SIEMENS

SINUMERIK 840C
Software Version 2, 3, 4, 5 and 6

Programming Guide **09.2001 Edition**

User Documentation

SINUMERIK 840C
Software Version 2, 3, 4, 5 and 6

Programming Guide

Applies to:

Control	*Software Version*
SINUMERIK 840C*/*CE (Standard*/*Export version)	2, 3, 4, 5, 6

09.2001 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
11.92	6FC5198-0AA10-1BP0	**A**
06.93	6FC5198-2AA10-0BP0	**C**
12.93	6FC5198-3AA10-0BP0	**C**
10.94	6FC5198-4AA10-0BP0	**C**
09.95	6FC5198-5AA10-0BP0	**C**
04.96	6FC5198-5AA10-0BP1	**C**
08.96	6FC5198-5AA10-0BP2	**C**
07.97	6FC5198-6AA10-0BP0	**C**
01.99	6FC5198-6AA10-0BP1	**C**
09.01	6FC5198-6AA10-0BP2	**C**

This manual is included in the documentation on CD-ROM (**DOCONCD**)

Edition	Order No.	Remarks
10.01	6FC5198-6CA00-0BG2	C

Trademarks

SIMATIC®, SIMATIC HMI®, SIMATIC NET®, SIROTEC®, SINUMERIK® and SIMODRIVE® are trademarks of Siemens AG. All other product and system names are registered trademarks of their respective companies and must be treated accordingly.

Further information can be found on the Internet under http:**//**www.ad.siemens.de**/**sinumerik

This publication was produced on the Siemens 5800 Office System.

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.

We have checked that the contents of this publication agree with the hardware and software described herein. The information given in this publication is reviewed at regular intervals and any corrections that might be necessary are made in the subsequent printings. Suggestions for improvement are welcome at all times.

Subject to change without prior notice.

Order No. 6FC5197-6AA10-0BP2
Printed in the Federal Republic of Germany

Siemens-Aktiengesellschaft

Preliminary Remarks

Notes for the Reader

The SINUMERIK 840C documentation is divided into four levels:

- General documentation
- User documentation
- Manufacturer documentation and
- Service documentation

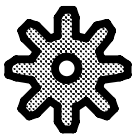
This Programming Guide is for the SINUMERIK 840C control and is part of the user documentation.

This Guide offers you specific technological information on the extended DIN and @ programming.

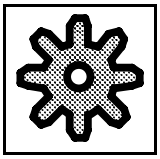
You will learn

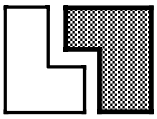
- to create part programs and
- to program motion blocks, switching, auxiliary and miscellaneous functions.

For further information on other SINUMERIK 840C publications and on publications which apply for all SINUMERIK controls (e.g. CL800 Cycle Language), please get in touch with your Siemens local branch office or Siemens national company.

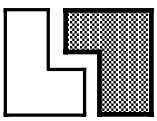


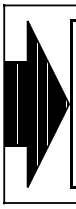
This symbol indicates that the machine tool manufacturer can change the described function by altering a machine data (MD).





This symbol indicates that the described function is only operable if the control contains the option shown. Please refer to the SINUMERIK 840C NC catalog for the order numbers of the options.





This symbol indicates important information for the reader.

Technical Comments

This Programming Guide describes the programming possibilities and the extent of the standard functions of the SINUMERIK 840C control. The stated maximum values are limiting values and can normally be limited by making machine-specific changes (machine data).

Functions which go beyond the functions described in the SINUMERIK 840C documentation but which might nonetheless be executable in the control, are not subject to guarantees, compatibility and service from SIEMENS AG.

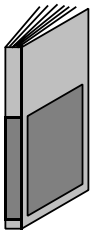
For simplicity, preparatory functions have been programmed in the examples even if they are basic settings.

Explanations regarding program syntax:
The contents of pointed brackets < > must be programmed.
The contents of curly brackets { } can be programmed although this is not obligatory.

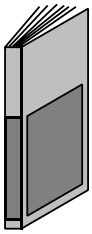
The program examples have been written in ISO code.

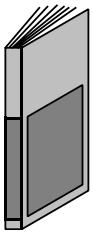
If not otherwise stated, the geometrical values are metric.

This document is subject to modification as a result of technical developments.

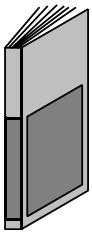


Machining cycles are available for standard machining cycles which are repeated several times. These are described in the documentation, SINUMERIK 840C, Cycles, Programming Guide.

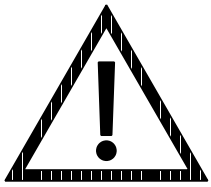




The high-level language CL 800 can be used for the structured programming of subroutines and machining cycles. A detailed description of the language is given in the following documentation
SINUMERIK WS 800A, User's Guide and
SINUMERIK WS 800A, Planning Guide.



	**DANGER** This warning notice means that loss of life, severe personal injury or substantial material damage **will** result if the appropriate precautions are not taken.

	**WARNING** This warning notice means that loss of life, severe personal injury or substantial material damage **can** result if the appropriate precautions are not taken.

	**CAUTION** This warning notice (with warning triangle) means that a minor personal injury **can** result if the appropriate precautions are not taken.

	CAUTION This warning notice (without warning triangle) means that a material damage **can** result if the appropriate precautions are not taken.

	NOTICE This warning notice means that an undesired event or an undesired state **can** result if the appropriate notices are not observed.

Contents

1 Overview and Organization of G Commands/Program Keys

1.1 Overview of G commands

G00	Rapid traverse, exact stop coarse
G01	Linear interpolation
G02	Circular interpolation clockwise
G03	Circular interpolation counterclockwise
G04	Dwell, under address X or F in seconds, under address S in spindle revolutions
G06	SPLINE- interpolation
G09	Speed reduction, exact stop fine (non modal) coarse or fine
G10	Polar coordinate programming, rapid traverse
G11	Polar coordinate programming, linear interpolation
G12	Polar coordinate programming, circular interpolation clockwise
G13	Polar coordinate programming, circular interpolation counterclockwise
G14	Dwell setpoint-related to axes or spindles [5]
G15	Define the axes for 5D tool length compensation [3]
G16	Plane selection with free choice of axis
G17	Plane selection X-Y
G18	Plane selection Z-X
G19	Plane selectionY-Z
G24	Dwell in relation to actual value for axes or spindles [4]
G25	Minimum working area limitation
G26	Maximum working area limitation or spindle speed limitation under address S
G[...]27	Handwheel override in Automatic mode, axis-specific [3]
G33	Thread cutting, constant lead
G34	Thread cutting, linear increasing lead
G35	Thread cutting, linear decreasing lead
G36	Thread cutting, position-controlled spindle (rigid tapping) [2]
G40	Tool radius compensation OFF
G41	Tool radius compensation, left of the contour
G42	Tool radius compensation, right of the contour
G48	Exit contour as approach contour
G50	Deselect scale modification
G51	Select scale modification
G53	Suppression of zero offset (non modal)
G54	1st settable zero offset
G55	2nd settable zero offset
G56	3rd settable zero offset
G57	4th settable zero offset
G58	1st programmable zero offset, several ZOs [5]
G59	2nd programmable zero offset, several ZOs [5]
G60	Feedrate reduction, exact stop fine
G62	Contouring, block transition with speed reduction
G63	Tapping with compensating chuck (tapping without encoder)
G64	Contouring, block transition without speed reduction
G68	Absolute dimensioning along shortest path (rotary axis only)
G70	Inch input system
G71	Metric input system
G74	Reference point approach by program
G75	Set reference dimension

---

1) *Software version 1 and 2 only*
2) *Software version 2 and higher*
3) *Software version 3 and higher*
4) *Software version 4 and higher*
5) *Software version 5 and higher*

G80 Clearing G81 to G89
G81 Call cycle L81: Drilling, centering
G82 Call cycle L82: Drilling, counterboring
G83 Call cycle L83: Deep-hole drilling
G84 Call cycle L84: Tapping with/without encoder
G85 Call cycle L85: Boring 1
G86 Call cycleL86: Boring 2
G87 Call cycle L87: Boring 3
G88 Call cycle L88: Boring 4
G89 Call cycle L89: Boring 5
G90 Absolute dimensioning
G91 Incremental dimensioning
G92 P . . Ratio of working diameter to unit diameter in cylindrical interpolation
S . . Spindle speed setpoint limitation with active G96
T . . Smoothing and feed run-up time when thread cutting
A . . Initial angle offset for multiple thread
G94 Feedrate under address F in mm/min or inch/min or degree/min, independent of G70/G71
G[...]94 Simulation feedrate F in mm/inch/degree per min [4]
G95 Feedrate under address F in mm/rev or inch/rev
G96 Feedrate under address F in mm/rev or inch/rev and constant cutting speed under address S in m/min or ft/min, independent of G70/G71
G97 Cancel G96, store last setpoint speed of G96
G98 Leading feedrate under address F in rev/min for a rotary axis [2]
G[...]98 Leading feedrate F in rev/min referred to the rotary axis [4]
G[...]103 Endless rotation ON clockwise [4]
G[...]104 Endless rotation ON counterclockwise [4]
G[...]105 Endless rotation OFF [4]
G110 Pole specification, relative to the last position reached
G111 Pole specification, absolute with reference to the workpiece zero
G112 Pole specification, relative to the last valid pole
G[...]119 Endless rotation OFF with oriented stop [4]
G130 Deselect transformation TRANSMIT or deselect transformation coordinate rotation 2D/3D
G131 Transformation TRANSMIT
G133 Transformation coordinate rotation 2D
G135 Transformation coordinate rotation 3D
G147 Approach contour with straight line
G148 Exit contour with straight line
G150 Deselect coupled motion
G151 1st coupled motion combination
G152 2nd coupled motion combination
G153 3rd coupled motion combination
G154 4th coupled motion combination
G155 5th coupled motion combination
G156 6th coupled motion combination
G157 7th coupled motion combination
G158 8th coupled motion combination
G159 9th coupled motion combination
G171 Fill up FIFO buffer [1] [5]
G172 Continue program execution [5]
G175 Update zero offset, length compensation and angle of rotation in every block [1]
G176 Freeze length compensation, zero offset and angle of rotation [1]

———

1) *Software version 1 and 2 only*
2) *Software version 2 and higher*
3) *Software version 3 and higher*
4) *Software version 4 and higher*
5) *Software version 5 and higher*

G180 Assignment of channel for deselection of protection space
G181 Assignment of channel for selection of protection space
G195 Revolutional feedrate F in mm/rev, inch/rev or degrees/rev with reference to the set speed of the rotary axis 5)
G[...]195 Revolut. feedr. F in degrees/rev with reference to the set speed of the rotary axis 5)
G200 Axis actual-value synchronization 3)
G220 Travel to fixed stop OFF 3)
G221 Travel to fixed stop ON 3)
G222 Change the clamping torque for travel to fixed stop 3)
G230 Deselect transformationTRANSMIT or
deselect transformation coordinate rotation 2D/3D
G231 Transformation TRANSMIT
G233 Transformation coordinate rotation 2D
G235 Transformation coordinate rotation 3D
G247 Approach contour in quadrant
G248 Exit contour in quadrant
G295 Revolutional feedrate F in mm/rev, inch/rev or degree/rev referred to the actual speed of the rotary axis 4)
G[...]295 Rotational feedrate F in degree/rev referred to the actual speed of the rotary axis 4)
G330 Deselect transformation TRANSMIT or
deselect transformation coordinate rotation 2D/3D
G331 Transformation TRANSMIT
G333 Transformation coordinate rotation 2D
G335 Transformation coordinate rotation 3D
G347 Approach contour in semi-circle
G348 Exit contour in semi-circle
G400 Gear interpolation OFF 3)
G401 Define/delete gear interpolation 3)
G402 SWITCH ON/OVER/OFF gear interpolation 3)
G403 Gear interpolation ON with on-the-fly synchronization 3)
G404 "Disable speed limitation for FA" 7)
G405 "Enable speed limitation for FA" 7)
G410 IKA link branch OFF 4)
G411 Define/delete IKA configuration 4)
G412 Switch IKA link branch ON/OVER 4)
G420 Deactivate ”Programmable emergency retraction” 4)
G421 Activate monitoring sources and enable reactions 4)
G422 Configure generator operation 4)
G423 Configure stop as open-loop control function 4)
G424 Configure stop as autonomous drive function 4)
G425 Configure retraction as open-loop control function 4)
G426 Configure retraction as autonomous drive function 4)
G431 Direction-dependent actual value link for thread machining 5)
G432 Absolute value dependent actual value link for thread machining 5)
G450 Tool radius compensation with transition circle 2)
G451 Tool radius compensation with intersection 2)
G455 Corrected approach to contour 2)
G456 Normal approach to contour2)
G511 Function output dependent on distance to go with setpoint reference 6)
G522 Function output dependent on position with actual value reference 6)
G600 Exact stop coarse with active G00 4)
G620 Continuous-path mode with speed reduction with active G00 4)
G640 Continuous-path mode without speed reduction with active G00 4)
G720 Extended measurement in one block with deletion of distance to go 4)
G721 Extended measurement in one block without deletion of distance to go 4)
G722 Extended measurement over a number of blocks 4)
G931 Circle - straight line - circle - sprint functions
G932 Straight line - circle with radius, chamfer - sprint functions
G933 Circle - straight line with radius, chamfer - sprint functions
G934 Circle with radius, chamfer - sprint functions
G935 Three-point definition with radius, chamfer - sprint functions

---

1) *Software version 1 and 2 only*
2) *Software version 2 and higher*
3) *Software version 3 and higher*
4) *Software version 4 and higher*
5) *Software version 5 and higher*
6) *Software version 5.6 and higher*
7) Software version 6 and higher

1.2 Program keys

Some G functions are only available until as of a certain software version. This is indicated in the description of the function or in Section 1.1. Overview of G commands.

1.2.1 Program keys for channel-specific functions

G gr.	ISO	Code	Function and meaning	Sect.
1	G	00 5)	Rapid traverse, exact stop coarse	8.1
		01 5)	Linear interpolation	8.2
		06 5)	SPLINE interpolation	8.9
		10 5)	Polar coordinate programming, rapid traverse	8.5
		11 5)	Polar coordinate programming, linear interpolation	8.5
		02 5)	Circular interpolation clockwise	8.3
		03 5)	Circular interpolation counterclockwise	8.3
		12 5)	Polar coordinate programming, circular interpolation clockwise	8.5
		13 5)	Polar coordinate programming, circular interpolation counterclockwise	8.5
		33 5)	Thread cutting, lead constant	8.8.1
		34 5)	Thread cutting, linear lead increase	8.8.1
		35 5)	Thread cutting, linear lead decrease	8.8.1
		36 5)	Thread cutting, position-controlled spindle	8.8.1
2	G	09 #	Speed reduction, exact stop fine coarse or fine	6.1
3	G	16	Plane selection with free choice of axis	3.1.6
		17 5)	Plane selection X-Y (initial setting in machine data)	3.1.5
		18 5)	Plane selection Z-X (initial setting in machine data)	3.1.5
		19 5)	Plane selection Y-Z (initial setting in machine data)	3.1.5
4	G	40	Deselect tool radius compensation	11.5
		41	Tool radius compensation left	11.3
		42	Tool radius compensation right	11.3
5	G	53 #	Suppression of zero offset	3.6
6	G	54 5)	Settable zero offset 1	3.6.1
		55 5)	Settable zero offset 2	3.6.1
		56 5)	Settable zero offset 3	3.6.1
		57 5)	Settable zero offset 4	3.6.1

Note:

The explanation of symbols given below will appear unchanged on the following pages to ensure consistency of footnotes even though not all of them do actually occur in the table.

---

1) No further functions must be written in this block except block number and comment

2) Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)

3) Other addresses are selectable (A, B, C, E, U, V, W and Q)

4) G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;
G80 to G89 cannot be used in the seventh subroutine level with SW 3.

*5) Initial setting can be **freely** set in MD.*

6) The stated values depend on the input and/or position control resolution

 # Non modal, all other modal
Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)
Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

G gr.	ISO	Code	Function and meaning	Sect.
7	G	04 # [1]	Dwell time specified under address X or F in seconds and address S in spindle revolutions	8.18.1 8.18.1
		25 # [1]	Minimum working area limitation	3.3
		26 # [1]	Maximum working area limitation	3.3
			G26 S.. Permanently active spindle speed limitation	5.3
		58 # [1]	Programmable zero offset 1.. n	3.6.2
		59 # [1]	Programmable zero offset 1.. n	3.6.2
		92 # [1]	Spindle speed setpoint limitation under address S (acts only with "G96 active")	5.2
			Smoothing and feed ramp-up time with thread cutting under address T..	8.8.6
			Cylindrical interpolation, ratio of working diameter to unit diameter under address P..	8.11
			Initial angle offset for multiple turn threads under address A..	8.8.4
		74 # [1]	Reference point approach by program	8.17.1
		200 #	Axis actual-value synchronization	7.3
		24 #	Dwell in relation to actual value for axes or spindles	8.18.2
		14 #	Dwell in relation to setpoint for axes or spindles	8.18.2
		75 #	Set reference dimension	8.17.1.1
8	G	60 [5]	Speed reduction, exact stop fine	6.1
		62 [5]	Contouring, block transition with speed reduction	6.2
		63 [5]	Tapping without encoder, feedrate override 100 %	8.8.3
		64 [5]	Contouring, block transition without speed reduction	6.2
9	G	70 [5]	Input system inch	3.5
		71 [5]	Input system metric	3.5
10	G	80 [4]	Deselect G81 to G89	14
		81 [4]	Call cycle L81: - Drilling, centering	14
		82 [4]	Call cycle L82: - Drilling, counterboring	14
		83 [4]	Call cycle L83: - Deep-hole drilling	14
		84 [4]	Call cycle L84: - Tapping with/without encoder	14
		85 [4]	Call cycle L85: - Bore 1	14
		86 [4]	Call cycle L86: - Bore 2	14
		87 [4]	Call cycle L87: - Bore 3	14
		88 [4]	Call cycle L88: - Bore 4	14
		89 [4]	Call cycle L89: - Bore 5	14
11	G	68	Absolute dimension. along shortest path (rotary axis only)	3.4.6
		90	Absolute dimensioning	3.4.1
		91	Incremental dimensioning	3.4.2

---

[1] No further functions must be written in this block except block number and comment

[2] Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)

[3] Other addresses are selectable (A, B, C, E, U, V, W and Q)

[4] G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;
G80 to G89 cannot be used in the seventh subroutine level with SW3.

*[5] Initial setting can be **freely** set in MD.*

[6] The stated values depend on the input and/or position control resolution

 # Non modal, all other modal

Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)

Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

G gr.	ISO	Code	Function and meaning	Sect.
12	G	94 5) 95 5) 96 5) 97 5) 98 5) 195 295	Feedrate under address F in mm/min or inch/min Feedrate under address F in mm/rev or inch/rev Feedrate under address F in mm/rev or inch/rev and constant cutting speed under address S in m/min or ft/min Cancel G96, store last set speed Feedrate under address F in rev/min for a rotary axis Revolutional feedrate in mm/rev, inch/rev, degrees/rev with reference to the set speed of the rotary axis Revolutional feedrate in mm/rev, inch/rev or degree/rev referred to the actual speed of the rotary axis	4.3 4.3 5.5 5.5 5.5 4.3 4.3 4.3
13	G	147 # 1) 247 # 1) 347 # 1) 148 # 1) 248 # 1) 348 # 1) 48 # 1) 110 # 1) 111 # 1) 112 # 1) 15 # 1)	Approach contour with straight line Approach contour with quadrant Approach contour with semi-circle Exit contour with straight line Exit contour with quadrant Exit contour with semi-circle Exit contour as approach contour Polar coordinate programming, relative to last position Polar coord. progr., absolute with reference to workp. zero Polar coordinate programm., relative to the last valid pole Define the axes for 5D tool length compensation	8.17 8.17 8.17 8.17 8.17 8.17 8.17 8.5 8.5 8.5 10.10.1
14	G	50 1) 51 1)	Scale modification Deselection of scale modification	3.8 3.8
15	G	150 5) 151 5) 152 5) 153 5) 154 5) 155 5) 156 5) 157 5) 158 5) 159 5)	Deselection of coupled motion Coupled axis combination 1 Coupled axis combination 2 Coupled axis combination 3 Coupled axis combination 4 Coupled axis combination 5 Coupled axis combination 6 Coupled axis combination 7 Coupled axis combination 8 Coupled axis combination 9	8.16 8.16 8.16 8.16 8.16 8.16 8.16 8.16 8.16 8.16
16	G	 130 * 1) 131 1) 133 1) 135 1)	GROUP 1 Deselection of transformation TRANSMIT or Deselection of transformation coordinate rotation 2D/3D Transformation TRANSMIT Transformation coordinate rotation 2D Transformation coordinate rotation 3D	 8.14 8.14 8.14 8.15 8.15
17	G	 230 * 1) 231 1) 233 1) 235 1)	GROUP 2 Deselection of transformation TRANSMIT or Deselection of transformation coordinate rotation 2D/3D Transformation TRANSMIT Transformation coordinate rotation 2D Transformation coordinate rotation 3D	 8.14 8.14 8.14 8.15 8.15

1) No further functions must be written in this block except block number and comment

2) Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)

3) Other addresses are selectable (A, B, C, E, U, V, W and Q)

4) G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;
G80 to G89 cannot be used in the seventh subroutine level with SW3.

*5) Initial setting can be **freely** set in MD.*

6) The stated values depend on the input and/or position control resolution

 # Non modal, all other modal

Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)

Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

G gr.	ISO	Code	Function and meaning	Sect.
18	G		GROUP 3	
		330 * [1]	Deselection of transformation TRANSMIT or	8.14
			Deselection of transformation coordinate rotation 2D/3D	8.14
		331 [1]	Transformation TRANSMIT	8.14
		333 [1]	Transformation coordinate rotation 2D	8.15
		335 [1]	Transformation coordinate rotation 3D	8.15
19	G		Contour definitions:	
		931 #	Circle - line - circle	8.4.1
		932 #	Line - circle with radius / chamfer	8.4.1
		933 #	Circle - line with radius / chamfer	8.4.1
		934 #	Circle with radius	8.4.1
		935 #	Three-point definition with radius / chamfer	8.4.1
20	G	400 #	Gear interpolation OFF	8.12
		401 #	Define/delete gear interpolation	8.12
		402 #	SWITCH ON/OVER/OFF gear interpolation	8.12
		403 #	Gear interpolation ON with on-the-fly synchronization	8.12
		404 #	"Disable speed limitation for FA"	8.12
		405 #	"Enable speed limitation for FA"	8.12
21	G	171 #	Fill up FIFO buffer	8.19.4
		172 #	Continue program execution	8.19.4
		511 #	Function output dependent on distance to go with setpoint reference	9.6.1
		522 #	Funct. output dependent on position with actual value ref.	9.6.2
22	G	175 #	ZO, length compensation, angle of rotation update in each block	8.19.3
		176 #	Freeze length compensation/zero offsets/ angle of rotation	8.19.3
		410 #	IKA link branch OFF	8.13
		411 #	Define/delete IKA link branch	8.13
		412 #	Switch IKA link branch ON/OVER	8.13
23	G	720 #	Extended measurement with deletion of distance to go	8.19.5
		721 #	Extended measurement without deletion of distance to go	8.19.5
		722 #	Extended measurement without deletion of distance to go on completion of programmed number of measurements	8.19.5
24	G	220	Travel to fixed stop OFF	8.17.2
		221	Travel to fixed stop ON	8.17.2
		222	Change the clamping torque for travel to a fixed stop	8.17.2
25	G	450 [5]	TRC with transition circle	11.1
		451 [5]	TRC with intersection	11.1
26	G	455	Corrected approach to contour	11.2
		456	Normal position, approach to contour	11.2
28	G	420 #	Deactivate programmable emergency retraction	8.19
		421 #	Activate monitoring sources	8.19
		422 #	Configure generator operation	8.19
		423 #	Configure stop as open-loop control function	8.19
		424 #	Configure stop as autonomous drive function	8.19
		425 #	Configure retraction as open-loop control function	8.19
		426 #	Configure retraction as autonomous drive function	8.19
		180 #	Assignment of channel for deselection of protection space	8.19
		181 #	Assignment of channel for selection of protection space	8.19

1) No further functions must be written in this block except block number and comment
2) Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)
3) Other addresses are selectable (A, B, C, E, U, V, W and Q)
4) G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;
G80 to G89 cannot be used in the seventh subroutine level with SW3.
*5) Initial setting can be **freely** set in MD.*
6) The stated values depend on the input and/or position control resolution
 # Non modal, all other modal
Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)
Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

G gr.	ISO	Code	Function and meaning	Sect.
29	G	600 620 640	Exact stop coarse with active G00 Continuous-path mode with speed reduction Continuous-path mode without speed reduction	6.1 6.2 6.2
30	G	431 432	Direction-dependent actual value link for G33, G34, G35 Absolute value dep. actual value link for G33, G34, G35	8.8.1 8.8.1
	A 3)	0 to 359.99999° – 359.99999 to 359.99999°	Angle in degrees with contour definition Angle in degrees with polar coordinates	8.4 8.5
	B 3)	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	Radius with circular interpolation and polar coordinates in mm 6) Radius with circular interpolation and polar coordinates in inch 6) Corner with contour definition Chamfer with contour def. in mm 6) Chamfer with contour def. in inch 6) Radius with contour def. in mm 6) Radius with contour def. in inch 6)	8.3.2 8.4 8.3.2 8.4 8.4 8.4 8.4 8.4 8.4
	D 2)	1 to 1600 0	Selection of tool offset Deselection of tool offset	10 10
	F F1=	0.01 to 10720000 0.001 to 1072000 0.001 to 99.999 0.001 to 10720000 0.0001 to 1072000 0.001 to 16.000 0.0001 to 0.6000	Feedrate in mm/min 6) Feedrate in inch/min 6) Dwell in s Feedrate in mm/rev 6) Feedrate in inch/rev 6) Thread, lead increase or decrease in mm/rev 6) Thread, lead increase or decrease in inch/rev 6)	4 4 8.18.1 4 4 8 8
	H 2)	1 to 99999999	Auxiliary function	9.2
	I	±0.001 to ± 99999.999 ±0.0001 to ±3999.9999 0.001 to 120000 0.0001 to 12000	Interpolation parameter for X axis in mm 6) Interpolation parameter for X axis in inch 6) Thread lead in mm 6) Thread lead in Inch 6)	8.3.1 8.3.1 8.8 8.8
	J	±0.001 to ± 99999.999 ±0.0001 to ±3999.9999 0.001 to 120000 0.0001 to 12000	Interpolation parameter for Y axis in mm 6) Interpolation parameter for Y axis in inch 6) Thread lead in mm 6) Thread lead in Inch 6)	8.3.1 8.3.1 8.8 8.8
	K	±0.001 to ± 99999.999 ±0.0001 to ±3999.9999 0.001 to 120000 0.0001 to 12000	Interpolation parameter for Z axis in mm 6) Interpolation parameter for Z axis in inch 6) Thread lead in mm 6) Thread lead in Inch 6)	8.3.1 8.3.1 8.8 8.8
	L	1 to 9999	Subroutine number	13
	$^{L}_{F}$		End of block	2.1
	MPF	1 to 9999	Main program	2
	M	00 01	Programmed stop, unconditional Programmed stop, conditional	9.1.1 9.1.2

1) *No further functions must be written in this block except block number and comment*
2) *Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)*
3) *Other addresses are selectable (A, B, C, E, U, V, W and Q)*
4) *G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;*
G80 to G89 cannot be used in the seventh subroutine level with SW3.
5) *Initial setting can be **freely** set in MD.*
6) *The stated values depend on the input and/or position control resolution*
\# *Non modal, all other modal*
Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)
Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

G gr.	ISO	Code	Function and meaning	Sect.
	M	02 17 30	End of program End of program End of program	9.1.3 9.1.3 9.1.3
	M [2)]	03 04 05 19 M1=... M2=... M3=...	Spindle rotation clockwise Spindle rotation counterclockwise Spindle stop, without orientation Oriented spindle stop, angle under address S in degrees Extended address with specification of spindle number	5.4 5.4 5.4 5.4 5.4
	M	36 37	Feedrate as programmed under F (also acts with G33) Feedrate in mm/min, m/min or mm/rev reduced by 1:100 (also acts with G33)	 9.1.5 9.1.5
	M [2)]	0 to 9999 1= ... bis 99= ...	Miscellaneous functions, user-assignable except for groups M1 to M4 Extended address with specification of number	9.1 9.1
	N		Subblock	2.3.1
	P	1 to 9999 0.00100 to 99.99999 0.00001 to 99.99999	Number of subroutine passes Factor for cylindrical interpolation Scale factor	13 8.11 3.8
	P[...]	±0.001 to ±99999.999 ±0.0001 to±3999.9999 ±0.001 to ±99999.999 ± 0.001 to 359.999	Positional data of a simultaneous axis in mm [6)] Positional data of a simultaneous axis in inch [6)] Positional data of a rotary axis in degrees with G91 Positional data of a rotary axis in degrees with G90	8.10 8.10 8.10 8.10
	R [2)]	0 to 10019	R parameter	12
	S [2)]	1 to 99999 1 to 99999 0.1 to 359.99 0.1 to 99.9	Spindle speed in rev/min or 0.1 rev/min and constant cutting speed in m/min or 0.1 m/min or ft/min or 0.1 ft/min Spindle speed limitation in rev/min or 0.1 rev/min Oriented spindle stop in degrees, distance from the zero mark of the encoder Dwell in spindle revolutions	 5 5.2 5 8.18.1
	SPF	1 to 9999	Subroutine	13
	T [2)]	1 to 99999999	Tool number	9.3

---

1) *No further functions must be written in this block except block number and comment*

2) *Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)*

3) *Other addresses are selectable (A, B, C, E, U, V, W and Q)*

4) *G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;*
G80 to G89 cannot be used in the seventh subroutine level with SW3.

5) *Initial setting can be **freely** set in MD.*

6) *The stated values depend on the input and/or position control resolution*

\# *Non modal, all other modal*
Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)
Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

G gr.	ISO	Code	Function and meaning	Sect.
	X 2)	±0.001 to ±99999.999 ±0.0001 to ±3999.9999 0.001 to 99999.999 ±0.001 to ±99999.999 ± 0.001 to 359.999	Positional data of a linear axis in mm 6) Positional data of a linear axis in inch 6) Dwell in s Positional data of a rotary axis in degrees with G91 Positional data of a rotary axis in degrees with G90	3 3 8.18.1 3 3
	Y 2)	±0.001 to ±99999.999 ±0.0001 to ±3999.9999 ±0.001 to±99999.999 ± 0.001 to 359.999	Positional data of a linear axis in mm 6) Positional data of a linear axis in inch 6) Positional data of a rotary axis in degrees with G91 Positional data of a rotary axis in degrees with G90	3 3 3 3
	Z 2)	±0.001 to ±99999.999 ±0.0001 to ±3999.9999 ±0.001 to ±99999.999 ± 0.001 to 359.999	Positional data of a linear axis in mm 6) Positional data of a linear axis in inch 6) Positional data of a rotary axis in degrees with G91 Positional data of a rotary axis in degrees with G90	3 3 3 3
	Axis 4 to 30	±0.001 to ±99999.999 ±0.0001 to ±3999.9999 ±0.001 to ±99999.999 ± 0.001 to 359.999	Positional data of a linear axis in mm 6) Positional data of a linear axis in inch 6) Positional data of a rotary axis in degrees with G91 Positional data of a rotary axis in degrees with G90	3 3 3 3
	%		Beginning of program	2.1
	:		Main block	2.3.1
	***/*:**		Skippable main block	2.3.2
	/N		Skippable subblock	2.3.2
	/		Division with parameters, skippable block	12.3
	( )		Beginning of comment End of comment	2.3.3 2.3.3
	[ ]		Program coordination or axis-specific programming	7.2 2.4.2
	=		Delimiter, mandatory for address extensions e.g. R35=123.5, X=100 Q1=1234.567	2.4.1 12
	+		Addition with parameters	12.3
	—		Subtraction with parameters	12.3
	*		Multiplication with parameters	12.3

---

1) No further functions must be written in this block except block number and comment

2) Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)

3) Other addresses are selectable (A, B, C, E, U, V, W and Q)

4) G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;
G80 to G89 cannot be used in the seventh subroutine level with SW3.

*5) Initial setting can be **freely** set in MD.*

6) The stated values depend on the input and/or position control resolution

 # Non modal, all other modal

Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)

Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

Stan-dard	@36b	Channel-specific G group organization														
1	0	00 MD	01 MD	10 MD	11M D	02M D	03M D	33M D	34M D	35M D	06M D	12M D	13M D	36M D		
2	1	09														
3	2	17 M MD	18 T MD	19M D	16											
4	3	40 M/T	41	42												
5	4	53														
6	5	54 M/T MD	55 MD	56 MD	57 MD											
7	6	04	25	26	58	59	92	74	200	14	24					
8	7	60 M MD	63 MD	64 T MD	62 MD											
9	8	70 MD	71 M/T MD													
10	9	80 M/T	81	82	83	84	85	86	87	88	89					
11	10	90 M/T	91	68												
12	11	94 M MD	95 T MD	96 MD	97 MD	195	98 MD	295								
13	12	147	247	347	148	248	348	48	110	111	112	15				
14	13	50 M/T	51													
15	14	150 M/T MD	151 MD	152 MD	153 MD	154 MD	155 MD	156 MD	157 MD	158 MD	159 MD					
16	15	130 M/T	131		133		135									
17	16	230 M/T	231		233		235									
18	17	330 M/T	331		333		335									
19	18	931	932	933	934	935										
20	19			400	401	402	403	404	405							
21	20	171	172	511	522											
22	21	175 M/T	176	410	411	412										
23	22				720	721	722									
24	23	220	221	222												
25	24	450 M/T MD	451 MD													
26	25	455 M/T	456													
27	26	27														
28	27	420	421	422	423	424	425	426								
29	28	600		640	620											
30	29	431 M	432 T													

1.2.2 Program keys for axis-specific functions

G gr.	ISO	Code	Function and meaning	Sect.
1	G[...]	27	Handwheel override in Automatic mode	8.10.3
2	G[...]	103 104 105 119	Endless rotation ON chlockwise Endless rotation ON counterclockwise Endless rotation OFF Endless rotation OFF with oriented STOP	8.10.3 8.10.3 8.10.3 8.10.3
3	G[...]	94 5) 98 5) 195 295	Feedrate F in mm/min, inch/min, degree/min Leading feedrate in degree/min referred to the rotary axis Revolutional feedrate F in degrees/rev with reference to the set speed of the rotary axis Revolutional feedrate F in degrees/rev with reference to the actual speed of the rotary axis	4.4 4.4 4.4
	F[...]	 0.01 to 10720000 0.001 to 1072000	Simultaneous axes Axis-specific feedrate in mm/min 6) Axis-specific feedrate in inch/min 6)	8.10.1 4.4 4.4

Stan-dard	@36b	**Axis-specific G group organization**														
1	0	27														
2	1	105	103	104	119											
3	2	94				195	98	295								

END OF SECTION

1) *No further functions must be written in this block except block number and comment*
2) *Extended address possible (e.g. S1=.., M6=.., ..., see Section 2.4.1)*
3) *Other addresses are selectable (A, B, C, E, U, V, W and Q)*
4) *G80 to G89 cannot be used in the third subroutine level with SW 1 and 2;*
G80 to G89 cannot be used in the seventh subroutine level with SW3
5) *Initial setting can be **freely** set in MD.*
6) *The stated values depend on the input and/or position control resolution*
\# *Non modal, all other modal*
Initial setting turning (initial setting, after RESET, M02/M30, after switching on the control)
Initial setting milling (initial setting, after RESET, M02/M30, after switching on the control)

2 Fundamentals of Programming

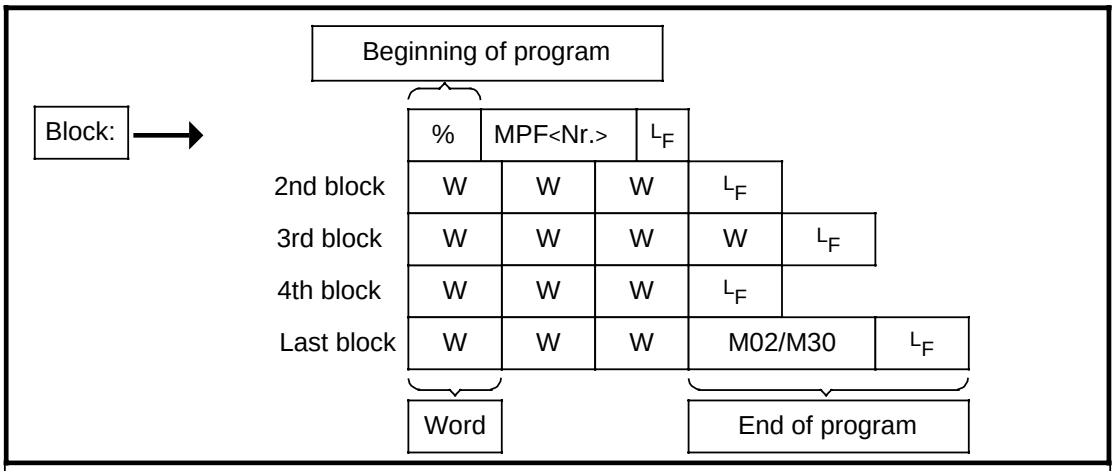
2.1 Program structure

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

A part program comprises:

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

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



Program structure: Part program in input/output format

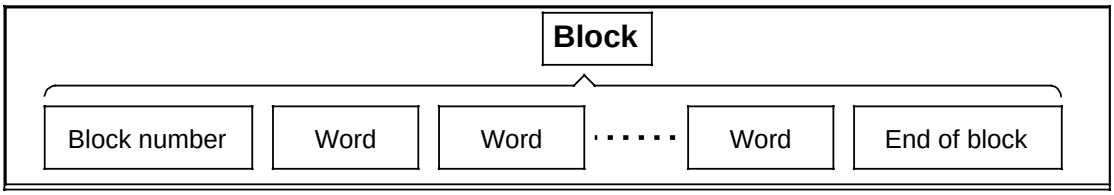
Subroutines and cycles can be components of the program. Cycles are subroutines which have been created either by the machine manufacturer or by SIEMENS.

Up to 900 programs can be stored in the NCK program memory and, in each case, 250 workpieces with 250 files each can be stored per directory (local or global) in the MMC.

2.2 Block format

A block contains all data required to execute a working step. The block comprises several words and the ”L_F” character for “end of block” .

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



Block structure: Format of a block

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

Programming without a block number is permissible. However, a block search and the jump function can be implemented only by means of block numbers.

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

Block example:

N9235 G.. X.. Y.. Z.. F.. S.. T.. M.. H.. L_F

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

Each block must be terminated with the “L_F” end-of-block character. This character appears on the screen as the special character L_F. When the program is printed out, this character does not appear.

2.3 Block elements

2.3.1 Main blocks and subblocks

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

IThe main block must contain all words required to start the machining cycle in the program section beginning there.

A **main block** is identified by means of the character ":".
A **subblock** is identified by means of the character "N".

If a block is given a block number, this follows immediately after the ":" or "N" character. The main blocks and subblocks must be identified by different numbers.

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

Example main block: `:10 G01 X10 Y-15 F200 S900 M03 LF`
Example subblock: `N15 Y20 LF`

A **program section** comprises one main block and several subblocks, e.g.:

```
:10  G01  X10  Y-15  F200  LF   (Main block)
N20  Y35  LF                    (1st subblock)
N30  X20  Y40  LF               (2nd subblock)
N40  Y-10  LF                   (3rd subblock etc.)
```

2.3.2 Skippable blocks

To individually control operation sequences, it must be possible to run or skip certain functions (e.g. inprocess measurement, part loading and unloading) and therefore the corresponding program 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 via the menu "Program modification" or via the PLC user program. A section can be skipped by skipping several consecutive blocks.

```
:10  G01  X10  Y-15  F200  LF
N20  Y35  LF
/N30  X20  Y40  LF                  Skipped block
N40  Y10  LF
```

Note:

To obtain shorter block change times, several blocks are buffered. If the machine interrupts machining on account of M00 (Programmed Stop), the next blocks will already have been input. "Block skip" acts only on those blocks which have not yet been buffered. This buffering can be prevented by programming L999 (disable pre-reading; @ 714) after the block containing M00.

In addition to the skipping of blocks available until now, 8 additional skippable levels can be programmed. The skip character is extended by an additional number between 1 and 8 to distinguish the skip criteria. The skip characters "**/**1" to "**/**8" must be programmed at the beginning of the block. The function "Several skip block levels" is modal. It is not possible to parameterize these characters (e.g. **/**R10 or /P20).

Example:

Part program %2; skip levels "**/**1" and "**/**2" were activated on program start.

```
%2
N5
/ N10 G X Y LF
/1 N15
/1 N20
/1 N25
/1 N30
/2 N35
/2 N40
/3 N45
/1 N50
```

Skip level "**/**1" was deselected between blocks N15 and N20. Because the function is modal, blocks N20, N25 and N30 are also skipped. The skip character "**/**2" in block N35 cancels the modal skip character "**/**1", and "**/**2" is activated. Blocks N35 and N40 are also skipped because skip character "**/**2" is active for these. The blocks are again processed from N45 inclusive ("/1" in block N50 is not active).

Note:

In the same way that an active condition is maintained, an inactive condition within a block is also maintained by skipped blocks. In the following example %3, if the skip level "**/**1" is selected between blocks N10 and N15, then the blocks from N30 onwards are skipped.

```
%3
/1 N5
/1 N10
/1 N15
/1 N20
   N25
/1 N30
/1 N35
   N40
```

2.3.3 Comments and notes

The blocks in a program can be explained by means of comments and notes (as from SW6).

– A comment permits instructions for the operator to be displayed on the screen (in the comments display).

– A **note** merely documents a program block. **Notes** are not displayed in the comments display.

Comments:

- A **comment** must be enclosed in round brackets "(..)". If the ")" is missing, the **comment** reaches the end of the block ("LF)". More than one **comment** are permitted in a block.
- No percentage characters ”%” or additional brackets ”( )” are permitted within the **comment**.
- The **comment** must not be located between the address and a digit or between a word and the associated parameter.
- The **comment** can be up to 254 characters long. A maximum of 41 characters are shown on the screen in the comments line.
- An arithmetic logic block should not contain a **comment**, as this would lead to a slowing down of the process. In any case the arithmetic logic block then counts as an intermediate block for contour definitions and tool radius compensation.

Notes:

- A note begins with ";" and always reaches to the end of the block. This means that only one note is possible for each block.
- If a note is contained within a comment, it may be displayed in the comments display. A ";" in a comment is not recognized as as the beginning of a note.
- A note can be up to 254 long.
- A note does not lead to a slowing down of the program.

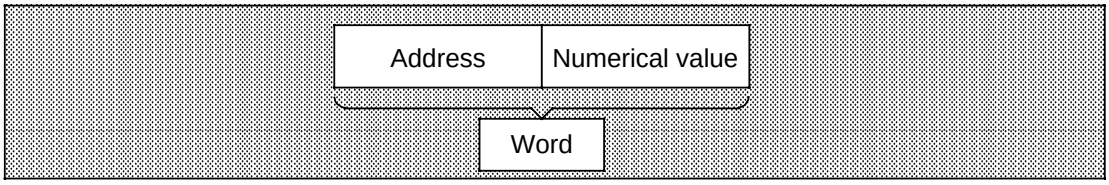
Correct	Wrong
%200 **N005 T01 D01 (tool call) LF** **(roughing) LF** **N020 G01 Z200 +R1 (1st roughing cut) LF** **;A note can reach over a whole line LF**	**%210** **N005 T01 D (tool call) 01 LF** **(ruoghing LF)** **N020 G01 Z200+ (1st roughing cut) R1 LF**
@400 K3 K200 ; change G62 velocity LF	**;@400 cannot be carried out @400 K3 K200 LF**

Note:

Up to software version 2, there must be a space (blank) between the opening bracket and the beginning of the comment as well as between the end of the comment and the closing bracket.

2.4 Word format

A word is an element of a block. It comprises an address character and a string of digits. The address character is normally a letter. The string of digits 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.

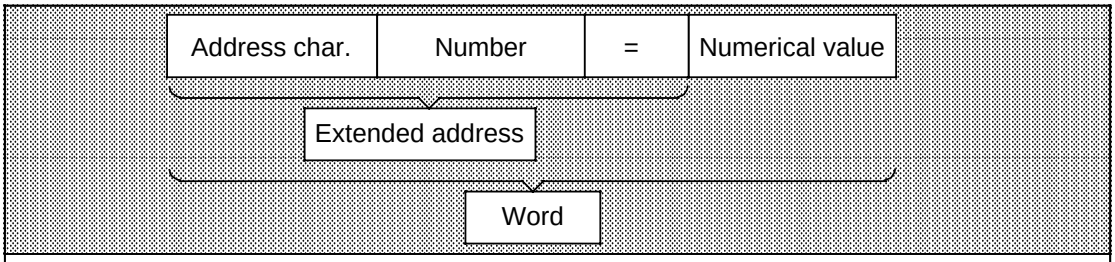


Word structure: Format of a word

Note:

Inputs of more than eight places (including decimal places) are not permitted.

2.4.1 Extended address (not axis-specific)



Address char.:	Axis (e.g. X,Y,Z...) or auxiliary functions (e.g. M,S,T,H,D)
Number:	0...99
=-character:	To separate the address extension number from the numerical value
Numerical value:	± 8 places (including decimal places)

Extended addressing cannot be used for angle, radius and non axis-specific G functions.

Word examples (word format is based on DIN 66025):

`X1=100`	Axis X1 travels to position 100.
`M2=19 S2=0`	Oriented spindle stop of the 2nd spindle to spindle position 0.
`G1=0`	Illegal, error message

Decimal point input:

Value		Programmed value with decimal point
0.1	µm	X0.0001
1	µm	X0.001
10	µm	X0.01
100	µm	X0.1
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, V, W, Q, I, J, K, F, S.

R is valid only in the follwing format: R10 = 50.0
(restriction for S, see program key)

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

Example:

Input	X1.1	means	1100 µm	=	1.1 mm
Input	X10.	means	10000 µm	=	10.0 mm
Input	X1.	means	1000 µm	=	1.0 mm

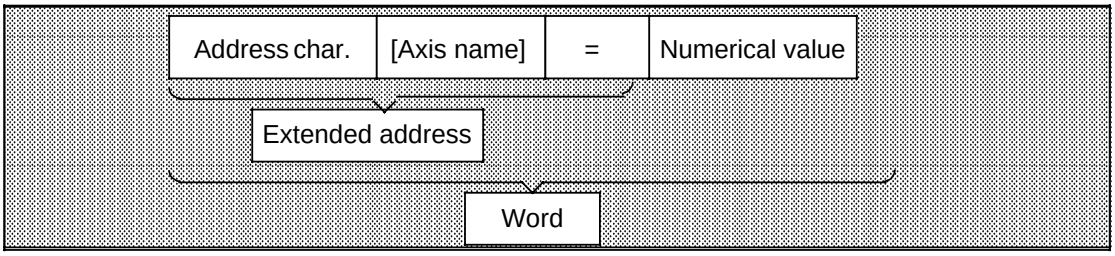
All values with input resolution 0.001 mm.

Note:

Extended measurement is only possible up to the extended address 9.

2.4.2 Extended address (axis-specific)

If axis-specific functions are programmed, square brackets must be programmed after the address character. The square brackets must contain the axis name. It is not necessary to program the "=" sign behind the square brackets.



Examples:

```
N10 P[Y]150   F[Y]1000    G[Y]27      or
N10 P[Y1]=150 F[Y1]=1000  G[Y1]27
```

2.5 Character set

The code used for programming is DIN 66025 (ISO).

The examples used in this 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, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

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

Hexadecimal digits with CL 800 machine code

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

Displayable special characters: see the section entitled "Tape code table"

Permissible addresses:

A Freely assignable address
B Freely assignable address
C Freely assignable address
D Tool offset number
E Freely assignable address
F Feedrate
G Preparatory function
H Auxiliary function
I Interpolation parameter
J Interpolation parameter
K Interpolation parameter
L Subroutine
M M function
N Subblock
P Number of subroutine passes . . .
Q Freely assignable address
R Arithmetic parameter
S Spindle speed, S function
T Tool
U Freely assignable address
V Freely assignable address
W Freely assignable address
X Freely assignable address
Y Freely assignable address
Z Freely assignable address

2.6 Character and file formats for serial transfer in tape format

2.6.1 Reading in data into the control

The data can be read in from any data medium (e.g. PC, external disc drive, tape etc.). The control requires certain identifiers and formats to store the data read in via the serial interface. Only data which have a valid identifier will be stored.

