard input resolution of the thread lead is 1 mm/revolution. The thread lead can be programmed between 0.001 mm and 400.000 mm. If a thread lead of 1 mm is programmed as the input resolution, it is possible to obtain a resolution of 0.01 mm/revolution with M37.

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

The **spindle direction of 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
N15 G33 Z... K... L_F
```

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 performed at the **same spindle speed** to avoid different following errors.

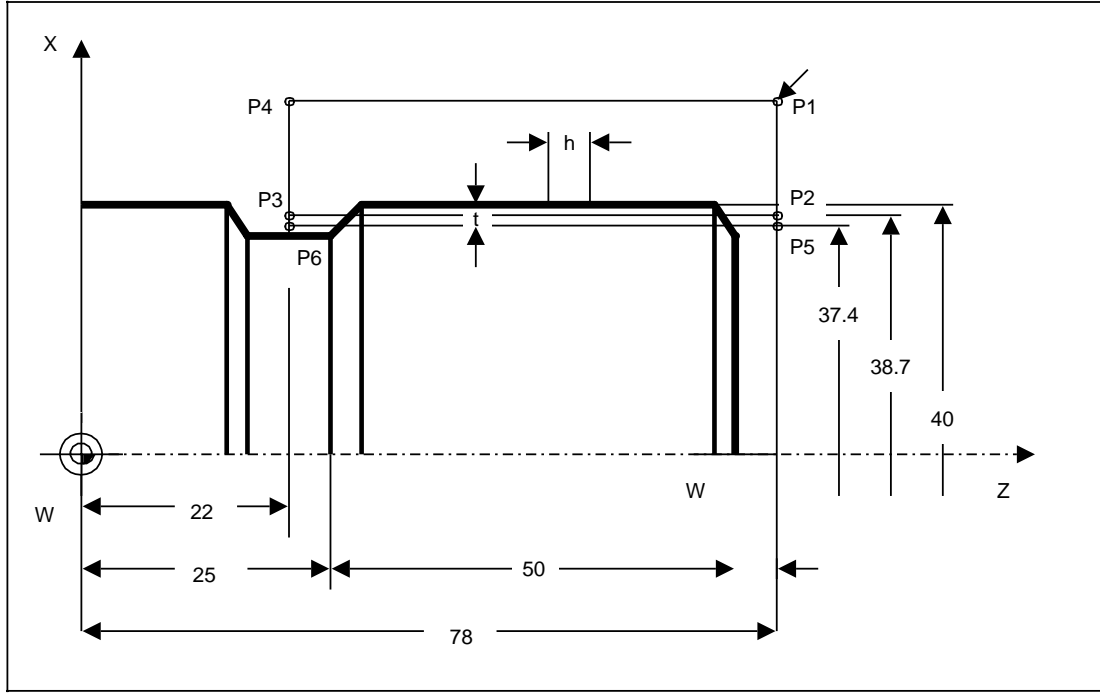
The feedrate override switch, the "FEED OFF" key, the spindle speed override switch and the "SINGLE BLOCK" switch have **no effect** during thread cutting.

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

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.

Example: Longitudinal thread



Thread on a cylindrical block

Lead $h = 2 \text{ mm}$
 Thread depth $t = 1.3 \text{ mm}$
 Radial infeed direction

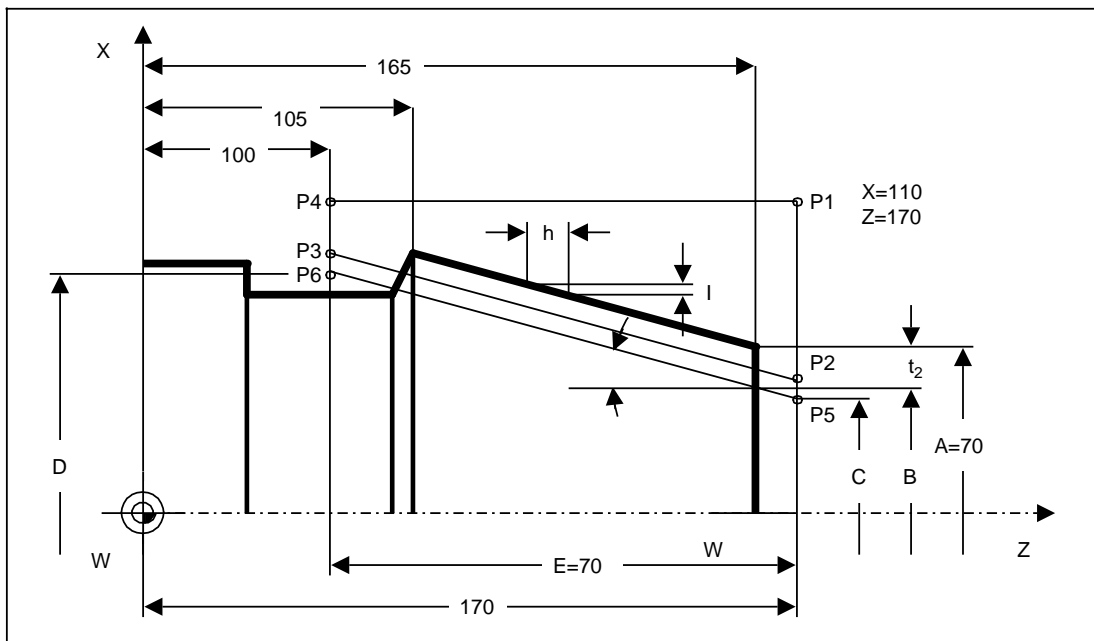
```
%387
(G33, G90 THREAD CUTTING, LONGITUDINAL) L_F
N5 G90 S400 M3 L_F           Spindle speed
N10 G00 X46 Z78 L_F         (P1) Approach P1 at rapid traverse
N15 X38.7 L_F               (P2) Infeed to 1st cutting depth
N20 G33 Z22 K2 L_F         (P3) Thread cutting, 1st pass from P2 to P3
N25 G00 X46 L_F           (P4) Retract to P4 in rapid traverse
N30 Z78 L_F                (P1) Approach P1 in rapid traverse
N35 X37.4 L_F              (P5) Infeed to 2nd cutting depth
N40 G33 Z22 K2 L_F         (P6) Thread cutting, 2nd pass from P5 to P6
N45 G00 X46 L_F           (P4) Retract to P4 in rapid traverse
N50 M30 L_F
```

```

% 388
(G33, G91 THREAD CUTTING, LONGITUDINAL) L_F
N5 G91 S400 M3 L_F
N10 G00 G90 X46 Z78 L_F (P1)
N15 G91 X-3.65 L_F (P2)
N20 G33 Z-56 K2 L_F (P3)
N25 G00 X3.65 L_F (P4)
N30 Z56 L_F (P1)
N35 X-4.3 L_F (P5)
N40 G33 Z-56 K2 L_F (P6)
N45 G00 X4.3 L_F (P4)
N50 M30 L_F

```

Example: Tapered thread



Thread on a tapered block

Lead $h = 5 \text{ mm}$
 Thread depth $t = 1.73 \text{ mm}; \alpha = 15^\circ$
 Radial infeed direction

Both end point coordinates must be written. The lead h is entered under K .

Calculation of thread start and end point coordinates

(A, B, C, etc. are diameters)

1st cut P2...P3, $t_1 = 1$ mm (roughing)

2nd cut P5...P6, $t_2 = 1.73$ mm (finish cut)

$$\begin{aligned}
 A &= 70 \\
 B &= A - 2 t_2 = 66.54 \\
 C &= B - 2 \cdot (h \cdot \tan \alpha) = 63.86 \\
 D &= C + 2 \cdot (E \cdot \tan \alpha) = 101.366 \\
 K &= h = 5 \\
 l &= h \cdot \tan \alpha = 1.34 \text{ mm}
 \end{aligned}$$

Calculation of points P2 and P3

$$X(P2) = C + 2 \text{ mm} = 65.86 \text{ mm}$$

$$X(P3) = D + 2 \text{ mm} = 103.366 \text{ mm}$$

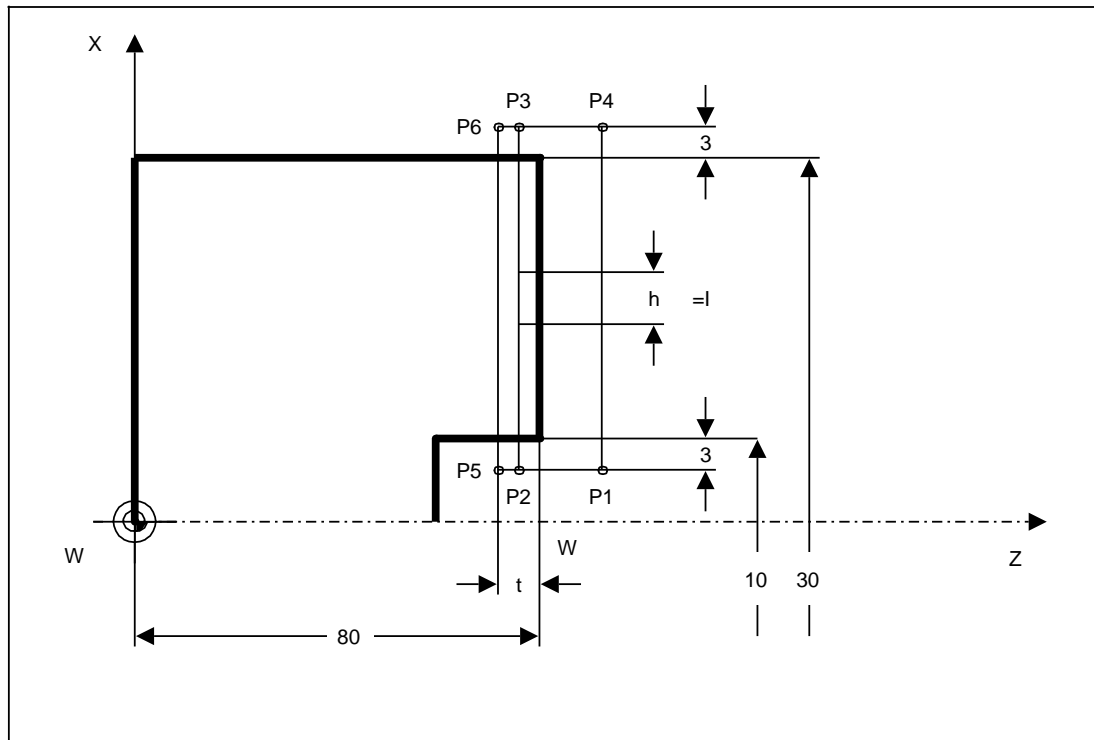
%389

```

(G33 THREAD CUTTING, TAPER) L_F
N31 G90 F1000 S200 M3 L_F
N32 G00 X110 Z170 L_F
N33 X65.86 L_F
N34 G33 X103.366 Z100 K5 L_F
N35 G00 X110 L_F
N36 Z170 L_F
N37 X63.86 L_F
N38 G33 X101.366 Z100 K5 L_F
N39 G00 X110 L_F
N40 M30 L_F
    
```

- Absolute dimension input
- (P1) Approach P1 at rapid traverse
- (P2) Infeed to 1st cutting depth
- (P3) Thread cutting, 1st pass from P2 to P3
- (P4) Retract to P4 in rapid traverse
- (P1) Retract to P1 in rapid traverse
- (P5) Infeed to 2nd cutting depth
- (P6) Thread cutting, 2nd pass from P5 to P6
- (P4) Retract to P4 in rapid traverse

If $\alpha > 45^\circ$, the thread lead in the X direction must be programmed (with parameter l) instead of the lead in the Z direction.

Example: Transversal thread

Transversal thread

Lead $h = 2$ mmThread depth $t = 1.3$ mm

Infeed direction perpendicular to cutting direction I

%390

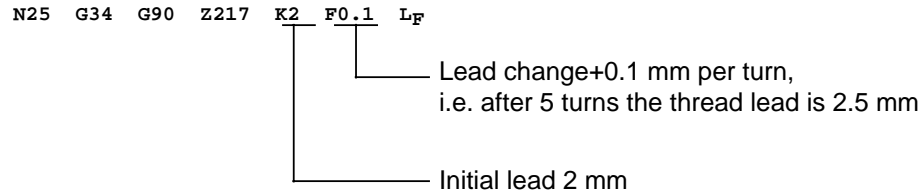
(G33 THREAD CUTTING, TRANSVERSAL) L_F

N41	G90	S50	M3	L _F	Absolute dimension input, spindle speed
N42	G00	X4	Z82	L _F	(P1) Approach P1 in rapid traverse
N43	Z79.35	L _F			(P2) Infeed to 1st cutting depth
N44	G33	X36	I2	L _F	(P3) Thread cutting, 1st pass from P2 to P3
N45	G00	Z82	L _F		(P4) Approach P4 in rapid traverse
N46	X4	L _F			(P1) Approach P1 in rapid traverse
N47	Z78.7	L _F			(P5) Infeed to 2nd cutting depth
N48	G33	X36	I2	L _F	(P6) Thread cutting, 2nd pass from P5 to P6
N49	G00	Z82	L _F		(P4) Retract to P4 in rapid traverse
N50	M30	L _F			

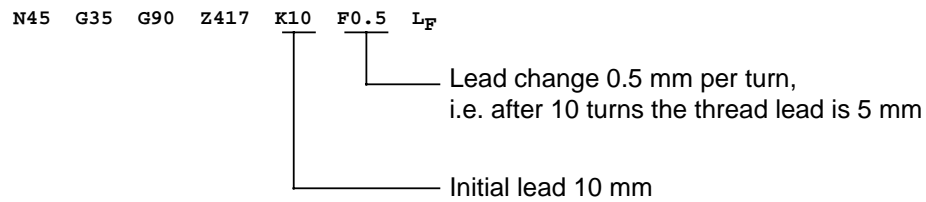
3.2.7.2 Thread with variable lead

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

G34 Lead increase:



G35 Lead decrease:



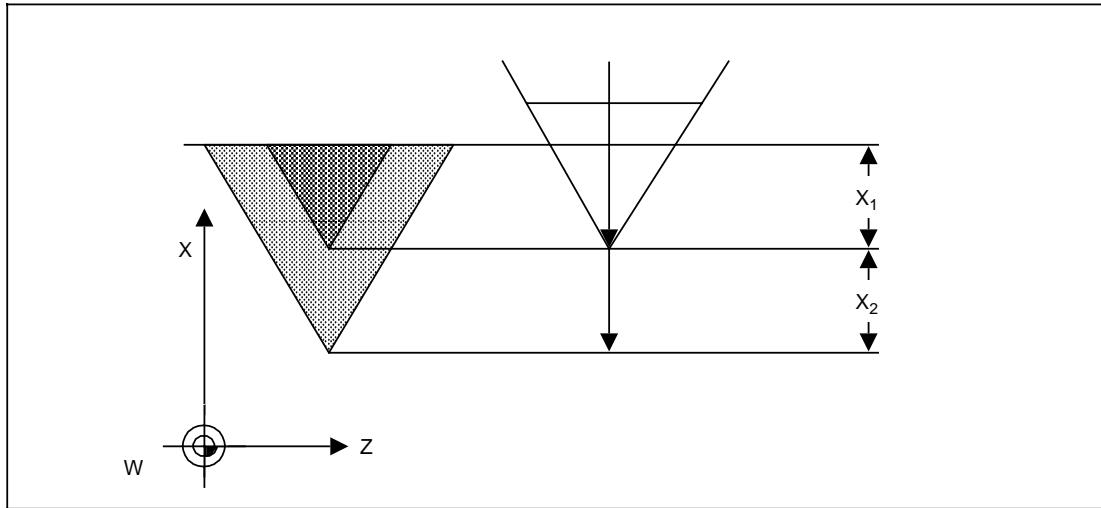
The value **F** is **calculated** from the initial and final leads:

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

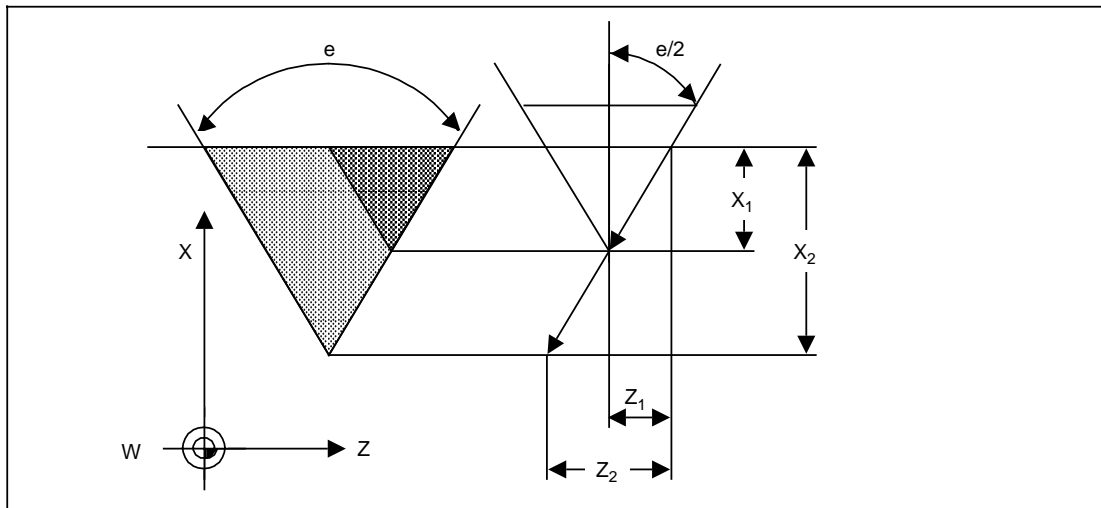
The value must be used without a sign.

3.2.7.3 Infeed options

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



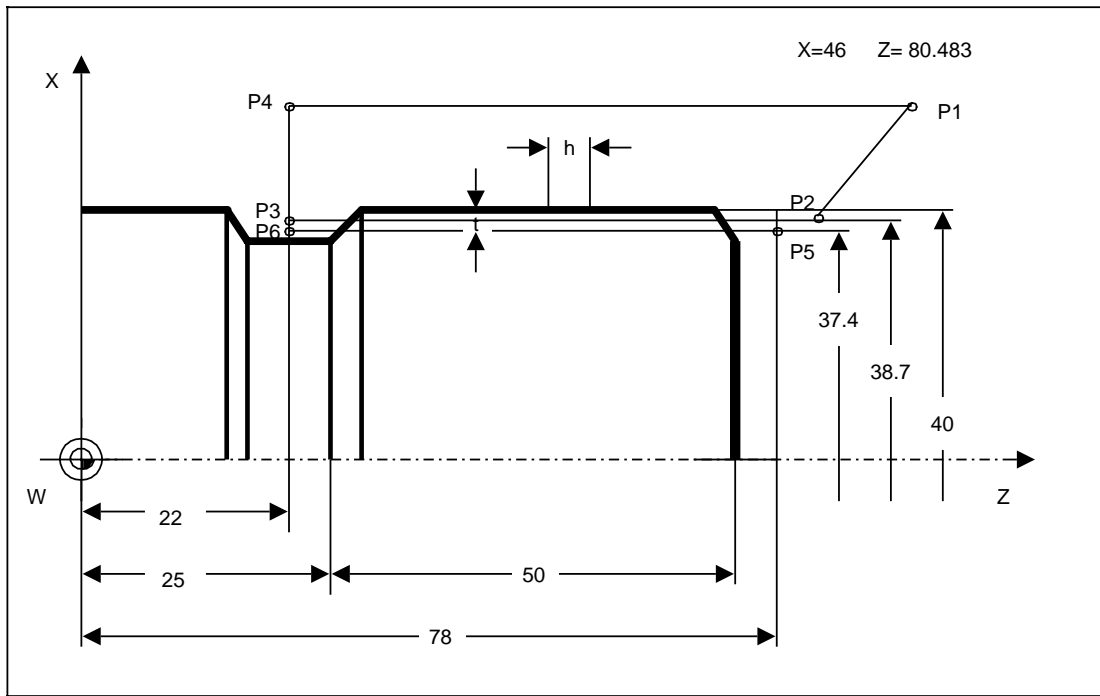
Infeed option "perpendicular to the cutting direction"



"Flank infeed" $Z=X \cdot \tan e/2$ - cutting with one side of the tool

Example: Longitudinal thread with constant lead (flank infeed)

Lead $h = 2$ mm
 Thread depth $t = 1.3$ mm
 Infeed angle $e = 60^\circ$



Flank infeed, longitudinal thread with constant lead

$$x_1 = (46 - 38.7) / 2 = 3.65$$

$$z_1 = X_1 \cdot \tan e / 2 = 2.136$$

$$x_2 = 3.65 + 1.3 / 2 = 4.3$$

$$z_2 = X_2 \cdot \tan e / 2 = 2.483$$

%391

(G33 THREAD CUTTING) L_F

N33 G90 S50 M3 L_F

Absolute dimension input, spindle speed

N34 G00 X46 Z80.483 L_F

(P1) Approach P1 in rapid traverse

N35 X38.7 Z78.347 L_F

(P2) Infeed to 1st cutting depth
 (Z value = thread cutting starting point)

N36 G33 Z22 K2 L_F

(P3) Thread cutting, 1st pass

N37 G00 X46 L_F

(P4) Approach P4 in rapid traverse

N38 Z80.483 L_F

(P1) Approach P1 in rapid traverse

N39 X37.4 Z78 L_F

(P5) Infeed to 2nd cutting depth
 (Z value = Z value from P2 - (Z₂ - Z₁))
 Flank infeed

N40 G33 Z22 K2 L_F

(P6) Thread cutting, 2nd pass

N41 G00 X46 L_F

(P4) Retract to P4 in rapid traverse

N42 M30 L_F

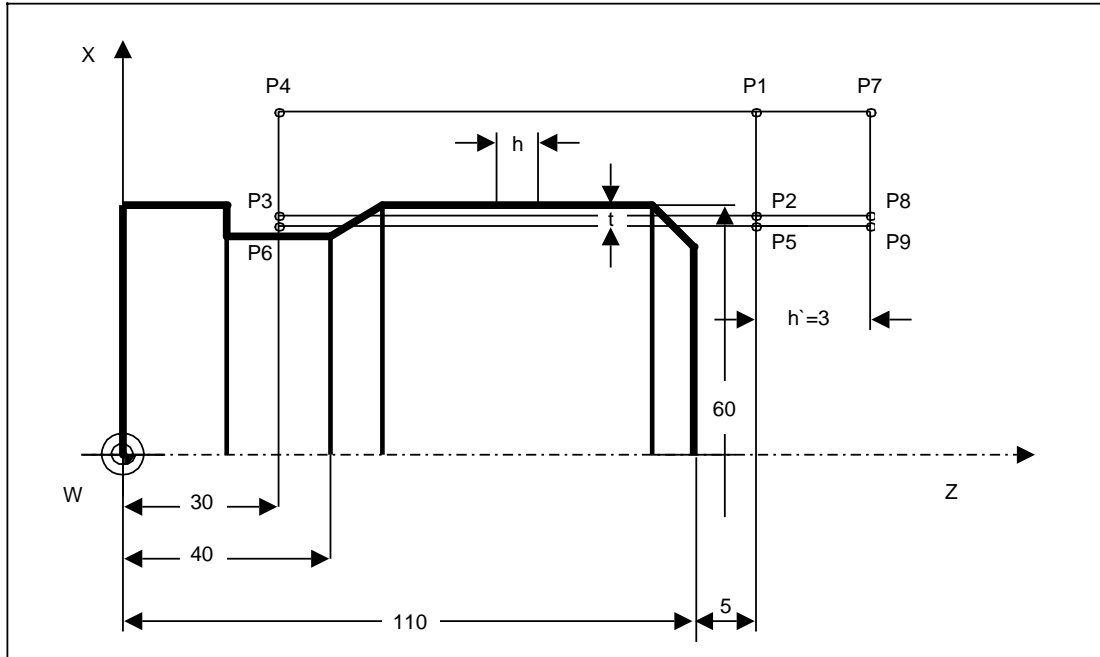
3.2.7.4 Multiple threads

Thread cutting always begins at the synchronization point of the zero mark of the pulse encoder. The **feed** is not **enabled** until 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. Each thread of a multiple thread is programmed in the same manner as a **single thread**. When the first thread has been fully machined, the **starting position is offset by h'** and the next thread is machined.

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

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

Example: Multiple thread with constant lead



Multiple thread with constant lead

Radial infeed direction, lead $h = 6$ mm
 Thread depth $t = 3.9$ mm, 2-thread

In this example each thread is machined in two steps. When the first thread has been fully machined, the second thread is machined by offsetting the starting position by h' .

$$h' = \text{Thread lead/number of threads}$$

$$h' = 6/2 = 3 \text{ mm}$$

The Z coordinate is thus at Z118, not Z115, for the 2nd thread.

%392

(G33, MULTIPLE THREAD) L_F

N35 G90 S50 M3 L_F

N36 G00 X66 Z115 L_F

N37 X56 L_F

N38 G33 Z30 K6 L_F

N39 G00 X66 L_F

N40 Z115 L_F

N41 X52.2 L_F

N42 G33 Z30 K6 L_F

N43 G00 X66 L_F

N44 Z118 L_F

N45 X56 L_F

N46 G33 Z30 K6 L_F

N47 G00 X66 L_F

N48 Z118 L_F

N49 X52.2 L_F

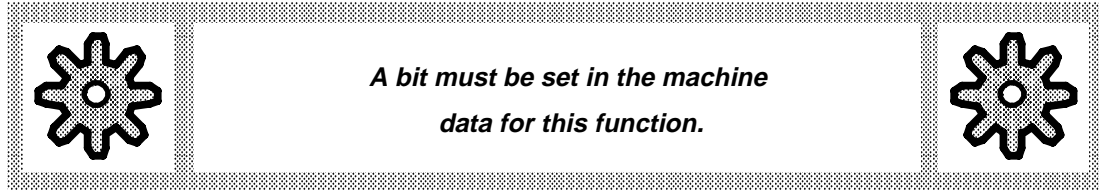
N50 G33 Z30 K6 L_F

N51 G00 X66 L_F

N52 M30 L_F

(P1)	}	1st thread
(P2)		
(P3)		
(P4)		
(P1)		
(P5)		
(P6)	}	2nd thread
(P4)		
(P7)		
(P8)		
(P3)		
(P4)		
(P7)		
(P9)		
(P6)		
(P4)		

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 control. The spindle speed override switch is active.

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

3.2.9 Exact positioning G09, G60, G00, continuous path operation G62, G64

3.2.9.1 Fine and coarse exact stop tolerance ranges G09, G60, G00

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

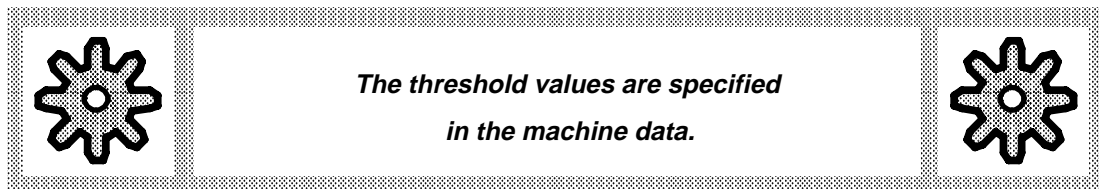
G09, G60 and G00 can be used to approach a target position within a specified **exact stop tolerance range**. When the **exact positioning window** is reached, the feedrate of the traversed axis (from P1 to P2) is reduced to 0. The **following error** is eliminated. At the same time a **block change**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

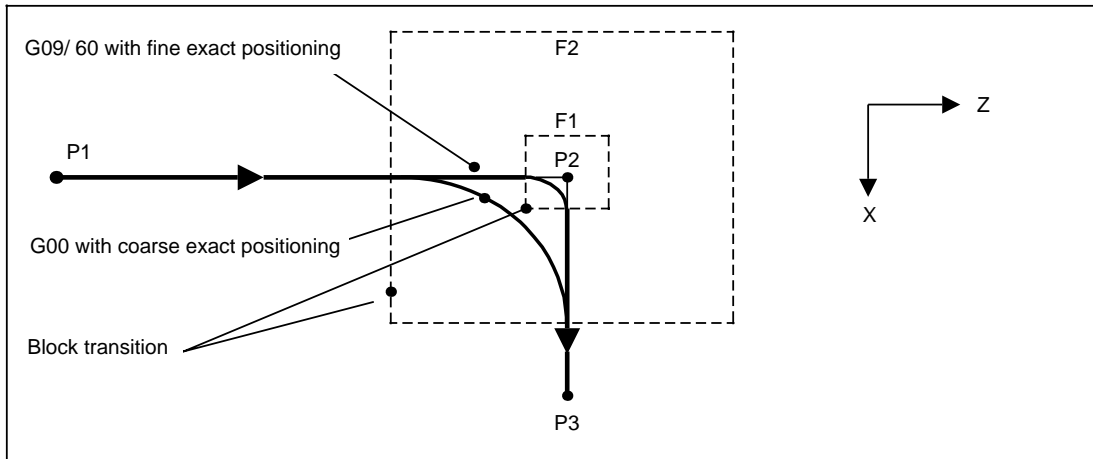
Functions G09 and G60 can be used, for example, for machining **sharp corners**, for **plunge-cutting** or when reversing the direction of movement.

Fine and coarse exact stop tolerance ranges

Fine exact positioning: 10 μ (window F1)
Coarse exact positioning: 25 μ (window F2)

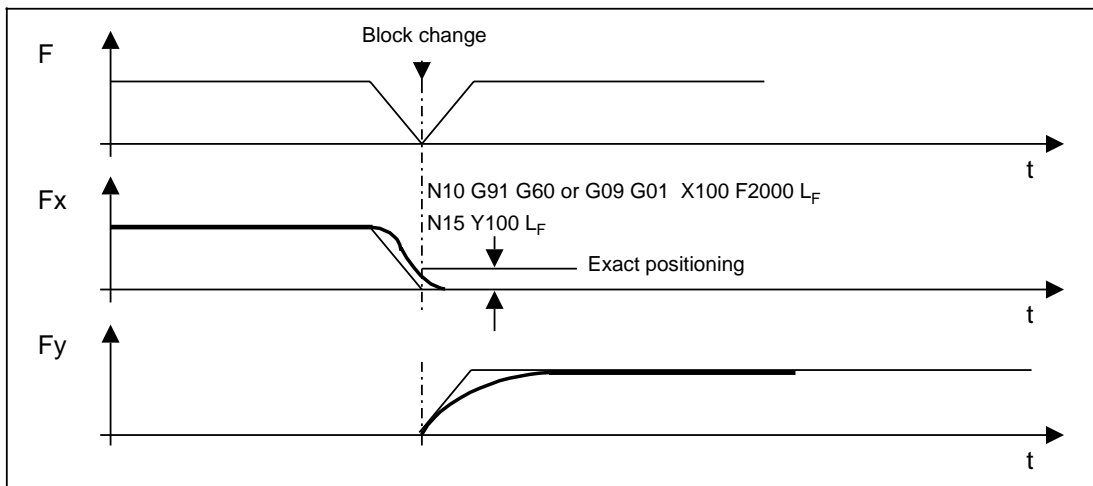


3.2.9 Exact positioning G09, G60, G00, continuous path operation G62, G64



Exact positioning window

If the two exact stop tolerance ranges are identical (window $F1 = \text{window } F2$), 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).

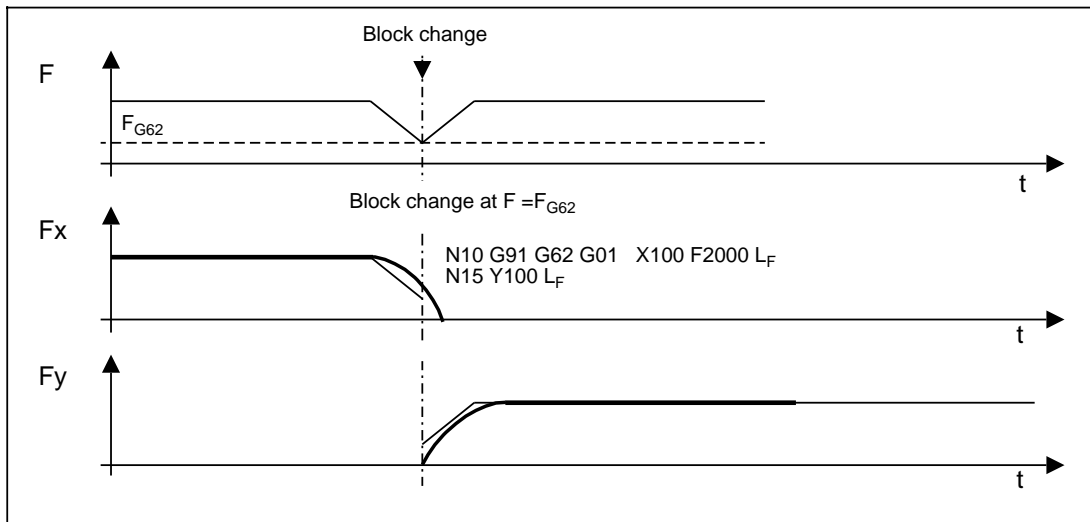


Exact positioning G60, G09

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

3.2.9.2 Continuous path operation G62, G64

Function G62 **reduces the feedrate towards the end of the block** to a rate defined in the machine data **NC MD 3**. The action of G62 is modal.



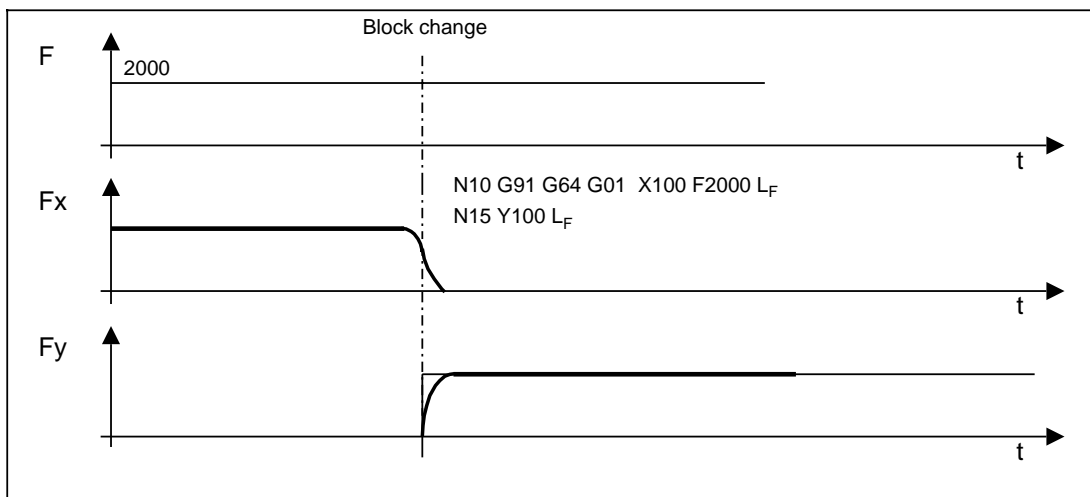
Continuous path operation with G62

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.

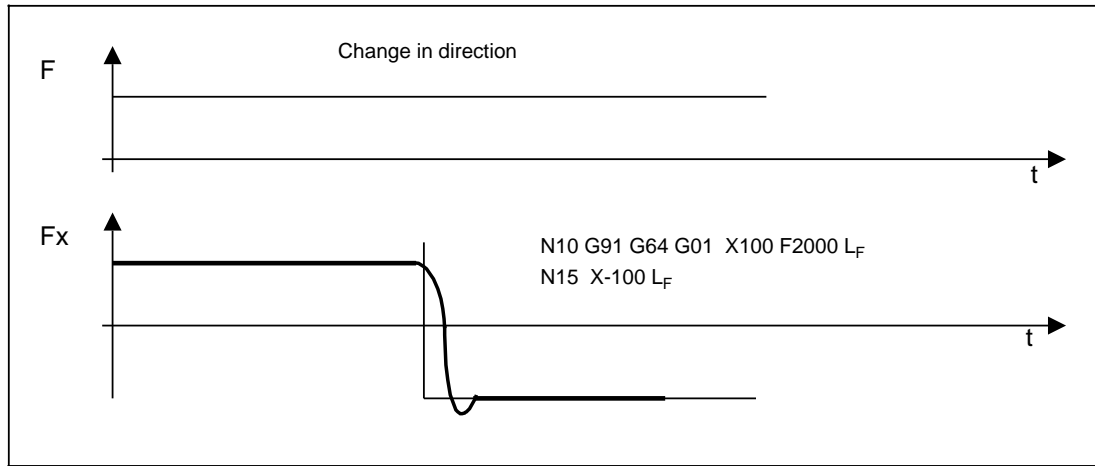
Block transition without feedrate reduction G64:

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. The action of G64 is modal.

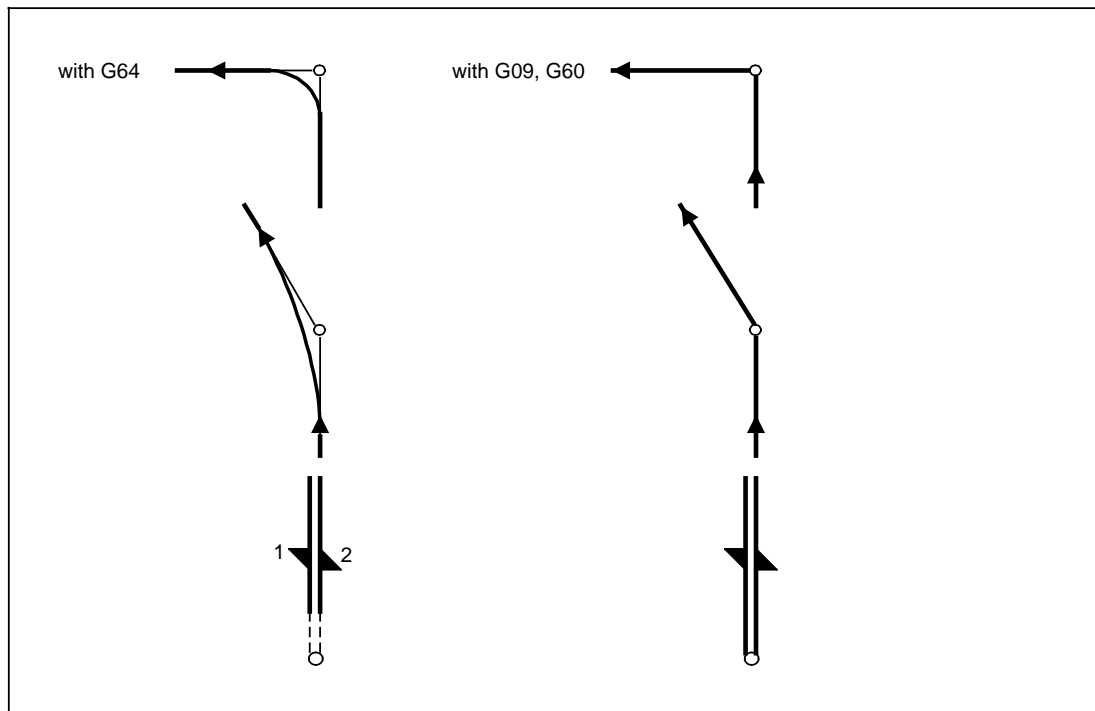


Continuous path operation with G64 without a reduction in feedrate and with different axes

3.2.9 Exact positioning G09, G60, G00, continuous path operation G62, G64

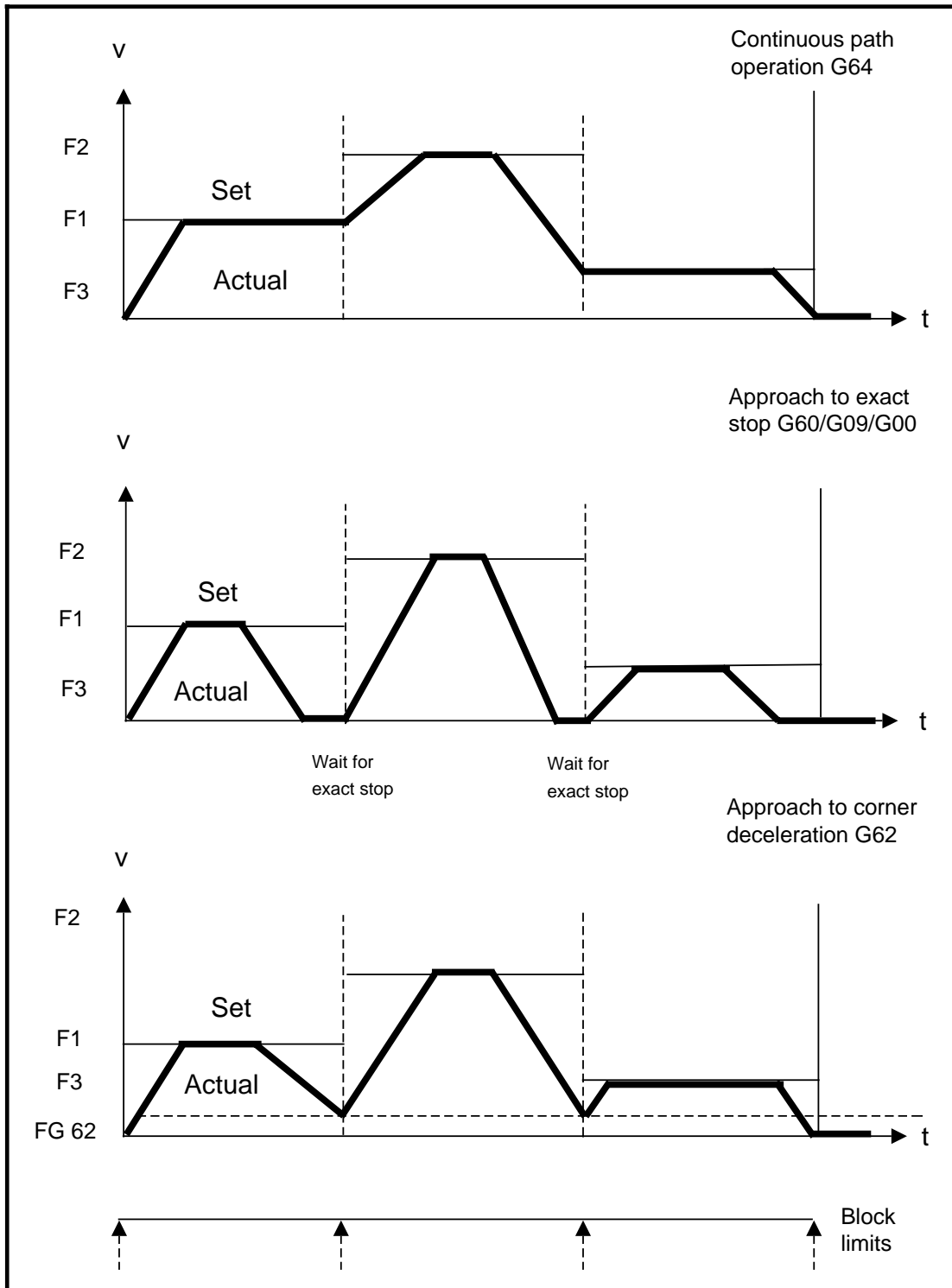


Continuous path operation with G64 and with a change in direction in one axis



Change in direction with and without a reduction in feedrate in one axis

Example: Behaviour of an axis with G00, G09, G60, G62 and G64



Speed behaviour at block limits

3.2.10 Dwell time G04

The dwell time is stated under address **X** or **F**. The **time range** is from

0.001 to 99999.999 s with X and
0.001 to 99.999 s with F and
0.1 to 99.9 spindle revolutions.

G04 is effective **block by block**. **No other functions** can be written in a block with dwell time.

Example:

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

└─ Dwell time 11.5 s always without sign

The dwell time can also be programmed according to the number of spindle revolutions. It is programmed under S in the range from 0.1 to 99.9 revolutions.

3.2.11 Plane selection G16, G17, G18, G19

Plane selection is required for correct **calculation of the tool offset data**. It is possible, therefore, to program circles only with the necessary axis addresses without having to select a plane if no tool offset is selected.

Example: Circle in the X-Z plane

```
N10 G00 X50 Z50 F1000 LF  
N15 G02 X50 Z50 I-10 K-10 LF
```

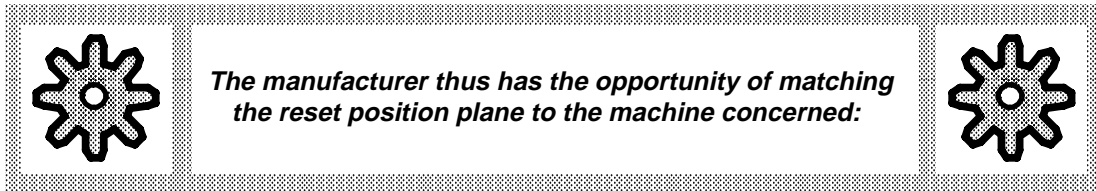
If a tool offset is to be effective in this case, the correct plane also has to be selected.

Example: Circle in the X-Z plane with tool offset D1

```
N10 G00 X50 Z50 G18 D1 F1000 LF (G18 = Z-X-plane selection)  
N15 G02 X50 Z50 I-10 K-10 LF
```

Plane selection G16

The **reset position plane** on the SINUMERIK 805 is **G16** with the plane selection defined by NC MD 5480, NC MD 5500 and NC MD 5520.

**Examples:**

NC MD	5480	5500	5520
Milling machine	X	Y	Z
Turning machine	Z	X	Z
Grinding machine	X	Z	Y

Programming G16 with axis addresses, e.g. G16 Y Z X, offers the option of completely free plane selection. Free plane selection with G16 must stand alone in a block.

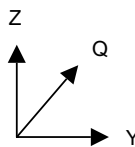
Possibilities:

4-axis machine

Defined axes X Y Z Q

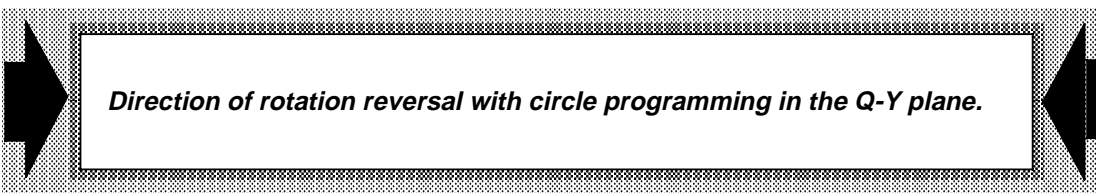
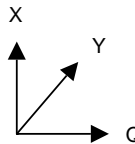
Desired reset position plane: Y Q Z

NC MD 5480 : Y
 5500 : Q
 5520 : Z



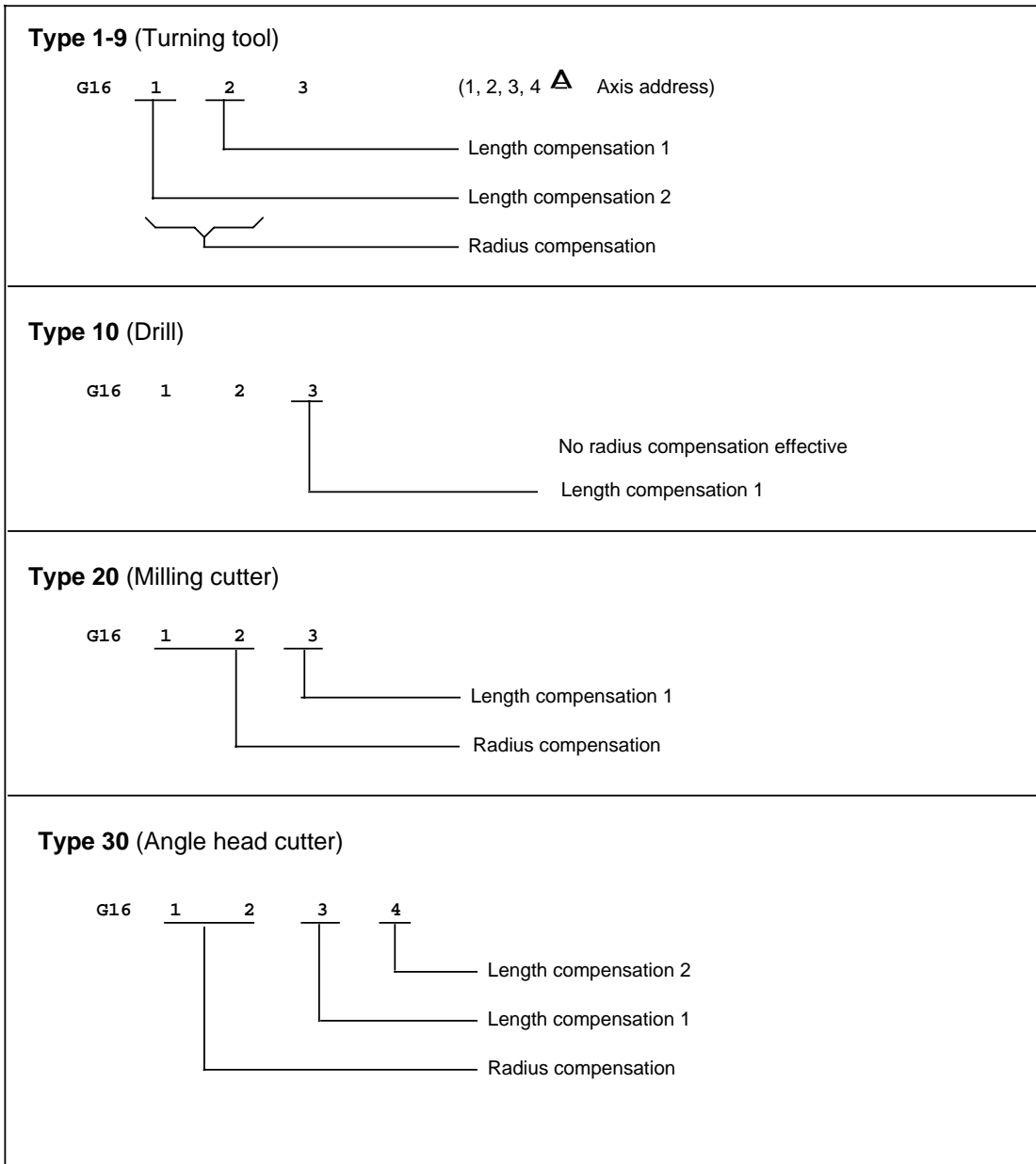
new plane selection:
(programmed in part program)

G16 Q Y X



Inclusion of tool offset with G16 calculation

Depending on the G16 programming and the selected tool type, the **radius** and **length compensation** are taken into account in **other axes**.



G16 has to be programmed with 4 axis addresses with type 30 otherwise length compensation 2 is not effective.

Example:

NC MD 5480 = 0 0 0 0 0 0 0 0 X axis
 NC MD 5500 = 0 0 0 0 0 0 0 1 Y axis
 NC MD 5520 = 0 0 0 0 0 0 1 0 Z axis

D1 = Tool type 30 Angle head cutter
 Radius = 25
 Length 1 = 100
 Length 2 = 80

1st program

N10 G0 G41 D1 X Y Z Radius accounted for in X-Y
 L1 accounted for in Z
 L2 not accounted for

2nd program

N10 G16 X Y Y Z (No traversing movement, only free plane selection)
 N15 G0 G41 D1 X Y Z Radius accounted for in X-Y
 L1 accounted for in Y
 L2 accounted for in Z

3rd program

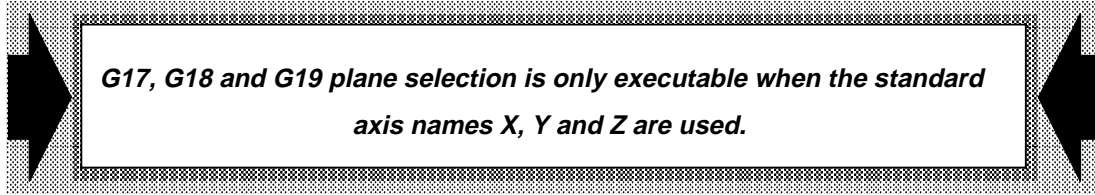
N10 G16 X Z Z Y (No traversing movement, only free plane selection)
 N15 G0 G41 D1 X Y Z Radius accounted for in X-Z
 L1 accounted for in Z
 L2 accounted for in Y

On a turning machine with X and Z axes, Z-X should be selected as the reset position plane after Power On. The NC MD must therefore be set as follows:

NC MD 548* = 0 0 0 0 0 0 1 0
 NC MD 550* = 0 0 0 0 0 0 0 0
 NC MD 552* = 0 0 0 0 0 0 1 0

Plane Z-X is specified by NC MD 548* and 550*.

Plane selection G17, G18, G19



These plane selection options refer to the **first 3 axes** which are displayed in the automatic basic image.

X	0.000	X	A	1st axis
Y	0.000	Y	A	2nd axis
Z	0.000	Z	A	3rd axis
Q	0.000			

G17 always means	
G18 always means	
G19 always means	

G17, G18 and G19 are always programmed **without axis addresses**.
 The plane is assigned **automatically**.

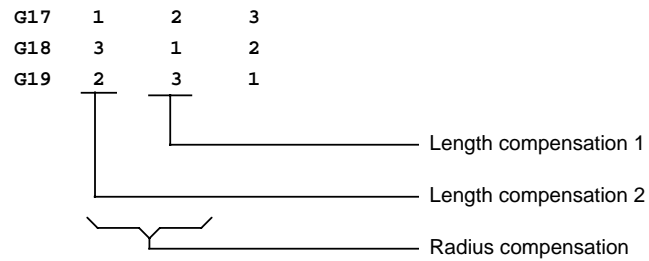
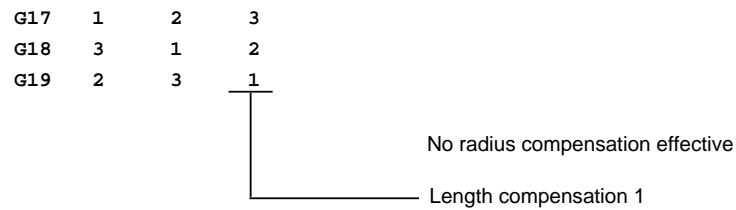
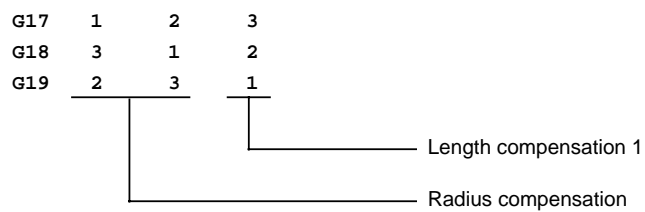
Example:

```

:
:
N10 G18 L_F
:
:
    
```

Inclusion of tool set with G17, G18, G19

- 1 **A** 1st axis (Standard: X)
- 2 **A** 2nd axis (Standard: Y)
- 3 **A** 3rd axis (Standard: Z)

Type 1-9 (Turning tool)**Type 10 (Drill)****Type 20 (Milling cutter)**

Type 30 cannot be used when programming with G17, G18, G19.

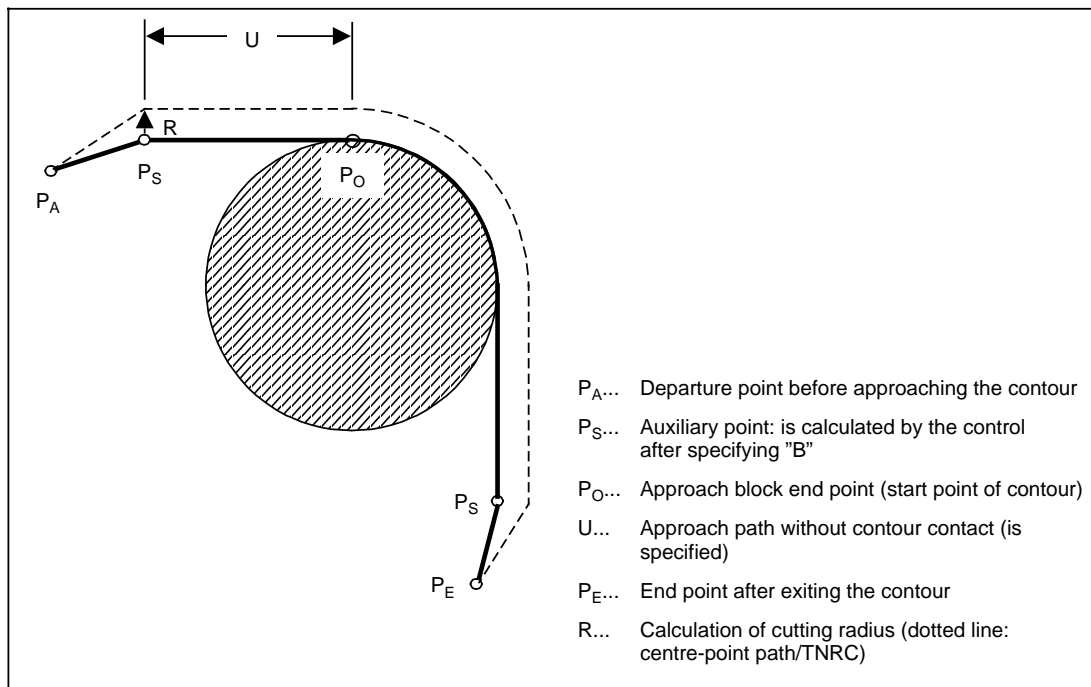
3.2.12 Soft approach to and exit from the contour

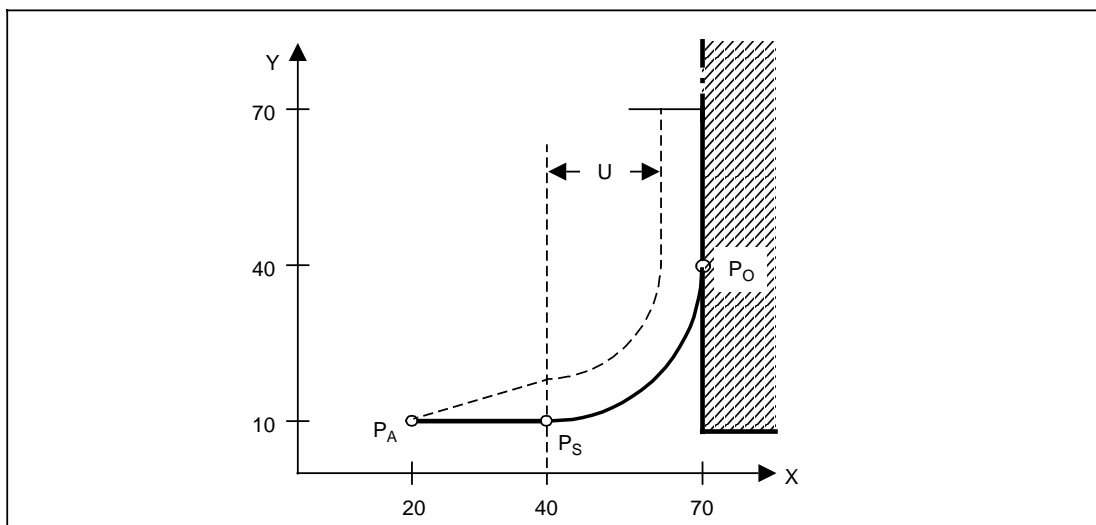
To avoid cutting marks, a contour is approached and exited **tangentially**.
Soft approach to and exit from a contour can be used for all **types of tools**. "Radius" and "Wear" are calculated.

Approach of the contour and **exiting** the contour can be programmed with the following functions:

- G147 Approach linear
- G247 Approach in quarter circle
- G347 Approach in semicircle
- G148 Exit linear
- G248 Exit in quarter circle
- G348 Exit in semicircle
- G48 Exit the contour as it was approached

Example: Straight to a circle



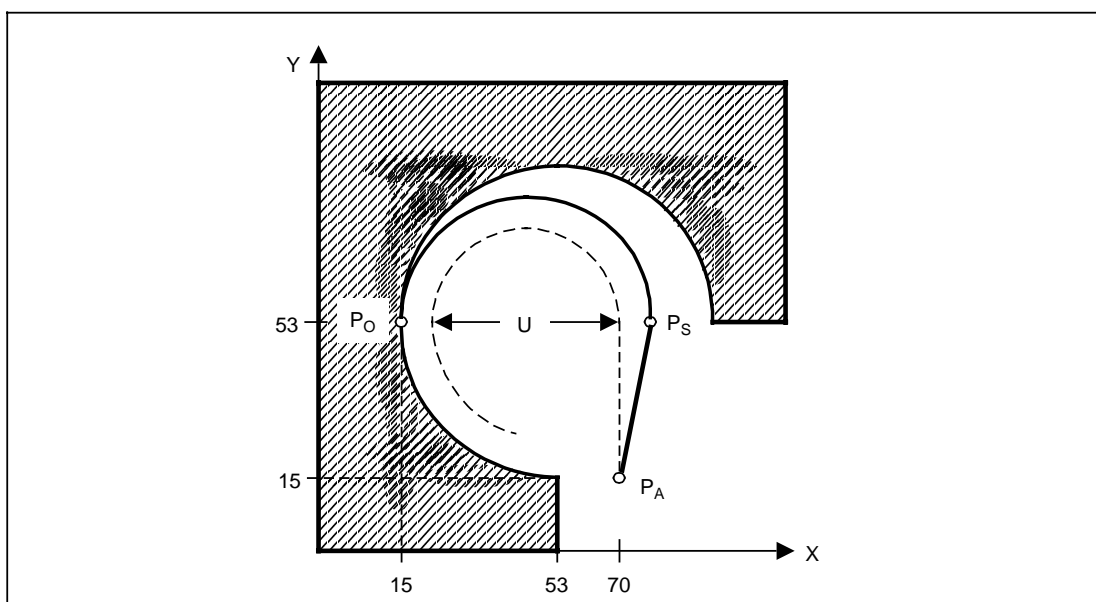
Example: Approaching a linear contour in a quarter circle (with programming example)

Approaching a linear contour in a quarter circle

```

:
N20 G01 G41 Y10 X20 D1 F1000 G41 L_F      Select TNRC/CRC left (G41) select tool offset (D1)
N25 G247 Y40 X70 U25 L_F                 Select "Approach in quarter circle" (G247)
N30 G01 Y70 L_F                           Exit from the linear contour
N35 M30 L_F                               Program end

```

Example: Approaching an inside circle in a semicircle (with programming example)

Approaching an inside circle in a semicircle

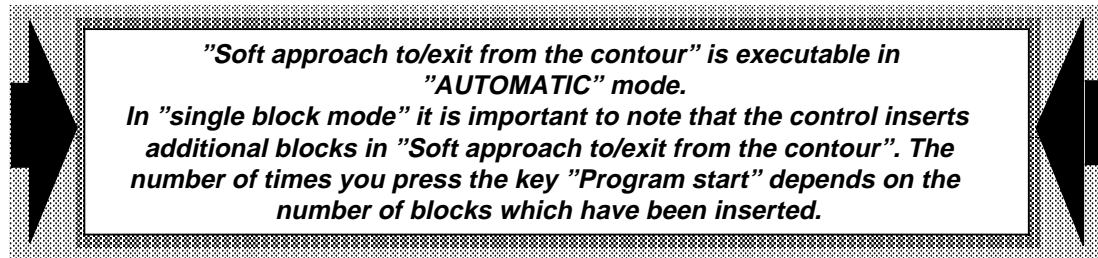
```

:
N10 G01 G41 Y15 X70 D1 F1000 L_F        Select TNRC/CRC left (G41) select tool offset (D1)
N15 G347 Y53 X15 U50 L_F                 Select "Approach in semicircle" (G347)
N20 G03 Y15 X53 I38 L_F                   Exit from the contour in quarter circle
N25 G0 Y15 X70 L_F                       Approach P_A in rapid traverse
N30 M30 L_F                               Program end

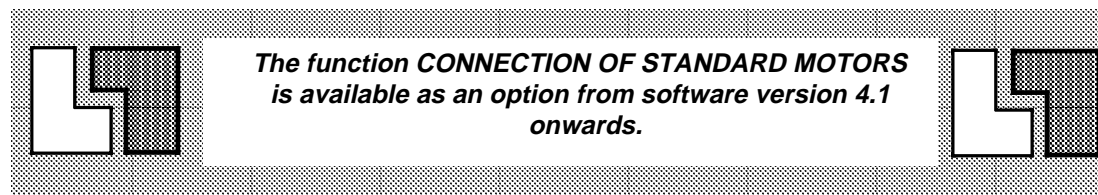
```

Programming characteristics

- The functions for approaching and exiting the contour are only active in **block** in which they entered.
- The following must be specified in the approach block (for G147, G247, G347):
 - the coordinates of the starting point P_0 of the contour
 - the value for U (approach path without contour contact).
- The following must be specified in the exit block (for G148, G248, G348, G347):
 - the coordinates of the end point P_E after exiting the contour
 - the value for U (exit path without contour contact).
- No other traversing movements must be programmed in a block with "G47" or "G48".
- Auxiliary functions can be programmed in the approach block as well as in the exit block.
- No pure auxiliary function block may be programmed **after** an approach block or **before** an exit block.
- When combining approach blocks and exit blocks, care should be taken that in the exit block "G40" (deselection of TNRC) is generated by the control. Program **G41** before every approach block.



3.2.13 Axes with standard motors



Axes with standard motors cannot be programmed with as many functions as normal NC axes.

Axes with standard motors cannot be interpolated with other axes. The movement of a standard motor axis cannot be dependent on the movement of other axes or spindles (as, for example, in circular movement or continuous-path mode).

The programmed feed or rapid traverse is not valid for standard motor axes, which can only be traversed at high speed or creep speed (depends on the construction of the axis (type or motor, gearing etc.)).

Standard motor axes are programmed in the same way as conventional NC axes and are subject to the same restrictions.

Example:

```

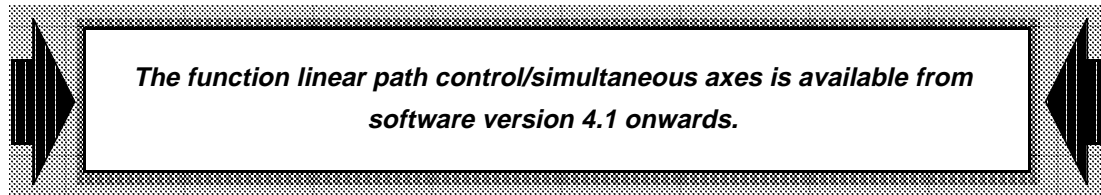
.
.
.
N500 G01 X200 Y-250 Z100 F50 L_F
.
.
.

```

X and Y are standard motor axes.
Z is a normal axis. All three axes start to traverse at the same time, the Z axis traversing with feed F50. The axes can arrive at the programmed position at different times. Only when all the axes are in position can the next block be processed.

A standard motor axis cannot be programmed with the following functions:

- G02/G03 Circular interpolation
- G12/G13 Polar coordinate programming circular interpolation
- G16-G19 Plane selection with standard motor axis
- G33/G34/G35 Thread cutting
- G41/G42 Cutter radius compensation/tool nose radius compensation
- G09/G60 Velocity reduction exact stop
- G62/G64 Continuous-path mode
- G68 Absolute dimensioning rotary axis
- G92 Spindle speed setpoint limitation
- G94-G97 Feedrate control
- G110/G111 Polar coordinate programming centrepoint
- Gx47/Gx48 Soft approach to and exit from contour

3.2.14 Linear path control/simultaneous axes

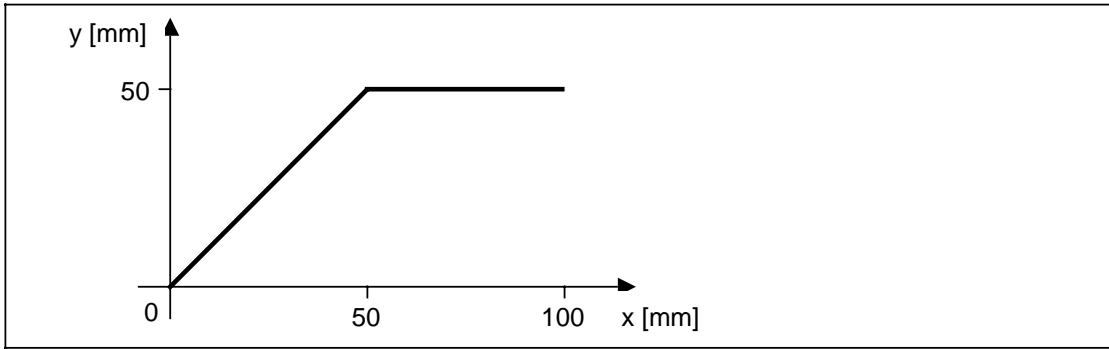
Any NC linear axis can be programmed as a simultaneous axes. An axis can be defined as an interpolating axis or a simultaneous axis block by block.

Axes programmed as simultaneous axes cannot interpolate with each other or with other axes.

Each simultaneous axis traverses at its own simultaneous feedrate, i.e. axes which are programmed as simultaneous axes in one block begin to traverse at the same time but can reach the programmed end point at different times (linear path control).

Example:

```
N5 G0 G90 X0 Y0 LF  
N10 G1 X100 F500 Y50 F1000 LF (simultaneous axes)  
.  
.  
.
```



Up to three axes can be programmed at the same time in one NC block. These can be either interpolating axes or simultaneous axes or both.

Identification and programming of simultaneous axes

Simultaneous axes are programmed by entering the simultaneous feedrate value directly after the programmed axis value.

If one axis is programmed on its own without a feedrate value, the last path feedrate to have been programmed remains valid.

If only one axis together with a feedrate value is programmed in a block, this axis is regarded as an interpolating axis (not as a simultaneous axis) and the feedrate value is regarded as the new path feedrate.

Example:

```
N20 G90 G01 X100 F500 LF  
X is an interpolating axis, F500 becomes the path feedrate.
```

If several simultaneous axes are programmed in one block, the first of these is regarded as an interpolating axis, i.e. all compensations (tool nose radius compensation, cutter radius compensation) are active for this axis. The feedrate for this axis is a path feedrate (modal). Otherwise, this axis behaves in the same way as the other simultaneous axes.

Example:

```
N30 G90 G01 X100 F300 Y250 F1500 Z50 F100 LF  
X is an interpolating axis, F300 becomes the path feedrate;  
Y and Z are simultaneous axes
```

If interpolating and simultaneous axes are to be programmed in one block, the interpolating axes must always be programmed before the simultaneous axes.

Example:

```
N30 G90 G01 X100 Y100 F333 Z100 F1000  
X and Y interpolate with each other, F333 is the path feedrate, Z is a simultaneous axis with a feedrate of F1000.
```

Programming examples:

Empty field means:
axis traverses with path feedrate if the
axis is programmed in the block

		F_{path}	F_{SIM}			
			F_X	F_Y	F_Z	F_Q
N1	G01 X.. Y.. Z.. F1000	1000				
N2	X.. F900	900				
N3	G02 Z.. Q..	900				
N4	G01 Y.. Z..	900				
N5	Q..	900				
N6	Y.. F800 Z.. F700 Q.. F600	800			700	600
N7	X.. Y.. F1100 Q.. F500	1100				500
N8	Y.. F800 Z.. F700 Q..					
		Alarm: Feedrate missing/incorrect, axis Q without simultaneous feedrate				
N10	F1200	1200				
N11	Q..	1200				
N12	X.. Y..	1200				
N13	F1400 Y.. Z..	1400				
N14	Z.. Q..	1400				
N15	F1200 X.. Y.. Z.. Q..	1200				
N16	X.. F900 Y.. F800 Z.. F700	900		800	700	
N17	G02 X.. Y.. I.. J.. F700 Z.. F5000	700			5000	
N18	G02 X.. Y.. F1000	1000				
N19	G02 X.. Y..	1000				


The axes for which no simultaneous feedrate F_{SIM} has been entered interpolate at F_{path} .

The following applies to a programmed simultaneous feedrate:

- It is non-modal
- It only applies to the axis programmed directly before it

If there are several simultaneous axes programmed in one block, compensations (tool nose radius compensation, cutter radius compensation, tool offset etc.) only apply to the first axis programmed in the block (this axis is interpreted as an interpolating axis).

Zero offsets, tool length compensation, PRESET, DRF and mirroring are active with simultaneous axes.

	WARNING
	<p>Only the 0% position on the feedrate override switch affects the simultaneous axes.</p> <p>In all other positions, the programmed simultaneous feedrate is active.</p>

Any functions which would cause a simultaneous axis to interpolate with another axis or cause the traversing movement of the simultaneous axis which is dependent on the spindle are not possible.

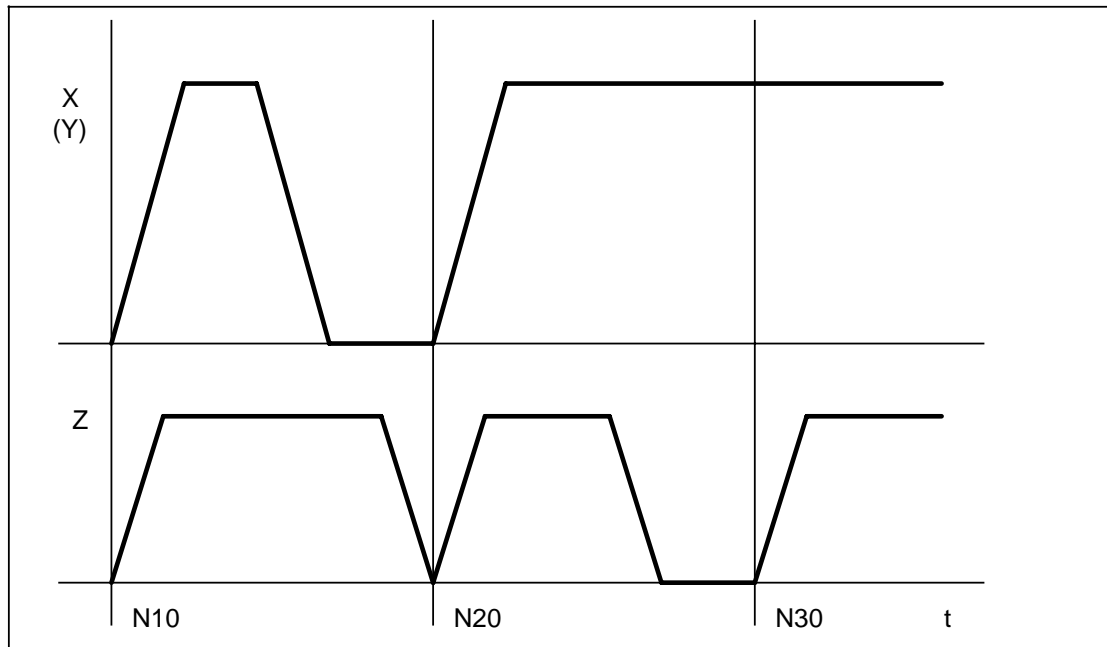
Simultaneous axes must not be programmed together with G00 (rapid traverse).

Simultaneous axes with G62/G64

Block change only takes place when all the programmed axes have reached the end point. This means that even interpolating axes that have been programmed with G62/G64 stop at the end of the block if the simultaneous axes have not yet reached their end point. This can lead to problems with G33.

Example: Operation with G64

```
G64 G01
N10 X80 Y80 F1000 Z100 F700
N20 X180 Y180 F1000 Z150 F700
N30 X250 Y300 Z200 F1000
```



Velocity diagram showing operation with G64; in block N10, axes X and Y brake in spite of G64 because simultaneous axis Z has not yet reached its end point. At the end of block N20, the simultaneous axis Z has already stopped so that axes X and Y can continue to move.

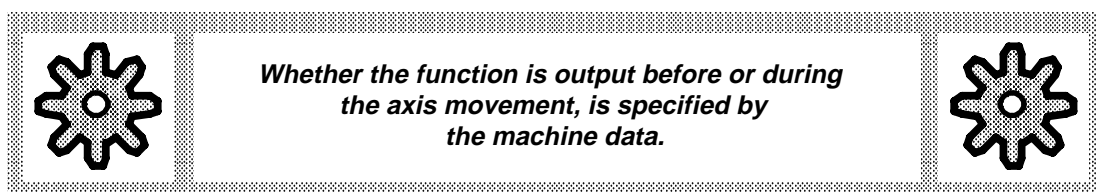
4 Switching, Auxiliary and Miscellaneous Functions

4.1 M, S, T, H

The miscellaneous functions contain primarily technological specifications which are not programmed in the words provided with address letters F, S, and T:

- Miscellaneous function M (2- digit)
- Spindle speed S (5- digit)
- Tool number T (4- digit)
- Auxiliary function H (4- digit)

Each block can contain up to three M functions, one S function, one T function and one H function. They are output to the interface control in the following sequence: M, S, T, H.



If the functions are output during axis movement, the following applies:

If a new value must be active before the axes are traversed, the new function must be written in the preceding block.

4.2 Miscellaneous function M

The miscellaneous functions are partly defined by DIN 66 025/Part 2, and partly by the tool machine manufacturer.

M00 Programmed stop (unconditional)

Function M00 permits the program to be interrupted, e.g. in order to perform a measurement. On termination of the task, machining can be continued by pressing the "NC START" key. The information entered is retained. Miscellaneous function M00 is active in all automatic operating modes. Whether or not the spindle drive is also stopped must be determined from the specific Programming Guide for each machine. M00 is also active in a block without position data.

M01 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 the softkey or at the interface.

M02 End of program

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 control is reset.
M02 can 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 End of subroutine

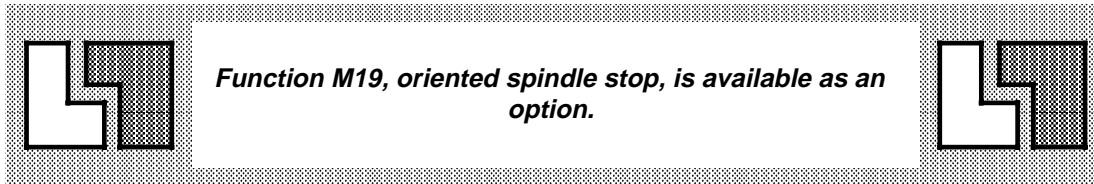
M17 is written in the last block of a subroutine. It can be written in a separate block or in a block containing other functions. M17 must not be written in the same block as a subroutine (nesting).

M30 End of program

Function M30 is the same as M02.

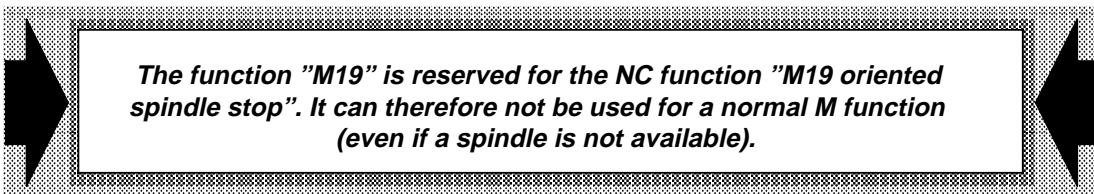
M03, M04, M05, M19 Main spindle control

(M19 only with pulse encoder at main spindle)



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

M03	Clockwise spindle direction of rotation
M04	Counterclockwise spindle direction of rotation
M05	Non-oriented spindle stop
M19	Oriented spindle stop



M19 S.. can be used to perform an oriented main spindle stop. The relevant angle is programmed under S (within the range 0.5° to 359.5°). The angle is measured from the zero mark in the clockwise direction of rotation.

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 can also be input via the operator panel under "Spindle setting data". A machine data 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.

M36, M37 Reduction of feedrate 1:100

M36 The programmed feedrate is valid again.
cancel M37

M37 The programmed feedrate is reduced by 1:100 (is not active with revolutionary feedrate G95).

Example:

```

:
N20 X32 F1000 L_F           Feedrate 1000 mm/min
:
:
N40 M37 L_F                 Feedrate 10 mm/min
:
:
N50 M36 L_F                 Feedrate 1000 mm/min
:
:

```

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.

Further details on the use of the various functions can be found in the specific machine program key. 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
or cutting speed in m/min or 0.1 m/min *)
- Cutting speed in rev/min or 0.1 rev/min*)
- Spindle speed limitation G92 S in rev/min or 0.1 rev/min*)
- Spindle stop in degrees
- Spindle dwell in revolutions (see G04)

**Spindle speed limitation G92 S..**

With constant cutting rate G96, it may be necessary to prevent the spindle speed from increasing, i.e., to continue machining at a constant speed once a certain limit is reached. The speed limitation is programmed under address S in rev/min in a separate block before the program part in which the machining is programmed. The function G92 S.. can be written into the program.

Example:

```
N10 G92 S300 L_F           (Spindle speed limited to 300 rev/min)
```

No additional commands must be written into a block with G92. The limitation has no effect with G94 or G95.

Spindle speed limitation is also deselected with G92 S.., the maximum speed of the selected gear stage being written in S.. . The spindle is brought to a stop with G92 S0.

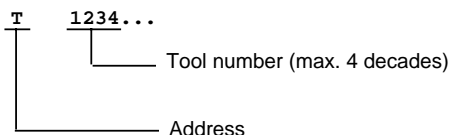
*) The speed and cutting speed must be programmed in the same input format.

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 up to 4 decades. The meaning of the functions is described in the Programming Guide of the machine tool manufacturer.

4.5 Tool number T

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



4.6 Special auxiliary functions

Special auxiliary functions are functions which are not displayed in the operator interface window AUXILIARY FUNCTIONS and which cannot be overstored. The special auxiliary functions can be programmed in addition to the auxiliary functions (max. 6 auxiliary functions and, additionally, the special auxiliary function can be programmed in a block).

By means of the special auxiliary functions the output voltages (+10 V to -10 V) can be programmed for the "Analog setpoint output" function. For this purpose, the special auxiliary functions H11=, H12= and H13= are used, and the following assignments are made:

H11= <Voltage>	Analog output 1
H12= <Voltage>	Analog output 2
H13= <Voltage>	Analog output 3

The voltage depends on a machine data programmable in millivolt (mV) with 5 digits or in volt (V) with 2 digits before the comma and three decimal places.

Example:

H11=2,321 Analog output 1 is assigned the voltage 2.321 (V)
 (NC MD setting:(V))
H13=3271 Analog output 3 is assigned the voltage 3.271 (V)
 (NC MD setting:(mV))

Notes:

The "Analog setpoint output" function is described in more detail in the Installation Instructions, Section 11.12.

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.

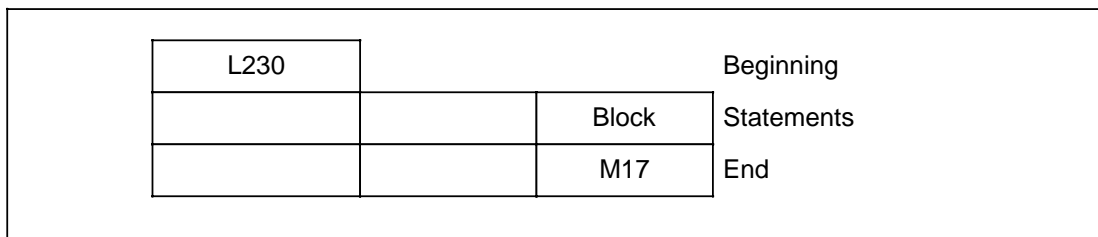
The program memory can be used to store a standard of 50 (max. 200) part programs and subroutines simultaneously.

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

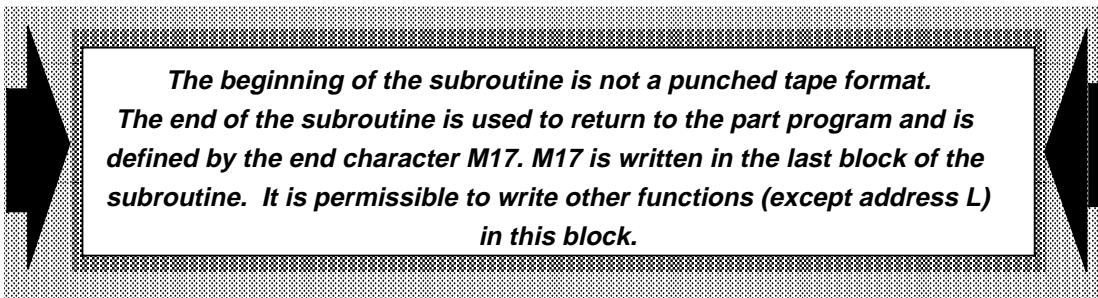
A subroutine comprises:

- the beginning of the subroutine,
- the subroutine blocks,
- the end of the subroutine



Subroutine structure

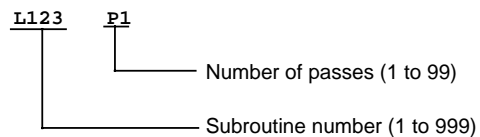
The beginning of the subroutine comprises address L and the three-digit subroutine number (see program key).



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, it is automatically assumed that the number of passes is P1 (1 run).

Example:



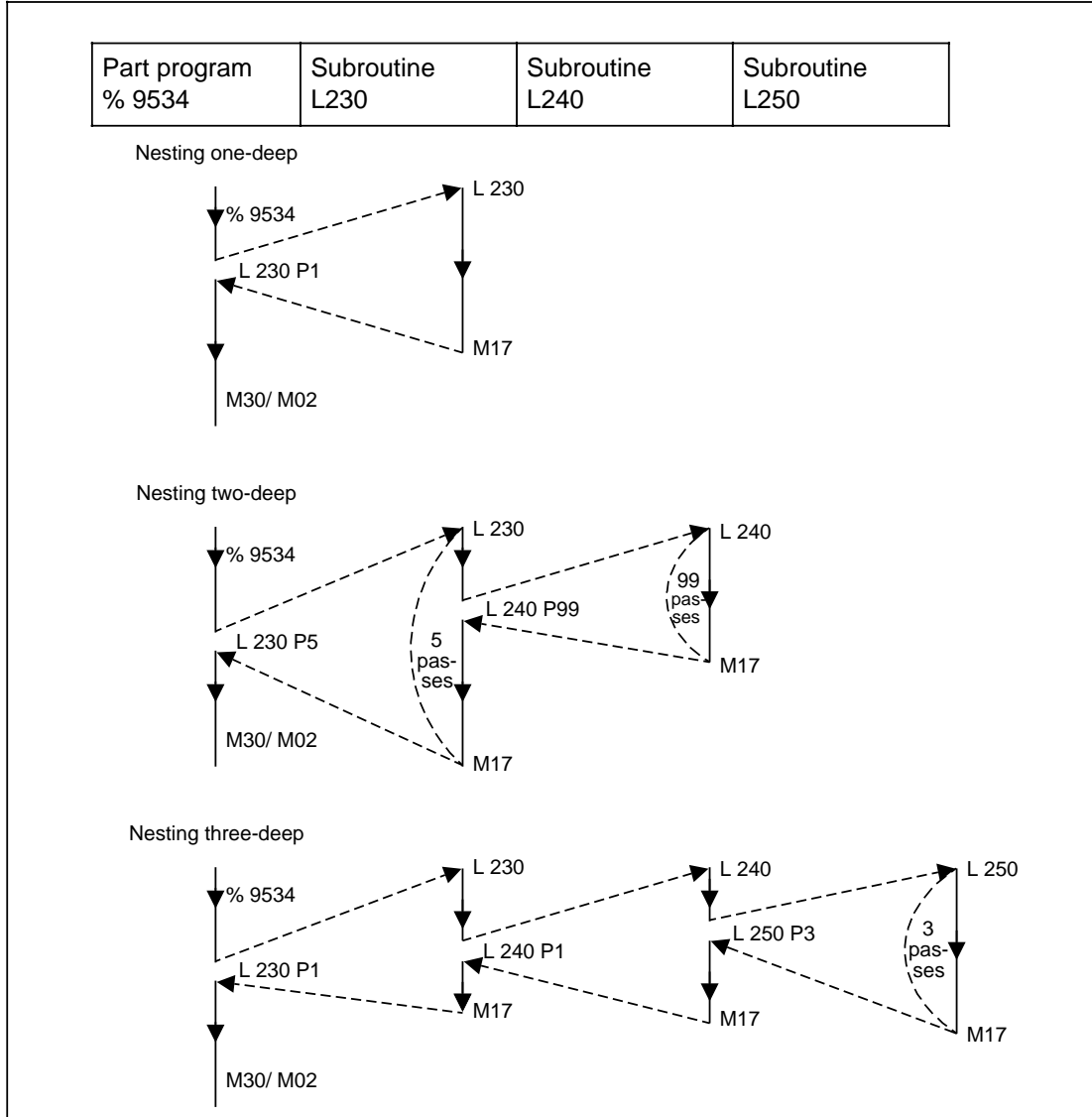
The following should be noted during programming:

- The subroutine call must not be written in a block together with M02, M30 or M17.
- 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.

5.4 Subroutine nesting

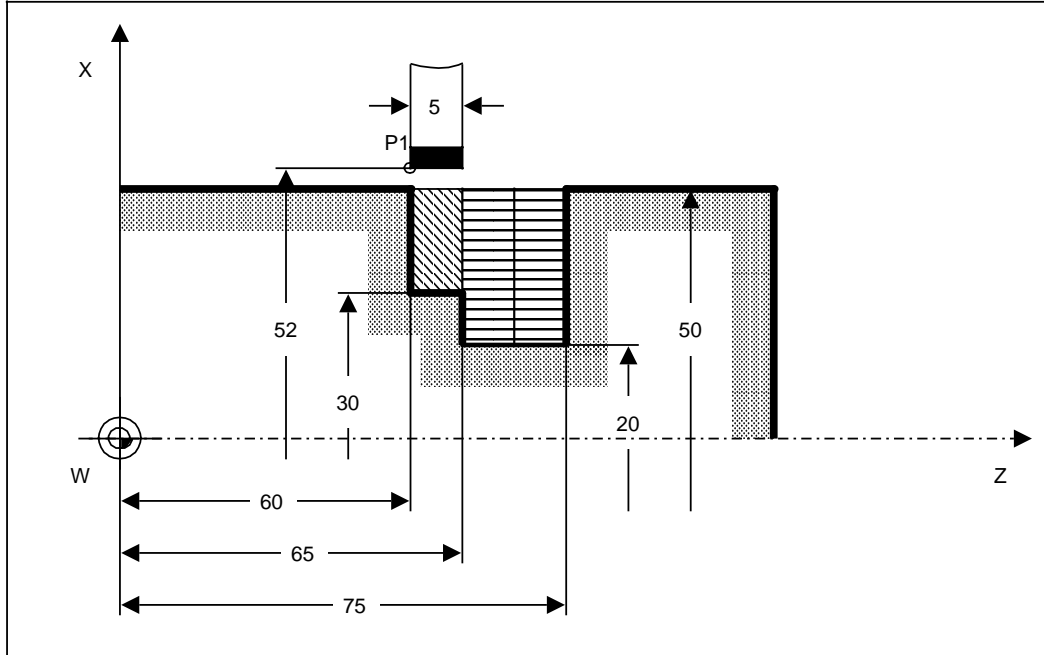
Subroutines can be called not only from a part program, but also from other subroutines. This process is referred to as subroutine nesting.

Nesting of the subroutine to a depth of three is possible.





Subroutine call and subroutine nesting

Example: Subroutine nesting



Subroutine nesting

<code>% 4011</code>	Program header % 4011 (main program)	
<code>N5 G90 G94 F500 S2000 M3 T3 LF</code>	Select absolute dimension, feedrate m/rev, S, T, D, M function	
<code>N10 G00 X52 Z60 LF</code>	Approach P1 at rapid traverse	
<code>N15 L230 P1 LF</code>	Call subroutine L230 with 1 pass	
<code>.</code>		
<code>.</code>		
<code>N90 M30 LF</code>	End of program	
<code>L230</code>	Program header L230 (subroutine)	
<code>N5 G91 G01 X-11 LF</code>	Program part for turning off segment	
<code>N10 G09 X11 LF</code>	Program part for turning off segment	
<code>N15 L240 P2 LF</code>	Call subroutine L240 with 2 passes	
<code>N20 M17 LF</code>	End of subroutine L230	
<code>L240</code>	Program header L240 (subroutine)	
<code>N5 G91 G00 Z5 LF</code>	Program part for turning off segment	
<code>N10 G01 G09 X-16 LF</code>	Program part for turning off segment	
<code>N15 G00 X16 LF</code>	Program part for turning off segment	
<code>N20 M17 LF</code>	End of subroutine L240	

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, 1000 parameters are available to the control; they are subdivided into various areas.

All addresses except for N can be assigned a parameter instead of a value.

N5 Z=R5 L _F

R parameter No.	Permanently assigned	Function
R0 . R49	As long as standard cycles are being processed	Transfer parameters from standard cycles
R50 . R99	As long as standard cycles are being processed	Local R parameters. The cycle calculation is performed with these R parameters.
R100 . R109	yes	Reserved for Siemens cycles
R110 . R199	yes	Reserved for measuring cycles. If no measuring cycles are used, this area is at the free disposal of the user.
R200 . R499	yes	These R parameters are used internally (when using CL800)
R500 . R699		Reserved for Siemens-Nürnberg.
R700 . R999		User assignable

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 LF
```

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
N5...
.
.
.
N85 R1=10 R29=-20.05 R5=50 LF           Parameter definition
N90 L51 P2 LF                           Call subroutine L51, two passes
N95 M02 LF

L51
N5 Z=-R5 B=-R1 LF
N10 X=-R29 LF
.
.
.
N50 M17 LF
```

6.3 Parameter calculations

Logic operations with parameters

All four basic arithmetic operations are possible with parameters. The sequence of the logic operations determines the result of the calculation. The rule whereby multiplication and division are performed before addition and subtraction does not apply here.

Arithmetic operation	Programmed operation	Result in
Definition	R1=100	R1
Assignment	R1=R2	R1
Negation	R1=- R2	R1
Addition	R2=R2+R3	R2
Subtraction	R2=R2 - R3	R2
Multiplication	R2=R2 · R3	R2
Division	R2=R2 / R3	R2

The result of an operation is written in the first parameter of an equation; its initial value is thus overwritten 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 applies:

$R1=R3$

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. The following sequence must be used: Address, numerical value, parameter. Calculation signs must be written. No sign signifies a positive number.

$Y=10+R100$

The calculation sign „+” must always be written.

Example:

```
N5  R1=9.7 R2=-2.1 LF
N10 X=20.3+R1 LF
N15 Y=32.9-R2 LF
N20 Z=19.7-R1 LF
```

Result:

```
X=30
Y=35
Z=10
```

- Equation with numbers and parameters

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

$R10 = 15 + R11$

6.5 Programming examples with parameters

Example: Rectangle (diagram on the following page)

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

```
L46                                     Subroutine
N5  G01 G64 G91 Z=-R2 LF
N10 X=R0 LF
N15 G02 X= R3 Y=-R3 I0 J=-R3 LF
N20 G01 Y=-R1 LF
N25 G02 X=-R3 Y=-R3 I=-R3 J0 LF
N30 G01 X=-R0 LF
N35 G02 X=-R3 Y= R3 I0 J= R3 LF
N40 G01 Y= R1 LF
N45 G02 X= R3 Y= R3 I= R3 J0 LF
N50 G01 Z= R2 LF
N55 M17 LF
```

% 4012

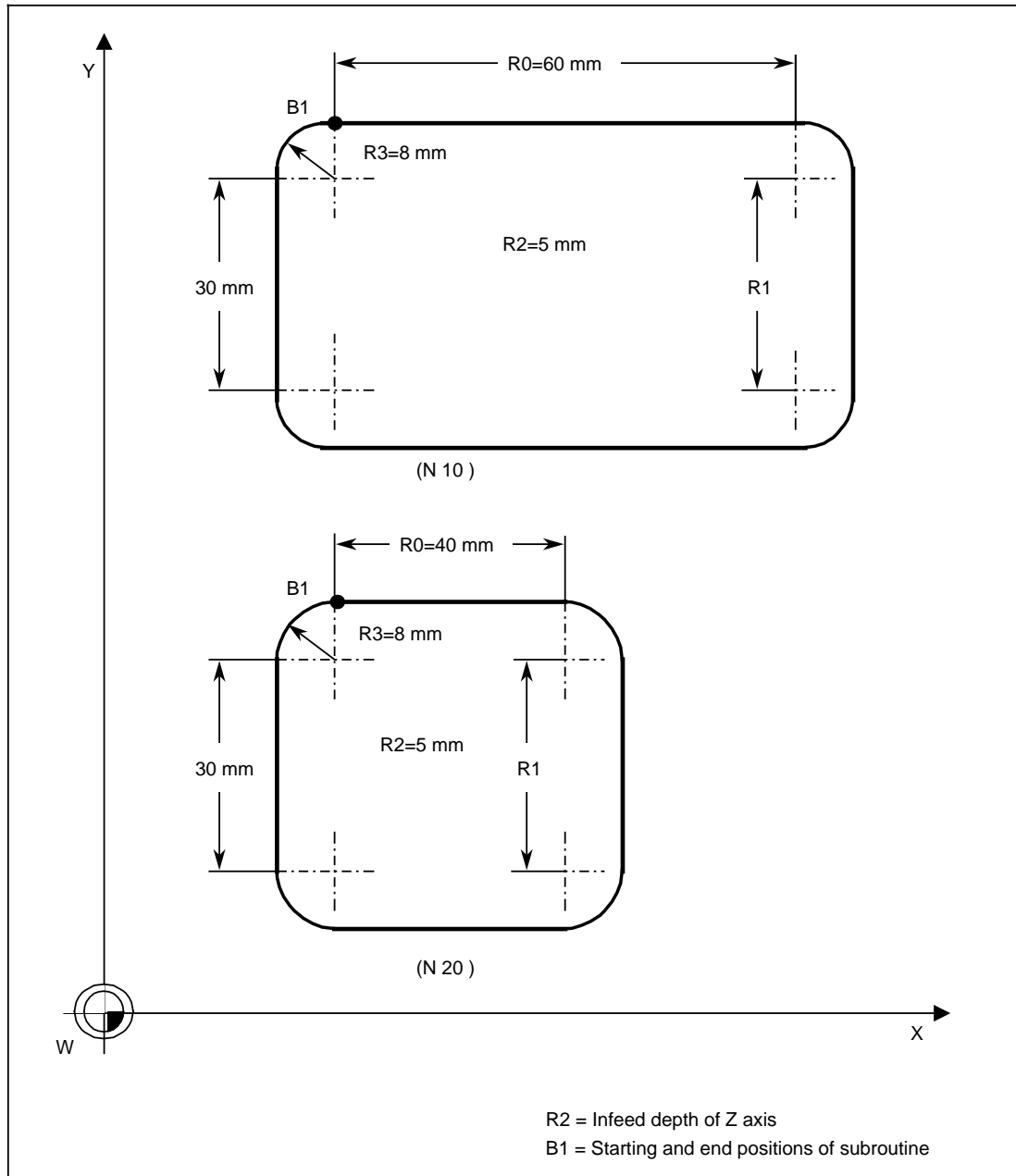
```
N5  G90 X50 Y50 F500 LF
N10 L46 P1 R0=60 R1=30 R2=5 R3=8 LF

N15 G90 X50 Y0 LF
N20 L46 P1 R0=40 LF

N25 M02 LF
```

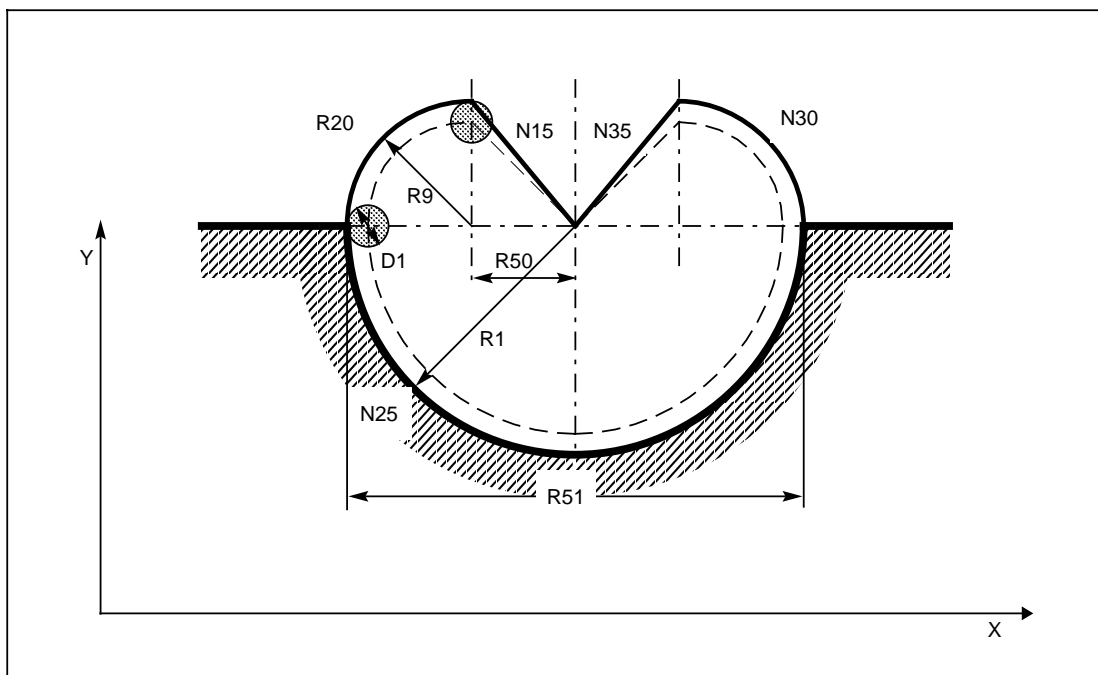
First starting position of current program
 Call subroutine L46, 1 pass
 – Initialization of R parameters (variables)
 Second starting position
 Call subroutine L46, 1 pass
 – Value of R0 changed from 60 to 40, other
 R parameters remain unchanged.

6.5 Programming examples with parameters



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. Then this difference is entered as additive tool wear.



```

L11
N5  R50=R1-R9  LF
N10 R51=2*R1  LF
N15  G00  G64  G91  G41  X=-R50  Y= R9  F500  D1  LF

N20  G03  X=-R9  Y=-R9  U=R9  LF

N25  X= R51  U=R1  LF
N30  X=-R9  Y=R9  U=R9  LF

N735 G00  G40  X=-R50  Y=-R9  LF
N40  M17  LF

```

Subroutine call:

```

% 5873
N5  G00  X100  Y100
N10  L11  P1  R1=50  R9=10  LF

N15  M30  LF

```

Subroutine

```

Calculate approach circle
Definition of semicircle
Approach start of cutout quarter-circle,
tool offset activated
Approach contour on quarter-circle
(radius programming)
Machine semicircle
Exit from contour on quarter-circle
(radius programming)
Position to subroutine starting point
End of subroutine

```

Main program

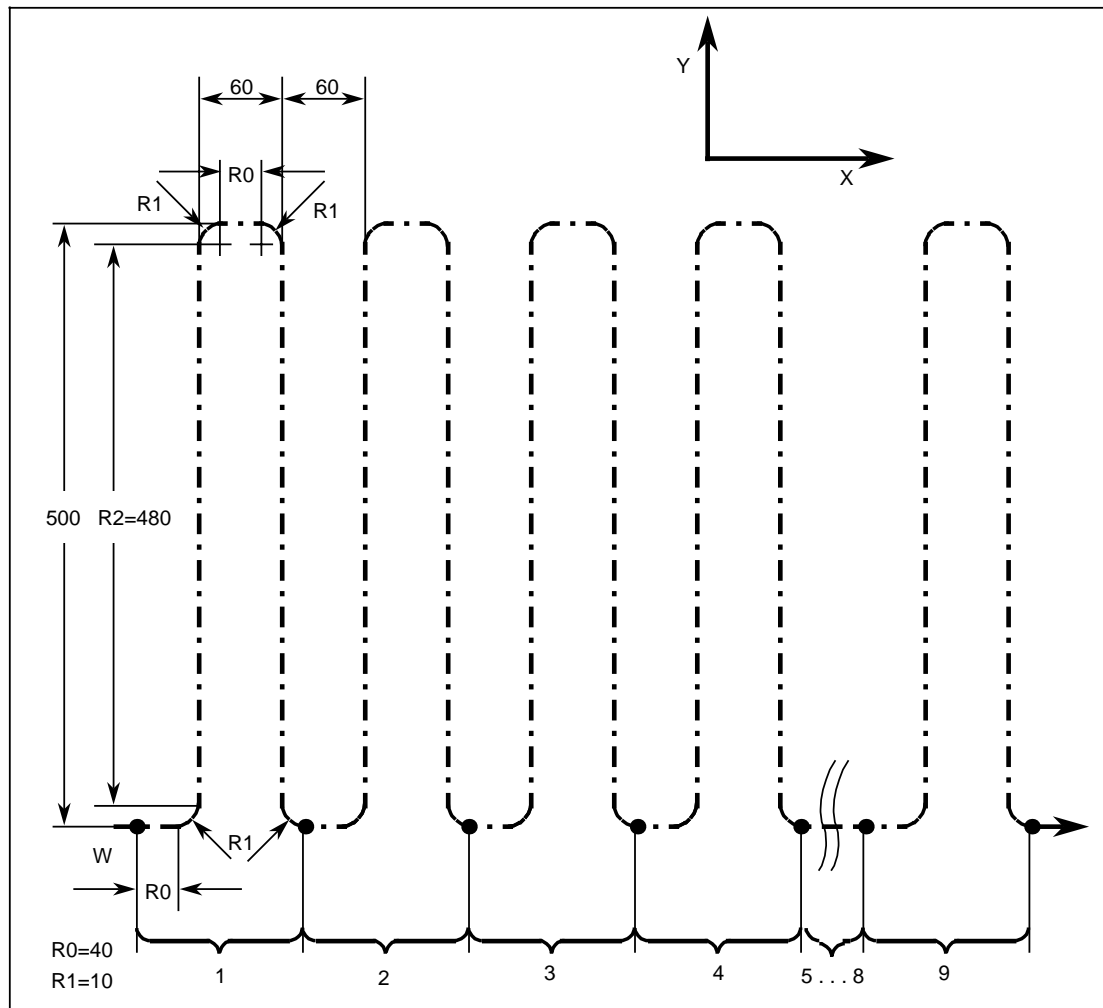
```

Subroutine call with R parameter
initialization

```

Example: Line-by-line 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.



```
% 5874
```

```
N5 G00 X30 Y30 L_F
```

```
N10 L34 P9 R0=40 R1=10 R2=480 F200 L_F
```

Call subroutine L34; 9 passes,
R parameter initialization

```
N20 M02 L_F
```

```
L34
```

Subroutine

```
N5 G01 G64 G91 X=R0 L_F
```

```
N10 G03 X=R1 Y=R1 I0 J=R1 L_F
```

```
N15 G01 Y=R2 L_F
```

```
N20 G02 X=R1 Y=R1 I=R1 J0 L_F
```

```
N25 G01 X=R0 L_F
```

```
N30 G02 X=R1 Y=-R1 I0 J=-R1 L_F
```

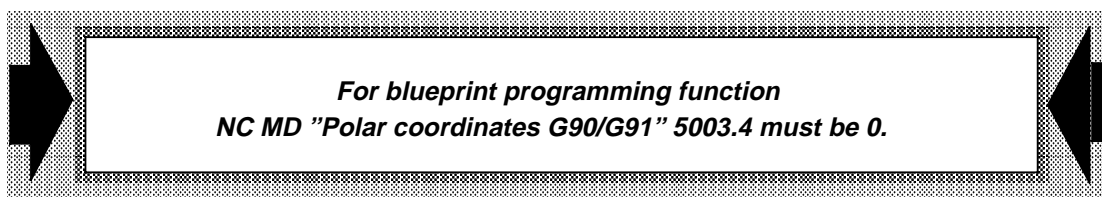
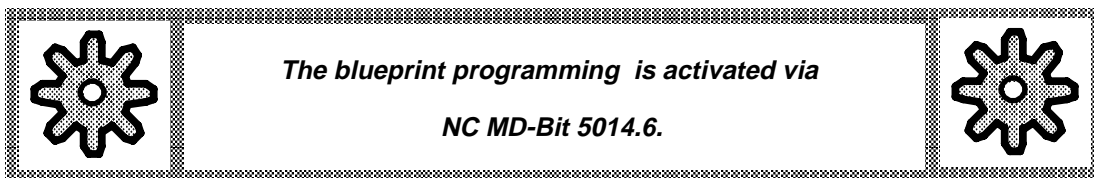
```
N35 G01 Y=-R2 L_F
```

```
N40 G03 X=R1 Y=-R1 I=R1 J0 L_F
```

```
N45 M17 L_F
```

7 Contour Definition

7.1 Blueprint programming



Multi-point definitions for direct programming in accordance with the workpiece drawing are provided for blueprint programming. The points of intersection of the straight lines are specified as coordinate values or by means of angles.

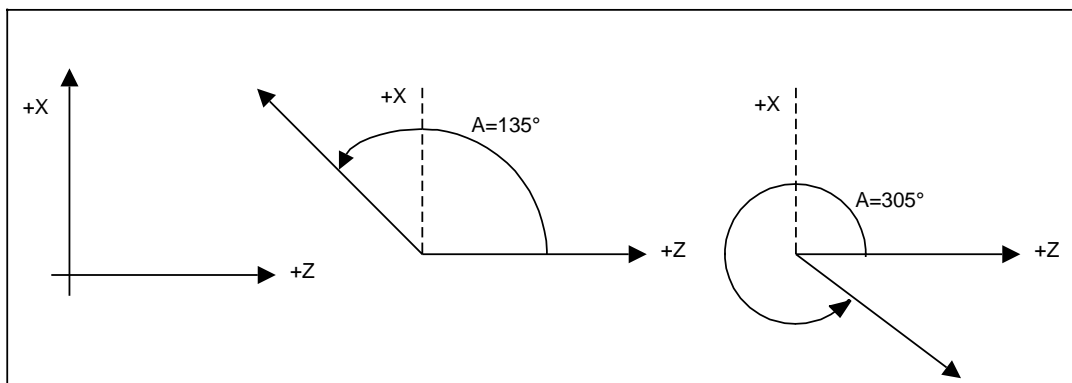
The various straight lines can be joined together directly in the form of a corner, rounded via radii or chamfered. Chamfer and transition radii are specified only by their size. The geometrical calculation is performed by the control. The end position coordinates can be programmed using either absolute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

Angle (A):

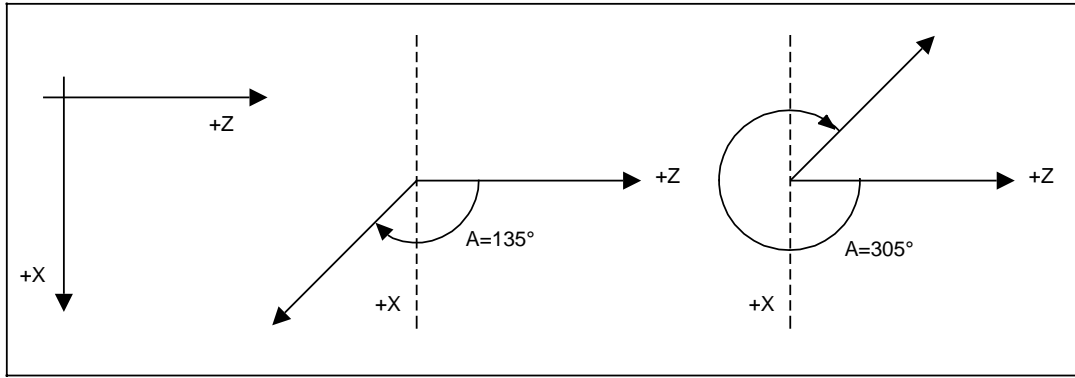
Input resolution 0.00001 corresponds to 10^{-5}° .

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

Turning machine:



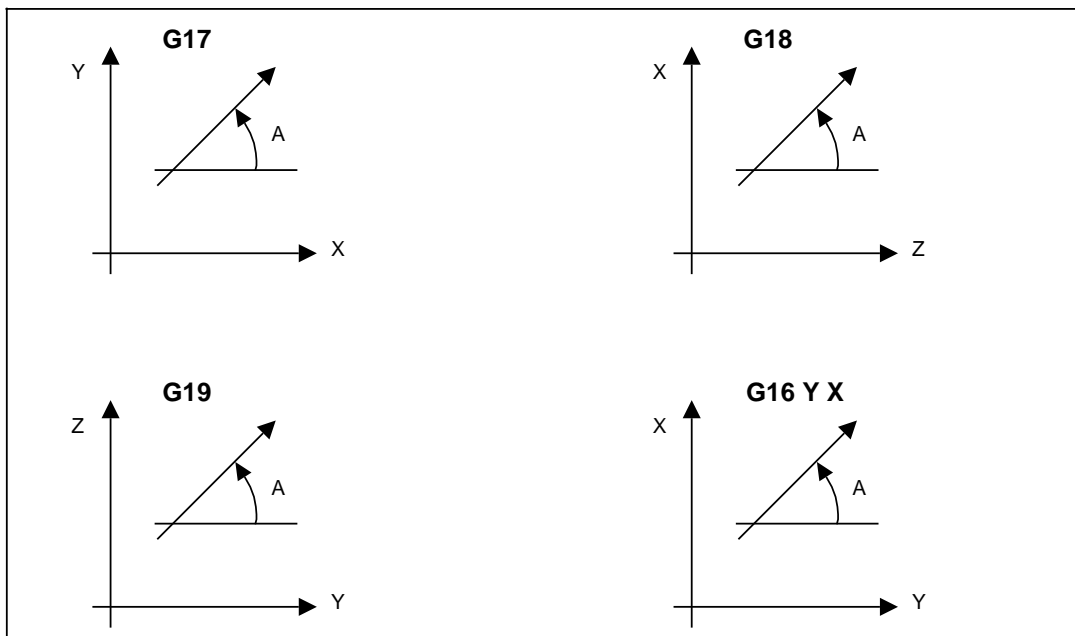
Clockwise system and operating area behind turning centre



Clockwise system and operating area in front of turning centre

Milling machine: Plane selection

The required plane is selected with G17, G18 or G19.



Plane selection

If the plane is freely selected (G16), the plane 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 referred to the reference axis.

***Blueprint programming is only permissible in the selected plane.
 3D machining is not possible.***

7.2 Contour definition programming

The elements described are valid for a turning machine with an operating area after the turning centre and for a milling machine in the selected plane Z-X (G18).

Examples 1 to 8 represent the basic elements of contour definition programming. These contour elements can be combined in a number of ways. The addresses for the angle (in this case A) and the radius (in this case B) are freely selectable in the control. The addresses must not be allocated more than once. The starting position is defined by the previous block.

Function	Programming	Example
(1) 2-point definition	N... A... X ₂ ... (or Z ₂) LF The second end coordinate is calculated by the control.	
(2) Circular arc	N...G02 (or G03) I.. K.. B.. X ₂ .. (or Z ₂) LF The circular arc is limited to one quadrant. The second end position coordinate is calculated by the control. In the contour definitions parameters I and K must both be programmed, even if one of the values is zero.	
(3) 3-point definition	N.. A ₁ .. A ₂ .. X ₃ .. Z ₃ ... LF The control calculates the coordinates of the vertex and generates 2 blocks. Angle A ₂ is referred to the second straight line.	

7.2 Contour definition programming

Function	Programming	Example
<p>(4) Chamfer</p>	<p>N... X₂... Z₂... B-... LF N... X₃... Z₃... LF 1)</p> <p>B- ...means insert a chamfer B ...means insert a radius (The "minus" character is not a sign here; instead it is a special identifier for B = chamfer)</p>	
<p>(5) Radius</p>	<p>N... X₂ ... Z₂ ... B... N... X₃ ... Z₃ ... LF 1)</p> <p>The inserted radius must not be larger than the smaller of the two paths.</p>	
<p>(6) Straight line-circular arc (tangent)</p>	<p>N.. G02 (or G03) A.. B.. X₃.. Z₃.. LF</p> <p>The circular arc must not exceed 180°. The sequence A (angle) followed by B (radius) must be used.</p>	
<p>(7) Circular arc - straight line (tangent)</p>	<p>N.. G02 (or G03) B.. A.. X₃.. Z₃.. LF</p> <p>The circular arc must not exceed 180°. The sequence B, A must be used. A radius must not be inserted in X₃, Z₃.</p>	
<p>(8) Circular arc - circular arc (tangent)</p>	<p>N.. G02 (or G03) I1.. K1.. I2.. K2.. X₃.. Z₃.. LF Kreis 1</p> <p>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 of the values is zero.</p>	

1) Second block can also be a contour definition.

Function	Programming	Example
(1) + (4) 2-point definition + chamfer	N... A.. X ₂ ... (or Z ₂ ..) B... LF N... X ₃ .. Z ₃ ... LF 1)	
(1) + (5) 2-point definition + radius	N... A.. X ₂ ... (or Z ₂ ...) B... LF N... X ₃ .. Z ₃ ... LF 1) The inserted radius must not be larger than the smaller of the two paths.	
(3) + (4) 3-point definition + chamfer	N.. A ₁ .. A ₂ .. X ₃ .. Z ₃ .. B... LF	
(3) + (5) 3-point definition + radius	N.. A ₁ .. A ₂ .. X ₃ .. Z ₃ .. B... LF	
(3) + (4) + (4) 3-point definition + chamfer + chamfer	N... A ₁ .. A ₂ .. X ₃ ... Z ₃ .. B ₁ ... B ₂ ... LF 1) N... X ₄ .. Z ₄ .. LF 1) Addition of a second chamfer at end position X ₃ , Z ₃ .	

1) Second block can also be a contour definition.

Function	Programming	Example
(3) + (5) + (5) 3-point definition + radius + radius	N... A1.. A2.. X3.. Z3.. B1... B2.. LF N... X4.. Z4.. LF 1)	
(3) + (4) + (5) 3-point definition + chamfer + radius	N... A1.. A2.. X3.. Z3.. B-... B.. LF N... X4.. Z4.. LF 1)	
(3) + (5) + (4) 3-point definition + radius + chamfer	N... A1.. A2.. X3.. Z3.. B.. B-.. LF N... X4.. Z4.. LF 1)	

B0 must be programmed for corners where no chamfer or radius is to be inserted if a further radius or chamfer follows in the contour definition.

When this is programmed, the control generates a block with a distance of 0. This must be noted if TNRC/CRC is active. B-0 is interpreted as B0. A radius or chamfer can only be inserted between two linear blocks.

The sequence of addresses A, X, Z, B, 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 can also be a contour definition.

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

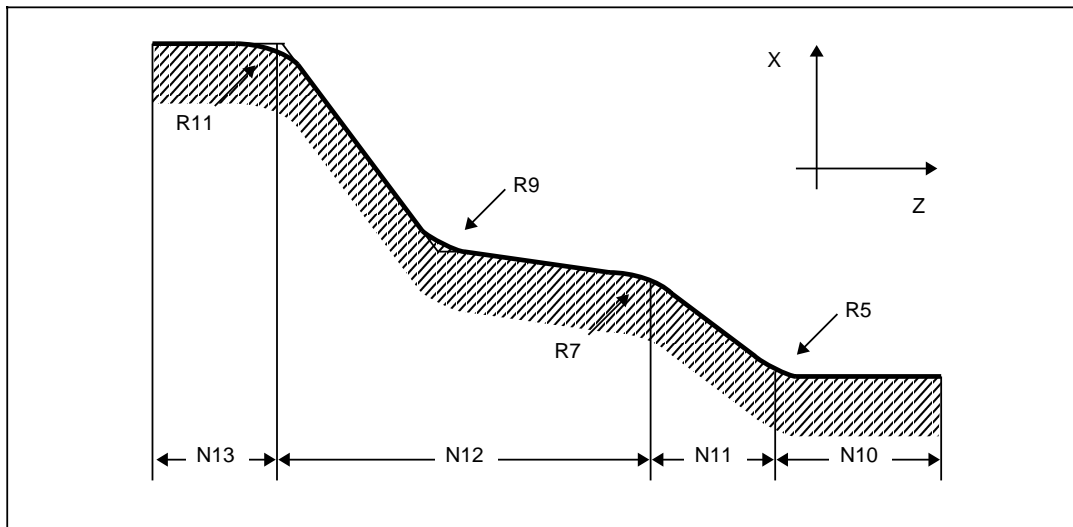
If G09 is programmed in a contour definition block, it is not active until the end of the block, i.e. when the end position is reached. G09 is automatically generated by the control at irregular points (corners, edges) in the contour definition.

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

7.4 Chaining of blocks

It is possible to chain blocks with or without angle inputs and with inserted radii or chamfers in any sequence.

Example: Chaining of blocks

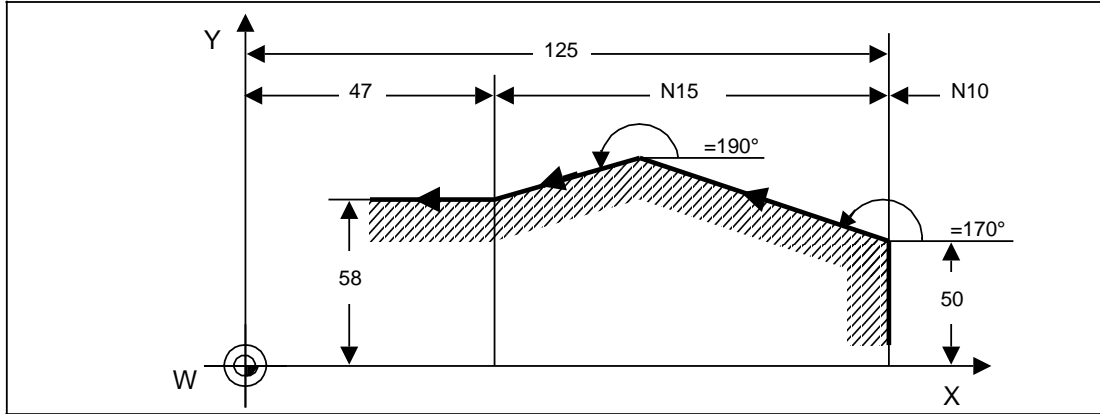


Chaining of blocks

N10	Z... B5 L _F	Straight line with radius
N15	A... X... B7 L _F	Straight line with radius
N20	A... A... X... Z... B9 B11 L _F	3-point definition with radius at both corners
N25	Z... L _F	Straight line

7.5 Example: Milling machine

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

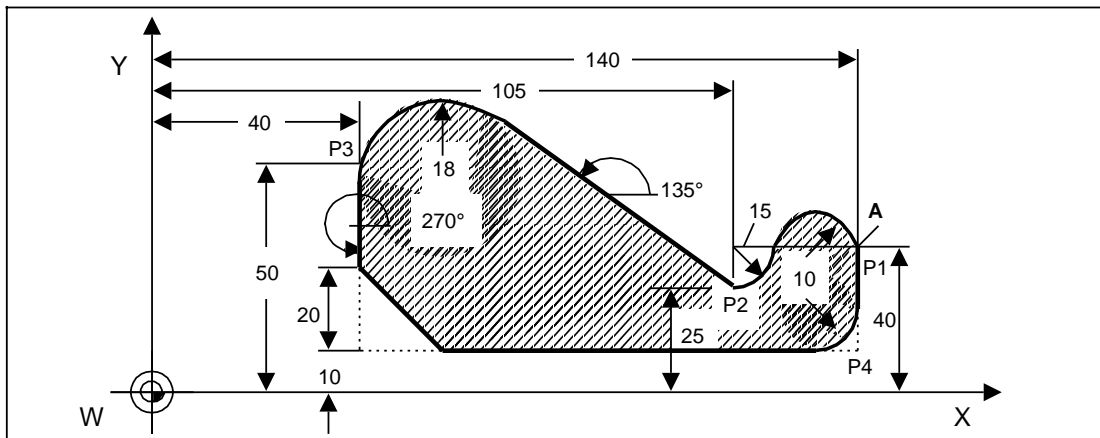