Identifier	Meaning
UMS	**U**ser **m**emory **s**ubmodule
GIA 2)	**G**ear **i**nterpol**a**tion
IKA 2)	**I**nterpolation and compensation with tables
MPF	Part program (**M**ain **P**rogram **F**ile)
PCF 1)	PLC error message text (**P**L**C F**iles)
RPA	R parameter numbers with value assignments (**R P**arameter **A**ctive)
SEA	Addresses with value assignments (**Se**tting Data **A**ctive)
SEA4	Cycle setting data (**Se**tting Data **A**ctive)
SPF	Subroutine (**S**ub **P**rogram **F**ile)
TEA1	NC machine data (**Te**sting Data **A**ctive **1**)
TEA2	PLC machine data (**Te**sting Data **A**ctive **2**)
TEA3 2)	Drive machine data (**Te**sting Data **A**ctive **3**)
TEA4	Cycle machine data (**Te**sting Data **A**ctive **4**)
TOA	Tool offsets (**T**ool **O**ffset **A**ctive)
ZOA	Zero offsets (**Z**ero **O**ffset **A**ctive)
ZOA1...4	Angle of rotation (**Z**ero **O**ffset **A**ctive)

The Section ”Input/output formats” gives an exact description of the program formats.

2.6.2 Reading out data onto external data media

When reading out data, the control automatically generates the correct identifiers and formats which are needed to read in the data.

———

1) Up to SW 2

2) SW 3 and higher

2.6.3 Reading in and reading out data in PC format

During serial transmission in PC format, special headers are generated which are read out before the actual identifier (e.g. %MPF...). You will find additional information on this subject in the SINUMERIK 840C Operator's Guide.

Note:

When programming, no blank is permitted between % and identifier (MPF, SPF).

2.6.4 Leader and file comments

The leader and file comments are used to identify the programs. All characters except for the beginning-of-program character (%) can be entered in the file comments. The leader is not stored and is ignored by the control during program processing.

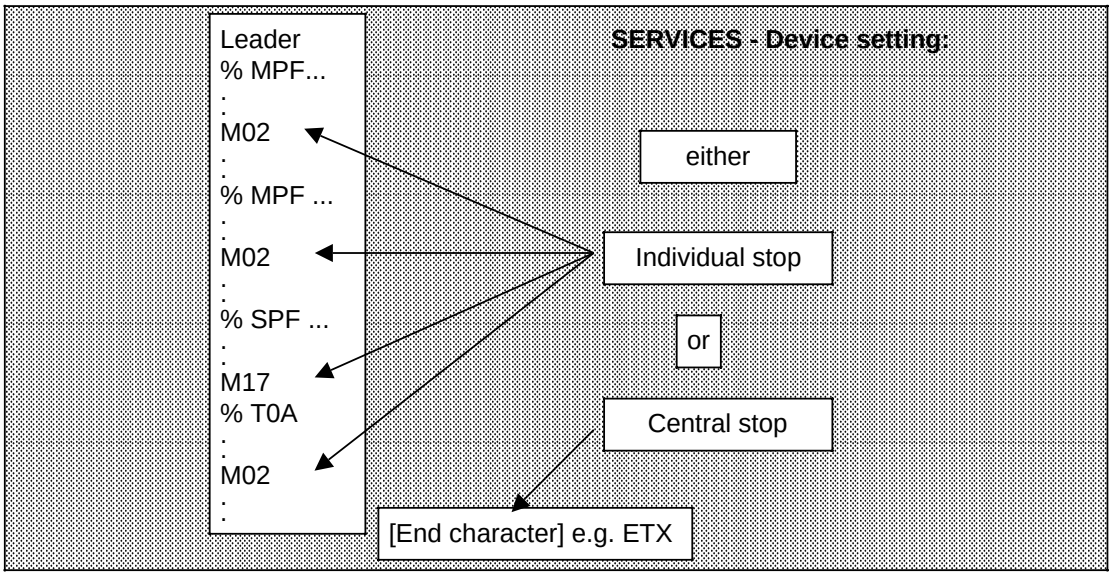
Shaft	%	MPF	1579	$\llcorner_F$

File comments

2.6.5 Read-in stop

The read-in process is halted by M02, M30 or M17 if no central end-of-transmission character has been defined, e. g. ETX (ASCII code: 03H).

If an end-of-transmission character has been specified in the device setting in the SERVICES area, the program or data block end (M02, M17, M30) will not stop transmission during reading-in of data. The read-in process is not halted until the central end-of-transmission character (ETX) is reached.



2.7 Input/output formats

%UMS $\llcorner_F$	**U**ser **m**emory **s**ubmodule
:Hexcode	Configured data
:Hexcode	Configured data
M02 $\llcorner_F$ (M30 $\llcorner_F$)	User memory submodule End of data block

%GIA	**G**earbox **i**nterpol**a**tion
T0=.. N0=.. : : N509=. : T45	T = Following axis or following spindle Nxyy = Parameter for leading axis or leading spindle (No. x max. 5)
M02 $\llcorner_F$ (M30 $\llcorner_F$)	End of data block

%IKA1	IKA configuration
N1 T0=.. T1=.. .. T44 : T0=.. T1=.. : N32	N = IKA number T = IKA parameter (variable) (up to software version 3 to T16, from software version 4 to T44)
M02 $\llcorner_F$ (M30 $\llcorner_F$)	End of data block

%IKA2	IKA curve pointer
N1 T5=.. T6=.. T55=.. N2 T5=.. T6=.. T55=.. N3 T5=.. T6=.. T55=.. : N32	N = Curve number T5, T6 = Curve pointer T55 = Curve status (software version 4 and higher)
M02 $\llcorner_F$ (M30 $\llcorner_F$)	End of data block

%IKA3	IKA points
N1 T7=.. T8=.. N2 T7=.. T8=.. : N4000T7=.. T8=..	N = Point number T7 = Input T8 = Output (max. 65 535 IKA points)
M02 L_F (M30 L_F)	End of data block

Program	**Leader**
%MPF 1234 L_F	Part program 1234 (**M**AIN **P**ROGRAM **F**ILE)
(Perform measurem.) N...L_F	Remark Part program
N...L_F	
M02 L_F (M30 L_F)	End of part program

%RPA 0...4 L_F	**(R P**ARAMETERS **A**CTIVE**)** Channel N**o**. (0 = central R parameters)
R...=...L_F R...=...L_F	(1...4 channel-specific) Parameter numbers with value assignments
M02 L_F (M30 L_F)	R parameters End of data block

%SEA L_F oder **SEA1** L_F	**General setting data (SE**TTING DATA **A**CTIVE**)**
N...=...L_F N...=...L_F	Addresses with value assignments (0...9, 2000...2003, 3000...3171, 4000...4033, 5000...5771)
M02 L_F (M30 L_F)	Setting data End of data block

%SEA4 1...4 $\llcorner_F$	**Cycle setting data** (channel-specific) **(SE**TTING DATA **A**CTIVE **4)**
N0 : N99	System data
N400 : N499	User data
N800 : N849	System bits
N900 : N949	User bits
M02 $\llcorner_F$ **(M30** $\llcorner_F$**)**	Cycle setting data End of data block

Subroutines	**Leader**
%SPF1234 $\llcorner_F$	**Subroutine** (**S**UB **P**ROGRAM **F**ILE)
(Bohrzyklus)...$\llcorner_F$ N1...$\llcorner_F$	Remark Subroutine
N2...$\llcorner_F$	
M17 $\llcorner_F$	End of subroutine

%TEA1 $\llcorner_F$	**NC machine data (TE**STING DATA **A**CTIVE **1)**
N...=...$\llcorner_F$ N...=...$\llcorner_F$	Machine data
M02 $\llcorner_F$ **(M30** $\llcorner_F$**)**	Machine data End of data block

%TEA2 $\llcorner_F$	**PLC machine data (TE**STING DATA **A**CTIVE **2)**
N...=...$\llcorner_F$ N...=...$\llcorner_F$	Machine data
M02 $\llcorner_F$ **(M30** $\llcorner_F$**)**	Machine data End of data block

%TEA3 $\llcorner_F$	**Drive machine data (TE**STING DATA **A**CTIVE **3)**
N...=...$\llcorner_F$ N...=...$\llcorner_F$	Machine data
M02 $\llcorner_F$ (M30 $\llcorner_F$)	Machine data block end

%TEA4 $\llcorner_F$	**Cycle machine data (TE**STING DATA **A**CTIVE **4)**
N...=...$\llcorner_F$ N...=...$\llcorner_F$	Machine data 0 = central 1...4 channel-specific
M02 $\llcorner_F$ (M30 $\llcorner_F$)	Cycle machine data End of data block

%TOA 1...4 $\llcorner_F$	**Tool offsets (T**OOL **O**FFSET **A**CTIVE**)** **per TO area (MD)**
D1 P0=...P1=...P9=...$\llcorner_F$ D2 P0=...P1=...$\llcorner_F$	Tool offsets (number of parameters corresp. to MD)
M02 $\llcorner_F$ (M30 $\llcorner_F$)	Tool offsets End of data block

%ZOA 0 $\llcorner_F$	**Settable zero offsets (Z**ERO **O**FFSET **A**CTIVE**)**
G154 X=...Z=...L_F	1st to 4th settable offset coarse (axis-specific)
:	
G157 X=...Z=...$\llcorner_F$	
G254 X=...Z=...$\llcorner_F$	1st to 4th settable offset fine (axis-specific)
:	
G257 X=...Z=...$\llcorner_F$	
M02 $\llcorner_F$ (M30 $\llcorner_F$)	Zero offset End of data block

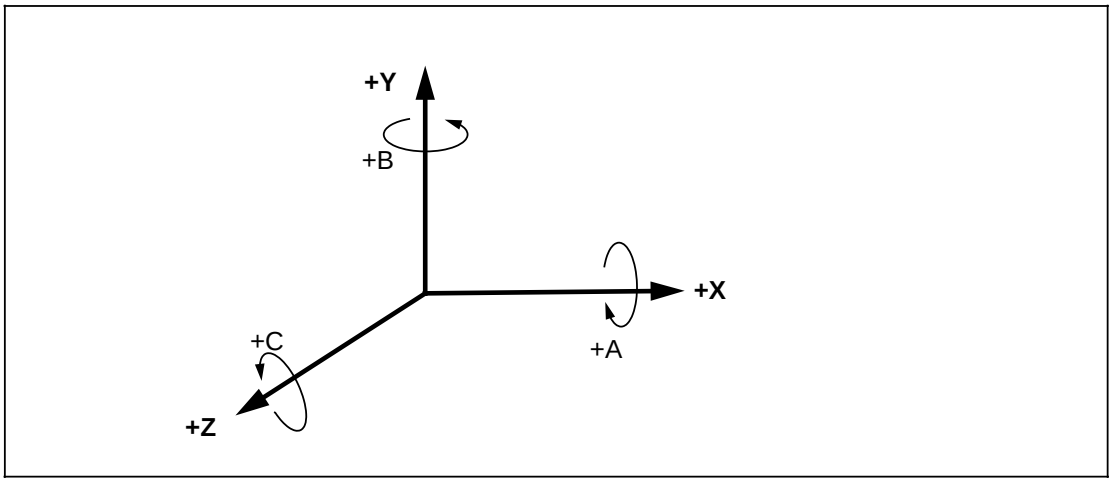
END OF SECTION

3 Path Information

3.1 Coordinate systems

3.1.1 Machine coordinate system

ISO Standard 841, DIN 66217, states that right-handed rectangular Cartesian coordinate systems are used for machine tools.



The preferred axis assignment for turning machines is: Main axes X and Z
The preferred axis assignment for milling machines is: Main axes X, Y and Z.

The preferred assignment of rotary axes to main axes is summed up in the following table:

Main axis	Rotary axis
X (U)	A
Y (V)	B
Z (W)	C

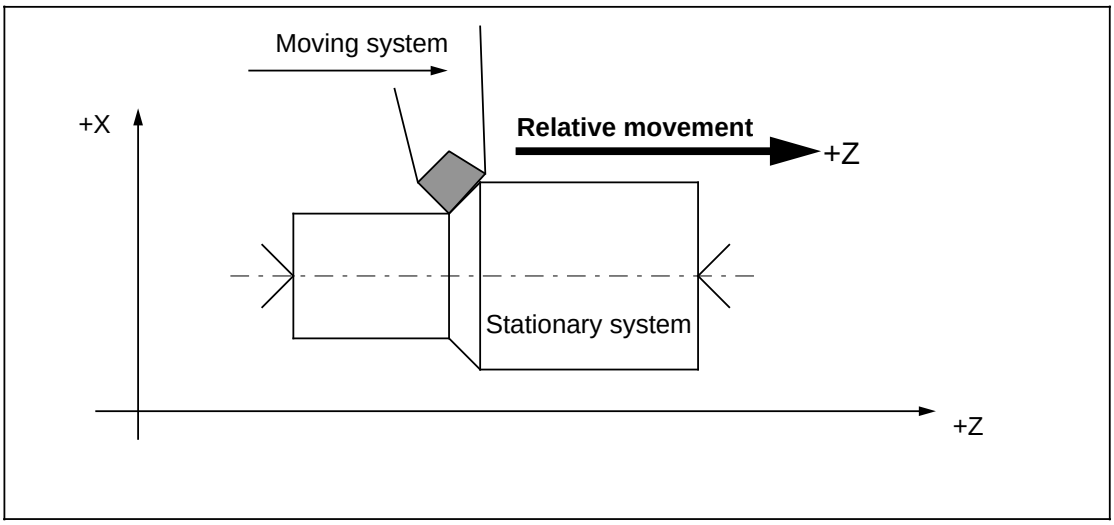
Extended addressing should be used for parallel axes (e.g. Z1=, Z2=)

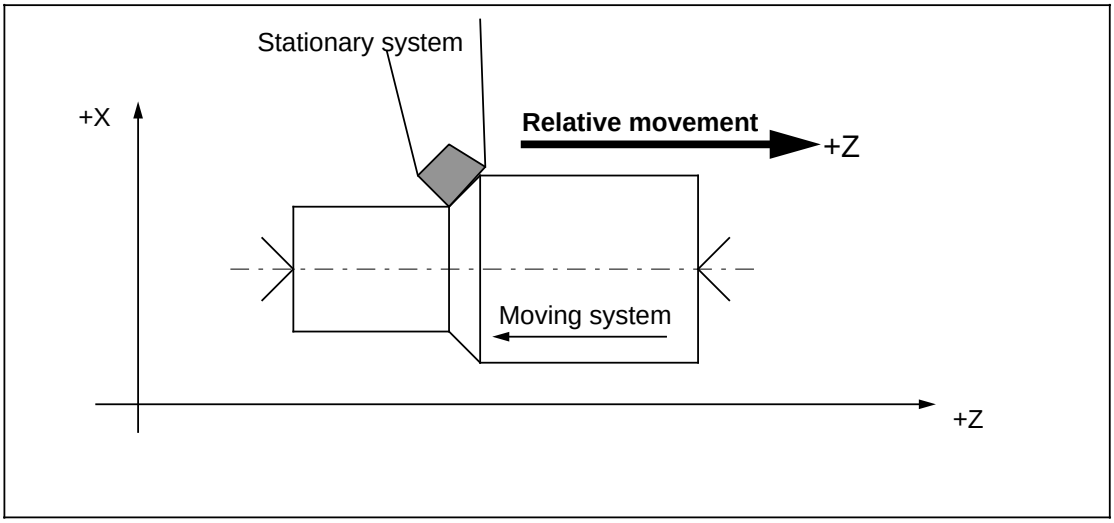
Note:

Address letters D, P and R **cannot** be used as an axis address.

Axis addresses for the main axes and rotary axes are freely selectable via the machine data.

The programmed axis movement **always** refers to the relative movement between tool and workpiece.

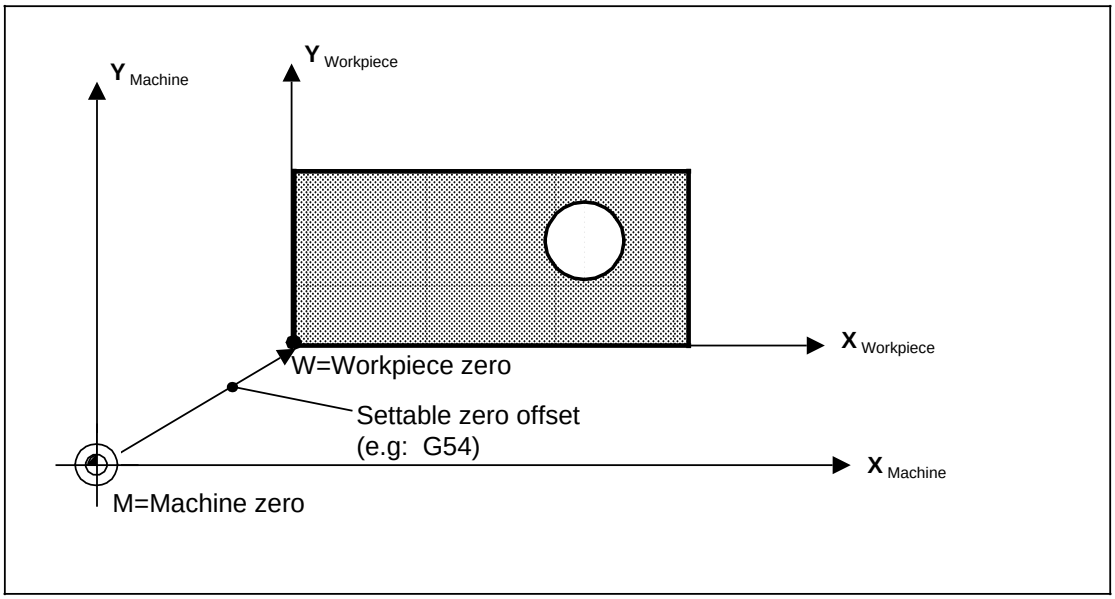




3.1.2 Workpiece coordinate system

Depending on the type of machining, either two or three dimensional workpiece geometry is programmed in the workpiece coordinate system.

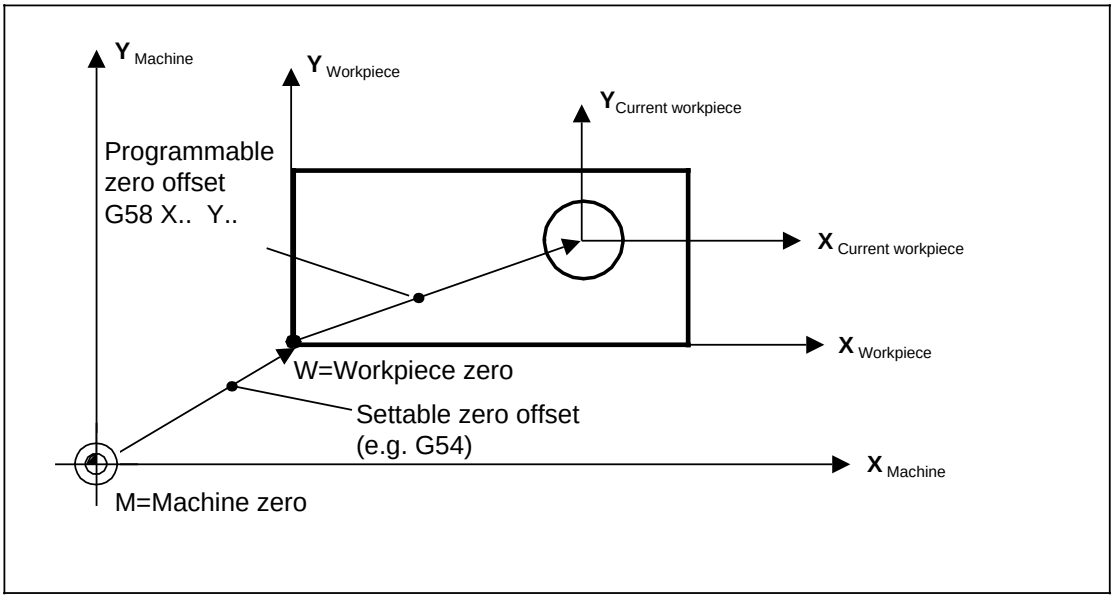
The workpiece clamping is usually defined in the settable zero offset and/or the settable coordinate rotation (inclined clamping). This defines the position of the workpiece coordinate system in relation to the machine coordinate system when the workpiece is clamped on the machine.



The workpiece zero is defined in the program. The position of the part and the related zero offsets are usually established in setting up operations.

3.1.3 Current workpiece coordinate system

If the programmer finds it easier to continue his geometric descriptions from a zero point other than the one originally selected, he can define a new zero point (programmable zero offset). Its position is programmed with reference to the original workpiece zero, the clamping position.



The current workpiece coordinate system can be changed in size, turned or mirrored with reference to the previous system.

3.1.4 Examples with reference points

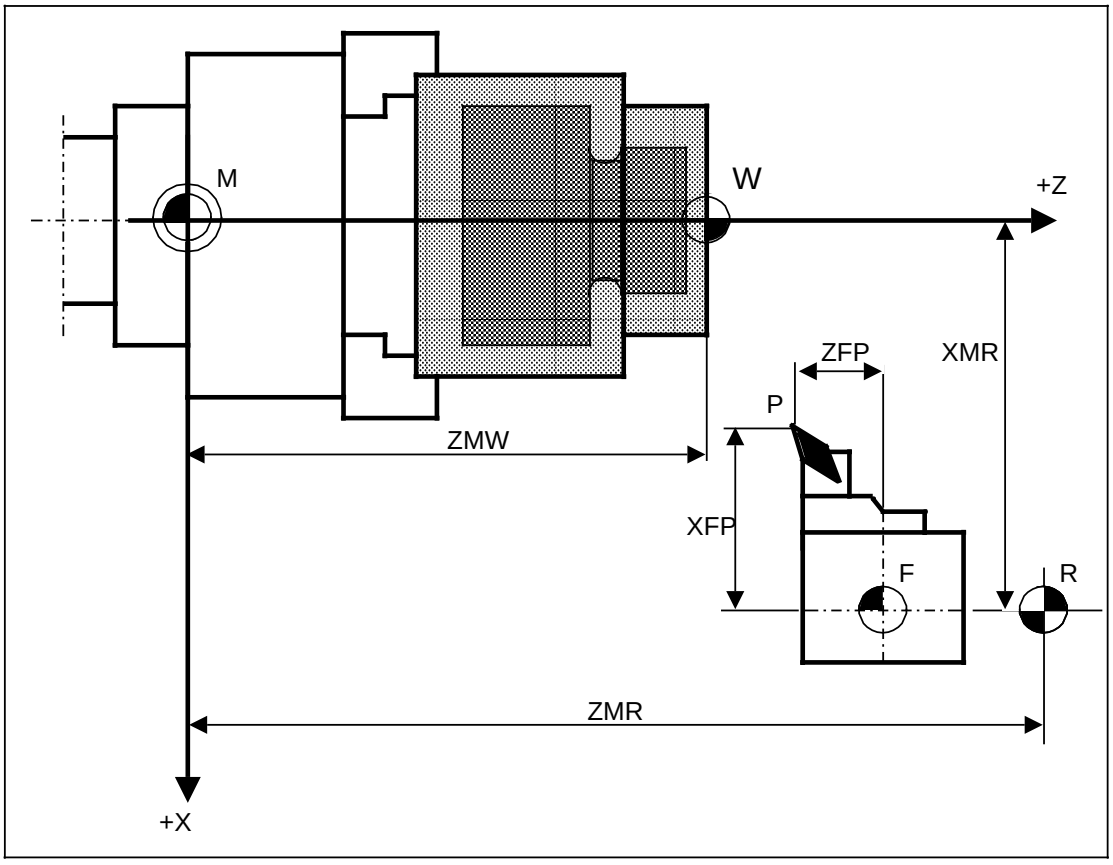
The zeros and various reference points are defined on all numerically controlled machine tools.

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

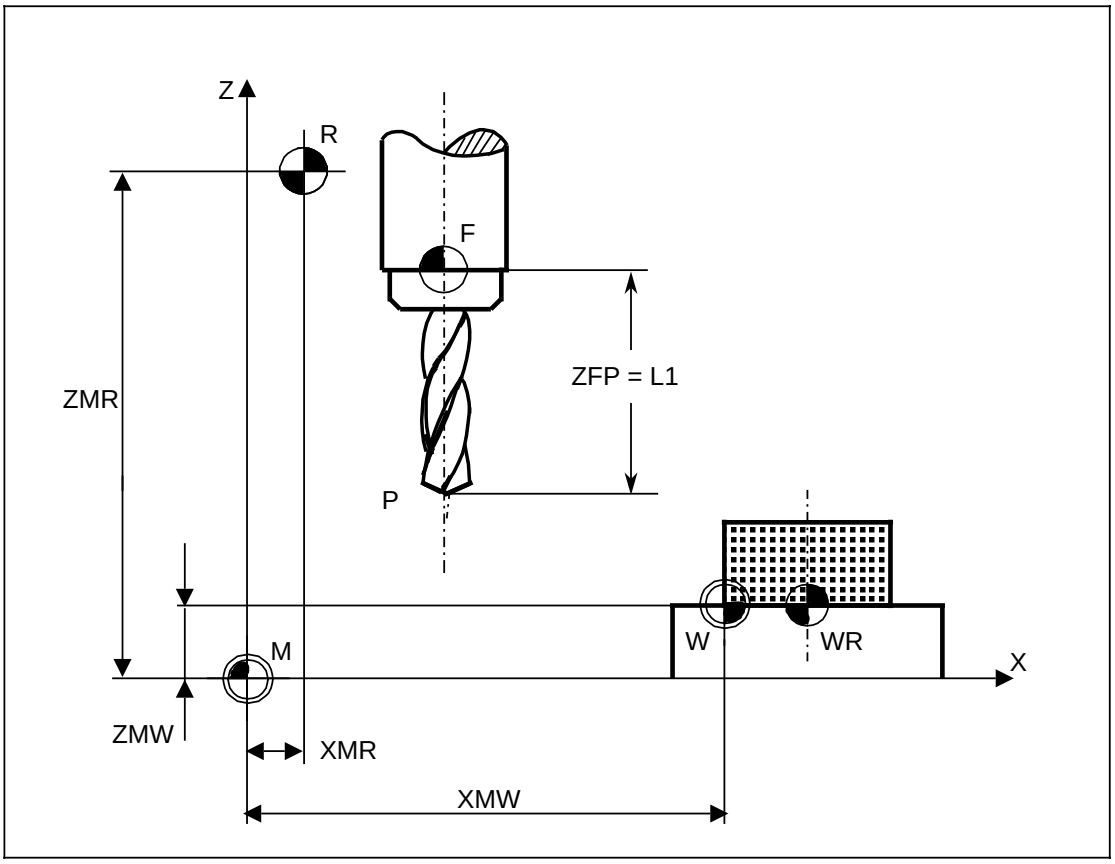
The **workpiece zero W** is the zero defined for programming the workpiece dimensions. It can be freely selected by the programmer. The relationship to machine zero is defined by the zero offset.

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.

The reference point coordinates are defined axis-specifically in the machine data.



Example: Turning machine (machining in front of the turning centre)



Example: Reference points on the drilling plane

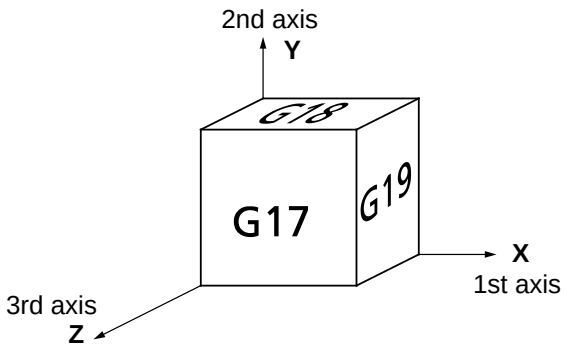
P	Tool setting point
M	Machine zero
W	Workpiece zero
WR	Workpiece reference point
R	Machine reference point
F	Tool carrier reference point (slide reference point)
XMR, ZMR	Reference point coordinates
XMW, ZMW	Zero offsets
XFP, ZFP	Tool geometry

3.1.5 Machining planes (G17/G18/G19)

Plane	Axis assignment	Axis address (preferred assignment)
G17	1st axis – 2nd axis	X – Y
G18	1st axis – 3rd axis	Z – X
G19	2nd axis – 3rd axis	Y – Z

Freely selectable axes can be assigned to planes G17, G18 and G19 via machine data.

If channel-specific, the plane reset position can be defined in the machine data.



3.1.6 Flexible plane selection (G16)

G16 allows the plane to be programmed independently of the MD (application of particular interest for tool offsets, see Section entitled "Tool Offsets").
Up to 5 axes can be programmed.

Example:

```
G16 X1= Y1= Z1±
```

3.2 Positional data/axis types/preparatory functions

3.2.1 Positional data

The positional data comprises an axis address and a numerical value which describes the path of the addressed axis.

The following addresses can be used for axis addresses: A, B, C, E, Q, U, V, W, X, Y and Z. Extended address notation can also be used (e.g. X1 = ..., P [Y1]. The axis addresses name the physical axes. They are also used to define the coordinate axes of a fictitious coordinate system which describes the geometry of the workpiece.

One axis cannot be addressed in several channels simultaneously.

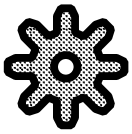
If a sign is used, it must be entered between the address and the numerical value (e.g. X-50) or between the extended address and the numerical value (e.g. X1=–50, P[Y1]–50).

3.2.2 Axis types (linear, rotary, contouring and simultaneous axes)

Linear and rotary axes

Linear and rotary axes can be programmed either as part of a contouring group or as simultaneous axes.

Every axis can be defined as a rotary axis in the machine data. It is possible to define several axes as rotary axes at any one time.



Contouring axes

Contouring axes are interpolated as a function of time and are therefore interdependent. The contouring axes reach the end point at the same time. Up to 5 interpolating axes can be programmed in one block and traversed. These axes describe the geometry two or three-dimensionally within the workpiece coordinate system. The path produced by these axes is traversed at the programmed path feedrate (F).

Simultaneous axes (SW 3 and higher)

The traversing movement of the simultaneous axis bears no relation to the traversing movement of the contouring axes or other simultaneous axes. Simultaneous axes do not influence each other in terms of speed. Up to 5 simultaneous axes can be programmed and traversed individually or together with 5 other contouring axes in any one block.

Block change takes place when all the simultaneous and contouring axes have reached their end point.

A simultaneous axis can only addressed together with an axis-specific feedrate, i.e. every simultaneous axis is governed by a G01 function even if this command has not been programmed. The programmed end point is approached with 1-axis linear interpolation (see Section entitled "Simultaneous axes").

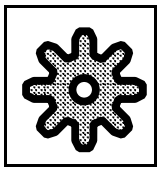
3.2.3 Preparatory functions

In order to start positioning, the positional data must be supplemented by the preparatory function (G function) and the feedrate (F). The preparatory functions describe the type of machine movements, the type of interpolation and the type of measurement.

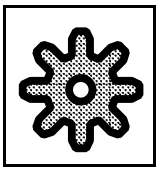
The G functions are divided into G groups (see Section entitled "Program key").
A program block may contain only one function from each G group (exception: G90***/***G91).

The G functions are either modal 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 non-modal.

The resets take effect after powering up the control, and after reset or at the end of the program. They need not be programmed.



The G command to be set initially can be defined in the machine data (see breakdown of G groups).



3.3 Programmable working area limitation (G25/G26)

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

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

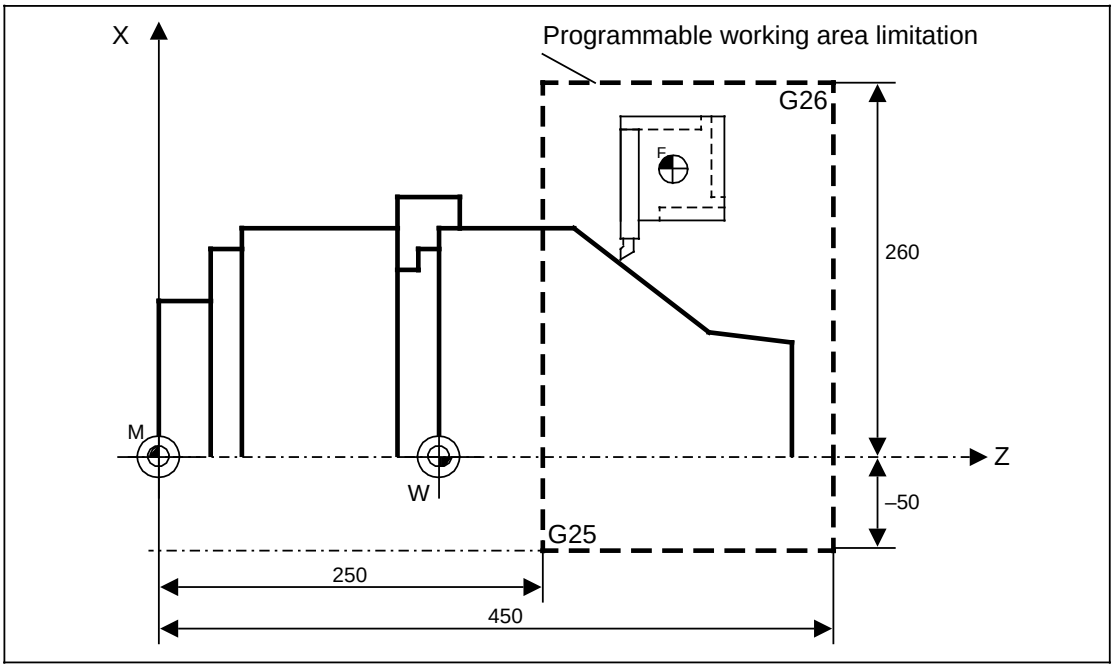
Can also be activated via machine data for the JOG mode.

Programmable working area limitation is called using G25 and G26:

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

```
N10  G25  X–50   Z250  LF
N20  G26  X260  Z450  LF
```

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



Example of working area limitation for turning machine

3.4 Dimension systems (G90/G91/G68)

The traversing movement to a particular point in the coordinate system can be described by means of absolute or incremental position data. A ”zero offset” is allowed for in both absolute and incremental programming.

It can be specified by machine data whether the radius or the diameter is to be programmed with G90 and/or G91 (machining by turning).

3.4.1 Absolute dimension (G90)

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.

Notes:

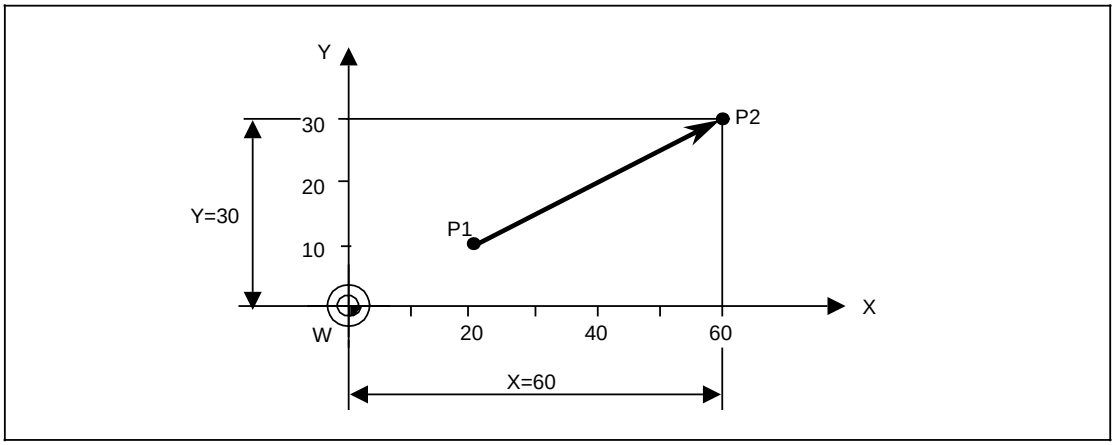
Up to SW 5.2, after the following functions have been programmed, the axes concerned must be programmed once with G90. As from SW 5.4, after the functions have been programmed it is also possible to traverse with incremental dimensions (G91) provided no coordinate-rotation/scale-factor is active for the axis or axes concerned:

- G74 (reference point approach per program) *
- @720 (in-process measurement)
- G200 (axis actual value synchronization)
- Delete distance-to-go
- After initiating C axis mode
- Transformation selection and deselection

Example:

The tool is at position (P1) X = 20, Y = 10 and is to move to position (P2) X = 60, Y = 30.
The input is: **`G90 X60 Y30`**

The tool moves to the programmed target position irrespective of its current position.



**) As from SW 4 with G91*

3.4.2 Incremental dimension (G91)

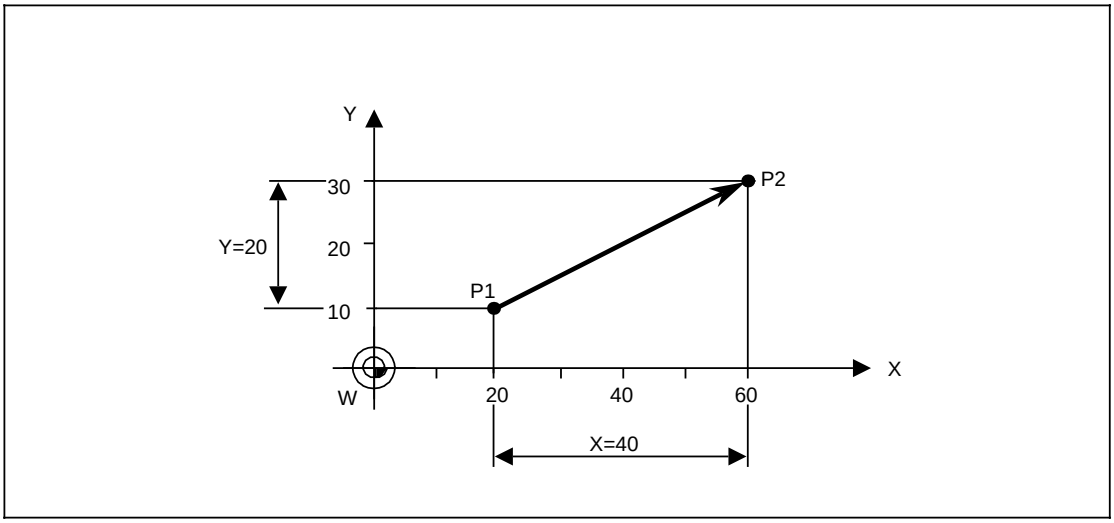
In the case of incremental data input, the numerical value corresponds to the path to be traversed. The sign indicates the direction of movement.

Example:

The tool is at position (P1) X = 20, Y = 10 and is to move to position (P2) X = 60, Y = 30.
The input is:

```
G91 X40 Y20
```

The tool moves on by the distance specified by the programmed values with reference to its current position.



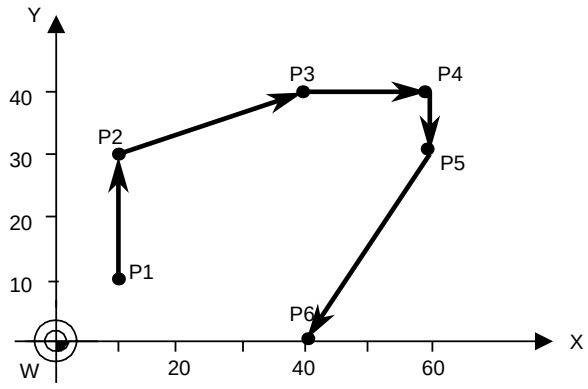
3.4.3 Switching between absolute dimension (G90) and incremental dimension (G91)

It is possible to switch between absolute and incremental position data input from block to block as desired.

Example:

The tool is at position (P1) X = 10, Y = 10 and is to move consecutively to the subsequent positions. The program contains absolute and incremental position data alternately. The input is:

```
G90 X10 Y30      (P2)
G91 X30 Y10      (P3)
G90 X60 Y40      (P4)
G91 X0 Y-10      (P5)
G90 X40 Y0       (P6)
```



3.4.4 Absolute dimension (G90) and incremental dimension (G91) mixed in one block

If the machine data are set you can program axes, polar coordinate angles (not contour definition angles) and interpolation parameters together in one block depending on G90/91.

3.4.5 Absolute dimension (G90) and incremental dimension (G91) with rotary axes

With absolute dimension programming (G90), the traversing range is ± 360.000 degrees. The sign defines the traversing direction to be used to approach the absolute position within one revolution.

With incremental dimension programming (G91) the traversing range is ± 99999.999 degrees. The sign defines the direction of movement.

Note:

A rotary axis must be programmed once with G90 before it can be programmed with G91. Otherwise the alarm "Wrong block structure" appears.

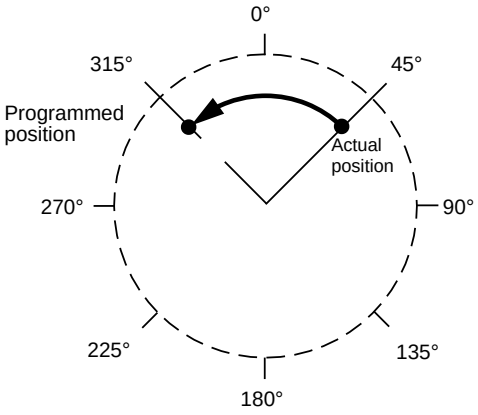
3.4.6 Rotary axis along shortest path (G68)

A setting in the machine data defines whether the function for the first axis programming is disabled. [1)]

In addition to absolute and incremental dimension programming, the rotary axis can be programmed in a third way. This is when the rotary axis is programmed to approach the programmed absolute position along the shortest path (positive or negative direction).

The traversing range that can be defined is limited to 0° to + 360°.

When a rotary axis is programmed with G90 in a part program for the first time, G68 automatically becomes active for this block. The automatic generation of this command can be deselected with G91 C0 LF. This also applies when a rotary axis is programmed with G90 for the first time after a block search or after G74 has been programmed.



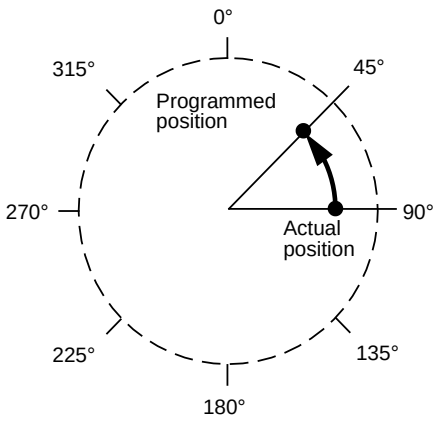
Example:

The effect of a software limit switch in the case of rotary axes programmed to approach a position along the shortest path:

```
%MPF
N010 G0 G68 B45 LF
N020
```

Software limit switch minus at 0°
Software limit switch plus at 270°

Actual position of rotary axis 90°
Programmed position of rotary axis 45°.



1) *Software version 5 and higher*

If alarm 2065 "Programmed position behind software limit switch" is triggered.

Reason: Software limit switch check does not read G68 command, i.e. the control assumes that position 45° is not approached along the shortest path and is therefore outside the software limit switch.

Remedy:

N010 G0 G68 B–45 L_F	By programming B-45°, position 45° is approached along the shortest path.

Example:

Programming with R parameters

Calculated R parameter R22=0

```
N10 ...
N20 ...
N30 ...
```

N40 G1 G90 F1000 A1=R22	The rotary axis turns to 0° in the positive direction
N50 G1 G90 F1000 A1=–R22	The rotary axis turns to 0° in the positive direction
N60 G1 G90 F1000 A1=–0+R22	The rotary axis turns to 0° in the negative direction

Note:

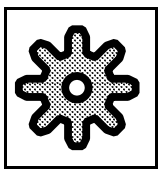
If a rotary axis is be turned to zero in the negative direction via R parameters, this must be programmed with **C=–0+Rxx**.

It is then possible to specify by means of the G commands G68**/**G90**/**G91 just how the control acts when programming for the first time.

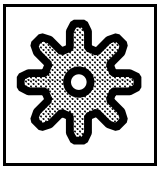
During block search, automatic generation of G68 is maintained.

3.5 Metric/inch (G71/G70)

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



The initial setting of the input system is determined by the machine data.



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).

G71**/**G70 must be programmed before the axes otherwise alarm 3006 ”Wrong block structure” is triggered.

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

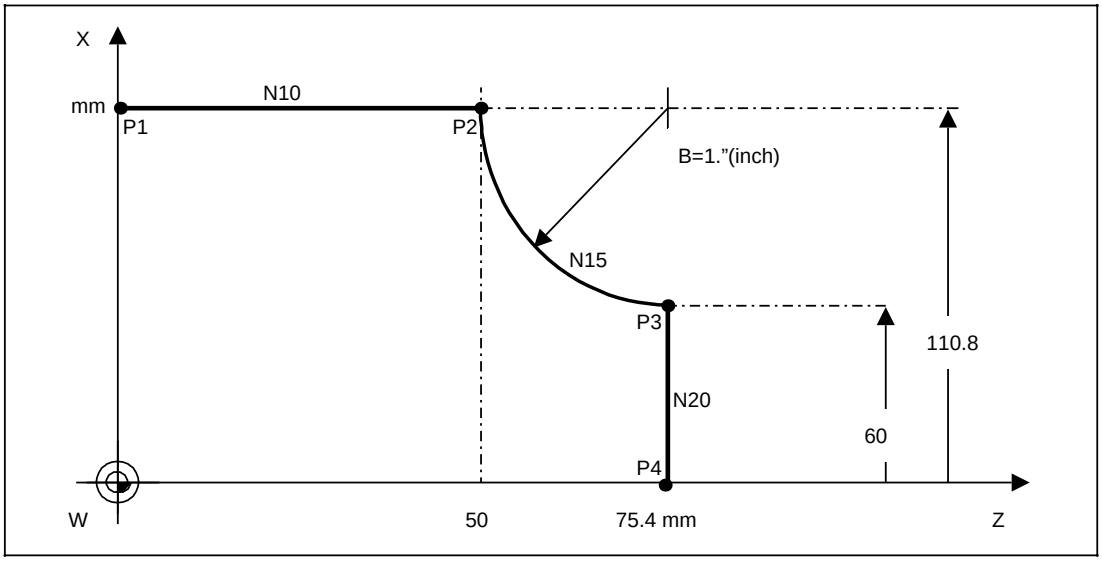
- Actual-value display (including setpoint**/**actual value difference)
- Zero offset
- Feedrate**/**cutting speed G94**/**G95
- Tool offset

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

- Position data X, Y, Z
- Interpolation parameters I, J, K
- Chamfers**/**radii U-**/**U (B, B-)
- Parameters related to position data, interpolation parameters and chamfers**/**radii.
- Programmable zero offset (G58**/**G59).

Example:

G71 - Initial setting (metric)

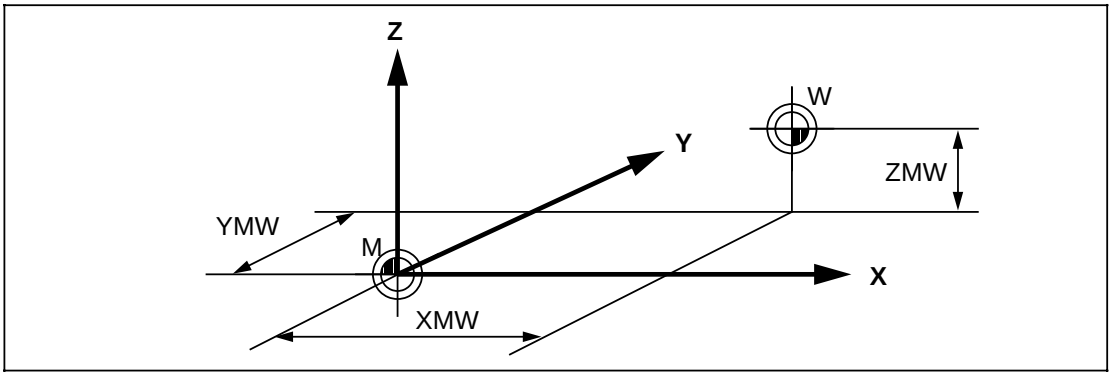


Input in inches initial setting G71

```
N05  (Shaft)  LF
N10  G91  Z50.  LF                    (P2)
N15  G03  G70  X-1  Z1  K1  I0  LF    (P3)
N20  G01  G71  X-30.  LF              (P4)
N30
```

3.6 Zero offset (G54 ... G59)

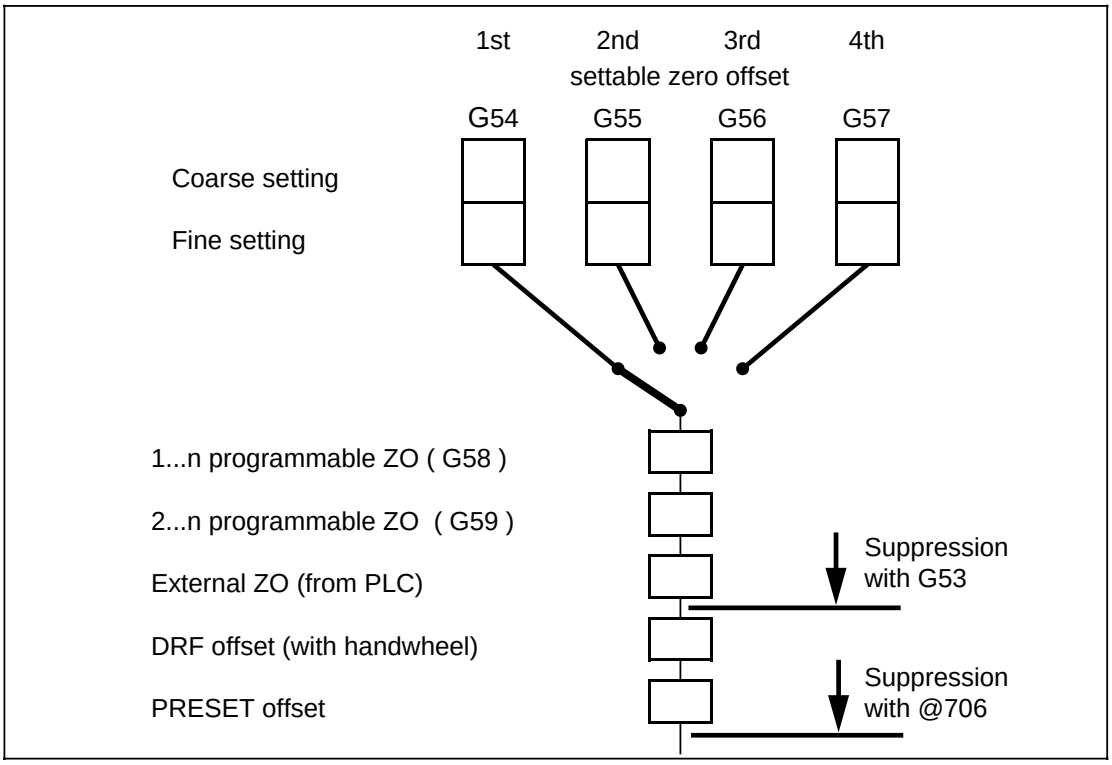
Zero offset is the distance between the workpiece zero W (on which the dimensions are based) and the machine zero M.



Zero offset in 3 axes

The following types of zero offset can be activated:

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



Sum of zero offsets

The sum of the zero offsets results from:

Settable zero offset (G54 ... G57) + external zero offset (from PLC) + programmable zero offset (G58, G59).

3.6.1 Settable zero offset (G54/G55/G56/G57)

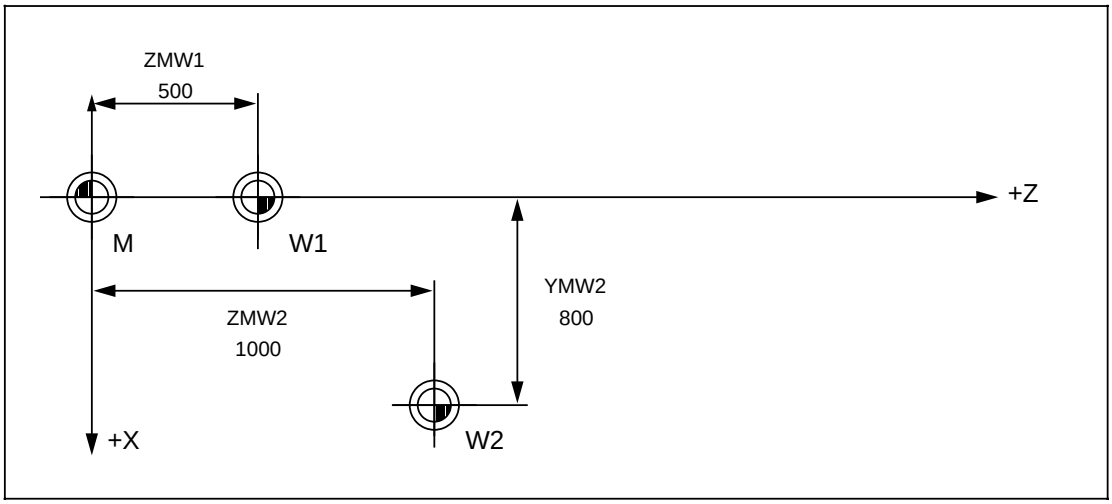
The settable zero offset values for each axis can be entered in the control via the operator panel or via the universal interface.

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, each with two settings, can be selected for the individual 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.



Settable zero offset

Input of the settable zero offset via the universal interface:

```
% ZOA LF
G154 X = 250 Y = 280 LF
G155 X = 220 Y = 250 LF
.
G157 X = 320 Y = 350 LF
```

Settable ZO (coarse)

```
G254 X = 0.1 Y = 0.3 LF
:
G257 X = 0.4 Y = 0.5 LF
M02 LF
```

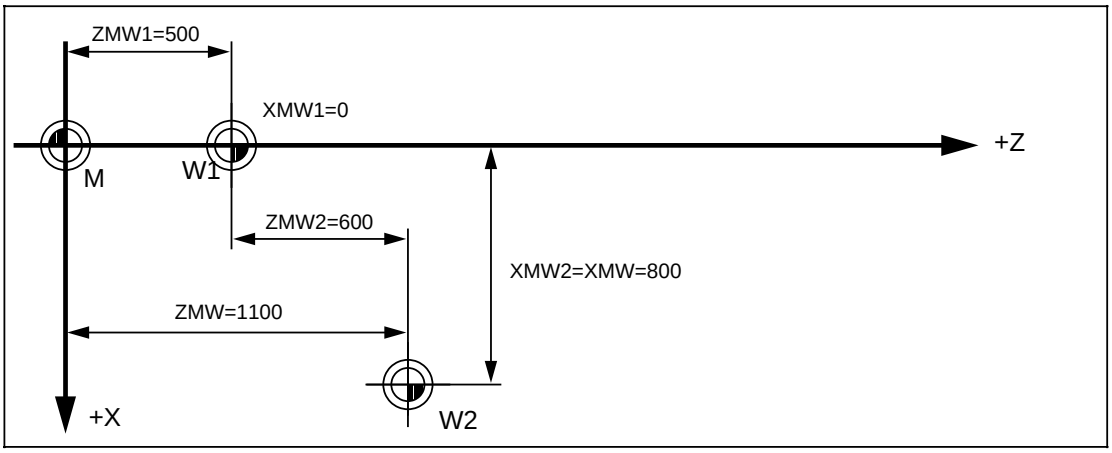
Settable ZO (fine)

For reasons of compatibility, the format **G54 X = 250 LF** can be read in, the values then being entered into the settable ZO (coarse).

3.6.2 Programmable zero offset (G58/G59)

An additional zero offset or several programmed zero offsets 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.

The signed sum of all zero offsets is used to calculate the path.



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

A machine data can be set to define whether mixed programming (absolute dimensioning/incremental dimensioning) is possible.

A setting in the machine data defines whether the offset is always absolute (G90) or whether G90 must be taken into account. In this case mixed programming with G90/G91 is also possible.

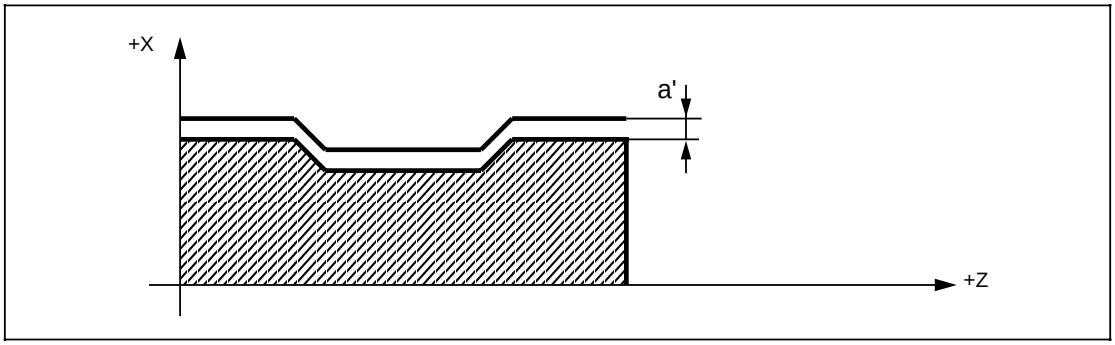
Programming:

```
N30
N35  G54  LF
N40  G59  Z600  X800  LF
N45
```

A block containing G58 or G59 must not include any functions other than the zero offsets. Up to 5 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 Xa' LF`
Cancel: `N.. G59 X0 LF`

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

If incremental dimensioning is active (G91, depending on MD), programmable zero offset (G58/G59) is not referred to workpiece zero absolutely but additively to the last offset to be programmed.

Example:

Programmed zero offset with active incremental dimension (depending on MD).

```
N5 G0 X0  LF                (Position  X0)
N10 G91  X20  LF            (Position  X20)
N15 G58 X10.111  LF         Activation of programmed zero offset
N20 G90  X0  LF             (Position  X10.111)
N25 G91 X20  LF             (Position  X30.111)
N30 G58  X20.333  LF
N35 G90 X0  LF              (Position  X30.444)
N40 M30  LF
```

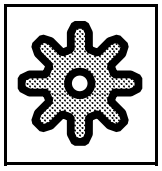
3.6.3 Several programmable zero offsets 1)

Several programmable zero offsets (linear and angle of rotation) can be programmed in a subroutine and then activated via the part program with G98**/**G59.

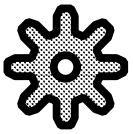
The sequence **G.. L.. K..** used in the part program must be maintained. The subroutine number must be defined under address L. The block number of the zero offset programmed in the subroutine must be defined under address K.

The zero offset becomes active in the subsequent block.

The subroutine block used for the zero offset must only contain axis positions.



A setting in the machine data defines whether G90/G91 is included for axes with G95/G98.



Example:

Several programmable zero offsets in a subroutine

```
%200                          SPF123
N010 ...                      N010 X100 Y50
N020 ...                      N020 X20 Y5 Z200
N110 G58 L123 K55             ...
N120 X130 Z60                 ...
N130 ...                      N55 X40 Z10
N200 ...                      ...
```

1) *Software version 5 and higher*

3.6.4 Cancelling zero offsets (G53)

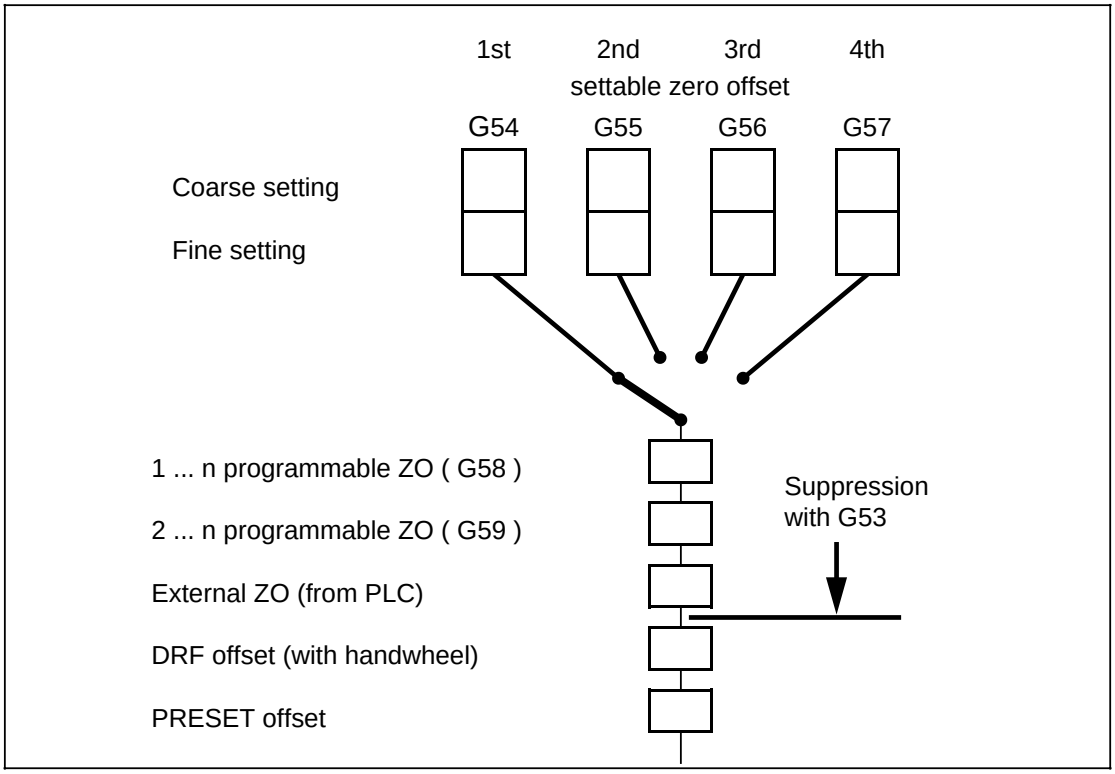
The machine tool manufacturer defines the G53 action using the NC machine data.

With G53 the zero offsets described below under ”Action a” and ”Action b”are suppressed non-modally:

Action a

Non-modal suppression of:

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

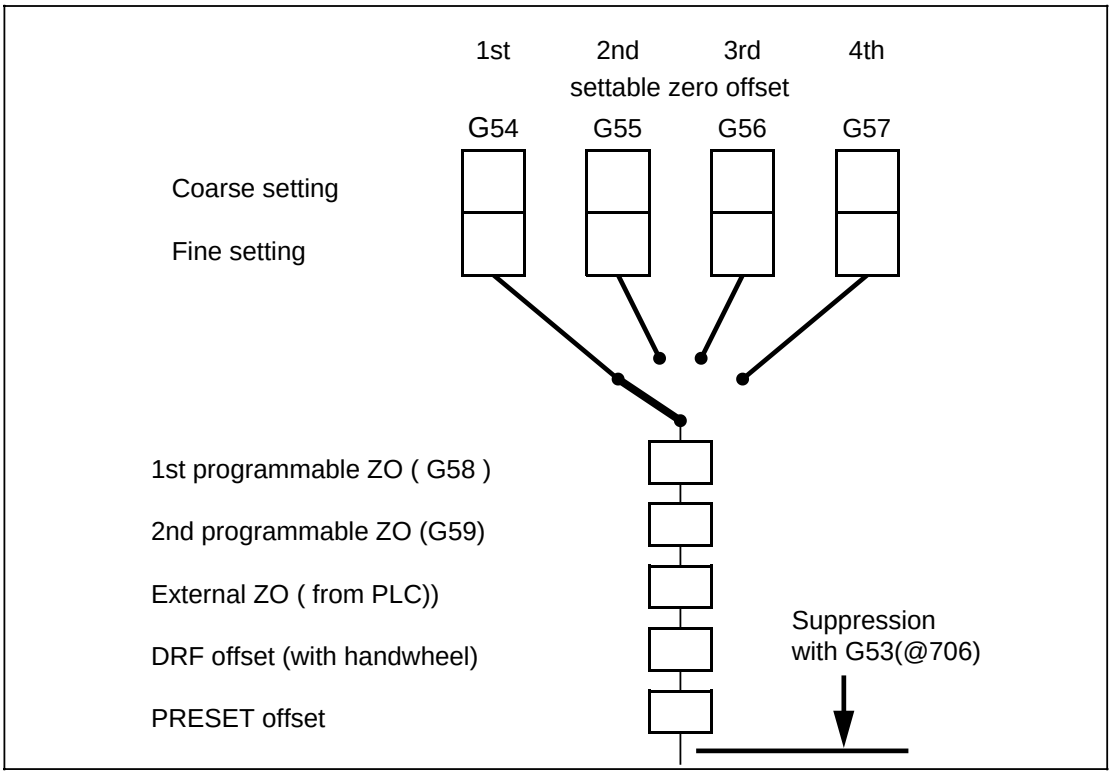


Suppression of zero offsets

Action b

Non-modal suppression 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



Suppression of zero offsets

Reference to machine zero

```
N30  D0  LF
N35  G53  X...  Y...  LF
```

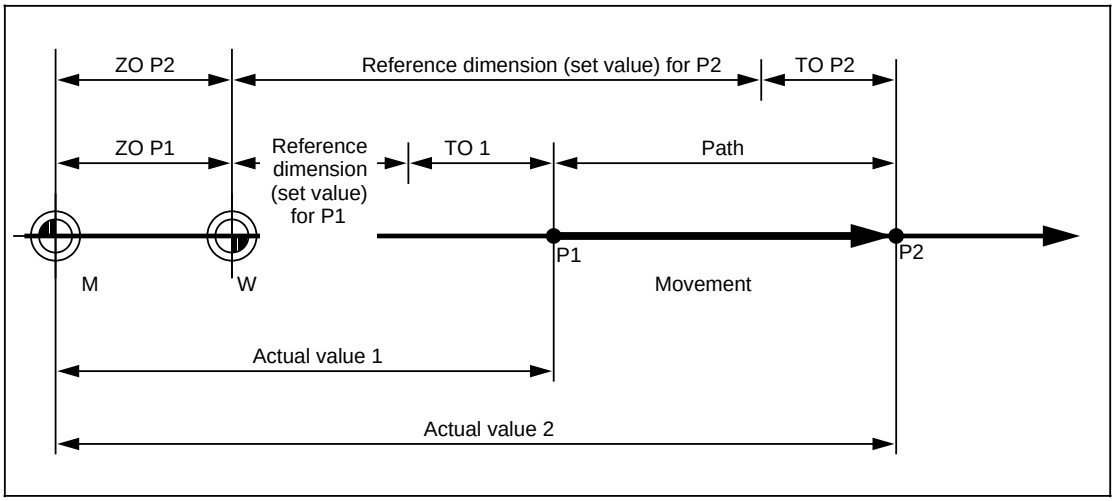
(Cancellation of tool length compensation)
(Cancellation of all zero offsets and travel to position in machine system)

3.6.5 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 **incremental position data** input is used, the zero offset is incorporated normally in the first block.

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**

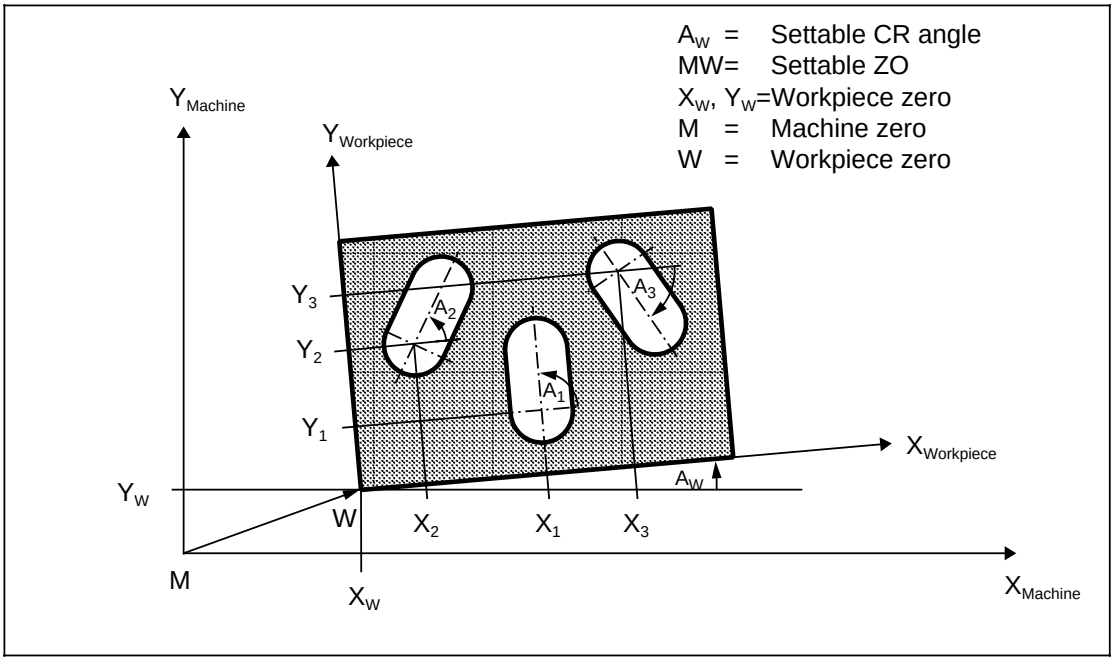
Path = incremental position data + ZOP2 - ZOP1 +TOP2 - TOP1.

3.7 Coordinate rotation (G54 ... G57, G58/G59 A..)

By coordinate rotation the coordinate system of the workpiece can be defined independently of the coordinate system of the machine. Thus, a workpiece does not have to be oriented along the machine axes. The **settable coordinate rotation** is intended for this purpose.

If several identical contours pointing in different directions have to be machined on one workpiece, you can use the **programmable coordinate rotation**.

Example: Three geometrical shapes are to be cut out which are identical but have different positions and orientations: *Settable and programmable coordinate rotation*

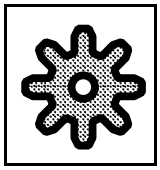


Settable and programmable coordinate rotation

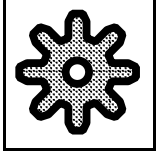
```
%  10
N005  G55                          ( XW, YW, AW, settable coordinate rotation )
N010  G00  F500  X..  Y..  Z..     ( Machine the outside edges of the workpiece )
N015  G01  ...
N020  ...
N025  G58  X1  Y1  A1              ( Select programmable ZO1 and CR1 )
N030  L10                          ( Subroutine e.g. for elongated hole machining )
N035 G58 X0 Y0 A0                  ( Deselect programmable ZO1 and CR1 )
N040  G58  X2  Y2  A2              ( Select programmable ZO2 and CR2 )
N045  L10
N050 G58 X0 Y 0 A0                 ( Deselect programmable ZO2 and CR2 )
N055  G58  X3  Y3  A3              ( Select programmable ZO3 and CR3 )
N060  L10
N065  M02
```

Note:

Before a new zero offset and coordinate rotation can be selected, the previously active zero offset and coordinate rotation must first be deselected.



The address letter for the input of the angle of rotation depends on the machine data.



3.7.1 Characteristics of coordinate rotations

- Coordinate rotations are channel specific.
- Coordinate rotations are active in the current plane.
- Coordinate rotations are suppressed modally if G53 was programmed.
- Circular interpolation must not be programmed immediately after coordinate rotation. Exception: arc centre and turning centre coincide.
- If tool radius compensation is active the angle of rotation can be changed but not the position of the coordinate zero.
- If an angle of rotation is changed with @ commands while G54 ... G57 are active, @715 ”STOP DEC until coordinate rotation” must first be programmed. If also the settable zero offset is changed at the same time, command @714 ”STOP DEC until buffer empty” must be used. Commands @714 ”STOP DEC until buffer empty” or @715 ”STOP DEC until coordinate rotation” must be in a block of their own.
- If the zero offset is altered after a coordinate rotation, it is calculated with reference to the rotated coordinate system.
- If a coordinate rotation was activated once with G58***/***G59 (total angle of rotation 0), it is only possible to change the plane or the set coordinate rotation with G54 to G57 if all the values programmed for a coordinate rotation and zero offset programmed under G58 and G59 are deselected beforehand, i.e. set to zero. Otherwise the alarm 2087 ”Coordinate rotation***/***zero offset not allowed” is output.
- Zero offsets programmed with G58***/***G59 outside the coordinate rotation plane do not have to be deselected.

Note:

Block search or program start must always stand at the beginning of a block containing all the necessary information. This especially applies to blocks written in abbreviated form (that is, only the axes which are to be traversed are programmed).

3.7.2 Settable coordinate rotation (G54 ... G57)

The values can be entered in various ways:

- Via softkeys and keyboard in the input display ”Angle of rotation”
- There is no distinction between coarse and fine setting
- The coordinate rotation is defined using program parameters
- Input of the value for angle of rotation “A” keyboard
- The angle of rotation “A” can be input via the universal interface for G54 ... G57 or for G154 ...G157

The rotation is activated by G54 ... G57 in the part program. If a zero offset and a coordinate rotation are set under the settable coordinate rotation, first the zero offset is performed and then the coordinate system is rotated through the set angle in the new zero. The angles of rotation set are not deleted by program end (M02, M30), program abort or switching the control OFF and then ON again.

3.7.3 Programmable coordinate rotation (G58 A.. /G59 A..)

The rotation is activated by G58/G59 under address A in the part program. A zero offset can also be contained in the command. If a zero offset and a coordinate rotation are programmed under the programmable coordinate rotation, first the zero offset is performed and then the coordinate system is rotated through the set angle in the new zero.

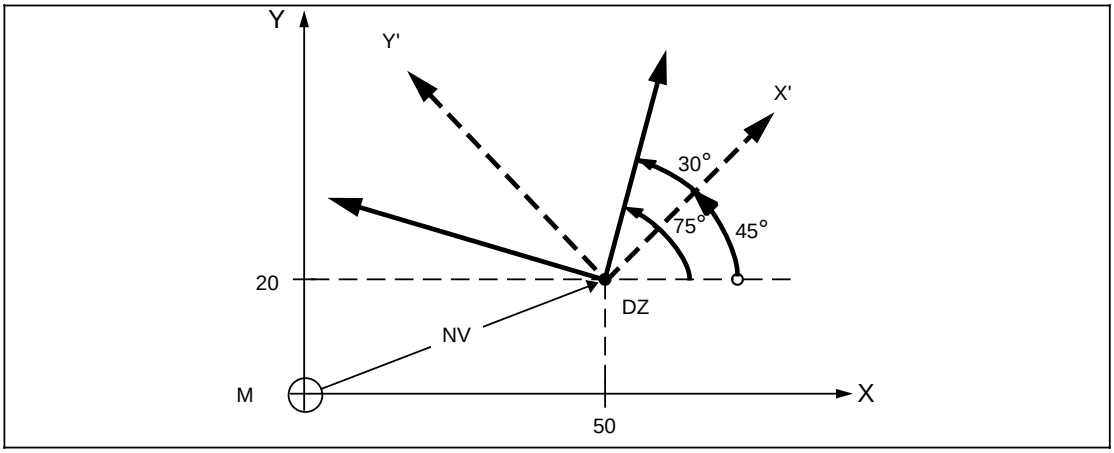
If several rotations (if necessary combined with zero offsets) are set, the following coordinate system is always based on the previous one. This also applies to a combination of settable and programmable rotations.

Programmed angles of rotation can be programmed in absolute or incremental dimensions. They are deleted by program end (M02, M30), program abort or switching the control OFF and then ON again.

Example:

```
N...
N10...  G90  G58  X50  Y20  A45  LF
N15...  G91 G58 A30  LF
```

Angle is added to the angle already programmed in block N10 (sum = 75°)



Example 1:

```
:
G17                    Select plane X-Y-Z
G58 X... Y... A...     Select programmable ZO and CR under G58
G59 X... Y... A...     Select programmable ZO and CR under G59
:
G58 X0  Y0  A0         Deselect ZO and CR under G58
G59 X0  Y0  A0         Deselect ZO and CR under G59
G18                    Select the new plane
```

Example 2:

```
:
G54 ...                Select settable zero offset and coordinate rotation under G54 in plane X-Y
G58 X... Y... A...     Select programmable zero offset and coordinate rotation under G58
G59 X... Y... A...     Select programmable zero offset and coordinate rotation under G59
:
G58 X0   Y0   A0       Deselect zero offset and coordinate rotation under G58
G59 X0   Y0   A0       Deselect zero offset and coordinate rotation under G59
```

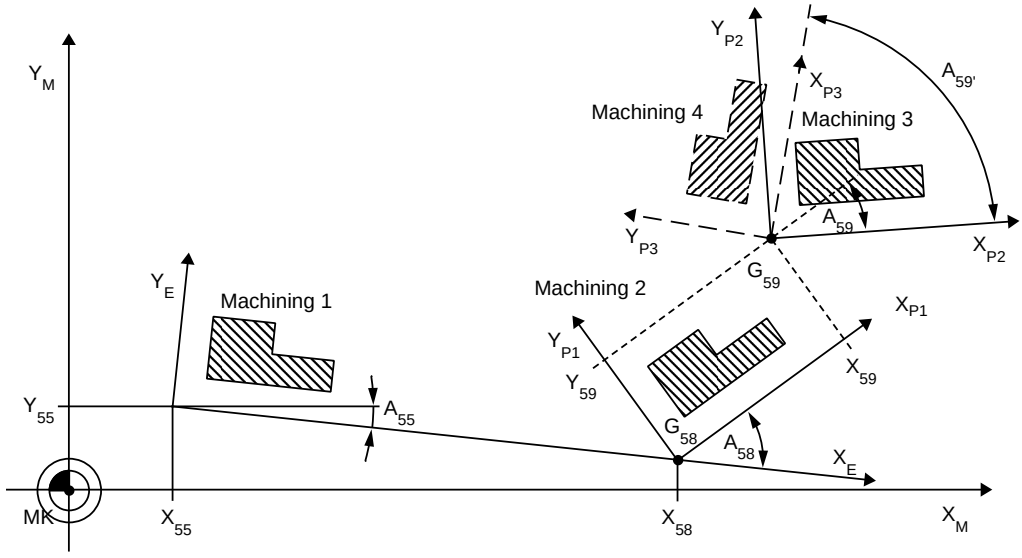
After a plane has been changed, both axes of rotation must be programmed in the next axis travel block of the new axes of rotation. If an axis is not programmed, incorrect calculations result because the control supplies the axis which has not been programmed with the corresponding ordinates of the old plane which has already been deselected.

Example 3:

```
:
G54 ...
G16 X Z Y              Select the X-Z plane
G58 X... Y... A...     Select the programmable zero offset and coordinate rotation under G58
G59 X... Y... A...     Select the programmable zero offset and coordinate rotation under G59
:
G58 X0   Y0 Z0 A0      Deselect zero offset and coordinate rotation under G58
G59 X0   Y0 Z0 A0      Deselect zero offset and coordinate rotation under G59
G0   Y50Z              Both axes of rotation must be programmed
:
```

Note:

Several programmable zero offsets can be programmed with G58**/**G59 in subroutines (see Section 3.6.3).



MK	Machine coordinate system
X_E/Y_E	Settable coordinate rotation
X_{P1}/Y_{P1}	1st programmable coordinate rotation
X_{P2}/Y_{P2}	2nd programmable coordinate rotation
	Geometry to be machined
	The contour is always the same. The angle of rotation differs.

Interaction of the various offsets and rotations

```
% 10
N005  G55                                  (Settable coordinate rotation)
N010  ..                                   (Machining 1, rotated through A55)
.
N070  G90  G58  X58  A58                   (1st coordinate rotation, absolute)
N075  ...                                  (Machining 2,  rotated through A55+A58)
.
N140  G59  X59  Y59  A59                   (2nd coordinate rotation, absolute)
N145  ...                                  (Machining 3,  rotated through A55+A58+A59)
.
N205  G91  G59  A59'                       (3rd coordinate rotation, incremental)
N210  ...                                  (Machining 4,  rotation through A55+A58+A59+A59')
```

Special case: programmable coordinate rotation (CR)

The control refers to the current actual value when calculating the end position.

If only one coordinate is programmed and a block search with calculation is executed or the program is aborted with "Reset" and restarted, it is no longer certain that the programmed positions will be approached.

Example:

A coordinate rotation of 30° was programmed.
Start position: X0, Z0

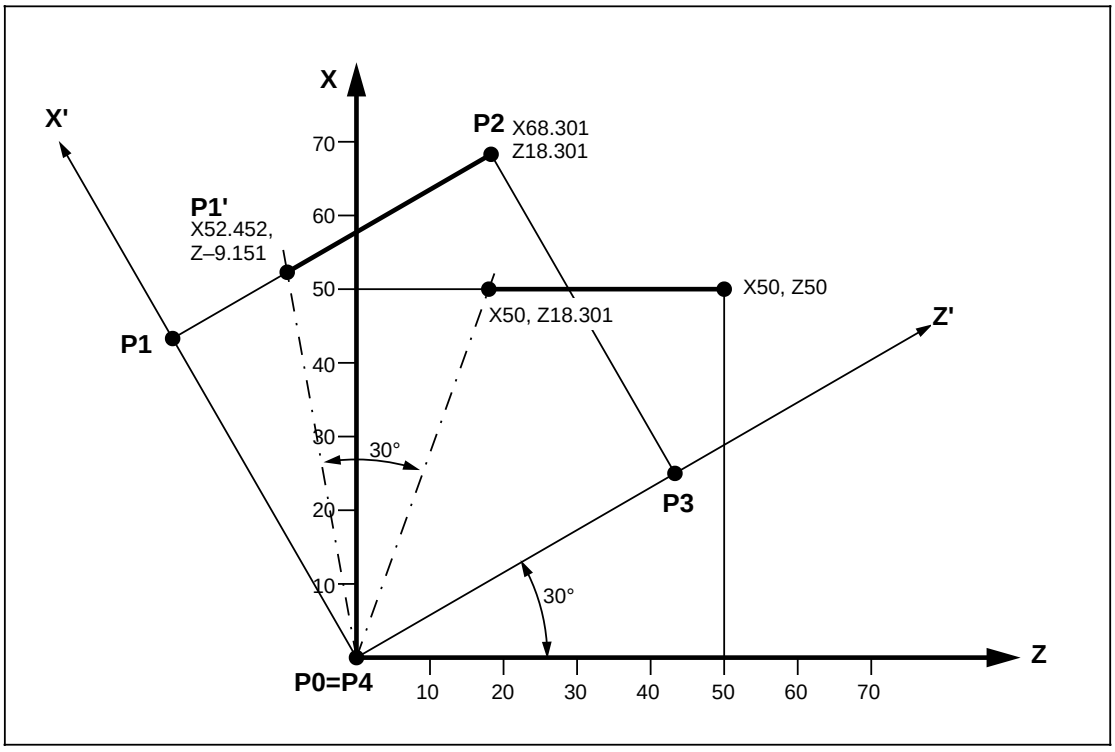
```
%35  LF
N05  G58 X Z A30  LF
N10  X50  LF
N20  Z50  LF
N30  X0  LF
N40  Z0  LF
```

If the program is aborted with "RESET" after block "N10" and then restarted, the programmer assumes that P1 (X50, Z0 rotated through 30°) will be approached.

The program requires the second axis value in order to calculate the end position. If no second coordinate was programmed, the current actual value of the second axis (Z18.301) is used for calculation purposes.

The axes thus traverse to position P1' (X50, Z18.301 rotated through 30°) and not, as expected, to position P1 (X50, Z0 rotated through 30°).

To avoid this response, block `N05  X50 Z0` must be programmed.



3.8 Scale modification (G50/G51)

This function is used to process existing NC programs with similar parts of different sizes. When the function is called, both the scale modification and the position of the new zero point of the coordinate system (scale centre) can be defined. The sign of the scale centre is determined by whether the ”old” or ”new” coordinate system is selected.

Both plane axes must be programmed in the first traversing block after G51.

The following values are recalculated by scale modification

- Axis coordinates
- Interpolation parameters
- Radius
- Programmable zero offset
- Pitch, decrease and increase

The distance ”B” or ”U” in the approach block and exit block of the ”Soft approach and exit from the contour” function is not subjected to the scale factor.

Program syntax:

```
G51 <Scale centre> <Scale factor>

G51 <Scale factor>
```

The function is selected with G51. The programmed values of the scale centre and the scale factor are stored in the setting data. This values are valid until deselected with G50.

Both plane axes must be programmed in the first traversed block after G51.

- The scale modification refers to a **scale centre** (P_M).
- The scale centre is limited to a maximum of 5 axis values.
- The value for the **scale factor** is specified at address ”P...” in the range 0.00001 to 99.99999.

  If the scale factor (P) > 1 then the size of the original part is enlarged.

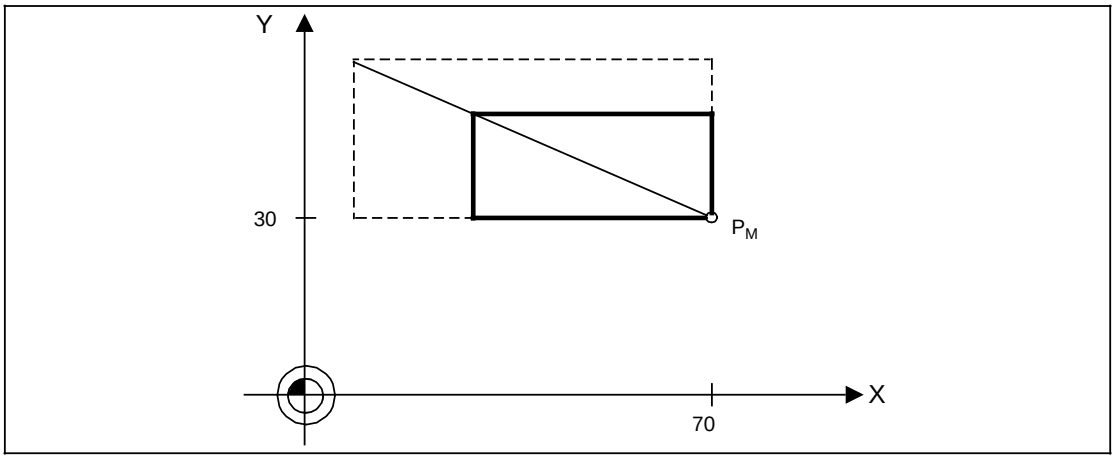
  If the scale factor (P) < 1 then the size of the original part is reduced.
- If G51 <Scale factor> only is programmed, the scale centre set in the setting data is valid.

Example 1:

```
N05 ...
N10  G51  X70  Y30  P1.5  LF  (PM)
N15  X...  Y...
```

or

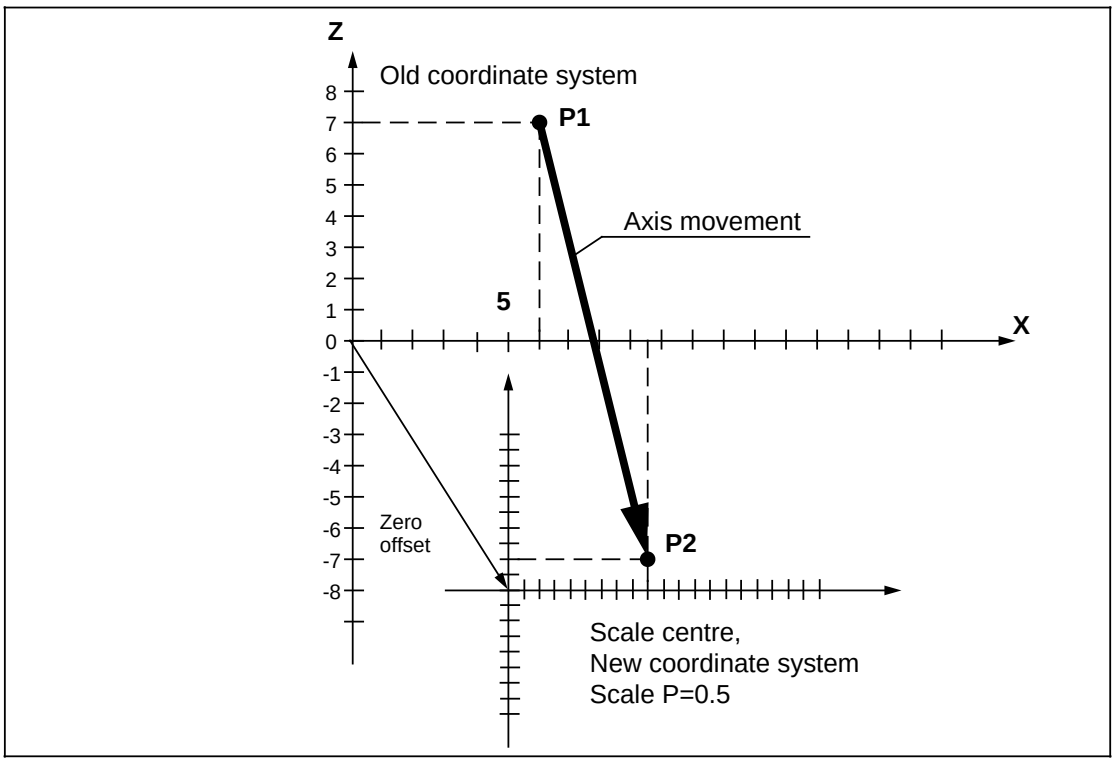
```
N...
N10  G51  P1.5  LF
N15  X70  Y30  LF  (PM)
N20  X...  Y...
```



Example 2:

```
N10 G54
N15 G00  G90  X6  Z7  LF
N20 G51  X-5  Z8  P0.5  LF
N25 G00  G90  X9  Z2  LF
```

(Position P1 in old coordinate system)
(Define scale centre and scale factor)
(The scale centre also defines the new coordinate system at the same time)
(Position P2 in new coordinate system)



3.9 Mirroring

The ”mirroring” function can only be used with linear axes.

Mirroring an axis

The ”mirroring ” function is selected via the PLC. The machine manufacturer determines the M function with which the signal for mirroring is implemented.

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 mirrored 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).

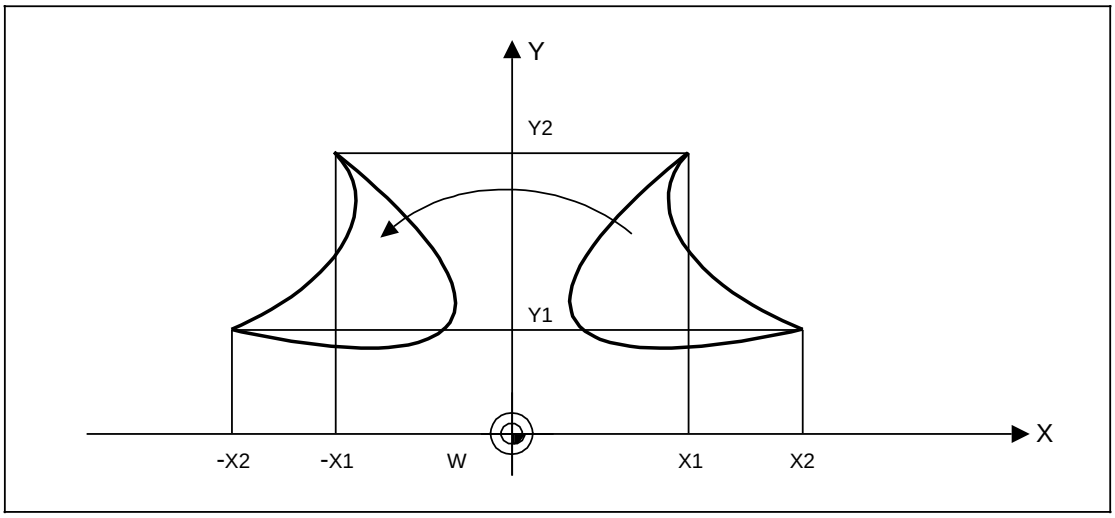
***No mirroring of
tool length offsets and
zero offsets.***

***This initial setting can be modified via machine data
(TO mirroring / no TO mirroring
along facing axis).***

In the case of turning machines the following will be mirrored with a facing axis as a function of machine data:

- ***Tool length offset***
- ***Position of tool point.***

When mirroring the Z axis this does not apply.

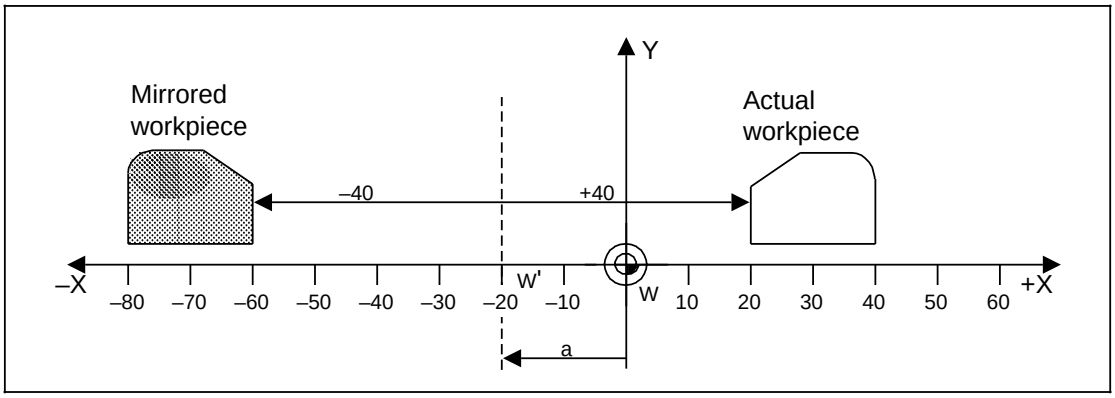


Mirroring of X values

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

If necessary, the zero of the coordinate system must be offset to the correct position before mirroring is called in the program.

Example:



Offsetting workpiece zero

In order to mirror the workpiece at the correct position, the zero must be offset by the value ”a”. This results in the distance of both workpieces from zero being the same. After mirroring, the zero can be reset to its original position.

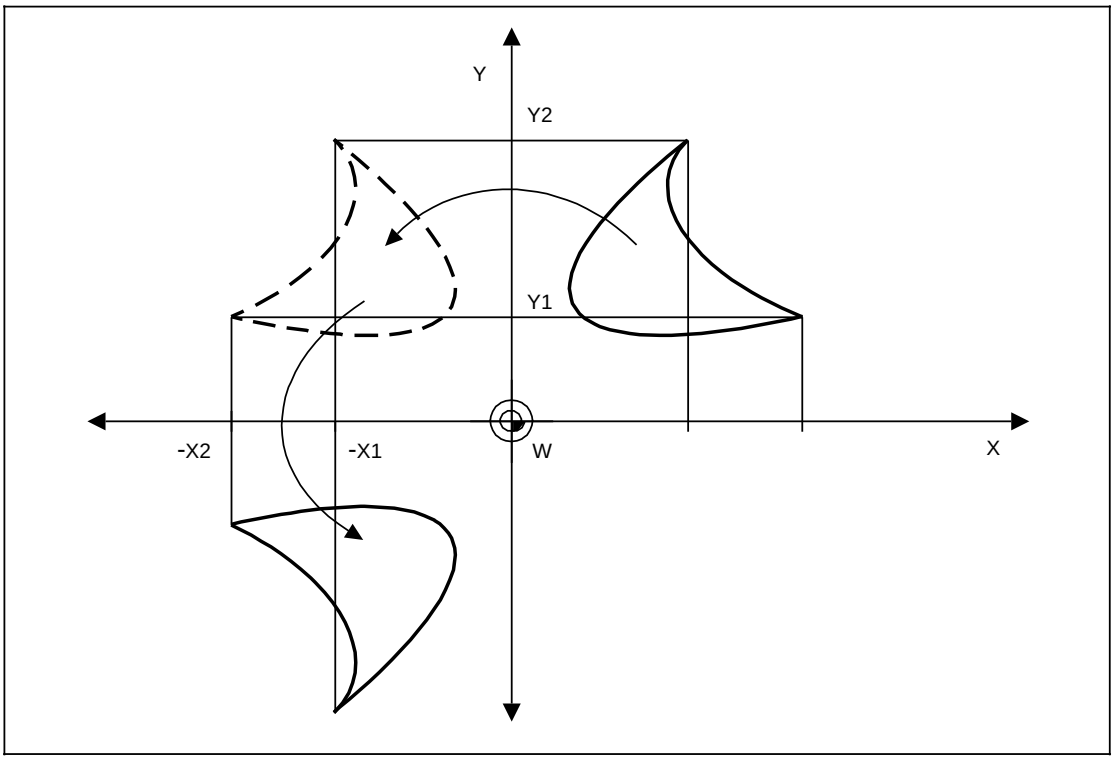
Mirroring of two axes

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

- the X values are mirrored once.
- the Y values are mirrored once.

This can be performed in any sequence.