```

N10 G00 G90 X125 Y50 LF
N15 G01 A170 A190 X47 Y58 F... LF           3-point definition
:
    
```

Example: Contour definition programming for milling machine

In the example described below the following contour definitions are used: Circular arc - circular arc, straight line - circular arc, 3-point definition + chamfer + radius.



A = Starting point

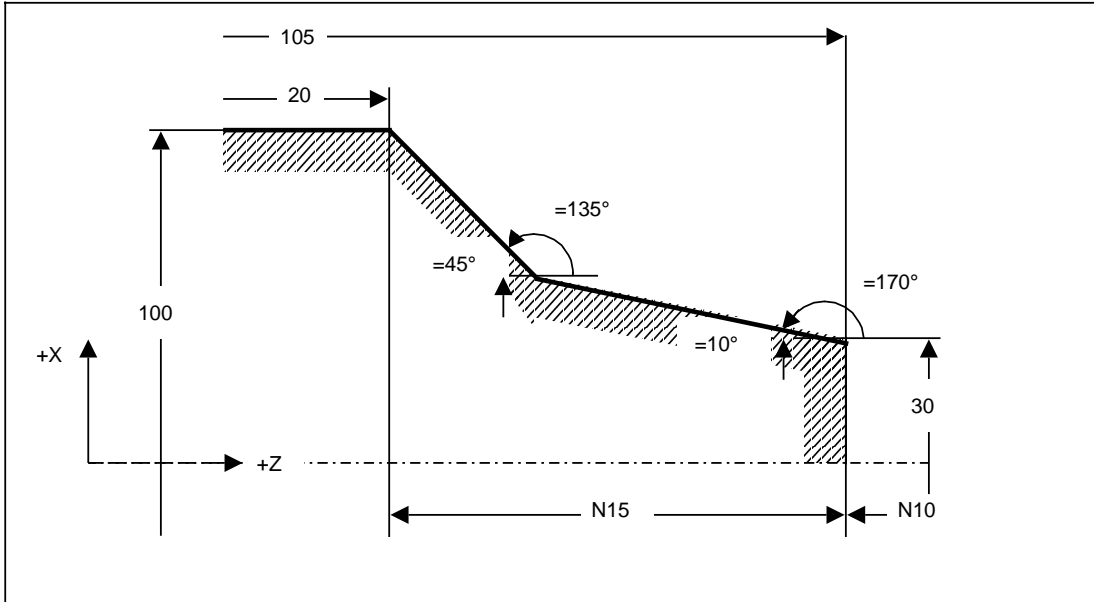
```

% 486
G00 X140 Y40 F1000 LF
N5 G90 G03 I-10 J0 I0 J15 X105 Y25 LF      (P2) Circular arc - circular arc
N10 G03 A135 U18 X40 Y50 LF                (P3) Straight line - circular arc
N15 G01 A270 A0 X140 Y10 U-20 U10 LF      (P4) 3-point definition + chamfer (U20)
                                           + radius (U10)
N20 Y40 LF                                  (P1) Straight line
N25 M02 LF                                  End of subroutine
    
```

7.6 Example: Turning machine

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

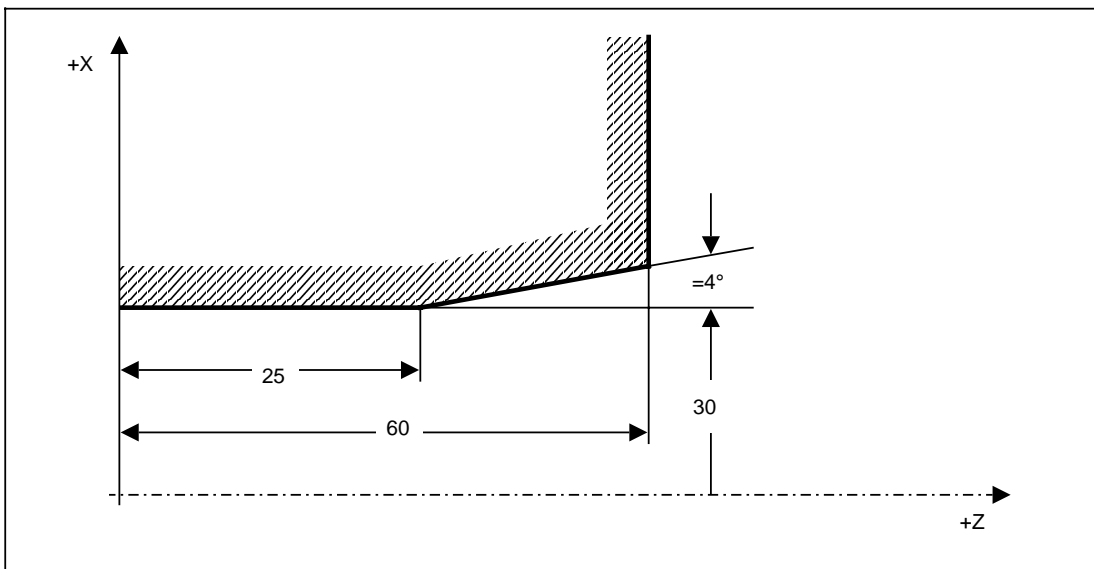
Example: External machining



⋮

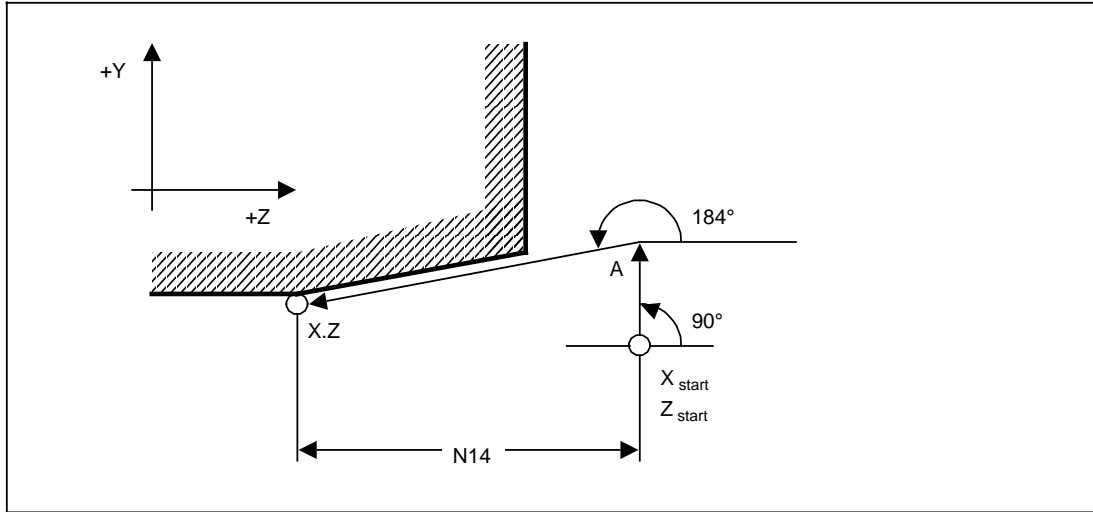
```
N10 G00 G90 X30 Z105 LF
N15 G01 A170 A135 X100 Z20 F... LF           3-point definition
```

Example: Internal machining



Drawing dimensions

The starting point can be defined anywhere outside the inner taper.



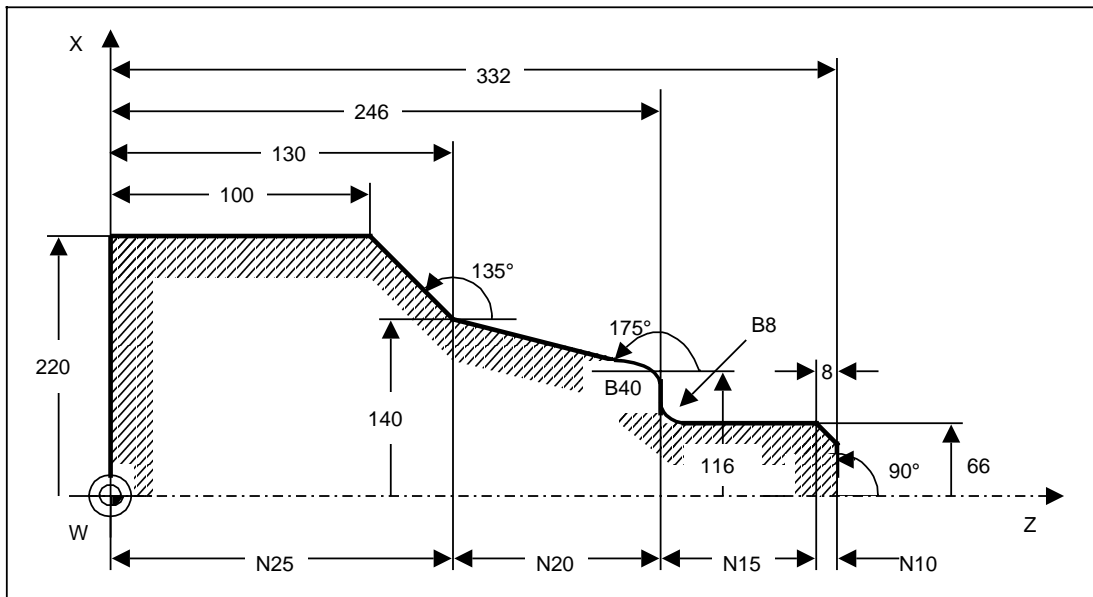
Drawing dimensions

The perpendicular through the starting point and the extension of the inner taper yield point of intersection A.

The program is then as follows:

```
N15 G00 Xstart Zstart LF
N20 G01 A90 A184 X... Z... LF
```

Example: Contour definition programming for turning machine



Contour definition programming for turning machine

Blueprint programming

```
% 495
```

```
N5 G00 G90 X Z332 L_F
```

Absolute dimension data, starting point definition

```
N10 G01 G09 A90 X66 B-8 F0.2 L_F
```

2-point definition + radius (B-8) (Chaining with B-)

```
N15 A180 A90 X116 Z246 B8 L_F
```

3-point definition + radius (B8)

```
N20 G03 B40 A175 X140 Z130 L_F
```

Circular arc - straight line

```
N25 G01 A135 A180 X220 Z0 L_F
```

3-point definition

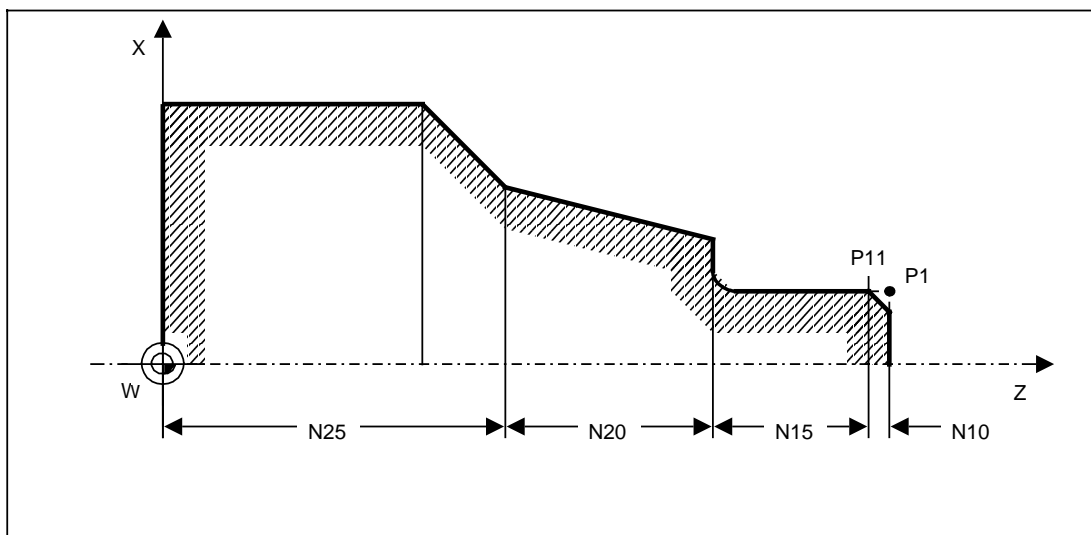
```
N30 M02 L_F
```

End of subroutine

7.7 Miscellaneous functions in chained blocks for turning and milling machines

Blocks are said to be chained whenever they are joined together by means of radii or chamfers.

Example: Turning machine



Miscellaneous functions in chained blocks

A block with miscellaneous functions can be located between two chained blocks:

```
N10 G01 G09 A90 X66 B-8 F0.2 L_F (P1)
```

```
N101 M... H... ...
```

```
N15 A180 A90 X116 Z246 B8 L_F
```

Miscellaneous functions are active at point P11 (see above).

Relief-cutting is thus effected at point P11. The F-value programmed in block N10 is active at the start of block N10.

8 Tool Offsets

8.1 Tool data

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

Length	± 9999.999 mm	} input resolution 1µm
Radius	± 999.999 mm	

T number 4 decades

Tool type

In the standard control the tool offset block is divided into 10 columns (P0 to P9). The format of the tool offset block is identified by the tool type (P1). Ninety nine tool offset blocks are available to the user.

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

The geometry and wear are updated, for example by measuring cycles in the NC. The wear and basic dimension are summated in the NC in accordance with the machine data.

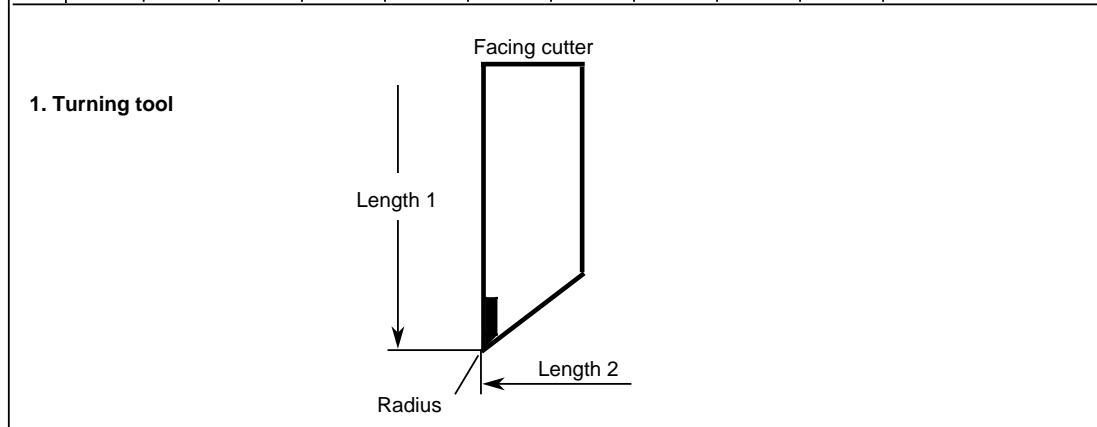
Breakdown of tool type P1

Type 0	Tool not defined
Type 1 . . . 9	Lathe tools, position of tool tip
Type 10	Tools with active length compensation only (e.g. drills)
Type 20	Tools with radius compensation and one length compensation (e.g. cutters)
Type 30	Tools with radius compensation and two length compensations (e.g. angle cutter)

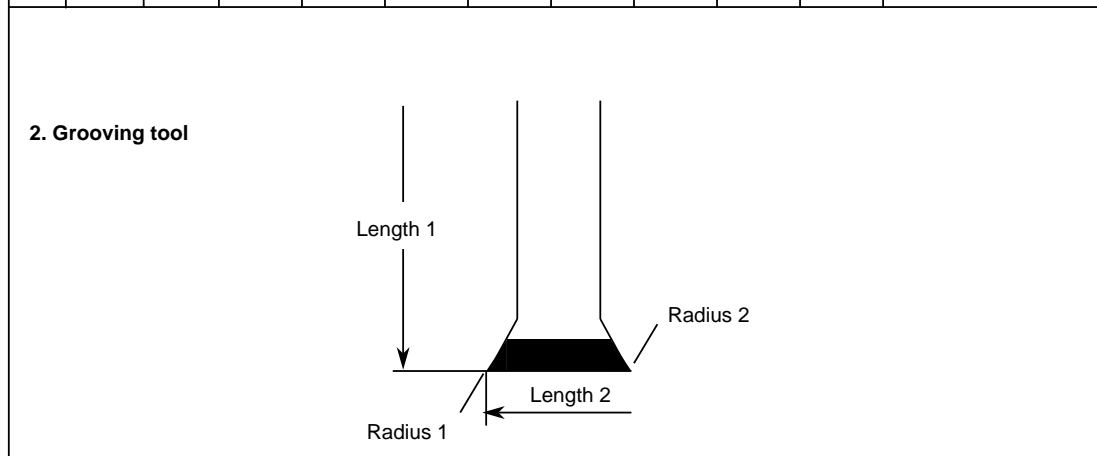
Tool offsets can be entered both through the operator keyboard and the data input interface, but no block numbers must be programmed (see punched tape formats).

Structure of tool offset memory

	Offset memory structure										Programming
	T No.	Type	Geometry			Wear			Basis (supp. TO)		Tool call
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	
Dn	8 digits	1...9	Length 1	Length 2	Radius	Length 1	Length 2	Radius	Length 1	Length 2	T100 . G16 ZX . . D50 Length 1 in X axis (plane axis) Length 2 in Z axis



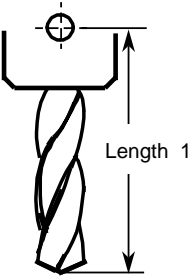
	Offset memory structure										Programming
	T No.	Type	Geometry			Wear			Basis (supp. TO)		Tool call
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	
Dn	8 digits	1...9	Length 1	Length 2	Rad. 1	Length 1	Length 2	Radius	Length 1	Length 2	T100 . D50 (e.g. for left-hand cutting edge) . . D51 (e.g. for right-hand cutting edge)
Dn +1	8 digits	1...9	Length 1	Length 2	Rad. 2	Length 1	Length 2	Radius	Length 1	Length 2	



Note: See also section 3.2.11, Plane selection

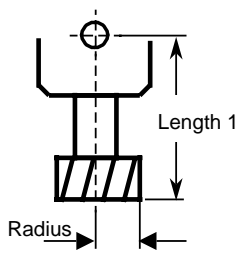
Dn	Offset memory structure					Programming	
	T No.	Type	Geometry		Wear	Basis (supp. TO)	Tool call
	P0	P1	P2	P5	P8		
	8 digits	10	Length 1		Length 1	Length 1	T100 . . M06 G17 (D50) *) or M06 G16 UV W +/- (D50)

3. Drill



Dn	Offset memory structure						Programming	
	T No.	Type	Geometry		Wear		Basis (supp. TO)	Tool call
	P0	P1	P2	P4	P5	P7	P8	
	8 digits	20	Length 1	Radius	Length 1	Radius	Length 1	T100 . . M06 G17 (D50) Length1 in Z, Length 1 active in Z radius active in X/Y plane or M06 G16 UV W +/- (D50) Length 1 active in W radius active in U/V plane *)

4. End mill



Note: See also section 3.2.11, plane selection

*) G16 is strictly a setting function: travel is not possible in these blocks.
D is not coupled to plane selection

Dn	Offset memory structure										Programming
	T No.	Type	Geometry			Wear			Basis (supp. TO)		Tool call
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	
8 digits	30	Length 1	Length 2	Radius	Length 1	Length 2	Radius	Length 1	Length 2	T100 . . M06 G16 UV Z +/- U+/- (D50) Radius active in U/V plane length 1 active in Z length 2 active in U	

5. Angle cutter

Note: See also section 3.2.11, Plane selection

The question as to whether the difference

- *is traversed directly following the change*
- *or is not taken into consideration until the programmed traverse of the corresponding axis*

is resolved on start-up in the case of T versions.

With TNRC type 1 to 9 (G41, G42) the difference in addition to the tool nose radius is traversed in both axes.

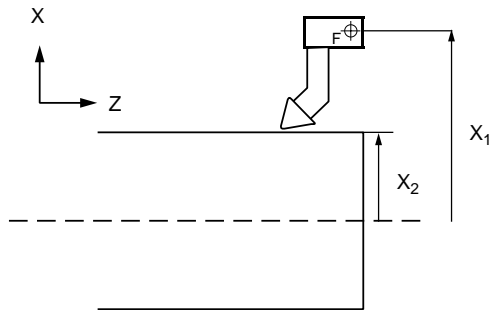
Determining the tool offsets

If no measuring device is available, the tool length compensation can be measured as follows. (Workpiece related actual value system must not be active in RESET).

Example:

Turning tool, type 3 (prerequisite: D0 active, no preset offset, no zero offset).

1. First, a workpiece is scratched by a clamped turning tool in the X direction.



The scratching produces an axis actual value (display) and the resulting turning radius of the workpiece (to be measured).

Tool length compensation 1 is found by subtracting the turning radius (X_2) from the axis actual value (X_1) and then dividing the result by 2.

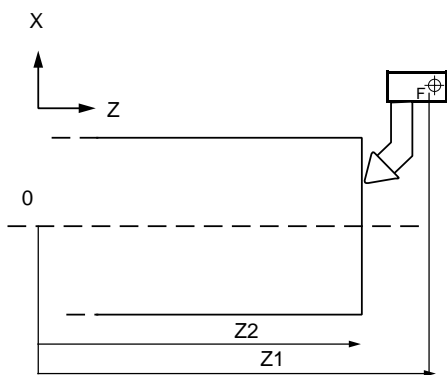
Method:

The actual value display (X_1) is entered directly in TO length 1. Subtract the measured actual value (X_2) from TO length 1 using the calculator function.

The result is a diameter, TO length 1 must however be entered as a radius.

The result must be divided by 2 using the calculator function.

- Then the workpiece is scratched by a clamped turning tool in the Z direction.



The scratching now produces an axis actual value Z₁ (display) and an actual workpiece length Z₂ (to be measured).

Tool length compensation 2 is produced by subtracting the workpiece length (Z₂) from the axis actual value (Z₁).

Note:

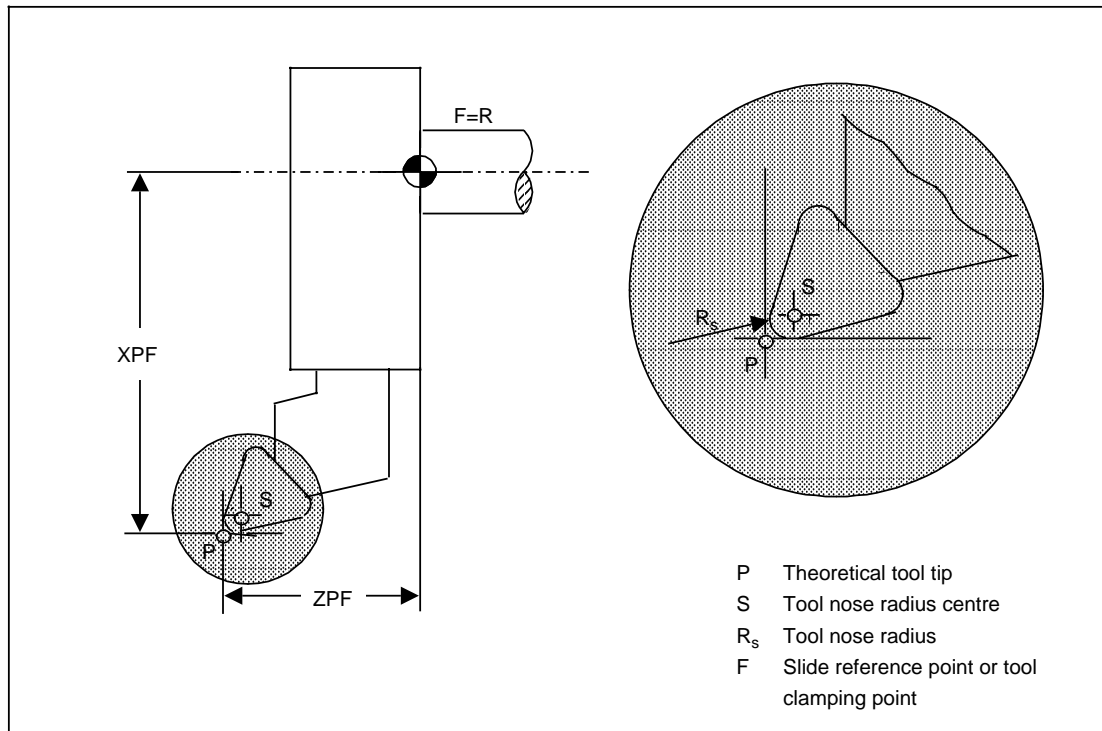
Length 2 refers to machine zero. This should be taken into account if zero offset is active in the Z axis.

- The measured tool can be checked for accuracy in MDA mode.

A workpiece with the dimensions X₂ and Y₂ is programmed with tool offset (D) selected.

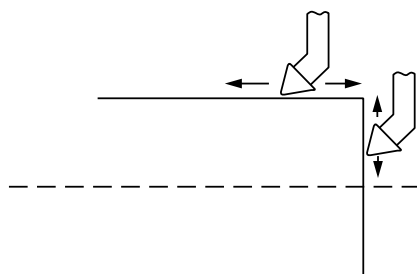
The diameter of the turned workpiece is then measured; if the diameter is too large, the tool length compensation must be entered as the differential dimension $\left(\frac{\text{Set-Act.}}{2}\right)$ and as a positive value in the wear memory.

This "scratching method" can be used to determine tool length compensation values of the turning tool. These compensation values refer to the theoretical tool tip P.



Using the determined tool offsets without selecting tool nose radius compensation (turning tool)

No dimensional errors occur when longitudinal and face surfaces are turned.



No dimensional errors

A positive dimensional error occurs when slanting surfaces are turned without selecting tool nose radius compensation.

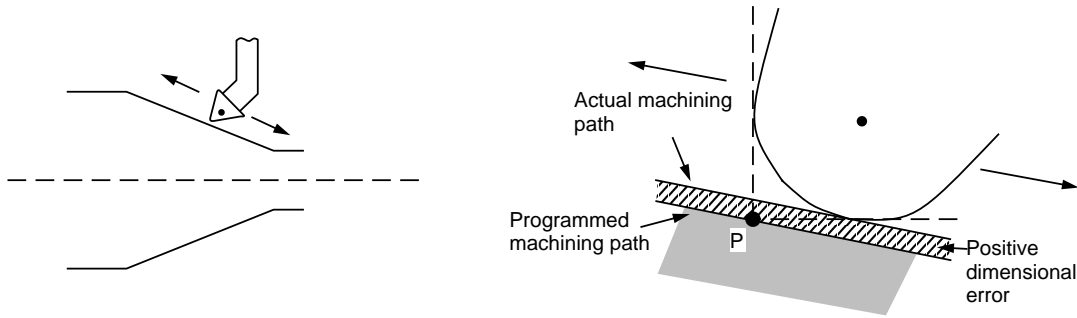


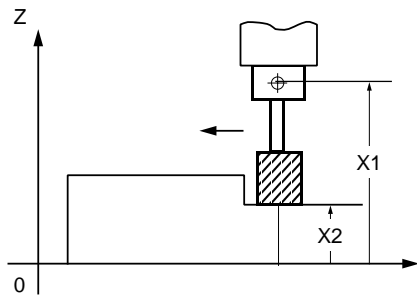
Figure showing detail

The positive dimensional error occurs because the tool length compensations are determined from the theoretical tool tip P, although the tool does in fact have a radius.

The dimensional error does not occur if the radius is entered in the tool offset and the tool nose radius compensation is activated.

Example:

Determining the tool length compensation of milling cutters (prerequisite: D0 active, no preset , no zero offset)

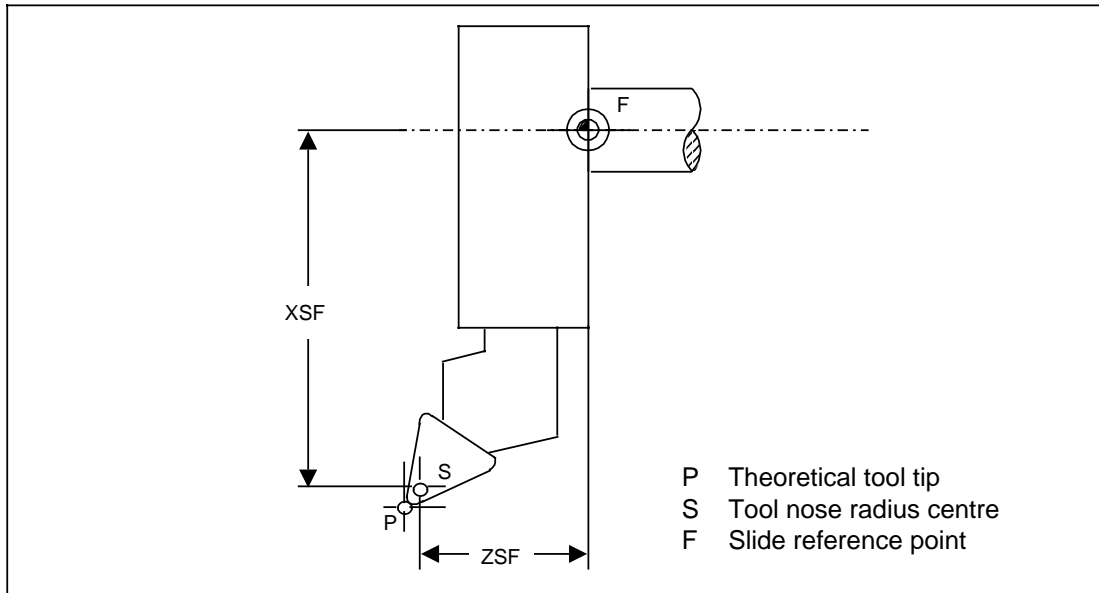


First, a workpiece is contacted by a clamped milling cutter. An axis actual value X1 results for the Z axis (display).

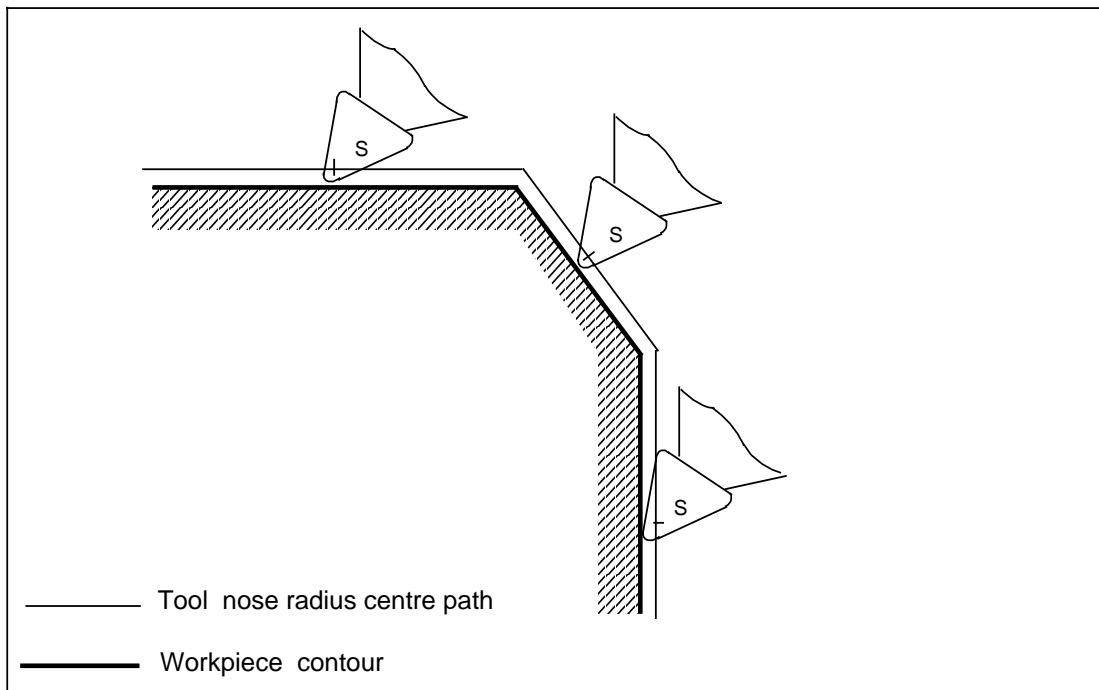
The resulting height of the workpiece (X2) must then be subtracted from the displayed axis actual value. This result can then be entered as the tool length compensation value in the tool offset.

8.2 Turning machine: Tool offset without using tool nose radius compensation (TNRC)

The effective tool offset is derived from the sum of the tool length compensation and any external, additive tool length compensation. The sum corresponds to dimension XSF or ZSF.



The path of tool nose radius centre S is programmed. The length compensation is referred to the tool nose radius centre.



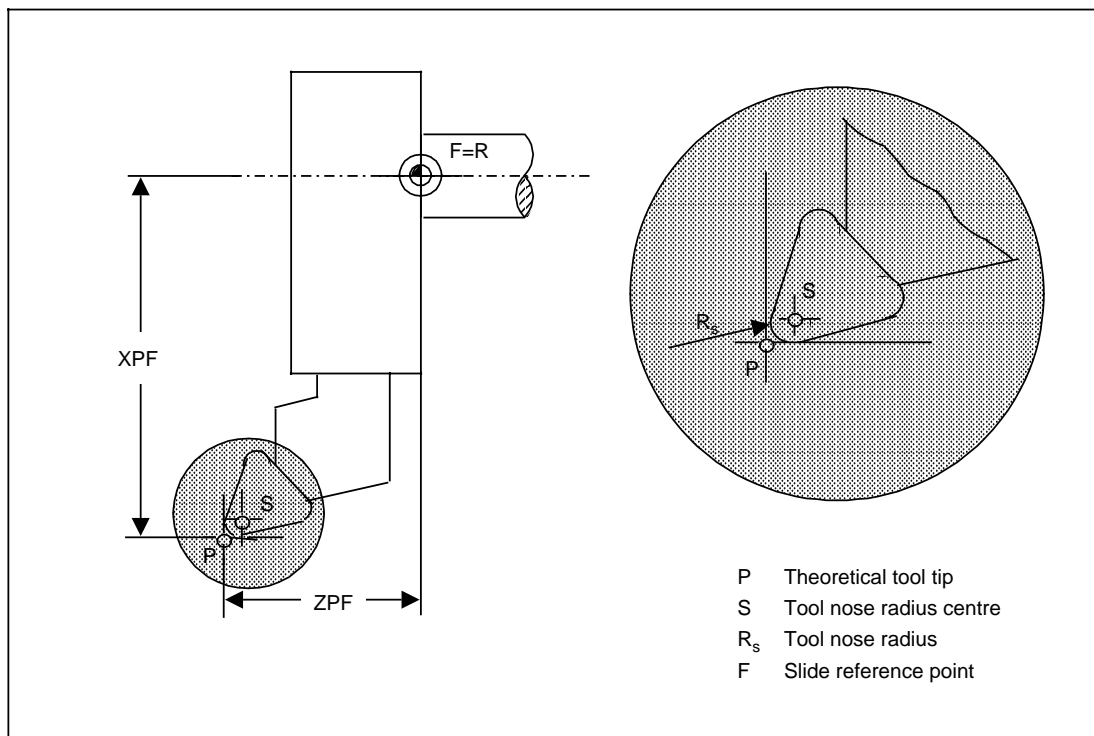
Incorporation of compensation

The difference between the old and new offset values is formed when the tool offset number is changed.

8.3 Tool offset using tool nose radius compensation (TNRC)

The workpiece contour can be programmed in conjunction with TNRC. The length compensation entered is referred to theoretical tool tip "P". The tool nose radius and the position of the cutting point (tool nose vector) must also be entered. The control then computes the path to be traversed. No contour errors occur.

The tool nose radius compensation takes effect at the end position of the block in which it is called (G41, G42), i.e. the next block is executed correctly.

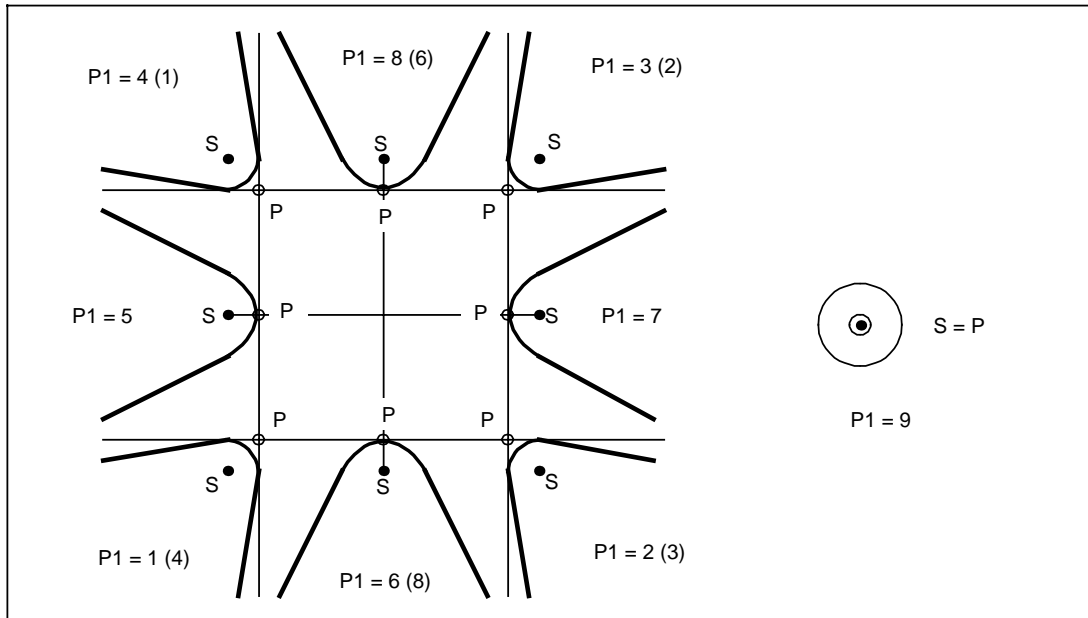


In order to calculate the tool nose radius compensation, the control requires the cutter radius (compensation value P4) and details of how the turning tool is clamped in the tool carrier. Identifier P1 = 1 to P1 = 9 must be entered in tool offset column P1 (tool type). The position of point P relative to tool nose radius centre S has thus been defined.

If tool nose radius centre S is used rather than point P as the reference point to determine the tool length compensation, identifier P1 = 9 must be entered. Identifier P1 = 0 is not allowed.

The diagram shows the possible positions of the turning tool with associated identifiers P1 = 1 to P1 = 9. The values in brackets apply to machining in front of the turning centre.

Machining after (before) turning centre



Position of tool nose radius centre

8.4 Milling machine: 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 compensation 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 takes effect only when the corresponding axis has been programmed (standard machine data).

Length compensation without cutter radius compensation

```
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
N51 D0 Z... LF                       Cancellation of length compensation
```

8.5 G40, G41, G42 Intersection cutter radius compensation (CRC)

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.

When mirroring is used, the path travelled by the tool is as follows (taking the sign 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	Right	Right	Left
-	Right	Left	Left	Right

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.

Example: Selection

```
N10 G01 G17 G41 D07 X... Y... LF
```

At the end of this block the compensated path has been 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
```

Selection of plane

```
N15 G41 D07 LF
```

Selection of compensation

```
N20 G01 X... Y... LF
```

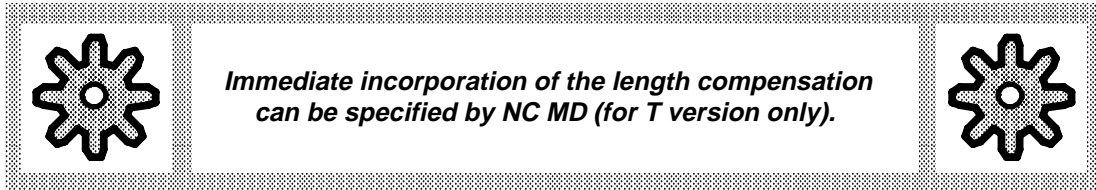
At the end of this block the compensated path has been reached in the selected plane. Only the radius compensation value is incorporated.

```
N25 Z... LF
```

The tool length compensation is incorporated.

The cutter radius compensation (G41 or G42) in linear blocks (G00, G01) can be cancelled with G40. In order to effect the required 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.



(Immediate incorporation of the length compensation can be specified by NC-MD; effective only in the case of T versions).

Example: Cancellation

N30 G40 X... L_F

Cancellation of tool compensation. Only the radius compensation value is cancelled.

N35 D0 Z... L_F

Length compensation value = 0 is active.

OR

N30 G41 D0 X... L_F

Replacement of all compensations by compensation values = 0. Only the radius compensation value is effectively cancelled (X-Y plane).

N35 Z... L_F

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... L_F

Incorporation of radius compensation (CRC selection)

N15 Z... L_F

Incorporation of length compensation

N20 G42 X... Y... L_F

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

N25 Z... L_F

No change in length compensation

Change in tool offset number

The G function **needs not be reentered.**

N10 G01 G17 G41 D12 X... Y... L_F

N15 Z... L_F

Change in length compensation

N20 D10 Z... L_F

N25 X... Y... L_F

Change in cutter radius compensation

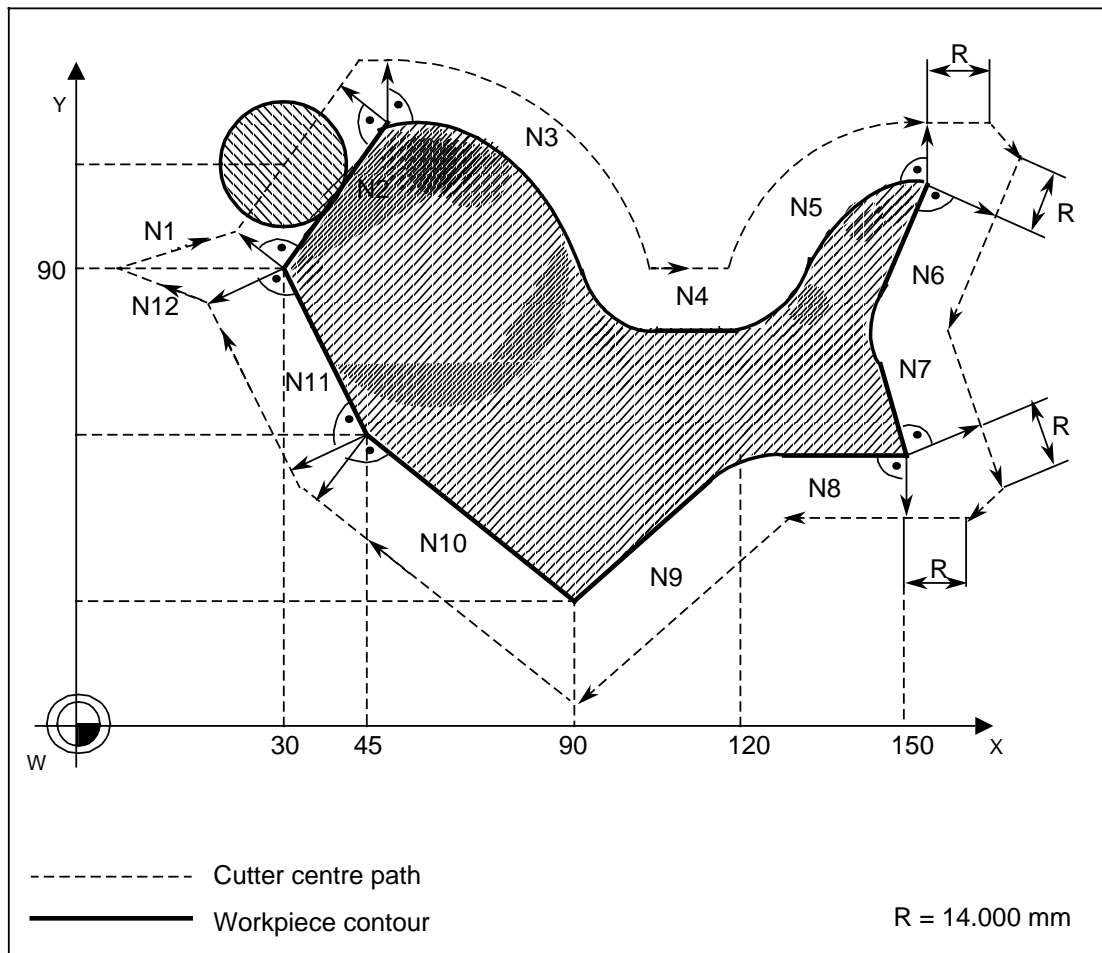
If CRC is selected, it is not permissible to program G58, G59 or G33.

Remedy:

Program the functions G58, G59, G33 before selecting the cutter radius compensation, or cancel the cutter radius compensation (select G58.. G33) 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

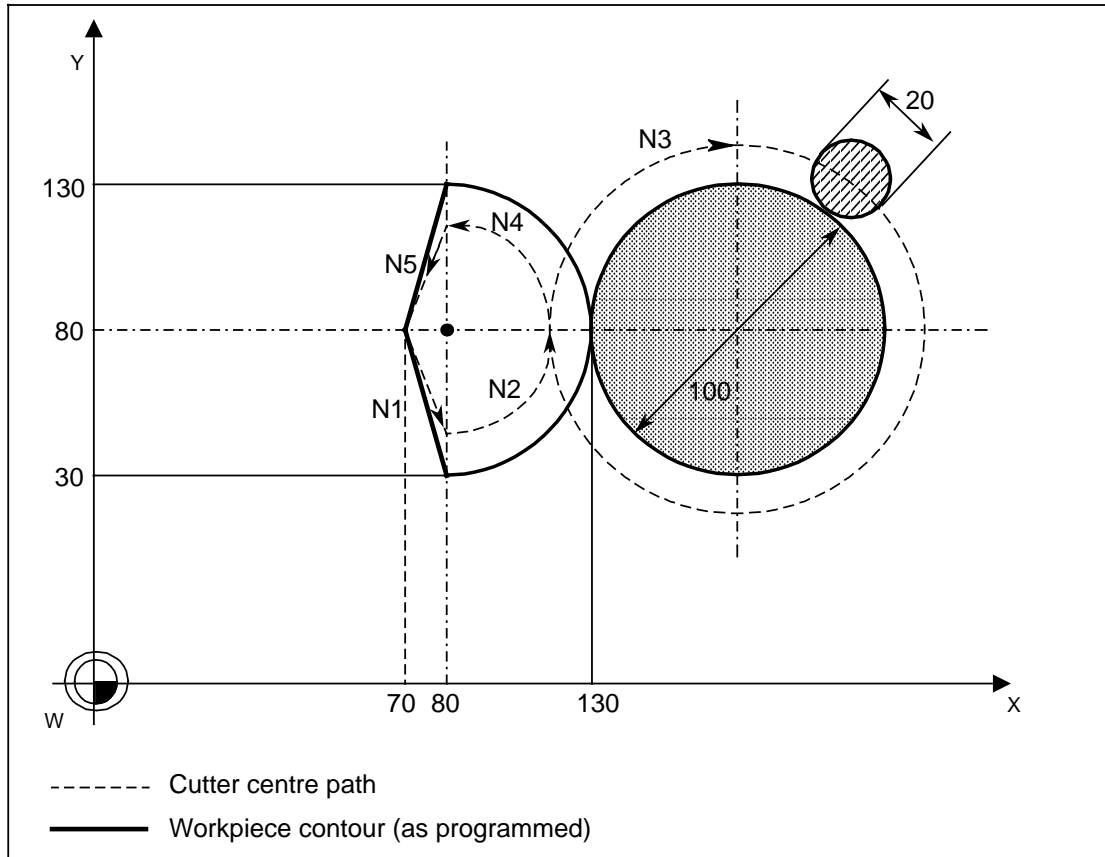


Milling with cutter radius compensation

```

% 700
N1 G01 G41 D1 G90 G17 X30 Y90 F500 S56 M03 L_F
N2 G91 X30 Y30 L_F
N3 G02 X30 Y-30 I0 J-30 L_F
N4 G01 X30 L_F
N5 G02 X30 Y30 I30 J0 L_F
N6 G01 X-15 Y-30 L_F
N7 X15 Y-30 L_F
N8 X-30 L_F
N9 X-30 Y-30 L_F
N10 X-45 Y30 L_F
N11 X-15 Y30 L_F
N12 G40 G90 X0 Y90 L_F
N13 M02
    
```

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 CRC*Full circle programming using CRC*

```

% 701
N1 G90 G00 G17 G41 D1 X80 Y30 L_F
N2 G03 X130 Y80 I0 J50 L_F
N3 G91 G02 X0 Y0 I50 J0 L_F
N4 G90 G03 X80 Y130 I-50 J0 L_F
N5 G00 G40 X70 Y80 L_F
N6 M02

```

8.6 Tool length compensation (positive or negative)

When length compensation is selected, compensation is positive. 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.

Defining the sign

A plus sign is entered by the operator if the tool length compensation entered (P2 and/or P3) is to be accounted for in positive direction. If a minus sign is entered, the tool length compensation applies in the negative direction.

8.7 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 axis perpendicular to this plane.

The following applies:

G17: Cutter radius XY, cutter length Z

G18: Cutter radius ZY, cutter length Y

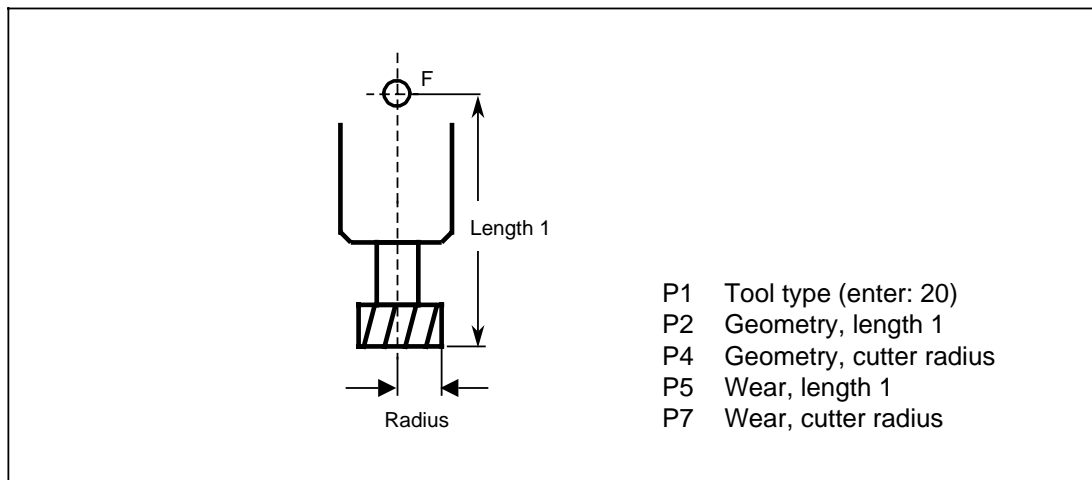
G19: Cutter radius YZ, cutter length X

It is also possible to incorporate a 4th or another axis using G16.

G16 X Y U cutter radius in XY plane, cutter length U

G16 X Y Z cutter radius in XY plane, cutter length in Z (negative direction)

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



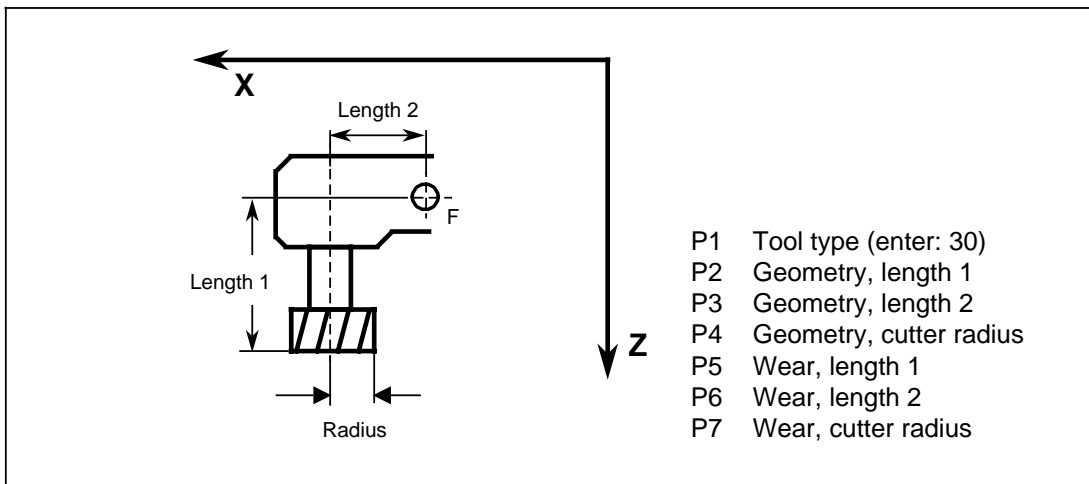
8.8 Tool offsets for angle cutter

As opposed to the end mill, an angle cutter requires an additional length compensation located in the plane of the cutter radius compensation. 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) while the fourth address specifies the additional length compensation 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).



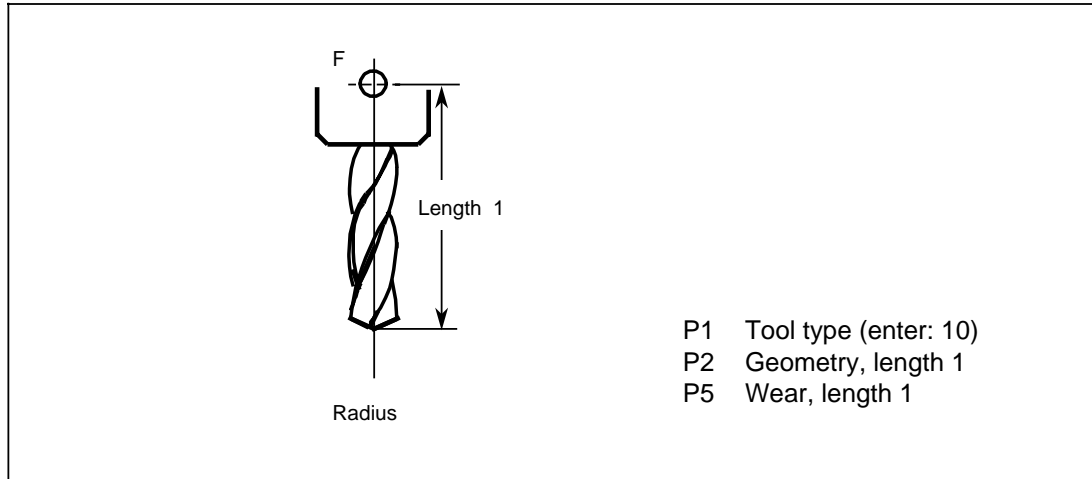
Compensation assignments: `G16 X Y Z X`

↑ ↑
Length 2
Length 1

8.9 Tool offset for drill

The number "10" must be entered under P1 (tool type).

In the case of a drill, length compensation (geometry = P2, wear = P5) acts in the axis perpendicular to the selected axis.



9 Cutter Radius Compensation (CRC), Tool Nose Radius Compensation (TNRC)

All stop positions for single blocks are marked S followed by the relevant block number in brackets.

Contour elements of the programmed path are marked with a thick line: **—————**

The diagrams are shown with G42. In the case of programs with G41, the angle transition is $\beta = 360^\circ - \dots$. The behaviour of the cutter radius compensation complies with the recalculated angle transition.