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.



Mirroring of X and Y values

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.

Note:

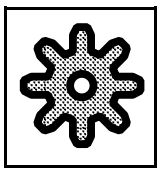
With special function @714 (buffer empty) the next block increment calculation can be stopped until the buffer is empty.
@714 is necessary for all offsets that are influenced externally, e.g. mirroring.

END OF SECTION

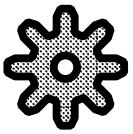
4 Feedrates

4.1 Path feedrate F..

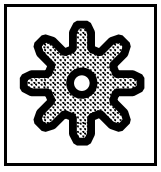
The path feedrate is programmed with address F. It determines the path velocity. It is modally active until a new F value is programmed. It is deleted with Program End or Reset. A new F value must be programmed at the latest in a block containing a traversing movement.



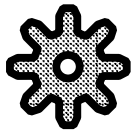
Depending on the setting in the machine data, the programmed feedrate F is maintained either along the contour or along the tool centre path (exception: spline and helical interpolation).



The programmed feedrate F can be stepped down by a factor of 100 with M37 (reduced feedrate). The function is cancelled with M36.



The path velocity must not exceed the max. velocity of the axis (machine data), otherwise the alarm "Programmed velocity too high" is triggered.



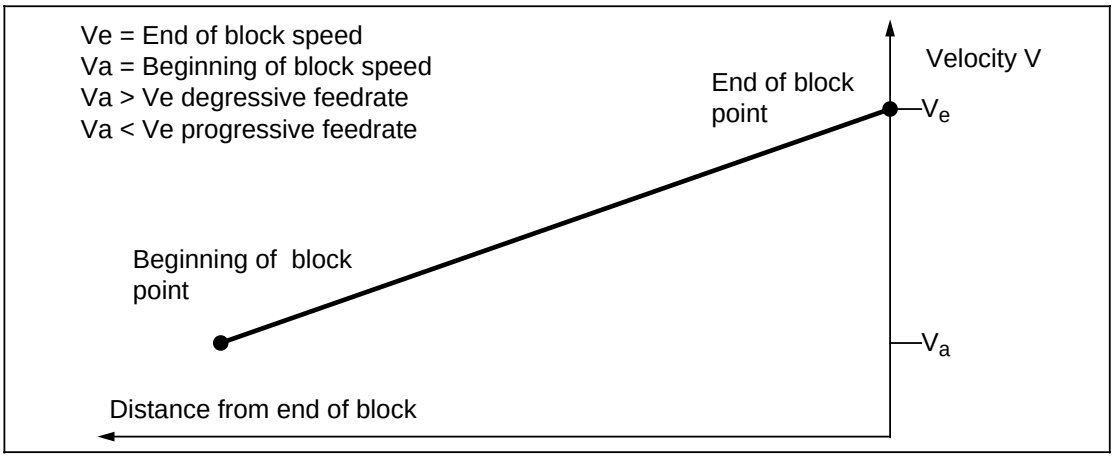
4.2 Progressive/degressive feedrate F1=... 1)

A progressive**/**degressive feedrate that is not to be reached until the end of the block is programmed with F1= ... . The feedrate is continuously changed during the progress of the NC block so that it reaches the end of block feedrate programmed in this block at the end of the block.

If no new feedrate is programmed in the next block, the end of block feedrate becomes the path feedrate. If feedrate F1=... is programmed in the next block the end of block feedrate or the path feedrate becomes the beginning of block feedrate.

---

1) Software version 5 and higher



Linearly increasing velocity, i.e. progressive feedrate

Example with progressive feedrate

```
N10  G01  F1000                 Path feedrate = 1000
N20  X10  Y20  F1=2000          Block traversed at beginning with F1000,
                                Velocity increases linearly to F2000
N30  X40                        New path feedrate F2000
```

The degressive**/**progressive feedrate is also active during the transition of interpolation feedrate (G01..) from or to rapid traverse (G00):

Example:

Change from G01 to G00 with progressive feedrate:

```
G01  G64  X10  F100             ; Approach position with feedrate F100
G00  G640  X80  F1=500          ; Traverse position with progressive feedrate
                                ; Rapid traverse reached at the end of the block
                                ; The value programmed under F1= is not relevant, but
                                ; must be allowed.
```

Example:

Change from G00 to G01 with degressive feedrate:

```
G00  G640  X10  F100            ; Approach position with rapid traverse
G01  G64  X20  F1=500           ; Approach position with degressive feedrate F1 = 500
```

Notes:

- Feedrate type (G94, G95, G98, G195, G295) must not be changed, otherwise alarm "Wrong block structure" is output.
- Progressive**/**degressive feedrate has no effect with dry run feedrate and thread (G33, G34, G35).
- The last feedrate to be active (e.g. G0 feedrate) and not the last feedrate to be programmed is always the beginning of block feedrate.
- It cannot be programmed with simultaneous axes.

4.3 Units for feedrate F (G94/G95/G98/G195/G295)

A group of G commands is used to assign a dimensional unit to the feedrate F. This applies to linear and/or rotary axes.

G94 The feedrate is programmed absolutely.

$\frac{mm}{min}$	for linear axes in a metric basic system
$\frac{inch}{min}$	for linear axes in an inch basic system
$\frac{deg.}{min}$	for rotary axes in metric/inch basic system

G95 The feedrate is dependent on the speed of the leading spindle.

$\frac{mm}{revolution}$	for linear axes in a metric basic system
$\frac{inch}{revolution}$	for linear axes in an inch basic system
$\frac{degrees}{revolution}$	for rotary axes in metric/inch basic system

G98 The feedrate is programmed absolutely for a rotary axis.

$\frac{rev}{min}$	The control automatically calculates the path feedrate of all axes involved from this leading feedrate.

G195 The feedrate is dependent on the set velocity of an endlessly rotating rotary axis.

$\frac{mm}{revolution}$	for linear axes in metric basic system
$\frac{inch}{revolution}$	for linear axes in inch basic system
$\frac{degrees}{revolution}$	for rotary axes in metr./inch

G295 The feedrate depends on the actual velocity of the endlessly rotating axis.

$\frac{mm}{revolution}$	for linear axes in metric basic system
$\frac{inch}{revolution}$	for linear axes in inch basic system
$\frac{degrees}{revolution}$	for rotary axes in metr./inch

Notes:

- If a G function is changed, a suitable new F value must be programmed for it, because the dimensional unit of a stored F value changes.
- If G98 is programmed together with several rotary axes or with one or several linear axes, the alarm "Wrong block structure" appears.
- The G functions are modal.
- If rotary axes and linear axes traverse a path with G94/G95 together, the feedrate must be interpreted in the dimensional unit of the linear axes [mm/min], [inch/min], [mm/rev], [inch/rev].
- With G195/G295 the associated endlessly rotating axis must be programmed, e.g. G [C2] 195.
- It is not possible to program the velocity with reference to the setpoint of a following axis (e.g. G195 <following axis number>).

4.4 Simultaneous feedrate F[axis name] 1)

Every simultaneous axis traverses at its axis-specific feedrate. This is programmed under **F[axis name] axis feedrate**. The feedrate programmed for each individual axis remains active until reprogrammed, a Reset or End of Program.

For endlessly rotating rotary axes an axis-specific feedrate type (G[...]94, G[...]98, G[...]195, G[...]295) can be programmed.

For simultaneous axes with end-of-block response P[..] the axis-specific feedrate G[...]94 must be selected otherwise alarm 1308* ”Simultaneous axis incorrectly programmed” will be output.

The axis-specific override switch acts on simultaneous axes.

Extended addressing can be used for simultaneous axes, e.g.:

`F[X]1000`	F[X] is the feedrate assigned to simultaneous axis X
`F[Z2]=500.70`	F[Z2] is the axis feedrate assigned to simultaneous axis Z2
`F[R20] 1000`	F[R20] is the feedrate of the simultaneous axis given in parameter R20
`F[P5] 800`	F[P5] is the feedrate of the simultaneous axis given in the parameter to which R5 points.
`F[A1]=R10`	F[A1] feedrate of simultaneous axis A1 is given in parameter R10

Note:

Feedrate modification with M37 is permitted

When the feedrate type is modified, a new feedrate must be programmed.

Example:

```
N10 P[X]20 F[X]1000
N20 G[C]103 G[C]98 F[C]1000
```

END OF SECTION

1) *Software version 3 and higher*

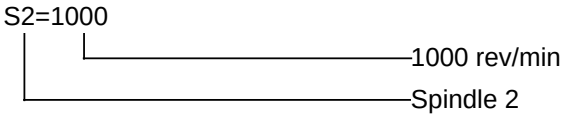
5 Spindle Movement

5.1 Spindle function S..

The data below can optionally be entered under address S:

- Spindle speed
- Cutting speed
- Spindle speed limitation
- Spindle stop in degrees (M19)
- Spindle dwell in revolutions (see G04)

The extended address notation must be used and the spindle number specified for the S word, e. g.:



5.2 Spindle speed limitation (G92 S...)

In certain circumstances it may be necessary to prevent the spindle speed from increasing further (e.g. with constant cutting speed G96) and therefore to continue machining at a constant speed from a certain point onwards. The speed limitation is programmed in rev**/**min under S in a separate block before the program section in which the relevant machining operation is programmed. The G92 S... function can be written more than once in the program and is effective only with G96.

Example:

```
N10  G92  S300  LF
```
(spindle speed limited to 300 rev**/**min)

No further commands must be entered in a block containing G92. This restriction does not apply with G94 and G95.

G92 S... is also used to cancel the limitation, whereby the maximum speed of the selected gear stage is written under S. The spindle is stopped with G92 S0.

5.3 Constant spindle speed limitation (G26 S...)

The speed monitor programmable under G26 S... is used as a workpiece or chuck-dependent safety speed for the entire part program irrespective of the speed programming with G94 to G97.

G26 S ... limits the setpoint speed and monitors the actual speed value.

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

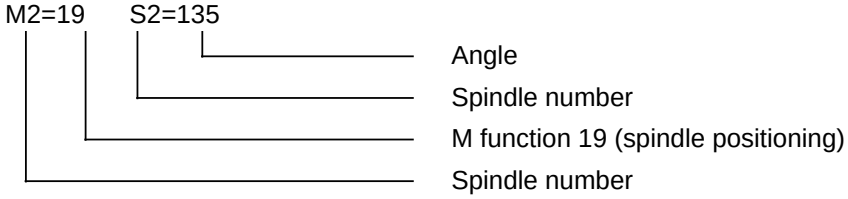
5.4 Main spindle control (M03/M04/M05/M19)

(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 Counter-clockwise spindle direction of rotation
M05 Non-oriented spindle stop
M19 Oriented spindle stop

An extended address notation with output of the spindle number (dependent on the type of control) is available for miscellaneous functions in this group: e.g.:



M19 S.. can be used to perform an oriented main spindle stop (lead spindle with 840C). The relevant angle is programmed under S in degrees. 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”. M19 does not cancel M03 or M04. M19 S.. must be programmed in a separate block.

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.

Machine data can specify whether the spindle is reset with M02/M30 or by a PLC signal only.The oriented spindle stop (M19) does not depend on this setting.

Note:

Several M03, M04, M05, M19 spindle M functions as well as C axis ON/OFF must not be programmed together in one block, otherwise alarm 3006 ”Wrong block structure” appears.

5.5 Constant cutting speed (G96/G97)

Selection: **G96 S ...** (constant cutting speed ON)

Deselection: **G97, G94, G95** (constant cutting speed OFF and last set speed from G96 stored)

Under **G96 S ...** , you can enter a constant cutting speed in $^{m}/_{min}$. The control automatically switches to revolutional feedrate, i.e. the programmed feedrate F is interpreted in $^{mm}/_{rev}$.

The spindle speed depends on the programmed cutting speed (S) and the workpiece-related actual value (D). It is calculated by the control according to the formula

$$n = \frac{1000 \cdot S}{\cdot D}$$

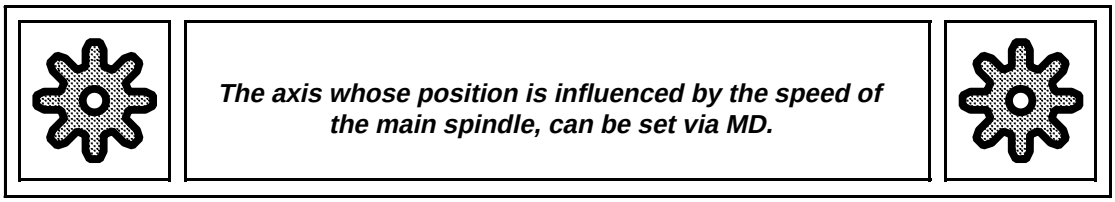
The function G96 assumes that the machine zero and the workpiece zero of the facing axis are located in the centre of the spindle.

Tool offsets and zero offsets in the facing axis are **not** taken into account in calculation of the spindle speed.

Example:

```
% 1
N5  G0  X100  Z50  D1  G55
N10  G96  S100  M3  F0.1
N15  G1  X22
N20
```

In block N15, a set spindle speed of 1446 $^{rev}/_{min}$ is calculated, regardless of whether there is an offset in the X axis (facing axis) in the tool offset memory D1 or in the zero offset G55.



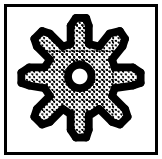
Note:

- At constant cutting speed you can only work in one gear speed.
- While G96 is active you cannot execute M19.
- If tool radius compensation (G41/G42) is active, G96 S... must be programmed in a block without traversing movements in the TRC plane.
- The spindle offset switch only applies to the spindle speed limitation under **G92 S ...** .
- A preset or DRF offset in the facing axis influences calculation of the set spindle speed.

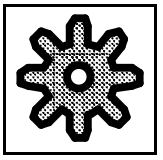
The spindle speed last calculated is stored under G97. With G97, you can avoid speed changes when the facing axis moves without machining (e.g. during a tool change). The feedrate is interpreted as a revolutional feedrate. When G97 is active, you cannot influence the cutting speed by programming an S value again.

5.6 Spindle as rotary axis (SWITCH ON/OFF C axis mode)

C axis mode is selected and deselected using customer specific M functions. These must stand alone in a part program block.



The M function numbers are stored in the MDs.



When programming use the same address extension as for spindles.

Notes:

Up to SW 5.3: After switching on, the C axis must be programmed for the first time with G90. Switchover to G91 is possible only then.

With SW 5.4 and higher: The C axis can be programmed immediately with G91 if no scale factor/coordinate rotation is active for the C axis.

END OF SECTION

6 Velocity Modification/Block Change Behaviour

6.1 Fine (G60) and coarse (G600) exact stop tolerance ranges [1)]

G09, G60 and G600 can be used to approach a target position within a specified exact stop tolerance range. In the case of G60, the set velocity is reduced to 0. When the following error falls below the exact stop window, block change is initiated and the axis motion (from P2 to P3) programmed in the next block commences.

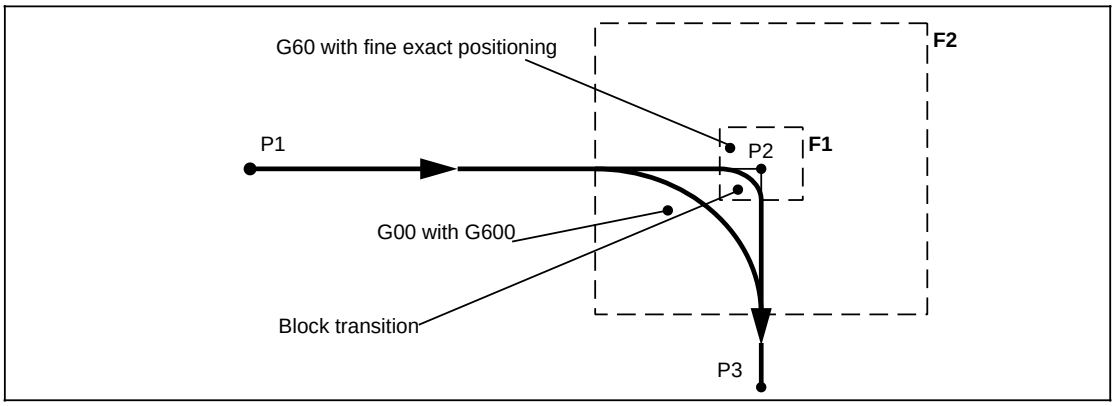
Functions G09 or G60 and G600 can be used, for example, for machining sharp corners, for plunge-cutting or when reversing the direction of movement. Functions G60 and G600 have no effect when thread cutting.

09 is active for all interpolation types and is non-modal.
G60 is active for all interpolation types except with an active G00 command and is modal.
G600 is active only in conjunction with G00 and is modal.

Fine and coarse exact stop tolerance ranges

G60	Fine exact positioning:	10 µ (window F1)
G600	Coarse exact positioning:	250 µ (window F2)

When G09 is active, the interpolation type determines whether the fine (G01) or coarse (G00) exact stop limit is applied.



Exact positioning window; the thin line represents the velocity control action of the control system. The characteristic (thick line) within the NC is smooth.

If the two exact stop tolerance ranges are identical (window F1 = window F2), the effect of G00 with G600 will be the same as that of G60. The rapid traverse motion generally has a larger exact positioning window. This saves time during rapid traversing (earlier block change).

The exact stop limits, fine and coarse, can be changed with machine data.

1) *Software version 4 and higher*

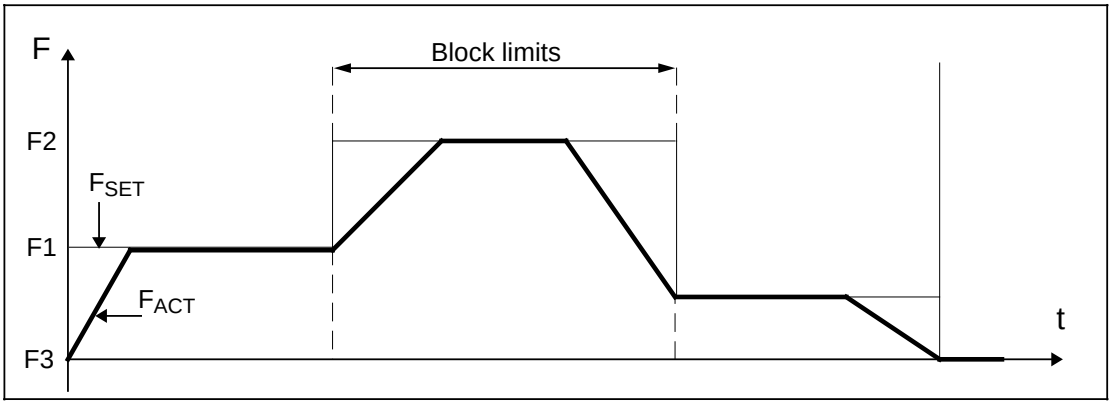
6.2 Continuous-path mode (G62/G64/G620/G640) 1)

The preparatory functions G64 and G640 are used if no relief cutting is to occur at transitions from block to block. This function also causes transitions to be smoothed when changing the direction of movement. When applying a dry run feedrate with G64, the dry run feedrate is used both during the block and at the end of the block. The programmed feedrate value is not used.

G62/G64/G620/G640 are modal.

G62 and G64 are active for all interpolation types except with G00.

G620 and G640 are only active with G00.



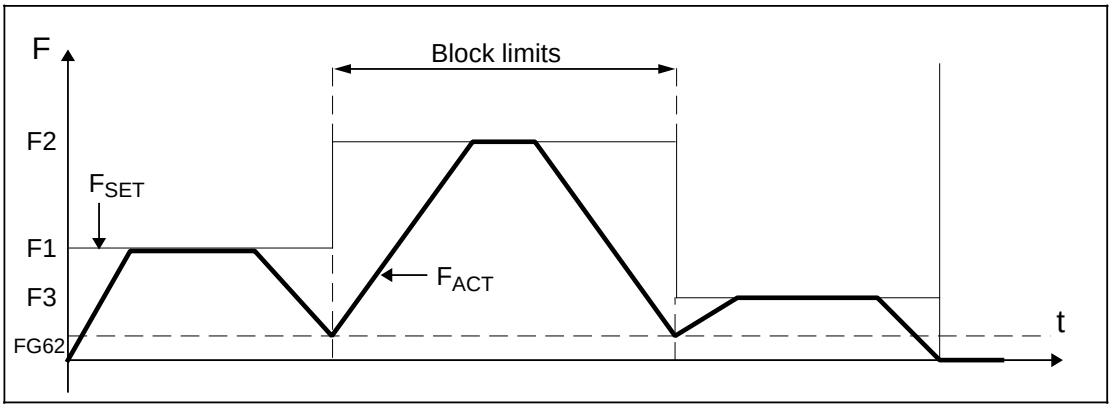
Contouring G64, G640

Note:

If, on a transition from continuous-path control to exact stop, the path of the transition block is shorter than the required deceleration distance, the velocity is immediately reduced to zero by function G09.

Alarm 3092 ”Programmed speed too high” is displayed.

The feed motion is reduced towards the end of the block to a rate defined in the machine data using the function G62 and G620.



Approach to corner deceleration G62, G620

1) Programming of G620/G640 is possible with software version 4 and higher

Note:

The reduction velocity for G62 and G620 is not attained precisely. The error tolerance band is described as follows:

FG62 = Reduction velocity (MD 3, MD 146*, MD 148*, MD 150*)
F_{act} = Attained "actual" reduction velocity
F_{Decel} = Deceleration step (velocity reduction per IPO cycle)

$$FG62 \quad F_{act} \quad (FG62 + F_{Decel})$$

FG62 values < F_{Decel} values are not advisable.

$$F_{Decel} \left[\frac{mm}{min}\right] = a \left[\frac{m}{s^2}\right] \cdot tipo\ [ms] \cdot 60$$

a = Acceleration (defined in machine data) [m/s^2]
tipo = IPO cycle [ms]

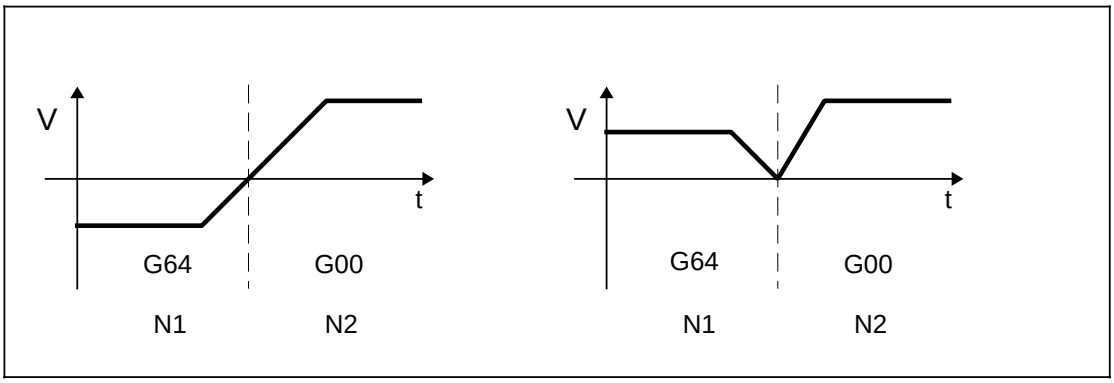
No reduction occurs at the first block limit if F1 FG62 + F_{Decel}
No reduction occurs at the second block limit if F2 FG62 + F_{Decel}

Separate values can be set in the machine data for G620 with G00, G62 with G01 and G62 with G36. 1)

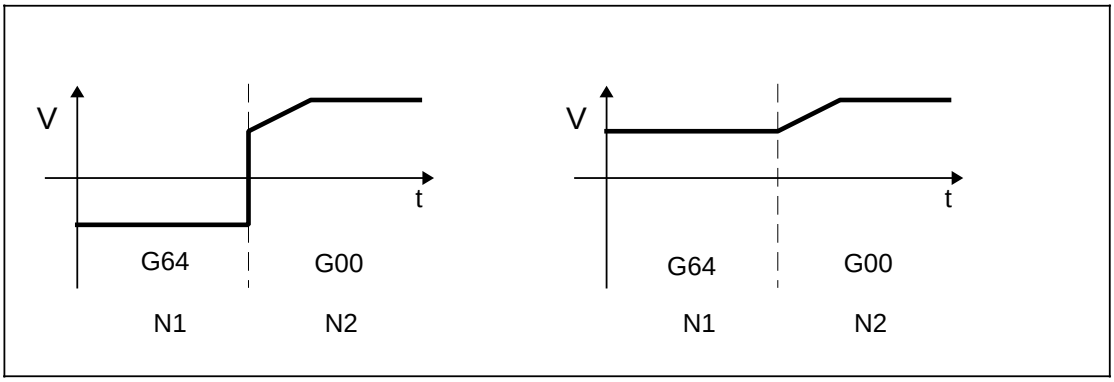
1) *Software version 5 and higher*

6.3 Change from continuous-path mode to rapid traverse

Depending on the machine data, the control automatically generates “Speed reduction with exact stop coarse” between a G01 block with G64 and a G00 block with G600.



Block change with automatic generation of speed reduction with “Exact stop coarse”.



Block change without automatic generation of speed reduction with “Exact stop coarse”.

If, on a transition from continuous-path control to exact stop, the path of the transition block is shorter than the required deceleration distance, the velocity is abruptly reduced to zero and alarm 3092 ”Programmed speed too high” is displayed.

The function G60 has no effect when machining threads.

Special case: active thread function

When the thread function is active, deceleration towards the block must not occur, as this will cause damage to the thread, i.e. the velocity is abruptly reduced to zero after the last thread block. If another axis is then programmed and the servo enable is already missing, alarm 168* ”Servo enable trav. axis” is displayed.

Note:

Note MD 1513.0 "G63 without delay".

Remedial action:

- If G09 is programmed in the last G33 block, the speed is reduced (exact stop) at the end of the thread block. In this case no alarm is triggered.
- The servo enable should be set in good time in the last thread block.

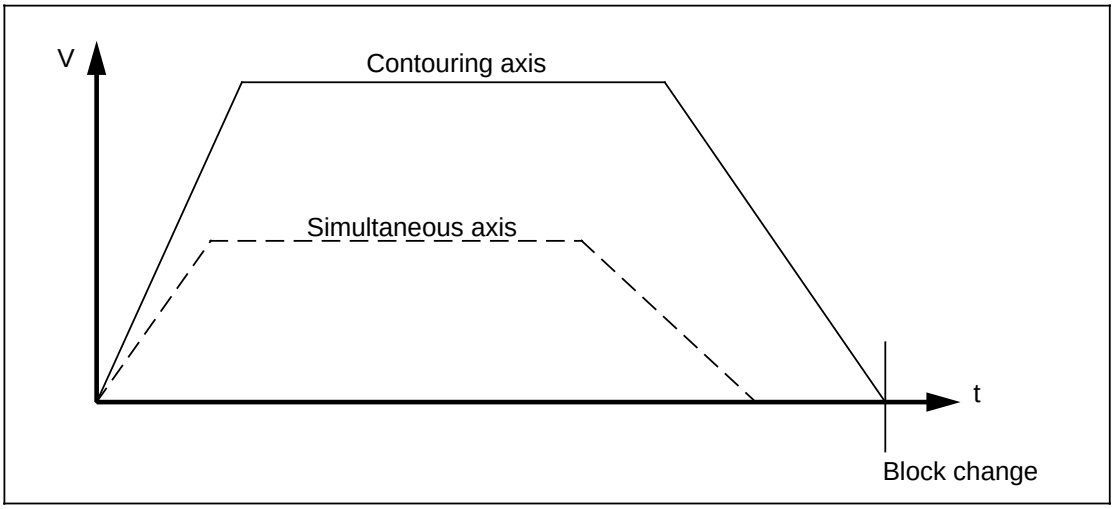
The exact stop function G09 is also active with thread machining and must therefore be programmed in the last thread block.

Deceleration does not occur if the function (G64 or G00 with G600) is active and if an additional thread function (G33, G34, G35) is active during the change to rapid traverse. The block change is then made as before with the current path speed.

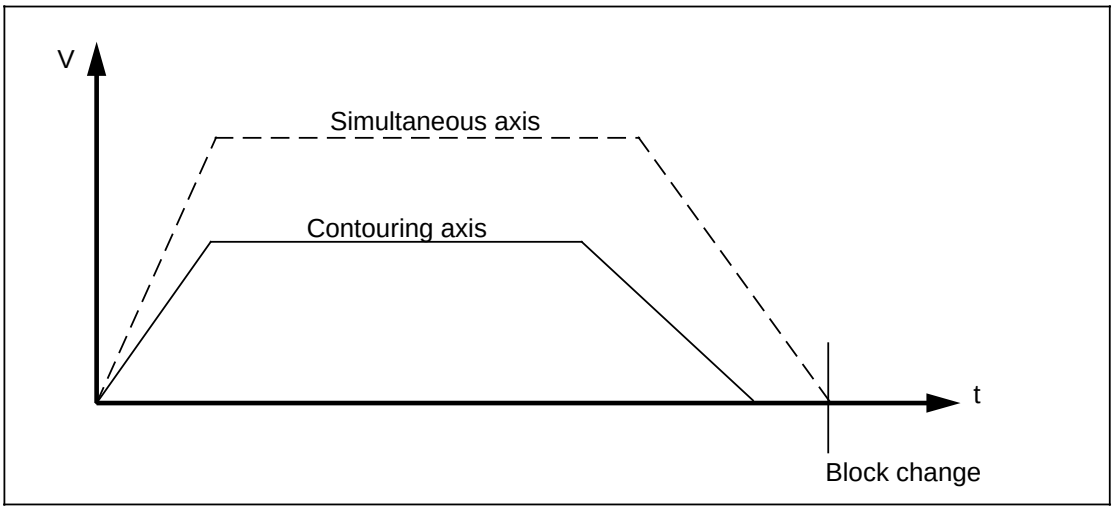
6.4 Block change behaviour with contouring axes and simultaneous axes

The block change which so far has been influenced by the interpolating axes is now influenced by both axis types. There are two possibilities:

- The contouring axis reaches the end point before the simultaneous axis
- The simultaneous axis reaches the end point before the contouring axis



The contouring axis determines when block change takes place, although the simultaneous axis has been programmed with end-of-block behaviour.



The simultaneous axis determines when block change takes place, because it has been programmed with end-of-block behaviour. In such cases G60 is forced on the contouring axes.

6.5 Programmable reduction velocity with G62

Function up to SW 6:

In blocks with G62 or G620, the velocity at the end of the block is reduced to the velocity which is preset in MD 3 (or MD 146*, MD 148*, MD 150*).

Function as from SW 6:

Programming F62= cancels the machine data 3, 146*, 148* and 150* until the end of the program and activates the programmed value instead as reduction velocity with G62 or G620 in the part program. The programmed reduction velocity is immediately active, i.e. as from that block in which it is programmed and is modal either until the end of the program or until "F62=" is programmed again.

The function must be enabled via MD 5052-7=1 ("F62 programming"). The programmed reduction velocity is then output to the PLC as F auxiliary function with extended address.

CAUTION:

If MD 5052.7 is 0, "F62=..." triggers a path feedrate (F=62) and no error message occurs.

END OF SECTION

7 Coordination of Programs and Axes in the Channels

7.1 Channel-specific programming

The NC area of the SINUMERIK 840C control can be split into a maximum of 6 channels. Each channel has its own block preparation feature, which combines data from the part program with NC data such as zero offsets, tool offsets etc. and transfers it to the interpolator via a buffer memory.

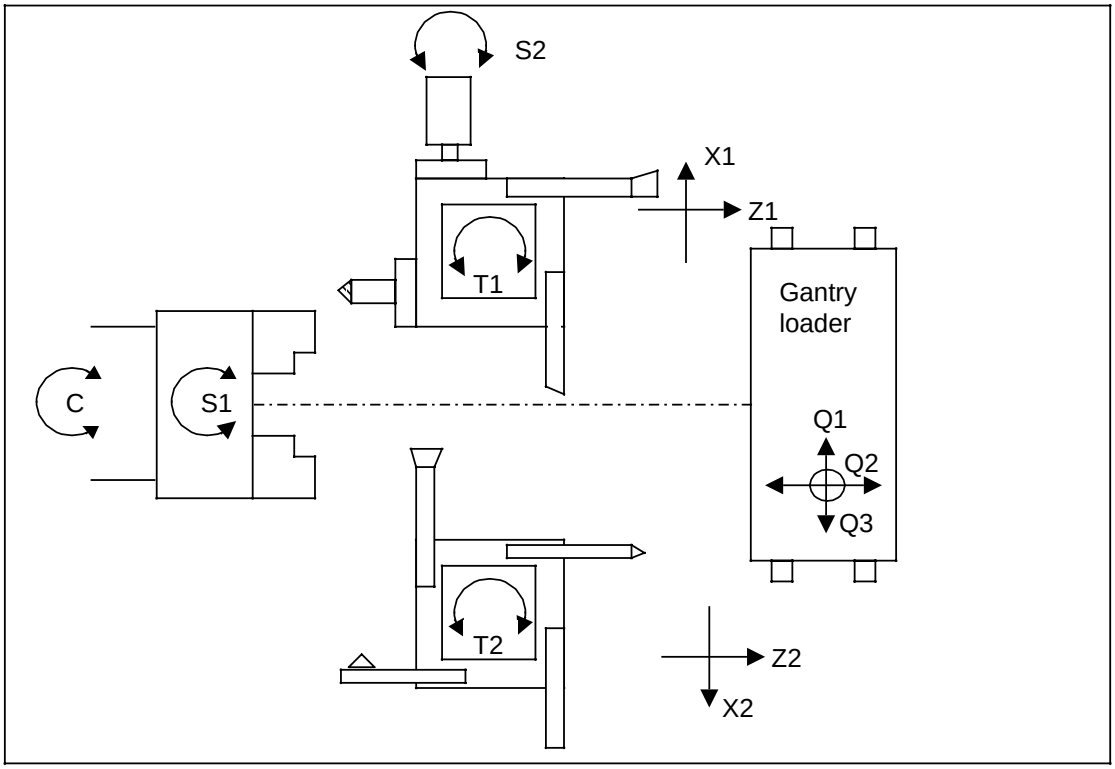
The NC channels are combined to form operating mode groups. Channel assignment to these groups is specified by machine data. Depending on the workpiece involved, machining programs can be generated for all channels and these are entered into the central NC program memory.

Example: Use of the channel structure on a double-slide single-spindle lathe with milling attachment and NC-controlled tool turrets.

NC channel 1 for the NC axes X_1, Z_1, main spindle S_1 and auxiliary spindle S_2 as well as the C axis (main spindle for milling) and tool turret T_1.

NC channel 2 for the axes X_2, Z_2 and tool turret T_2. NC channel 3 controls the gantry loader with axes Q1, Q2, Q3.

The channel structure permits simultaneous and asynchronous processing of the programs for the individual channels. Synchronization can be performed by the PLC if required.



Schematic representation of a double-slide single-spindle lathe for turning and milling with integrated loader.

7.2 Program coordination

The program coordination function allows several programs to be synchronized in their execution. Programs running in different channels can be matched to one another.

The program coordination function must be implemented in the PLC. This can be done by FB38 and FB39 (package 0) as standard procedure.

Note:

From SW 5.6 onwards the synchronization of NCK channels can be carried out via the WAIT M commands instead of via the PLC (FX38 and FX39) NCK-internally. The WAIT M commands are then no longer issued by the PLC. The NCK channels synchronize themselves internally on the programmed WAIT marks.

If a programmed WAIT mark is reached by all the NCK channels concerned, the block in the next IPO cycle changes synchronously in these channels.

From SW6.1 onwards: While an NCK channel is waiting for a WAIT mark to be synchronized internally, the channel-specific interface signal ”Internal WAIT mark synchronization: Wait for synchronization”(DBD10. DL15.8) is set to 1. The interface signal is reset when the WAIT M requirements have been met by the other NCK channels or the NC-STOP/RESET in the channel.

The internal WAIT mark synchronization is activated via the MD 5003.1.

Channel-specific interface
Internal WAIT mark synchronization, wait for synchronization: DM10, DL15.8

A coordination command consists of the command identifier and a maximum of four parameters. The command identifier and the first parameter must be separated by a blank (space). The individual parameters of the parameter chain must be separated by a comma or/and a blank. The last parameter can be followed by a blank before the square bracket.

In each part program block only one coordination command is permitted. One comment is allowed in each block. Within a square bracket of the coordination command no further brackets (no comment either) are allowed.

Commands as defined in DIN 66025 are permitted in the same block before or after a coordination command; however, the following limitations apply: In multiple programming, the information in the coordination command bears no relation to the other information given in the same block
e.g. **`N075 [WAIT M 10, 2, 3] G1 X100 F1000`**

Mark 10 declared in block N75 has no connection with axis value X100. If mark 10 is not to be signalled to channels 2 and 3 until the axis end value X100 has been reached, the traversing command must be programmed in the previous block.

”[ ]” can be used for coordination commands only; not, however, for comments. ” , ” can be entered only within the square brackets. Characters ” , ” and ”[”, ”]” cannot be searched for within a part program.

SW 1 and 2: A coordination command programmed in a block and a comment are displayed together. If the displays are longer than the maximum number of characters of the command line, the remaining text is truncated.

As from SW 3: If program coordination commands are to be displayed as a comment, they must be programmed as a comment (e.g. WAIT).

The following coordination commands can be programmed:

```
START
WAIT E
WAIT M
INIT MPF
INIT SPF
END
```

The parameters must be separated by a comma or a blank. A blank can be inserted before the comma.

Numerical values between 1 and 6 and**/**or R parameters are allowed as global channel numbers. A maximum of 6 channel numbers/parameters can be specified. R parameters are permissible from 0 to 599.

- **Coordination command "START":** Start of the part programs in the specified channels.

  Syntax: [START K., K., ...] K. = channel number (specification only digits 1... 4)
  Programming example: [START 1, 2, 3] or [START R50, 2, R52]

The start of channels 1, 2 and 3 is performed in channel 4.

- **Coordination command "WAIT E":** Waiting for end of program in the specified channels. (They must incorporate the END command).

  Syntax: [WAIT E, K., K., ...]
  Programming example: [WAIT E, 1, 2] programmed in channel 3, i.e. channel 3 program effects end message [END3] of channel 1 or channel 2.

- **Coordination command "WAIT M":** Waiting for the programmed mark to be reached in the programs of the specified channels. The co-ordination command must be written in each program which is to take part in the coordination. There is no connection with the block numbers or auxiliary functions.

  Syntax: [WAIT M, mark, K., K., ...]
  Programming example: [WAIT M, 555, 1, 2] or [WAIT M, R50, 1, R52]
  programmed in channel 3; i.e. the program in channel 3 waits at this point for the mark 555 in channel 1 and channel 2 to be reached.
  Possible channel entries 1 ... 6.
  The mark can be input as a number with a maximum of 4 digits (0...9999) or an R parameter.

Note:

Always make sure that all the channels involved in a mark M are listed as parameters of the WAIT M command in each channel.

Example of correct programming:

Channel 1	Channel 2	Channel 3	Channel 4
[WAIT M 2,2,4] [WAIT M 3,3]	[WAIT M 2,1,4]	[WAIT M 3,1]	[WAIT M 2,1,2]
[WAIT M 2,2,3,4]	[WAIT M 2,1,4]	[WAIT M 2,1]	[WAIT M 2,1,2]

Example of incorrect programming:

Channel 1	Channel 2	Channel 3	Channel 4
[WAIT M 2,2,3,4]	[WAIT M 2,4]		[WAIT M 7,1,2]
	Error: Channel 1 missing	**Error:** WAIT M missing in channel 3	**Error:** Incorrect mark

Example of internal wait mark synchronization WAIT M and G5xx (Sect. 9.6)

Channel 1

```
N30 X100 F1000 G90 [WAIT M 10, 2] G511 F50
```

Channel 2

```
N5 [WAIT M 10, 1]
N10 Y100
```

The internal wait mark synchronization WAIT M together with G5xx causes the wait mark to become visible for possible further channels only after reaching the position (actual axis position or residual setpoint distance to go) behind G5xx.

- **Coordination command ”INIT MPF”:** Select main program in the specified channel.

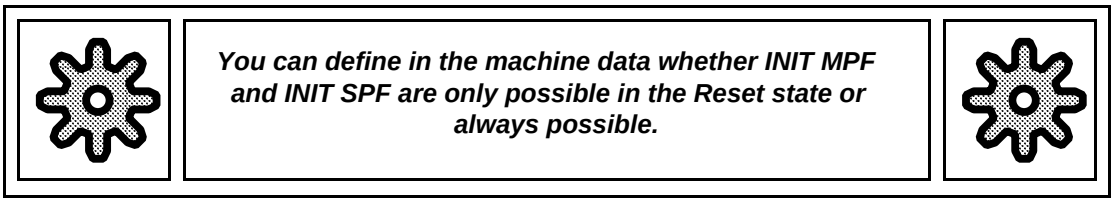
Syntax:	[INIT MPF, program number, K.]	Program number (1 ... 9999)
Programming example:	[INIT MPF, 1234, 1]	In channel 1, %1234 is preselected.

- **Coordination command ”INIT SPF”:** Select main program in the specified channel.

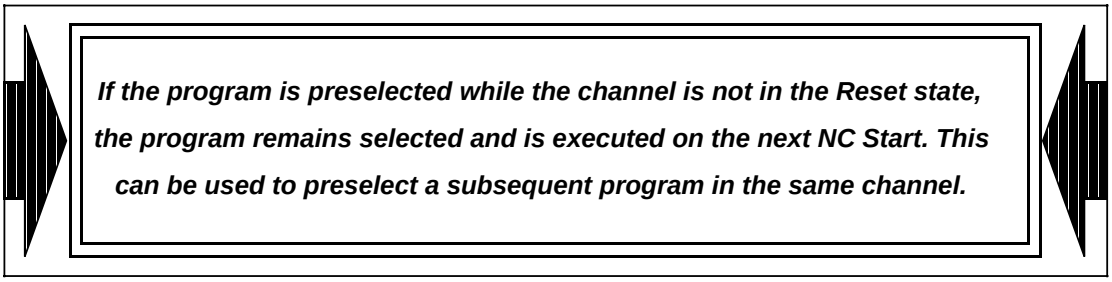
Syntax:	[INIT SPF, subroutine number, K.]	Subroutine number = (1 ... 9999)
Programming example:	[INIT SPF, 123, 1]	In channel 1, L1234 is preselected .

  Program number, subroutine number and channel number can also be entered as R parameters.

The following applies to commands "INIT MPF" and "INIT SPF" with SW 3 and higher:



You can define in the machine data whether INIT MPF and INIT SPF are only possible in the Reset state or always possible.



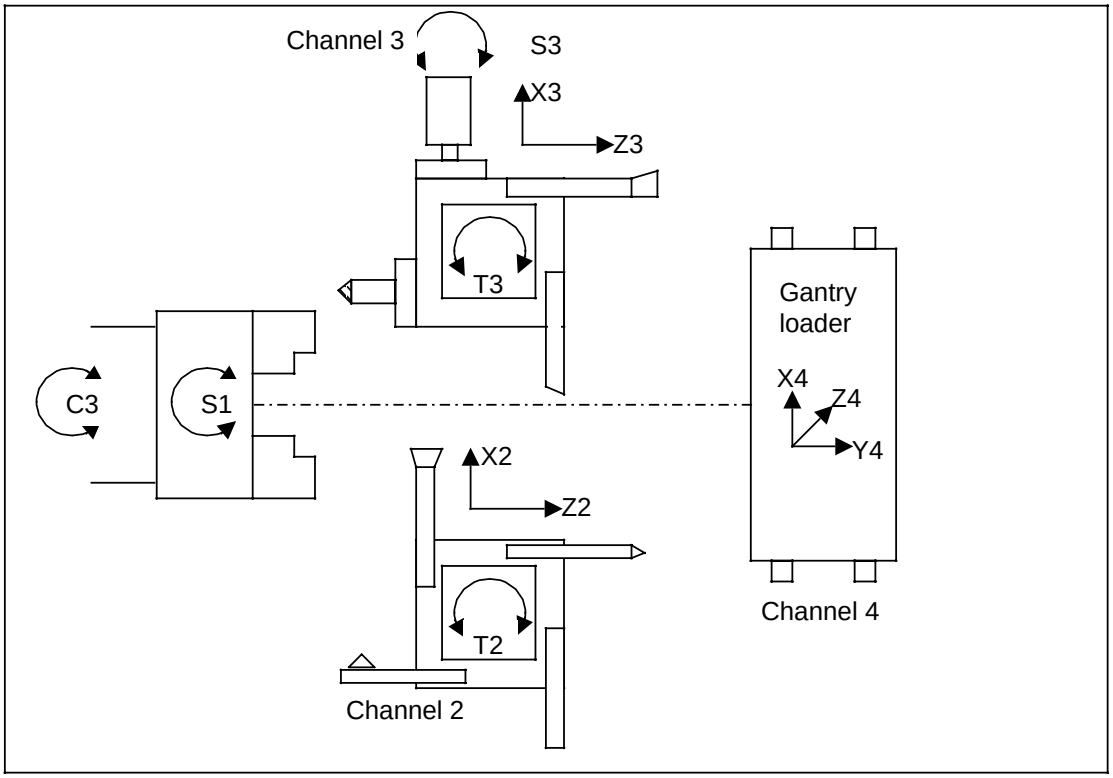
If the program is preselected while the channel is not in the Reset state, the program remains selected and is executed on the next NC Start. This can be used to preselect a subsequent program in the same channel.

- **Coordination command ”END”:** Indication of end of program to the channels waiting with WAIT E. The command must be located immediately before M30 or in the M30 block.

Syntax:	[END, K., K., ...]	
Programming example	[END, 1,3]	Communicating program end to channels 1 and 3.

  K.: 1 ... 6.
  ”END” and ”WAIT E” are permitted in the main program only.

Example 1: Program coordination channel 1



Example 2:

Mode group1	Mode group 2		Mode group 1
Channel 1	Channel 2 X2 Z2 T2 (S1)	Channel 3 X3 Z3 C3 T3 S3 (S1)	Channel 4 X4 Y4 Z4

Channel 1	Channel 2	Channel 3	Channel 4
MPF 11	MPF 12	MPF 13	MPF 14
[INIT MPF,12,2]	.	.	.
[INIT MPF,13,3]	.	.	.
[INIT MPF,14,4]	.	.	.
[START 2,3]	.	.	.
.	[WAIT M,8,3]	[WAIT M,8,2]	.
.	.	.	.
.	[WAIT M,9,3]	.	.
.	.	[WAIT M,9,2]	.
[WAIT M,10,2,3]	[WAIT M,10,1,3]	[WAIT M,10,1,2]	.
[START 4]	.	.	.
M30	M30	M30	M30

7.3 Axis actual-value synchronization (G200)

With software version 2, the user must provide the axis actual-value synchronization function, for example, by means of an NC stop or through programming of @714 STOP-DEC (empty buffer memory). The control then automatically executes an axis actual-value synchronization.

From software version 3 onwards, the ”Axis actual-value synchronization” function offers the user the facility of updating the actual values of axes **within a channel**. This function is primarily needed when an axis is traversed within several channels or in connection with the functions ”Endlessly rotating rotary axis”, ”Gear interpolation” or "Interpolative compensation with tables”.

The function is selected with G200 and is non-modal. A further 5 axes and G53 can be programmed in a G200 block.

If G200 is programmed without axes, all linear and rotary axes of the mode group are synchronized.

While synchronization is in progress, the axis positions cannot be updated until the specified axes have reached a standstill.

`N20 G200 LF`	(Synchronization of all axes in the mode group)
`N20 G200 X Y Z ... LF`	(Synchronization of individual axes in the mode group; max. 5 axes)
`N20 G200 X2= Y Z3= ... LF`	(Synchronization of up to 5 axes using ”Extended address”)

In the block following G200, the axes can be programmed directly with G90 or G91. An axis containing an active coordinate rotation or a scale factor must be programmed with G90.

By programming G200 with G53, the zero offsets are allowed for in the specified axes when programmed after the G200 block with G91.

Example 1: Traverse X axis in several channels **without axis actual-value synchronization**

Channel 1	Channel 2	
`%1`		Start of % 1 in channel 1
`N10 G90 X100 LF`		Axis X traverses to position 100
`...`	`%2`	Start of %2 in channel 2 (autom. synchr)
`...`	`N15 G90 X200 LF`	Axis X traverses to position 200
`N20 G90 X250 LF`		Axis X traverses to position 350

Axis X travels 150 mm further in block N20 to X=350 even though X=250 has been programmed, resulting in a path offset of 100 mm.

Example 2: Traverse X axis in several channels **with axis actual-value synchronization (G200)**

Channel 1	Channel 2	
`%1`		Start of %1 in channel 1
`N10 G90 X100 LF`		Axis X traverses to position 100
`...`	`%2`	Start of %2 in channel 2 (autom. synchr)
`...`	`N15 G90 X200 LF`	Axis X traverses to position 200
`N20 G200 X LF`		Synchronization of axis X
`N30 G90 X250 LF`		Axis X traverses to position 350

Before block N30 is executed in channel 1, an axis synchronization command is executed for the X axis (G200 X) in block N20. This rules out a path offset.

Notes:

- Tool radius compensation must not be active.
- If G200 is programmed while a transformation for fictitious axes is selected, the command is not executed until all axes of the mode group have reached "Exact stop fine".
- Axis actual-value synchronization is executed automatically by the control on NC Start for the following automatic submodes:
  - Block search
  - Single block
  - DEC single block
  - NC Start from Program Stop state
  - NC Start from NC Stop state

7.4 Interruption of program

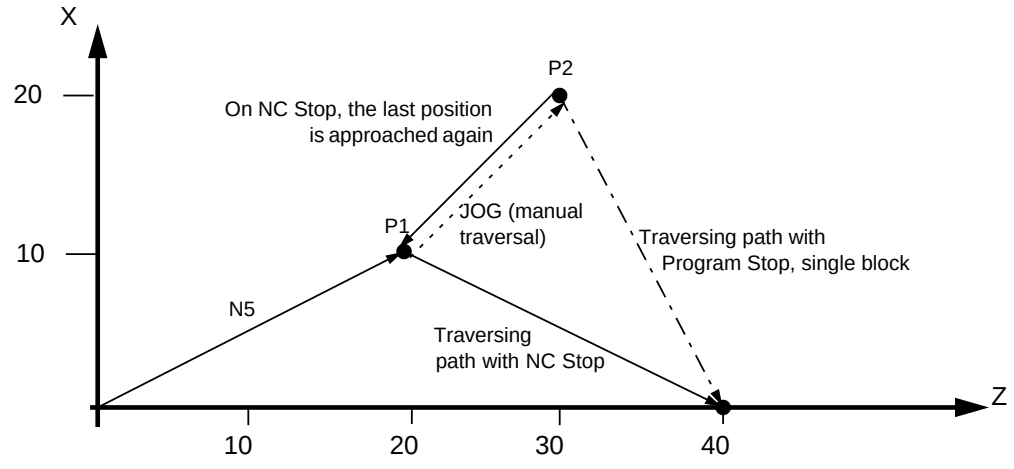
Processing of the NC program can be interrupted by switching over to **single block** or by means of **NC Stop**. It is possible, for example, to approach any point in the "JOG" mode. Actuation of the NC Start key initiates axis actual-value synchronization; the next programmed position is approached according to the method determined by the selected program interruption (single block or NC Stop):

NC Start from "Program Stop (single block)" state:
The position programmed in the next block is approached directly.

NC Start from "NC Stop" state:
The position at which the program was interrupted is approached initially; the next programmed NC block is then executed.

Example

```
%10
N005 X10 Z20 LF  (P1)
       Program is interrupted with NC Stop or with
       Program Stop (single block) and
       position P2 approached in JOG mode
N010 X0 Z40 LF   (P2)
```



END OF SECTION

8 Programming Motion Blocks

8.1 Rapid traverse (G00)

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

The path programmed using G00 is traversed at the maximum possible speed (rapid traverse) along a straight line without machining the workpiece (linear interpolation).

The maximum permissible axis speed is monitored by the control. The preparatory function G00 automatically initiates exact positioning (coarse).

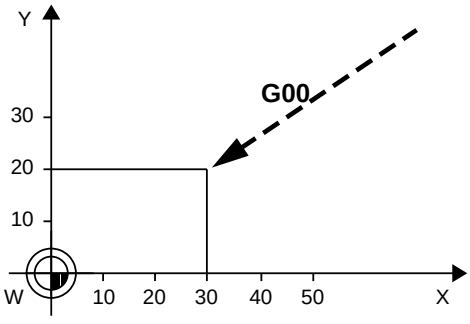
This speed is defined as machine data for each axis. If the rapid traverse is executed simultaneously in several axes, the traversing speed is determined by the lowest axis speed specified in the machine data.

Note:

When G00 is programmed, the feedrate programmed under address F remains stored and is made active again with G01.

Example:

```
N05  G00  G90  X30  Y20  LF
```



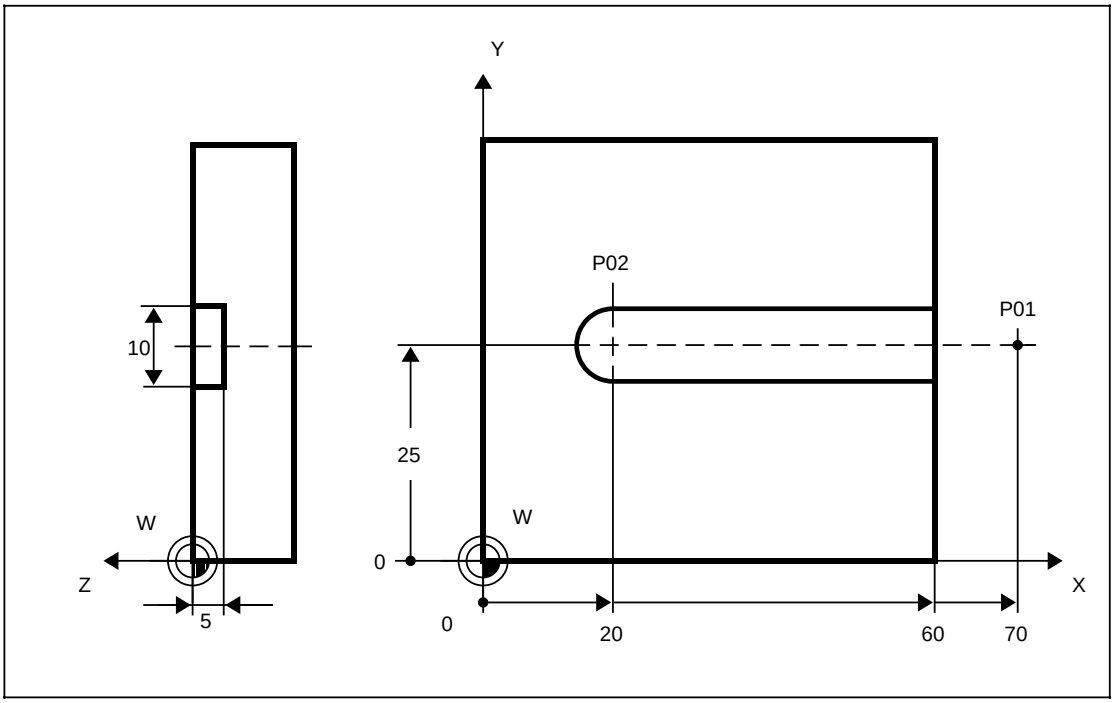
8.2 Linear interpolation (G01)

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

Linear interpolation affects motion

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

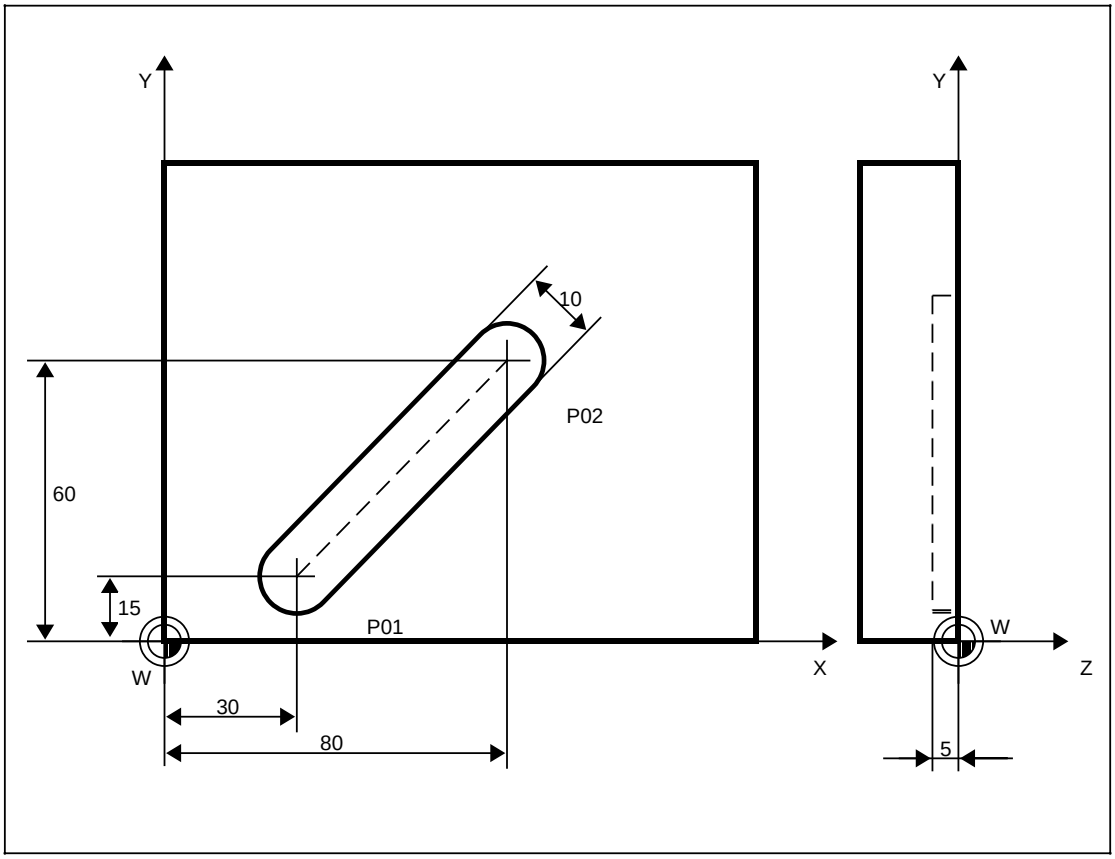
Example of paraxial milling:




```
% 15 LF
N05 G00 G90 X70 Y25 Z1 S800 M3 LF
N10 Z-5 LF
N15 G01 X20 F150 LF
N20 G00 Z100 LF
N25 X-25 Y50 LF
N30 M30 LF
```

N05	Spindle on, tool rapid traverse to P01, clockwise rotation at 800 rev/min
N10	Infeed in Z
N15	Tool traverse P01 to P02, feedrate 150 mm/min
N20/N25	Rapid traverse retraction
N30	End of program

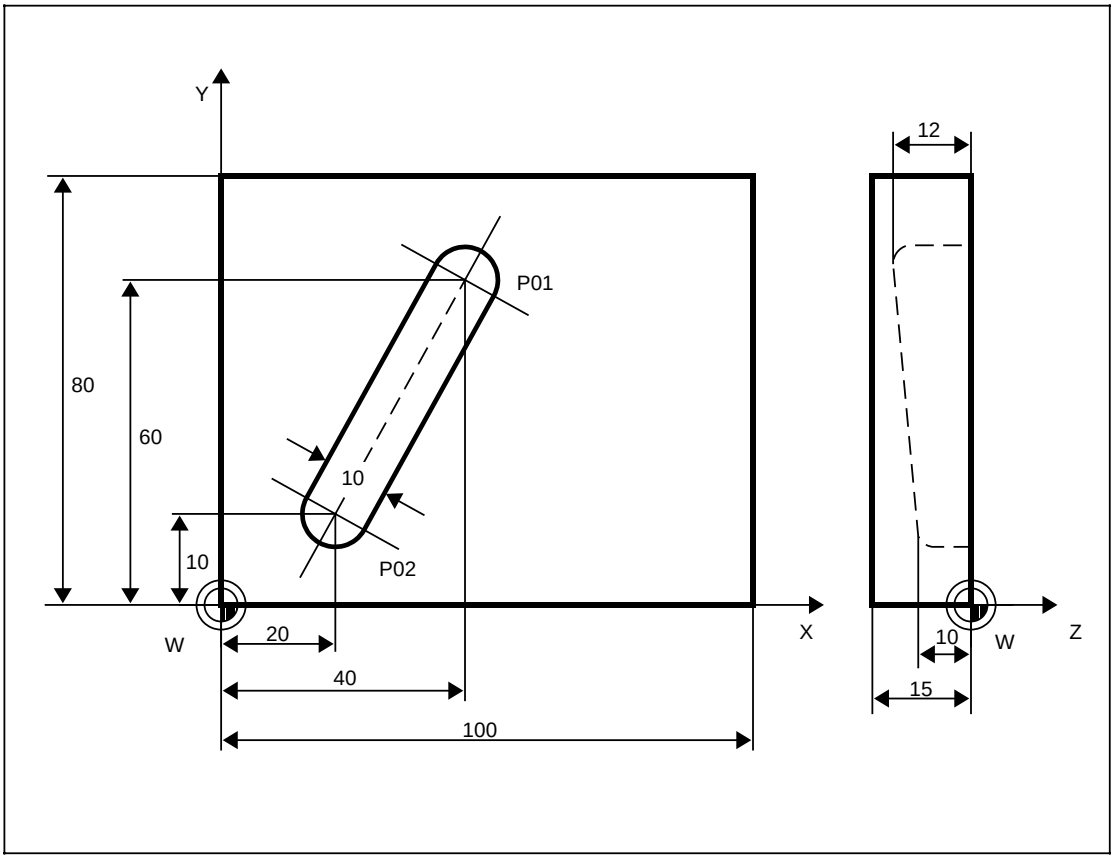
Example of linear interpolation in 2 axes:




```
% 25 LF
N05 G00 G90 X30 Y15 Z1 S500 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
```

N05 Tool rapid traverse to P01 with coordinates X30, Y15
N10 Infeed to Z-5, feedrate 100 mm**/**min
N15 Tool traverse along a straight line in space from P01 to P02, feedrate 200 mm**/**min
N20/N25 Rapid traverse retraction
N30 End of program

Example of linear interpolation in 3 axes:




```
% 30 LF
N05 G00 G90 X40 Y60 Z2 S500 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
```

N05 Tool rapid traverse to P01
N10 Infeed to Z-12, feedrate 100 mm/min
N15 Tool traverse along a straight line in space to P02
N20/N25 Rapid traverse retraction
N30 End of program

8.3 Circular interpolation (G02/G03)

The tool is to travel between two points on the contour along a circular arc and machine the workpiece. The control calculates the traversing path by means of circular interpolation.

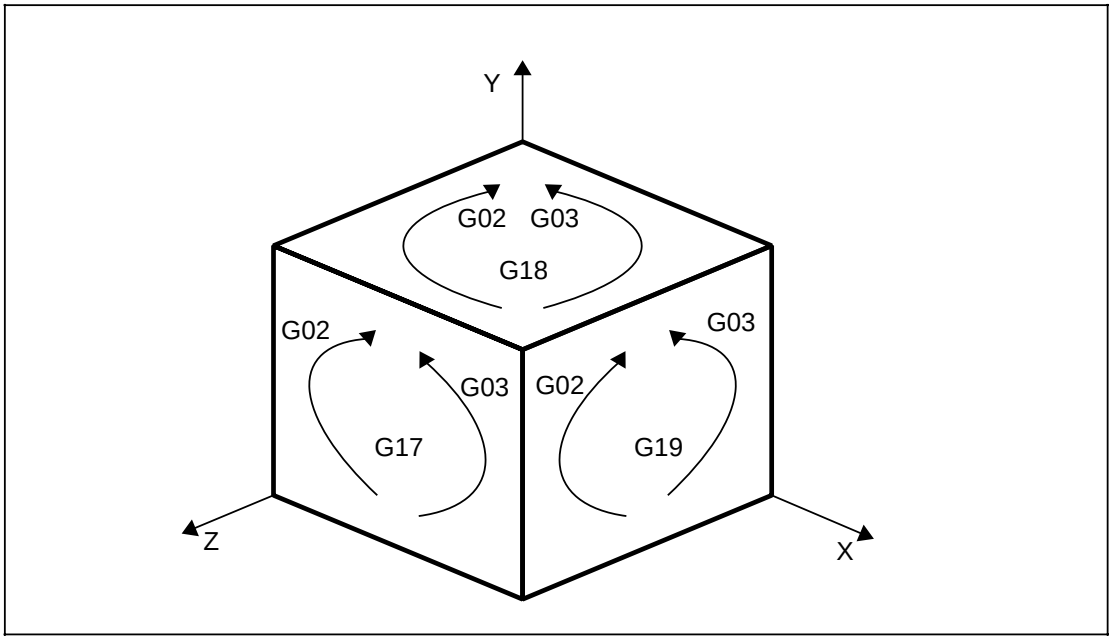
Circular interpolation causes the tool to move relative to the workpiece

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

The preparatory functions G02 and G03 are modal.

The direction of rotation in the various tool offset planes is defined as follows, looking towards the axis perpendicular to the plane. The tool will move in a clockwise direction with G02 or in a counter-clockwise direction with G03.

To define the direction of rotation, a right-handed coordinate system is assumed in which the axes used to describe the circle are abscissa (horizontal axis) and ordinate (perpendicular axis).



If the circle is on the tool offset plane, set with G16 to G19, the direction of rotation is defined by the plane defined here.

However, if the circle is not on the tool offset plane, the written sequence of axis addresses is valid:
Abscissa=1st programmed axis and ordinate=2nd programmed axis.

N05 G17	Tool offset plane X/Y plane
N10 G02 Z C I.. J..	Z = 1st programmed axis, C = 2nd programmed axis

If only one axis is programmed for the circle and this axis is on the tool offset plane, then the 2nd axis of this plane is automatically added. If the programmed axis is not part of the tool offset plane, alarm 3006 (Wrong block structure) is given.

```
N05  G17                    (Tool offset plane X/Y plane)
N10  G02 X  I..    J..      (2nd axis is automatically added
..                          because of plane)
N50  G02 Z                  (Alarm 3006 Wrong block structure)
```

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

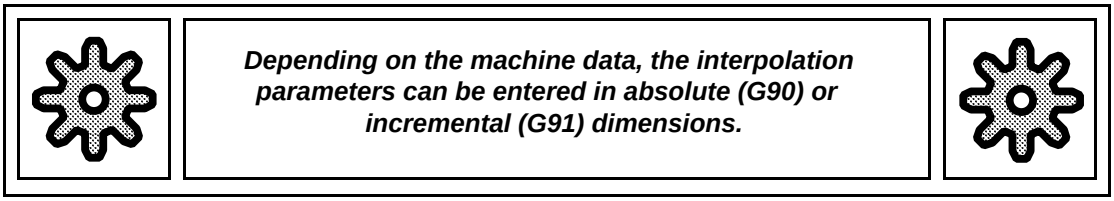
- by the interpolation parameters (see Section 8.3.1) or
- directly using the radius (see Section 8.3.2).

Note:

In a circle block, the path feedrate is reduced to the lowest feedrate of the axes included. Message 3092 "Preset feedrate too high" is output.

8.3.1 Programming a circle by means of interpolation parameters (I, J, K)

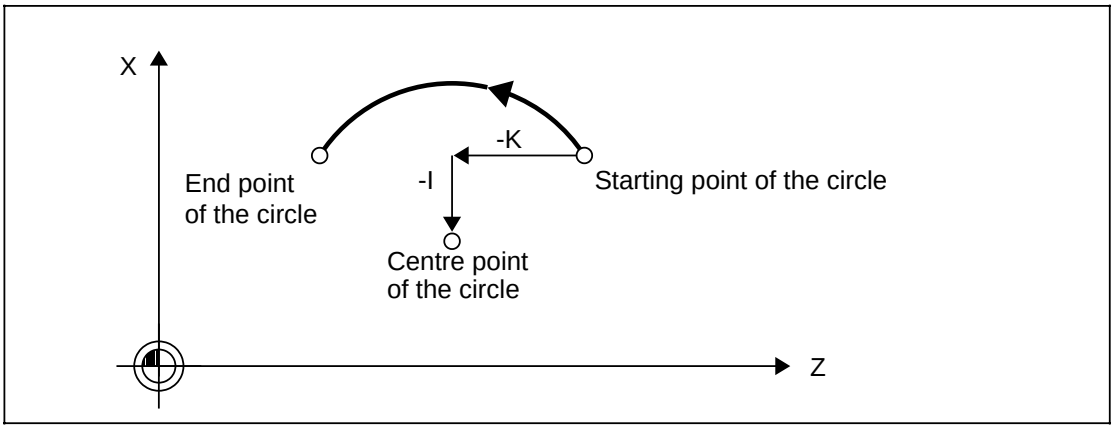
The interpolation parameters are the paraxial coordinates of the distance vector from the starting position to the centre point of the circle. According to DIN 66025 interpolation parameters I, J and K are allocated to axes X, Y and Z.



Depending on the machine data, the interpolation parameters can be entered in absolute (G90) or incremental (G91) dimensions.

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.

Also, the end point coordinate that is unchanged with respect to the circle need not be programmed. At least one axis must be programmed for a full circle.

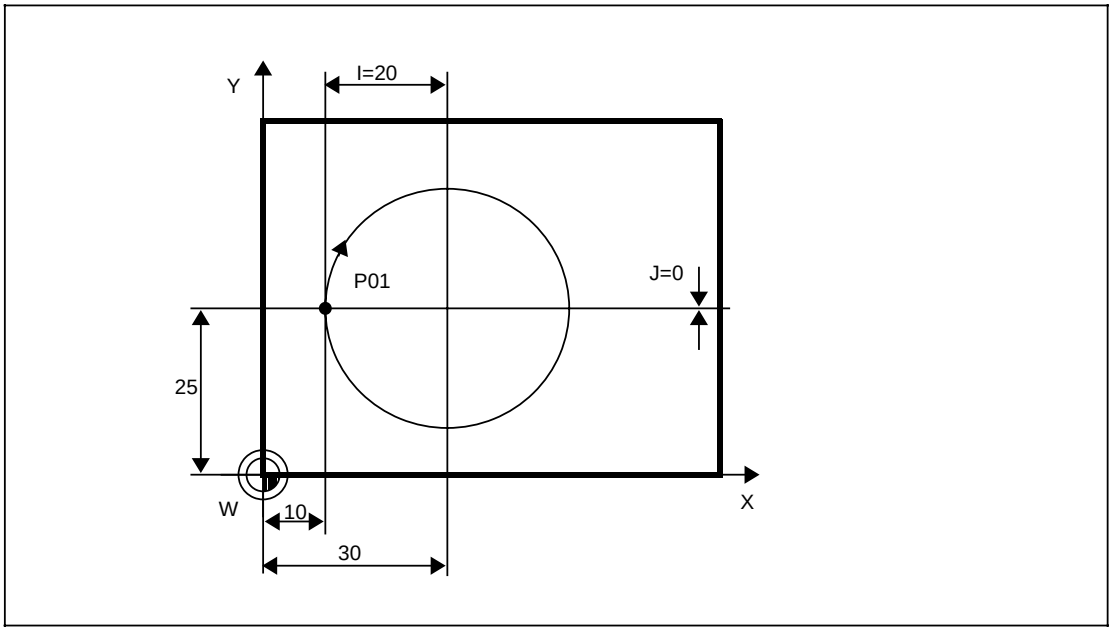


Circular interpolation with interpolation parameters

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

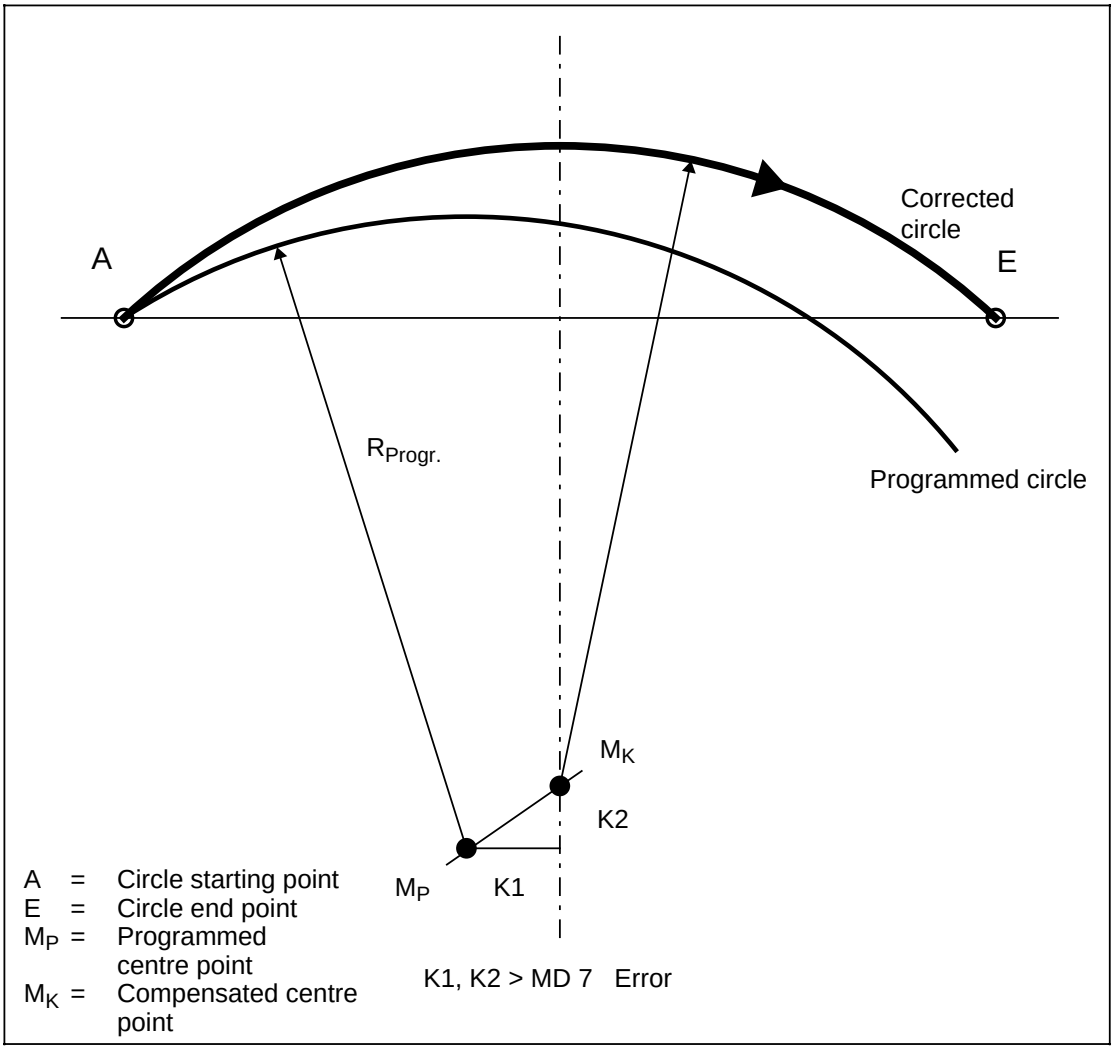
```
% 40 LF
N05 G00 X10 Y25 Z1 S1250 M3 LF
N10 G01 Z-5 F100 LF
N15 G02 X10 Y25 I20 J0 F125 LF
N20 G00 Z100 M5 LF
N25 X-20 LF
N30 M30 LF
```

N05 Tool rapid traverse to point P01.
N10 Infeed to Z-5
N15 X-Y plane selected automatically (initial setting). Tool travels clockwise around a full circle (G02).
N20/N25 Rapid traverse retraction
N30 End of program.

Inaccurate input of radius or interpolation parameters (circle end point monitoring)

Before a circle block is processed, the NC checks the ”correctness” of the programmed values by determining the difference in the interpolation parameters for the starting and end points. If the difference exceeds the specified upper limit, the block is not enabled for processing. Alarm 2048 (Circle end point error) is issued.

If the difference is less than but not equal to zero, the centre point parameters are corrected since it is assumed that the end point has been correctly programmed. The circle with the new centre point is then traversed.



Principle of circle compensation

The tolerance range is determined by machine data.

8.3.2 Programming a circle by radius programming

In many cases the dimensions of a drawing are such that it is easier to specify radius B or 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:

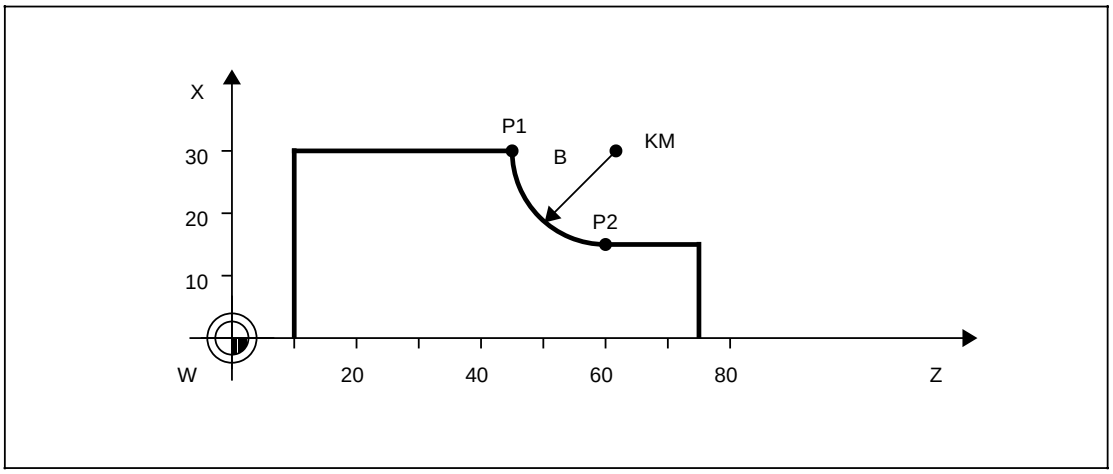
+B / +U: Angle less than or equal to 180°
–B / –U: Angle greater than 180°

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

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 B or U.

Example:

Radius programming



Example: Radius programming

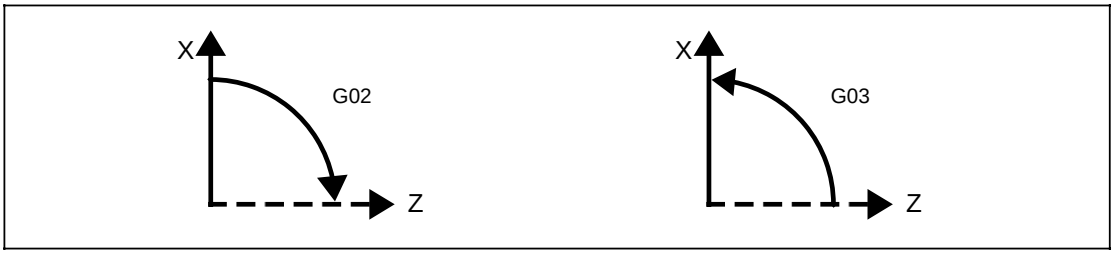
```
N05  G03  G90  X15  Z60  B15  LF
N10  G02  G90  X30  Z45  B15  LF
```

Tool travels from P1 to P2
Tool travels from P2 to P1

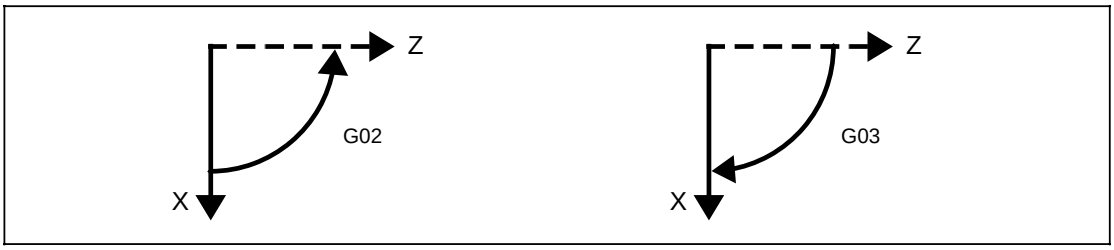
The radius designation is defined by machine data as follows: preferably B for turning machines and U for milling machines.

Turning machine:

Action of G02 and G03 behind and in front of the turning centre

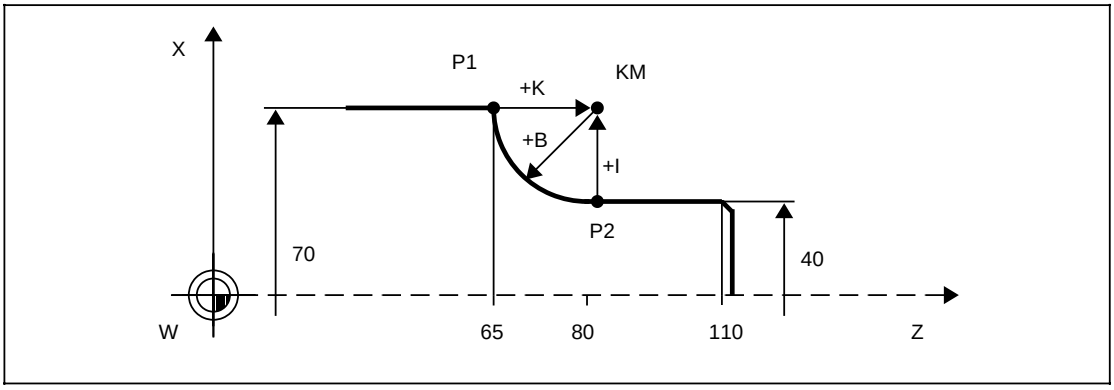


Example: Behind turning centre



Example: In front of turning centre

Programming example for turning machine:



Example: Circular interpolation

```
N05  G03  G90  X70.  Z65.  K15.  I0.  F500  LF
N10  G02  X40.  Z80.  K0.  I15.  LF
```

Tool travels from P1 to P2
Tool travels from P2 to P1
(Interpolation parameters)

or

```
N05  G03  G90  X70.  Z65.  B+15.  F500  LF
N10  G02  X40.  Z80.  B+15.  LF
```

Tool travels from P1 nach P2
Tool travels from P2 nach P1
(Radius programming)

8.4 Brief description of contour

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

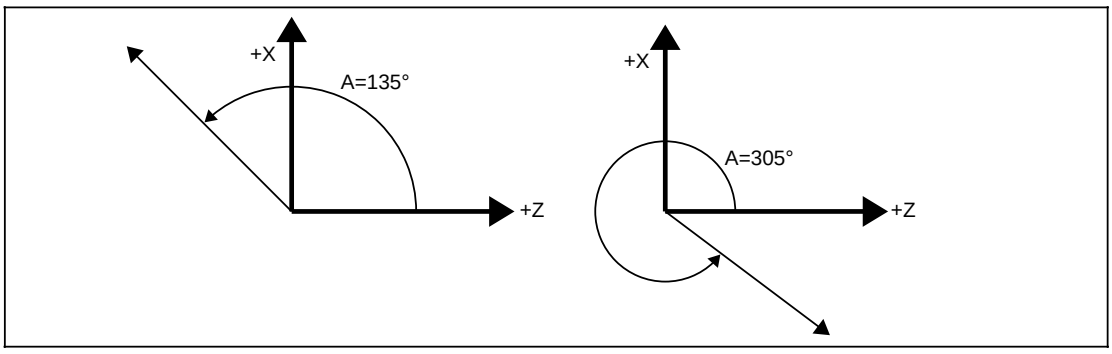
The various straight lines can be joined together directly in the form of a corner, rounded via radii or chamfered. Chamfer and transition radii are specified only by their size. The geometrical calculation is performed by the control. The end position coordinates can be programmed using either absolute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

Angle (A):

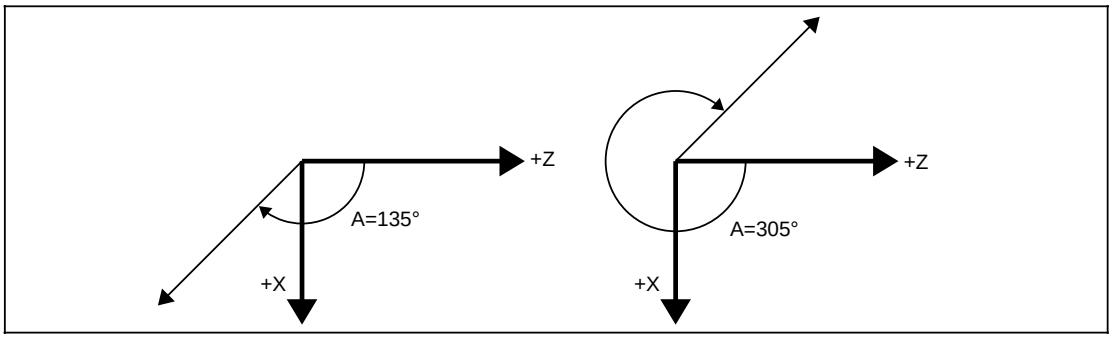
Input resolution 0.00001 corresponds to 10^{-5} degrees.

In the right-handed coordinate system, the angle (max. 359.99999 degrees) is always measured from the positive horizontal axis towards the positive vertical axis.

Turning machine



Right-handed system and working area behind turning centre

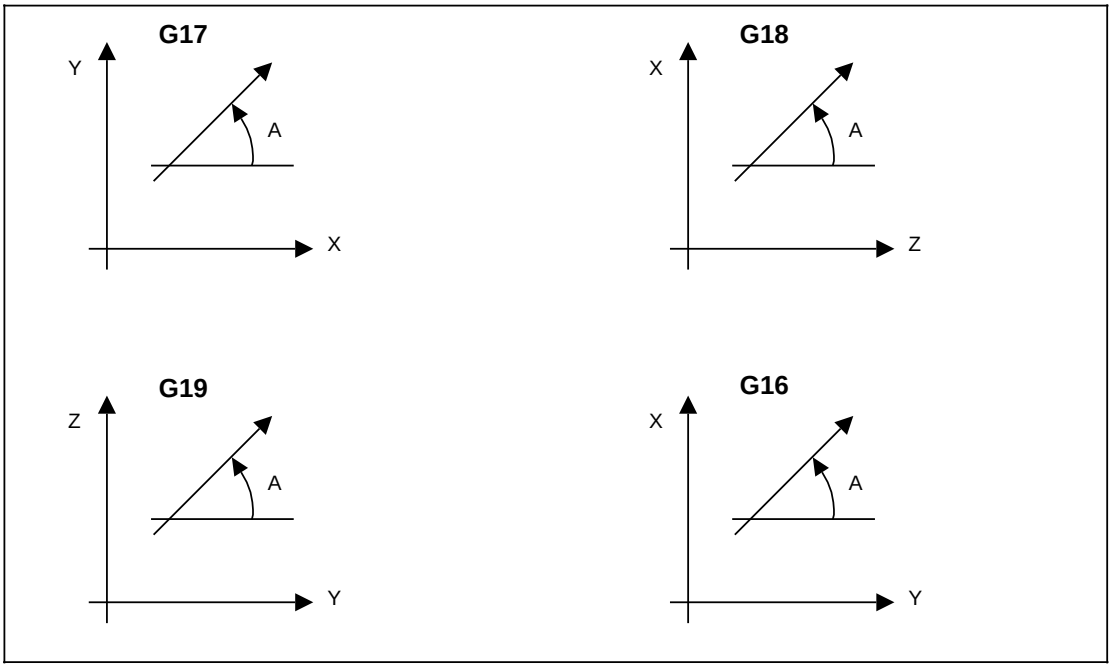


Right-handed system and working area in front of turning centre

Milling machine

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

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



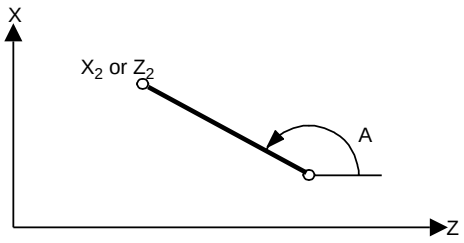
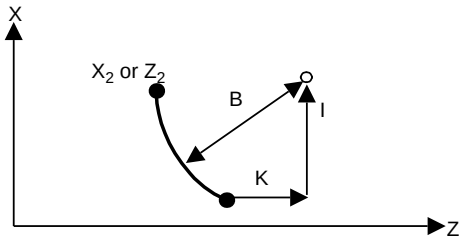
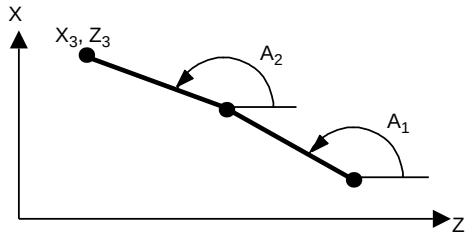
Plane selection

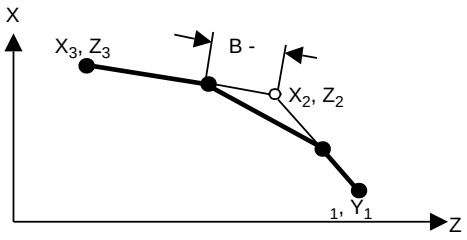
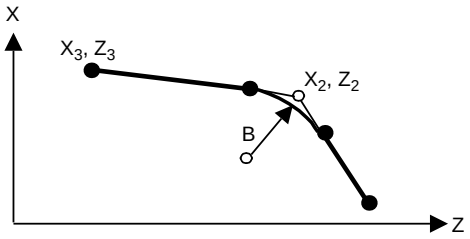
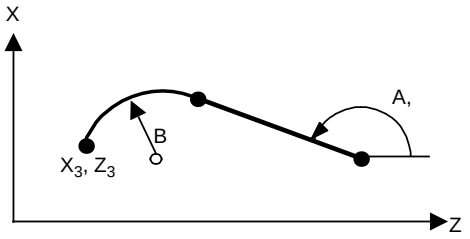
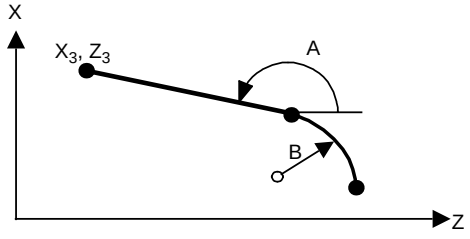
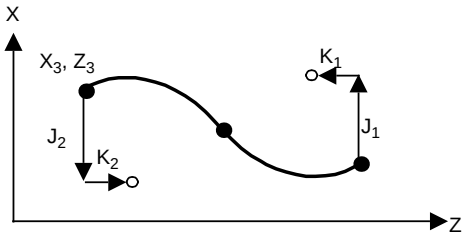
Contour definition programming is only permissible in the selected plane. 3D machining is not possible.

8.4.1 Contour definition programming

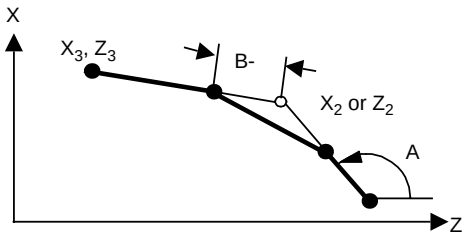
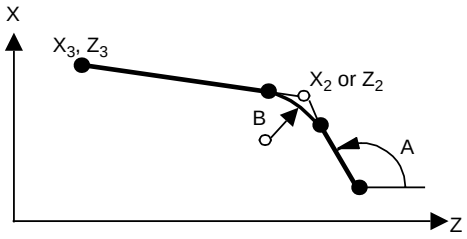
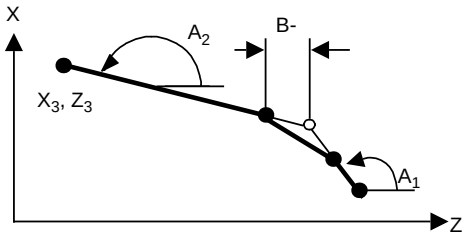
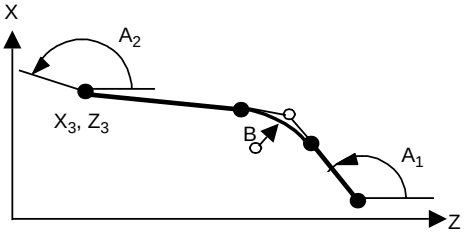
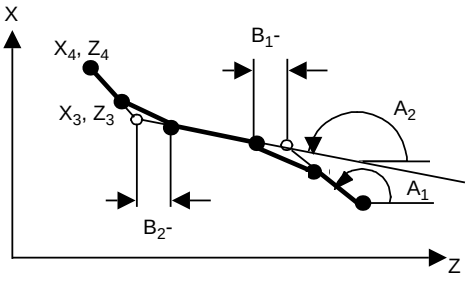
The elements described apply to a turning machine with working area behind the turning centre and to a milling machine in the selected plane Z-X (G18).

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.

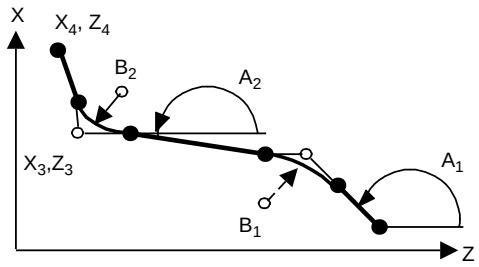
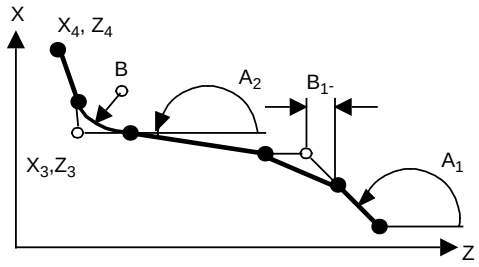
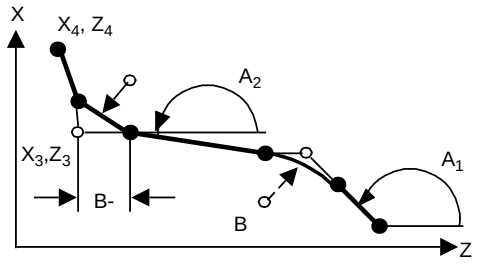
Function	Programming	Example
(1) 2-point definition	N... A... X_2 ... (or Z_2) L_F The second end coordinate is calculated by the control.	
(2) Circular arc	N...G02 (or G03) I.. K.. B.. X_2.. (or Z_2) L_F The circular arc is limited to one quadrant. The second end position coordinate is calculated by 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.. L_F The control calculates the coordinates of the vertex and generates 2 blocks. Angle A2 is referred to the second straight line.	

Function	Programming	Example
(4) Chamfer	N... X_2... Z_2... B-... L_F N... X_3... Z_3... L_F 1) 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)	
(5) Radius	N... X_2 ... Z_2 ... B... L_F N... X_3 ... Z_3 ... L_F 1) The inserted radius must not be larger than the smaller of the two paths.	
(6) Straight line - circular arc (tangent)	N.. G02 (or G03) A... B... X_3... Z_3... L_F The circular arc must not exceed 180°. The sequence A (angle) followed by B (radius) must be used.	
(7) Circular arc - straight line (tangent)	N.. G02 (or G03) B... A... X_3... Z_3... L_F The circular arc must not exceed 180°. The sequence B, A must be used. A radius must not be inserted in X_3, Z_3.	
(8) Circular arc - circular arc (tangent)	N.. G02 (or G03) I_1... K_1... I_2... K_2... X_3... Z_3... L_F Circle 1 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.	

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- ... L_F N... X_3... Z_3... L_F 1)	
(1) + (5) 2-point definition + radius	N... A... X_2 (or Z_2...) B... L_F N... X_3... Z_3... L_F 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- ... L_F	
(3) + (5) 3-point definition + radius	N... A_1.. A_2... X_3... Z_3... B... L_F	
(3) + (4) + (4) 3-point definition + chamfer + chamfer	N... A_1... A_2... X_3.. Z_3... B_1- ... B_2- ... L_F 1) N... X_4... Z_4... L_F 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... A_1... A_2... X_3... Z_3... B_1... B_2... L_F N... X_4... Z_4... L_F 1) Addition of a second radius at end position X_3, Z_3.	
(3) + (4)+ (5) 3-point definition + chamfer + radius	N... A_1... A_2... X_3... Z_3... B_1-... B... L_F N... X_4... Z_4... L_F1) Addition of a radius at end position X3, Z3. The next block is always taken into consideration automatically.	
(3) + (5)+ (4) 3-point definition + radius + chamfer	N... A_1... A_2... X_3... Z_3... B... B-... L_F N... X_4... Z_4...L_F1) Addition of a chamfer B- at the end position.	