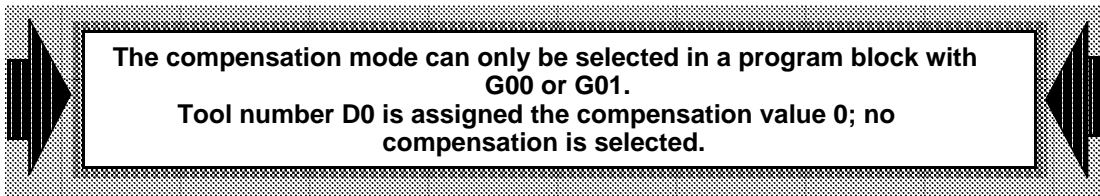
9.1 Selection of CRC/TNRC

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

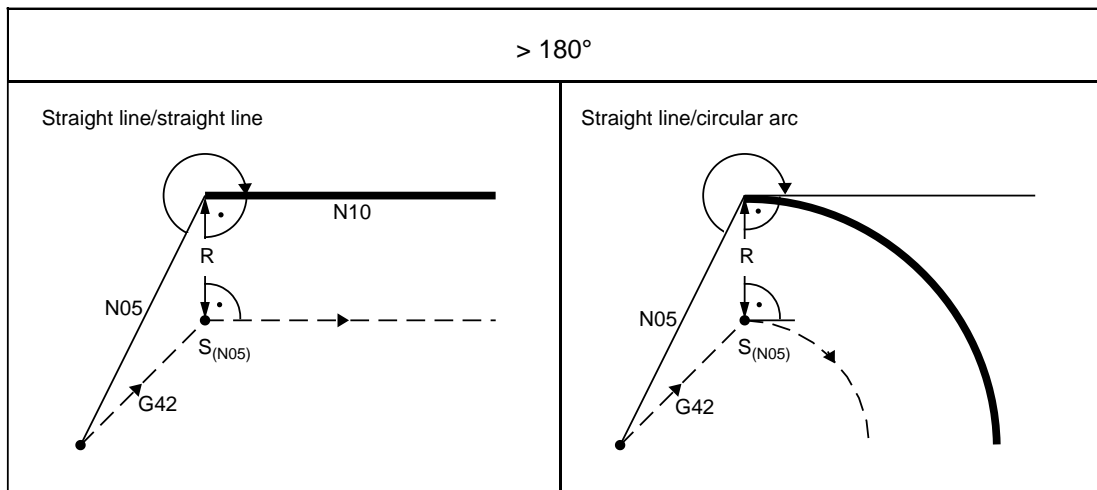
The compensation is effected to the left of the workpiece contour (in the traversing direction) with G41 and to the right of the workpiece contour with G42.

When selecting CRC/TNRC, two program blocks are read in to calculate the intersection point.

In the block after the selection block a block start vector (length R) is erected perpendicular to the programmed path.

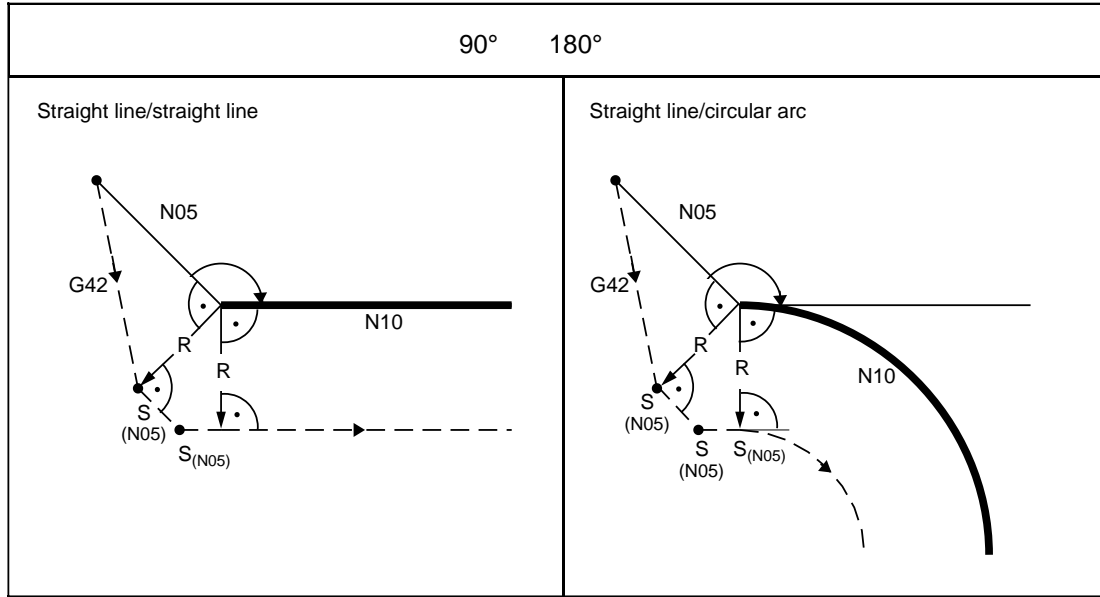


The diagrams below show the compensation selected for various approach angles.

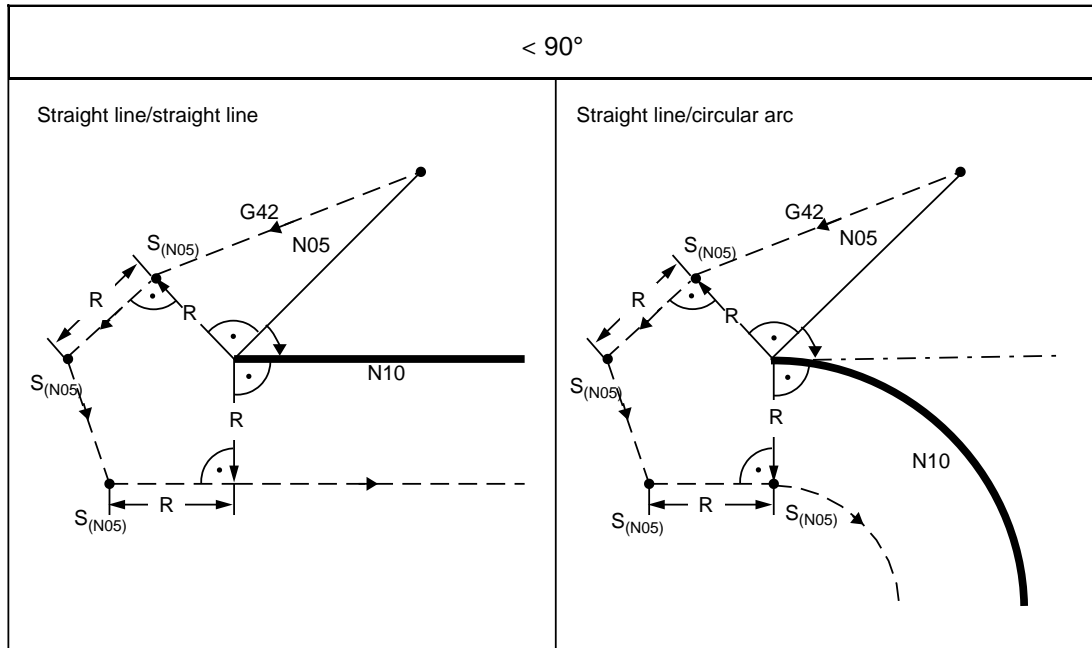


Selection of compensation mode for $> 180^\circ$

Selection of compensation mode



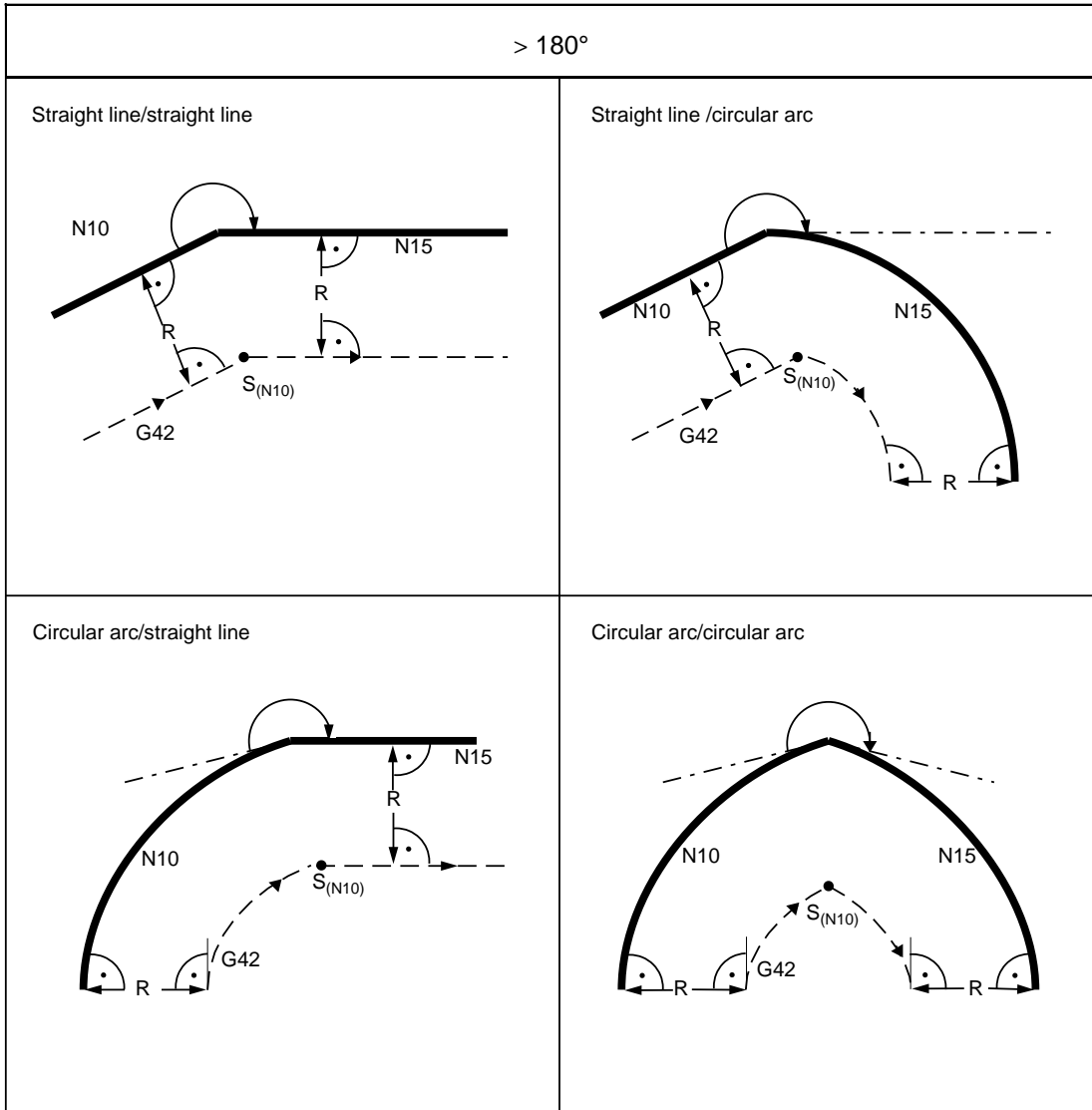
Selection of compensation mode for 90° 180°



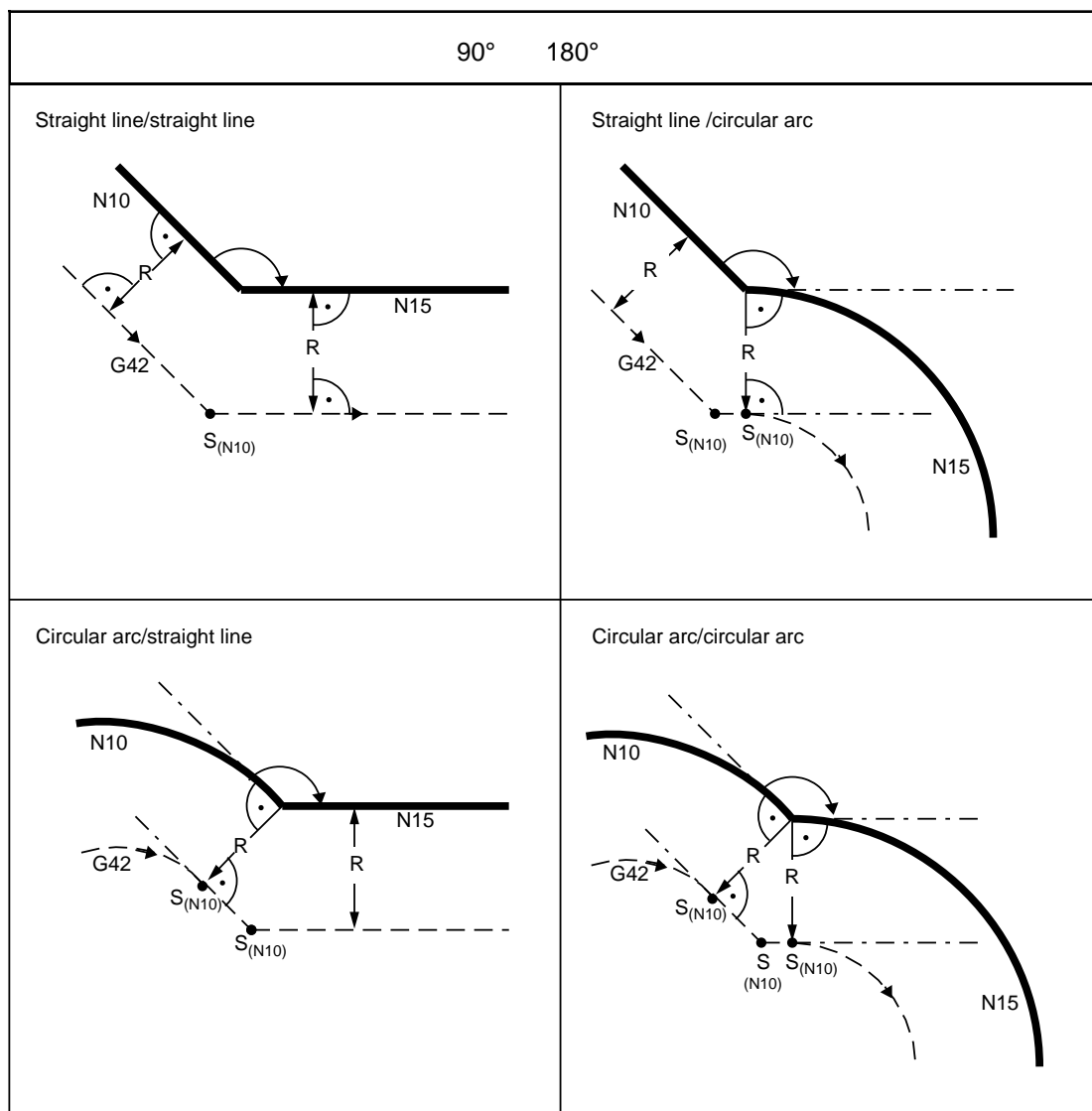
Selection of compensation mode for < 90°

9.2 CRC/TNRC in program

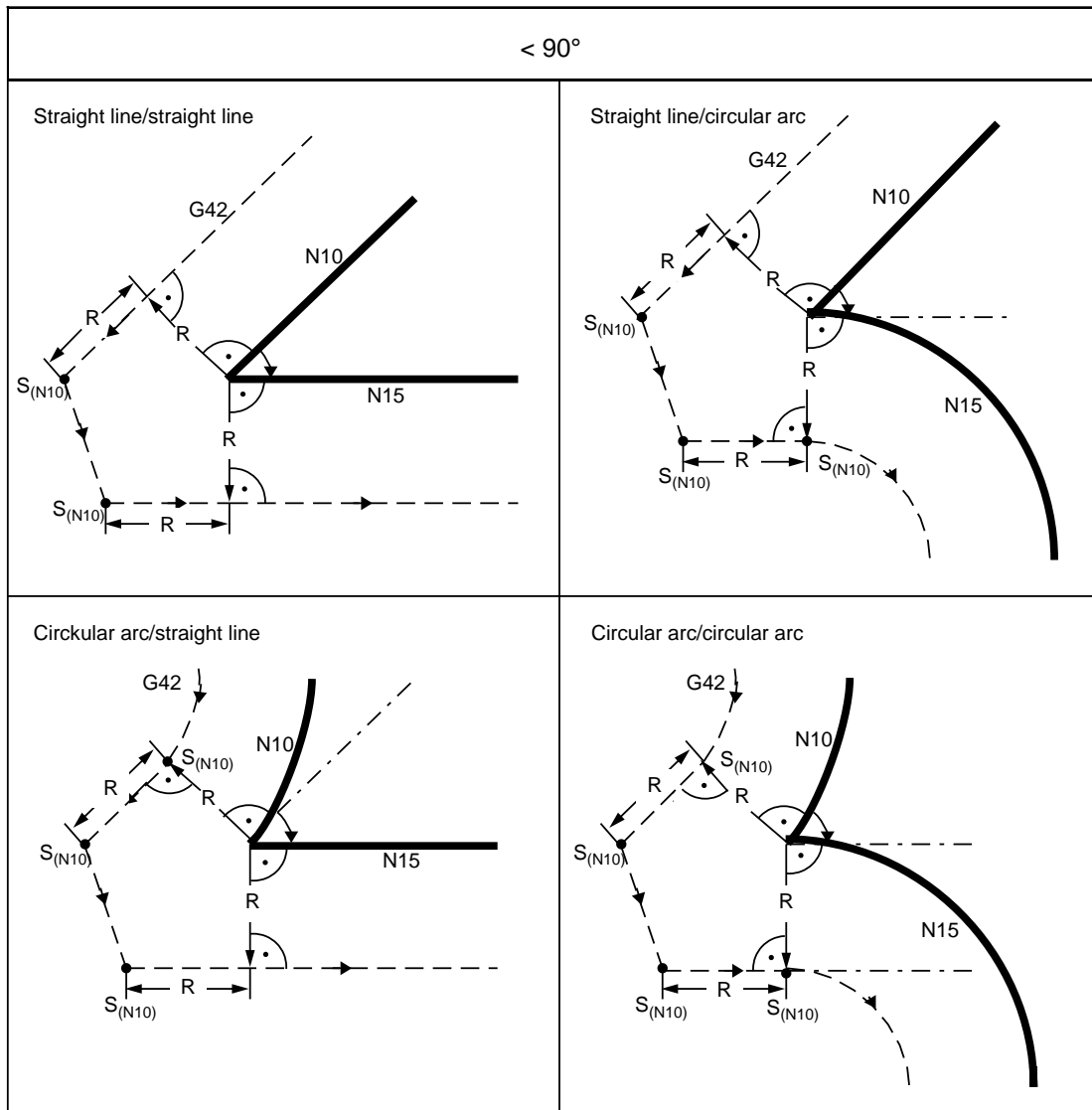
When CRC/TNRC is selected the control reads in two further blocks in advance during processing of the current block and calculates the intersection point of the compensated paths. The diagrams below show compensation for various transitions.



Compensation mode for various transitions for $> 180^\circ$



Compensation mode for various transitions for 90° 180°



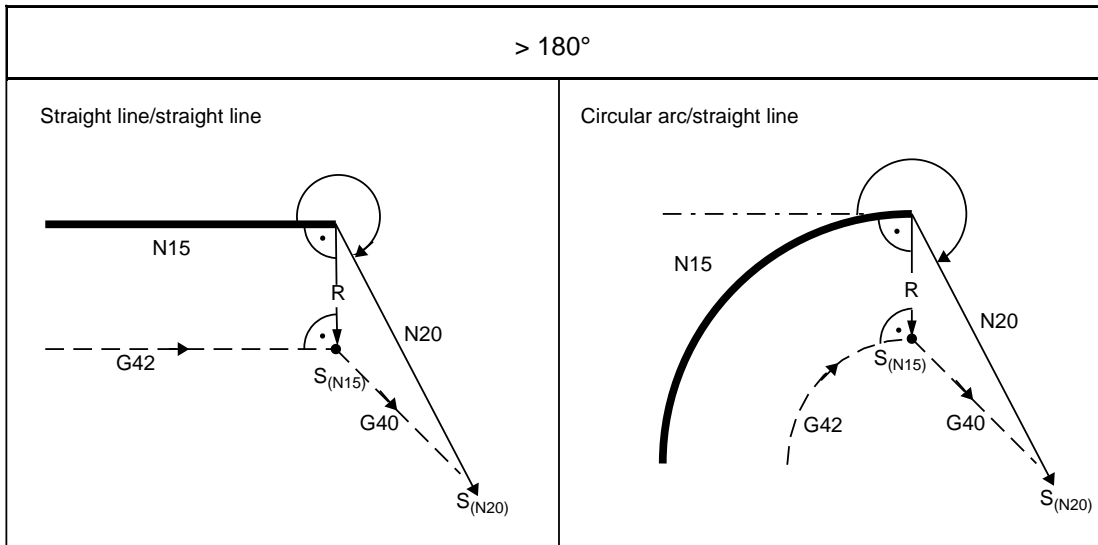
Compensation mode for various transitions for $< 90^\circ$

9.3 Cancellation of CRC/TNRC

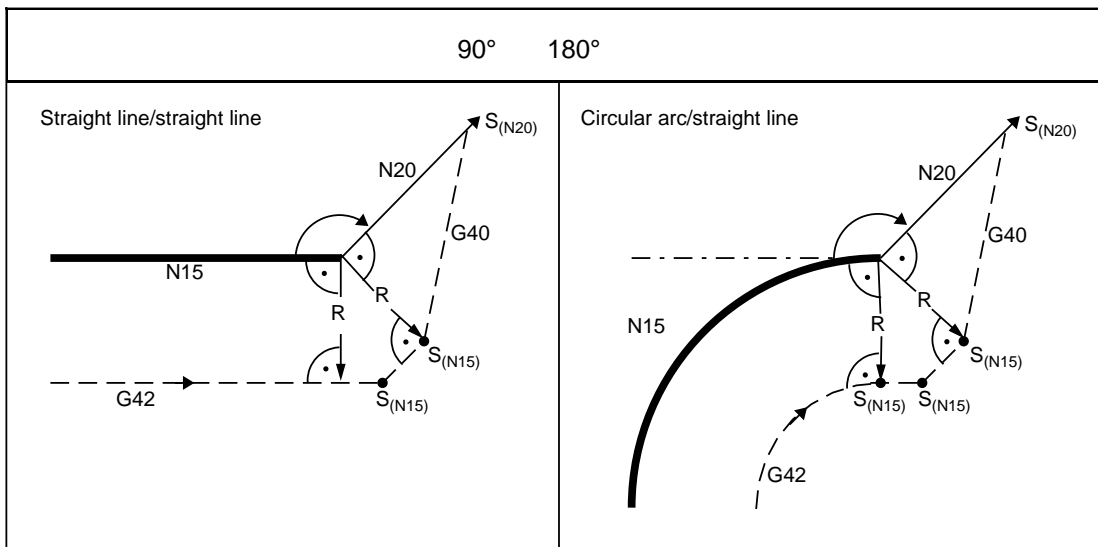
Compensation mode is cancelled using preparatory function G40.

Compensation mode can only be cancelled in a program block with G00 or G01.
Tool number D0 corresponds to compensation value 0.
It can also be used to cancel compensation.

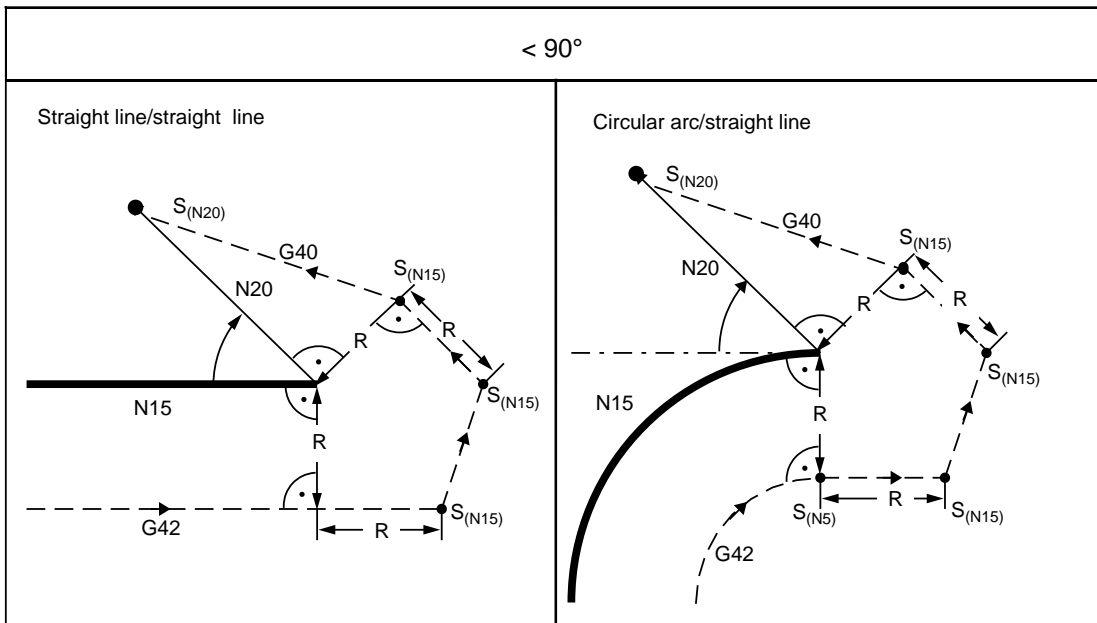
Cancellation of compensation



Cancellation of compensation mode for > 180°

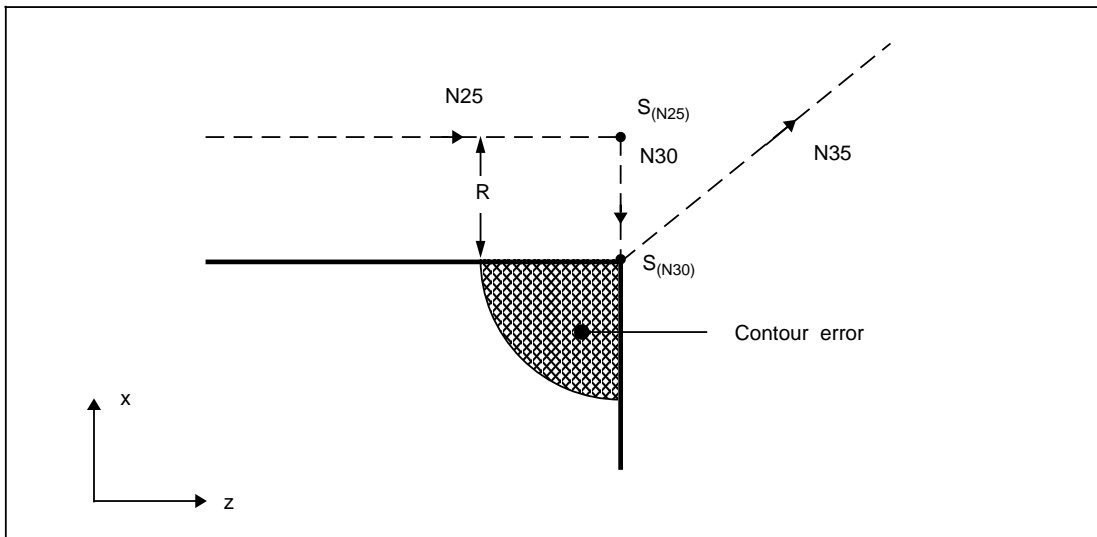


Cancellation of compensation mode for 90° 180°



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

Cancellation of compensation mode (special case)



Cancellation of TNRC in a block "distance = 0"

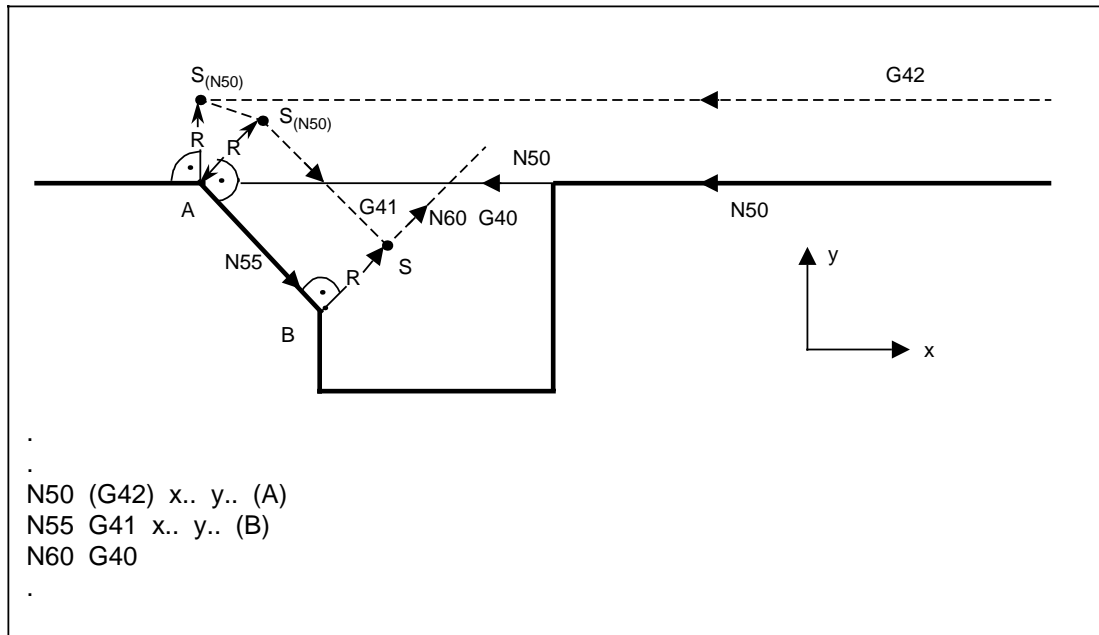
```

N25 G91 Z100 L_F
N30 G40 Z0 L_F
N35 Z100 X100 L_F
    
```

Contour error!

9.4 Changing direction of compensation

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

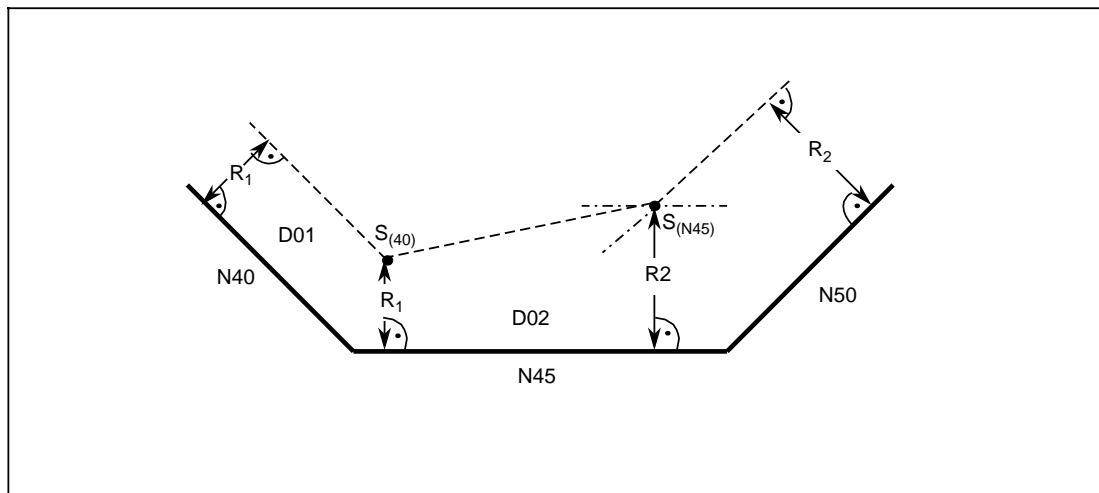


Changing direction of compensation

9.5 Changing offset number (D..)

When the offset number is changed:

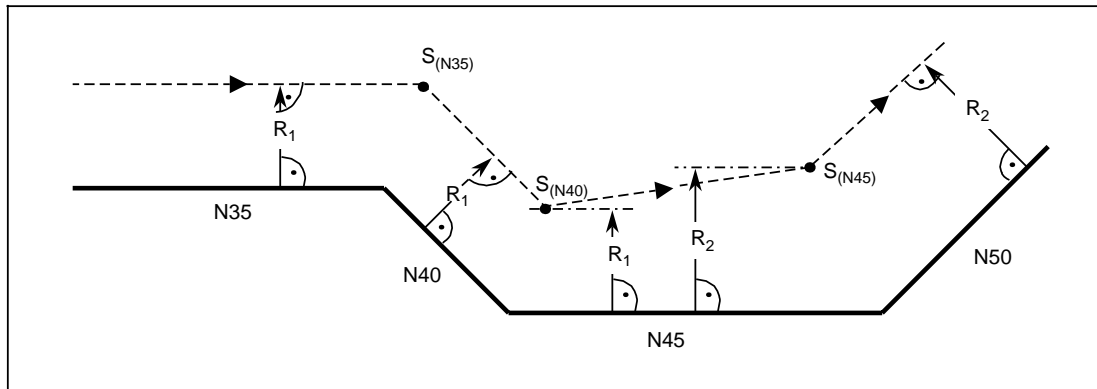
No block start intersection is calculated with the old compensation; a perpendicular vector with length R1 is created at the end position of the block with the old offset number; the block end intersection is calculated with the new compensation.



Changing offset number

9.6 Changing compensation values

The compensation values can be changed via the operator panel, via tape input or via the external tool offset. The new compensation value takes effect in the next block but one.



Changing compensation values

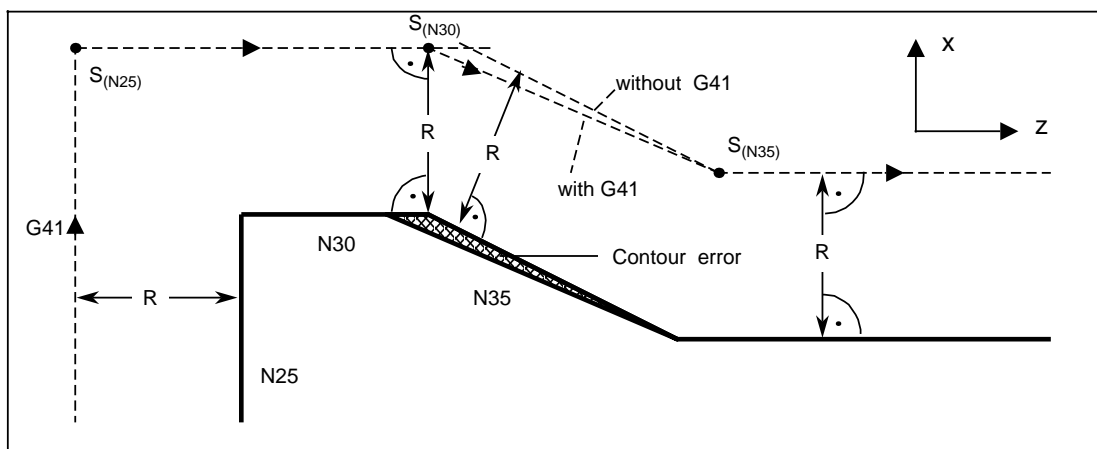
9.7 Repetition of selected G function (G41, G42) with same offset number

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

The block start intersection is calculated for the following block:

```
N20 G91 D10 G41 X... Z... LF
N25 X... LF
N30 Z... LF
N35 G41 X... Z... LF
N40 Z... LF
```

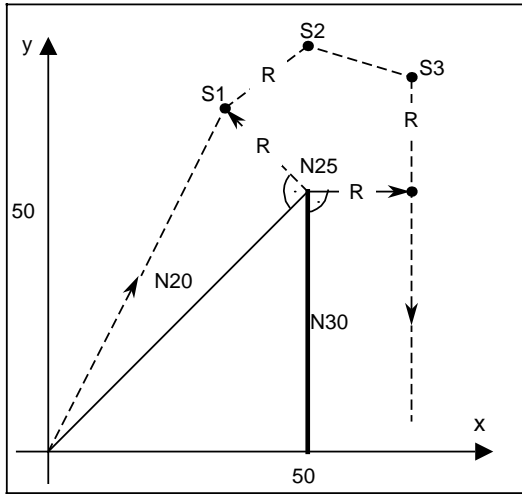
Attention: G41 repeated in N35. This results in a contour error!



Repeated selection

9.8 M00, M01, M02 and M30 with CRC/TNRC selected

M00, M01



M00, M01 with CRC/TNRC selected.

A distinction is made between two cases:

Case 1: N10 G01 F100 X0 Y0 L_F
 N20 G41 X50 Y50 L_F
 N25 H111 M00 L_F
 N30 Y0 L_F

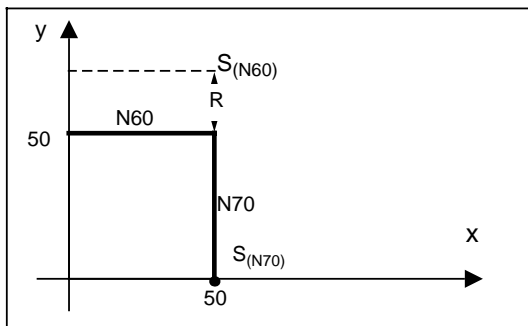
H and M functions are executed at point S3.

Case 2: N10 G01 F100 X0 Y0 L_F
 N20 G41 X50 Y50 L_F
 N25 H111 M00 Z-5 L_F
 N30 Y0 L_F

H and M functions and Z movement are executed at point S1.

M02, M30

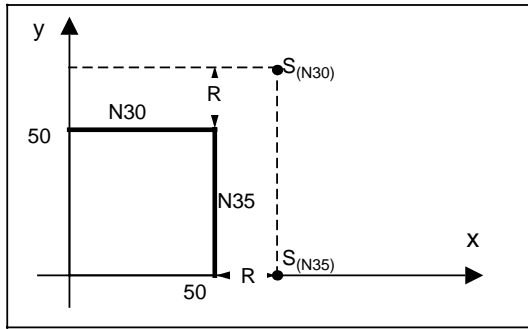
Cancellation with G40:



N60 X50 Y50 L_F
 N70 G40 X50 Y0 M30 L_F

The compensation is withdrawn one block before the block in which it was cancelled with G40 (here: N60).

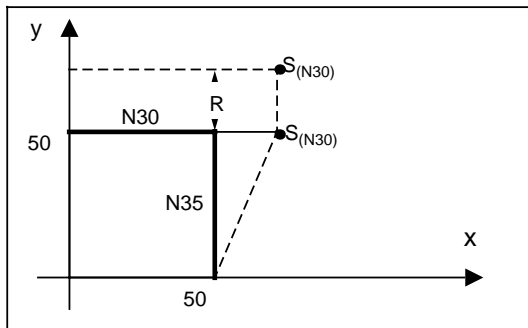
No path with G40 or programming of one axis in the final block:



```
N30 X50 Y50 LF
N35 Y0 LF
N40 G40 LF
N45 M30 LF
```

The compensation is **not** withdrawn.

G40 in the penultimate block or programming of two axes in the final block:



```
N30 X50 Y50 LF
N35 G40 Y0 LF
N40 M30 LF
```

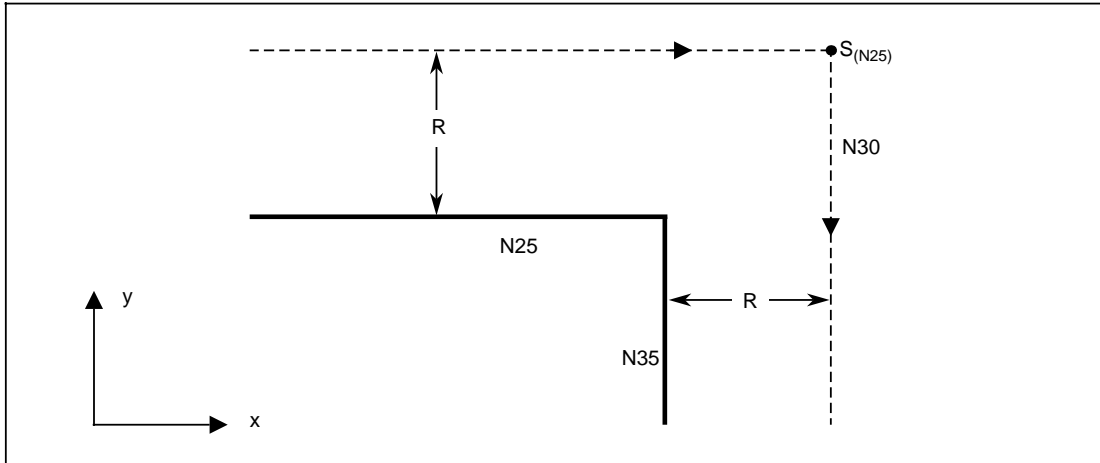
The compensation is withdrawn in N40.

9.9 CRC/TNRC 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:

- Blocks "without path"
Auxiliary functions, dwell time or a zero offset are programmed in the compensation plane instead of paths.
- Blocks with "distance = 0"
Path addresses are programmed, but there is no movement since the distance is 0.
- Blocks "not in compensation plane"
Axis addresses outside the compensation plane are programmed.

- **One** “auxiliary function block” between distances in the compensation plane

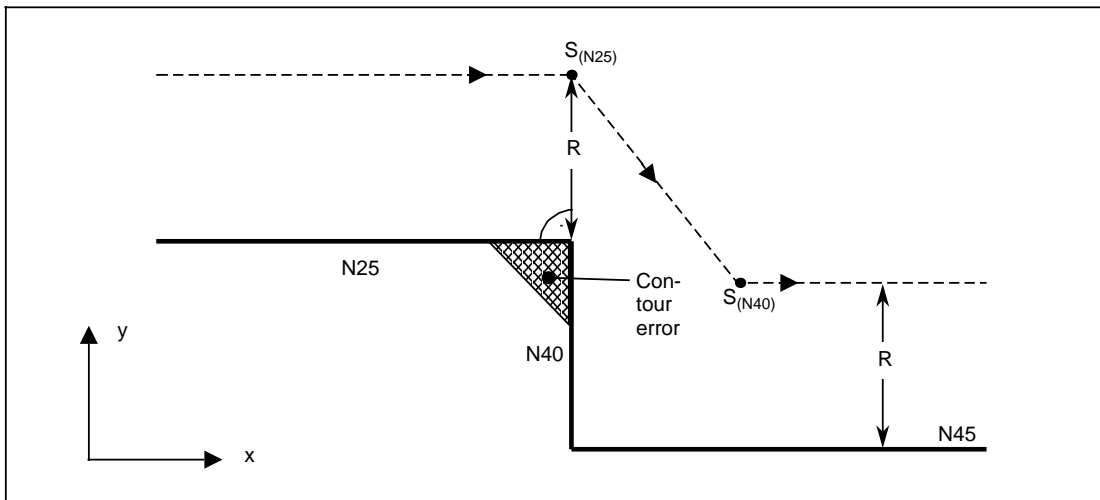


CRC/TNRC: One “auxiliary function block” between two movement blocks

```
N25 G91 X200 LF
N30 M08 LF
N35 Y-100 LF
```

Block N30 is executed at point S.

- **Two** “auxiliary function blocks” between distances in the compensation plane



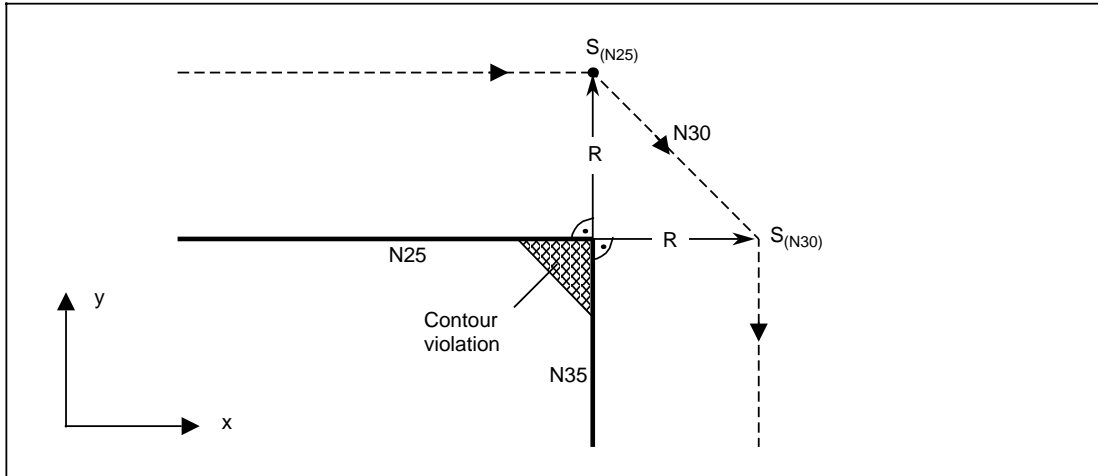
CRC/TNRC: Two “auxiliary function blocks” between two movement blocks

```
N25 G91 X200 LF
N30 M08 LF
N35 M09 LF
N40 Y-100 LF
N45 X200 LF
```

Blocks N30 and N35 are executed at point S_(N25).

Contour violation!

- **One** “distance = 0” block between distances in the compensation plane

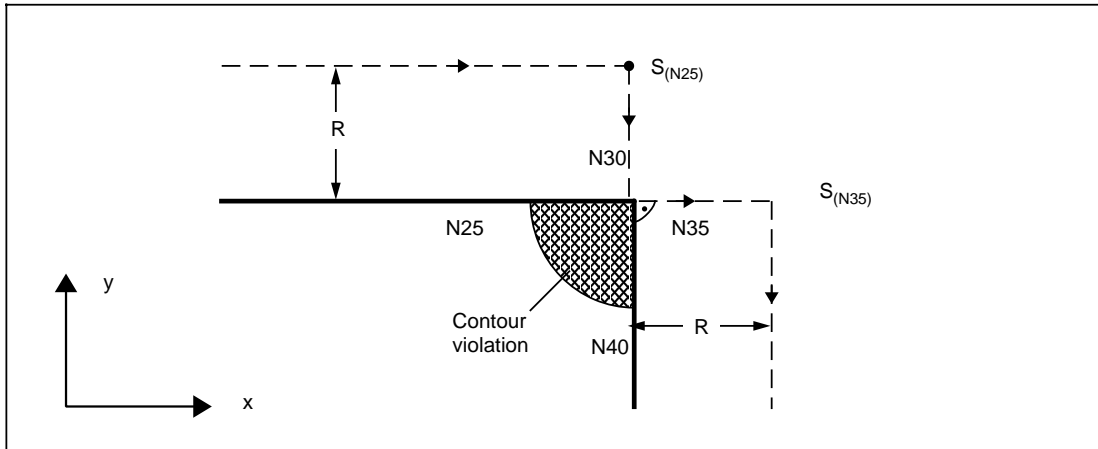


CRC/TNRC: One “distance = 0” block

```
N25 G91 X200 LF
N30 X0 LF
N35 Y-100 LF
```

Contour violation!

- **Two** “distance = 0” blocks between distances in the compensation plane

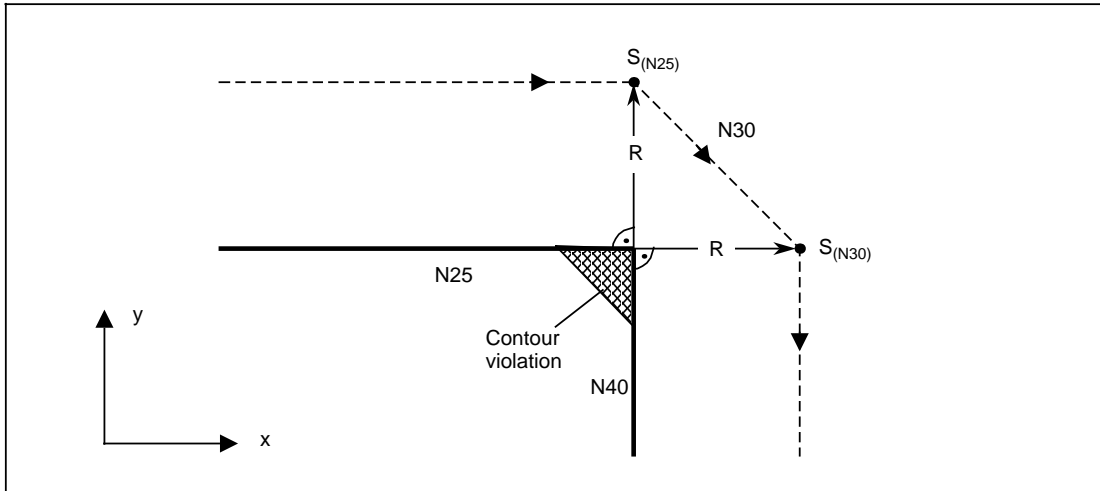


CRC/TNRC: Two “distance = 0” blocks

```
N25 G91 X200 LF
N30 X0 LF
N35 X0 LF
N40 Y-100 LF
```

Contour violation!

- **One** “distance = 0” block and one “auxiliary function block” between distances in the compensation plane



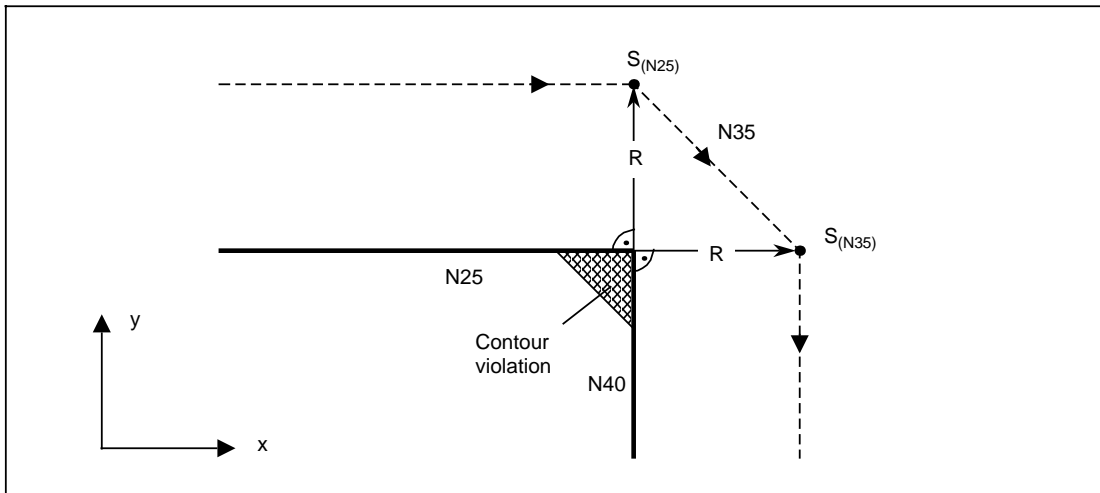
CRC/TNRC: One “distance = 0” block and one “auxiliary function block”

```
N25 G91 X200 LF
N30 X0 LF
N35 M08 LF
N40 Y-100 LF
```

Contour violation!

Block N35 is executed at point $S_{(N30)}$.

- **One** “auxiliary function block” and one “distance = 0” block between distances in the compensation plane



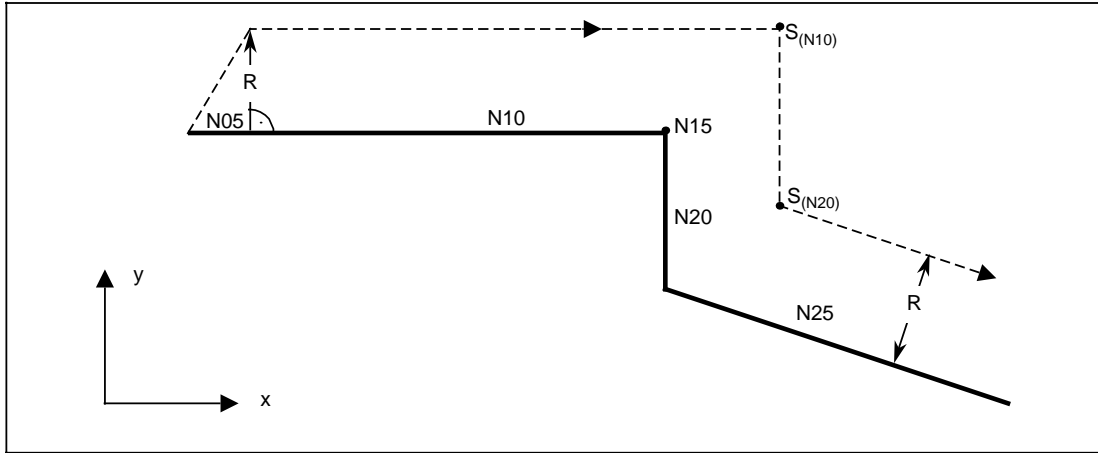
CRC/TNRC: One “auxiliary function block” and one “distance = 0” block

```
N25 G91 X200 LF
N30 M08 LF
N35 X0 LF
N40 Y-100 LF
```

Contour violation!

Block N30 is executed at point $S_{(N25)}$.

- **One** “not in compensation plane” block between distances in the compensation plane

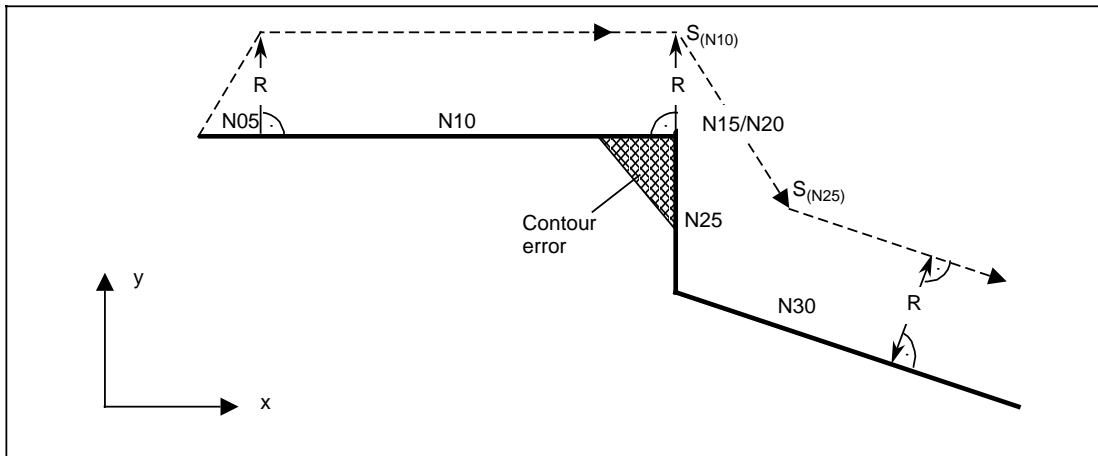


CRC/TNCR: One “not in compensation plane” block

```

N05 G91 G41 G01 G17 X100 F100 D01 LF
N10 X500 LF
*)N15 Z500 LF
N20 Y-150 LF
N25 X500 Y-150 LF
    
```

- **Two** “not in compensation plane” blocks between distances in the compensation plane



CRC/TNCR: Two blocks “not in compensation plane”

```

N05 G91 G41 G01 G17 X100 F100 D01 LF
N10 X500 LF
*)N15 Z500 LF
*)N20 Z-500 LF
N25 Y-150 LF
N30 X500 Y-150 LF
    
```

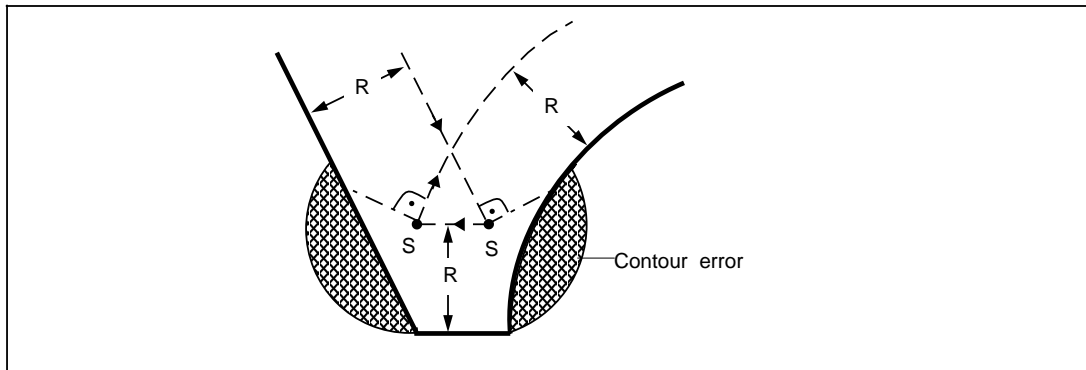
Contour violation!

*) Block not in the compensation plane

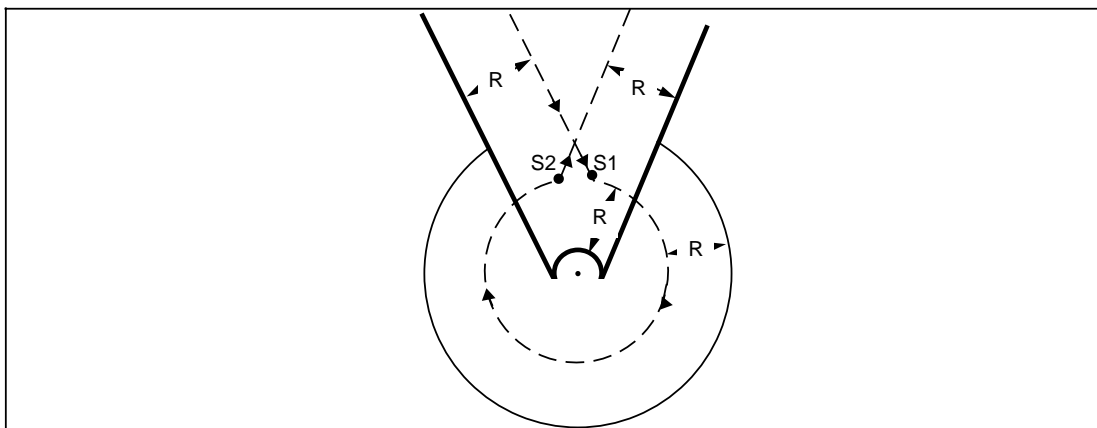
9.10 Special cases for CRC/TNRC

The control 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 control uses the next block but one. Contour errors can occur if the intermediate block is smaller than the selected compensation value.

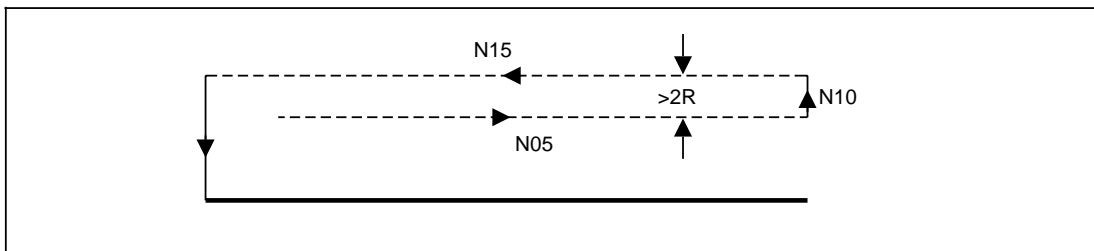
Machining is not interrupted, but an alarm is displayed.



CRC/TNRC special case: Intermediate block < compensation value



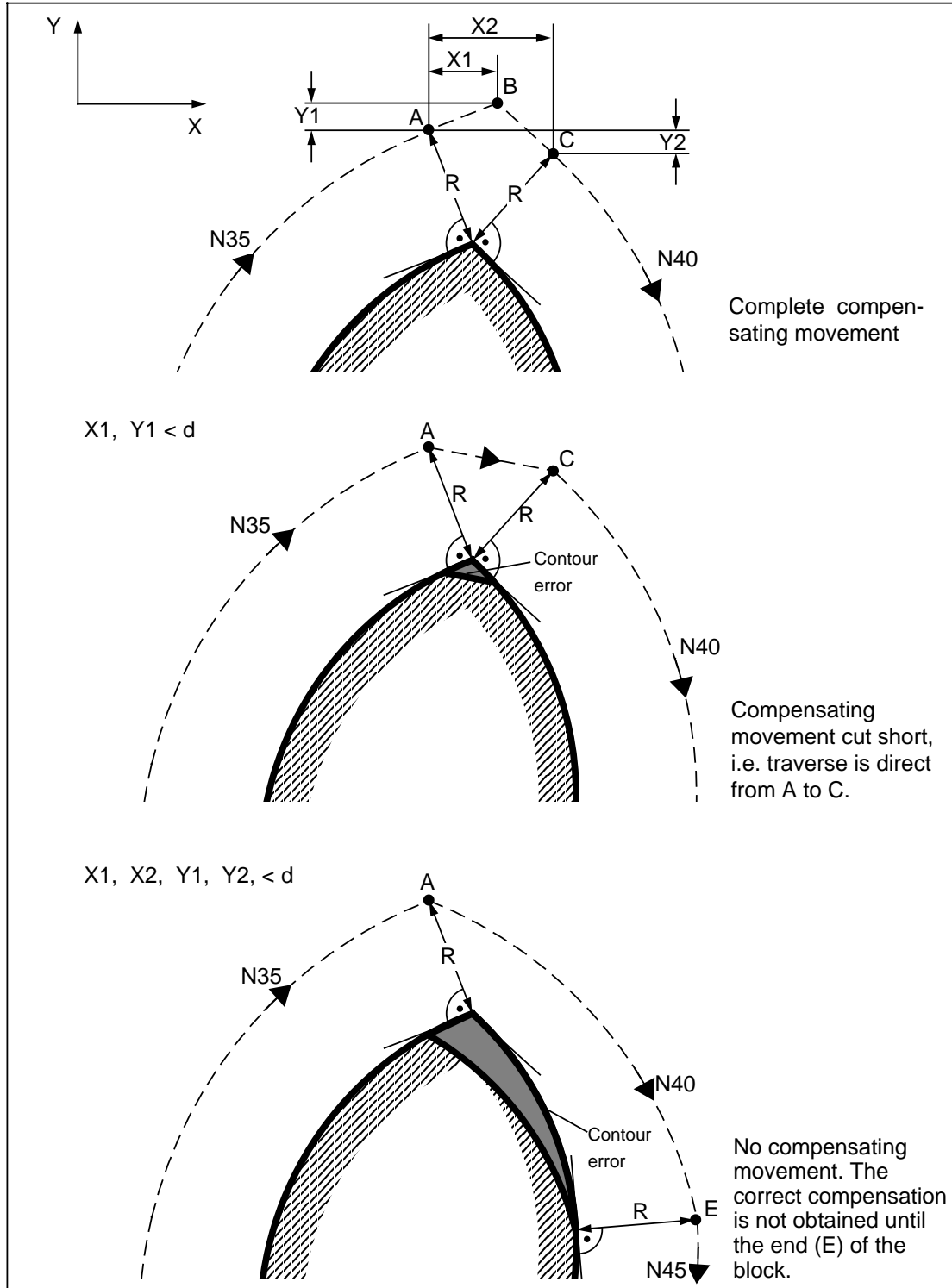
CRC/TNRC special case: Intermediate block too small for compensation



CRC/TNRC special case: Same direction of compensation and change in traversing direction

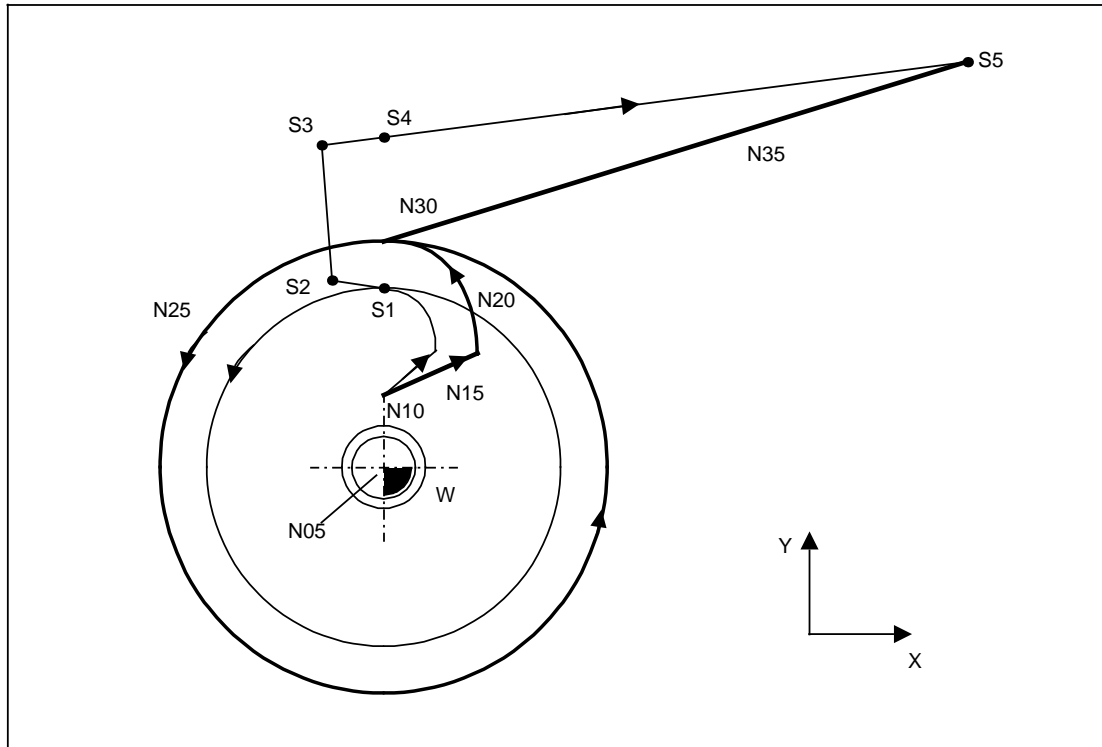
The direction of compensation for CRC/TNRC is retained and the traversing direction is reversed. The return path in N10 must exceed twice the cutter radius/tool nose radius, or the tool will move in the wrong direction.

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, distances AB and BC can be omitted by the NC. The course of the path in this case depends on tolerance d defined on start-up (max. 2000 units).

The block numbers are interchanged if CRC/TNRC generates intermediate blocks (including on selection and cancellation), and if an axis movement outside the compensation plane is programmed between these.



```

N05 G01 Z-5 LF
N10 X0 Y10 LF
N15 G41 D01 X20 Y20 LF
N20 G03 X0 Y40 I-20 J0 LF
N25 X0 Y40 I0- J-40 LF
N30 G00 Z50 LF
N35 G40 X80 Y60 LF

```

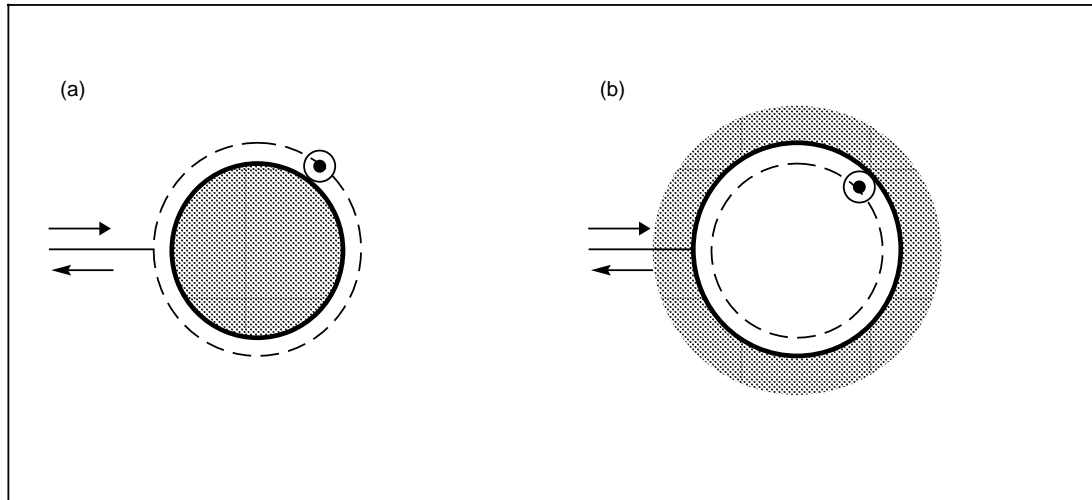
Points S1, S2, S3 and S4 belong logically to block N25. The machining sequence (visible in the single block) is as follows:

..., N20, N25 (S1), N30 (retraction), N25 (S2), N25 (S3), N25 (S4), N35... .This sequence is also valid if N25 is 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.

In the cutter centre path shown in the diagram below (a) a positive compensation value has been entered. A negative compensation value in conjunction with the same machining program will effect the machining shown on the right of the diagram (b).



If the program is written as shown in (b) with a positive compensation value, a negative compensation value will effect machining as shown in (a). The two operations are distinguished by entering a positive or negative compensation value.

10 Siemens Standard Cycles (Option)

10.1 Introduction

Machining cycles are available for frequently recurring standard machining processes.

The cycles can be assigned the desired data by direct input of the R parameters in the program or using parameterization support during operator guidance (see section 12).

Machining cycles are called in the part program or subroutine.

In the following examples the R parameters have been assigned directly in the part program.

The cycles described here can be modified if required. Note any additional information provided by the machine tool manufacturer.

All the cycles end with the preparatory functions G00, G60, G90. When continuing the program, any other desired G functions must be reprogrammed in addition.

Overview of subroutine numbers:

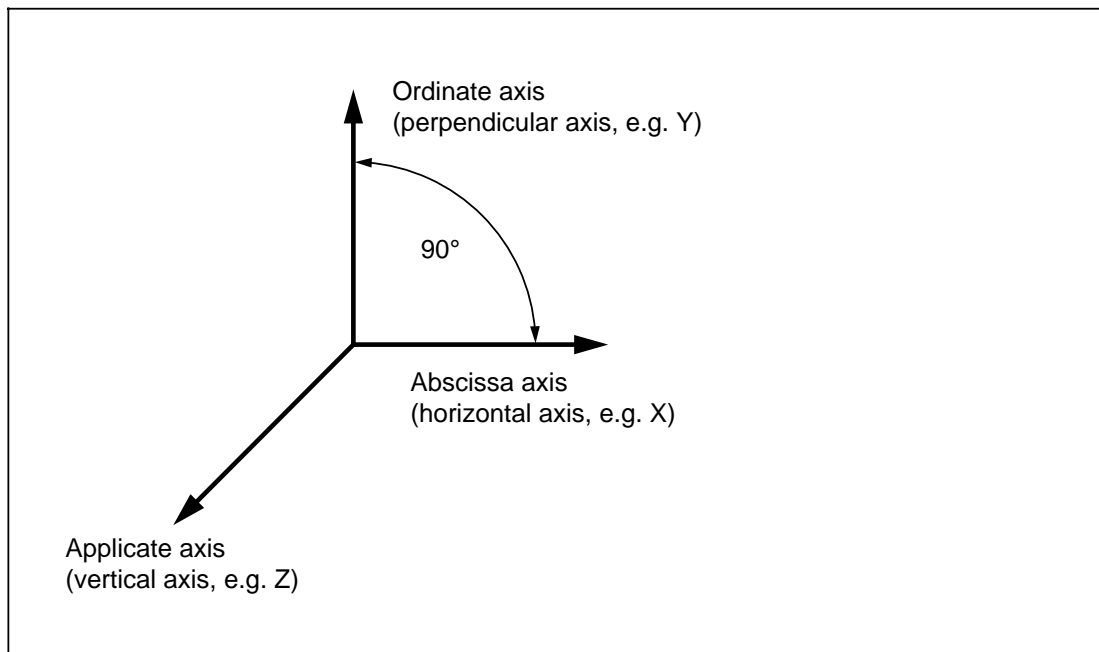
L No.	Function
L01 . . L05	User assignable
L06	Siemens
L07 . . L80	User assignable
L81 . . L99	Siemens
L100 . . L799	User assignable
L800 . . L999	Siemens

The descriptions of the cycles presuppose the following:

- The user must be familiar with the relevant Programming Guide, Operator's Guide, User's Guide.
- Polar coordinate programming is required for the drilling, boring and milling patterns.
- The "blueprint programming" Option is a precondition for grooving cycle L93.

The control does not provide the programmer with screen guidance in the case of the "blueprint programming" option.

- The current plane must be selected before calling the cycles with G16 or G17 to G19. The infeed (drilling) axis is always the axis perpendicular to the current plane.



Machine data (MD) 157

Machine data MD 157 is used to identify the control type of the cycles used and the software version on all System 800 controls. In the interest of consistency, these data are used in cycles from the SINUMERIK 805 through to the SINUMERIK 880.

Control type	MD 157
805, turning cycles	06 xx
805, milling cycles	07 xx
810 T	11 xx
810 M	12 xx
810 G	13 xx
820 T	21 xx
820 M	22 xx
850 T	51 xx
850 M	52 xx
880 T	81 xx
880 M	82 xx

Example:

SINUMERIK 805 with NC software version 2.1 and milling cycles:

MD 157 = 07 21

The following alarms are output in the cycles:

Alarm	Description
4100	No D number active (L901-904, L930, L93-L95)
4101	Tool radius = 0 (L903, L930)
4102	Cutter radius too great (L901, L903, L904, L930)
4103	Tool too wide (L93)
4120	No direction of spindle rotation programmed (L84)
4121	Spindle outside tolerance range (L84)
4140	Machined part diameter too small (L94)
4180	Option not available (L94)
4200	Check definition R (Nxxx)

Example: 4200 N 32 Check definition R (Nxxx)

Incorrect definition of parameter R32 was discovered in the cycle.

Note:

The alarm texts are supplied on the diskette as subroutine L998 if the standard cycles are ordered. This subroutine must then be read into the control. In the event of faults, users can also call the standard cycle alarms in the cycles they have generated themselves. If these alarms are insufficient (different fault description required, for instance), users can store their own alarm texts in the control under numbers 5000 - 5099.

The standard cycle alarms (4000 - 4200) and user cycle alarms (5000 - 5099) can be called by means of @4c0 (Section 11 Programming of Cycles).

10.2 Machining cycles for drilling, boring and milling

(Precondition: polar coordinate programming)

The following drilling and milling cycles are available for machining:

L81 Drilling, centering	L903 Milling rectangular pocket
L82 Drilling, spot facing	
L83 Deep hole drilling	
L84 Tapping (with/without encoder)	
L85 Boring 1	
L86 Boring 2	
L87 Boring 3	
L88 Boring 4	
L89 Boring 5	

Milling cycle L903 is programmed absolutely. The axis name, radius and angle can be selected with variable addresses by machine data.

The current plane must be selected before calling the cycles with G16 or G17 to G19. The infeed (drilling) axis is always the axis perpendicular to the current plane. This permits the use of the drilling/boring and milling cycles in all axes.

The length compensation must be selected before calling the cycles. The length compensation of the tool (cutter, drill) is always effective perpendicular to the selected plane and also remains active after the end of the cycle.

The associated feedrate, spindle speed and direction of spindle rotation must be programmed in the part program (except for cycles in which the values can be programmed as input parameters).

The centre-point coordinates R22 and R23 are programmed in a right handed system, e.g.:

G17 plane R22 = X, R23 = Y, infeed axis = Z

G18 plane R22 = Z, R23 = X, infeed axis = Y

G19 plane R22 = Y, R23 = Z, infeed axis = X

10.2.1 Drilling and boring cycles G81 to G89

A drilling/boring (work) cycle is a sequence of individual machine movements for drilling, boring, tapping etc. as determined by DIN 66025. Drilling and boring cycles G81 to G89 take the form of subroutines L81 to L89 which have to be read into the control.

Users can also freely determine drilling and boring cycles that deviate from the standard in conjunction with more favourable machine or tool-specific circumstances. The parameters R00 to R17 are used in the subroutines and numerically defined in the superordinate program for the variable values in the cycles (reference plane, final drilling depth, drilling feedrate, dwell time etc.).

With the parameters initialized in the program, subroutines L81 to L89 can be called modally with G81 to G89 and cancelled with G80. G80 to G89 can be selected and cancelled only in a single program level (cf. example).

Example: Call G81 (drilling cycle)

<code>%81</code>	
<code>N8101 G90 F130 S710 M03 LF</code>	
<code>N8102 G00 D01 Z50 T03 LF</code>	
<code>N8103 X10 Y15 LF</code>	Approach 1st drilling position
<code>N8104 G81 R2=2 R3=-15 R10=10 LF</code>	Call L81, parameter assignment
<code>N8105 X30 Y40 LF</code>	Approach 2nd drilling position and automatic call of L81
<code>N8110 G80 Z50 LF</code>	Cancel L81
<code>N8115 M30 LF</code>	

The hole position must have been approached in the current plane by the calling program. The drilling/boring cycle called with G81 to G89 is executed in every NC block until cancelled with G80. Users must therefore note that the drilling/boring cycle also becomes effective after NC blocks without positional data.

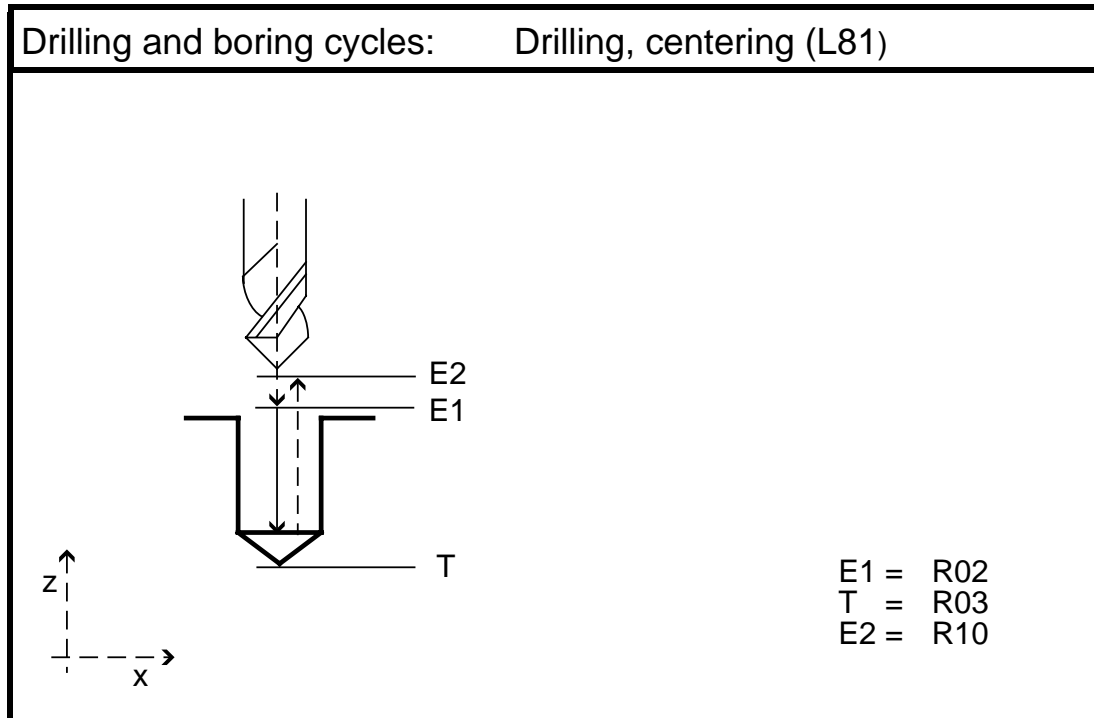
The following parameters are used in cycles L81 to L89:

Symbol	Parameter	Description
t1	R00	Dwell time at start point (swarf removal)
T1	R01	Input first drilling depth without sign (incremental)
E1	R02	Reference plane (absolute)
T_	R03	Final drilling depth (absolute)
t_	R04	Dwell time at drilling depth (chip breaking)
Db	R05	Amount of degression (incremental)
Me	R06	Direction of rotation for retraction (M03/M04)
M_	R07	Direction of rotation (M03/M04)
	R08	Tapping with/without encoder
P	R09	Lead (only for tapping with encoder)
E2	R10	Retract plane (absolute)
	R11	Deep hole drilling with chip breaking or swarf removal
	R11	Number of drilling axis
sa	R12	Retract travel (horizontal with sign) (incremental)
so	R13	Retract travel (vertical with sign) (incremental)
Ft	R16	Feedrate
Fr	R17	Retract feedrate

Subroutine L81: Drilling, centering

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
E2	R10	Retract plane (absolute)



Example: Drilling, centering

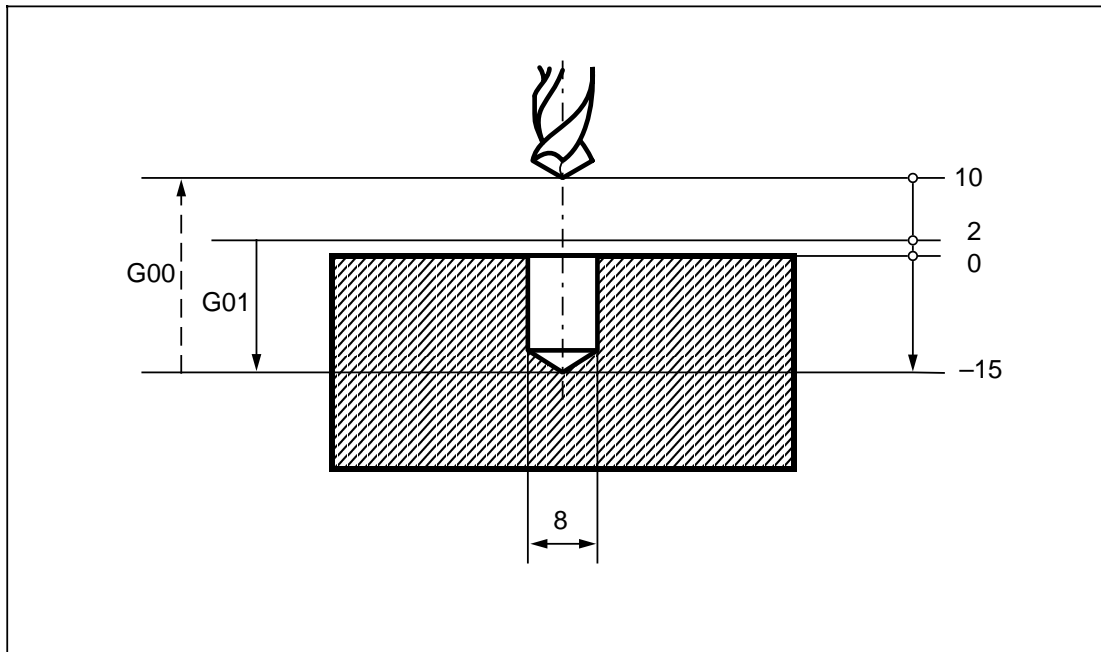
```

%81
N8101 G90 F130 S710 M03 LF
N8102 G00 D01 Z50 T03 LF
N8103 X10 Y15 LF
N8104 G81 R2=2 R3=-15 R10=10 LF
N8105 X25 Y60 LF

N8106 G80 Z50 LF
N8107 M30 LF

```

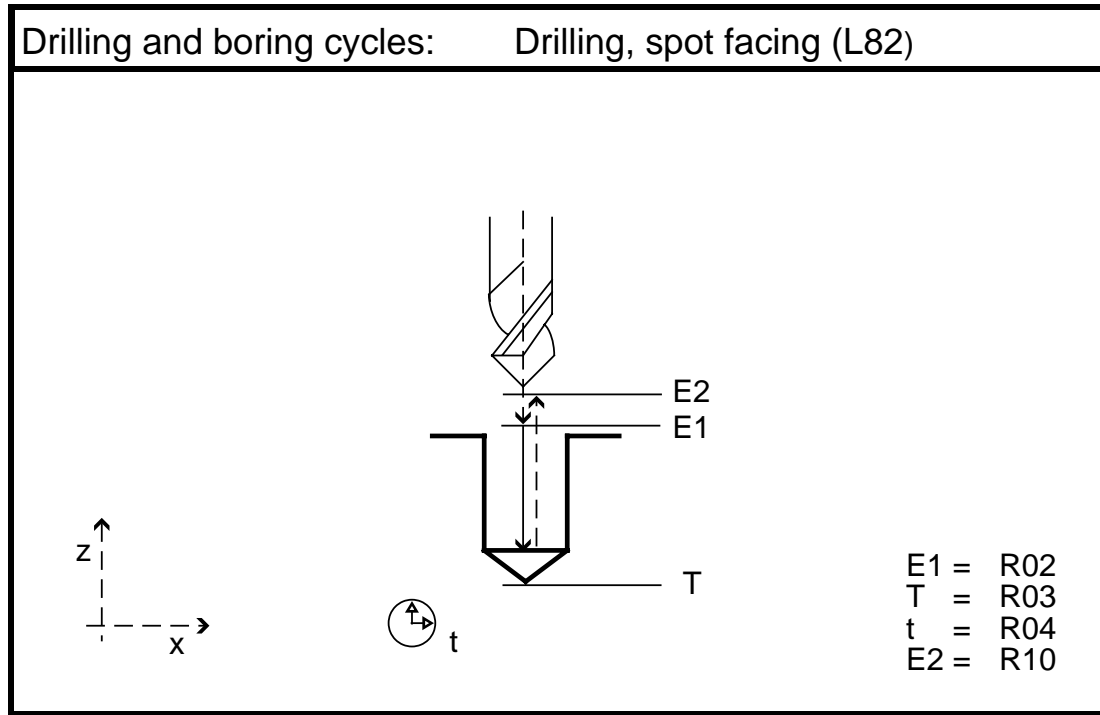
Select 1st drilling position
Call drilling cycle, 1st hole
Select 2nd drilling position and
automatic call of drilling cycle, 2nd hole
Cancel L81



Subroutine L82: Drilling, spot facing

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
t	R04	Dwell time at drilling depth (chip breaking)
E2	R10	Retract plane (absolute)



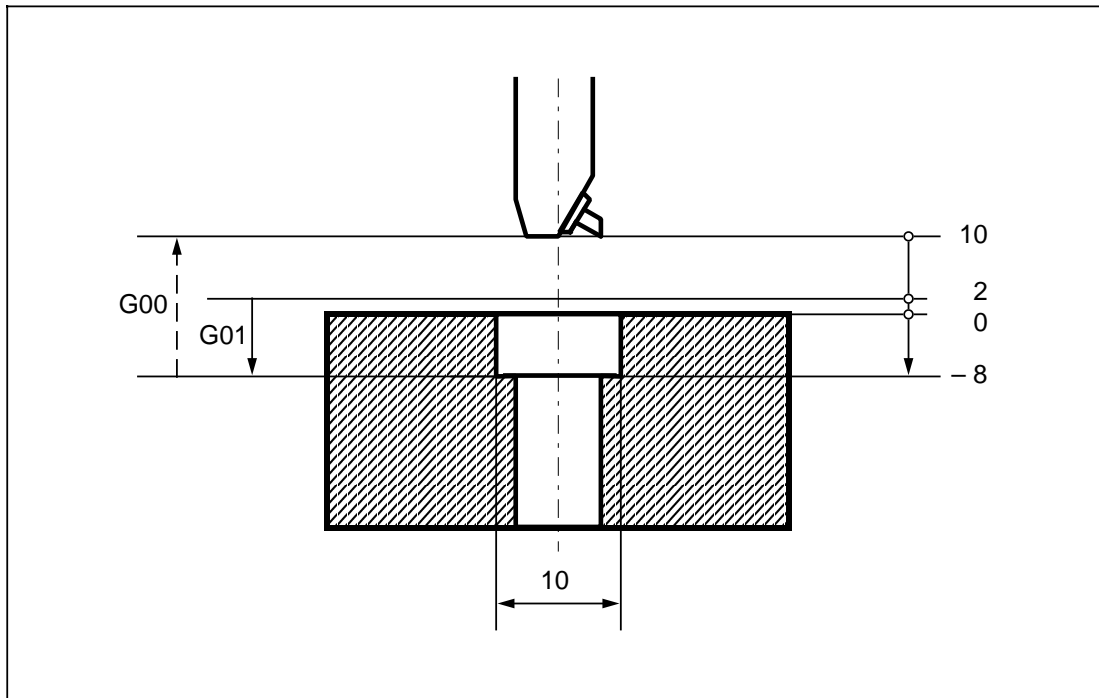
Example: Drilling, spot facing

```

%82
N8201 G90 F130 S710 M03 LF
N8202 G00 D01 Z50 T03 LF
N8203 X10 Y15 LF
N8204 R2=2 R3=-8 R4=1 R10=10 L82 P1 LF
N8205 X25 Y60 LF
N8206 R2=2 R3=-8 R4=1 R10=10 L82 P1 LF
N8207 Z50 LF
N8208 M30 LF

```

Select 1st drilling position
Call drilling cycle, 1st hole
Select 2nd drilling position and
Call drilling cycle, 2nd hole

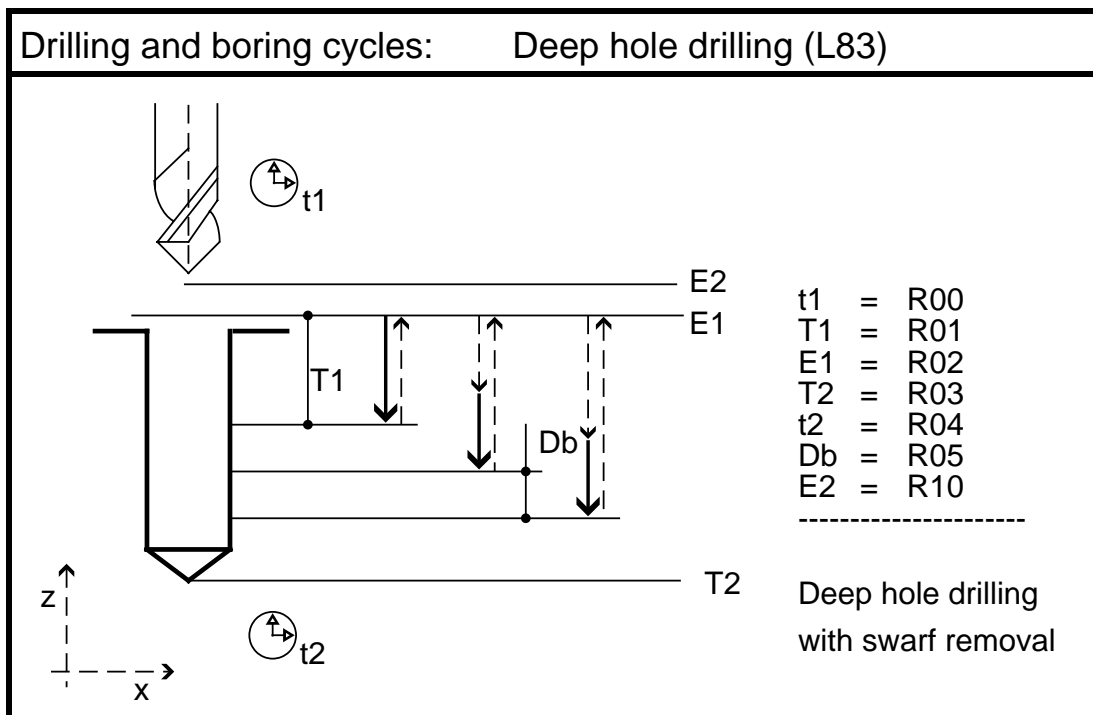


Subroutine L83: Deep hole drilling

This cycle is used to machine deep holes. On reaching each infeed depth the drill can be retracted to the reference plane for swarf removal.

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
t1	R00	Dwell time at start point (swarf removal)
T1	R01	First drilling depth; input without sign (incremental)
E1	R02	Reference plane (absolute)
T2	R03	Final drilling depth (absolute)
t2	R04	Dwell time at drilling depth (chip breaking)
Db	R05	Amount of degression; input without sign (incremental)
E2	R10	Retract plane (absolute)
	R11	0 = with chip breaking, 1 = with swarf removal



Drilling depth

R01 must be input as an incremental value without sign.

T2 R03: Final drilling depth (absolute)

1. Execution of the first drilling stroke according to the programmed R01 = first drilling depth.
2. Execution of the second drilling stroke resulting from R01 = first drilling depth minus R05 = amount of degression. If the drilling stroke calculated internally in the cycle is smaller than the amount of degression, the subsequent drilling strokes are executed with the value of R05 = amount of degression until the remaining drilling depth has been reached.
3. If the remaining drilling depth $> R05 =$ amount of degression and $< 2 \cdot R05$, it is divided into two drilling strokes.

$$R05 < a < 2 \cdot R05 \quad (a = \text{remaining drilling depth})$$

The retract movement after execution of each drilling stroke depends on the programming of R11 = chip breaking/swarf removal.

R11: Chip breaking/swarf removal

If R11 is initialized with 0, the drill retracts by 1 mm for chip breaking after each drilling depth is reached.

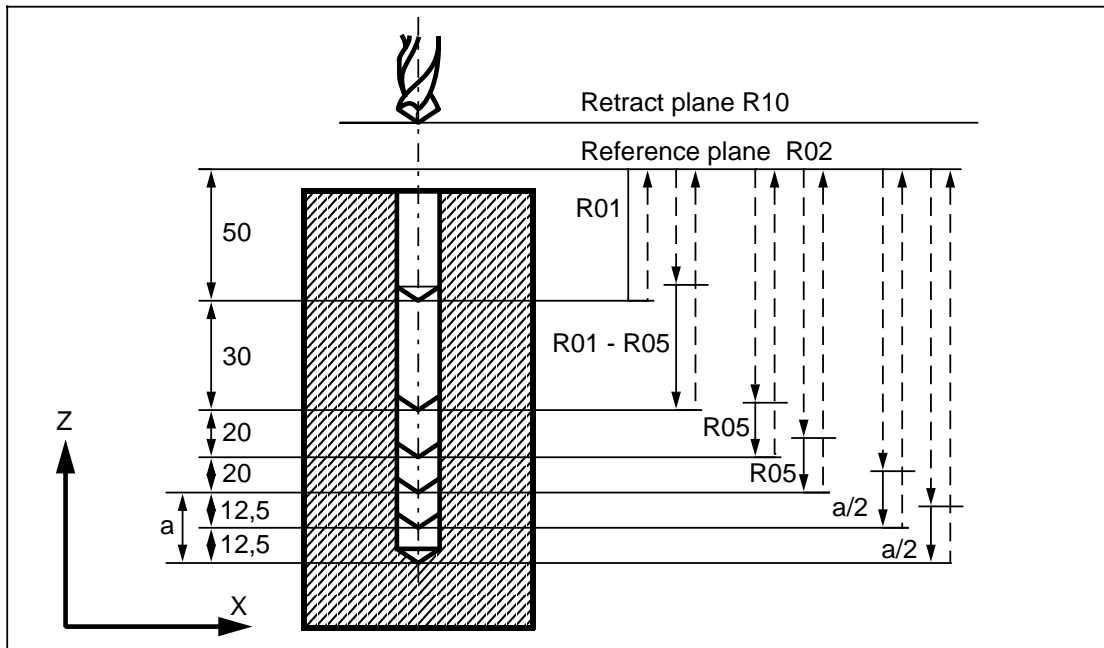
If R11 is initialized with 1, the drill travels to the reference plane for swarf removal after each drilling depth is reached.

Example: Deep hole drilling

```

%83
N8310 G90 F30 S500 M03 LF
N8320 G00 D01 Z50 T03 LF
N8330 X40 Y40 LF                               Select drilling position
N8340 R0=1 R1=50 R2=4 R3=-141
      R4=1 R5=20 R10=10 R11=1 L83 P1 LF       Call drilling cycle
N8350 Z50 LF
N8360 M30 LF

```



Subroutine L84: Tapping for machines with and without encoder

Cycle L84 permits tapping both with and without an encoder. A compensating chuck must be used in both cases.

The cycle determines from machine data 5013.1 whether the thread is to be tapped with or without an encoder.

MD 5013.1 = 1: tapping without encoder, MD 5013.1 = 0: tapping with encoder.

The spindle and feedrate override switches must be set at 100 %.

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
t	R04	Dwell time at thread depth
Me	R06	Direction of rotation for retraction (M03/M04)
Ma	R07	Direction of rotation after cycle (M03/M04)
	R08	Tapping 1 = with encoder, tapping 0 = without encoder
P	R09	Lead
E2	R10	Retract plane (absolute)
	R11	Number of drilling axis

t R04: Dwell time at thread depth

The dwell time is effective only for tapping without an encoder.

Me R06: Direction of rotation for retraction

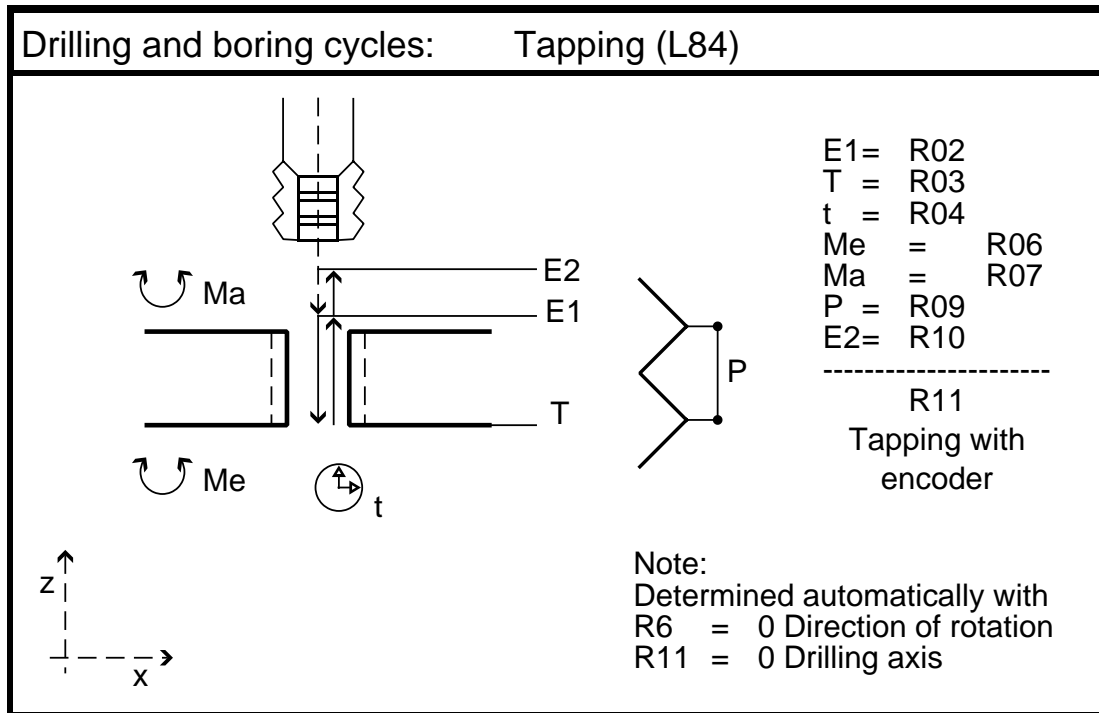
If the direction of spindle rotation is to be reversed automatically, R06 must be initialized with 0. If MD 5013.1 indicates no encoder, R06 **must** be programmed or the **error message** 4120 (no direction of spindle rotation programmed) will appear.

Ma R07: Direction of rotation after cycle

If the tapping cycle is called with modal function G84, the cycle requires a direction of rotation for execution of the subsequent tapped holes. This is programmed in R07.

For machining the first tapped hole, the direction of rotation must be written in the part program with M03 or M04 before the cycle is called. This also applies in the case of a one-off call.

If R06=0 is programmed (automatic reversal of direction of spindle rotation), R07 does **not** have to be initialized.

**R08: Tapping with/without encoder**

If tapping is to be performed without an encoder although one is available (MD 5013.1 = 0), R08 must be initialized with 0.

If MD 5013.1 = 1 without encoder and R08 = 1 with encoder are selected, R08 is ignored.

P R09: Lead

The lead is effective only in conjunction with tapping **with** encoder. The required feedrate is calculated from the entered spindle speed and the lead.

A feedrate value has to be entered in the part program in the case of tapping without encoder.

R11: Number of drilling axis

SINUMERIK 850M/SINUMERIK 880M:

If cycle L84 is used in these control types, R11 is ineffective. The drilling axis is determined from the selected plane.

SINUMERIK 805:

If cycle L84 is used in these control types, R11 **can** be initialized with the number of the drilling axis; otherwise (R11 = 0 programmed) the drilling axis is determined from the selected plane.

If R11 is initialized with the number of the drilling axis, however, no STOP DECODING is output in the cycle. L84 is processed faster in this case.

Example 1: Tapping with encoder

MD 5013.Bit 1=0

```
%1  
N05 G90 D01 T03 S500 M03 LF  
N10 G0 X20 Y20 Z15 LF  
N15 R2=2 R3=-25 R4=0 R6=0  
R7=4 R8=1 R9=1.25 R10=10 LF  
N20 L84 P1 LF  
N25 G0 X200 Y200 Z100 LF  
N30 M30 LF
```

Select drilling position

Call drilling cycle

Example 2: Tapping without encoder

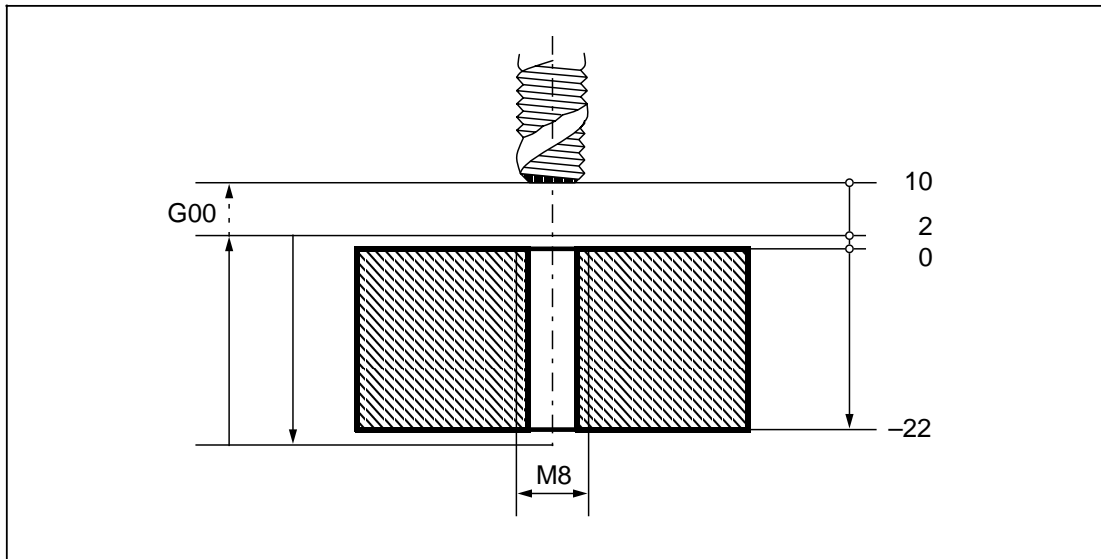
MD 5013.Bit 1=1

```
%2  
N05 G90 D01 T03 S500 M03 LF  
N15 G0 X20 Y20 Z15 LF  
N20 G1 F1.25 LF  
N25 R2=2 R3=-25 R4=1 R6=4  
R7=3 R8=0 R9=0 R10=10 LF  
N30 L84 P1  
N35 G0 X200 Y200 Z100 LF  
N40 M30 LF
```

Select drilling position

Feedrate value

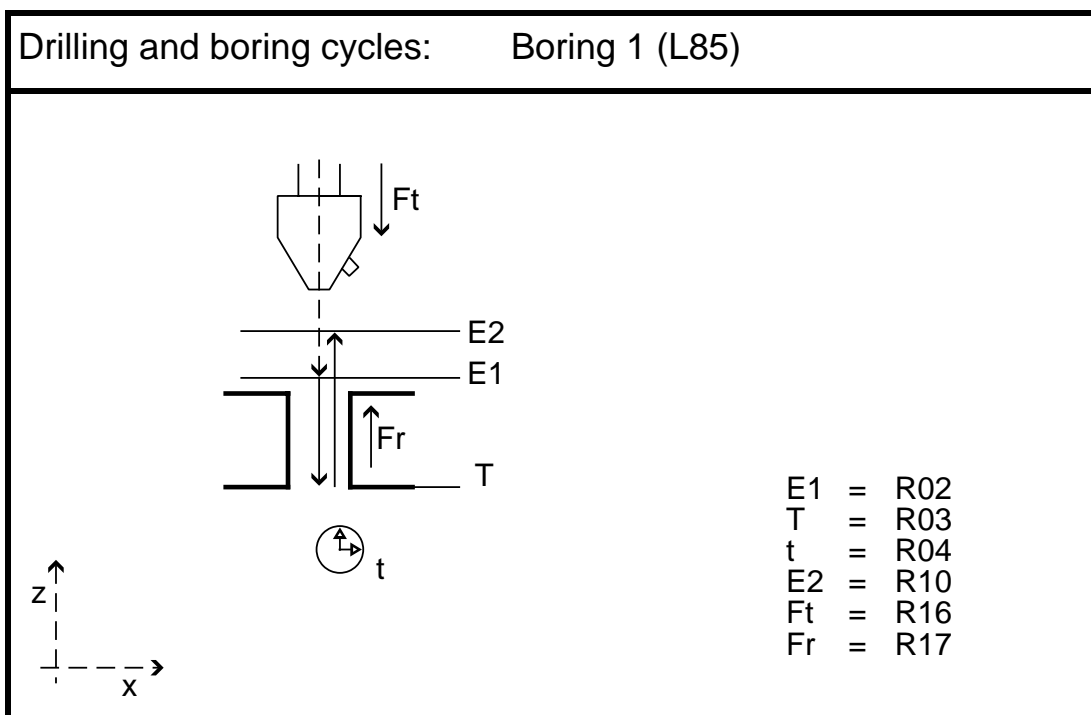
Call drilling cycle



Subroutine L85: Boring 1

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
t	R04	Dwell time at drilling depth (chip breaking)
E2	R10	Retract plane (absolute)
Ft	R16	Feedrate
Fr	R17	Retraction feedrate

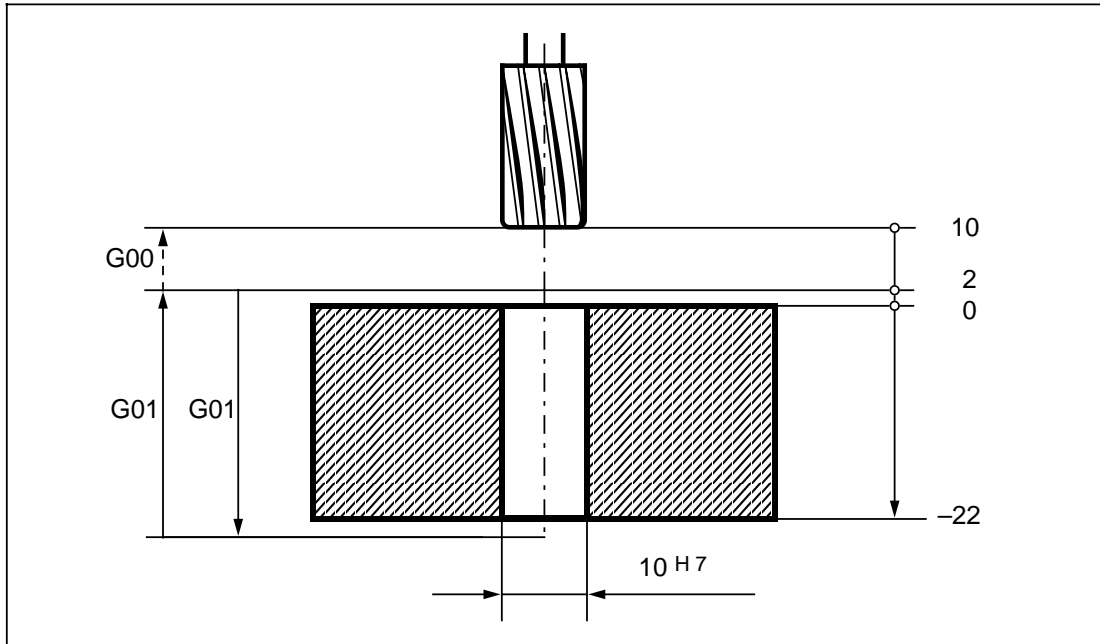


Example: Boring 1

```
%85  
N8501 G90 S150 M03 LF  
N8502 G00 D01 Z50 T03 LF  
N8503 X40 Y40 LF  
N8504 R2=2 R3=-25 R4=0 R10=10 LF  
N8505 R16=60 R17=1000 L85 P1 LF  
N8506 G00 Z50 LF  
N8507 M30 LF
```

Select drilling position

Call drilling cycle

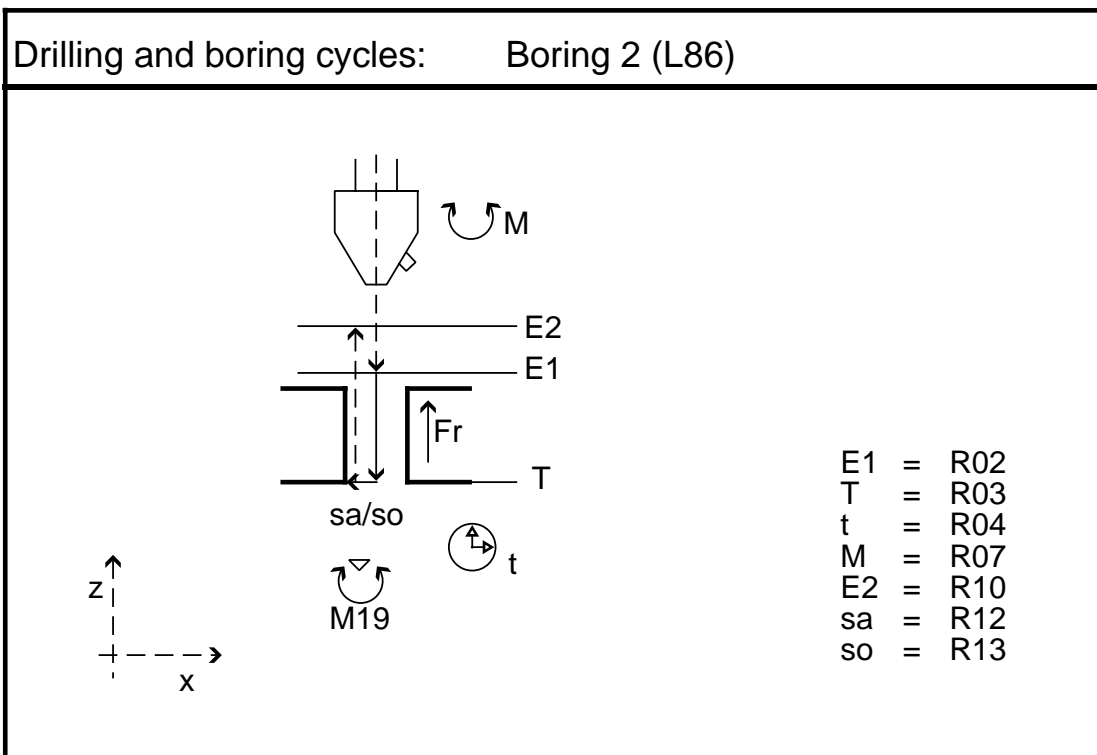


With subroutine L85, the insertion and retraction movements are executed at the feedrate programmed in R parameters R16 and R17.

Subroutine L86: Boring 2

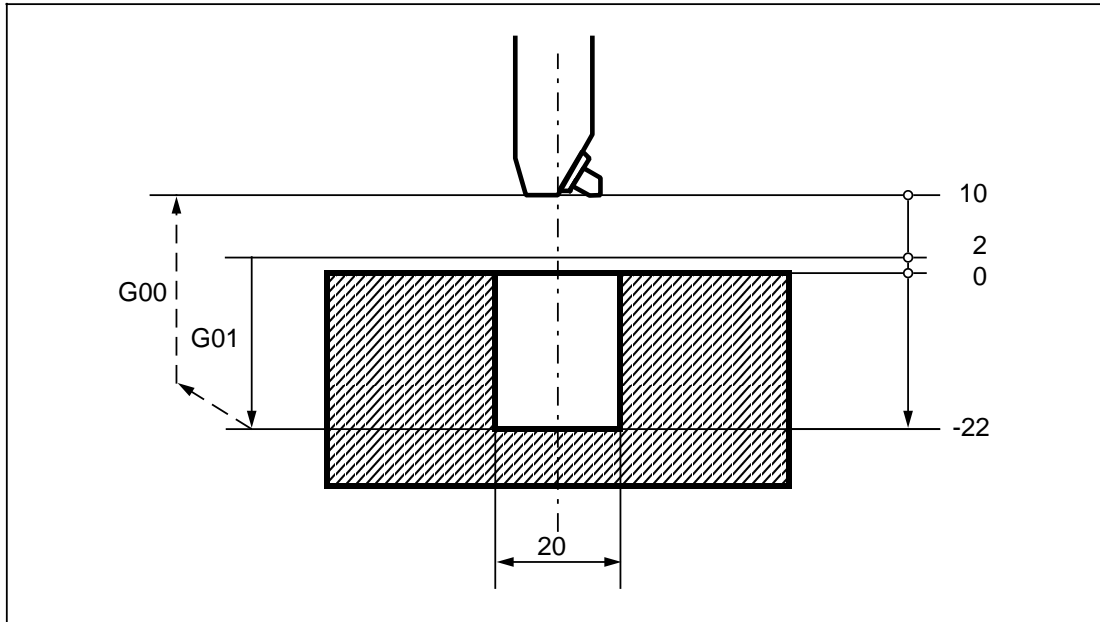
The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
t	R04	Dwell time at drilling depth (chip breaking)
M	R07	Direction of rotation (M03/M04)
E2	R10	Retract plane (absolute)
sa	R12	Retract travel (horizontal with sign) (incremental)
so	R13	Retract travel (vertical with sign) (incremental)



Example: Boring 2

```
%86  
N8601 G90 F100 S500 LF  
N8602 G00 D01 Z50 T03 LF  
N8603 X40 Y40 LF                               Select 1st drilling position  
N8604 R2=2 R3=-22 R4=1 R7=3  
      R10=10 R12=-2 R13=2 L86 P1 LF           Call drilling cycle, 1st hole  
N8605 Z50 LF  
N8606 M30 LF
```



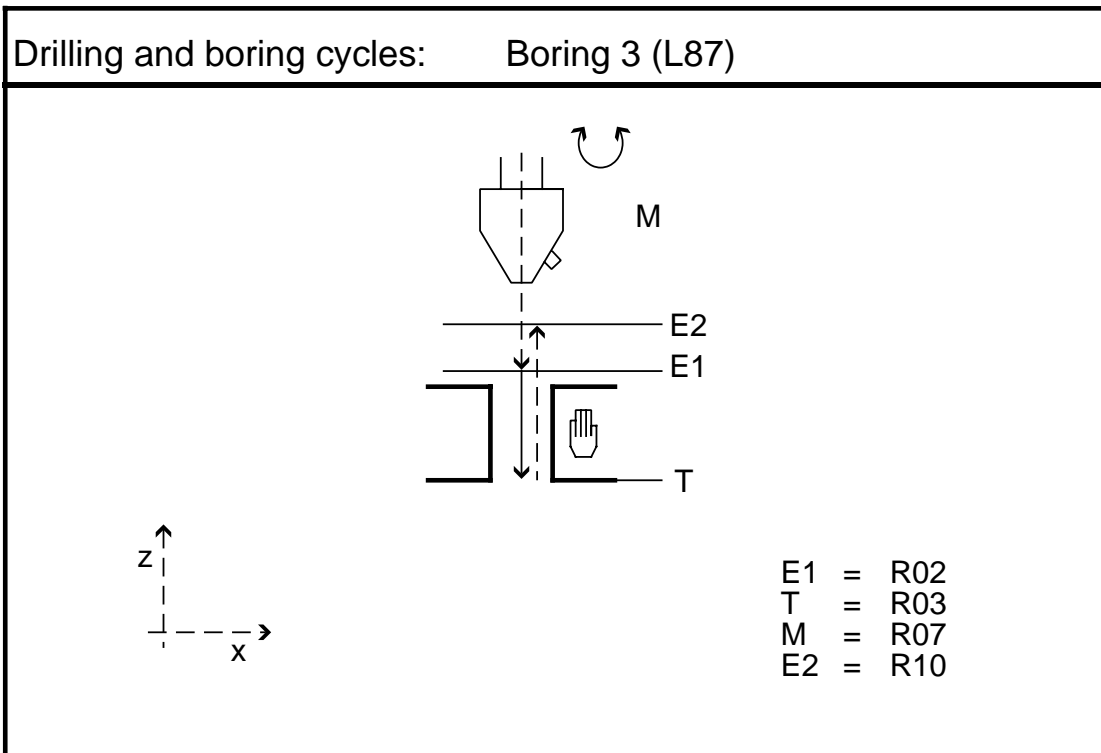
M19 oriented spindle stop is initiated on reaching the final drilling depth. This is followed by a rapid traverse movement to the programmed retract positions R12, R13 as far as the retract plane.

M19 provides a facility for executing an oriented spindle stop. The relevant angle is programmed at the operator panel under "setting data".

Subroutine L87: Boring 3

The following values are entered directly in the part program as parameter assignments:

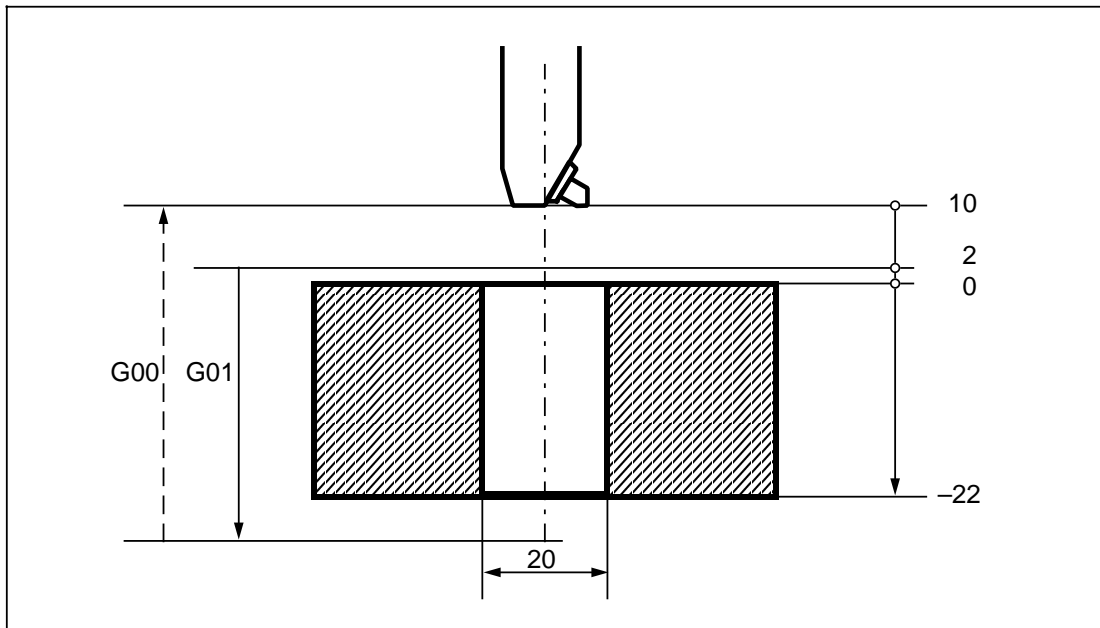
Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
M	R07	Direction of rotation (M03/M04)
E2	R10	Retract plane (absolute)



Example: Boring 3

```
%87  
N8701 G90 F100 S500 LF  
N8702 G00 D01 Z50 T03 LF  
N8703 X40 Y40 LF  
N8704 R2=2 R3=-24 R4=1 R7=3  
      R10=10 L87 P1 LF  
N8705 X80 Y70 LF  
N8706 R2=2 R3=-24 R7=3  
      R10=10 L87 P1 LF  
N8707 Z50 LF  
N8708 M30 LF
```

Select 1st drilling position
Call drilling cycle, 1st hole
Select 2nd drilling position
Call drilling cycle, 2nd hole

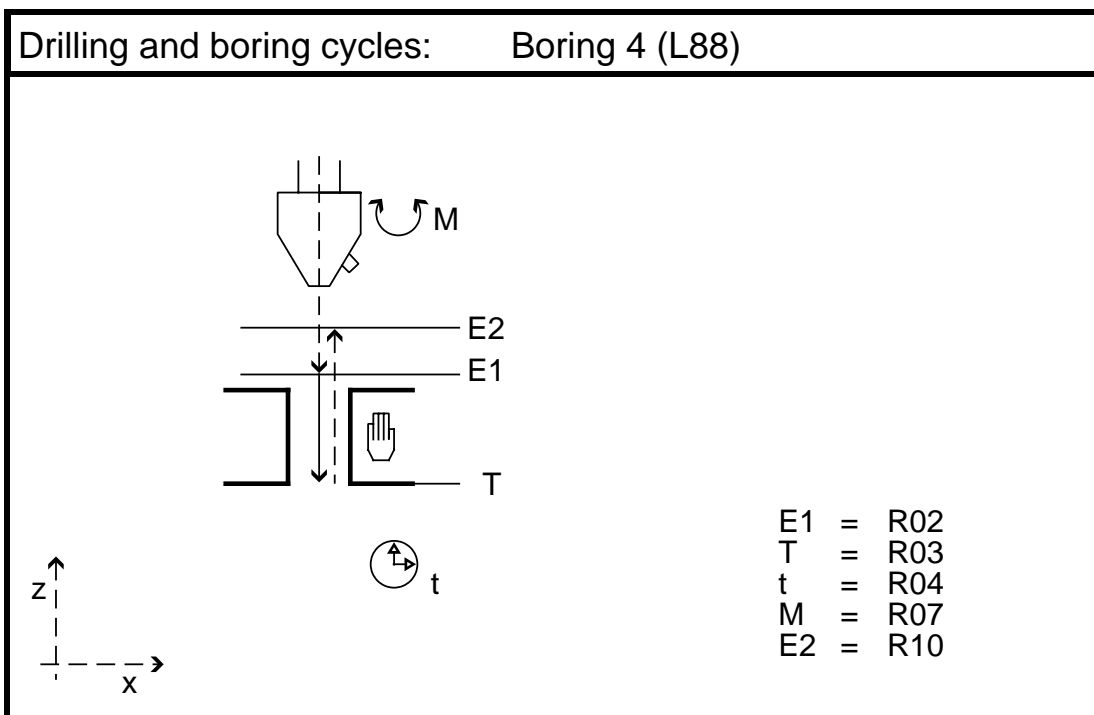


M05 spindle stop without orientation and M00 program stop are initiated on reaching the final drilling depth. Press the NC Start key to execute the withdrawal movement as far as the retract plane at rapid traverse.