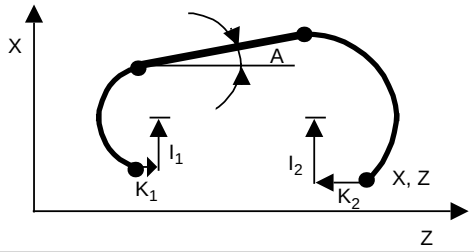
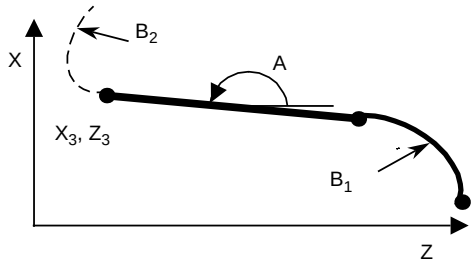
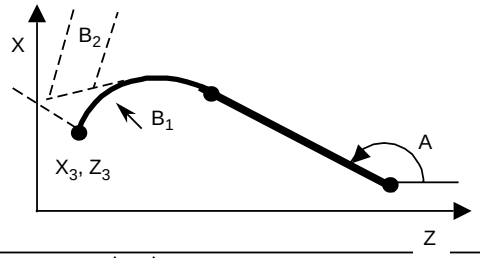
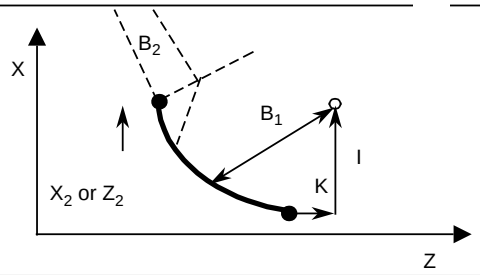
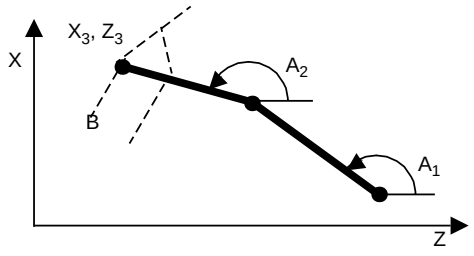
B0 must be programmed for corners where no chamfer or radius is to be inserted if a further radius or chamfer follows in the contour definition.

When this is programmed, the control generates a block with a distance of 0. This must be noted if tool radius compensation 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

Extended contour definition programming G931 ... G935

Function	Programming	Example
Tangential transition circle - straight line - circle	N...G931 G02 (or G03) I_1... K_1... A... G02 (or G03) I_2... K_2... X... Z... L_F	
Circular arc + radius or chamfer	N...G932 G02 (or G03) B_1 ...A...X_3 Z_3 ...B_2 ...L_F N...G932 G02 (or G03) B_1....A...X_3 ...Z_3 ...B_2–...L_F	
Straight line- circular arc + radius or chamfer	N...G933 G02 (or G03) A...B_1 ...X_3 ...Z_3 ...B_2 –...L_F N...G933 G02 (or G03) A...B_1 ...X_3 ...Z_3 ...B_2 ...L_F	
Circular arc + radius or chamfer	N...G934 G02 (or G03) I...K...B1 ...X2 (or Z2)...B2– ...L_F N...G934 G02 (or G03 I...K...B1 ...X2 (or Z2)...B2 ...L_F	
3-point definition + radius or chamfer	N...G935 A1 ...A2 ... B–...X3 ...Z3 ...L_F N...G935 A1 ...A2 ...B...X3 ...Z3 ...L_F	

Depending on a machine data, extended contour definition programming can be applied.

8.4.2 Effect of functions G09, F, S, T, H, M in contour definition

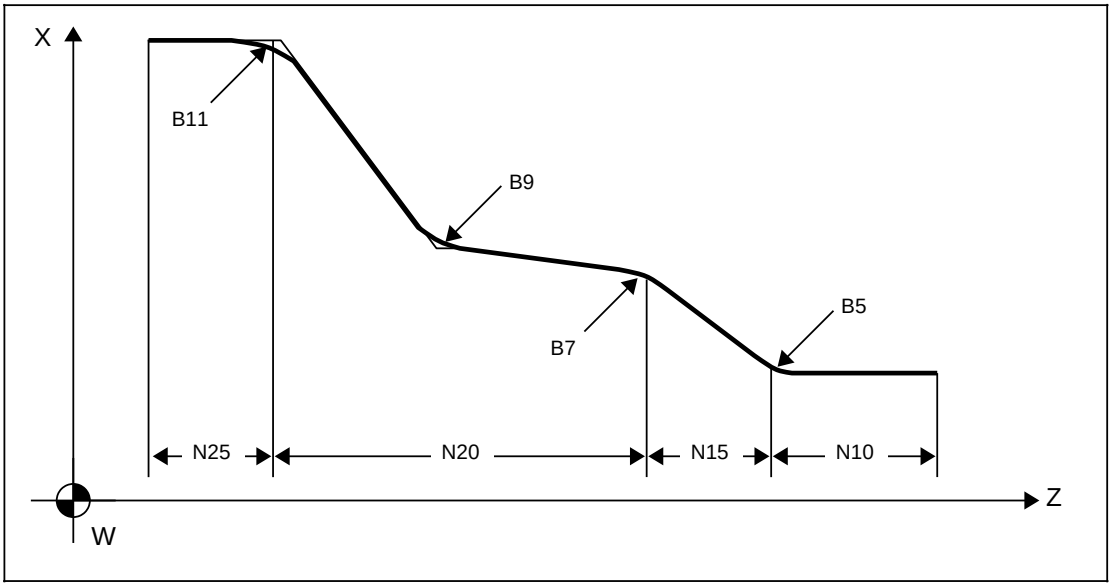
If **G09** is programmed in a contour definition block, it is not active until the end of the block, i.e. when the end position is reached. G09 is automatically generated by the control at irregular points (corners, edges) in the contour definition.

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

8.4.3 Chaining of blocks

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

Example 1: Turning machine, external machining

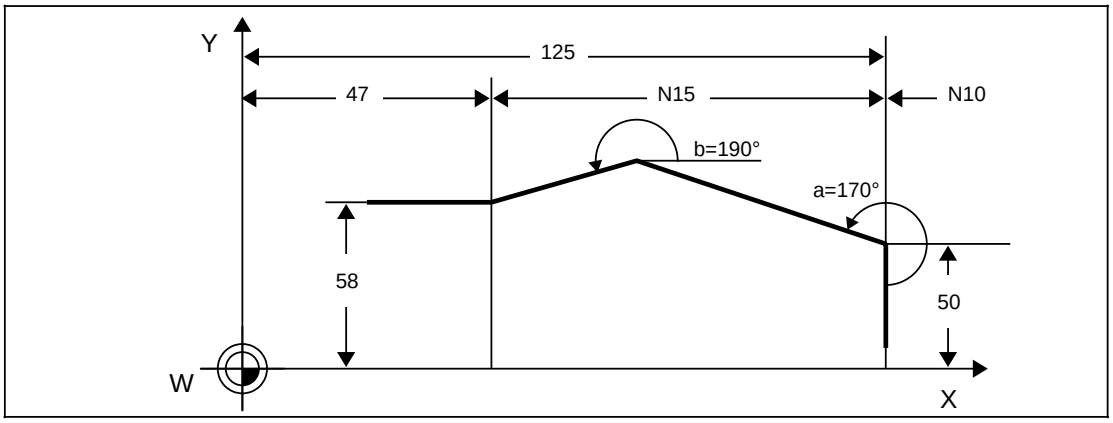


Chaining of blocks on a turning machine

```
N10  Z...  B5  LF
N15  A...  X...  B7.  LF
N20  A...  A...  X...  Z...  B9.  B11.  LF
N25  Z...  LF
```

Example 2: Milling machine

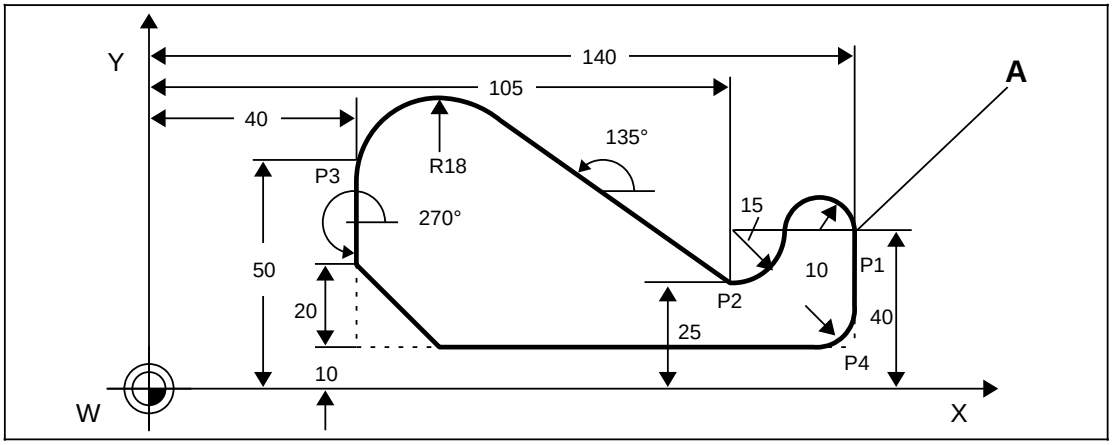
Angle a refers to the starting point; angle b refers to the missing intermediate point. The end point can be programmed using absolute point data G90 or incremental position data G91. Both end point coordinates must be specified. The control determines the intermediate point from the known starting point,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




```
.
N10  G00  G90  X125.  Y50.  LF
N15  G01  A170.  A190.  X47.  Y58.  F...LF
.
```

Example 3: 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.

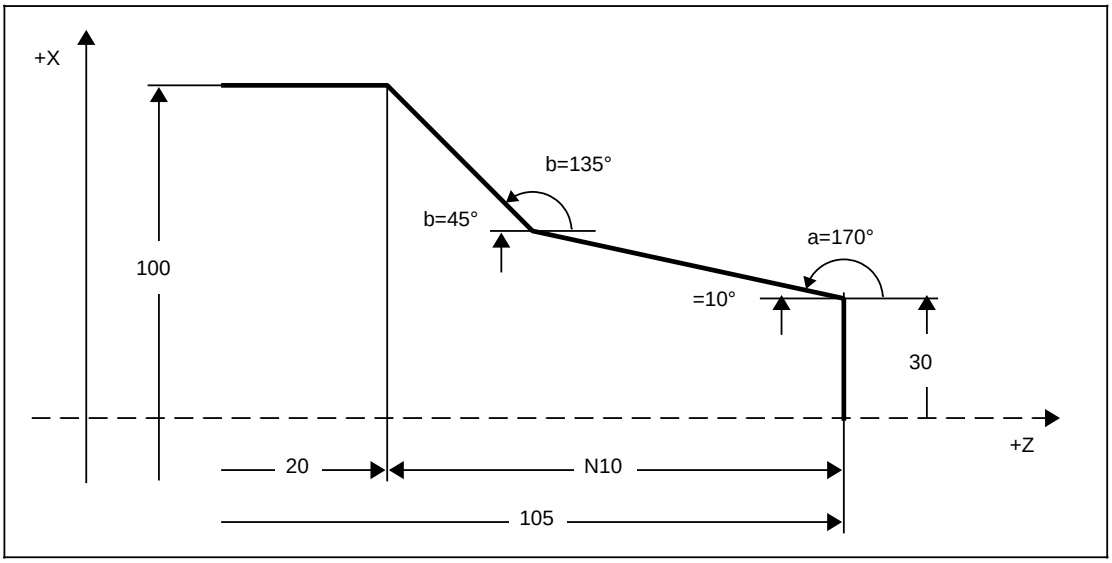


A = Start point

```
L168
N05  G90  G03  I-10.  J0.  I0.  J15.  X105.  Y25.  LF  (P2)
N10  G03  A135.  U18.  X40.  Y50.  LF (P3)
N15  G01  A270.  A0.  X140.  Y10.  U-20.  U10.  LF  (P4)
N20  Y40.  LF  (P1)
N25  M17  LF
```

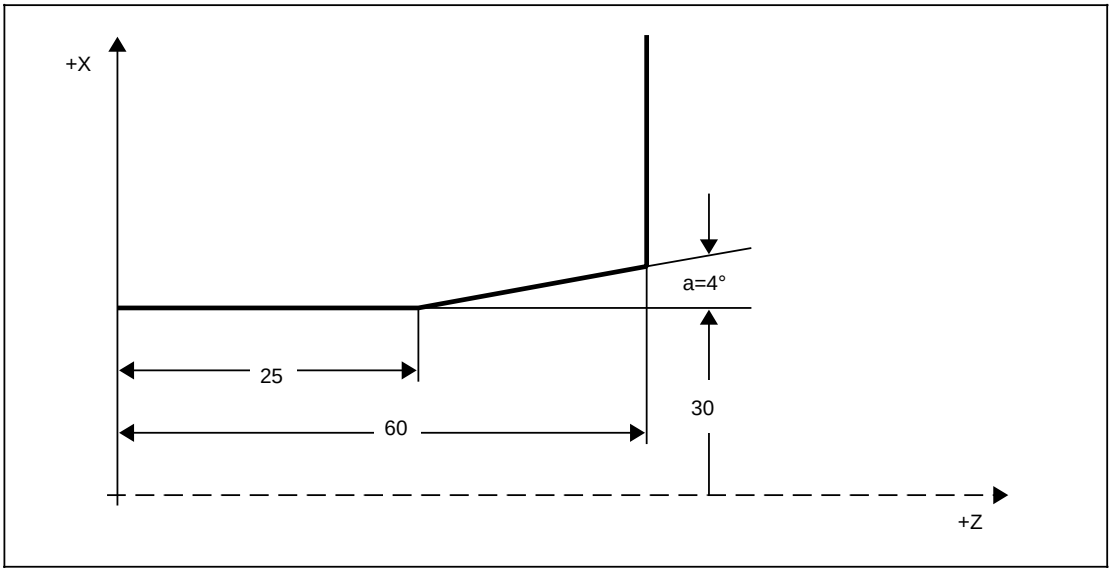
Example 4: Turning machine, external machining

Angle a refers to the starting point; angle b refers to the missing intermediate point. 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 intermediate point from the known starting point, the two angles and the end point.



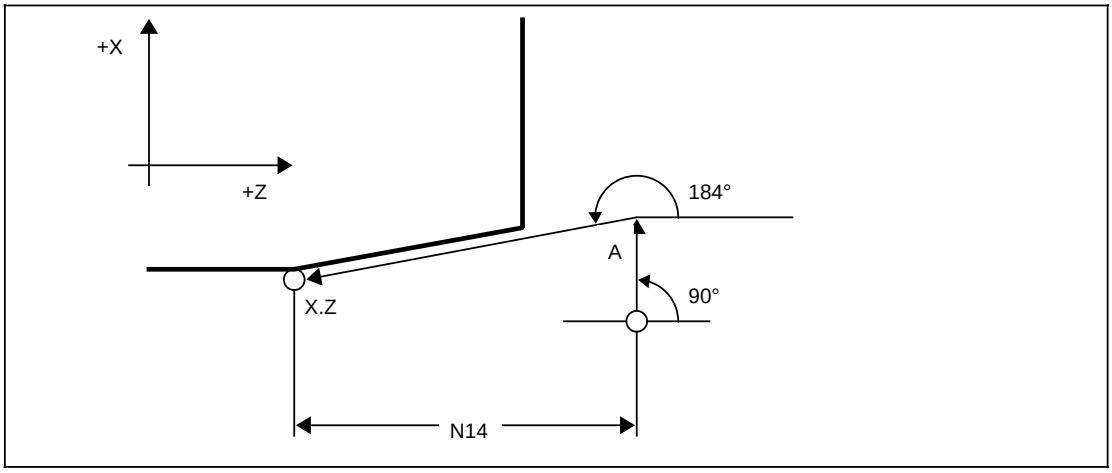

```
:
N05  G00  G90  X30.  Z105.  LF
N10  G01  A170.  A135.  X100.  Z20.  F...  LF
```

Example 5: Turning machine, internal machining



Drawing dimensions for example 5

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



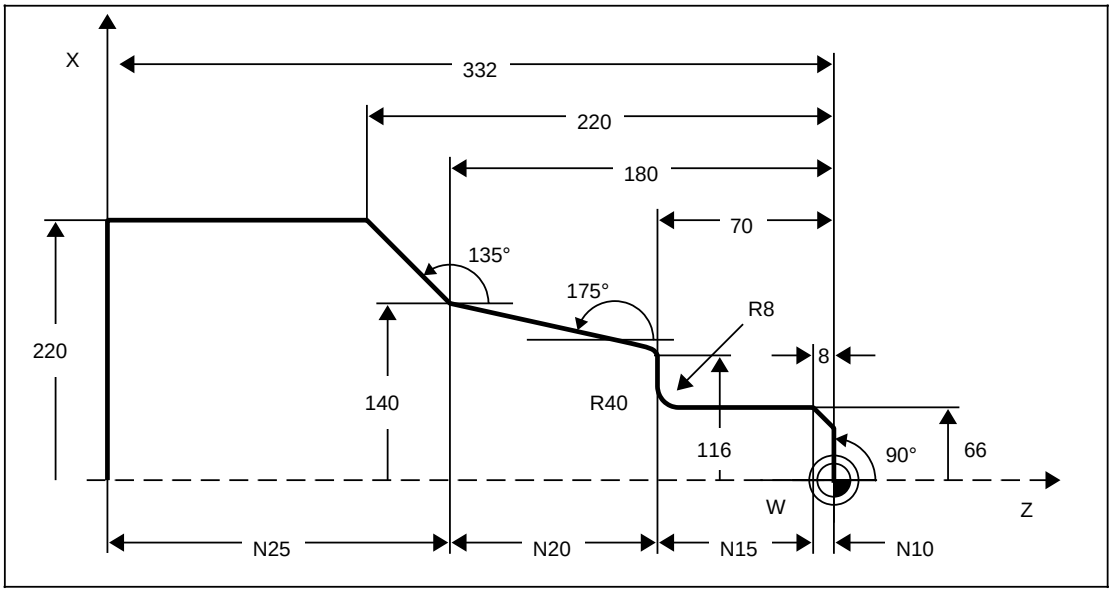
Drawing dimensions

The perpendicular through the start point and the extension of the inner taper gives the intersection A.

The program is then as follows:

```
:
:
N13  G00  Xstart  Zstart  LF
N14  G01  A90.  A184.  X...  Z...  LF
```

Example 6: External machining



Contour definition programming for external machining

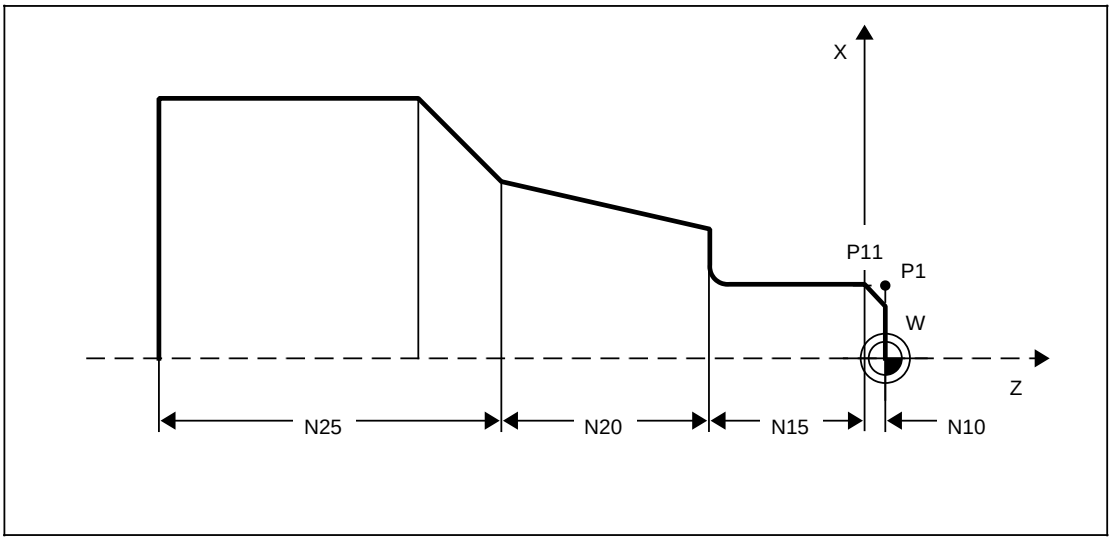
Blueprint programming

```
L105
N05  G00  G90  X.  Z0.  LF
N10  G01  G09  A90.  X66.  B–8.  F0.2  LF          (Chaining with B–)
N15  A180.  A90.  X116.  Z–70.  B8.  LF
N20  G03  B40.  A175.  X140.  Z–180.  LF
N25  G01  A135.  A180.  X220.  Z–332.  LF
N30  M17  LF
```

8.4.4 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 (including also arithmetic logic blocks with comments):

```
N10  G01  G09  A90.  X66.  B–8.  F0.2  LF          (P1)
N15  M...  H...
N20  A180.  A90.  X116.  Z-70.  B8.  LF
```

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

Tool retraction takes place at point P11. The F-value programmed in block N10 is active at the beginning of block N10.

Note:

A block with miscellaneous functions can be located between two chained blocks (including also arithmetic logic blocks with comments).

8.5 Polar coordinates (G10/G11/G12/G13/G110/G111/G112)

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

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

G10 Linear interpolation, rapid traverse
G11 Linear interpolation, feedrate
G12 Circular interpolation, clockwise
G13 Circular interpolation, counterclockwise
G110 Pole specification relative to the last position reached
G111 Pole specification absolute with reference to the workpiece zero
G112 Pole specification relative to the last valid pole

The G functions G10, G11, G12 and G13 are modal; G110, G111 and G112 are non modal.

When G10/G11/G12/G13 are programmed for the first time, 2 axes must be programmed using absolute position data G90, as well as angle and radius. The sequence in which the axes are programmed defines the **pole plane**. The **pole** is defined by the programmed coordinate values.

Programming using incremental position data (G91) at a later stage alters the coordinate values of the present pole. This does not change the present pole plane. Only those axes used in the present pole plane can be programmed. The pole values can also be changed using the G functions G110, G111 and G112. The pole is modal and can be reset by means of M02/M30.

The angle/radius programmed with G10/G11/G12/G13 remain active until a block is programmed with G110/G111/G112 or M02/M30 is programmed.

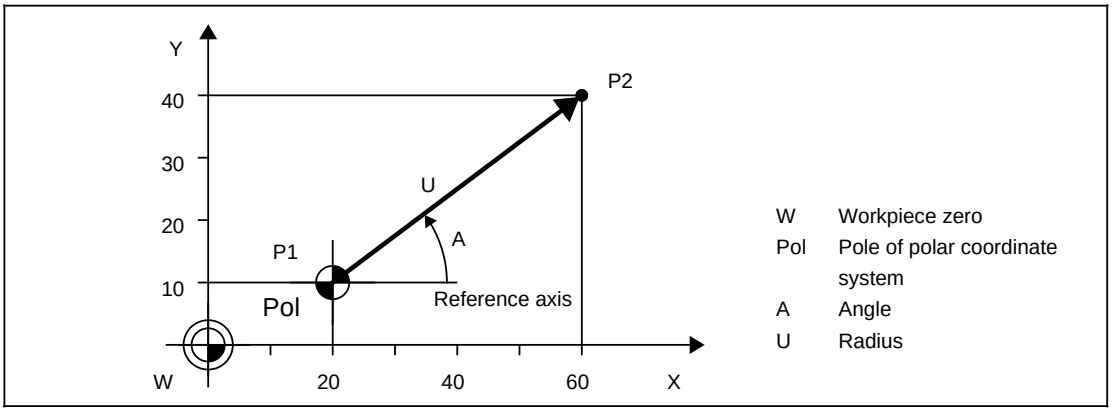
The radius is programmed without a sign. The angle can be entered with a sign (input resolution10^{-5} degrees). It always refers to the **first** axis of the pole (reference axis) to be **programmed**. The positive direction of this axis corresponds to an angle of 0 degrees.

Mixed programming with G10/G11/G12/G13 together G90/G91 in one block is possible if set in the machine data.

The designation for the radius is defined by machine data: Preferred usage is B for turning and U for milling machines; the preferred designation for the angle is A.

Whether the polar coordinate angle with G10/G11/G12/G13 is entered as an absolute dimension (G90) or as an incremental dimension (G91) is set in the machine data.

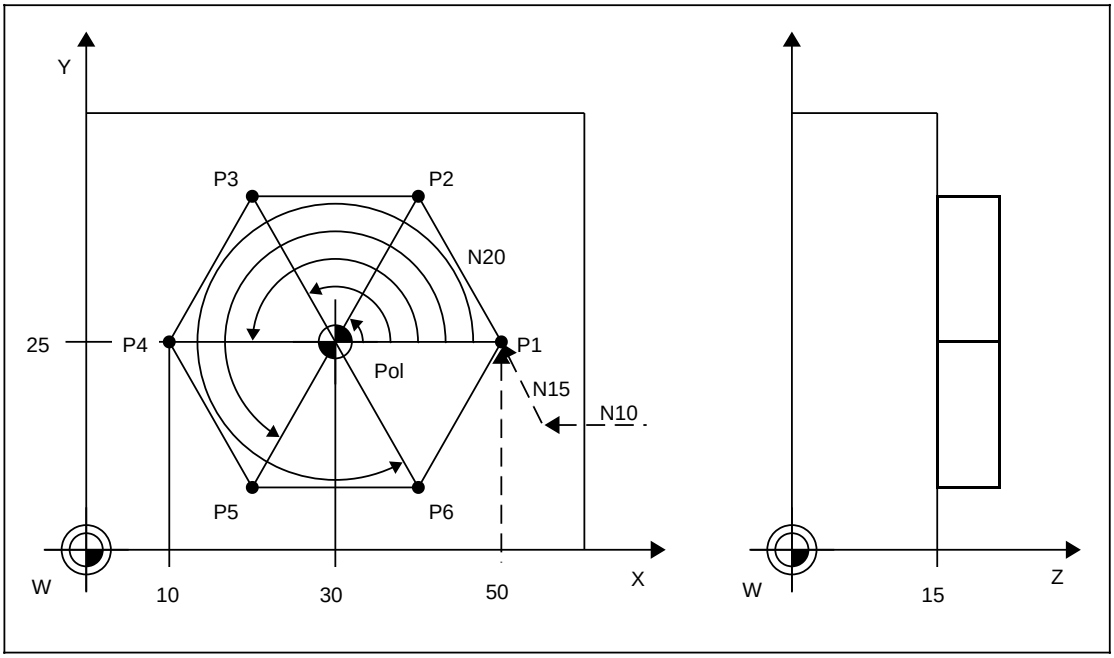
Example: G10 (Linear interpolation rapid traverse)




```
N5  G00  X0  Y0  LF
N10  G10  X20  Y10  A  ...  U ...  LF
```

(Approach workpiece zero)
(Approach P2 in rapid traverse)
(Pole plane X, Y: Pole values X20, Y10)

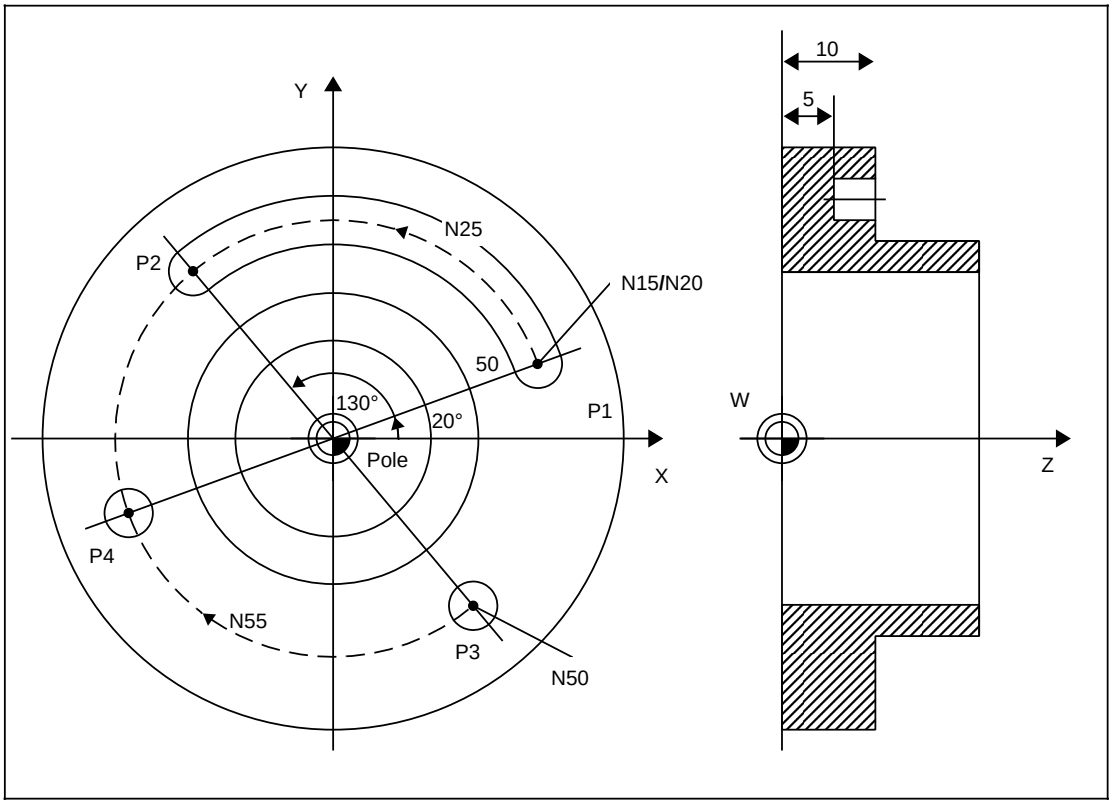
Example: G11 (Linear interpolation feedrate)




```
N10  G90  G0  Z15  LF
N15  G11  X30  Y25  U20  A0  F100  LF
N20  A60  LF (P2)
N25  A120  LF (P3)
N30  A180  LF (P4)
N35  A240  LF (P5)
N40  A300  LF (P6)
N45  A    0  LF (P1)
N50  G0  Z100  LF
```

(Infeed in Z)
(Pole definition and travel to P1)

Example: G12, G13 (Circular interpolation in clockwise direction/counterclockwise direction)

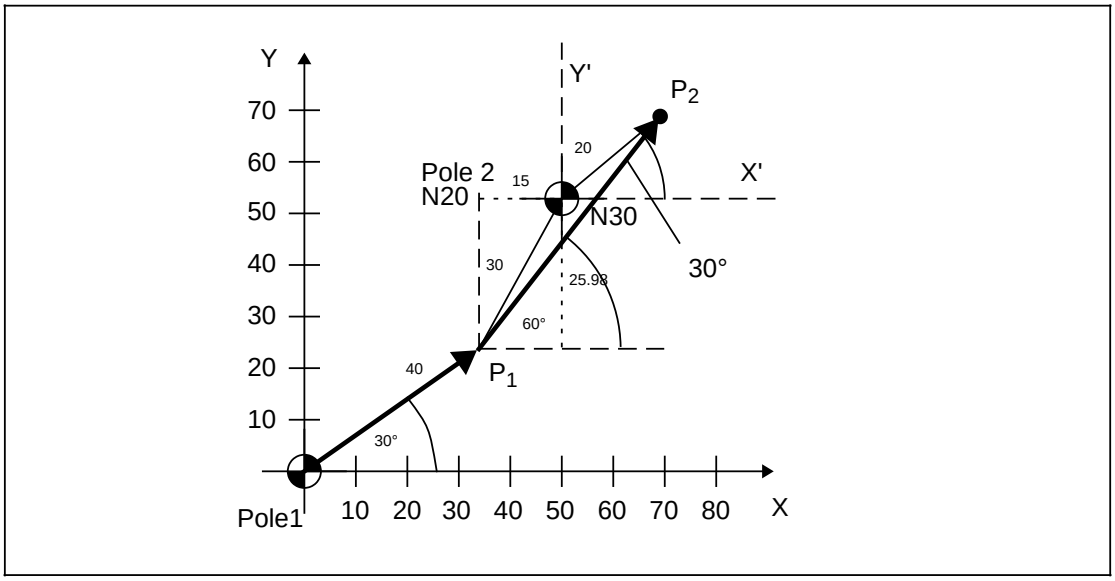



```
N05  G90  G0  X200  Y300  Z150  LF          (Pole definition and travel to P1)
N10  G90  G10  X0  Y0  U50  A20  LF         (Approach P1)
N15  G0 Z10  LF
N20  G1  Z5  F1000  S800  M3  LF            (Infeed in Z)
N25  G13  A130  LF                          (Slot milling from P1 to P2)
N30  G0  Z100  LF                           (Approach tool change point)
N35  G0  X100  Y100  M5  LF                 (Approach tool change point)
N40  T.2  M...  LF                          (Change tool)
N45  Z20  S800  M3  D2  LF
N50  G81  G10  A310  R2=..  R3=..  LF (P3)  (Approach first drilling position with cycle
                                            L81)
N55  G12  A200  R2=...  R3=...  LF (P4)     (Approach second drilling position and
                                            automatic call of cycle L81)
N60  G80  G0  Z100  LF                      (Deselect drilling cycle and back out)
```

Common features when specifying poles (G110/G111/G112)

- In a block containing G110/G111/G112 either axes or angle and radius are permitted.
- They do not initiate traverse movements.
- They are independent of a programmed G90/G91.
- A pole plane must already have been defined (with G10/G11/G12/G13) before G110/G111/G112 are called for the first time.
- The function is executable in MDA, AUTOMATIC and TEACH IN modes.
- The angle and radius are always taken as absolute values with every function of the type G110/G111/G112 A.. B.. . Modal values of A and B of the functions G10/G11/G12 and G13 are deleted. It is clearer for the user and a safer programming style if the modal A and B values are reset to zero when a pole offset is reset. The usual applications of this function (e.g. drilling patterns) are not affected by the pole offset.
- G110/G111/G112 overwrite the present pole values but do not alter the pole plane. Only axes which are contained in the present pole plane can be programmed.
- The pole can be specified in Cartesian or polar coordinates.
- G110/G111/G112 on its own has the same effect as G110/G111/G112 X0 Y0 or G110/G111/G112 A0 B0.
- G110/G111/G112 A20 has the same effect as G110/G111/G112 A20 B0 and G110/G111/G112 B20 has the same effect as G110/G111/G112 A0 B20.

Example with G110 (Pole specified relative to last position to be reached)

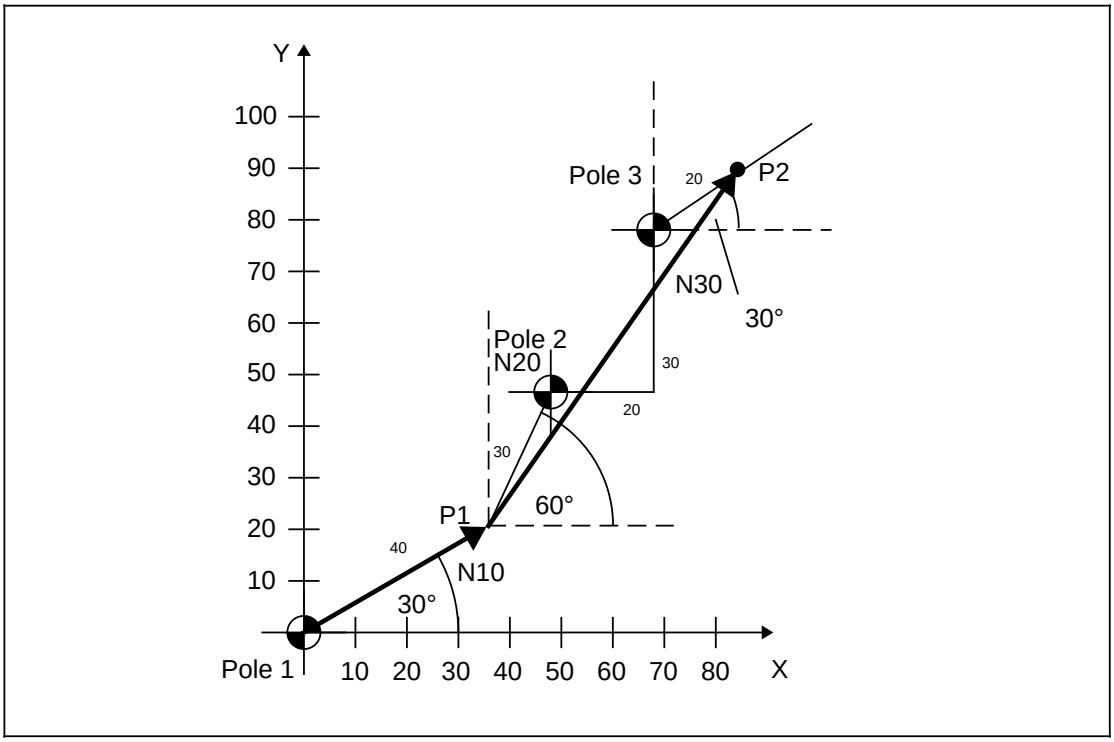



```
N5  G0  X0  Y0  LF
N10  G10  X0  Y0  A30  B40  LF      (Define pole 1 and travel to P1)
N20 G110 A60 B30  LF                (Define pole 2 in polar coordinates taking the block end
                                     point P1 into account)
N30 G11 A30 B20 F1000  LF           (Travel to P2 taking pole 2 into account)
N40 M30  LF
```

The above program causes the same traverse movement as this one:

```
N5  G0  X0  Y0  LF
N10 G10 X0 Y0 A30 B40  LF           (Define pole 1 and travel to P1)
N20 G110 X15 Y25.98  LF             (Define pole 2 using Cartesian coordinates taking the block
                                     end point P1 into account)
N30 G11 A30 B20 F1000  LF           (Travel to P2 taking pole 2 into account)
N40  M30  LF
```

Example with G110 (Pole specified relative to last position to be reached)

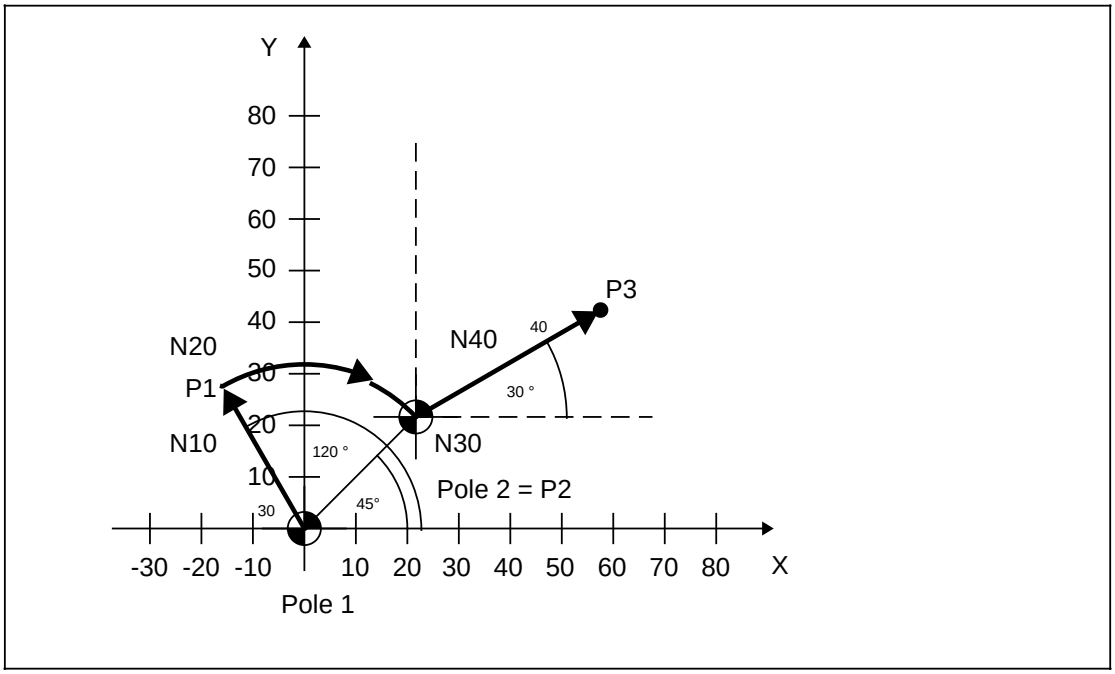



```
N05 G0 X0 Y0  LF
N10 G10 X0 Y0 A30 B40  LF                (Define pole 1 and travel to P1)
N20 G110 A60 B30  LF                     (Define pole 2 in polar coordinates taking the
                                          block end point P1 into account)
N30 G91 G11 X20 Y30 A30 B20 F1000  LF    (Define pole 3 taking pole 2 into account and
                                          travel to P2)
N40 M30
```

Explanation:

G110 is programmed in block N20; i.e. the values A and B of G10 in block N10 are deleted (see "Common features when specifying poles"). However, because G11 is programmed for the first time again in block N30, one angle has to be programmed absolutely again. If G10, G11, G12, G13 are programmed after G110, G111, G112, the angle has to be programmed for the first time in absolute form.

Example with G110 (Pole specified relative to last position to be reached)

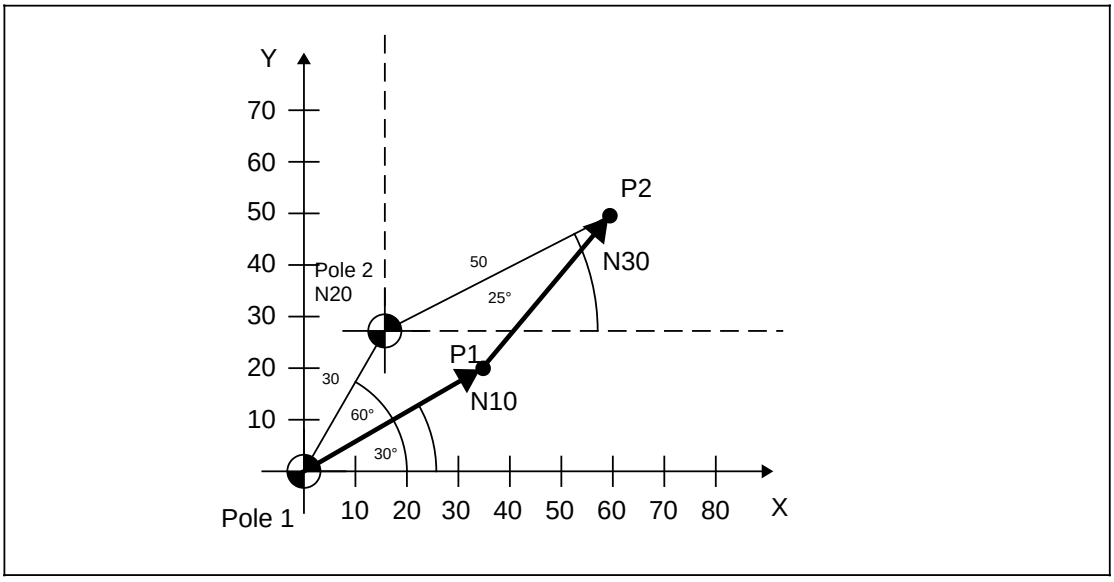



```
N5   G0  X0  Y0  LF
N10  G11  X0  Y0  A120  B30  F1000  LF   (Define pole 1 and travel to P1)
N20 G12 A45  LF                          (Travel to P2)
N30 G110  LF                             (Define pole 2)
```

The current position is taken as the new pole.

```
N40 G10 A30 B40  LF                      (Travel to P3 taking pole 2 into account)
N50  M30  LF
```

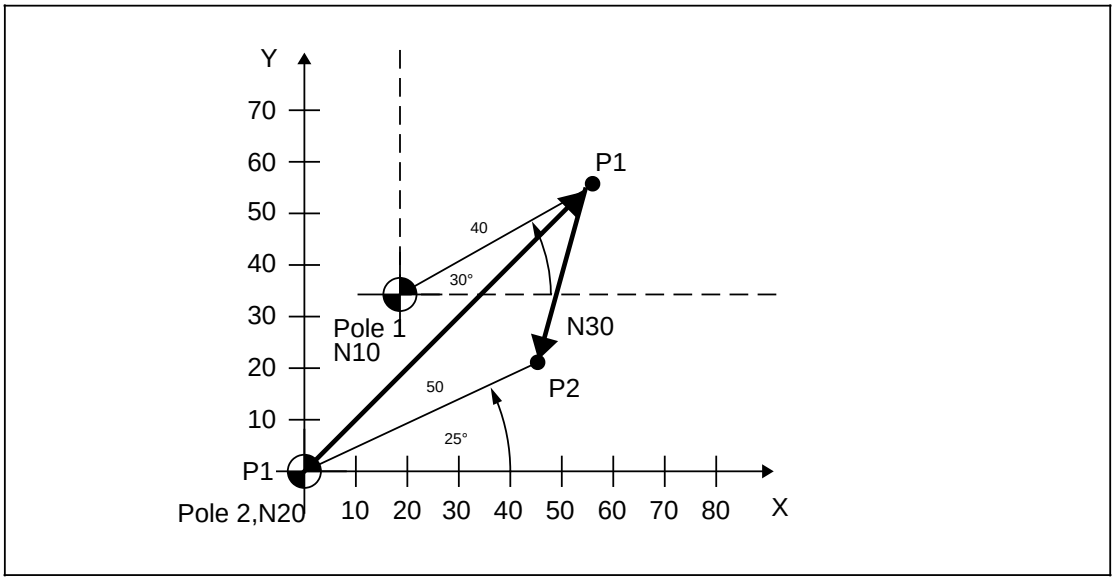
Example with G111 (Pole specified absolute with reference to workpiece zero)




```
N5   G0  X0  Y0  LF
N10  G10  X0  Y0  A30  B40      (Define pole 1 and travel to P1)
N20  G111  A60  B30             (Define pole 2 taking workpiece zero into account)
N30  G11  A25  B50  F1000       (Travel to P2 taking pole 2 into account)
N40  M30
```

This program functions just as well if you enter the Cartesian values (delta values) for X (15 mm) and Y (approx. 25.98 mm) instead of the values for the polar coordinates in block N20.

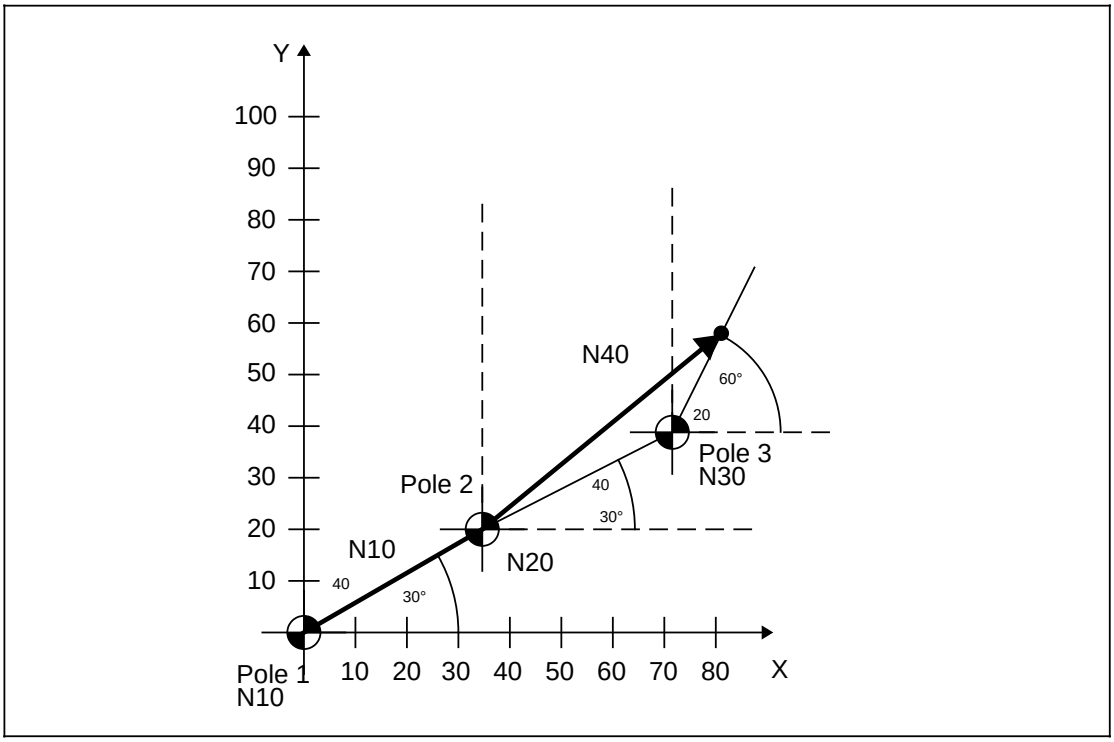
Example with G111 (Pole specified absolute with reference to workpiece zero)




```
N0   G0  X  Y  LF                          (Travel at rapid traverse to P1)
N10  G10  X20  Y35  A30  B40  LF           (Define pole 1 and travel to P1)
N20  G111  LF                             (Define pole 2; no traverse motion)
N30  G11  A25  B50  F1000  LF             (Travel to P2 from pole 2)
N40  M30  LF
```

G111 on its own has the same effect as G111 A0 B0 or G111 X0 Y0

Example G112 (Pole specified relative to last valid pole)




```
N10  G10  X0  Y0  A30  B40
N20  G110
N30  G112  A30  B40
N40  G11  A60  B20  F1000
N50  M30
```

(Definition of pole 1, motion to pole 2)
(Definition of the new pole 2; no motion)
(In contrast to that: definition of a new pole 3 **relative** to pole 2. Both polar (A.. B..) and Cartesian (X.. Y..) forms are possible, but never both at the same time in one block)
(Movement to the end point taking pole 3 into account)

G112 on its own is permitted but has no effect!