Subroutine L88: Boring 4

The following values are entered directly in the part program as parameter assignments:

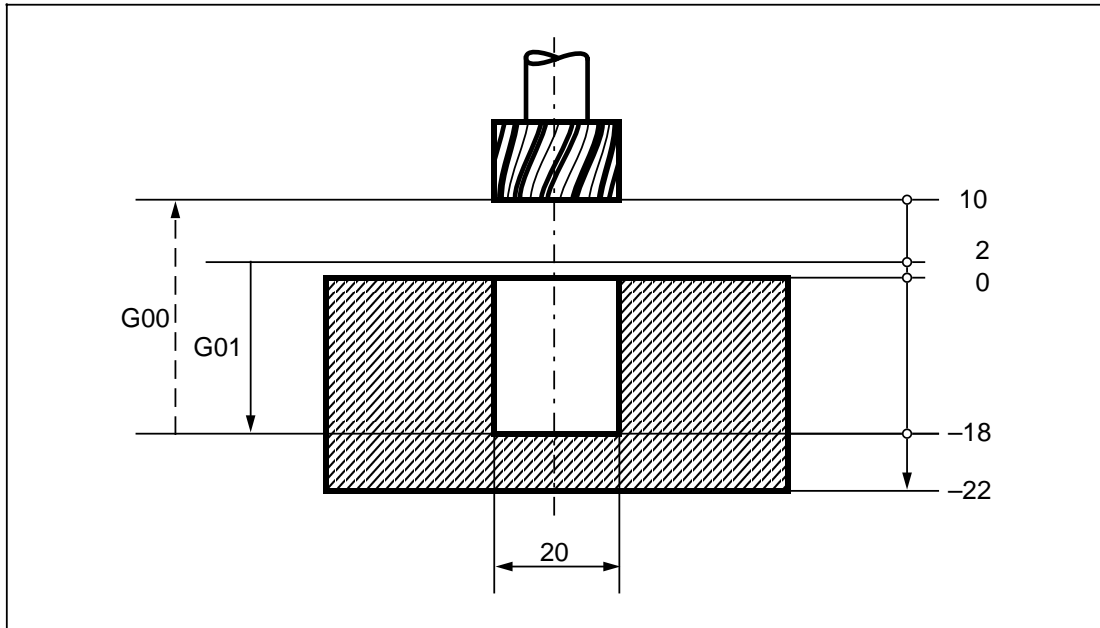
Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
t	R04	Dwell time at drilling depth (chip breaking)
M	R07	Direction of rotation (M03/M04)
E2	R10	Retract plane (absolute)



Example: Boring 4

```
%88  
N8801 G90 F100 S500 LF  
N8802 G00 D01 Z50 T03 LF  
N8803 X40 Y40 LF  
N8804 R2=2 R3=-18 R4=1 R7=3  
      R10=10 L88 P1 LF  
N8805 X80 Y70 LF  
N8806 R2=2 R3=-18 R4=1 R7=3  
      R10=10 L88 P1 LF  
N8807 Z50 LF  
N8808 M30 LF
```

Select 1st drilling position
Call drilling cycle, 1st hole
Select 2nd drilling position
Call drilling cycle, 2nd hole

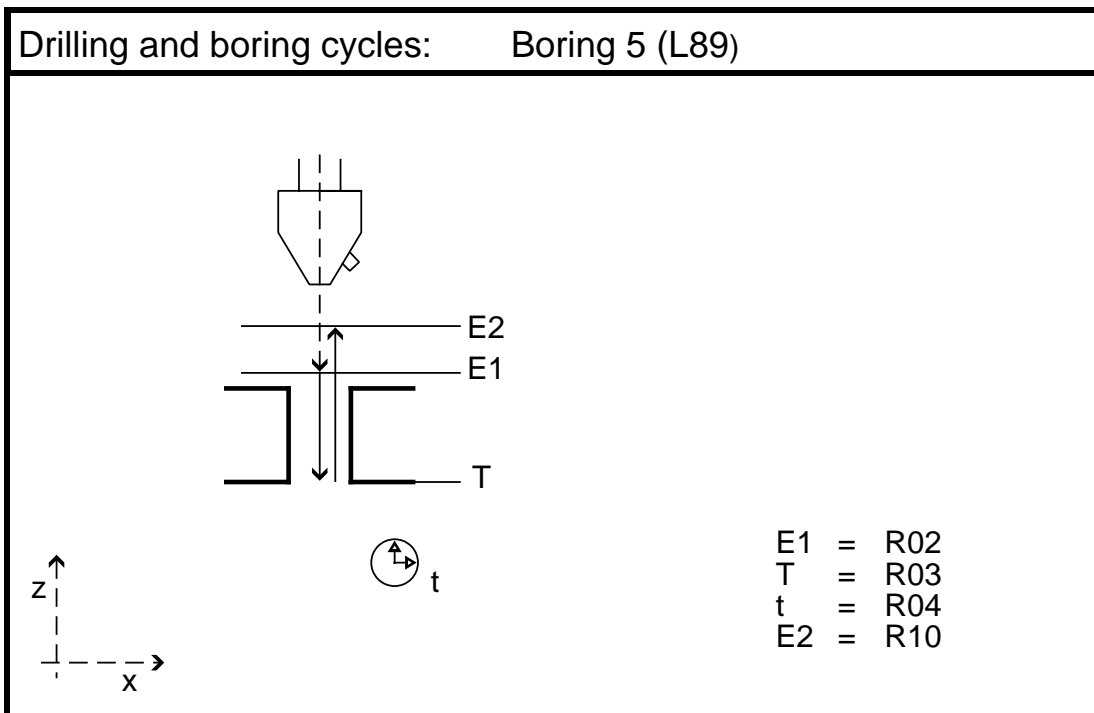


M05 spindle stop without orientation and M00 program stop are initiated on reaching the final drilling depth. Press the NC Start key to execute the withdrawal movement as far as the retract plane at rapid traverse. A dwell time can be programmed at the final drilling depth.

Subroutine L89: Boring 5

The following values are entered directly in the part program as parameter assignments:

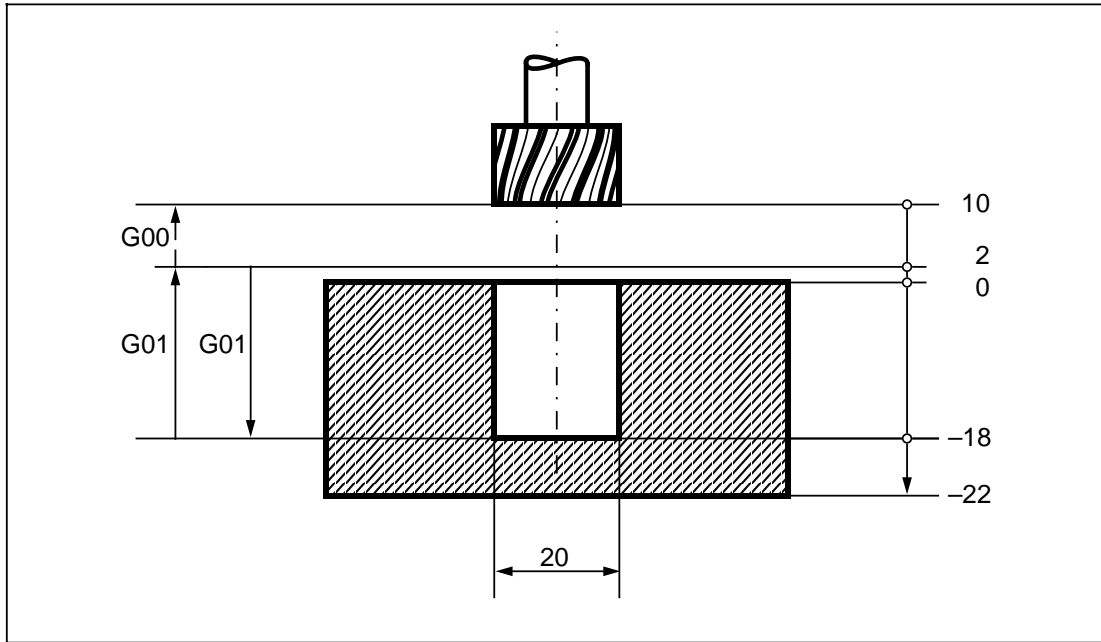
Symbol	Parameter	Description
E1	R02	Reference plane (absolute)
T	R03	Final drilling depth (absolute)
t	R04	Dwell time at drilling depth (chip breaking)
E2	R10	Retract plane (absolute)



Example: Boring 5

```
%89  
N8901 G90 F60 S500 M03 LF  
N8902 G00 D01 Z50 T03 LF  
N8903 X40 Y40 LF  
N8904 R2=2 R3=-18 R4=1 R10=10 L89 P1 LF  
N8905 X80 Y70 LF  
N8906 R2=2 R3=-18 R4=1 R10=10 L89 P1 LF  
N8907 Z50 LF  
N8908 M30 LF
```

Select 1st drilling position
Call drilling cycle, 1st hole
Select 2nd drilling position
Call drilling cycle, 2nd hole



With subroutine L89, the insertion and withdrawal movements are executed at the same feedrate. A dwell time can be programmed at the final drilling depth.

10.3 Machining cycles for turning

10.3.1 L93 Grooving cycle (precondition: blueprint-programming)

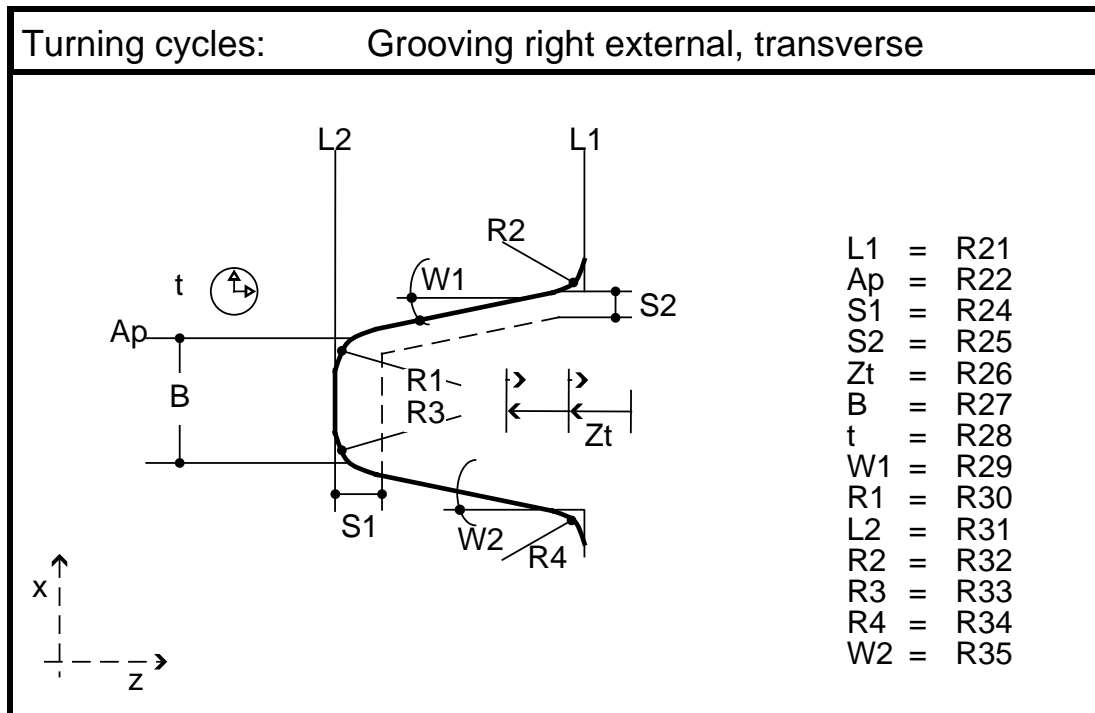
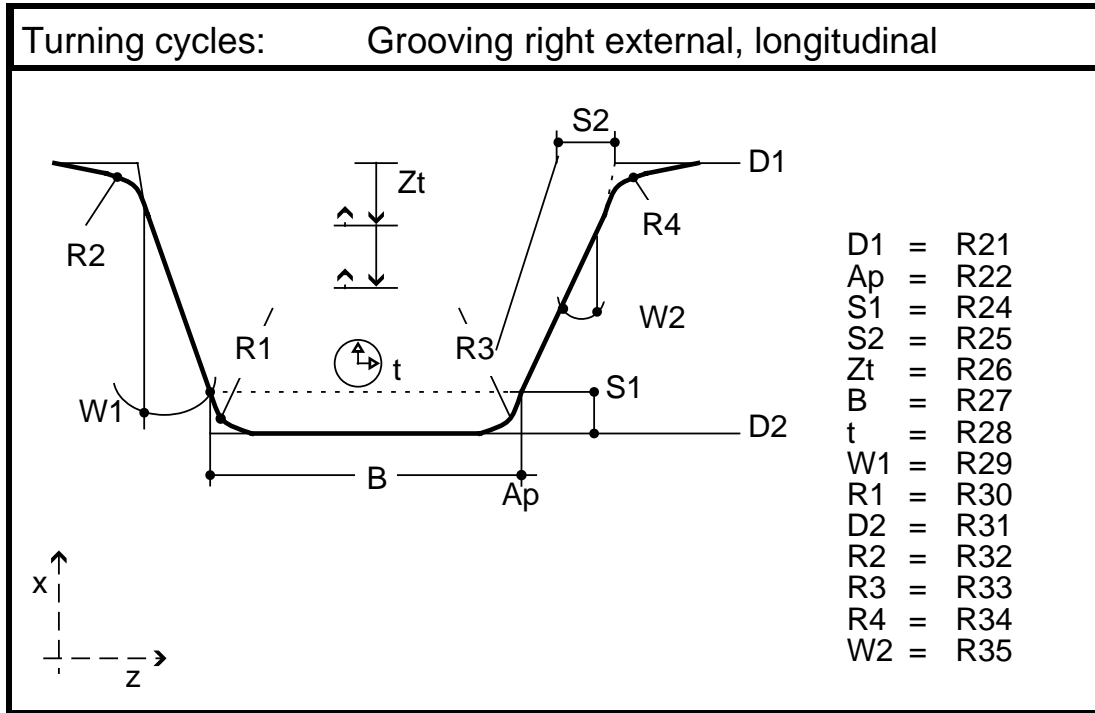
Grooving cycle L93 permits machining of symmetrical and asymmetrical external and internal grooves in longitudinal and transverse directions.

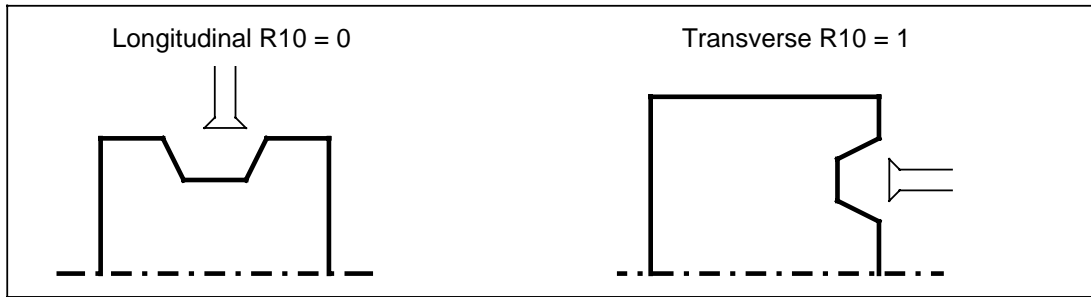
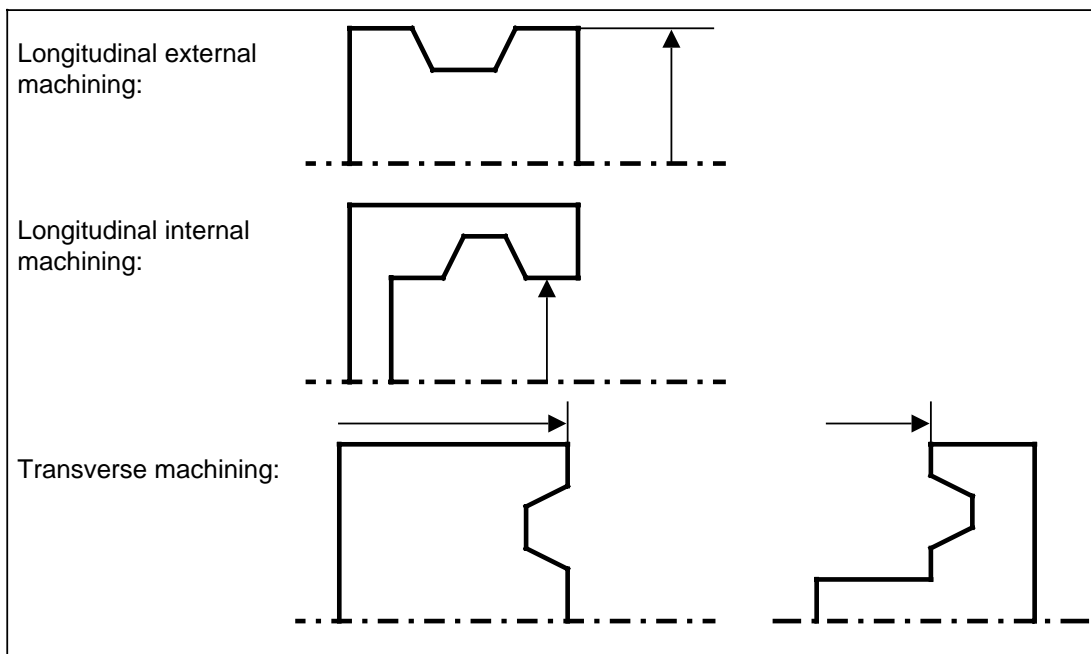
Before the grooving cycle is called in a machining program, the tool offset for one cutting edge of the grooving tool must be selected and the desired compensation value programmed in the case of double-edged (grooving) tools.

The tool offset for the second edge of the grooving tool must then be stored under the next higher offset number in the tool offset memory. If the tool offset for the first edge is $D = n$, the offset for the second edge is identified with offset No. $D = n + 1$.

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
	R10	Machining type: longitudinal R10 = 0, transverse R10 = 1
D1/L1	R21	External/internal diameter or initial length (transverse) (absolute)
Ap	R22	Start point: longitudinal Z, transverse X (absolute)
	R23	Control parameter: start point left or right
S1	R24	Finishing allowance for groove bottom (incremental)
S2	R25	Finishing allowance for edges (incremental)
Zt	R26	Infeed depth (incremental)
B	R27	Groove width (incremental)
t	R28	Dwell time at groove bottom
W1	R29	Angle (0° to 89°)
R1	R30	Radius or chamfer at groove bottom
D2/L2	R31	Groove diameter or length of groove depth (transverse) (absolute)
R2	R32	Radius or chamfer at groove edge
R3	R33	Radius or chamfer at groove bottom
R4	R34	Radius or chamfer at groove edge
W2	R35	Angle (0° to 89°)

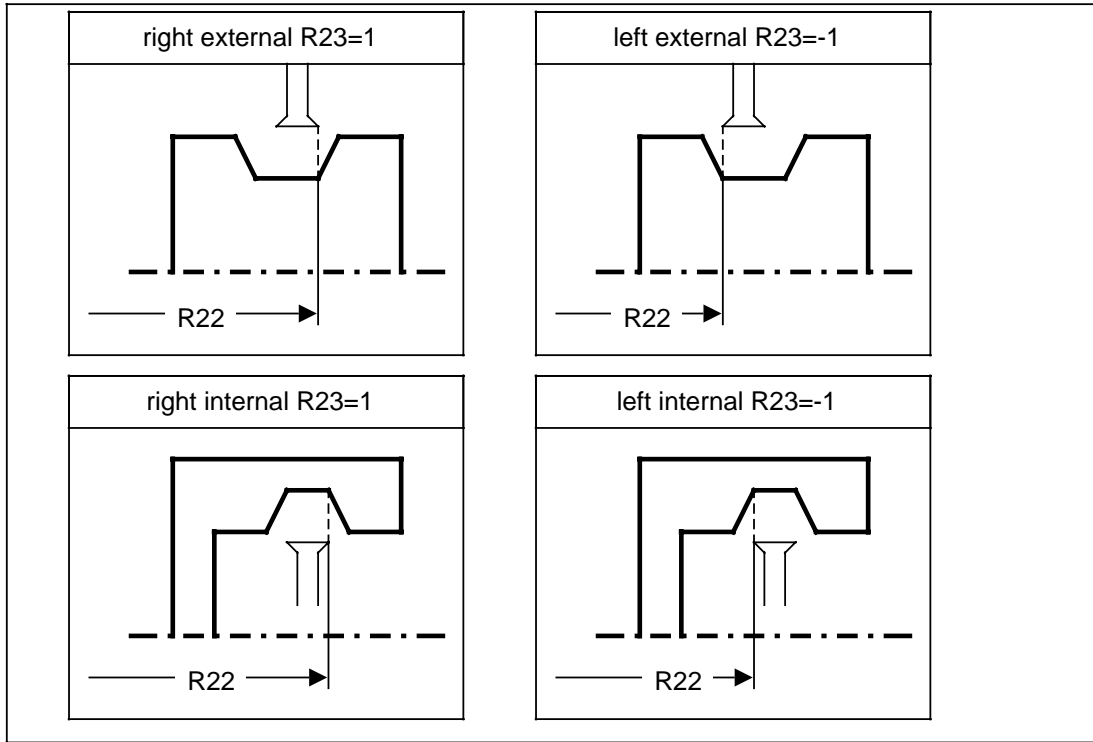


Explanation of parameters:**R10:** Machining type**R21:** External/internal diameter or initial length (transverse)

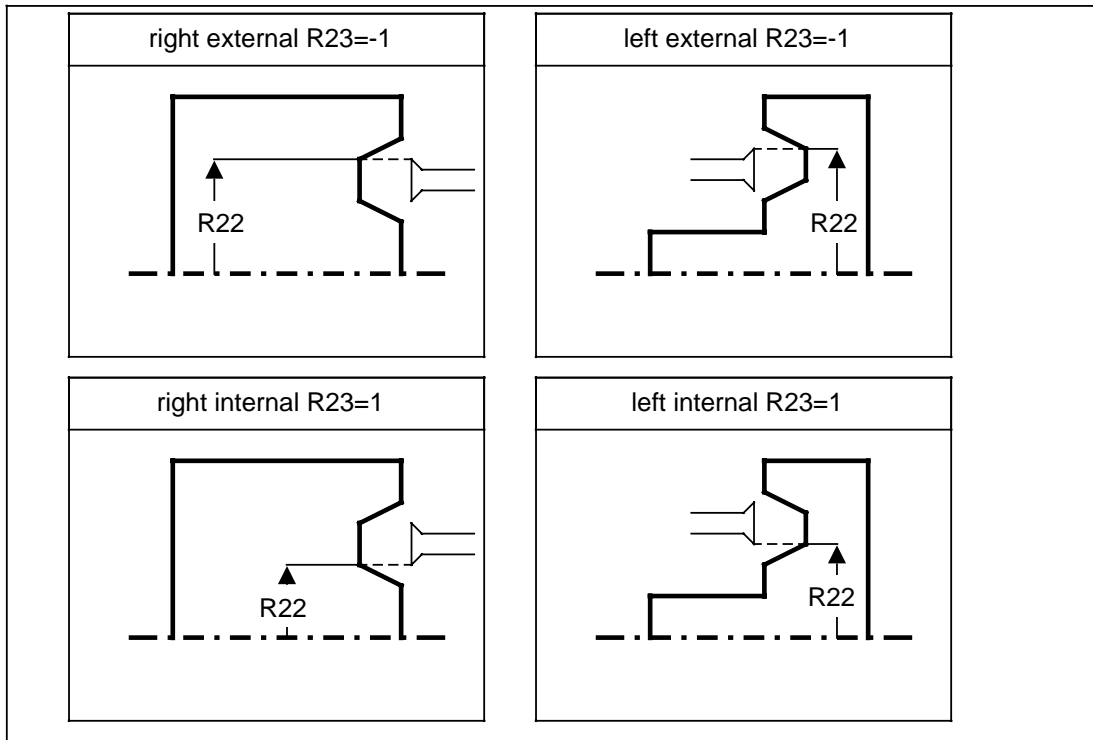
R21 is approached with a safety clearance of 1 mm.

R22: Start point, R23: Control parameter

Longitudinal groove



Transverse groove



S1 R24: Finishing allowance for groove bottom (incremental)**S2 R25: Finishing allowance for edges (incremental)**

The finishing allowances R24/R25 can be input with different values.

Stock removal (roughing) continues until the programmed finishing allowance and then the allowance is removed parallel to contour with the same tool.

If radii or chamfers are to be machined at the groove bottom, a check is made to determine whether these would be damaged by the grooving operation.

With R24/R25 = 0 programmed: no stock removal parallel to contour.

Zt R26: Infeed depth (incremental)

The programmer can use the infeed depth to determine whether the groove depth is to be reached with one or several cuts. If several cuts are required, the tool is retracted by 1 mm for chip breaking after each infeed (cf. step 2).

B R27: Groove width (incremental)

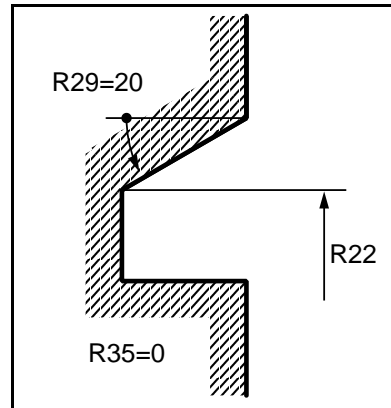
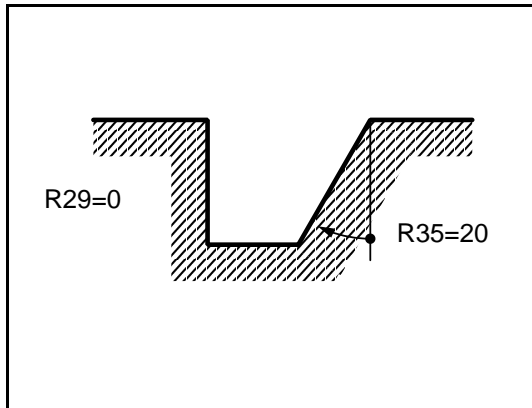
If the groove is wider than the grooving tool, the infeed is divided into equal-size elements. The maximum infeed depends on the tool width. It is 95 % of the tool width after subtraction of the tool nose radii. This ensures overlapping cuts.

t R28: Dwell time at groove bottom

The dwell time must be selected to allow at least one revolution of the spindle.

W1 R29: Angle**W2 R35: Angle**

The edge angle can be between 0° and 89°. The angle must be entered from the vertical with longitudinal grooves and from the horizontal with transverse grooves.

**D2/L2 R31: Groove diameter or length of groove depth (transverse) (absolute)**

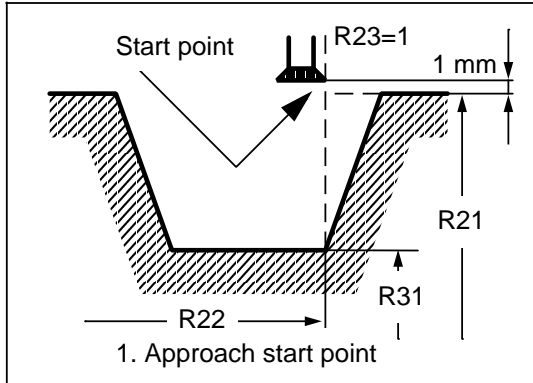
R31 determines the depth of the groove.

R1 R30: Radius or chamfer at groove bottom**R2 R32: Radius or chamfer at groove edge****R3 R33: Radius or chamfer at groove bottom****R4 R34: Radius or chamfer at groove edge**

Parameter R30, R32, R33, R34 can be used to insert radii or chamfers at the groove bottom and/or edge.

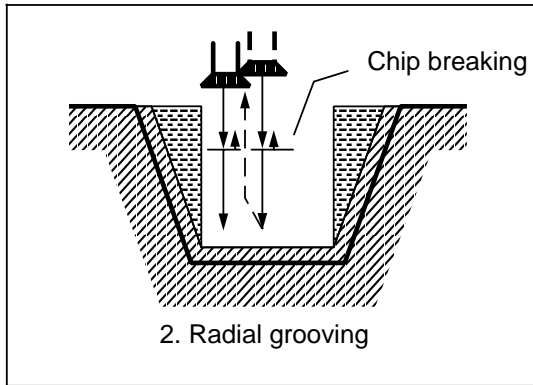
Sign + = radius
 - = chamfer

Machining sequence:



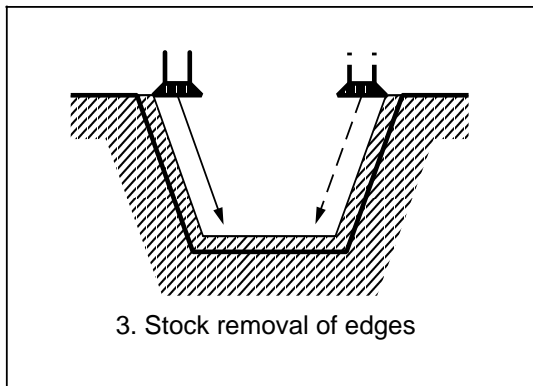
Step 1:

Automatic approach to programmed start point at rapid traverse.



Step 2:

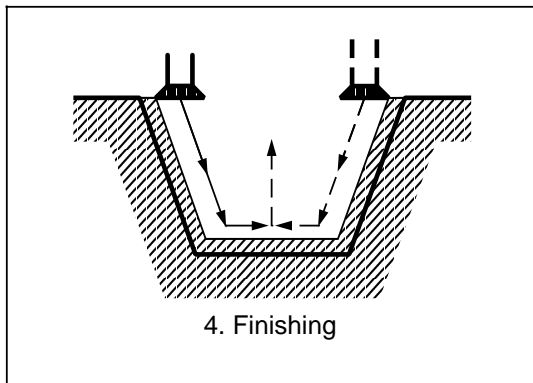
Machine groove perpendicular to the turning axis in one or several cuts. Before withdrawal from the groove, retract by 1 mm from the second step in the Z direction onwards.



Step 3:

Stock removal of edges in one cut if an angle is programmed with R29 or R35.

The infeed in the Z direction is executed in several steps if the tool is not as wide as the edge.



Step 4:

Stock removal of finishing allowance parallel to contour as far as the groove centre.

Example 1: Longitudinal groove "EXTERNAL LEFT"

```
%1
```

```
N05 G95 G0 X65 Z105 D03 T03 S500 M04 LF
```

Select grooving position

```
N10 G01 F0.2 LF
```

```
N15 R10=0 R21=60 R22=100 R23=-1 LF
```

```
N20 R24=1 R25=1 R26=5 R27=20 LF
```

```
N25 R28=0 R29=10 R30=-2 R31=40 LF
```

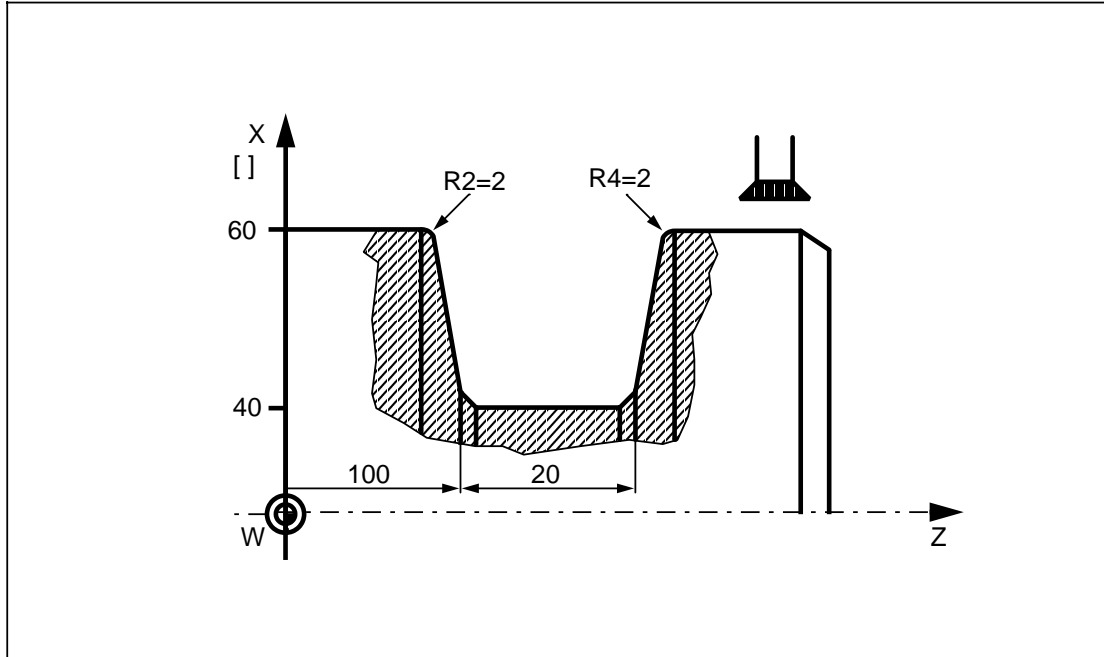
```
N30 R32=2 R33=-2 R34=2 R35=15 LF
```

```
N35 L93 P1 LF
```

Call grooving cycle

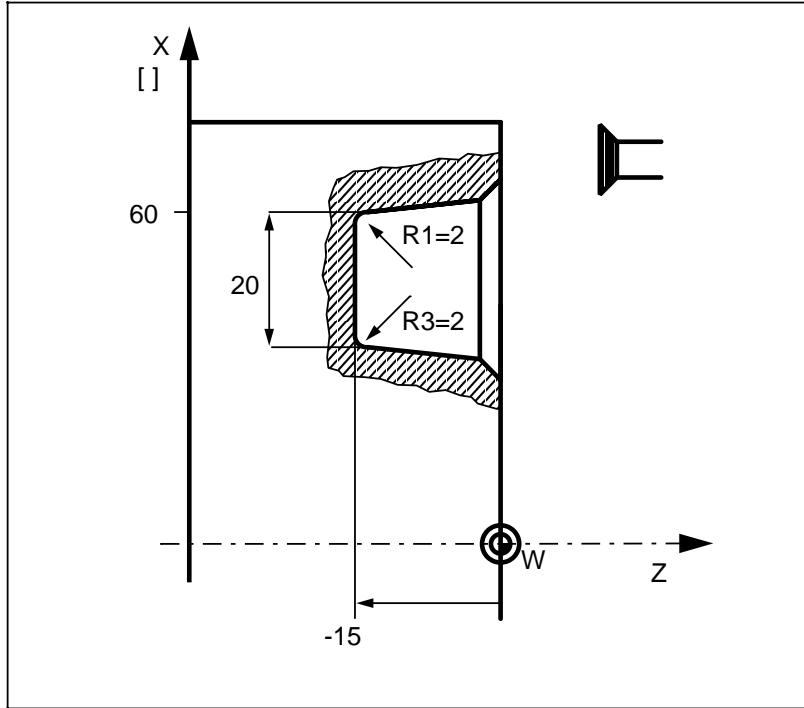
```
N40 G0 X100 Z200 LF
```

```
N45 M30 LF
```



Example 2: Transverse groove "EXTERNAL RIGHT"

```
%2  
N05 G95 G0 X65 Z10 D03 T03 S500 M04 LF           Select grooving position  
N10 G01 F0.2 LF  
N15 R10=1 R21=0 R22=60 R23=-1 LF  
N20 R24=1 R25=1 R26=5 R27=20 LF  
R25 R28=0 R29=10 R30=2 R31=-15 LF  
N30 R32=-2 R33=2 R34=-2 R35=15 LF  
N35 L93 P1 LF                                     Call grooving cycle  
N40 G0 X100 Z200 LF  
N45 M30 LF
```



10.3.2 L96 Stock removal cycle without relief cut elements

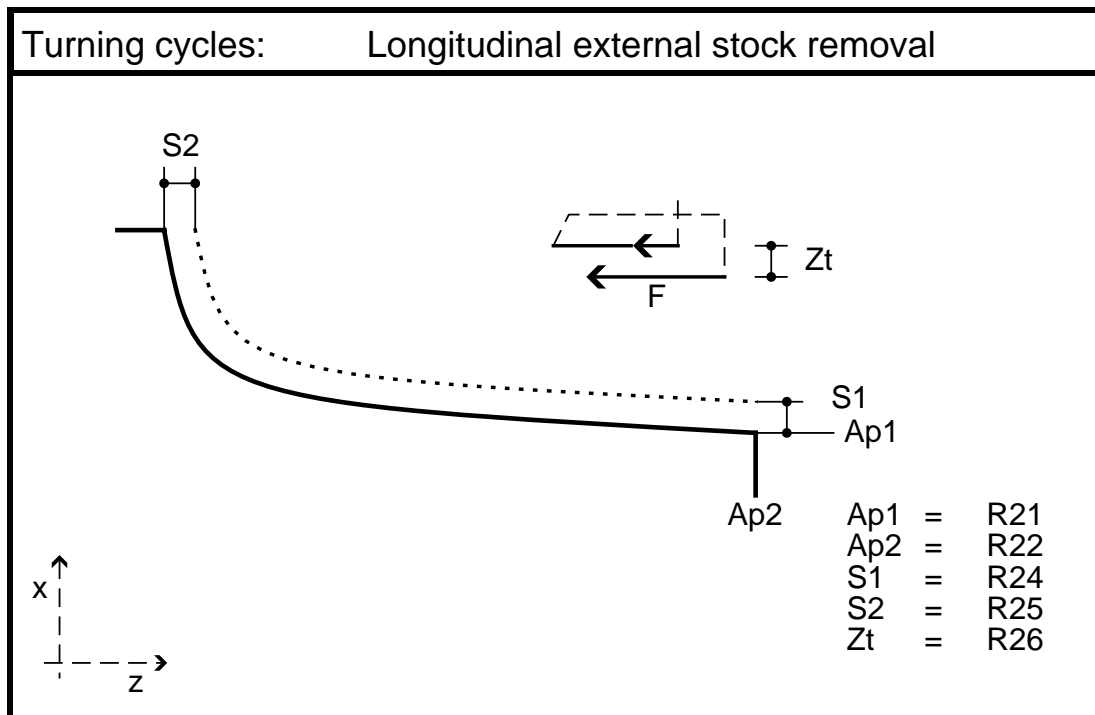
Stock removal cycle L96 permits paraxial cutting of a contour in a blank as programmed in a subroutine. The machining cycle can be called from any collision-free position. The control automatically calculates the start point from the finished contour description.

L96

L96 achieves a fast start of the first traversing movement. With L96 the residual corner stock removal and finishing are executed in the direction of the programmed contour. A feedrate must be entered in the part program before calling L96.

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
	R20	Subroutine number in which the contour is stored
Ap1	R21	Start point of contour in X (absolute)
Ap2	R22	Start point of contour in Z (absolute)
S1	R24	Finishing allowance in X (incremental)
S2	R25	Finishing allowance in Z (incremental)
Zt	R26	Roughing cut depth in X or Z (incremental) (not required for finishing with R29 = 21 to 24)
	R27	Tool nose radius compensation (G41, G42)
	R29	Type determination for roughing and finishing



R20: Subroutine number

The subroutine number under which the finished contour is to be programmed, must be specified in R20.

The subroutine can comprise any number of blocks, but must contain at least two. A traversing path must be programmed in every block. The finished contour can be determined by means of blueprint programming (cf. Programming Guide, section 7.2). Skippable blocks are permitted in the contour.

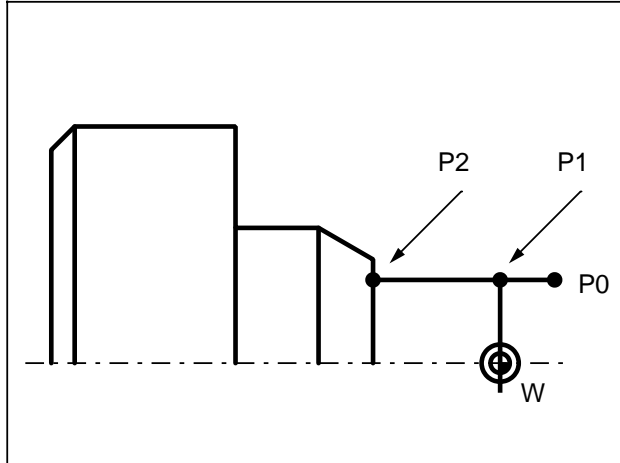
A contour element with the **maximum** diameter must be provided at the end of the contour.

The start point of the contour must not be programmed in the contour subroutine. It is defined with parameters R21 and R22 (contour start points).

Ap1 R21: Start point of contour in X

Ap2 R22: Start point of contour in Z

Parameters R21 and R22 must be initialized with the start points of the contour. With roughing, these points are automatically approached by the finishing allowance R24, R25 plus 1 mm safety clearance. If this represents insufficient clearance, the contour start points R21, R22 must be displaced accordingly.



Example:

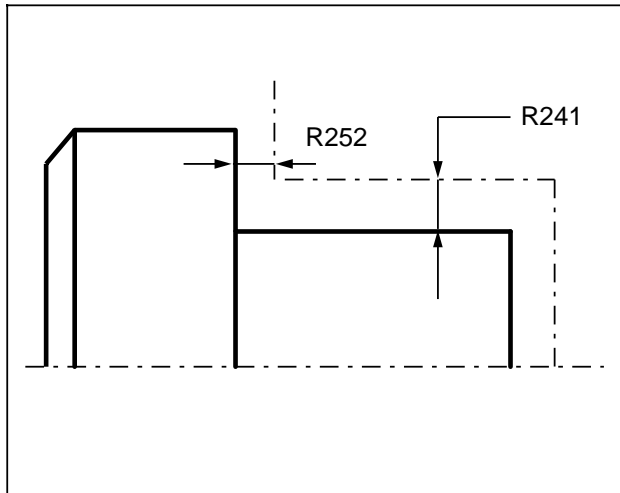
Point P1 corresponds to start points R21 and R22, e.g. R21 = 20, R22 = 0. P0 displaced start point.

Point P2 must be programmed as the first point in the part program, e.g.

```
L102
N05 X20 Z-15 LF
N10 X25
N15 ...
N20 ...
```

S1 R24: Finishing allowance in X (incremental)

S2 R25: Finishing allowance in Z (incremental)



The input finishing allowance R24 and R25 displaces the contour. With the "roughing" machining type, roughing is executed as far as this allowance.

With the finishing machining type, a single stock removal operation is performed parallel to contour as far as the finishing allowance. Roughing cut depth R26 does not have to be programmed; this depends on the machining allowance (e.g. castings).

If R24/25 is initialized with 0, the tool traverses directly along the finished contour dimensions.

Zt R26: Roughing cut depth in X or Z (incremental)

During roughing, the cycle checks whether the current cut depth is less than twice the roughing cut depth R26. When the current cut depth understeps twice the roughing cut depth, the following applies to the final two cuts:

R26 roughing cut depth = current cut depth/2.

R27: Tool nose radius compensation (G41, G42)

The cycle automatically selects and cancels the tool nose radius compensation when selected by R27 = 41 or 42 (G41/G42). (See R29 for further information.)

R29: Type determination for roughing and finishing

When initializing the parameters, the machining type (R29) is defined according to the table.

The selected machining type (R29) indicates the type of stock removal operation: roughing or finishing, external or internal machining, type of cut segmentation, longitudinal or transverse.

		longitud. external	longitud. internal	transverse external	transverse internal
 1. Paraxial roughing	 Removed stock	R29=11	R29=13	R29=12	R29=14
 Finishing parallel to contour		R29=21 *)	R29=23 *)	R29=22 *)	R29=24 *)
 1. Paraxial roughing	 2. Roughing cut parallel to contour	R29=31 *)	R29=33 *)	R29=32 *)	R29=34 *)
 1. Paraxial roughing	 2. Roughing cut parallel to contour	 3. Cut parallel to contour	R29=41 *)	R29=43 *)	R29=42 *)
			R29=44 *)		

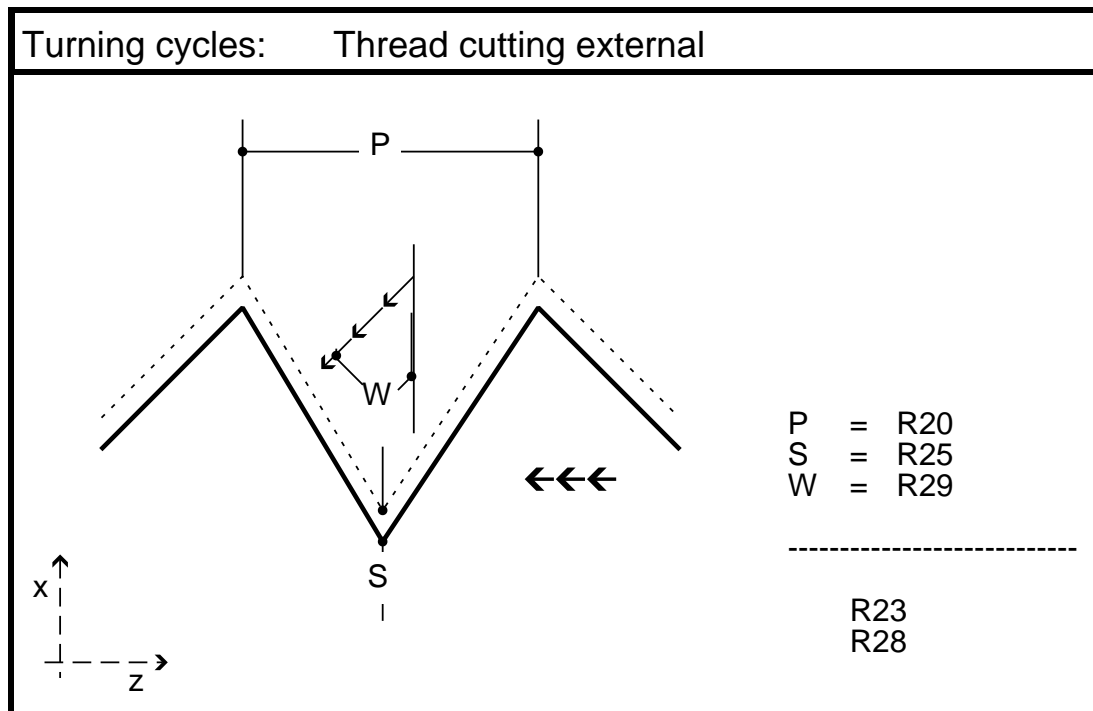
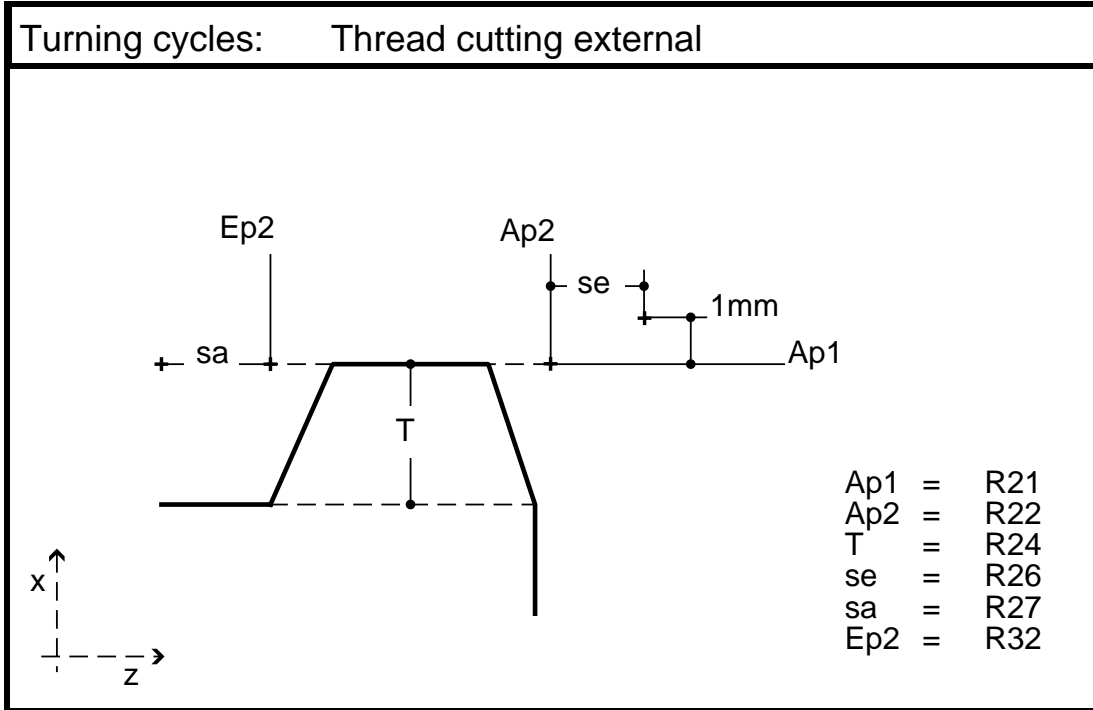
*) In these cases, the cycle automatically activates the tool nose radius compensation (TNRC) in the correct direction if preceded by selection with R27 = (G41 or G42). The cycle also automatically controls the correctly timed selection and cancellation of the TNRC. The TNRC is suppressed internally in the case of paraxial roughing. It is cancelled at the end of the cycle and must be reprogrammed where appropriate.

10.3.3 L97 Thread cutting cycle

This cycle can be used to cut external, internal, taper and transversal threads. Infeed is automatic and squared degressive; the cut cross section thus remains constant.

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
Ap1	R21	Start point of thread in X (absolute)
Ap2	R22	Start point of thread in Z (absolute)
Ep1	R31	End point of thread in X (absolute)
Ep2	R32	End point of thread in Z (absolute)
se	R26	Run-in path (incremental)
sa	R27	Run-out path (incremental)
T	R24	Thread depth (incremental) Enter with sign depending on internal or external thread: += internal thread, - = external thread, transversal thread
P	R20	Lead
S	R25	Finishing allowance (incremental)
W	R29	Infeed angle
	R23	Number of noncuts
	R28	Number of roughing cuts



Thread cutting cycles: differentiation between transversal and longitudinal threads

Both longitudinal and transversal threads can be cut with L97. The differentiation depends on the angle resulting from P1, the start point of the thread, and P2, the first intermediate point/end point of the thread. If the angle is $> 45^\circ$, a transversal thread is machined (cf. example).

- Longitudinal thread
The value $|Z1|$ must be the value $|X1|$, i.e. the angle is 45° .

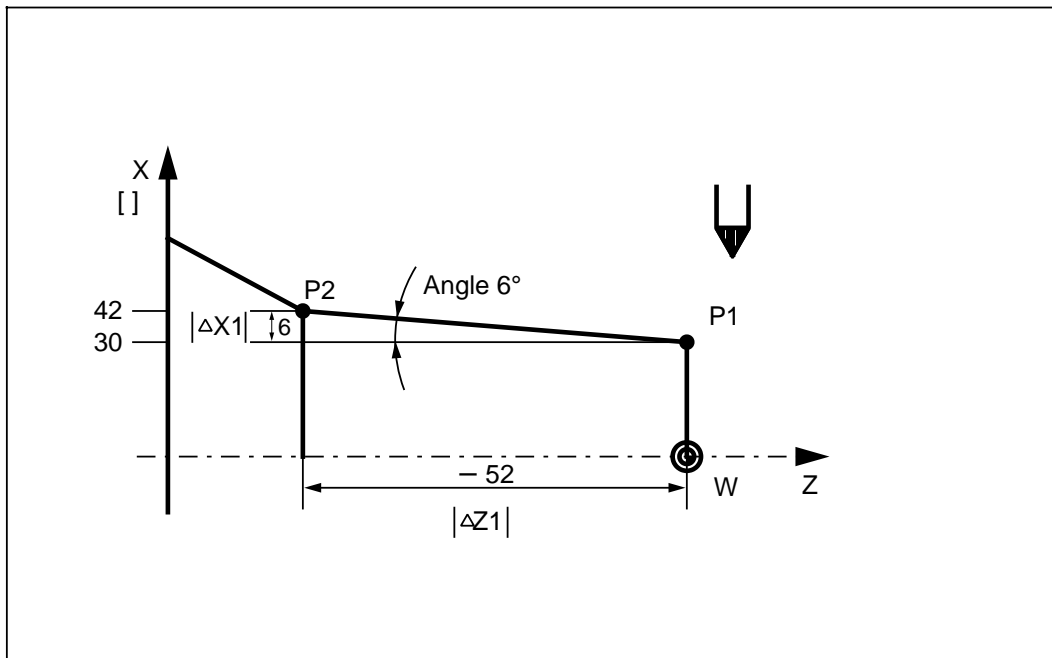
L97

Ap1 R21=30 Start point in X

Ap2 R22=0 Start point in Z

Ep1 R31=42 End point of thread in X

Ep2 R32=-52 End point of thread in Z



- Transversal thread

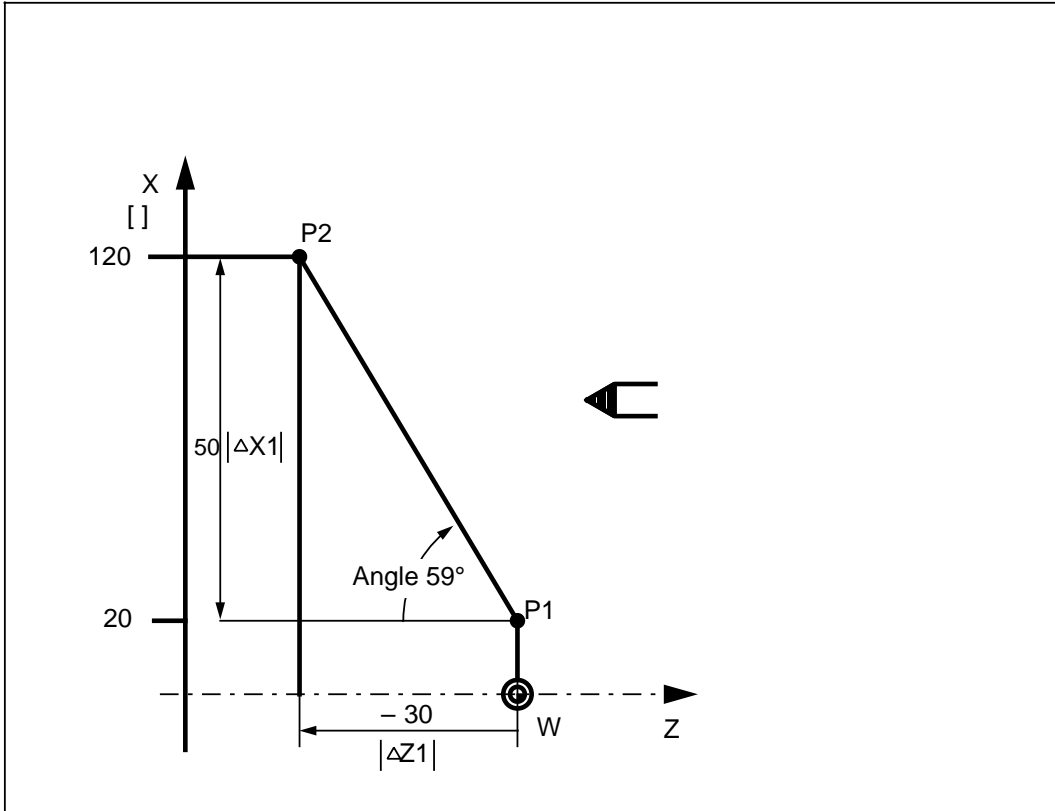
The value $|Z1|$ must be $<$ the value $|X1|$, i.e. the angle is $> 45^\circ$.

L99

Ap1 R11= 20 Start point in X
 Zp1 R12= 120 First intermediate point in X
 Ap2 R21= 0 Start point in Z
 Zp3 R22= -30 First intermediate point in Z

L97

Ap1 R21=20 Start point in X
 Ap2 R22=0 Start point in Z
 Ep1 R31=120 End point of thread in X
 Ep2 R32=-30 End point of thread in Z



P R20: Lead

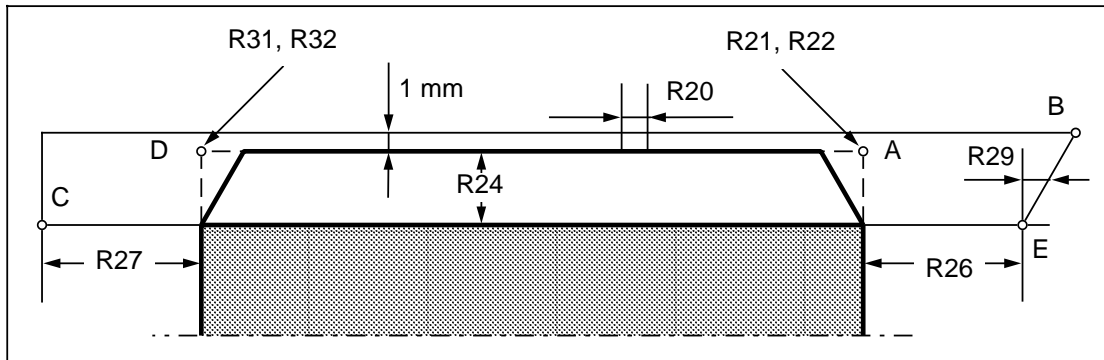
The lead must be input as a paraxial value without sign.

Ap1 R21: Start point of thread in X (absolute)**Ap2 R22: Start point of thread in Z (absolute)**

The parameters R21 and R22 represent the original start points of the thread (A). The start point of the thread cycle is at point B, located $R26 =$ run-in path in front of the thread start point.

In the case of the longitudinal thread, the start point B is 1 mm above value R22; with the transversal thread, the start point is located 1 mm in front of value R22. This retract plane is automatically generated by the control.