8.6 Helical interpolation

8.6.1 Helical line interpolation

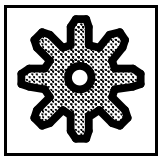
Helical line interpolation is possible between 3 linear axes which are perpendicular to one another.

Helical line interpolation involves programming a circular arc and a straight line perpendicular to its end point 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.

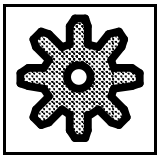
The axis coordinates X, Y and Z must always be programmed.

The circular plane is defined by the axes first programmed in the circular block. If the plane and interpolation parameters defined in machine data are not identical, alarm 3006 (Wrong block structure) is triggered.

The axis for linear interpolation can be programmed either before or after the interpolation parameters.



If an additional axis is incorporated in the circular interpolation, the interpolation parameter specified via machine data must be used for this axis.



The axes for circular interpolation must be programmed first.

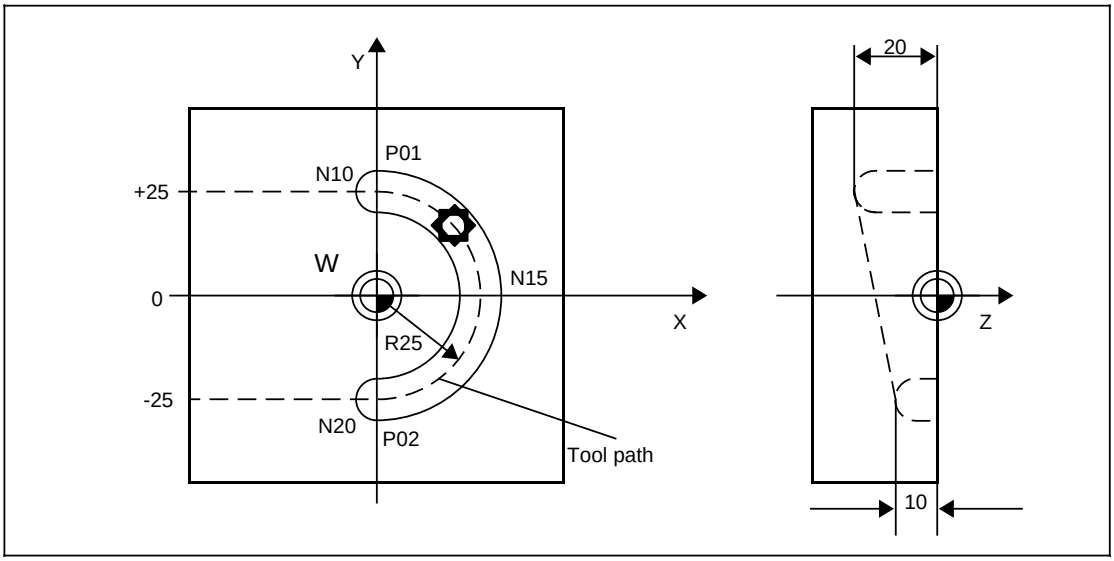
The required feedrate must be programmed before or in the helical block.

Note:

The programmed feedrate acts on the circular axes only, whilst control of the linear axes synchronizes motions so that they reach their end point when the circular axes reach the circle end point, i. e. the programmed feedrate is maintained on the circular path rather than on the actual tool path.

The maximum permissible velocity of the axes used for interpolation is monitored, and the feedrate is restricted if the limit is exceeded (i. e. Vprog > Vmax).

Example: Helical line interpolation

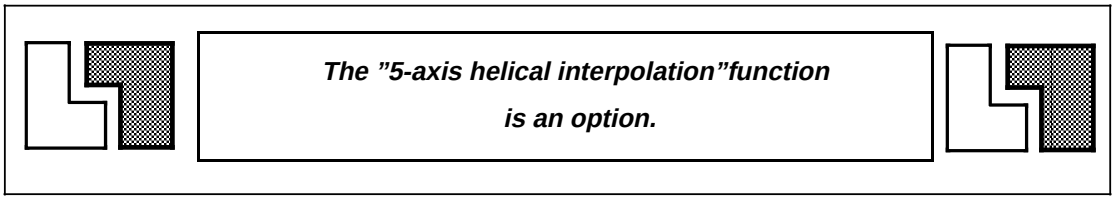


Helical interpolation

```
% 80 LF
N05 G00 X0 Y25 Z1 S800 M3 LF
N10 G01 Z-20 F150 LF
N15 G02 X0 Y-25 Z-10 I0 J-25 LF
N20 G00 Z100 M5 LF
N25 M30 LF
```

N05 XY plane selected automatically (initial setting)
Tool rapid traverse to point P01.
N10 Linear interpolation. Infeed to Z-20
N15 Tool clockwise (G02) traverse on a helix to P02
N20 Rapid traverse retraction
N25 End of program

8.6.2 5-axis helical interpolation [1)]



The 5-axis helical interpolation function enables two more axes to be programmed.

When programming, a fixed sequence within the block must be adhered to:

1. Circle information including
   – type of interpolation (G02/G03)
   – the circular axes,
   – the interpolation parameter or circle radius.
2. Linear information including
   – 1 to 3 axes to be traversed linearly.

Example:

```
N10 G01 X50 Y50 F500 LF
N20 G02 X90 Y70 I40 J–30 Z10 V20 W30 LF
```

The required feedrate must be programmed before or in the helical block.

The programmed feedrate acts on the circular axes only. The linear axes are controlled to reach the end point as the circular axes arrive at the circle end point.

The axes involved in interpolation are monitored for their maximum permissible velocity. If this limit is exceeded (i.e. $V_{prog}>V_{max}$) this will lead to a feedrate limitation.

1) only available with the standard version

8.7 Soft approach to and retraction from the contour (G147, G247, G347, G148, G248, G348, G48)

To avoid cutting marks, approach to and retraction from a contour is tangential.

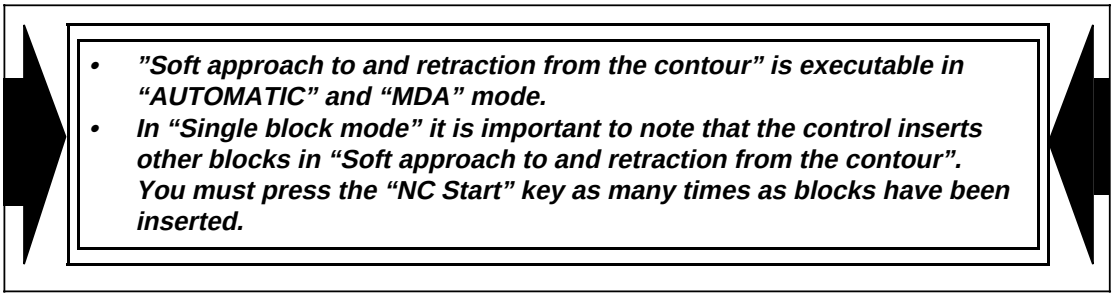
Soft approach to and retraction from a contour can be used for all types of tools. “Radius” and “Wear” are calculated.

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

G147 Approach contour linear
G247 Approach contour in quarter circle
G347 Approach contour in semi circle
G148 Retract from contour linear
G248 Retract from contour in quarter circle
G348 Retract from contour in semi circle
G48 Retract from contour as it was approached

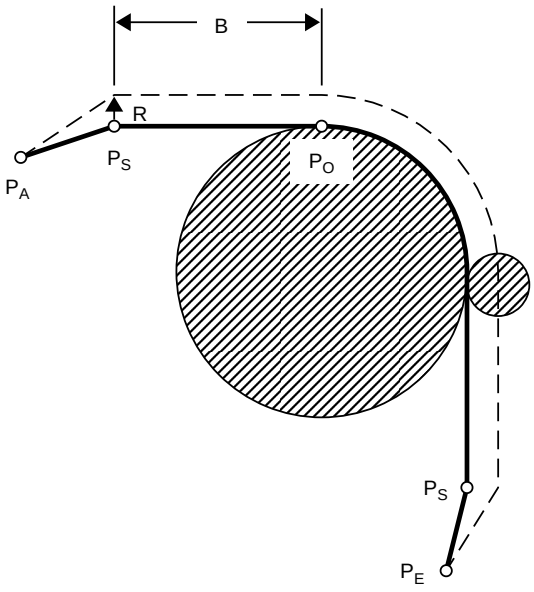
Programming characteristics

- The functions for approaching and retracting from the contour are **non modal**
- Tool radius compensation (TRC) (G41/G42) must be programmed before approach.
- The following must be specified in the approach block
  – coordinates of the initial point P_O of the contour
  – the value of B (minimum distance to contour)
- The following must be specified in the retraction block:
  – the coordinates of the final point P_E after retracting from the contour
  – the value of B (minimum distance to contour)
- No other traverse movements must be programmed in the approach and retraction blocks.
- Auxiliary functions can be programmed both in the approach block and in the retraction block. From SW5, an auxiliary function block an be programmed here.
- A pure auxiliary function block may not be programmed **after** an approach block or **before** a retraction block. From SW5, an auxiliary function block can be programmed here.
- For the combination of approach and retraction blocks it is important to note that “G40” (cancellation of the TRC) is generated in the retraction block by the control: program G41 or G42 before every new approach block!
- In the case of part programs to be prepared block by block (TEACH IN) the finished part program can be subsequently supplemented.



The address for the approach path is freely selectable via machine data (address B or U).

Example: Straight line to a circle



P_A... Starting point before approaching the contour

P_S... Intermediate point: is calculated by the control after “B” is specified

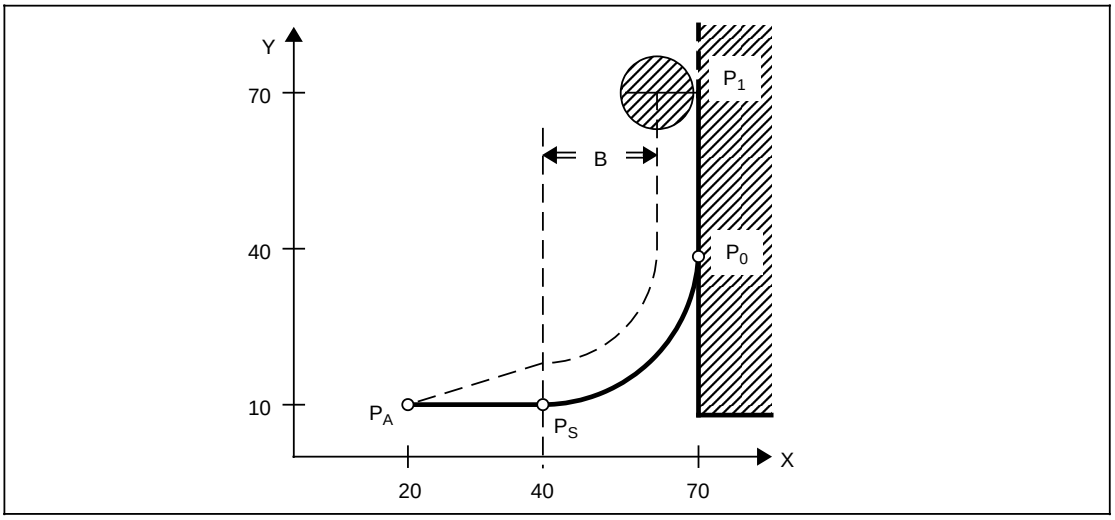
P_O... End point of the approach block (initial point of the contour)

B... Minimum distance to contour

P_E... End point after retraction from contour

R... Calculation of tool radius compensation (dotted line: centre point path of the tool)

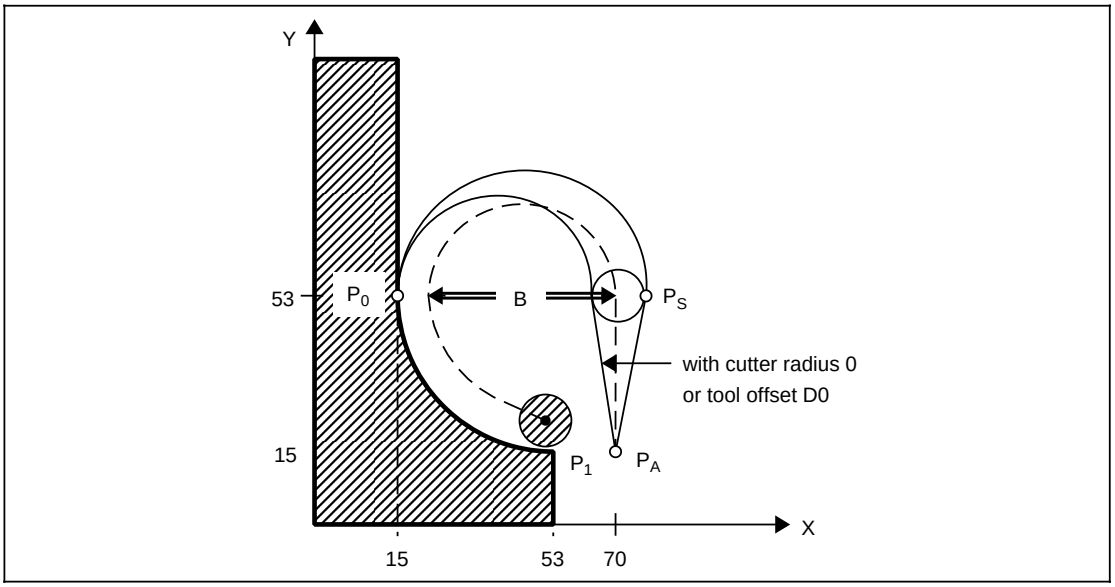
Example: Approaching a linear contour in a quarter circle (with program example)



Approaching a linear contour in a quarter circle

```
N05  G01  G41  Y10  X20  D...  F...  LF        (PA)
N10  G247  Y40  X70  B25  LF                   (P0)
N15  G01  Y70  LF                              (P1)
N20  M30  LF
```

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



Approaching an inside circle in a semicircle

```
N10  G01  G41  Y15  X70  D...  F...  LF        (PA)
N15  G347  Y53  X15  B50  LF                   (P0)
N20  G03  Y15  X53  38  J0  LF                 (P1)
N25  G0  Y15  X70  LF                          (PA)
N30  M30  LF
```

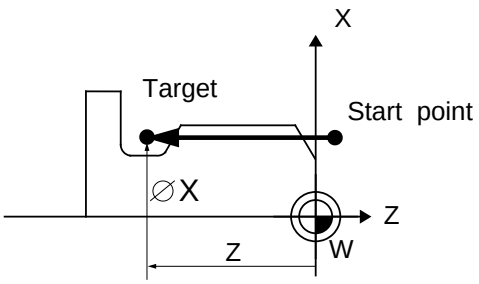
8.8 Threading (G33/G34/G35/G36/G63)

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

- **Single** threads with **constant lead** (G33, G36, G63)
- **Single** threads with **linear increasing lead** (G34)
- **Single** threads with **linear decreasing lead** (G35)
- **Multiple** threads (G33, G34, G35, G36)
- Longitudinal, taper and cross threads (G33, G34, G35, G36)
- Through-hole**/**blind-hole threads (G33, G36, G63)
- External or internal threads (G33**/**G34**/**G35**/**G36**/**G63)
- Right-hand or left-hand threads (G33**/**G34**/**G35**/**G36**/**G63)

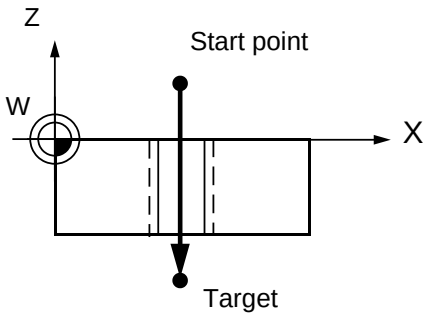
Thread length/thread depth (target**/**start point) for turning and milling

Turning



The **thread length** is entered under the corresponding path address (e.g. Z) taking into account are-in and are-out sections at which the feedrate and spindle speed are increased or reduced. The values can be entered in absolute or incremental dimensions.

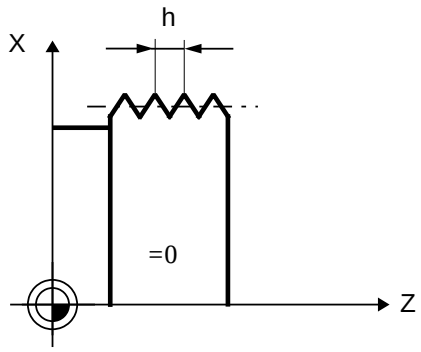
Milling



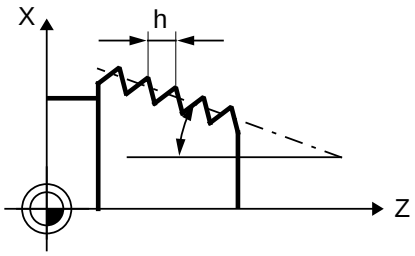
The **thread depth** is entered under the corresponding path address (e.g. Z) taking into account are-in and are-out sections at which the feedrate and spindle speed are increased or reduced. The values can be entered in absolute or incremental dimensions.

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

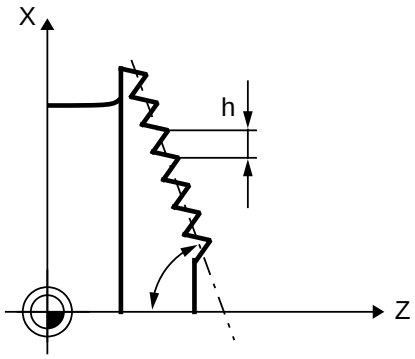
Note that the interpolation parameters I, J or K are assigned to the corresponding axis via a machine data.



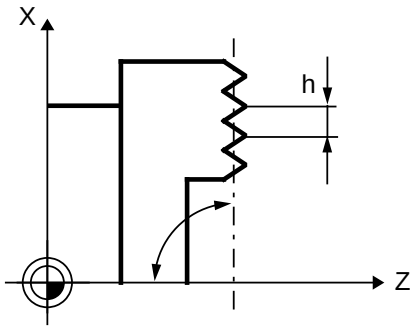
Longitudinal thread, angle = 0°.
Pitch h is programmed under parameter K.



Taper thread, angle 45°.
Pitch h is programmed under parameter K.



Taper thread, angle 45°.
Pitch h is programmed under parameter I.



Face thread, angle =90°.
Pitch h is programmed under parameter I.

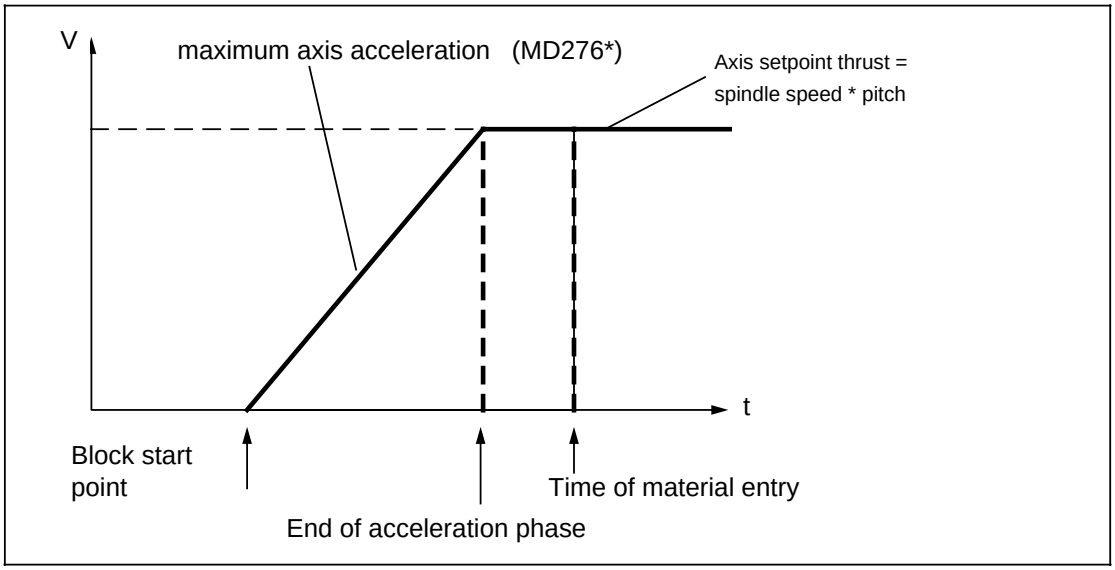
Lead for longitudinal, taper and face thread

Threading with G33 with minimum machine stress (SW 5.4 and higher)

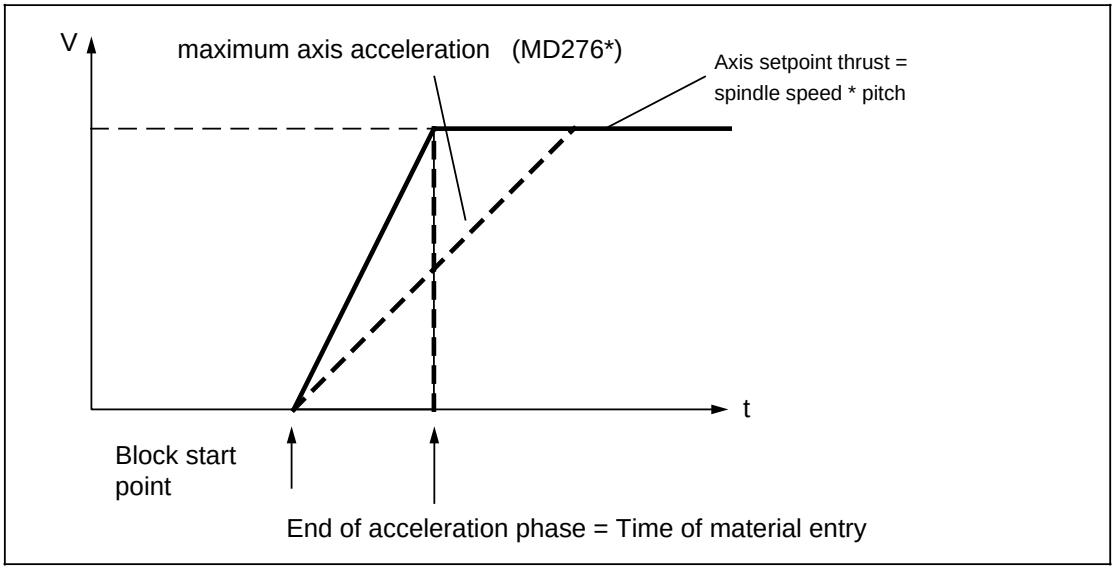
Threading with G33 with minimum machine stress is obtained by programming an additional material entering coordinate. As a result, the infeed axis is no longer accelerated abruptly but on an acceleration ramp.

Function:
The slope of the acceleration ramp is calculated automatically, the following two cases are to be distinguished:

- **Case 1:**
  The axis is accelerated according to the machine data MD276* (maximum acceleration) and reaches its velocity already before material entry!



- **Case 2:**
  The axis is accelerated faster than MD276* in order to obtain its set speed with material entry.



The following applies to both cases:
At the end of the acceleration ramp, the position coupling is set from the actual spindle value to the setpoint axis value. At that time, the axis is positioned in relation to the zero mark of the spindle so as if the axis was accelerated abruptly at the block start point depending on the zero mark. Since the axis is no longer accelerated abruptly, the start of the axis is correspondingly earlier in relation to the spindle position.

Programming:

Syntax: G33<Target point coordinates><pitch><Material entering coordinate>

The material entering coordinate is programmed via machine data under

- address letters for name of radius and chamfer (MD5000) and
- mixed programming G90/G91 in the block, absolute or incremental (MD5007.5).

In the case of taper/face threads, the material entering coordinate is allocated to the leading axis. The material entry coordinate is taken into account for all functions in which geometric machining is performed, e.g. scale factor, 2D coordinate rotation, zero offset, tool length compensation.

If the position of the material entering coordinate is outside the block path, the alarm 3263 "Inadmissible axis position" is output and machining is stopped. After changing the program, machining can be continued with NC start.

In the case of chained thread cutting blocks, the material entering coordinate is only effective in the first thread cutting block. If the material entering coordinate is programmed in the second or following thread cutting block, this is ineffective and no alarm is output.

Example (acc. to case 1):

```
N90   M3   S200
N100  G90  G00   Z143
N110  G33  Z100  K12  B140  B is the address for the material entering coordinate
N120  G90  G...
```

In block N110, a feedrate of 2400 mm/min results from the spindle speed of 200 rev/min and the pitch of 12 mm/rev. With an assumed acceleration in the Z-axis of 0.5 m/s^2, the acceleration path is 1.6 mm, at the end of which the synchronism between spindle and axis is formed. The programmed path of 3 mm up to the material entering point (Z143-B140) comprises in this example also a path of 1.4 mm which is traversed by the Z-axis at a constant set speed of 2400 mm/min. This serves for stabilizing the following error or the actual velocity of the Z-axis after the synchronization point.

Notes:

With the function "threading with G33 with minimum machine stress"

- ... the functionality "thread smoothing with G92T" is automatically switched off until the end of the thread cutting block or the end of the chained thread cutting blocks.
- ... it is possible to combine G09 (exact stop), so that braking is smooth also at the end of the block (MD276*).

8.8.1 Thread cutting with constant lead (G33)
Thread cutting with linear increasing lead (G34)
Thread cutting with linear decreasing lead (G35)
Switching over the actual value link (G431/G432)[1)]

Program syntax:

```
G33  <Coordinates of target point> <thread lead> <material entering coordinate>
<G431/G4321>
G34  <Coordinates of target point> <initial lead>  <lead change> <G431/G4321>
G35  <Coordinates of target point> <initial lead>  <lead change> <G431/G4321>
```

The G functions G33, G34, G35 und G431, G432 are modal.

The programmed feedrate F has no meaning but remains modal, i.e. the feedrate is derived internally from the actual speed value of the leadscrew.

G functions G431/G432 are used for switching between the direction-dependent and absolute value dependent actual value link.

G431 M version:

If the control is initialized for milling, the actual value coupling between the spindle speed actual value and feed setpoint is **direction dependent**. If the traversing direction of the feed axis is to be altered, then the spindle direction of rotation (M3 -->M4) must also be altered. Additionally, the overrun when the spindle is reversed (e.g. M3 -->M4) must be taken into account when programming the **thread depth**.

Note:

If, in thread tapping blocks (thread cutting) on milling machines, after the first thread block, the direction of movement but not the spindle direction of rotation is programmed in the opposite direction, traversing is still carried out in the previous direction. The software limit switches are active.

When the software limit switches are triggered, the following error must be eliminated. The software limit switch is therefore overtravelled by an amount equal to the following error.

Only the leadscrew of the channel may be programmed before the initial programming and when G33, G96 and G97 are active.

G432 T version:

If the control is initialized for turning, the actual value coupling between the spindle speed actual value and the feed setpoint is **dependent on the absolute value**. This means that the spindle can continue turning in one direction while the feed axis is reprogrammed to change direction.

With G34/G35 the thread lead is altered per turn by the value programmed under address F until the maximum or minimum possible value is obtained.

The value F must be used without a sign and is derived from the known initial and final lead and thread length:

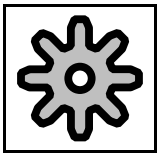
$$F = \frac{\text{Initial lead}^{\,2} - \text{Final lead}^{\,2}}{2 \cdot \text{Thread length}}$$

---

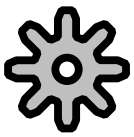
1) Software version 5 and higher

The spindle direction of rotation and speed must be programmed in one block before the actual thread cutting operation so that the spindle can run up to its set speed. "Exact stop fine" (G09) is permissible in thread cutting blocks. If G60 is active this is ignored; then G64 is active.

A right-hand or left-hand thread is specified by programming the spindle direction of rotation M3 or M4.

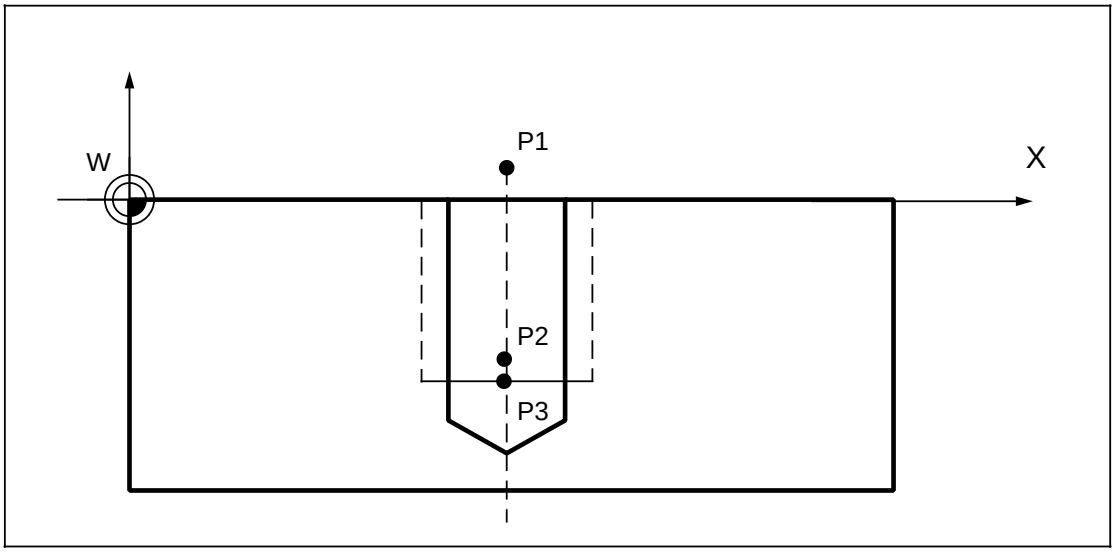


The spindle speed override switch is active if set in the machine data.



The feedrate override switch, Feed Hold and feedrate override switch setting zero are not active. NC Stop is active.

Example: Blind-hole thread with constant lead with G33

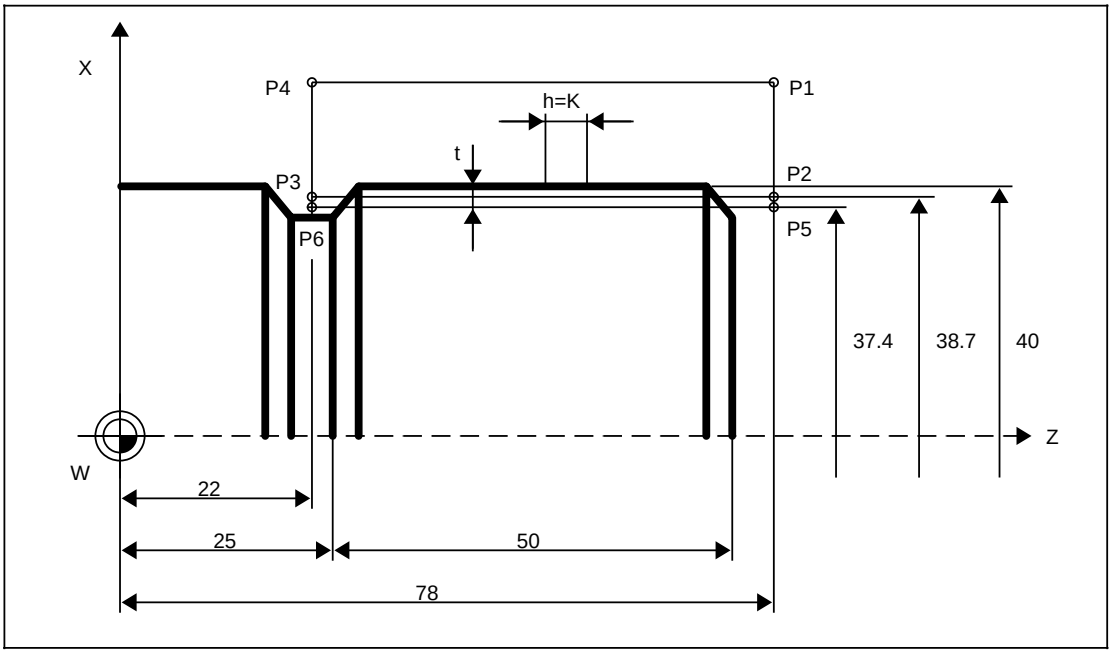


Absolute position data input:

```
N10  M3 S1000  LF      (Spindle speed in clockwise direction at 1000 rev/min)
N15  G00 Z2  LF        (Travel to reference plane P1)
N20  G33 Z-17 K1  LF   (Thread cutting to thread depth P2)
                       (Overrun produces a real thread depth at P3)
N30  M4 Z2 K1   LF     (Reverse direction of rotation of spindle and withdraw to reference
                       plane)
N35 G00 Z..
```

The overrun from P2 to P3 depends on the spindle acceleration time constant.

Example: Longitudinal thread with constant lead with G33



Lead h = 2 mm; thread depth t = 1.3 mm; radial infeed direction

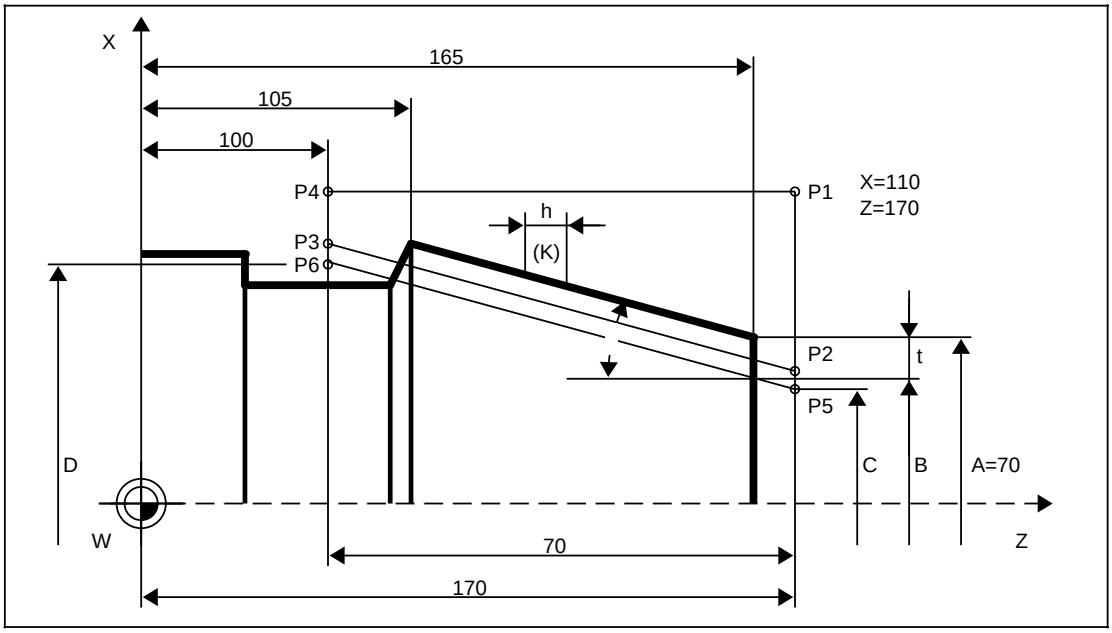
Absolute position data input:

```
N20  G90  S500  M3  LF
N25  G00  X46  Z78  LF          (P1)
N30  X38.7  LF                 (P2)
N35  G33  Z22  K2  LF          (P3)
N40  G00  X46  LF              (P4)
N45  Z78  LF                   (P1)
N50  X37.4  LF                 (P5)
N55  G33  Z22  K2  LF          (P6)
N60  G00  X46  LF              (P4)
```

Incremental position data input:

```
N20  G91  S500  M3  LF
N25  G00  X-...  Z-...  LF      (P1)
N30  X-3.65  LF                (P2)
N35  G33  Z-56  K2  LF         (P3)
N40  G00  X3.65  LF            (P4)
N45  Z56  LF                   (P1)
N50  X-0.65  LF                (P5)
N55  G33  Z-56  K2  LF         (P6)
N60  G00  X0.65  LF            (P4)
```

Example: Taper thread with constant lead with G33



Lead h = 5 mm; thread depth t = 1.73 mm; = 15 degrees; radial infeed direction

Both end point coordinates must be written. The lead h is programmed in parameter K. In the first cut (P2 ... P3), the infeed is 1 mm, in the 2nd cut (P5 ... P6), the infeed is 0.73 mm.

Calculation of the thread initial and end point coordinates

A = 70
B = A – 2t = 66.54
C = B – 2 (5·tan) = 63.86
D = C+2 (70·tan) = 101.366
K = h = 5

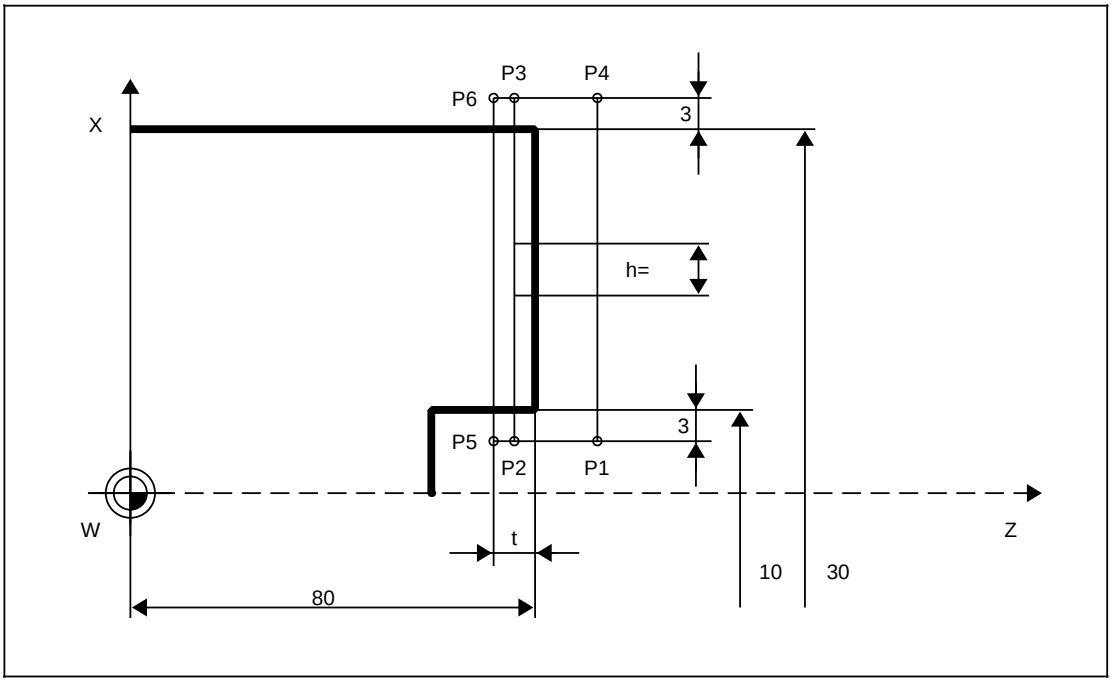
Calculation of points P2 and P3

P2 = C+2 mm = 65.86 mm in X direction
P3 = D+2 mm = 103.366 mm in X direction

Absolute position data input:

```
N30  G90  S500  M3  LF
N35  G00  X110  Z170  LF                (P1)
N40  X65.86  LF                         (P2)
N45  G33  X103.366  Z100.  K5  LF       (P3)
N50  G00  X110  LF                      (P4)
N55  Z170  LF                           (P1)
N60  X63.86  LF                         (P5)
N65  G33  X101.366  Z100  K5  LF        (P6)
N70  G00  X110  LF                      (P4)
```


Example: Face thread with constant lead with G33



Lead h = 2 mm; thread depth t = 1.3 mm; Infeed at right angles to cutting direction

Absolute position data input:

```
N40  G90  S500  M3  LF
N45  G00  X4  Z82  LF                (P1)
N50  Z79.35  LF                      (P2)
N55  G33  X36  I2  LF                (P3)
N60  G00  Z82  LF                    (P4)
N65  X4  LF                          (P1)
N70  Z78.7  LF                       (P5)
N75  G33  X36  I2  LF                (P6)
N80  G00  Z82  LF                    (P4)
N85
:
:
```

Example: Longitudinal thread with increasing lead with G34

```
N25  G34  G90  Z217.  K2.  F0.1  LF
```

K2 Initial lead 2 mm
F0.1 Lead change+ 0.1 mm per turn,
i.e. after 5 turns the thread lead is 2.5 mm

Example: Longitudinal thread with decreasing lead with G35

```
N45  G35  G90  Z417.  K10.  F0.5  LF
```

K10 Initial lead 10 mm
F0.5 Lead change 0.5 mm per turn, i.e. after 10 turns the lead is 5 mm.

8.8.2 Thread cutting position-controlled spindle (rigid tapping) with constant lead (G36)

The "Thread cutting position-controlled spindle" function is an option.

Program syntax:

```
G36 <Name of rotary axis> <Coordinate(s) of target point> <thread lead>
```

G36 is modal.

If the function (G36) is selected the control switches to linear interpolation between the spindle as a rotary axis and the infeed axis.

The function will only run if the spindle is in rotary axis mode.

Both blind-hole threads and through-hole threads can be tapped. There is no overrun, the programmed thread depth is met exactly.
It is also possible to cut longitudinal**/**face threads (spindle as rotary axis and one linear axis) and taper threads (spindle as rotary axis and two linear axes).

A linear increasing**/**decreasing lead **cannot** be cut with G36.

The sign before the thread lead shows the direction of rotation.

G98 is automatically triggered with G36 (leading feedrate for a rotary axis in rev**/**min), i.e. the active feedrate in the G36 block is interpreted as rev**/**min. The path feed is calculated internally from this leading feedrate based on the path axes used.

If G36 is deselected, the previous G command automatically becomes active (G94, G95, G98). The feedrate must be reprogrammed.

The travel behaviour at block transitions can be influenced by G09, G60, G62 and G64.

Example 1: Blind-hole thread:

```
N10  M70                          (Switch spindle to rotary axis mode, e.g. M70)
N20  G0  C30  Z2  G90             (Position rotary axis to 30 degrees, position Z = 2 mm;
                                  corresponds to thread insertion point)
N30  G36  C  Z-30  K1  F1000      (Thread lead 1 mm/revolution, clockwise;
                                  feedrate of C axis 1000 rev/min)
N40  C  Z2  K-1                   (Thread lead 1 mm/revolution, counter-clockwise,
                                  feedrate of C axis 1000 rev/min)
.
.
```

Example 2: Taper thread:

```
N10  M70                          (Switch spindle to rotary axis mode, e.g. M70)
N20  G0  C30  X10  Z2  G90        (Position rotary axis to 30 degrees, position Z = 2 mm,
                                  X = 10 mm; corresponds to thread insertion point)
N30  G36  C  Z-30  X20  K5  F1000 (Thread lead 5 mm/revolution, clockwise; feedrate of C
                                  axis 1000 rev/mm)
.
```

Explanation of examples:

In block N30 the rotary axis assigned to the spindle must be written with G36. A value written behind C is ignored.

To simplify programming, the end point of the rotary axis is not programmed. The control derives the final point from the programmed lead and the path of the linear axis.

The thread lead is programmed under address K (I, J). The sign in front of K (I, J) shows the direction of rotation. Right-handed and left-handed is defined in this Programming Guide in the section ”Coordinate systems”.

The address for the thread lead (for taper thread) must match the axis with the longest path (example 2: path X = 10 mm, path Z = 32 mm thread lead K belongs to axis Z), otherwise alarm 3006 ”Wrong block structure” appears.

Thread lead 0, or a programmed address where axis and thread lead do not harmonize, causes alarm 3006 ”Wrong block structure” to appear.

At least 2 axes (longitudinal thread/face thread) and 3 axes (taper thread) of which one is a rotary axis, must be programmed, otherwise alarm 3006 ”Wrong block structure” appears.

G98 is triggered internally with G36, i.e. the programmed feedrate F refers only to the rotary axis in **rev/min**.

The resulting path feedrate (C axis and linear axis) is derived internally from the programmed lead.

The rotary axis (in this case C axis) must be written in every block with G36 active (N30/N40).

8.8.3 Tapping with compensating chuck (tapping without encoder) with constant lead (G63)

G63 is modal.

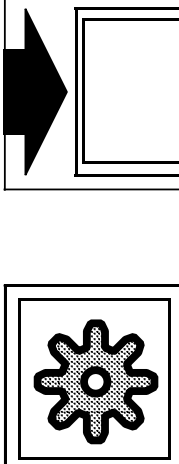
The function G63 is used for tapping threads with a tap in the compensating chuck.

There is **no** coupling between the spindle speed actual value and feedrate setpoint.

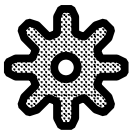
The desired thread lead is derived from the relationship between the programmed spindle speed and the programmed feedrate F of the infeed axis.

The compensating chuck must accommodate the tolerances between feedrate and speed as well as spindle run-out after the position is reached.

Function G63 can only be programmed in blocks with linear interpolation G01.



The spindle speed override switch is active if so defined in machine data.



The feedrate override switch, Feed Hold, single block and channel-specific feedrate override switch setting zero are not active. NC Stop and axis-specific feed disable are active.

8.8.4 Multiple thread cutting using start angle offset (G92 A..)

Program syntax:

```
G92  <Initial angle offset  A..>
```

The function must stand alone in a block.

G92 is non-modal. The start angle offset is stored via G92 A.. in a channel-specific setting data and remains active until G92 A0 is programmed.

The maximum input value of the start angle offset is 999.99999 without a sign. The input resolution is 10^{-5} degrees. The start angle offset is always calculated modulo 360 degrees.

The thread commencement point can be determined using the start angle offset. The thread can then be cut with G33, G34, G35 or G36. A multiple-start thread is obtained by repeating the start angle offset several times and selecting the above thread cutting functions.

Example: 3 start thread each offset by 120 degrees

```
N10  G01  G90  X50  Z5  S200  M3  LF  (Starting point)
N15  G92 A0 LF                       (1st thread start, 0 degrees)
N20  G33 Z-50 K1   LF                (Threading)
.
.
N100 X50 Z5 LF                       (Starting point)
N105 G92 A120 LF                     (2nd thread start, 120 degrees)
N110 G33 Z-50 K1   LF                (Threading)
.
.
N200 X50 Z5 LF                       (Starting point)
N205 G92 A240 LF                     (3rd thread start, 240 degrees)
N210 G33 Z-50 K1   LF                (Threading)
```

8.8.5 Multiple thread cutting using start point offset

One start of a multiple-start thread can be programmed in the same way as a single-start thread. After the first start has been completely machined the starting point is offset by $delta\ l$ (starting point offset).