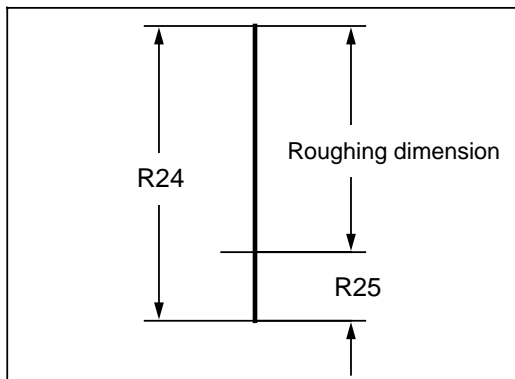
The thread cutting cycle can be called from any slide position; start point B is approached at rapid traverse.

**R23: Number of noncuts**

The number of noncuts is freely selectable (noncut = pass after finishing without infeed).

T R24: Thread depth (incremental)

The thread depth is entered as a paraxial value under parameter R24. The sign determines the infeed direction, that is to say it specifies whether the thread is external, internal or transversal: + = internal thread, - = external thread, transversal thread.

S R25: Finishing allowance (incremental)

If a finishing allowance is programmed under R25, this allowance is subtracted from thread depth R24 and the difference subdivided into roughing cuts.

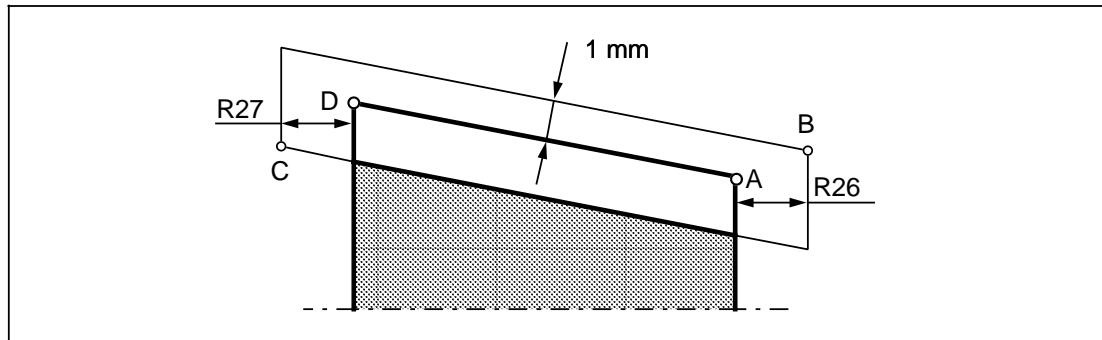
On completion of the roughing cuts, first a finishing cut and then the noncuts programmed under R23 are executed.

The roughing dimension is automatically calculated and broken down into roughing cuts.

se R26: Run-in path (incremental)
sa R27: Run-out path (incremental)

The run-in and run-out paths are entered as incremental, paraxial values without sign.

In the case of taper threads, the control converts the run-in and run-out paths in the taper ratio and determines the corner points B and C.



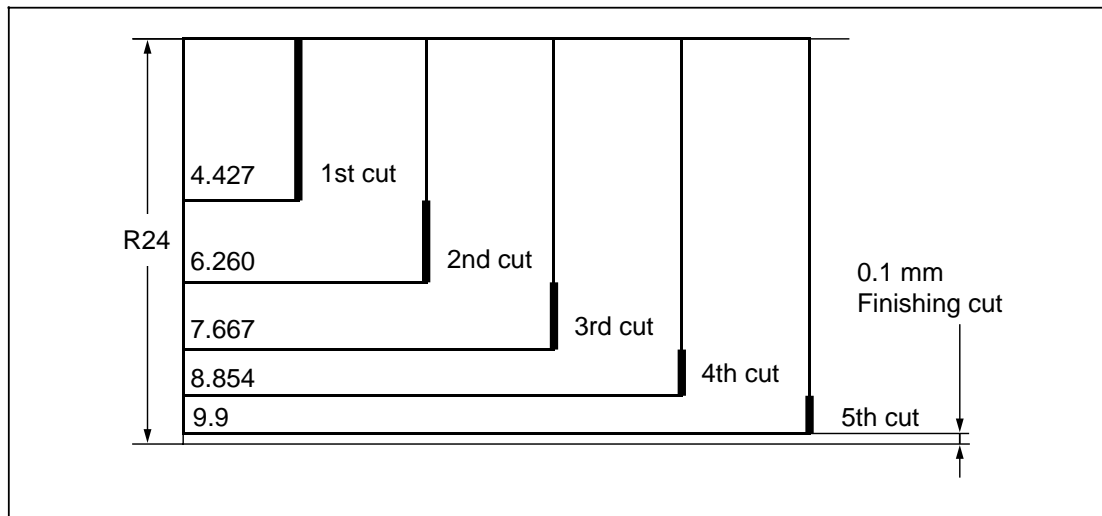
R28: Number of roughing cuts

The parameter value determines the number of thread roughing cuts. The control automatically calculates the individual infeed depths with the same cut cross-section. This ensures that the cutting pressure also remains constant from the first to the last roughing cut.

The current cut depth t is calculated according to the following formula:

$$t = \frac{t}{\sqrt{R28}} \cdot \sqrt{i} \quad \begin{array}{l} t = R24 - R25 \\ i = \text{Current cut} \end{array}$$

Example:

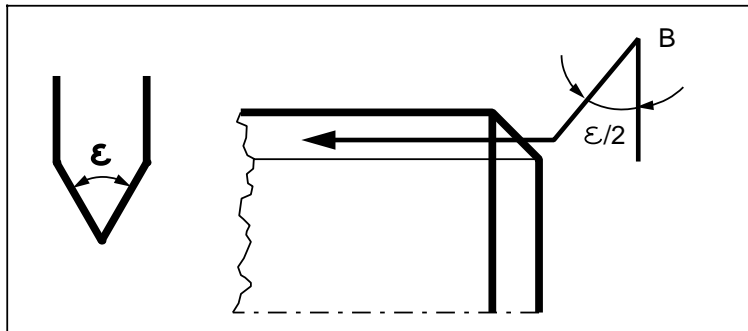


Thread depth: R24 = 10 mm
 Number of roughing cuts: R28 = 5
 Finishing allowance: R25 = 0.1 mm

W R29: Infeed angle with longitudinal or transversal thread

The tool can be infeed at a right angle to the cutting direction or along the flank. The angle is input without sign and must not exceed one-half of the flank angle.

If the tool is to be infeed at a right angle to the axis, R29 must be initialized with 0.



Metric thread 60°
/2=30°
R29=30

Ep1 R31: End point of thread in X (absolute)**Ep2 R32: End point of thread in Z (absolute)**

The parameters R31 and R32 represent the original end points of the thread (D). The reversing point of the thread cutting cycle is at point C which is located R27 = run-out path behind the thread end point.

Example: Machining type thread cutting "EXTERNAL"

%97

N05 G95 G0 X50 Z10 D01 T01 S1000 M04 LF

Select thread cutting position

N10 R20=2 R21=42 R22=0 R23=0 LF

R24=-1,23 R25=0 R26=10 R27=3 LF

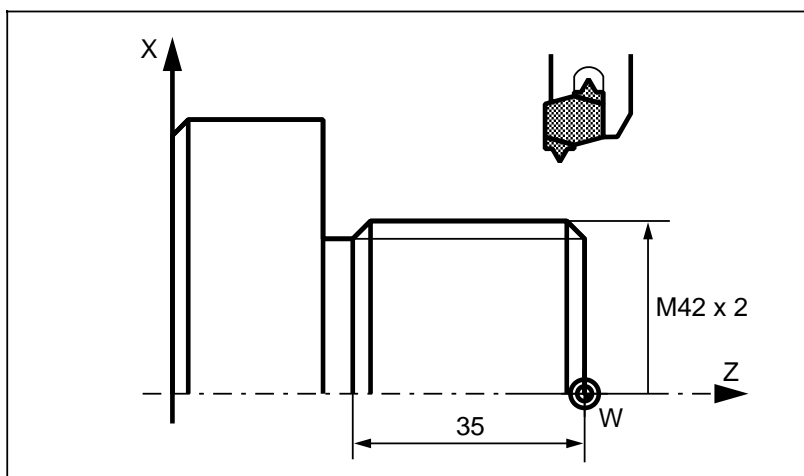
R28=5 R29=30 R31=42 R32=-35 LF

L97 P1 LF

Call thread cutting cycle

N15 G0 X200 Z200 LF

N20 M30 LF

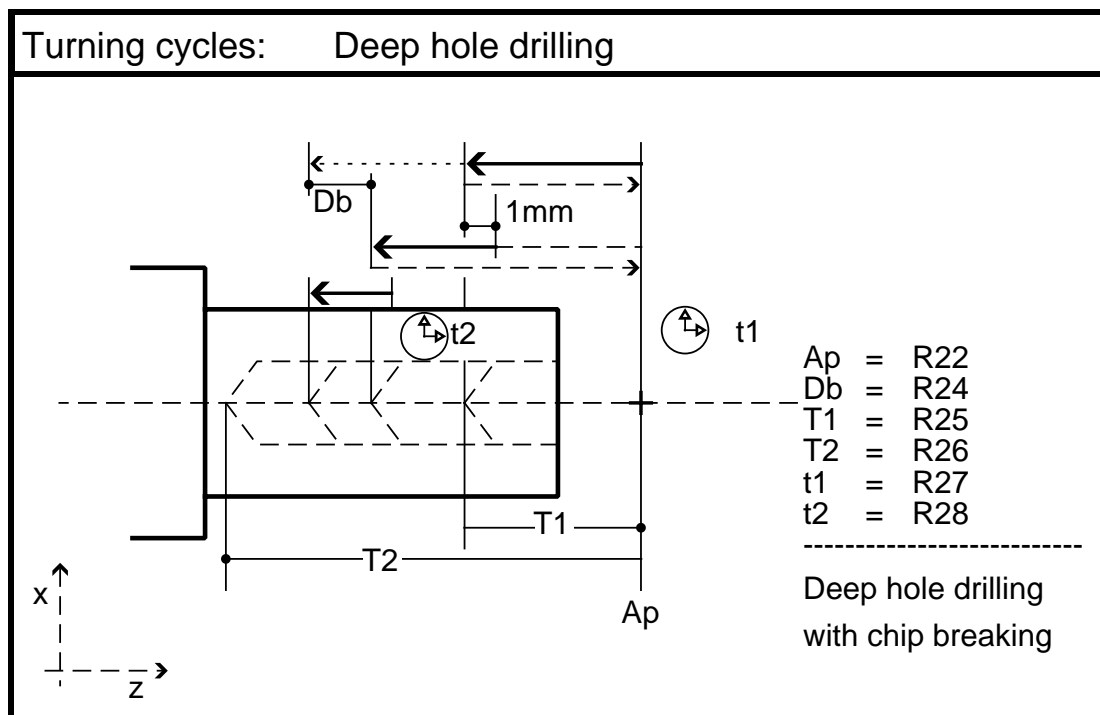


10.3.4 L98 Deep hole boring cycle

This cycle is used to machine deep holes. On reaching each infeed depth the drill can be retracted to the start point for swarf removal.

The following values are entered directly in the part program as parameter assignments:

Symbol	Parameter	Description
Ap	R22	Start point in Z (absolute)
T2	R26	Final drilling depth (absolute)
T1	R25	First drilling depth; input without sign (incremental)
Db	R24	Amount of degression; input without sign (incremental)
t1	R27	Dwell time at start point (swarf removal)
t2	R28	Dwell time at drilling depth (chip breaking)
	R11	0 = with chip breaking, 1 = with swarf removal



R11: Chip breaking/swarf removal

If R11 is initialized with 0, the drill retracts by 1 mm for chip breaking after each drilling depth is reached.

If R11 is initialized with 1, the drill travels to the reference plane for swarf removal after each drilling depth is reached.

Ap R22: Start point in Z (absolute)

The start point should be selected to provide sufficient space when drilling with swarf removal.
The final drilling depth is calculated from the start point.

Example: Machining type "deep hole drilling"

```
%98
```

```
N05 G95 G1 Z20 D01 T04 F0.1 S500 M03 LF
```

Select drilling position

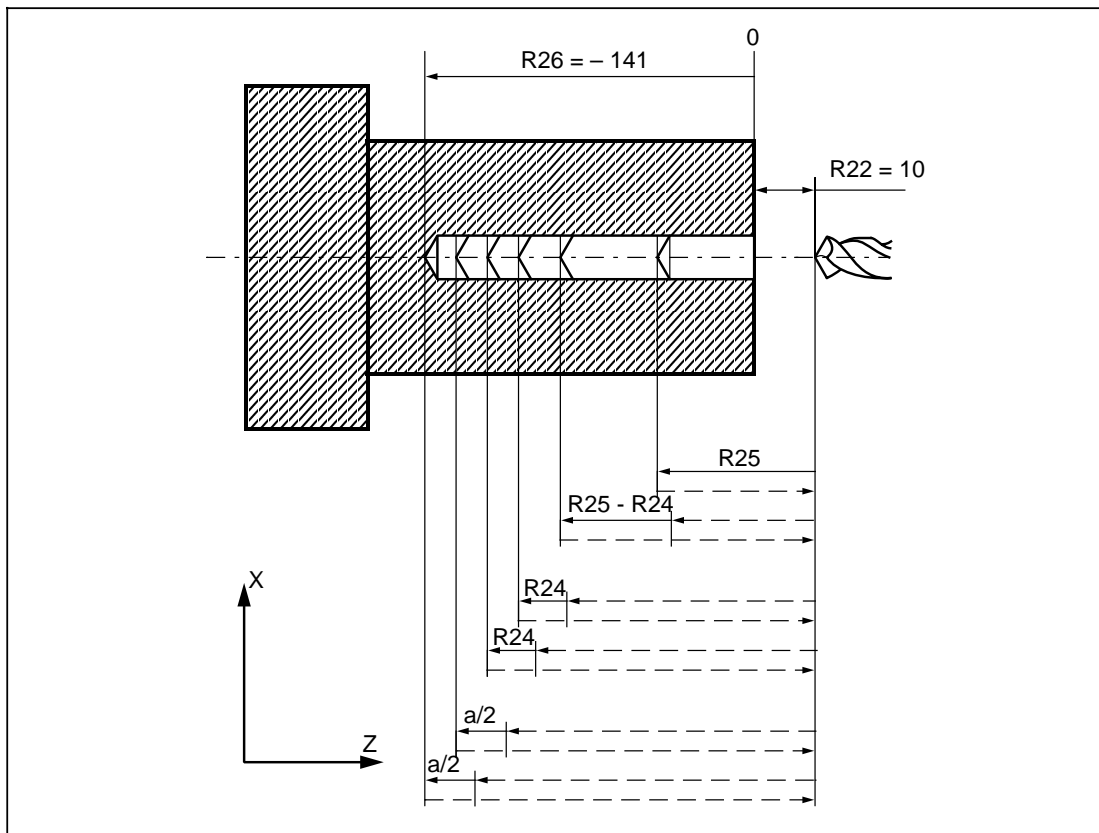
```
N10 R11=1 R22=10 R24=20 R25=50
```

```
R26=-141 R27=2 R28=0 LF
```

```
L98 P1 LF
```

Call deep hole drilling cycle

```
N15 M30 LF
```

**T2 R26: Final drilling depth**

The drilling depth is degressively reduced by the constant amount of depression in each case until end point R26 is reached.

If the drilling depth theoretically understeps the amount of depression, however, it is maintained at this amount.

If the remaining infeed is less than twice the amount of depression, the remaining amount is halved. The two final infeeds are then executed with this half amount. This prevents a final infeed with an amount that is too small. This calculation ensures a minimum infeed of one-half of the amount of depression.

10.4 Milling patterns

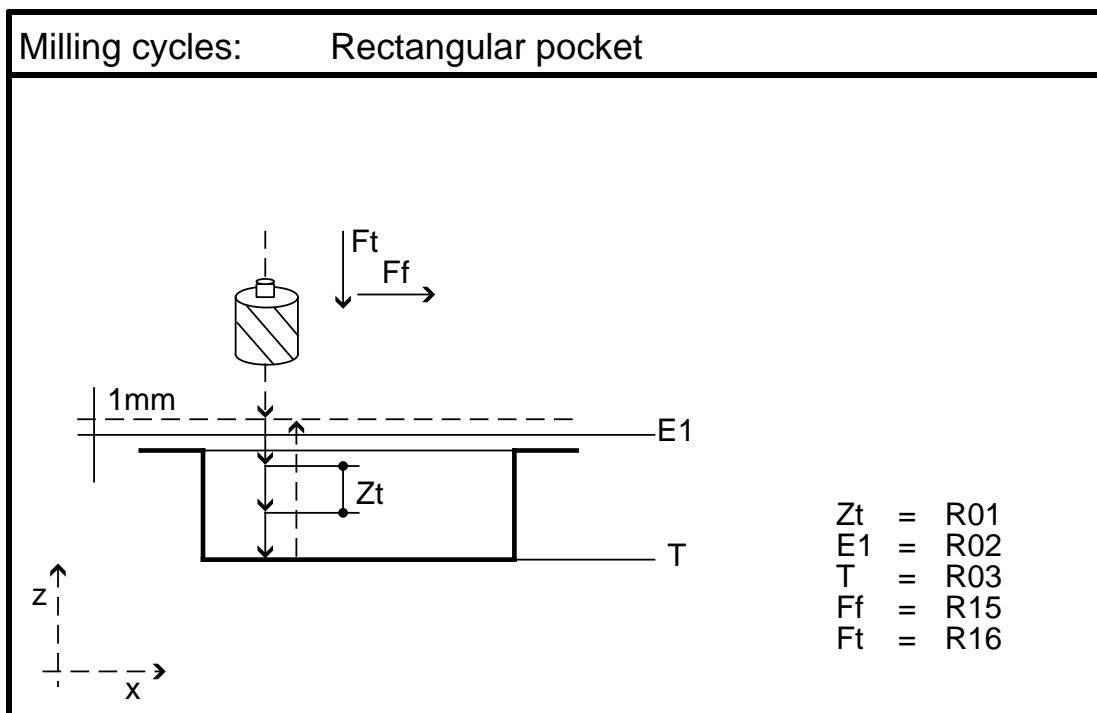
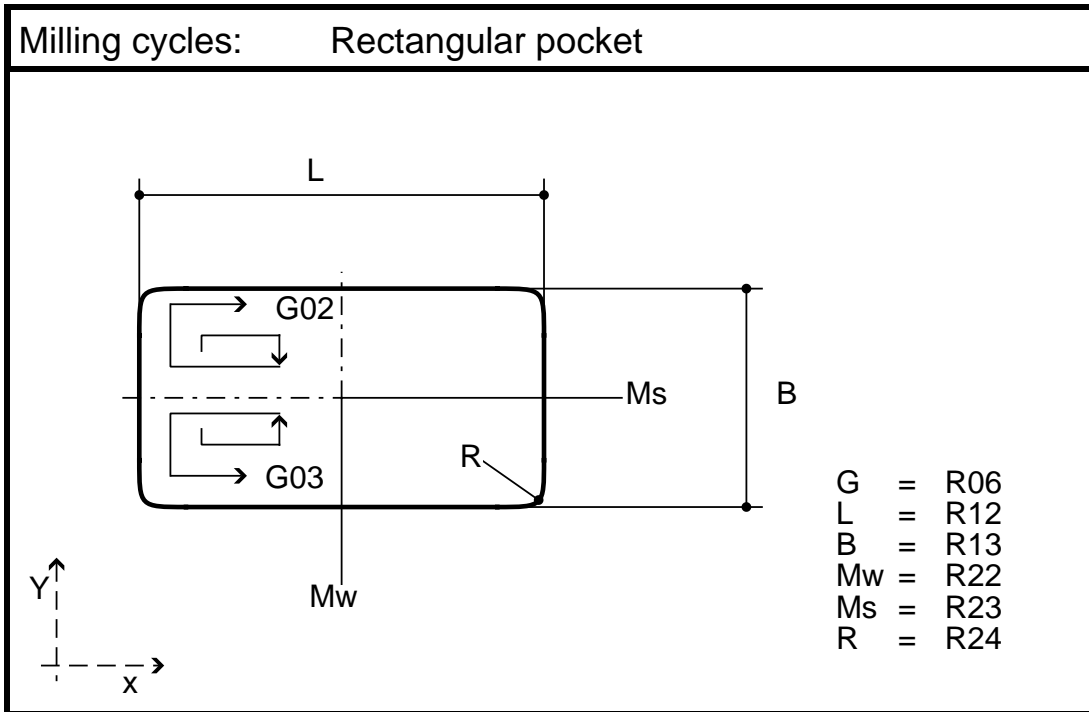
10.4.1 L903 Milling rectangular pockets

The R parameters are entered directly in the part program as parameter assignments. The subroutine L903 is effective in the current plane.

This cycle can also be used to mill circular pockets. If the values are programmed directly as parameter assignments, the R parameters R12 and R13 must be initialized with the value of the pocket diameter. R24 must also be initialized with the pocket radius value.

Symbol	Parameter	Description	
Mw	R22	Pocket centre point (horizontal)	(absolute)
Ms	R23	Pocket centre point (vertical) (referred to workpiece zero)	(absolute)
L	R12	Pocket length	(incremental)
B	R13	Pocket width	(incremental)
R	R24	Corner radius	
G	R06	Milling direction (G02/G03)	
E1	R02	Reference plane	(absolute)
T	R03	Pocket depth	(absolute)
Zt	R01	Input infeed depth without sign	(incremental)
Ff	R15	Feedrate (pocket surface)	
Ft	R16	Feedrate (pocket depth)	

Cutter radius compensation (G40) is cancelled in cycle L903. The cutter radius, which has to be stored in the tool offset memory, is automatically taken into account.



Zt R01: Infeed depth (incremental)

If the infeed depth is parameterized with $R1 = 0$, the infeed is executed directly to pocket depth at the feedrate. If the pocket cannot be milled with a single infeed, an infeed depth must be specified. The milling process is repeated until the pocket depth is reached. If the remaining infeed depth is $< 2 \cdot R01$, it is divided into two equal values. The infeed depth must be specified as an incremental value without sign.

G R06: Milling direction (G02/G03)

The milling direction (up-cut or down-cut milling) must be programmed in $R06 = 02/03$.

L R12: Pocket length (incremental)

B R13: Pocket width (incremental)

If the cutter radius is equal to or greater than one-half of the shorter pocket side, fault message 4102 is output (cutter radius too great).

R R24: Corner radius

Note that the cutter radius must not be greater than the desired corner radius. No fault message is issued.

Example: XY plane, infeed axis Z

%903

N05 G90 G0 X40 Y30 Z20 D05 T04 S600 M03 LF

Select milling position

N10 R1=2.5 R2=2 R3=-5 R6=3 LF

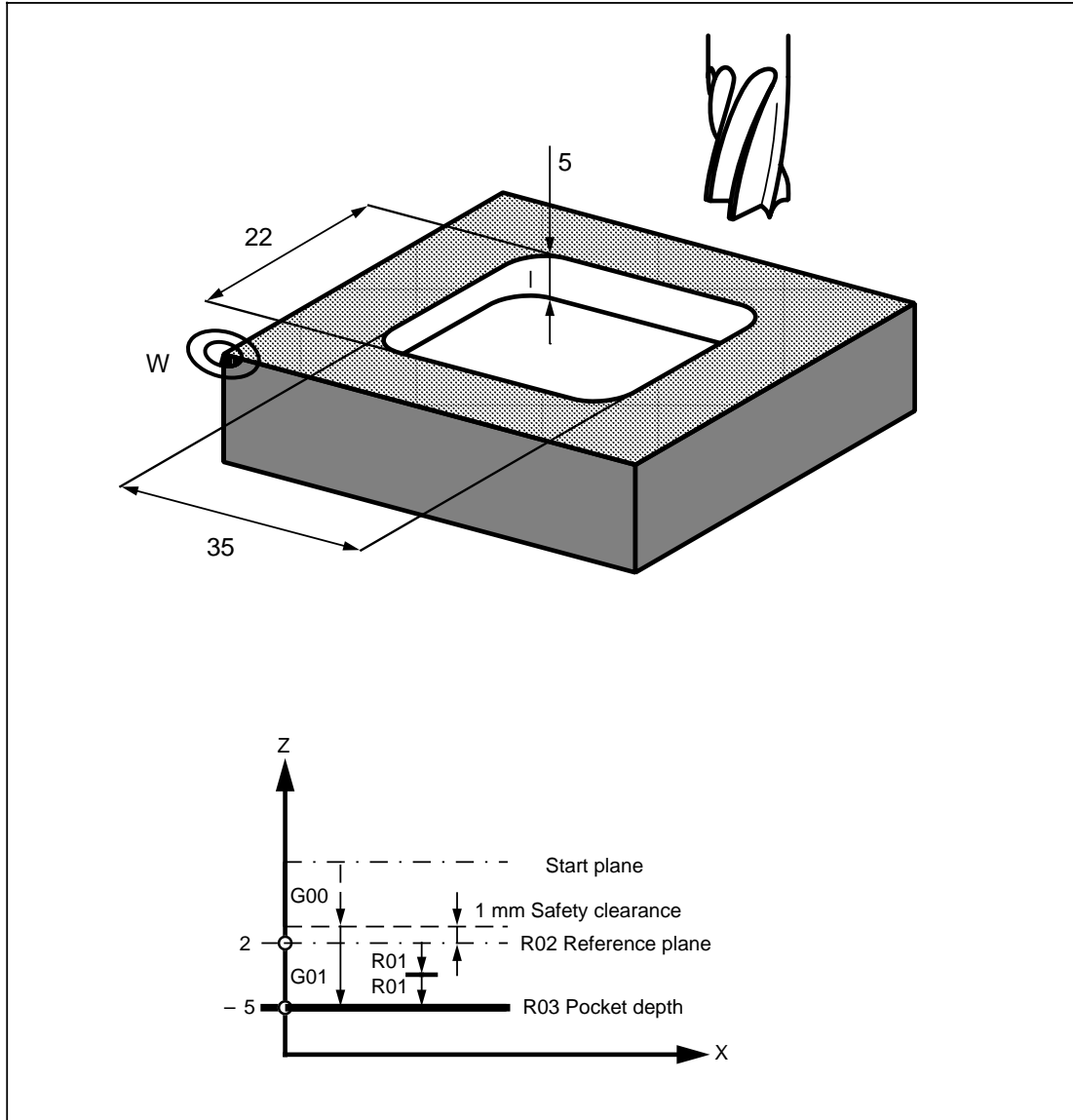
R12=35 R13=22 R15=300 R16=100 LF

R22=40 R23=30 R24=8 L903 P1 LF

Call rectangular pocket

N15 Z50 LF

N20 M30 LF



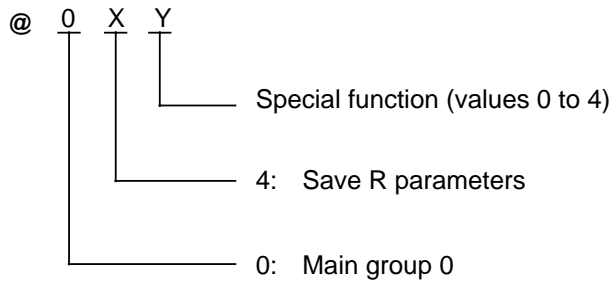
- <Var>** Indirect specification of value by means of R parameter (see <R-Par>) or pointer. A pointer is used to address an R parameter indirectly. "P12" means that the numerical value is located in the R parameter defined by the contents of R12.
e.g. "P10" used as <Var>
Contents of R10: 34
Contents of R34: 3
"3" is the final value for <Var>.
- <Value>** Mixed specification of value. This identifier can be replaced by a constant (e.g. K22), R parameter (e.g. R32) or pointer (e.g. P10).

**Several @ commands can appear in a consecutive sequence
or be mixed with normal part programming.**

11.2 @ code description

11.2.1 Main group 0: general instructions for program structure

The @ code of main group 0 has the following meaning:



Subgroup 4: Save R parameters

These commands are used when working in a subroutine with R parameters which may already have been used at a higher level. The values are saved by writing a push command (@040 or @041) at the beginning of the subroutine. A pop command (@042 or @043) is written at the end of the subroutine to re-establish the original condition.

@040 <Const> <R-Par 1> . . . <R-Par n>

The <Const> constant states the number of following R parameters <R-Par 1>, <R-Par 2>, <R-Par n>. The contents of the listed R parameters are copied into the R parameter stack starting at R300. The saved R parameters are then preset with 0. A maximum of 199 R parameters can be saved.

Example:

Saving the R parameters 7, 9 and 752.

Contents of the R parameters:

```

.
.
.
N15 @040 3 R7 R9 R725
.
.
.

```

R parameter	Contents
7	63
9	- 5
752	- 75

Then the contents of the R parameters are as follows:

R parameter	Contents
7	0
9	0
752	0
STACK B:	:
301	63
302	- 5
303	- 75

@041 <R-Par 1> <R-Par 2>

The contents of the R parameters are copied from <R-Par 1> and <R-Par 2> into the R parameter STACK starting at R300. The saved R parameter area is then overwritten with 0. A maximum of 199 R parameters can be saved.

Example:

Saving the R parameter area from R30 up to and including R76.

Contents of the R parameter area:

```

.
.
.
N15 @041 R30 R76
.
.
.

```

R parameter	Contents
30	1
31	39
76	- 67

Then the contents of the R parameters are as follows:

R parameter	Contents
30	0
31	0
76	0
STACK-B:	⋮
301	1
302	39
⋮	⋮
377	- 67

@042 <Const> <R-Par n> . . . <R-Par 1>

This command takes the saved values out of the STACK register and loads them in the stated R parameters. The values in the STACK are not then set to 0. The R parameters must be input in the reverse sequence to that in @040.

Example:

Reassigning R parameters 7, 9 and 752 their original values saved in the STACK (see @040).

Contents of the STACK:

R parameter	Contents
301	63 (R7)
302	- 5 (R9)
303	- 67 (R752)

⋮

N55 @042 R752 R9 R7

⋮

Contents of the R parameters:

R parameter	Contents
7	63
9	- 5
752	- 67

Subgroup 0: **Absolute jump**

@100 <Const> or @100 <R-Par>

The jump target (block number) and the direction of jump are specified with the constant or contents of the R parameter. A positive block number means that the block to be jumped to is in the direction of the end of the program, and if the block number is negative then the block will be found in the direction of the beginning of the program.

Examples:

Jump back from block N46 to N32.

```

.
.
.
N32 X10 Y10 F1000 L_F
.
.
.
N45 @100 K-32                               as <Const>
or
N45 @100 R12                                as R parameter; contents of R12:-32
    
```

Subgroup 1: **CASE branch**

**@111 <Var> <Value 1> <Const 1>
 <Value 2> <Const 2>
 ⋮ ⋮
 <Value n> <Const n>**

The numerical value defined with the <Var> variable is consecutively compared with the numerical values of the <Value 1> to <Value n> identifiers. If the values agree, a jump is executed to the block determined by the <Const> constant.

Example:

If the contents of R parameter R12 (ACTUAL dimension) are numerical value 21 (UNMACHINED dimension), a jump back to block N6 (machine part) is executed. If the contents of R parameter R12 are equal to that of R parameter R22 (MACHINED dimension), a jump to N40 (end machining) has to be effected.

```

.
.
.
N6 X100 F100 L_F                               Machine part
.
.
.
N35 @111 R12 K21 K-6 R22 K40 L_F              Scan
.
.
.
N40 M02 L_F                                   End machining
    
```

Subgroup 2: IF-THEN-ELSE branch

@121 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is equal to that defined with <Value>, the program continues with the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

The program continues normally if the contents of R parameter R32 (UNMACHINED ACTUAL dimension) are equal to that of R parameter R65 (UNMACHINED SET dimension). If this is not the case, a jump to block N43 is executed.

```

.
.
.
N20 @121 R32 R65 K43 LF          Scan
N25 X100 F100 LF                Unmachined dimension correct
                                   Machine part
.
.
.
N43 M01 LF                      Unmachined dimension incorrect
.                                  Abort machining
.
.

```

@122 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is not equal to that defined with <Value>, the program continues with the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

If the contents of the R parameter (axis ACTUAL value) whose address is in R33 are not equal to numerical value 73.5 (tool change position), the control processes the following block (traverse to tool change position) as normal. If this is not the case, a jump to block N15 (tool change) is executed.

```

.
.
.
N5 @122 P33 K73.5 K15 LF        Scan
N10 M06 T2 LF                  Tool change
N15 X1699 LF                   Traverse to tool change position
.
.
.

```

@123 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is greater than that defined with <Value>, the program continues with the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

If the contents of R parameter R700 (ACTUAL dimension) are greater than the contents of R parameter R702 (SET dimension), the control processes the following block. If this is not the case, a jump to block N40 (conditional stop) is executed.

```

.
.
.
N25 @123 R700 R702 K40 L_F          Scan
N30 X10 F100 L_F                    Continue part machining
.
.
.
N40 M01 L_F                          Conditional stop
.
.
.

```

@124 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is greater than or equal to that defined with <Value>, the program continues with the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

If the contents of R parameter R30 (ACTUAL diameter grinding wheel) are greater than or equal to numerical value 300.7 (minimum diameter), the control processes the following block. If the contents of R30 are less than 300.7, however, a jump to block N30 (wheel change) is executed.

```

.
.
.
N30 M01 L_F                          Wheel change
.
.
.
N55 @124 R30 K300.7 K-30 L_F        Scan
N60 M03 S8000 L_F                    Machine part
.
.
.

```

@125 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is less than that defined with <Value>, the program continues with the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

If the contents of R parameter R24 (dimensional deviation) are less than numerical value 55.3 (tolerance), the control processes the following block. If they are greater than or equal to this numerical value, however, a jump to block N25 (abort machining) is executed.

```

.
.
.
N10 @125 R24 K55.3 K25 L_F          Scan
N15 X100 F100 L_F                    Continue machining
.
.
.
N25 M01 L_F                          Abort machining
.
.
.

```

@126 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is less than or equal to that defined with <Value>, the program continues with the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

If the contents of the R parameter (axis ACTUAL value) whose address is in R44 are less than or equal to the contents of R parameter R34 (clamping position), the control processes the following block as normal. If not, a jump to block N41 is executed.

```

.
.
.
N41  X120  F200  LF           Traverse to clamping position
.
.
.
N85  @126  P44  R34  K-41  LF   Scan
N90  M01
.
.
.

```

Subgroup 3: WHILE loop

The WHILE loop is a repetition instruction with scanning of the repetition condition at the beginning of the loop. The following block is processed as long as the condition is satisfied. A jump to the scan block must always be programmed at the end of the block. If the condition is not fulfilled, an absolute jump to a block is executed.

@131 <Var> <Value> <Const>

As long as the numerical value defined with the <Var> identifier is equal to that defined with <Value>, the program processes the following block. If <Var> is not equal to <Value>, a jump to the block determined by the <Const> constant is executed.

Example:

The control continues to process the following block as long as the contents of R parameter R32 are equal to the contents of R parameter R65. If the values are unequal, a jump to block N43 is executed.

```

.
.
.
N20  @131  R32  R65  K43  LF   Scan
N25  X100  F100  LF
.
.
.
N40  @100  K-20  LF           End of block
N43  M01  LF                 Jump target
.
.
.

```

@132 <Var> <Value> <Const>

As long as the numerical value defined with the <Var> identifier is unequal to that defined with <Value>, the program processes the following block. If the values are equal, a jump to the block determined by the <Const> constant is executed.

Example:

The control processes the following block as long as the contents of the R parameter (ACTUAL value) whose address is in R33 are not equal to numerical value 73.5 (SET value). If the values are equal, a jump to block N15 (tool change position) is executed.

```

.
.
.
N5 @132 P33 K73.5 K15 LF          Scan
N10 X10 LF
.
.
.
N10 @100 K-5 LF                  End of block
.
.
.
N5 X1699 LF                      Tool change position
.
.
.
    
```

@133 <Var> <Value> <Const>

As long as the numerical value defined with the <Var> identifier is greater than that defined with <Value>, the program processes the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

The control processes the following block as long as the contents of R parameter R700 (ACTUAL dimension) are greater than the contents of R parameter R702 (SET dimension). If this is no longer the case, a jump to block N40 (roughing end) is executed.

```

.
.
.
N25 @133 R700 R702 K40 LF        Scan
N30 X10 F100 LF                  Continue part machining
.
.
.
N38 @100 K-25                    End of block
N40 M01 LF                      Roughing end
.
.
.
    
```

@134 <Var> <Value> <Const>

As long as the numerical value defined with the <Var> identifier is greater than or equal to that defined with <Value>, the program processes the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

The control processes the following block as long as the contents of R parameter R30 (ACTUAL diameter grinding wheel) are greater than or equal to numerical value 300.7 (minimum diameter). If the contents of R30 are less than 300.7, however, a jump to block N30 (wheel change) is executed.

```

:
:
N30 M01 LF Wheel change
:
:
N55 @134 R30 K300.7 K-30 LF Scan
N60 M03 S8000 LF Machine part
:
:
N75 @100 K-55 End of block
:
:

```

@135 <Var> <Value> <Const>

As long as the numerical value defined with the <Var> identifier is less than that defined with <Value>, the program processes the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

The control processes the following block as long as the contents of R parameter R24 (dimensional deviation) are less than numerical value 55.3 (tolerance). If they are greater than or equal to this numerical value, however, a jump to block N25 (abort machining) is executed.

```

:
:
N10 @135 R24 K55.3 K25 LF Scan
N15 X100 F100 LF Continue machining
:
:
N20 @100 K-10 LF
N25 M01 LF Abort machining
:
:

```

@136 <Var> <Value> <Const>

As long as the numerical value defined with the <Var> identifier is less than or equal to that defined with <Value>, the program processes the following block. If not, a jump to the block determined by the <Const> constant is executed.

Example:

The control processes the following block as normal as long as the contents of the R parameter (axis ACTUAL value) whose address is in R44 are less than or equal to the contents of R parameter R34 (end of axis). If this is no longer the case, a jump to block N140 is executed.

```

:
:
N85 @136 P44 R34 K-140 LF Scan
N90 X100 F100 LF
:
:
@100 K-850 LF End of block
N141 M01 Traverse end
:
:

```

Subgroup 4: REPEAT loop

The REPEAT loop is a repetition instruction with scanning of the condition at the end of the loop. As long as the condition is not satisfied, a jump back to the block defined under <Const> is executed. If the condition is fulfilled, the loop is exited.

@141 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is equal to that defined with <Value>, the loop is exited.

If <Var> is not equal to <Value>, the loop is processed again.

Example:

The control continues to process the loop (turn off blank) until the contents of R parameter R32 (ACTUAL value) are equal to the contents of R parameter R65 (SET value).

```
.  
. .  
N5 X10 F100 LF Turn off blank  
. .  
N20 @141 R32 R65 K-5 LF Scan  
N25 Y100 F100 LF Next step  
. .  
. .
```

@142 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is not equal to that defined with <Value>, the program is exited.

In case the values are equal, the loop is processed again.

Example:

The control continues to process the loop as long as the contents of the R parameter whose address is in R33 are equal to numerical value 73.5.

```
.  
. .  
N15 X100 F100 LF Start of loop  
. .  
N55 @142 R33 R65 K73.5 K-15 LF Scan  
N10 @100 K-5 LF Continue machining  
. .  
. .
```

@143 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is greater than that defined with <Value>, the loop is exited.

If <Var> is less than or equal to <Const>, the loop is processed again.

Example:

If the contents of R parameter R700 (SET dimension) are greater than the contents of R parameter R702 (ACTUAL dimension), the control exits the loop. If this is not the case, the loop is processed again.

```

:
:
N5  X100  F100  LF           Start of loop
:
:
N25 @143  R700  R702  K-5  LF       Scan
N30  X10   F100  LF           Continue machining part
:
:

```

@144 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is greater than or equal to that defined with <Value>, the loop is exited.

If <Var> is less than <Const>, the loop is processed again.

Example:

The control continues to process the loop as long as the contents of R parameter R30 (ACTUAL diameter milled pocket) are less than numerical value 300.7 (SET diameter).

```

:
:
N30  X10   F100  LF           Start of loop
:
:
N55  @144  R30   K300.7  K-30  LF       Scan
N60  M03   S8000  LF           Continue machining part
:
:

```

@145 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is less than that defined with <Value>, the loop is exited.

If <Var> is greater than or equal to <Value>, the loop is processed again.

Example:

The control exits the loop if the contents of R parameter R24 (ACTUAL dimension) are less than numerical value 55.3 (minimum). If they are greater than or equal to this numerical value, however, the loop is processed again.

```

:
:
N5  X100  F100  LF           Start of loop
:
:
N110 @145  R24  K55.3  K-5  LF       Scan
N115  M01   LF           Abort machining
:
:

```

@146 <Var> <Value> <Const>

If the numerical value defined with the <Var> identifier is less than or equal to that defined with <Value>, the loop is exited.
 If <Var> is greater than <Value>, the loop is processed again.

Example:

The control continues to process the loop as long as the contents of the R parameter (tool life) whose address is in R44 are greater than the contents of R parameter 34 (max. tool life). If this is no longer the case, the loop is exited.

```

:
:
N40 X100 F100 LF           Start of loop
:
:
N85 @146 P44 R34 K-141 LF   Scan
N90 M06 T2 LF             Tool change
:
:
    
```

Subgroup 5: **FOR-TO LOOP**

@151 <Var> <Value> <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. The scan for "equal to" is made at the beginning of the loop. As long as inequality exists the loop is processed, otherwise a jump is executed to the block defined under <Const>. At the end of the loop the <Var> variable must be incremented (@620) and an absolute jump executed back to the start of the loop.

Example:

10 passes of a loop.

```

:
:
N10 @201 R20 K1 LF         Load R20 with value 1
N13 @201 R23 K10 LF        Load R23 with value 10
N15 @151 R20 R23 K60       Scan
N20 X100 F100 LF
:
:
N50 @620 R20 LF            Increment R20
N55 @100 K-15 LF          Jump to scan
N60 M06 T2 LF             Next step
:
:
    
```

Subgroup 6: FOR-DOWN TO LOOP

@161 <Var> <Value> <Const>

The FOR DOWN TO loop is a counting loop in which the contents of the R parameter defined under <Var> are decremented with each pass. The scan for "equal to" is made at the beginning of the loop. As long as inequality exists the loop is processed, otherwise a jump is executed to the block defined under <Const>. At the end of the loop the <Var> variable must be decremented (@621) and an absolute jump executed back to the start of the loop.

Example:

Loop processing continues until R20 = 5.

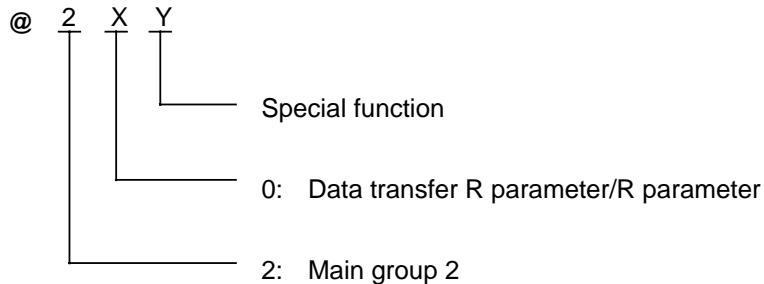
```

:
:
N10  @201  R20  K10  L_F           Load R20 with 10
N15  @201  R23  K5   L_F           Load R23 with 5
N20  @161  R20  R23  K120 L_F       Scan
N25  X100  F100 L_F
:
:
N110 @621  R20  L_F           Decrement R20
N115 @100  K-20 L_F          Jump to scan
N120 M06  T2   L_F           Next step
:
:

```

11.2.3 Main group 2: data transfer general

The @ code of main group 2 has the following meaning:



Subgroup 0: Data transfer R parameter/R parameter

@200 <Var>

The value of the R parameter defined with the <Var> identifier is deleted.

Example:

Assigning the value 0 to R parameter R20.

```

:
:
N20  @200  R20  L_F           R20 is deleted
:
:

```


@201 <Var> <Value>

The numerical value defined under <Value> is loaded under the R parameter defined under <Var>.

Example:

Numerical value 5.6 is loaded in R parameter R20.

```

:
:
N25 @201 R20 K5.6 LF
:
:

```

@202 <Var 1> <Var 2>

The contents of the two R parameters defined under <Var 1> and <Var 2> are exchanged.

Example:

The contents of R20 and R46 are exchanged.

```

:
:
N25 @202 R20 R46 LF
:
:

```

@203 <Var 1> <Var 2><Const>

The bit of the bit pattern in <Var 2> defined by <Const> is written in <Var 1>.

Example:

The 2nd bit from the bit pattern in R22 is read into R5.

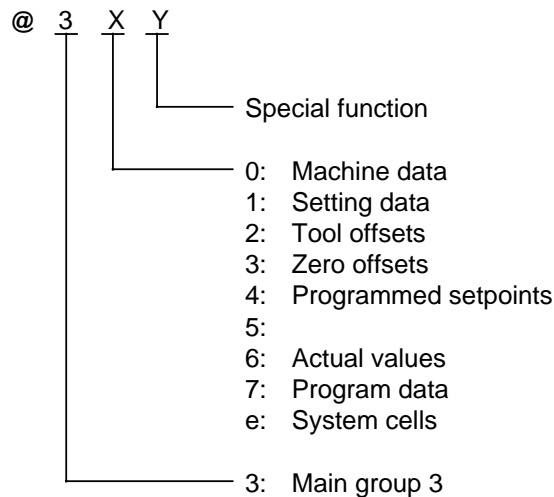
```

:
:
N35 @203 R5 R22 2 LF
:
:

```

11.2.4 Main group 3: data transfer system memory to R parameter

The @ code of main group 3 has the following meaning:



Subgroup 0: Transfer machine data to R parameter

The machine data are always stored in the R parameters in the same code in which they are stored in the machine data (hex hex, binary binary).

@300 <Var> <Value 1>

The NC machine data defined by <Value 1> is loaded in the R parameter defined by <Var>.

Address range for <Value 1>: 0 to 4999.

Example:

NC machine data 4000 is loaded in R parameter R30.

⋮

```
N15 @300 R30 K4000 LF
```

⋮

@301 <Var> <Value 1>

The NC machine data byte defined by <Value 1> is loaded in the R parameter determined by <Var>.

Address range for <Value 1>: 5000 to 6999.

Example:

NC machine data 6200 is loaded in R parameter R55

⋮

```
N30 @301 R55 R45 LF (Contents of R45:6200)
```

⋮

@302 <Var> <Value 1> <Value 2>

An NC machine data bit is loaded in the R parameter defined by <Var>. The byte address is in <Value 1> (range: 5000 to 6999), the bit address is in <Value 2> (range: 0 to 7).

Example:

NC machine data bit 5420.5 is loaded in R parameter R45.

⋮

```
N55 @302 R45 K5420 K5 LF
```

⋮

@306 <Var> <Value 1>

The PLC machine data defined by <Value 1> is loaded in the R parameter defined by <Var>.

Address range for <Value 1>: 0 to 1999.

Example:

PLC machine data 100 is loaded in R parameter R30.

·
·

```
N15 @306 R30 K100 LF
```

·
·

@307 <Var> <Value 1>

A PLC machine data byte is loaded in the R parameter defined by <Var>. The byte address is in <Value 1>.

Address range for <Value 1>: 2000 to 3999.

·
·

Example:

PLC machine data 2400 is loaded in R parameter R55.

·
·

```
N30 @307 R55 R45 LF Contents of R45:2400
```

·
·

@308 <Var> <Value 1> <Value 2>

A PLC machine data bit is loaded in the R parameter defined by <Var>. The byte address is in <Value 1> (range: 2000 to 3999), the bit address is in <Value 2> (range: 0 to 7).

Example:

PLC machine data bit 2420.5 is loaded in R parameter R45.

·
·

```
N55 @ 308 R45 K2420 K5 LF
```

·
·

Subgroup 1: Transfer setting data to R parameter

The setting data are always stored in the R parameters in the same code in which they are stored in the setting data (hex hex, binary binary).

@310 <Var> <Value 1>

The setting data defined by <Value 1> is loaded in the R parameter defined by <Var>.

Address range for <Value 1>: 0 to 4999.

Example:

Setting data 4000 is loaded in R parameter R30.

⋮

```
N15 @310 R30 K4000 LF
```

⋮

@311 <Var> <Value 1>

The setting data byte defined by <Value 1> is loaded in the R parameter determined by <Var>.

Address range for <Value 1>: 5000 to 9999.

Example:

Setting data byte 6200 is loaded in R parameter R55.

⋮

```
N30 @311 R55 R45 LF Contents of R45:6200
```

⋮

@312 <Var> <Value 1> <Value 2>

A setting data bit is loaded in the R parameter defined by <Var>. The byte address is in <Value 1> (range: 5000 to 9999), the bit address is in <Value 2> (range: 0 to 7).

Example:

Setting data bit 5420.5 is loaded in R parameter R45.

⋮

```
N55 @312 R45 K5420 K5 LF
```

⋮

Subgroup 2: Transfer tool offsets to R parameter

@320 <Var> <Value 1> <Value 2> <Value 3>

The identifiers must be assigned as follows:

<Var>	R parameter into which the transfer is being made
<Value 1>	TO area (address always 0)
<Value 2>	Tool offset number (address 1 to 99)
<Value 3>	Number of tool offset memory (P number) (address 0 to 9)

Example:

P number 8 for tool 27 is loaded in R parameter R45. The tool offset addresses are stored in R parameters.

```

:
:
N20 @201 R20 K27 L_F          Load tool No. 27 in R20
N25 @201 R21 K8 L_F           Load P No. 8 in R21
N30 @320 R45 K0 R20 R21 L_F   Load command
:
:

```

Subgroup 3: Transfer zero offsets to R parameter

@330 <Var> <Value 1> <Value 2> <Value 3>

The identifiers must be assigned as follows:

<Var> R parameter into which the transfer is being made
 <Value 1> Group of settable zero offsets (G54 = 1, G57 = 4)
 <Value 2> Axis number (1 to 4)
 <Value 3> Coarse (0) or fine (1) value

Example:

The fine value of zero offset G55 of the 2nd axis is loaded in R parameter R33.

```

:
:
N35 @330 R33 K2 K2 K1 L_F
:
:

```

@331 <Var> <Value 1> <Value 2>

The identifiers must be assigned as follows:

<Var> R parameter into which the transfer is being made
 <Value 1> Group of programmable additive zero offsets (G58 = 1, G59 = 2)
 <Value 2> Axis number (1 to 4)

Example:

The additive zero offset G59 of the 3rd axis is loaded in R parameter R44.

```

:
:
N55 @331 R44 R12 R14 L_F          Contents of R12: 2
:                                   R14:3
:
:

```

@ 332 <Var> <Value 2>

External zero offset with input via PLC. The identifiers must be assigned as follows:

<Var> R parameter into which the transfer is being made
 <Value 2> Number of axis from which the external ZO is being transferred (1 to 4)

Example:

An external zero offset has been specified by the PLC program (FB 62). The external ZO that is valid for the 2nd axis is loaded in R parameter R34.

```

  :
  :
N35 @332 R34 K2 L_F
  :
  :
```

@ 333 <Var> <Value 2>

DRF offset. The identifiers must be assigned as follows:

<Var> R parameter into which the transfer is being made
 <Value 2> Number of axis from which the DRF offset is being transferred (1 to 4)

Example:

A DRF offset has been entered in the control. The DRF offset that is valid for the 2nd axis is loaded in R parameter R34.

```

  :
  :
N35 @333 R34 K2 L_F
  :
  :
```

@ 334 <Var> <Value 2>

PRESET offset. The identifiers must be assigned as follows:

<Var> R parameter into which the transfer is being made
 <Value 2> Number of axis from which the PRESET offset is being transferred
 (1 to 4)

Example:

A PRESET offset has been entered in the control. The PRESET offset that is valid for the 2nd axis is loaded in R parameter R34.

```

  :
  :
N35 @334 R34 K2 L_F
  :
  :
```

@336 <Var> <Value 2>

The total zero offset contains the selected settable ZO, the programmable additive ZO, the external ZO and the selected tool offset. The identifiers must be assigned as follows:

<Var> R parameter into which the transfer is being made
 <Value 2> Number of axis from which the total zero offset is being transferred (1 to 4)

Example:

Various zero offsets have been entered in the control. The total zero offset that is valid for the 2nd axis is loaded in R parameter R34.

⋮
 ⋮
 ⋮

```
N35 @336 R34 K2 L_F
```

⋮
 ⋮

Subgroup 4: Programmed setpoints

@342 <Var> <Value 1> <Value 3>

The programmed spindle speed is read out in the <Var> identifier. 0 must always be in <Value 1> and <Value 3>.

Example:

The programmed spindle speed is written in R parameter R23.

⋮
 ⋮

```
N35 @342 R23 K0 K0 L_F
```

⋮
 ⋮

@345 <Var> <Value 1> <Value 2>

The programmed cutting rate is read out in the <Var> identifier. 0 must always be in <Value 1> and <Value 2>.

Example:

The programmed cutting rate is written in R parameter R23.

⋮
 ⋮

```
N45 @345 R23 K0 K0 L_F
```

⋮
 ⋮

Subgroup 6: Transfer actual values to R parameters

@360 <Var> <Value 2>

The workpiece-related actual value of the axis defined by <Value 2> (range: 1 to 4) is loaded in the R parameter defined by <Var>.

Example:

The workpiece-related actual value of the 3rd axis is loaded in R parameter R32.

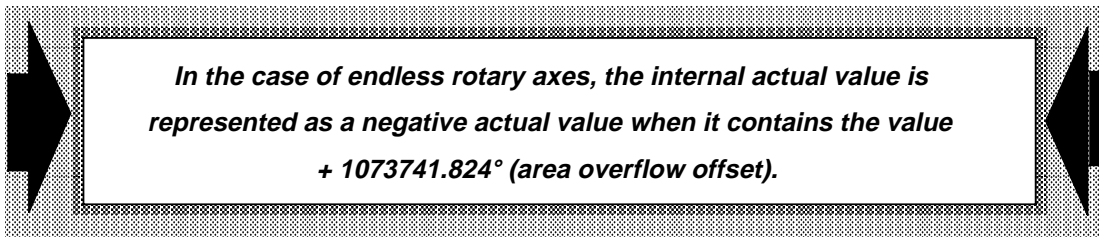
```

:
:
N35 @360 R32 R55 Lp           Contents of R55: 3
:
:

```

@361 <Var> <Value 2>

The machine-related actual value of the axis defined by <Value 2> (range: 1 to 4) is loaded in the R parameter defined by <Var>.



Example:

The machine-related actual value of the 3rd axis is loaded in R parameter R32.

```

:
:
N35 @361 R32 R55 Lp           Contents of R55: 3
:
:

```

@363 <Var> <Value 2>

The actual position of the spindle defined by <Value 2> (range: 1) is loaded in the R parameter defined by <Var>.

Example:

The actual position of the spindle is loaded in R parameter R32.

```

:
:
N35 @363 R32 R55 Lp           Contents of R55: 1
:
:

```


@364 <Var> <Value 2>

The actual speed of the spindle defined by <Value 2> (range: 1) is loaded in the R parameter defined by <Var>.

Example:

The actual speed of the spindle is loaded in R parameter R32.

```

.
.
.
N35 @364 R32 R55 L_F           Contents of R55: 1
.
.
.

```

@367 <Var> <Value 1>

This command can be used to store the axis numbers of the current plane and the spindle numbers in R parameters. The data are stored in a total of five R parameters starting with R parameter Rn defined by <Var>.

Rn Number of horizontal axis
Rn+1 Number of vertical axis
Rn+2 Number of axis perpendicular to plane
Rn+3 Number of axis effective in length 2 (see tool type 30)
Rn+4 Number of leadscrew

<Value 1> must always be 0.

Cycles can be programmed with general validity with this command.

Example:

The valid axis and spindle numbers are loaded in R parameters R50 to R54 (milling machine with X-Y plane).

```

.
.
.
N45 @367 R50 K0 L_F           Contents of R50: 1 (X)
.                               R51: 2 (Z)
.                               R52: 3 (Y)
.                               R53: 0
.                               R54: 1 (spindle)

```

@36a <Var> <Value 1>

The selected tool offset (D number) is transmitted to the R parameter defined by <Var>. <Value 1> must always be 0.

Example:

The valid D number is loaded in R parameter R32.

```

.
.
.
N35 @36a R32 K0 L_F
.
.
.

```

@36b <Var> <Value 1>> <Value 3>

This command transfers the G function of the part program block currently being processed to the R parameter defined by <Var>. <Value 1> must always be 0. <Value 3> defines the **internal** G group (note program key) to which the G function belongs.

Example:

The current G function of internal group 5 (zero offsets) is loaded in R parameter R44.

```

:
:
:
N55 @36b R44 K0 K5 L_F
:
:
:

```

Subgroup 7: **Load program data in R parameter**

@371 <Var> <Value 1> <Value 3>

This command reads special bits for the acquisition of various active signals in the R parameter defined by <Var>.

In order to read out the following bits:

- 0 must always be entered in <Value 1>.
- the desired bit number must be entered in <Value 3>.

Bit 0 = Block search active

Bit 1 = Dry run feedrate active

Example:

The state of the "dry run feedrate active" special bit is transferred to R parameter R22.

```

:
:
:
N55 @371 R22 K0 K1 L_F
:
:
:

```

In order to read out the following bits, 99 must always be entered in <Value 1>. The desired bit number must be entered in <Value 3>.

Bit 0 = Measuring input active

Bit 1 = Measuring input active

Example:

The state of measuring input 1 is transferred to R parameter R44.

```

:
:
:
N35 @371 R44 K99 K0 L_F
:
:
:

```

Subgroup e: **System cells**

@3e4 <Var> <Value 1>

The current gear stage is read in the <Var> identifier. <Value 1> must always be 0.

Example:

The current gear stage of the spindle is read out into R parameter R36.

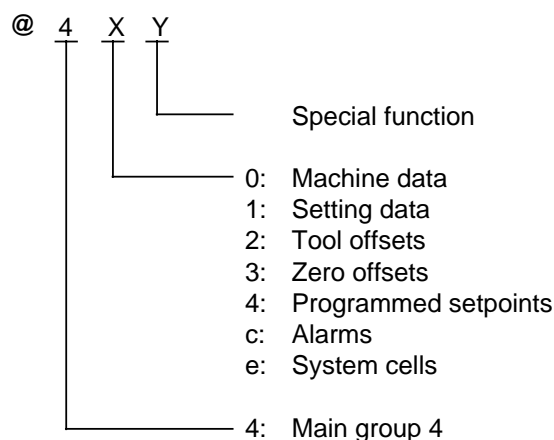
```

:
:
N35 @3e4 R36 K0 Lp
:
:

```

11.2.5 Main group 4: data transfer R parameter to system memory

The @ code of main group 4 has the following meaning:



Subgroup 0: **Transfer R parameter to machine data**

@400 <Value 1> <Value>

The numerical value defined by <Value> is loaded in the NC machine data defined by <Value 1> (address: 0 to 4999).

Example:

Value 0500 is entered in NC machine data 4000.

```

:
:
N45 @400 K4000 K500 Lp
:
:

```

@401 <Value 1> <Value>

The byte defined with <Value> is written in the NC machine data byte defined by <Value 1> (address: 5000 to 6999).

Example:

Byte 01011100 is written in NC machine data byte 5049.

```

:
:
N35 @401 R22 R35 LF           Contents of R22: 5049
:                               R35: 01011100
:

```

@402 <Value 1> <Value 2> <Value>

The bit state defined with <Value> is written in the NC machine data bit defined by <Value 1> (byte address: 5000 to 6999) and <Value 2> (bit address: 0 to 7).

Example:

The NC machine data bit 5003.6 is set to state 1.

```

:
:
N25 @402 K5003 K6 R3 LF       Contents of R3: 1
:
:

```

@406 <Value 1> <Value>

The numerical value defined with <Value> is written in the PLC machine data defined by <Value 1> (address: 0 to 1999).

Example:

The value 6 is entered in PLC machine data 0.

```

:
:
N45 @406 K0 K6 LF
:
:

```

@407 <Value 1> <Value>

The byte defined with <Value> is written in the PLC machine data byte defined by <Value 1> (address: 2000 to 3999).

Example:

Bit pattern 11110000 is written in PLC machine data byte 2001.

```

:
:
N35 @407 R22 R35 LF           Contents of R22: 2001
:                               R35: 11110000
:

```

@408 <Value 1> <Value 2> <Value>

The bit state defined with <Value> is written in the PLC machine data bit defined by <Value 1> (byte address: 2000 to 3999) and <Value 2> (bit address: 0 to 7).

Example:

The PLC machine data bit 2003.6 is set to state 1.

·
·
·

N25 @408 K2003 K6 R3 L_F Contents of R3: 1

·
·

Subgroup 1: Transfer R parameter to setting data

@410 <Value 1> <Value>

The numerical value defined with <Value> is written in the setting data defined by <Value 1> (address: 0 to 4999).

Example:

The value 9500 is entered in setting data 4010.

·
·

N45 @410 K4010 K9500 L_F

·
·

@411 <Value 1> <Value>

The byte defined with <Value> is written in the setting data byte defined by <Value 1> (address: 5000 to 9999).

Example:

The byte 00000100 is written in setting data byte 5010.

·
·

N35 @411 R22 R35 L_F Contents of R22: 5010
R35: 00000100

·
·

@412 <Value 1> <Value 2> <Value>

The bit state defined with <Value> is written in the setting data bit defined by <Value 1> (byte address: 5000 to 9999) and <Value 2> (bit address: 0 to 7).

Example:

The setting data bit 5001.1 is set to state 1.

·
·

N25 @412 K5001 K1 R3 L_F Contents of R3: 1

·
·

Subgroup 2: Write R parameter in tool offsets

@420 <Value 1> <Value 2> <Value 3> <Value>

The numerical value defined with <Value> is written in the tool offset memory defined by <Value 1> to <Value 3>.

The identifiers must be assigned as follows:

<Value 1> TO area (address always 0)
 <Value 2> Tool offset number (1 to 99)
 <Value 3> Number of tool offset value (0 to 9)

Example:

The numerical value 20 is transferred as the new radius value (P number 4) to tool offset number 15.

```

:
:
N65 @420 K0 K15 K4 R24 L_F           Contents of R24: 20
:
:

```

@423 <Value 1> <Value 2> <Value 3> <Value>

The numerical value defined with <Value> is added to the tool offset memory defined by <Value 1> to <Value 3>.

The identifiers must be assigned as follows:

<Value 1> TO area (address always 0)
 <Value 2> Tool offset number (1 to 99)
 <Value 3> Number of tool offset value (0 to 9)

Example:

The numerical value -3 is added as an additive radius value (P number 4) to tool offset number 15.

```

:
:
N65 @423 K0 K15 K4 R24 L_F           Contents of R24: -3
:
:

```

Subgroup 3: Write R parameter in zero offsets

@430 <Value 1> <Value 2> <Value 3> <Value>

The numerical value defined with <Value> is added to the zero offset memory defined by <Value 1> to <Value 3>.

The identifiers must be assigned as follows:

<Value 1> Group of settable zero offsets (G54 = 1 to G57 = 4)
 <Value 2> Axis number (1 to 4)
 <Value 3> Coarse (0) or fine (1) value

Example:

The numerical value 16 is written as a coarse value for the 2nd axis in the G56 zero offset group.

```

:
:
N65 @430 K3 K2 K0 R24 L_F           Contents of R24: 16
:
:

```

@431 <Value 1> <Value 2> <Value 3> <Value>

The numerical value defined with <Value> is added to the zero offset memory defined by <Value 1> to <Value 3>.

The identifiers must be assigned as follows:

- <Value 1> Group of programmable zero offsets (G54 = 1 to G57 = 4)
- <Value 2> Axis number (1 to 4)
- <Value 3> Coarse (0) or fine (1) value

Example:

The numerical value 2 is added to the coarse value for the 2nd axis in the G56 zero offset group.

```

:
:
N65 @431 K3 K2 K0 R24 L_F           Contents of R24: 2
:
:

```

@432 <Value 1> <Value 2> <Value>

The numerical value defined by <Value> is written in the additive zero offset defined by <Value 1> and <Value 2>.

The identifiers must be assigned as follows:

- <Value 1> Group of programmable additive zero offsets (G58 = 1, G59 = 2)
- <Value 2> Axis number (1 to 4)

Example:

The numerical value -3.4 is written in the additive zero offset G59 of the 3rd axis.

```

:
:
N55 @432 R44 R12 K-3.4 L_F           Contents of R12: 2
:                                       R44: 3
:
:

```

@434 <Value 2> <Value>

The numerical value defined by <Value> is entered as a DRF offset in the axis defined by <Value 2> (1 to 4).

Example:

The numerical value -5.35 is entered as the DRF offset for the 1st axis.

```

:
:
N35 @434 K1 K-5.35 L_F
:
:

```

@435 <Value 2> <Value>

The numerical value defined by <Value> is entered as a PRESET offset in the axis defined by <Value 2> (1 to 4).

Example:

The numerical value – 5.35 is entered as the PRESET offset for the 1st axis.

```

:
:
N35 @435 K1 K-5.35 LF
:
:

```

Subgroup 4: Write R parameter in programmed setpoints
@440 <Value 3> <Value>

This command permits axes to be programmed independent of an axis name. The number of the axis to be traversed (1 to 4) is entered under <Value 3> and the position to be approached or the path is specified under <Value>.

Example:

The 3rd axis is traversed to position 300.5 (G90 already effective).

```

:
:
N55 @440 K3 K300.5 LF
:
:

```

@442 <Value 3> <Value>

This command can be used to write the programmed spindle speed in the NC. The speed is specified by <Value>. 0 must always be entered in <Value 3>.

Example:

The numerical value 8000 is entered as the new programmed spindle speed.

```

:
:
N65 @442 K0 K8000 LF
:
:

```

@446 <Value>

This command permits the radius to be programmed independent of the address specified in the machine data. The numerical value is specified under <Value>.

Example:

Numerical value 6.5 is entered as the radius.

```

:
:
N65 @446 K6.5 LF
:
:

```


@447 <Value>

This command permits the angle to be programmed independent of the address specified in the machine data.

Example:

Numerical value 60 is entered as the angle.

⋮

N85 @447 R44

Contents of R44: 60

⋮

@448 <Value 3> <Value>

This command writes a value in the interpolation parameter of an axis. The numerical value is in <Value> and the axis (1 to 4) is in <Value 3>.

Example:

Circular path programming G02 X50 Y45 I5 J0.



⋮

N40 G01 G90 @440 K2 K50 @440 K1 K45 F500 L_P Start point of circular path

N45 G02 G90 @440 K1 K50 @440 K2 K45 @448 K1 K5 @448 K2 K0 L_P

⋮

Subgroup c: Alarms

@4c0 <Value>

This command can be used to display a cycle alarm at the control. The alarm No. is specified with <Value> (4000 to 4299; 5000 to 5099).

Example:

Cycle alarm 4093 is displayed.

⋮

N65 @4c0 K4093 L_P

⋮

Subgroup e: **System cells**

@4e1 <Value 1> <Value 2> <Value>

The spindle acceleration time constant is specified with <Value>, whereby the values specified in the machine data are not overwritten. The machine data become effective again after thread machining is cancelled, after RESET and after POWER ON. Values less than 4 are not accepted. 0 must always be entered in <Value 1>. The gear stage is defined by <Value 2>.

Example:

8 is the new value for the spindle acceleration time constant of the 3rd speed stage of the spindle.

```

:
:
:
N55 @4e1 K0 K3 K8 L_F
:
:
:

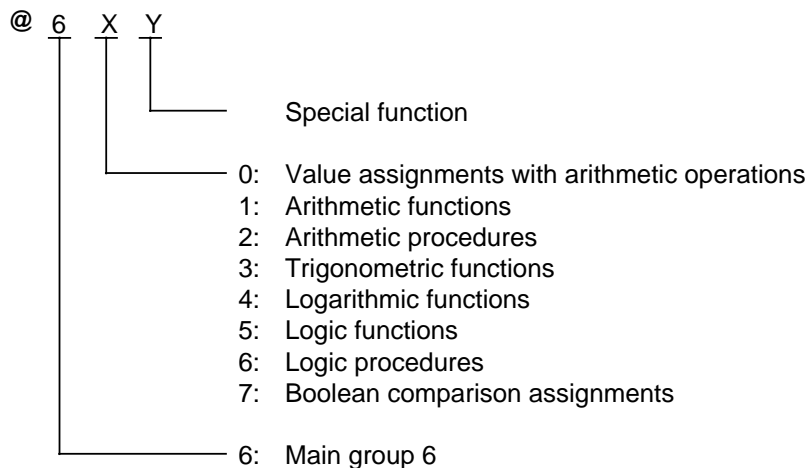
```

11.2.6 Main group 5: file handling general

(available soon)

11.2.7 Main group 6: mathematical functions

The @ code of main group 6 has the following meaning:



Subgroup 0: Value assignments with arithmetic operations

No @ is required in this subgroup. A chain calculation with several identifiers on the right 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

Subgroup 1: Arithmetic functions

@610 <Var> <Value>

The unsigned amount of the numerical value defined under <Value> is stored in the <Var> identifier.

Example:

The unsigned amount of the contents of R34 is stored in R parameter R23.

```

:
:
: Contents of R34: -5.78
N35 @610 R23 R34 LF
:
: Contents of R23: 5.78
    
```

@613 <Var> <Value>

The square root of the numerical value defined under <Value> is stored in the <Var> identifier.

Example:

The square root of the contents of R3 is stored in R parameter R45.

```

:
:
: Contents of R3: 16
N35 @613 R45 R3
:
: Contents of R45: 4
    
```

@614 <Var> <Value 1> <Value 2>

The sum of the squares of the numerical values defined under <Value 1> and <Value 2> is formed; the square root of the sum is stored in the <Var> identifier.

Example:

The sum of the squares of R36 and R55 is formed and the square root of the sum is stored in R22.

```

:
:
: Contents of R36: 3
: Contents of R55: 4
N25 @614 R22 R36 R55 LF
:
: 3 · 3 + 4 · 4 = 25
:
: Contents of R22: 5
    
```

Subgroup 2: Arithmetic procedures

@ 620 <Var>

The contents of the R parameter defined under <Var> are increased by 1 (incremented).

Example:

The contents of R46 are increased by 1 with each pass.

```

:
:
N65 @620 R46 LF
:
:

```

@ 621 <Var>

The contents of the R parameter defined under <Var> are decreased by 1 (decremented).

Example:

The contents of R46 are decremented with each pass.

```

:
:
N65 @621 R46 LF
:
:

```

@ 622 <Var>

The integer component of the numerical value defined by an R parameter or pointer is formed. The result is then contained in the same R parameter or pointer.

Example:

The integer component of the contents of R34 is formed.

```

:
:
:
:
N75 @622 R34
:
:
:
:

```

Contents of R34: 5.32 - before

Contents of R34: 5 - after

Subgroup 3: Trigonometric functions

@630 <Var> <Value>

The sine of the angle defined under <Value> is formed and stored in the <Var> identifier.

Example:

The sine of the numerical value contained in R parameter R30 is formed and the result stored in R parameter R46.

```
⋮  
⋮  
N35 @630 R46 R30 L_F Contents of R30: 45  
⋮ Contents of R46: 0.70710  
⋮
```

@631 <Var> <Value>

The cosine of the angle defined under <Value> is formed and stored in the <Var> identifier.

Example:

The cosine of the numerical value contained in R parameter R30 is formed and the result stored in R parameter R46.

```
⋮  
⋮  
N35 @631 R46 R30 L_F Contents of R30: 60  
⋮ Contents of R46: 0.5  
⋮
```

@632 <Var> <Value>

The tangent of the angle defined under <Value> is formed and stored in the <Var> identifier.

Example:

The tangent of the numerical value contained in R parameter R30 is formed and the result stored in R parameter R46.

```
⋮  
⋮  
N35 @632 R46 R30 L_F Contents of R30: 45  
⋮ Contents of R46: 1  
⋮
```

@634 <Var> <Value>

The arc sine of the angle defined under <Value> is formed and stored in the <Var> identifier.

Example:

The arc sine of the numerical value contained in R parameter R30 is formed and the result stored in R parameter R46.

⋮
⋮

N35 @634 R46 R30 L_F

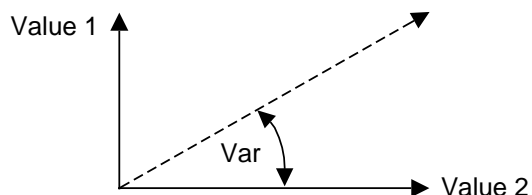
⋮
⋮

Contents of R30: 0.5
Contents of R46: 30

@637 <Var> <Value 1> <Value 2>

The numerical values defined under <Value 1> and <Value 2> are considered as vectors. The result, <Var>, is the angle between the component under <Value 2> and the resultant vector.

In mathematical terms, the result is the angle between the hypotenuse and the adjacent side, <Value 2>.



Only one constant <Const> is allowed as an operand for <Value 1> and <Value 2>. The other operand must be a variable (R parameter or pointer).

Subgroup 4: **Logarithmic functions**

@640 <Var> <Value>

The natural logarithm of the numerical value defined under <Value> is formed and stored in the <Var> identifier.

Example:

The natural logarithm of the numerical value contained in R parameter R12 is formed and the result stored in R parameter R55.

⋮
⋮

N45 @640 R55 R12 L_F

⋮
⋮

Contents of R12: 10
Contents of R55: 2.302585

Subgroup 5: **Logic functions**

@650 <Var> <Var 1> <Value>

The bit patterns under <Var 1> and <Value> (max. 8 bits) are logically ORed. The result is stored in <Var>.

Example:

The bit patterns in R33 and R34 are logically ORed. The result is stored in R45.

```
.  
. .  
N55 @650 R45 R33 R34 L_F Contents of R33: 10101100  
. R34: 00110111  
. Contents of R45: 10111111  
.
```

@651 <Var> <Var 1> <Value>

The bit patterns under <Var 1> and <Value> (max. 8 bits) are logically EXORed. The result is stored in <Var>.

Example:

The bit patterns in R33 and R34 are logically EXORed. The result is stored in R45.

```
.  
. .  
N55 @651 R45 R33 R34 L_F Contents of R33: 10101100  
. R34: 00110111  
. Contents of R45: 10010011  
.
```

@652 <Var> <Var 1> <Value>

The bit patterns under <Var 1> and <Value> (max. 8 bits) are logically ANDed. The result is stored in <Var>.

Example:

The bit patterns in R33 and R34 are logically ANDed. The result is stored in R45.

```
.  
. .  
N55 @652 R45 R33 R34 L_F Contents of R33: 10101100  
. R34: 00110111  
. Contents of R45: 00100100  
.
```

@ 653 <Var> <Var 1> <Value>

The bit patterns under <Var 1> and <Value> (max. 8 bits) are logically ANDed.
The result is negated and stored in <Var>.

Example:

The bit patterns in R33 and R34 are logically ANDed. The negated result is stored in R45.

```

.
.
N55 @653 R45 R33 R34 LF Contents of R33: 10101100
.                                     R34: 00110111
.                                     Contents of R45: 11011011
.

```

@ 654 <Var> <Value>

The bit pattern under <Value> (max. 8 bits) is negated. The result is stored in the identifier <Var>.

Example:

The bit pattern in R33 is negated. The result is stored in R45.

```

.
.
N55 @654 R45 R33 LF Contents of R33: 10101100
.                                     Contents of R45: 01010011
.

```

@ 655 <Var> <Var 1> <Value>

The bits under <Var 1> and <Value> are logically ORed. The result is stored in <Var>.

Example:

The bits in R33 and R34 are logically ORed. The result is stored in R45.

```

.
.
N55 @655 R45 R33 R34 LF Contents of R33: 1
.                                     R34: 0
.                                     Contents of R45: 1
.

```

@ 656 <Var> <Var 1> <Value>

The bits under <Var 1> and <Value> are logically EXORed. The result is stored in <Var>.

Example:

The bits in R33 and R34 are logically EXORed. The result is stored in R45.

```

.
.
N55 @656 R45 R33 R34 LF Contents of R33: 1
.                                     R34: 0
.                                     Contents of R45: 1
.

```


@657 <Var> <Var 1> <Value>

The bits under <Var 1> and <Value> are logically ANDed. The result is stored in the <Var> identifier.

Example:

The bits in R33 and R34 are logically ANDed. The result is stored in R45.

```
.  
. .  
N55 @657 R45 R33 R34 L_F Contents of R33: 1  
. R34: 0  
. Contents of R45: 0  
.
```

@658 <Var> <Var 1> <Value>

The bits under <Var 1> and <Value> are logically ANDed. The result is negated and stored in the <Var> identifier.

Example:

The bits in R33 and R34 are logically ANDed. The negated result is stored in R45.

```
.  
. .  
N55 @658 R45 R33 R34 L_F Contents of R33: 1  
. R34: 0  
. Contents of R45: 1  
.
```

@659 <Var> <Value>

The bit under <Value> is logically negated. The result is stored in the <Var> identifier.

Example:

The bit in R33 is negated. The result is stored in R45.

```
.  
. .  
N55 @659 R45 R33 R34 L_F Contents of R33: 0  
. Contents of R45: 1  
.
```

Subgroup 6: Logic procedures

@660 <Var> <Const>

A bit (0 to 7) defined by the <Const> constant is deleted in the bit pattern determined by <Var>.

Example:

Bit 3 in bit pattern 01101111 is reset to 0.

⋮

N15 @660 K3 R29 L_F

⋮

Contents of R29: 01101111

Contents of R29: 01100111

@661 <Var> <Const>

A bit (0 to 7) defined by the <Const> constant is set in the bit pattern determined by <Var>.

Example:

Bit 3 in bit pattern 01100011 is set to 1.

⋮

N15 @661 K3 R29 L_F

⋮

Contents of R29: 01100011

Contents of R29: 01101011

Subgroup 7: Boolean comparison assignments

@671 <Var 1> <Var 2> <Value >

If the numerical values defined under <Var 2> and <Value> are equal, the Boolean variable <Var 1> is set to 1.

Example:

If the numerical values in the R parameters R34 and R44 are equal, the contents of R parameter R17 are set to 1.

⋮

N35 @671 R17 R34 R44 L_F

⋮

Contents of R34: 478

R44: 478

Contents of R17: 1

@672 <Var 1> <Var 2> <Value >

If the numerical values defined under <Var 2> and <Value> are not equal, the Boolean variable <Var 1> is set to 1.

Example:

If the numerical values in the R parameters R34 and R44 are not equal, the contents of R parameter R17 are set to 1.

```
·  
·  
·  
N35 @672 R17 R34 R44 L_F Contents of R34: 478  
· R44: 405  
· Contents of R17: 1  
·
```

@673 <Var 1> <Var 2> <Value >

If the numerical value defined under <Var 2> is greater than that under <Value>, the Boolean variable <Var 1> is set to 1.

Example:

If the numerical value of R parameter R34 is greater than that under R44, the contents of R parameter R17 are set to 1.

```
·  
·  
·  
N35 @673 R17 R34 R44 L_F Contents of R34: 678  
· R44: 478  
· Contents of R17: 1  
·
```

@674 <Var 1> <Var 2> <Value >

If the numerical value defined under <Var 2> is greater than or equal to that under <Value>, the Boolean variable <Var 1> is set to 1.

Example:

If the numerical value of R parameter R34 is greater than or equal to that under R44, the contents of R parameter R17 are set to 1.

```
·  
·  
·  
N35 @674 R17 R34 R44 L_F Contents of R34: 478  
· R44: 478  
· Contents of R17: 1  
·
```

@675 <Var 1> <Var 2> <Value >

If the numerical value defined under <Var 2> is less than that under <Value>, the Boolean variable <Var 1> is set to 1.

Example:

If the numerical value of R parameter R34 is less than that under R44, the contents of R parameter R17 are set to 1.

```
·  
·  
·  
N35 @675 R17 R34 R44 L_F Contents of R34: 148  
· R44: 478  
· Contents of R17: 1  
·
```

@676 <Var 1> <Var 2> <Value >

If the numerical value defined under <Var 2> is less than or equal to that under <Value>, the Boolean variable <Var 1> is set to 1.

Example:

If the numerical value of R parameter R34 is less than or equal to that under R44, the contents of R parameter R17 are set to 1.

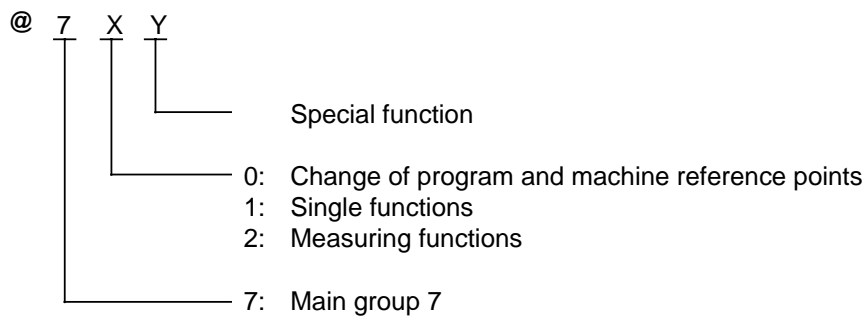
```

:
:
:
N35 @674 R17 R34 R44 LF           Contents of R34: 648
:                                     Contents of R44: 478
:                                     Contents of R17: 0
:

```

11.2.8 Main group 7: NC-specific functions

The @ code of main group 7 has the following meaning:



Subgroup 0: Change of program and machine reference points

@706

This command, which is effective block by block, suppresses all the zero offsets as well as the PRESET and DRF offsets. To approach a fixed machine point, the tool offsets must also be cancelled. If machine data 5007.1 is set, G53 acts like @706. If it is not set, G53 does not suppress the PRESET and DRF offsets.

Subgroup 1: **Single functions**

@710 <Var 1> <Var 2>

This command is required for reference point edit purposes in "stock removal" turning cycle L96. The identifiers have the following meanings:

- <Var 1> Refers to an R parameter which records the results of the reference point edit with seven additional R parameters following in immediate succession.
- <Var 2> Refers to an R parameter which contains the input data for the reference point edit with three additional R parameters following in immediate succession.

@711 <Var 1> <Var 2> <Var 3>

This command is required for calculation of intersection purposes in "stock removal" turning cycle L96. The identifiers have the following meanings:

- <Var 1> Refers to an R parameter which records the results of the calculation of intersection with two additional R parameters following in immediate succession.
- <Var 2> Refers to an R parameter which contains the input data for the calculation of intersection with seven additional R parameters following in immediate succession.
If @710 is combined with @711, <Var 1> agrees with <Var 2> from the reference point edit.
- <Var 3> This identifier is not required. However, it must be assigned an R parameter.

@713 <Var>

This command loads the numerical value which corresponds to the safety clearance of 1 mm in the current input format in the R parameter defined with <Var>. The numerical value 1 is loaded in the immediately following R parameter in the case of radius programming and the numerical value 2 in the case of diameter programming. Parameter range: R0 to R98 and R900 to R998.

@714

With the command @714, block preparation (decoding) is stopped until the buffer memory is empty.

When the program is being executed, several blocks are decoded in advance in the control and loaded in the buffer memory of the NC. As a result, program execution is faster, but in combination with certain NC commands (reading of actual values, measurements, data transfer NC-PLC) can lead to incorrect execution of the program. By means of the command @714 (stop decoding), the advance decoding of NC blocks written after this command is stopped until the block containing the command @714 has been executed. This means that the buffers are emptied and information required for subsequent NC blocks is available,

The @714 instruction must be programmed for the following information from the interface control, provided it is required in the next NC blocks:

- Machine data
- Setting data
- Toll offsets
- Zero offsets
- R parameters
- "Mirror" signal

The command @714 must be programmed after each measurement. This command must also be programmed before blocks in which zero offsets and tool offsets are written, if the new values are to be active from this block.

The command @714 must always be given in a separate NC block.

Example:

<pre>M94 L_F @714 L_F @123 R60 K100 K5 L_F</pre>	<p>The part number is transferred from the PLC in R60. Stop decoding so that the current part number can be evaluated in the next NC block.</p> <p>Branch in accordance with part number</p>
---	--

Subgroup 2: Measuring functions

@720 <Var> <Value >

This command is used in measuring cycles. It stores the axis positions referred to machine zero starting from the R parameter defined with <Var>. The number of the measuring input (1 or 2) is determined under <Value>.

Example:

The actual values of the X and Y axes are stored in R5 and R6 when probe 1 is actuated.

```

.
.
.
N15 G90 X50 Y75 @720 R5 K1 L_F
.
.
.
```


11.3 @ code table

@ code	CL-800 instruction	Function
@040 (Const) (R Par 1) . . . (R Par n)	(Push) ¹⁾	Save the specified local R parameters to the stack
@041 R- Par 1 R Par 2	(Push Block) ¹⁾	Save a group of local R parameters to the stack
@042 Const R Par n . . . R Par 1	(Pop) ¹⁾	Fetch the saved R parameters from the stack
@043 R- Par 1 R Par 2	(Pop Block) ¹⁾	Fetch a group of the saved R parameters from the stack
@100 Const @100 R Par ⁴⁾	GOTO Label ;	Absolute jump to NC block
@111 Var Value 1 Const 1 Value 2 Const 2 . . Value n Const n	CASE Var = Value 1 : Instruction 1 ; . . = Value n : Instruction n ;	Case branch
@12x Var Value Const	IF "Condition" ²⁾ THEN Instruction 1 ; [ELSE Instruction 2 ;] END IF;	IF-THEN-ELSE instruction x comparison operator vop Var R parameter or pointer
@13x Var Value Const	WHILE "Condition" ²⁾ DO Instruction ;	Repetition instruction with enquiry for repetition condition at the beginning. x comparison operator vop
@14x Var Value Const	REPEAT Instruction ; UNTIL "Condition"; ²⁾	Repetition instruction with enquiry for repetition condition at the end. x comparison operator vop

Explanation of symbols:

- x comparison operator vop**
- 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

- 1) Not at CL800 level
- 2) "Condition": a) Var =Boolean variable
b) Var . Const =Bit from pattern
c) Var "vop" Value
d) Extended condition

3) Option 

4) No pointers possible,
on CL 800 level only Const can be specified.

@ code	CL-800 instruction	Function
@151 <i>Var Value Const</i>	FOR <i>Var</i> = <i>Value 1</i> TO <i>Value 2</i> DO Instruction ;	Repetition instruction with repetitions until <i>Var</i> has reached <i>Value</i> incrementally
@161 <i>Var Value Const</i>	FOR <i>Var</i> = <i>Value 1</i> DOWN TO <i>Value 2</i> DO Instruction ;	Repetition instruction with repetitions until <i>Var</i> has reached <i>Value</i> decrementally
@200 <i>Var</i>	CLEAR(<i>Var</i>);	Delete variable
@201 <i>Var Value</i>	<i>Var</i> = <i>Value</i>	Load variable with value
@202 <i>Var 1 Var 2</i>	XCHG (<i>Var 1</i> , <i>Var 2</i>);	Exchange the variable contents ³⁾
@203 <i>Var 1 Var 2 Const</i>		Read a bit from bit pattern ³⁾
@300 <i>Var Value 1</i>	<i>Var</i> =MDN (<i>Value1</i>);	Machine data NC Value 1: Addr. 0 . . . 4999
@301 <i>Var Value 1</i>	<i>Var</i> =MDNBY (<i>Value1</i>);	Machine data NC bytes Value 1: Byte addr. 5000 . . . 6999
@302 <i>Var Value 1 Value 2</i>	<i>Var</i> =MDNBI (<i>Value1</i> , <i>Value2</i>);	Machine data NC bits Value 1: Byte addr. 5000 . . . 6999 Value 2: Bit addr. 0 . . . 7
@306 <i>Var Value 1</i>	<i>Var</i> =MDP (<i>Value1</i>);	Machine data PLC Value 1: Addr. 0 . . . 1999
@307 <i>Var Value 1</i>	<i>Var</i> =MDPBY (<i>Value1</i>);	Machine data PLC bytes Value 1: Byte addr. 6000 . . . 3999
@308 <i>Var Value 1 Value 2</i>	<i>Var</i> =MDPBI (<i>Value 1</i> , <i>Value 2</i>);	Machine data PLC bits Value 1: Byte addr. 6000 . . . 3999 Value 2: Bit addr. 0 . . . 7

Explanation of symbols:

x comparison operator vop

- 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

- 1) Not at CL800 level
 2) "Condition": a) *Var* =Boolean variable
 b) *Var* . *Const* =Bit from pattern
 c) *Var* "vop" *Value*
 d) Extended condition

3) Option 4) No pointers possible,
on CL 800 level only *Const* can be specified.


@ code	CL-800 instruction	Function
@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 bytes Value 1: Byte addr. 5000 . . . 9999
@312 Var Value 1 Value 2	Var =SENBI (Value 1 , Value 2);	Setting data NC bits Value 1: Byte addr. 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: 1 to 8 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 to 4 Value 3: 0/1
@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 to 4
@332 Var Value 2	Var =ZOE (Value 2);	External offset from PLC Value 2: Axis No. 1 to 4
@333 Var Value 2	Var =ZOD (Value 2);	DRF offset Value 2: Axis No. 1 to 4
@334 Var Value 2	Var =ZOPS (Value 2);	PRESET offset Value 2: Axis No. 1 to 4
@336 Var Value 2	Var =ZOS (Value 2);	Cumulative offset Value 2: Axis No. 1 to 4

Explanation of symbols:

x comparison operator vop

- 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

- 1) Not at CL800 level
- 2) "Condition": a) Var =Boolean variable
b) Var . Const =Bit from pattern
c) Var "vop" Value
d) Extended condition

3) Option 


4) No pointers possible,
on CL 800 level only Const can be specified.

@ code	CL-800 instruction	Function
@342 Var Value 1 Value 3	Var =PRSS (Value 1 , Value 2);	Programmed spindle speed Value 1: 0 Value 3: 0
@345 Var Value 1 Value 2	Var =PRVC (Value 1 , Value 2);	Programmed cutting rate Value 1: 0 Value 2: 0
@360 Var Value 2	Var =ACPW (Value 2);	Axis position actual referred to workpiece Value 2: Axis No. 1 to 4
@361 Var Value 2	Var =ACPM (Value 2);	Axis position actual referred to machine Value 2: Axis No. 1 to 4
@363 Var Value 2	Var =ACSP (Value 2);	Spindle position actual value Value 2: Spindle No. 1
@364 Var Value 2	Var =ACSS (Value 2);	Spindle speed actual value Value 2: Spindle No. 1
@367 Var Value 1	Var =ACAS (Value 1);	Axis and spindle number of current plane Value 1 = 0
@36a Var Value 1	Var =ACD (Value 1);	D function actual Value 1 = 0
@36b Var Value 1 Value 3	Var =ACG (Value 1 , Value 3)	Current G function Value 1 = 0 Value 3 = Internal G group
@3e4 Var Value 1	Var =AGS (Value 1);	Active gear stage Value 1: 0
@371 Var Value 1 Value 3	Var =SOB (Value 1 , Value 3);	Special bits Value 1 = 0 Value 3: Bit No. 0 . . . 7

Explanation of symbols:

x comparison operator vop


- 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

- 1) Not at CL800 level
2) "Condition": a) Var =Boolean variable
b) Var . Const =Bit from pattern
c) Var "vop" Value
d) Extended condition
3) Option 
4) No pointers possible,
on CL 800 level only Const can be specified.

@ code	CL-800 instruction	Function
@400 Value 1 Value	MDN (Value 1)= Value ;	Machine data NC Value 1: Addr. 0 . . 4999
@401 Value 1 Value	MDNBY (Value 1)= Value ;	Machine data NC bytes Value 1: Byte addr. 5000 . . . 6999
@402 Value 1 Value 2 Value	MDNBI (Value 1 , Value2) = Value ;	Machine data NC bits Value 1: Byte addr. 5000 . . . 6999 Value 2: Bit addr. 0 . . . 7
@406 Value 1 Value	MDP (Value 1)= Value ;	Machine data PLC Value 1: Addr. 0 . . 5999
@407 Value 1 Value	MDPBY (Value 1)= Value ;	Machine data PLC bytes Value 1: Byte addr. 2000 . . . 3999
@408 Value 1 Value 2 Value	MDNBI (Value 1 , Value 2) = Value ;	Machine data PLC bits Value 1: Byte addr. 2000 . . . 3999 Value 2: Bit addr. 0 . . . 7
@410 Value 1 Value	SEN (Value1)= Value ;	Setting data NC Value 1: Addr. 0 . . 4999
@411 Value 1 Value	SENBY (Value 1)= Value ;	Setting data NC bytes Value 1: Byte addr. 5000 . . . 9999
@412 Value 1 Value 2 Value	SENBI Value 1 , Value2 = Value ;	Setting data NC bits Value 1: Byte addr. 5000 . . . 9999 Value 2: Bit addr. 0 . . . 7
@420 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 . . . 9

Explanation of symbols:

- x **comparison operator vop**
- 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

- 1) Not at CL800 level
- 2) "Condition":
 - a) Var =Boolean variable
 - b) Var . Const =Bit from pattern
 - c) Var "vop" Value
 - d) Extended condition
- 3) Option 
- 4) No pointers possible,
on CL 800 level only Const can be specified.


@ code	CL-800 instruction	Function
@423 <i>Value 1 Value 2</i> <i>Value 3 Value</i>	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 . . . 9
@430 <i>Value 1 Value 2</i> <i>Value 3 Value</i>	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 to 4 Value 3: 0 / 1 (coarse/fine)
@431 <i>Value 1 Value 2</i> <i>Value 3 Value</i>	ZOFA (Value 1 , Value 2 , Value 3)= Value ;	Settable zero offset additive Value 1: Group 1 . . . 4 (G54-G57) Value 2: Axis No. 1 to 4 Value 3: 0 / 1 (coarse/fine)
@432 <i>Value 1 Value 2</i> <i>Value</i>	ZOPR (Value 1 , Value 2) = Value ;	Programmable zero offset (G58,G59) Value 1: Group1 or 2 (G58 or G59) Value 2: Axis No. 1 to 4
@434 <i>Value 2 Value</i>	ZOD (Value 2)= Value ;	DRF offset Value 2: Axis No. 1 to 4
@435 <i>Value 2 Value</i>	ZOPS (Value 2)= Value ;	PRESET offset Value 2: Axis No. 1 to 4

Explanation of symbols:

x comparison operator vop

- 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

- 1) Not at CL800 level
- 2) "Condition":
 - a) Var =Boolean variable
 - b) Var . Const =Bit from pattern
 - c) Var "vop" Value
 - d) Extended condition


- 3) Option 

- 4) No pointers possible,
on CL 800 level only Const can be specified.

@ code	CL-800 instruction	Function
@440 Value 3 Value	ZOPS (Value 3)= Value ;	Programmed axis position Value 3: Axis No. 1 to 4
@442 Value 3 Value	PRSS (Value 3)= Value ;	Programmed spindle speed Value 3: 0
@446 Value	PRAD= Value ;	Programmed radius
@447 Value	PANG= Value ;	Programmed angle
@448 Value 3 Value	PRIP (Value 3)= Value ;	Interpolation parameters Value 3: Axis No. 1 to 4
@4c0 Value	ALNC ()= Value ;	Cycle alarms Value= Alarm No. 4000 . . . 4299 5000 . . . 5099
@4e1 Value 1 Value 2 Value	SATC (Value 1 , Value 2) = Value ;	Spindle acceleration time constant
Var = Value 1 + Value 2 ; Var = Value 1 - Value 2 ; Var = Value 1 * Value 2 ; Var = Value 1 / Value 2 ;	Var = Value 1 + Value 2 ; Var = Value 1 - Value 2 ; Var = Value 1 * Value 2 ; Var = Value 1 / Value 2 ;	Addition Subtraction Multiplication Division
@610 Var Value	Var =ABS (Value) ;	Forming absolute value
@613 Var Value	Var =SQRT (Value) ;	Square root
@614 Var Value 1 Value 2	Var =SQRTS (Value 1 , Value 2) ;	Root of sum of squares
@620 Var	INC (Var) ;	Increment "Var" by 1

Explanation of symbols:

- x **comparison operator vop**
- 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

- 1) Not at CL800 level
- 2) "Condition":
 - a) Var =Boolean variable
 - b) Var . Const =Bit from pattern
 - c) Var "vop" Value
 - d) Extended condition
- 3) Option 
- 4) No pointers possible,
on CL 800 level only Const can be specified.

@ code	CL-800 instruction	Function
@ 621 <i>Var</i>	DEC (<i>Var</i>);	Decrement "Var" by 1
@ 622 <i>Var</i>	TRUNC (<i>Var</i>);	Integral part
@ 630 <i>Var Value</i>	<i>Var</i> = SIN (<i>Value</i>);	Sine
@ 631 <i>Var Value</i>	<i>Var</i> = COS (<i>Value</i>);	Cosine
@ 632 <i>Var Value</i>	<i>Var</i> = TAN (<i>Value</i>);	Tangent
@ 634 <i>Var Value</i>	<i>Var</i> = ARC SIN (<i>Value</i>);	Arc sine
@ 637 <i>Var Value 1 Value 2</i>	<i>Var</i> = ANGLE (<i>Value 1</i> , <i>Value 2</i>);	Angle from two vector components
@ 640 <i>Var Value</i>	<i>Var</i> = LN (<i>Value</i>);	Natural logarithm
@ 650 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> OR <i>Value</i> ;	OR
@ 651 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> XOR <i>Value</i> ;	EXCLUSIVE OR
@ 652 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> AND <i>Value</i> ;	AND
@ 653 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> NAND <i>Value</i> ;	NAND
@ 654 <i>Var Value</i>	<i>Var</i> = NOT <i>Value</i> ;	NOT
@ 655 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> ORB <i>Value</i> ;	OR bit
@ 656 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> XORB <i>Value</i> ;	EXCLUSIVE OR bit
@ 657 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> ANDB <i>Value</i> ;	AND bit
@ 658 <i>Var Var 1 Value</i>	<i>Var</i> = <i>Var 1</i> NANDB <i>Value</i> ;	NAND bit
@ 659 <i>Var Value</i>	<i>Var</i> = NOTB <i>Value</i>	NOT bit
@ 660 <i>Var Const</i>	CLEAR BIT (<i>Var</i> · <i>Const</i>);	Clear bit in pattern Const=Bit No. 0 . . . 7
@ 661 <i>Var Const</i>	SET BIT (<i>Var</i> · <i>Const</i>);	Set bit ; Const=Bit No. 0 . . . 7

Explanation of symbols:

x comparison operator vop

- 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

- 1) Not at CL800 level
 2) "Condition": a) *Var* = Boolean variable
 b) *Var* · *Const* = Bit from pattern
 c) *Var* "vop" *Value*
 d) Extended condition


- 3) Option 

- 4) No pointers possible,
 on CL 800 level only *Const* can be specified.

@ code	CL-800 instruction	Function
@67x Var 1 Var 2 Value		If the comparison between Var 2 and Value has been satisfied, the boolean variable Var 1 is set to 1.
@706	POS MSYS;	Specification of a position referred to the machine actual value system.
@710 Var 1 Var 2	Var 1 =PREP REF (Var 2);	Reference preparation Var 1: Output data starting at Var 1 Var 2: Input data starting at Var 2
@711 Var 1 Var 2 Var 3	Var 1 =INT SEC (Var 2 Var 3);	Intersection calculation Var 1: Output data starting at Var 1 Var 2: First contour starting at Var 2 Var 3: Assign 0
@713 Var	Var =PREP CYC;	Start preparation for cycles Var: Output data starting at Var
@714	STOP DEC;	Stop decoding; until buffer is empty.
@720 Var Value	Var =MEAS M Value	Value measurement Var: data stored starting at Var Value: No. of measurement inputs; 1 or 2

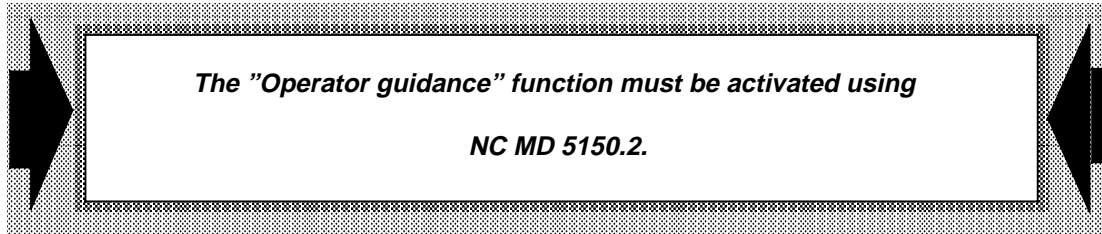
Explanation of symbols:

- x **comparison operator vop**
- 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

- 1) Not at CL800 level
- 2) "Condition":
 - a) Var =Boolean variable
 - b) Var . Const =Bit from pattern
 - c) Var "vop" Value
 - d) Extended condition
- 3) Option 
- 4) No pointers possible, on CL 800 level only Const can be specified.

12 Extended Programming Functions

12.1 Operator guidance



Application

The "Operator guidance" function assists the user with the parameterization of Siemens standard cycles as well as his own user cycles.

Any subroutine which can be parameterized using R parameters (also called cycle) can be provided with operator guidance.

When a subroutine is called (in edit or in MDA mode), the OPERATOR GUIDANCE softkey can be pressed to display an overview of the subroutines containing the operator guidance function.

The desired subroutine is now selected using the cursor or the SEARCH softkey. Operating the PARAMETER softkey opens a window containing parameterization help for the selected subroutine.

A maximum of 7 R parameters with accompanying texts can be displayed in the window at any one time. If more than 7 R parameters are to be set, they can be brought into the display by pressing the page keys.

The user now enters the values for the listed parameters and stores them together with subroutine call by operating the ACCEPT PARAMETER softkey.

Operator guidance generation

The user can generate the operator guidance texts as well as the input limitations for the R parameters himself either directly through the operator keyboard or externally using a programmer.

In order to do this, he must generate a declaration block for each cycle which is to be provided with operator guidance in a subroutine (the number of which must be entered in NC MD 32).

Such a subroutine consists of as many declaration blocks as programs to be provided with operator guidance.

A declaration block should look like this:

Initial identifier:

//L904 (text, max. 38 characters) L_F

Cycle number to be provided with operator guidance. The subsequent text then appears in the comment line of the subroutine overview in the operator guidance window.

R parameter list:

The R parameter list can contain a maximum of 50 R parameters. The three possibilities listed below can be used as often as required.

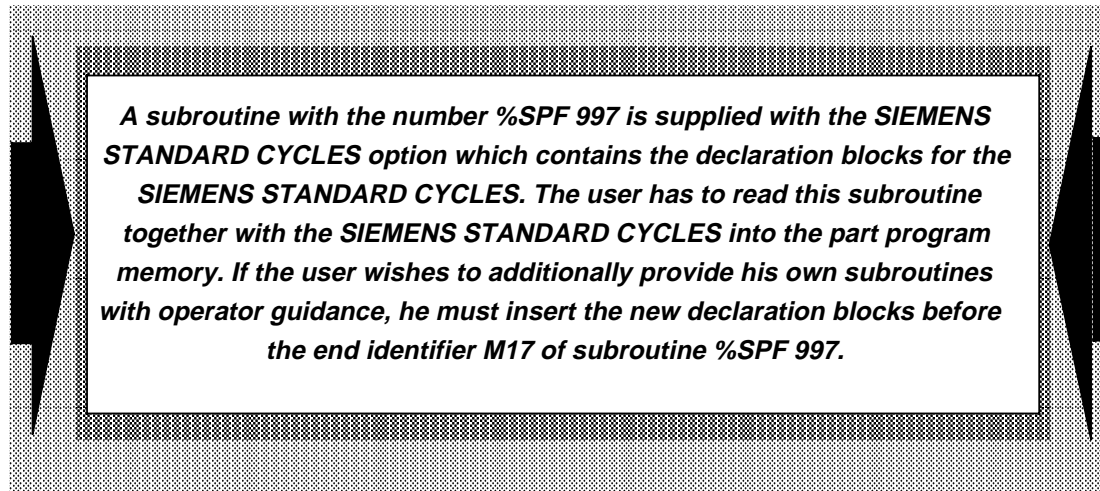
(R10/*527 324,434/text*, max. 30 characters) L_F Details of R parameter with upper and lower input limits (separated by blank). The text is displayed in the window as the R parameter explanation.

(R11/**3 7 16 23/text*, max. 30 characters) L_F Details of R parameter with possible input values (max. 20 whole figure values, separated by blanks). The text is displayed in the window as the R parameter explanation.

(R12 *text* max. 30 characters) L_F Details of R parameter without input limits, the text is displayed in the window as the R parameter explanation.

Sign at end:

L_F The end of the declaration block is characterized by a space line.



Example:

The following subroutine %SPF 73 to be assigned with parameters is to be provided with operator guidance:

```
%SPF 73
N5 G90 G00 Z=R12 LF (Approach reference plane)
N10 G01 Z=R13 F=R14 LF (Approach drilling depth)
N15 G00 Z=R12 LF (Approach reference plane)
N20 M17 LF
```

The declaration block for this subroutine must also be inserted in the subroutine %SPF 997 (if standard cycles are available).

%SPF 997

(Operator guidance texts) L_F

Siemens standard cycle declaration blocks

<pre>//L73 (drill hole, standard) L_F (R12 / 0 999999 /reference plane) L_F (R13 drilling depth) L_F (R13 / 0 10000 /drilling feed) L_F L_F</pre>	}	Declaration block for subroutine %SPF 73
---	---	---

M17

The window which appears as the parameterization help for subroutine %SPF 73 looks like this:

L73 Drilling hole, standard	
Reference plane	
Drilling depth	
Drilling feed	

The following input limitations are valid:

- R12, reference plane: permitted range from 0 to 999999
- R13, drilling depth: no input limit
- R14, drilling feed: permitted range from 0 to 10000

If any other values are entered, the warning message "input value too large" or "input value too small" appears when the information is to be saved.

12.2 User screen



Application

By means of the user screen function, the user can configure a screen containing display and input fields for selected R parameters. This screen is always displayed in the main menu of the data area (if configured). Within this screen it is possible to select various R parameter screenforms (e.g. for different part programs). Thus, the most important data can be checked at a glance and can be modified, if required.

The user screen is divided in two areas:

- On the left-hand side, the different R parameter screenforms with their identifiers are stored in the form of a table. If more than 10 identifiers are configured in this table, they can be displayed by means of the cursor or page keys.
- On the right-hand side, the R parameter screenform selected by means of the identifier is displayed (the identifier as title). The screenform comprises the R parameter texts and the associated fields with values. A maximum of 13 R parameters can be displayed at any one time. When configuring, limit values can be defined for the R parameter contents (e.g. maximum number of subroutines or temperature of the tank). The values during the program run must not exceed or fall below these limit values. If this happens, the relevant R parameter flashes. Flashing is not active if the cursor is in the right-hand part of the user screen. Values can be entered or changed in the R parameter fields. The desired value field is selected via the cursor keys; then editing can be started as described in the Operator's Guide. Editing of R parameters can be interlocked with the key switch (R parameter area selectable via machine data).

Creating the user screen

The user can create the user screen with its R parameter screenforms directly via the operator keyboard or externally on a programmer. For this purpose, he must create a declaration block for each R parameter screenform in a subroutine (whose number must be entered in NC MD 33).

Thus, the number of declaration blocks in a subroutine equals to the number of R parameter screenforms.

A declaration block must have the following structure:

Sign at beginning:

//P1 (text, max. 110 characters) L_F

P number of the R parameter screenform (any number with up to 4 digits). The text that follows appears in the left field of the user screen as identifier (max. number of displayed characters: 25) and in the right part of the screenform as title (max. number of characters displayed: 39).

R parameter list:

The R parameter list can contain a maximum of 13 R parameters. The text of the R parameter declaration is displayed with a maximum of 30 characters.

The three possibilities listed below can be used as often as required.

(R10/*527 324,434*/text, max. 30 characters) L_F Details of R parameter with upper and lower input limits (separated by blank). The text is displayed in the window as the R parameter explanation.

(R11/**3 7 16 23*/text, max. 30 characters) L_F Details of R parameter with possible input values (max. 20 whole figure values, separated by blanks). The text is displayed in the window as the R parameter explanation.

(R12 text max. 30 characters) L_F Details of R parameters without input limits, the text is displayed in the window as the R parameter explanation.

Sign at end:

L_F The end of the declaration block is identified by a blank line.

Example:

Information in the form of R parameters is to be displayed for the program machining the part "Pivot bearing 254".

The following is to be displayed:

- Number of the parts to be machined
- Parts already machined (message when >20)
- Parts yet to be machined
- Oil pressure in percent (message when <80%)
- Actual machining time
- Average machining time

Contents of the subroutine containing the declaration blocks for the user screen (must be entered in NC MD 33; here as example SPF 996):

```
%SPF 996
(user screen texts)
:
other declaration blocks
:
//P22 (PIVOT BEARING 254)LF
(R702 SPECIFIED NUMBER OF PARTS)LF
(R705 /0 20/ MACHINED PARTS)LF
(R740 REMAINING PARTS)LF
(R773 /80 100/ OIL PRESSURE IN %LF
(R729 ACTUAL MACHINING TIME IN MIN)LF
(R730 AVERAGE MACHINING TIME IN MIN)LF
LF
:
other declaration blocks
:
M17
```

The screenshot displays the SINUMERIK 805 user interface. At the top, there is a row of control buttons: MDA, SKP, DRY, ROV, DBL, DRF, M01, FST, and %0. Below this is a bar with a RESET button. The main area is divided into two columns. The left column contains a box with 'SINUMERIK 805' and a list of shaft and eccentric numbers. The right column is titled 'PIVOT BEARING' and contains a table with machining statistics. At the bottom, there is a row of function buttons: PROGRAM, DATA PARAMETERS, TOOL OFFSET, ZERO OFFSET, DATA TRANSFER, and DIAGNOSTICS.

PIVOT BEARING	
SPECIFIED NUMBER OF PARTS	110
MACHINED PARTS	13
REMAINING PARTS	97
OIL PRESSURE IN %	100
CURRENT MACINING TIME IN MIN.	19.8
AVERAGE MACHINING TIME IN MIN.	23.6

SHAFT NO. 23
SHAFT NO. 40
SHAFT NO. 41
RAM MOTOR
ECCENTRIC NO. 12
ECCENTRIC NO. 13
GEAR SHAFT NO. 235
PART OF SHAFT NO. 213
SHADT NO. 42

13 Program Key

13.1 Internal breakdown of G groups with @36b

Internal G group	G functions														
0	00	01	02	03	10	11	12	13	33	34	35				
1	09														
2	17	18	19	16											
3	40	41	42												
4	53														
5	54	55	56	57											
6	04	25	26	58	59	92									
7	60	62	63	64											
8	70	71													
9	80	81	82	83	84	85	86	87	88	89					
10	68	90	91												
11	94	95	96	97											
12	48	110	111	147	247	347	148	248	348						
13															
14															
15															

13.2 Program key SINUMERIK 805

Group	EIA	ISO	Code	Section	Function and meaning
	EOR	%		1.1	Beginning of program
	mpf ... spf EOB	MPF ... SPFLF	1 to 9999 1 to 999	1.6	Main program Subroutine Block end
	o n /o /n	: N /: /N	1 to 9999	1.2.1 1.2.2	Main block Subblock Skippable main block Skippable subblock
G0	g	G	00 01 + 02 03 10 11 12 13 33 34 35	3.1.1 3.2.1 3.2.2 3.2.4 3.2.7	Rapid traverse, exact stop coarse Linear interpolation Circular interpolation clockwise Circular interpolation counter-clockwise Polar coordinate programming, rapid traverse Polar coordinate programming, linear interpolation Polar coordinate programming, circular interpolation clockwise Polar coordinate programming, circular interpolation counter-clockwise Thread cutting with constant lead Thread cutting with linear increasing lead Thread cutting with linear decreasing lead
G1	g	G	09 #	3.2.9	Speed reduction, exact stop fine
G2	g	G	16 + 3) 17 18 19	3.2.11	Plane selection with free choice of axis (reset position see machine data) Plane selection X - Y Plane selection Z - X Plane selection Y - Z
G3	g	G	40 + 41 42	9	no CRC/TNRC CRC/TNRC counter-clockwise CRC/TNRC clockwise
G4	g	G	53 #	2.5	Suppression of zero offset
G5	g	G	54 + 55 56 57	2.5	Zero offset 1 Zero offset 2 Zero offset 3 Zero offset 4
G6	g g g g	G G G G	04# 3) 25 # 3) 26 # 3) 58 # 3) 59 # 3) 92 + 3)	3.2.10 2.9 2.5 4.3	Dwell time specified under address X or F in seconds and address S in spindle revolutions Minimum working area limitation Maximum working area limitation Programmable additive zero offset 1 Programmable additive zero offset 2 Spindle setpoint speed limitation under address S
G7	g	G	60 + (with M) 62 63 64 + (with T)	3.2.9	Speed reduction, exact stop fine Continuous path mode, block transition with speed reduction Tapping without encoder, feedrate override 100 % Continuous path mode, block transition without speed reduction

Group	EIA	ISO	Code	Section	Function and meaning
G8	g	G	70 71	2.7	Input system inch Reset position via machine data Input system metric Reset position via machine data
G9	g	G	80 81 82 83 84 85 86 87 88 89	10	Cancel G81 to G89 Call cycle L81 - Boring, centering only if option "standard cycles" is available
G10	g	G	90 * 91 68	2.3 2.3.2	Absolute position data Incremental position data Positioning rotary axes by the shortest path
G11	g	G	94 * 95 96 97	3.2.6	Feedrate F in mm/min Feedrate F in mm/rev or inch/rev Constant cutting speed S in m/min oder ft/min Cancellation of constant cutting speed and storage of last set speed of G96
G12	g	G	110 * 111* 147# 3) 247# 3) 347# 3) 148# 3) 248# 3) 348# 3) 48# 3)	3.2.5 3.2.12	Programming of polar coordinates, adopt the setpoint reached as the new centre point Programming of polar coordinates, centre-point programming with angle and radius without axis movement Soft approach to contour linear Soft approach to contour in quarter circle Soft approach to contour in semicircle Soft exit linear Soft exit in quarter circle Soft exit in semicircle Exit the contour as it was approached
	x	X	± 0.001 to± 99999.999 ± 0.0001 to± 3999.9999 0.001 to 99999.999 ± 0.001° to± 99999.999°	3.2	Positional data in mm Positional data in inch Dwell time in s Positional data in degrees
	y	Y	± 0.001 to± 99999.999 ± 0.0001 to 3999.9999 ± 0.001° to± 99999.999°	3.2	Positional data in mm Positional data in inch Positional data in degrees
	z	Z	± 0.001 to± 99999.999 ± 0.0001 to± 3999.9999 ± 0.001° to± 99999.999°	3.2	Positional data in mm Positional data in inch Positional data in degrees
	4.	4.	± 0.001 to± 99999.999 ± 0.0001 to± 3999.9999 ± 0.001° to± 99999.999°	3.2	Positional data in mm Positional data in inch Positional data in degrees Possible addresses, A, B, C, E, Q, U, V, W
	a 2)	A 1)	0 to 359.99999° 0 to 359.99999°	3.2.2.2	Angle in degrees with contour definition Angle in degrees with polar coordinates
	u 2)	U 1)	+ 0.001 to+ 99999.999 + 0.0001 to+ 3999.9999 0 - 0.001 to - 99999.999 - 0.0001 to - 3999.9999 + 0.001 to+ 99999.999 + 0.0001 to+ 3999.9999	3.2.2.2	Radius with circular interpolation and polar coordinates in mm Radius with circular interpolation and polar coordinates in inch Corner with contour definition Chamfer with contour definition in mm Chamfer with contour definition in inch Radius with contour definition in mm Radius with contour definition in inch

13.2 Program key SINUMERIK 805

Group	EIA	ISO	Code	Section	Function and meaning
	i	I	± 0.001 to ± 99999.999 ± 0.0001 to ± 3999.9999 0.001 to 400.000 0.0001 to 16.000	3.2.2.1	Interpolation parameter for X axis in mm Interpolation parameter for X axis in inch Thread lead in mm Thread lead in inch
	j	J	± 0.001 to ± 99999.999 ± 0.0001 to ± 3999.9999 0.001 to 400.000 0.0001 to 16.000	3.2.2.1	Interpolation parameter for Y axis in mm Interpolation parameter for Y axis in inch Thread lead in mm Thread lead in inch
	k	K	± 0.001 to ± 99999.999 ± 0.0001 to ± 3999.9999 0.001 to 400.000 0.0001 to 16.000	3.2.2.1	Interpolation parameter for Z axis in mm Interpolation parameter for Z axis in inch Thread lead in mm Thread lead in inch
	d	D	1 to 99 0	8	Select tool offset Deselect tool offset
	p	P	1 to 99	5	Number of subroutine passes
	r	R	0 to 49 50 to 99 100 to 699 700 to 999	6	Transfer parameters Arithmetic parameters Reserved for Siemens (user assignable in some cases) Parameters user assignable
	f	F	0.01 to 60000 0.1 to 2360.000 0.001 to 99.999 0.001 to 400.000 0.0001 to 16.0000 0.001 to 16.000 0.0001 to 0.6000	3.2.5	Feedrate in mm/min, m/min Feedrate in inch/min Dwell in s Feedrate in mm/rev Feedrate in inch/rev Thread lead increase or decrease in mm/rev Thread lead increase or decrease in inch/rev
	s	S	1 to 12000 1 to 12000 0.5 to 359.9 0.1 to 99.9	4.3	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, distance from zero mark of encoder Dwell time in revolutions
	t	T	1 to 9999	4.5	Tool number
	h	H	1 to 9999	4.4	Auxiliary functions
	l	L	1 to 999	5	Subroutine number
M1	m	M	00 # 01	4.2	Programmed stop unconditional Programmed stop conditional
M2	m	M	02 17 30	4.2	End of program, contained in last block of program End of subroutine, contained in last block of subroutine, without stop for repeat passes End of program, contained in last block of program
M3	m	M	03 04 05 * 19# 3)	4.2	Spindle rotation clockwise Spindle rotation counter-clockwise Spindle stop without orientation Oriented spindle stop, angle under address S in degrees

Group	EIA	ISO	Code	Section	Function and meaning
M4	m	M	36 37	3.2.6	Feedrate as programmed under F Feedrate in mm/min, m/min or mm/rev reduced by 1:100 also effective with G33
M5	m	M	0 to 99	4.2	Miscellaneous functions, freely assignable except groups M1 to M4
	@	@		11	Special function
	=	=		6	Separation sign, e.g. R35 = 123.5, Q1 = 1234.567
	+	+		6	Addition with parameters
	-	-		6	Subtraction with parameters
	*	*		6	Multiplication with parameters
	/	/		6	Division with parameters
	5-4-2 ²) 7-4-2 ²)	()		1.2.3	Beginning of comment End of comment
	EOB	LF		1.1	End of block

Explanation of symbols:

- * Delete position (basic position after reset, M02/M30, after switching on the control)
- # Effective block by block, all other modal
- 1) Other address can be selected (A, B, C, E, U, V, W)
- 2) Punched tracks
- 3) No further functions must be written in this block

Siemens AG

AUT V250
P.O. Box 31 80
D-91050 Erlangen
Federal Republic of Germany

Suggestions

Corrections

For Publication/Manual:

SINUMERIK 805
Software Version 4
Programming Guide

User Documentation

Order No.: 6ZB5 410-0EK02-0BA3
Edition: May 1993

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

This document was printed on paper
bleached using an environmentally
friendly chlorine-free method.

© Siemens AG 1990 All Rights Reserved
Subject to change without prior notice

Order No. 6ZB5 410-0EK02-0BA3
Printed in the Fed. Rep. of Germany