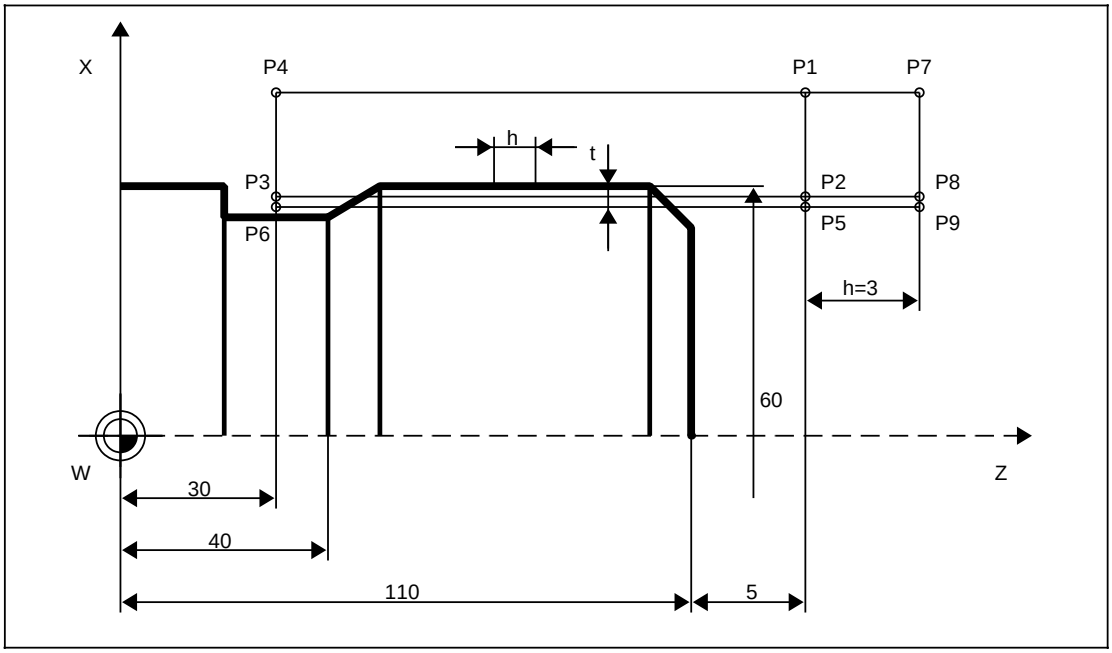
$$delta\ l = \frac{Thread\ pitch}{Number\ of\ starts}$$

Then the next start can be machined.

The individual starts must be machined using the same spindle speed to avoid having different following errors.

Example:

Multiple-start longitudinal thread with constant pitch with G33



Pitch h = 6 mm; thread depth t = 3.9 mm; radial infeed direction; two starts

In the example each start is machined in two steps. After the first start has been completely machined, the second start is machined by shifting the starting point by h'.

$$h' = \frac{\textit{Thread pitch}}{\textit{Number of starts}} = \frac{6}{2} = 3\ mm$$

Absolute position data input:

```
N35  G90  S500 M3  LF
N40  G00  X66  Z115  LF          (P1)
N45  X56  LF                     (P2)
N50  G33  Z30  K6  LF            (P3)
N55  G00  X66  LF                (P4)
N60  Z115  LF                    (P1)
N65  X52.2  LF                   (P5)
N70  G33  Z30  K6  LF            (P6)
N75  G00  X66  LF                (P4)
N80  Z118  LF                    (P7)
N85  X56  LF                     (P8)
N90  G33  Z30  K6  LF            (P3)
N95  G00  X66  LF                (P4)
N100  Z118  LF                   (P7)
N105  X52.2  LF                  (P9)
N110  G33  Z30  K6  LF           (P6)
N115  G00  X66  LF               (P4)
```

1st start

2nd start

8.8.6 Smoothing and feed ramp-up time with thread cutting (G92 T . .)

This function is used to protect the drive when cutting threads and to obtain a better speed stability. The function is only active with G33, G34 or G35.

G92 T.. must stand alone in a block.

G92 is non-modal. The function G92 T.. remains active until G92 T0 is programmed.

The feed ramp-up time until synchronization with the already running working spindle is reached can be programmed using the command G92 T... The feed ramp-up time should be matched to the run-in path available:

- short run-in path, short ramp-up time, low T value
- long run-in path, long ramp-up time, high T value

The programmed ramp-up time also acts as the smoothing time, i.e. the actual spindle speed is averaged via this value. A more uniform feedrate is thus obtained.

The smoothing and feed ramp-up time depends on

- the programmed value T..
- the interpolation cycle which is set in the machine data. This machine data should only be changed by a properly trained specialist.

The value T.. must be an integer. Possible values are 0 ... 5.

The following table shows which T value has to be programmed for a set interpolation cycle in order to obtain the required feed ramp-up time:

Programmed value T..	0	1	2	3	4	5
Multiplier	1	2	4	8	16	32
Ramp-up time to thread feedrate (ms) for an interpolation cycle of 16 ms	16	32	64	128	256	512
Ramp-up time to thread feedrate (ms) for an interpolation cycle of 8 ms	8	16	24	64	128	256

Calculation of the ramp-up time for any given interpolation cycle

For each T value there exists a certain multiplier which can be taken from the table. For example, the value T1 corresponds to the multiplier 2, the value T2 to the mulitplier 4 etc. The ramp-up time is calculated as follows:

$$\textit{Ramp-up time}\ (ms) = \textit{Multiplier} \cdot \textit{Interpolation cycle}\ (ms)$$

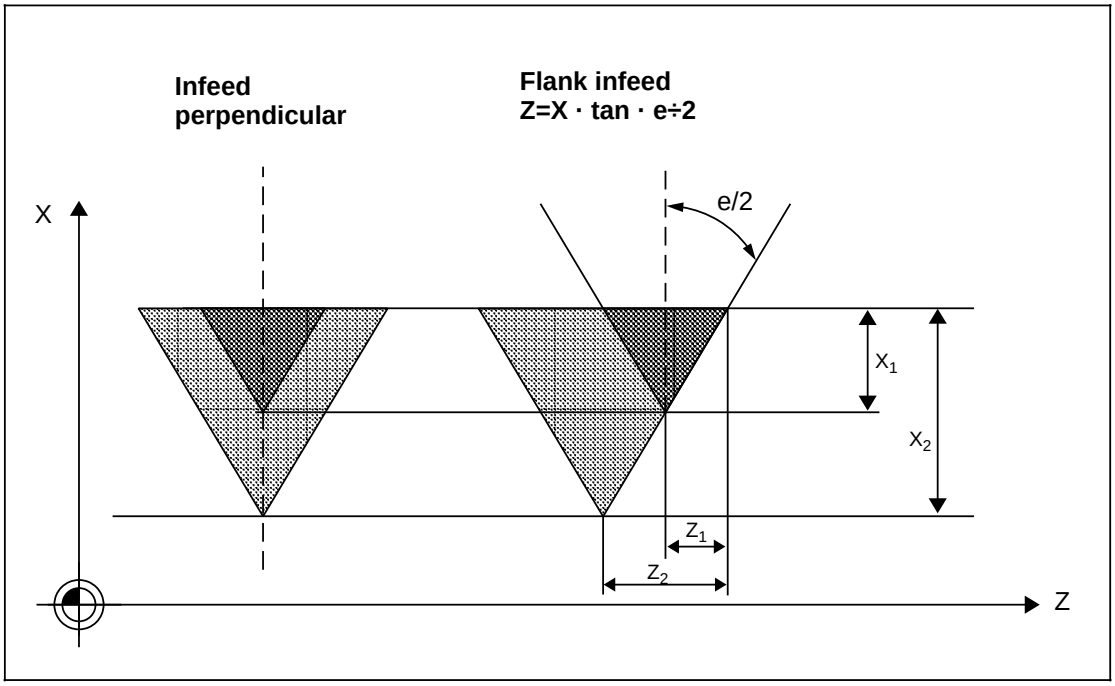
Example:

An interpolation cycle of 8 ms has been set. G92 T3 has been programmed in the NC program.

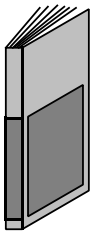
The table shows that the multiplier has the value 8 (with T3). The ramp-up time is derived from the product 8·8 ms = 64 ms.

8.8.7 Infeed options

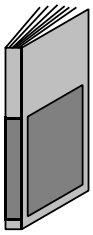
The tool can be advanced perpendicular to the cutting direction or along the flank. Flank infeed is used to load the tool optimally because cutting takes place on one side only and therefore the forces act on just this side.



Infeed perpendicular and flank infeed



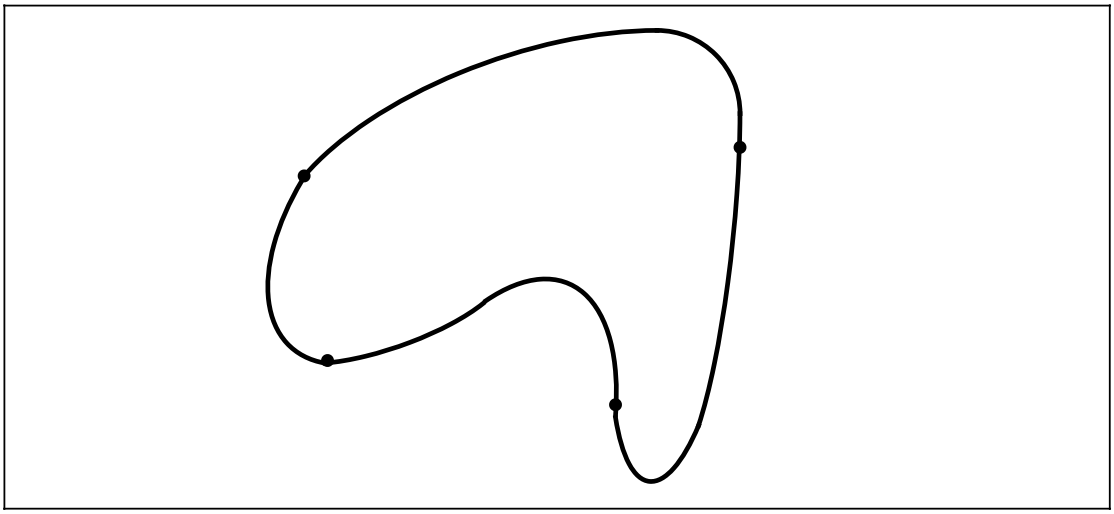
Standard cycle L97 exists for both infeed types.
This is described in the documentation ”SINUMERIK 840C, Cycles, Programming Guide”.



8.9 SPLINE interpolation (G06)

The “SPLINE interpolation” function is an Option.

A spline is a concatenation of curves which have the same function values, identical slope and identical curvature at their connecting points.



SPLINE interpolation G06

SPLINE interpolation reduces the amount of programming required for machining complex contours and permits shapes to be machined which cannot be described with standard geometries.

For programming of SPLINE interpolation, please refer to the “SPLINE Interpolation” Programming Guide.

G06 is **modal** and is deselected by another preparatory function of the 1st G group. SPLINE programs are executed with input resolution 10^{-3} mm only.

The coordinate blocks of the spline blocks can only be programmed with block numbers. A block search with calculation can therefore only be addressed at coordinate blocks and not coefficient blocks.

In all refresh cases (mode change, NC stop, NC start, reset and block search with calculation), the first processed spline block is converted into a linear block.

8.10 Simultaneous axes

8.10.1 Simultaneous axis motion with path and feedrate information

Every simultaneous axis requires positional and feedrate data. The motion is in accordance with G01.

An axis is declared non-modally as a simultaneous axis. Every simultaneous axis traverses at its axis-specific feedrate.

The traversing movement of a simultaneous axis bears no relation to the traversing movement of the contouring axes or other simultaneous axes.

The infeed motion is executed independently of interpolation grouping.

Interpolation types (G00, G01, G02, G03, ...) do not act on simultaneous axes.

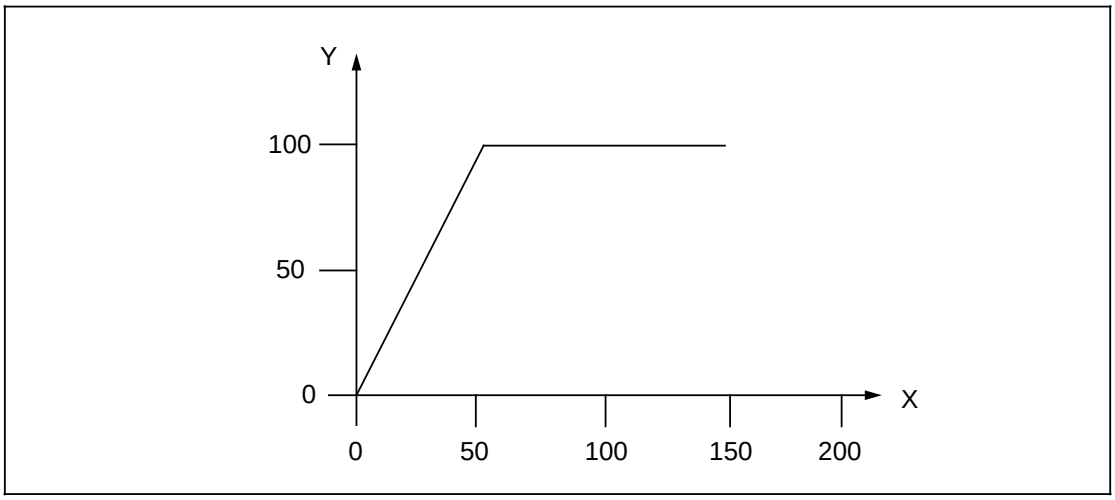
Program syntax:

```
P[axis name] {axis value} F[axis name] {feedrate}
```

Example:

```
P[X]150 P[Y]100 F[X]200 F[Y]400
```

Axes X and Y are declared as simultaneous axes **with** end-of-block behaviour. F[X] and F[Y] is the axis feedrate assigned to simultaneous axes X and Y.



If 2 simultaneous axes with different paths and feedrates are programmed and started at the same time, they will reach the end point at different times

The axis name and the feedrate can be programmed in plaintext, as constant, as R parameter or as pointer.

Example:

```
P[X]100 P[K1]1000 F[R10]200 F[P5]1000
```


Notes:

- Coordinate rotation and coordinate transformation cannot be used with simultaneous axes. Only real axes can be programmed as simultaneous axes.
- The following can be declared a simultaneous axis
  – a leading axis in coupled motion
  – a leading or following axis in gear interpolation
- If simultaneous axes are programmed in blocks in which tool radius compensation is active for the contouring axes, the simultaneous movement starts in the programmed path section of the contouring axes and not in inserted path sections (e.g.: transition circle, inserted intermediate blocks).
- Extended axis programming is possible e.g.: **`N10 P[Y2]=200 F[Y2]=1000`**
- If no axis value is programmed, e.g. **`P[Y]`**, this is the same as **`P[Y]=0`**
- If no feedrate is programmed, e.g. **`F[Y]`**, this the same as **`F[Y]=0`**

Example 1: Simultaneous axes

```
%10
N010 G01 X100 Y100 Z100 F1500          3D interpolation with path feedrate 1500 mm/min
N020 P[X]50 F[X]500                    Position the axis X with F = 500 mm/min
N030 P[X]0                             Axis feedrate of X is modal
N040 G02 X100 I0 Y0 J0 P[Z]10 F[Z]150  Circular interpolation with X and Y, path feedrate
                                       F= 1500 mm/min, position with Z and axis feed-
                                       rate F = 150 mm/min.
N050 P[Z]200
N060 M30
```

Example 2: Simultaneous axes

```
%20
N010 F[X]200 F[Y]200 F[Z]380 F[A]80 F[B2]=500   Simultaneous axis feedrates for all the si-
                                                multaneous axes to be traversed can be
                                                programmed in one block.
N020 G01 X100 Y200 Z300 F1000 P[A]50            3D interpolation for X, Y and Z and
                                                positioning with A.
N030 X200 P[Z]0                                 X axis travels at path feedrate and the Z
                                                axis at simultaneous-axis feedrate.
N040  P[Z]50 F[Z]100                            Z travels at the new axis feedrate.
N050 M30
```

Example 3: Coupled motion of simultaneous axes

```
%30
N010 G151 P[Y]400 F[Y]120  0                    Y is the leading axis in coupled motion; de-
                                                pending on the coupled-motion configuration,
                                                the axes in question can be defined as
                                                coupled-motion axes.
N020 P[Y]600 G01 A300 F400 P[B]45 F[B]760       In addition to the coupled-motion group of Y,
                                                additional contouring axes (A) or simultane-
                                                ous axis (B) can be traversed as long as they
                                                do not collide with the coupled-axis grouping
                                                of Y.
N030 G150                                       Deselect the coupled-motion function
N040 M30
```

8.10.2 Handwheel overlay in Automatic mode (G[...]27)

With this function it is possible to traverse a simultaneous axis with the handwheel in Automatic mode.

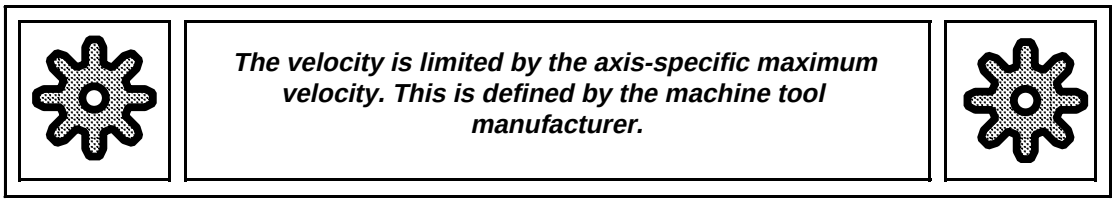
The handwheel overlay is enabled by programming G[...]27. This G command is **axis-specific** and non-modal.

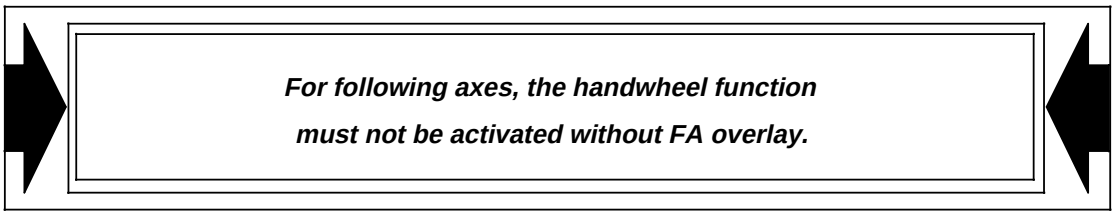
Either a velocity- or a position-dependent overlay can be programmed for the simultaneous axis.

Two handwheels can be activated in a block at the same time. Two handwheel overlays are possible in one block. The same handwheel can be assigned to both axes.

Notes:

- If a simultaneous axis is declared as a transverse axis, any radius*/*diameter settings in the program will apply here equally with respect to the handwheel pulses.
- All other limitations remain active (e.g. working area limitation)
- Only simultaneous axes can be programmed, otherwise alarm 3000, "General programming error" appears.
- If G27 and DRF are active at the same time, alarm 3007, "Illegal handwheel function" appears.
- If an axis is programmed with G27 even though no handwheel is connected, alarm 3000, "General programming error" appears.





Velocity overlay

A simultaneous axis is programmed together with its axis coordinate after a G27 command. The feedrate for this simultaneous axis can be programmed in the same block or the one before. The programmed position is reached at the velocity chosen with the handwheel overlay, i.e. the velocity is increased or decreased manually depending on the direction in which the handwheel is turned. The direction of the axis, however, stays the same. The axis can be decelerated down to no more than v = 0. It is not possible to move the axis in the opposite direction (negative velocities).

Program syntax: `G[axis name]27 P[axis name] axis value F[axis name]feedrate`

Example 1:

`N10 G[X]27 P[X]100 F[X]200`	Incremental processing with simultaneous axis X as axis overlaid by the handwheel

Example 2:

```
N10 G01 X100 Y200 F1000 F[Z]50
```

`N20 X200 Y300 G[Z]27 P[Z]50 G[A]27 P[A]100 F[A]60`	Axes Z and A are assigned a handwheel each or the same handwheel (depends on PLC)

The axis name and the feedrate can be programmed in plaintext, as constant, as R parameter or as pointer.

Example:

```
G[X]27 G[K1]27 G[R10]27 G[P5]27
```

Notes:

- The feedrate overlay does not affect the velocity components originating from the handwheel. It only affects the programmed feedrate (velocity overlay).
- If the override switch position is 0% or the feedrate disable is active, the axis cannot be moved by means of the handwheel. Block change, Reset or EMERGENCY STOP end the programmed handwheel overlay.

Path overlay

The programmed position can only be reached manually. The direction in which the wheel is turned determines the direction of travel of the programmed simultaneous axis. It is possible to traverse the axis in the opposite direction to the programmed direction but it is not possible to travel beyond the programmed end position. Once the programmed end position has been reached, block change occurs and the axis can only be traversed again by the handwheel if G[...]27 is reprogrammed. It is possible to abort the active block by deleting the axial distance to go (@736)

Program syntax: `G[axis name]27 P[axis name] {axis name} F[axis name] {feedrate value}=0`

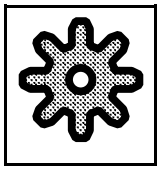
Example:

`N10 G[X]27 P[X]100 F[X]0`	Incremental processing with X as the axis overridden by the handwheel

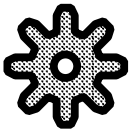
Notes:

- When dry run is active, "Feedrate stop" is active and the axis traverses at the dry run feedrate.
- With path overlay and fast handwheel movements the assigned axis lags behind the actual handwheel position. As long as the programmed end position has not been reached, no handwheel pulses will be lost. Pulses can, however, be lost in the following cases:
  - Override 0%
  - No handwheel enable from the PLC
  - No feed enable from the PLC
  - Velocity overlay

8.10.3 Endlessly rotating rotary axis (G103/G104/G105/G119)



Machine data can be set to determine whether the spindle is switched to active mode after power on (speed-controlled) or switched over directly after power-up into C-axis mode (position-controlled).



This function is programmed via axis-specific, modal G commands. It can be controlled from several channels.

G[axis name]103	Endlessly rotating rotary axis ON clockwise
G[axis name]104	Endlessly rotating rotary axis ON counter-clockwise
G[axis name]105	Endlessly rotating rotary axis OFF
G[axis name]119 <P[axis name]>	Endlessly rotating rotary axis OFF with oriented stop

The G commands act modally and may only be programmed in connection with a rotary axis.

The endlessly rotating rotary axis can be programmed as a rotary or simultaneous axis. A block may contain a maximum of 5 programmed simultaneous axes. For programming of endlessly rotating rotary axes **with positioning**, only **one axis** is allowed in the block.The axis name can be programmed in plaintext or as a constant, R parameter or pointer:

Example: G[C]103, G[K1]103, G[R10]103, G[P5]103

G[...]119 may only be programmed if the axis rotates endlessly, i.e. G[...]103 or G[...]104 must be active.

G commands G[...]103, G[...]104, G[...]105 and G[...]119 may be programmed in any sequence within a part program for an endlessly rotating axis.

The stop position of the endlessly rotating rotary axis must be programmed directly after G[...]119.

Example: `G[C]119 P[C]100`

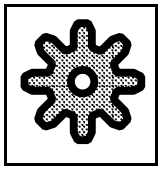
A new feedrate mode G[...]94, G[...]195, G[...]295, G[...]98 and a new feedrate value F[...] may be specified at any time for an endlessly rotating rotary axis. A new feedrate value must be programmed for every new feedrate mode.

As long as the axis is rotating endlessly, it may not be programmed as a path axis or simultaneous axis with block end response. The endless rotating operation must be terminated first by means of G[...]105 or G[...]119.

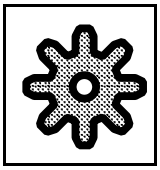
Before the axis can be traversed again as a path axis or simultaneous axis with block end response, the endless rotating operation must be terminated with G[...]105 or G[...]119 and an axis actual-value synchronization process initiated.

Failure to synchronize the axis actual value will result in incorrect positioning.

Following G200 Axis, the axis which previously operated in endless rotating mode can be positioned directly with G90/G91 characteristics. G91 can be programmed subsequently only if no coordinate rotation or scale factor is active for the axis.



Machine data can be set to determine whether the block end value is automatically refreshed (G200) when endless rotating mode is terminated by means of G[...]105 or G[...]119.



Examples:

`G[C]103 G[C]98 F[C]1000`	Endlessly rotating rotary axis ON, clockwise rotation, 1000 rev**/**min
`G[C]105`	Endlessly rotating rotary axis OFF
`G90 C50 F200`	C axis is used as path axis and traversed to 50 degrees
`G90 P[C]60 G[C]94 F[C]200`	C axis is traversed as simultaneous axis to 60 degrees at 200 degrees**/**min.

G[...]94 or G[...]98 can be defined as the general reset position for simultaneous axes. [1)] Simultaneous axes with block end response can be programmed only with G[...]94.

Endlessly rotating rotary axis in several channels

The "Endlessly rotating rotary axis function can be controlled from various channels. The control sequence is determined by the instant in time at which the appropriate G command occurs.

Channel 1	Channel 2	Description
`G[C]103 G[C]98 F[C]100`	`...`	Endlessly rotating rotary axis ON, clockwise rotation, 100 rev**/**min
`...`	`G[C]104 G[C]94 F[C]500`	Endlessly rotating rotary axis ON, counter-clockwise rotation, 500 degrees**/**min
`G[C]103`	`...`	Clockwise rotating axis at speed set for channel 2: 500 degrees**/**min
`G[C]104 F[C]20`	`...`	**Feedrate mode G98 continues to apply in this channel!** Axis, counter-clockwise, 20 rev**/**min

`G[C]105`
`G200 C`
`P[C]100 G[C]94 F[C]100`
	`...`	Endless rotation OFF;
		Axis synchronization;
		Axis must be positioned at 100 degrees;
		An error message will be output if the feedrate mode is not changed.

1) *Software version 4 and higher*

8.11 Milling of cylindrical paths (G92)

The "Milling of cylindrical paths" function is an option.

Cylindrical interpolation permits machining of cylindrical paths with one rotary axis and one linear axis on a constant rotary table diameter. Both linear and circular contours can be programmed. Circular contours are possible only with radius programming. Input of the interpolation parameters I, J and K is not possible.

The position of the rotary axis is entered in degrees. Conversion to the circumferential dimensions of the working diameter is performed internally in the control. The relationship

$$P = \frac{\text{working diameter}}{\text{unit diameter}} \text{ is programmed under G92 P.. .}$$

Internally, the working diameter is related to the unit diameter.

If the "Milling of cylindrical paths" function is active, the rotary axis must not be programmed with G68 (G90, G91 only).

Input system	**Unit diameter**
• mm	114.59156 mm
• inches (10^{-4})	114.5916 inches
• inches (10^{-5})	114.59156 inches

The unit diameter is derived from the equation ·d = 360.

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

No characters other than the G functions, the factor for unit circle of the rotary axis, the axis name, the block number and the end-of-block character must be written in a block containing G92 P.. .

N.. G92 P.. C L_F

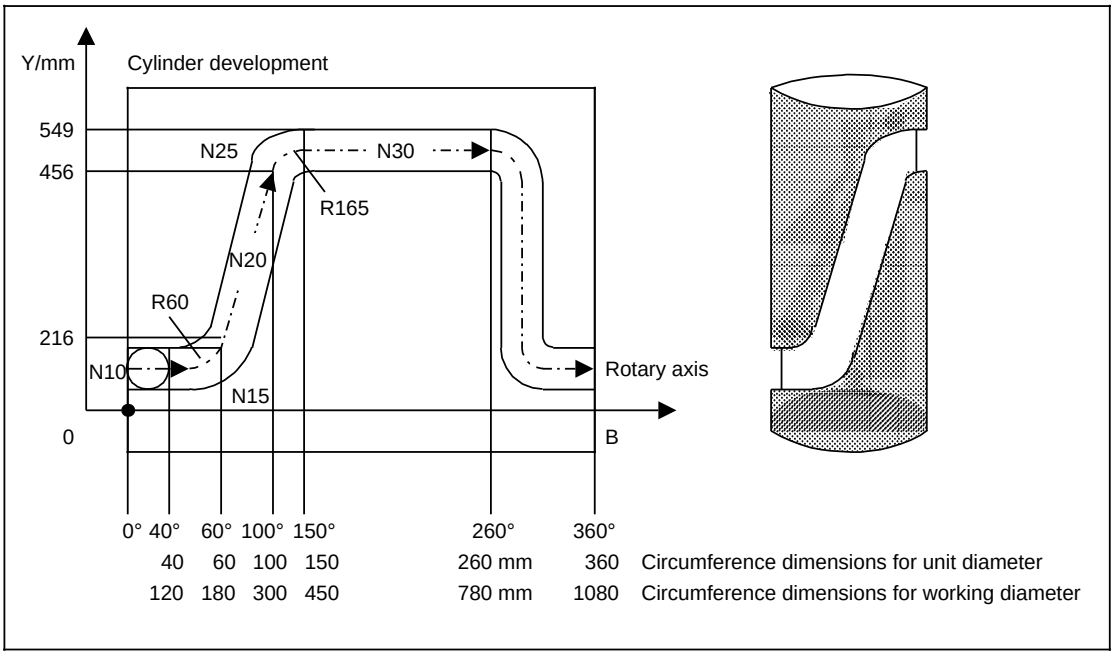
G92	Selection of cylindrical interpolation
P..	Factor for unit circle, working diameter ÷ unit diameter
C	Rotary axis

The input resolution for P is 10^{-5}. The value range for P is 0.00100 to 99.99999. The working diameter is modal until reprogrammed or reset with M02/M30 or RESET. The programmed feedrate is adhered to on the centre point path. As long as the working diameter is not equal to the unit diameter, this axis (e.g. C) can only be interpolated with one further axis, i.e. for interpolation with more than 2 axes, the working diameter must be set equal to the unit diameter.

Note:

For P < 1 note that the interpolation resolution of the rotary axis with active cylindrical interpolation is 1/p times greater than without cylindrical interpolation.

Example with B and Y axes:



Cylindrical interpolation

```
N05  G92  P3  B  LF                                  (Selection of cylindrical interpolation)
N10  G01  B40  Y200  F...  S...  M...  LF
N15  G03  B60  Y216  U+60  LF
N20  G01  B100  Y456  LF
N25  G02  B150  Y549  U+165  LF
N30  G01  B260  LF
:
N55  G92  P1  B  LF                                  (Deselection of cylindrical interpolation)
```

Note:

An interpolation up to 4 axis is possible from SW 6 onwards. The function is activated via MD 5052 bit 1.

A cylindrical interpolation may be programmed with up to 4 axes, the cylindrical interpolation, however, remains active.

Marginal condition:

If more than 4 axes are programmed in one block, reset alarm 2059 is triggered.

8.12 Gearbox interpolation

8.12.1 Gearbox interpolation types

The gearbox interpolation type functions are options

			Curve gearbox interpolation [1]	**Gearbox interpolation**
		Gearbox interpolation	Link factor also via table	
	Synchronous spindle	1 following axis or spindle		
Gantry axes	• Easy to program • 5 master drives • Link factor numerator to denominator			
1 leading/following axis link factor 1:1	1 following spindle			
This is a gearbox interpolation function with a fixed transmission ratio of 1:1	Angular precision synchronous operation of leading and following spindle	Kinematic link of axes with variable transmission ratios. • Angular precision synchronism • Transmission ratio changed during operation • Activation and deactivation of link in defined position • Max. 5 leading axes per following axis • Position-related linking of gearbox interpolation groups	As for gearbox interpolation. In addition, a variable link ratio in the form of a table can be stored and a curve gearbox simulated. Linking of axes **and spindles.**	**Brief description of function**
Gantries	The workpiece no longer has to be rechucked on lathes reduces downtimes	Universal applications in gear cutting technology	Machining of unround gears. Simulation of mechanical cam gears	**Application**

1) The option "interpolation and compensation with tables stage 2" is a prerequisite for curve gearbox interpolation. The function "curve gearbox interpolation" also includes the option "gearbox interpolation".

Interpolation and compensation with tables (SW 3 and higher) [2]	Interpolation and compensation with tables stage 2 (software version 4 and higher) [1] [2]	**Interpolation type**
32 independent error curves	Interpolation with any geometry and velocity profiles. Linear or cubic interpolation of intermediate points. Linking of different tables.	**Brief description of function**
Compensation of physical influences and machining tolerances	Cam machining	**Application**

1) The option "interpolation and compensation with tables stage 2" is a prerequisite for curve gearbox interpolation. The function "curve gearbox interpolation" also includes the option "gearbox interpolation".

2) Only available with the standard version, not in the export software.

8.12.2 Gear interpolation/curve gearbox interpolation

The "Gear interpolation and curve gearbox interpolation" function is an option.

Gear interpolation (GI) simulates a mechanical gear coupling (wheel gear and differential gear) with software functions and individual axis drives.

Curve gearbox interpolation (CGI) is a combination of gearbox interpolation with interpolation and compensation from tables (IKA) and allows simultaneous use of linear and non-linear links or curve links. The same tables are used for CGI as for IKA.

The advantage of this function over a mechanical gear transmission is that the speed ratio can be specified and driver and driven gear can be defined freely.

Gear interpolation achieves a highly accurate kinematic link between several axes and spindles which are to be moved in a defined positional and/or speed ratio to each other.

A gear interpolation grouping consists of at least one leading drive and one following drive. Depending on the type of link set, the setpoints of the following drive are either calculated from the setpoints or from the actual values of the leading drives.

When programming a gear interpolation function, the next block is not read in until the gear interpolation command of the previous block has been executed.

The behaviour of the gear interpolation function on block change can be influenced by machine data.

The G functions for gear interpolation can be programmed in one block together with any other legal NC functions, as long as the gear interpolation parameters are programmed contiguously at the end of the NC block.

The G functions and the gear interpolation functionality are modal.

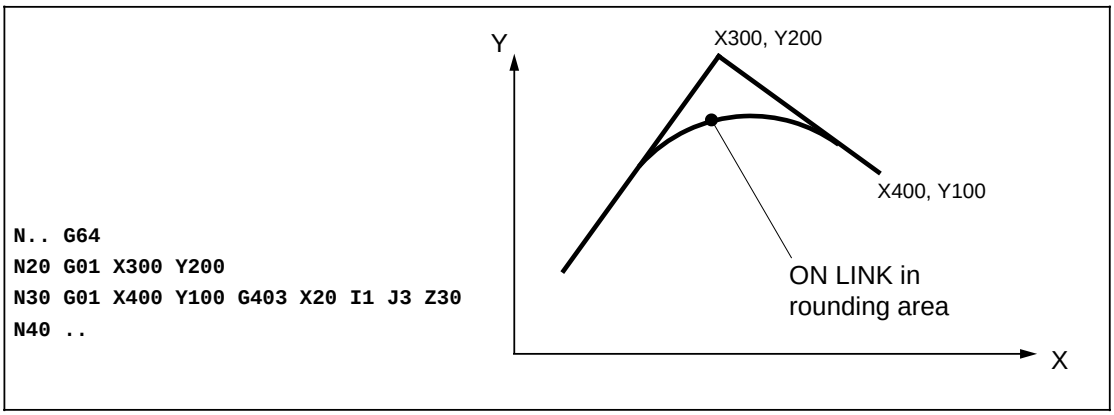
G90/G91 can also stand within a gear interpolation command.

Example:

```
N10  G91  G01  Z500  F200  G403  G90  X200  I1  J3  Y100  LF
```


The gear interpolation function acts within one mode group and for all channels (axis/spindle specific). The movement of the leading drives can be initiated from different NC channels of the mode group.

If a link is to be switched on while continuous-path mode is active, the gear interpolation selection command LINK_ON must be programmed together with the travel command in one block.

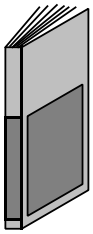


If an axis has been traversed as a following axis in a gear interpolation grouping, it must first be given the current actual position by programming G200 (axis actual-value synchronization) after the link has been switched off. Only then can the axis approach a programmed absolute position correctly.

Example: G200 FA L_F *FA=Following axis*

Gear train

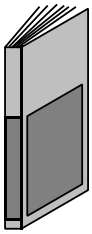
A gear train must be programmed in the order in which the setpoints are later to be generated. First the gear interpolation grouping which stands at the beginning of the train must be configured (G401) and then the gear interpolation grouping of which the leading drive is the following drive of the first grouping etc.



An additional general explanation is to be found

in the documentation

Installation Instructions, Section "Description of Functions".



8.12.2.1 Define/delete GI/CGI configuration (G401)

Definition a configuration involves producing a gear interpolation grouping of up to 5 leading drives and specifying the type of link for each leading axis and the following drive. Configuration of the GI**/**CGI link structure by means of G401 does not switch the link branches on, over or off, nor does it generate any setpoints for the following address.

The link structure may consist of up to 5 link branches which can drive a following address. Each leading address has a linear action on the following address via link types K1 to K4 (GI link branch) and a non-linear action via K11 to K12 (CGI link branch).

GI/CGI link types

GI link branch (linear)	FA position input is linked to	Output of link branch goes to	Link type
Set position link and possible actual position link via compensatory controller	Set position leading axis**/**leading spindle	Set position input of following axis**/**following spindle	K1
Actual position link	Actual position leading axis**/**leading spindle	Set position input of following axis**/**following spindle	K2
Set position link "simulated" leading axis**/**leading spindle	Setpoint position leading axis**/**leading spindle	Set position input of following axis**/**following spindle	K3
Actual position and set speed link	Actual position and set speed leading axis**/**leading spindle	Set position input of following axis**/**following spindle	K4 1)

CGI link branch (non-linear)	CGI input from	Output of link branch goes to	Link type
Set link	Set position leading axis**/**leading spindle	Set position input of following axis**/**following spindle	K11 1)
Actual-value link	Actual position leading axis**/**leading spindle	Set position input of following axis**/**following spindle	K12 1)

The programmed GI**/**CGI link structure remains in existence until it is deselected by means of G401.

1) Software version 4 and higher

Define GI/CGI configuration

Program syntax:

```
G401  <leading address 1> <link type for leading address 1> ...
      <leading address 5> <link type for leading address 5> <following address>
```

Leading address Up to 5($_n$) leading axis names**/**axis addresses**/**spindle names can be specified as the leading address.

- <leading axis**/**spindle name>
  =X1=S1 Direct specification of leading axis**/**spindle name X1**/**S1.
- <leading axis**/**spindle address>
  =@441**/**3 <value> By programming command @441**/**3, it is possible to specify the leading axis**/**spindle indirectly (for use with cycles).
  <value> Corresponds to the global axis/spindle number.

Link type for leading address

The composition of the link type to be specified depends on whether the link branch concerned is a GI branch or an CGI branch.

- GI link branch The GI link branch is defined by specifying a link type between K1 and K4.
  =K1 ... K4 See table link types
  =K=R1, P2 With R parameter contents of between 1 and 4.
- CGI link branch The CGI link branch is defined by specifying link type K11 or K12 and via the CGI number.
  =K11 ... K12 See table link types
  =K=R2, P1 With R parameter contents between 11 and 12.
  - <CGI number>
    =CGI1 ... IKA32 It is mandatory to specify the CGI number either
    =CGI=R2, P1 directly or indirectly.

Following address A following axis name**/**axis address**/**spindle name**/**spindle address can be specified as the following address.

- <following axis**/**spindle name>
  =X1=S1 Direct specification of following axis**/**spindle name X1**/**S1.
- <following axis**/**spindle name>
  =@441**/**3<value> By programming command @441**/**3, it is possible to specify the following axis/spindle indirectly (for use with cycles).
  <value> Corresponds to the global axis/spindle number.

Examples:

G401 X K1 X1= $^{L}_{F}$	Gear interpolation grouping with X1 as following drive and X as leading drive with link type K1.
G401=S2= K2 Z1 S1=$^{L}_{F}$	Leading spindle (S2) with link type K2 and following spindle (S1). Gear interpolation grouping with leading axis (Z1). The link type of the leading axis is read by the machine data (default setting).
G401 X1=K2 X2=K11 IKA17 Y K4 C1	C1 is linked to X1 via GI with K2, to X2 via IKA17 with K11 and to Y via GI with K4.
G401 S1=K1 Y K2 Z K1 C	

Notes:

- A G401 must always be preceded by a G400 (global LINK_OFF for all link branches of a link structure) which separates all the link branches configured to this axis/spindle.
- When the GI**/**CGI link branch is deactivated with G400 the last position of the following axis is maintained, i.e. the following axis path generated with CGI is not completed on deactivation (G400).
- If a position is programmed directly or indirectly via@ 440**/**2 with position specification for a leading or following axis**/**spindle, alarm 3006 ”Wrong block structure” is output.
- If a leading address is linked to the same following address via two link branches, alarm 1220***/**2066* Configuration impermissible is output.
- If an CGI number is configured for which an output quantity is already defined, alarm 1268* ”CGI path reconfiguration illegal” is output.
- If an IKA number is configured for which an output quantity is already defined, alarm 1268* ”CGI path reconfiguration illegal” is output
- As a result of the configuration, %IKA1 data and gear interpolation status data are adapted in relation to one another or the configuration data are set to their initial values.
- When a GI/CGI link structure is defined by a G401 block, all the GI**/**IKA link branches involved must always be specified by means of their GI or IKA link type and leading addresses. This also applies even if only one single GI**/**IKA is to be re-configured. A reconfiguration always constitutes a new configuration when the old configuration has been deleted beforehand.
- After **G401 FA**, no further computational load is placed on the grouping by the gear interpolation.
- Up to 5 leading drives and one following axis can be defined in a gear interpolation grouping. The leading drives can be real or simulated drives, rotary or linear axes or spindles. A following axis can be either a real rotary or linear axis.
- Up to 4 leading axes + 1 leading spindle and one following spindle can be defined in a gear interpolation grouping. The leading axes can be real or simulated rotary or linear axes. If a following spindle is defined in the grouping, then exactly one leading spindle must be defined in that grouping.

*FA = Following axis**/**following spindle*

- For the option "Synchronous spindle", only one leading spindle can be defined.
- Each gear interpolation grouping must have its own configuration block in the NC program.
- When the gear interpolation grouping is configured, all link factors are initially set with zero. Only when LINK_ON is programmed do the values programmed with LINK_ON become active and setpoints are generated for the following axis.
- It is not necessary to enter a number for the link type (Kn) of the leading drives if the link type has been preset in the machine data.
- On changing a configuration by adding or removing a leading drive or changing the link type, a complete NC block with all leading drives and link types must be written. The old configuration must first be deleted.
- LINK_OFF and DELETE CONFIGURATION must first be programmed before the configuration can be changed.
- When reprogramming the link factor to KF = 0, a setpoint value is no longer generated for the following drive, but the link remains in the LINK_ON state.
- The gear configuration can be changed in any channel belonging to the mode group. The last command to be programmed is active. There is no priority.
- The defined gear configuration remains active after
  - End of block
  - End of program
  - Mode group change
  - Warm restart
  - Power Off
- On deleting the configuration, only the following drive must be specified.
- A spindle which can also operate as a C axis must not be declared simultaneously as synchronous spindle in one gear interpolation grouping and a following axis (C axis) in another gear interpolation grouping.

Delete GI/CGI configuration

Program syntax

G401 <following address> This command deletes the entire link structure for this following address output.

Following address A following axis name/axis address/spindle name/spindle address can be specified as the following address.

- <following axis/spindle name>
  =X1=S1= Direct specification of following axis/spindle name X1/S1.
- <following axis/spindle address>
  =@441/3 <value> By programming command 441/3, it is possible to specify the following axis/spindle indirectly (for use with cycles).
  <value> Corresponds to the global axis/spindle number.

Example

G401 X1= L_F Delete configuration

Notes:

- The configured link types (all GI/CGI link types in up to 5 definable link branches) are also deleted by **Delete GI/CGI configuration**.
- Some of the %IKA1 data stored in the RAM are deleted by **Delete GI/CGI configuration**.

8.12.2.2 Switch on/over GI/CGI link (G402)

The G402 function is used to switch on/over the gear interpolation function and to change the link factor. It is also possible to select individual leading drives in the gear interpolation grouping for linking to the following drive.

A link activated with G402 can be deactivated with G400. Setting the link factor to zero does not alter the gear interpolation status LINK_ON.

G402 can be used to

- switch on/over/off individually selected link branches according to the GI/CGI link structure defined with G401. In this case, all addressed link branches are immediately and simultaneously switched.
- **conditionally** switch on/over/off individually selected link branches according to the GI/CGI link structure defined with G401. In this case, all addressed link branches are switched simultaneously only when all specified conditions are or have been fulfilled. A condition for switch on/over/off is defined through the specification of one or several setpoint/actual start positions of the leading axes/spindle (in input format) with optional specification of overtravel direction for the leading addresses.
- switch all configured link branches again according to the GI/CGI link structure defined with G401 and the previously programmed state (G402 selectively**) without consideration of any programmed start conditions.**

The link branches created with G402 can be cancelled by means of G400 or via the input display.

Switch GI/CGI link branches on/over/off selectively

The G402 function is used to switch on/over the gear interpolation function and to change the link factor. It is also possible to select individual leading drives in the gear interpolation grouping for linking to the following drive.

A link activated with G402 can be deactivated with G400. Setting the link factor to zero does not alter the gear interpolation status LINK_ON.

Program syntax:

```
G402  <leading addres 1>{start position}{overtravel direction of leading address 1}
      <link parameter for leading address1>
      ...
      <leading address 5>{start position}{overtravel direction of leading address 5}
      <link parameter for leading address5>
      <following address>
```

Up to 5($_n$) leading axis names/axis addresses/spindle names can be specified as the leading address.

- <leading axis/spindle name>
  X1=S1 Direct specification of leading axis/spindle name X1/S1.
- <leading axis/spindle address>
  =@441/3<value> By programming command @441/3, it is possible to specify the leading axis/spindle indirectly (for use with cycles).
  <value> Corresponds to the global axis/spindle number.
- <leading axis/spindle name$_n${<start position>}>
  Specification of the start position is permissible as an option.
  =X100 S1=90 Specification of start position 100 for leading axis X and 90 degrees for leading spindle S1.
- <leading axis/spindle address$_n${<start position>}>
  =@440/2 <value3><value>
  Specification of the start position is permissible as an option.
  <value3> Corresponds to the global axis/spindle number.
  <value> Corresponds to the start position.

  By programming command @440/2, it is possible to indirectly specify the leading axis/spindle address with start position (for use with cycles).

- <overtravel direction of leading address$_n$>

  Specification of the overtravel direction is optional. The overtravel direction is meaningless if no specific direction is stated. Specification of the direction is only relevant when G402 with start positions is programmed. If the value specified for the direction is not equal to 1/0/-1, error message 3006 ”Wrong block structure” is generated.

  =K-1, K=Rxx With K-1 or K=Rxx, where Rxx is set to -1, a negative overtravel/approach direction is specified as the switch on/over/off condition.
  =K1, K=Rxx With K1 or K=Rxx, where Rxx is set to +1, a positive overtravel/approach direction is specified as the switch on/over/off condition.
  =K, K0, K=Rxx With K, K0 or K=Rxx, where Rxx is set to 0, no overtravel/approach direction is specified.

Link parameters for leading address$_n$

The composition of the link parameters to be specified depends on whether the link branch concerned is a GI or CGI branch.

- GI link branch:

  The GI link branch can be configured for this leading address via G401. The link factor must be specified for a GI link branch.

  - <link factor>
    =I<numerator> J<denominator>
    =I2 J1
    =I=R2 J=P3
    The link factor of the GI link branch is defined through the direct or indirect specification of I and J. The numerator and denominator can amount to 8 decades with sign and comma depending on the input resolution.

- CGI link branch:

  The CGI link branch must have been configured for this leading address via G401. The IKP number, link factor and IKP number must be specified for the CGI link branch.

  - <IKA number>
    =IKA1 ... IKA32
    =IKA=R1, P2
    Specification of the IKA number, either directly or indirectly, is mandatory and is already defined in G401.

  - <link factor>
    =I<numerator> J<denominator>
    =I2 J1
    =I=R1 J=P2
    The link factor (GI status data) is defined through the direct or indirect specification of I and J. The weighting factor for CGI link branches is not taken into account (action 1:1), is output in the display as 1:1. The numerator and denominator can amount to 8 decades with sign and comma depending on the input resolution.

  - <IKP number>
    =IKP1 ... IKP32
    =IKP=R3, P4
    IKP is used to directly or indirectly assign one out of 32 possible control curves to the CGI link branch. The number specified for CGI need not correspond to that specified for IKP. The corresponding T parameter of the %IKA1 data field is overwritten. This allows block-related switching over from one control curve to another.

Following address

A following axis name/axis address/spindle name/spindle address can be specified as the following address.

- <following axis/spindle name>
  =X1=S1
  Direct specification of following axis/spindle name X1/S1.

- <following axis/spindle address>
  @441/3 <value>
  By programming command @441/3, it is possible to specify the following axis/spindle indirectly (for use with cycles).
  <value>
  Corresponds to the global axis/spindle number.

Notes:

- LINK_ON**/**CHANGE LINK can be selected for individual leading drives or for the whole gear interpolation grouping in the NC part program.
- Two NC program blocks each have to be written for LINK_ON**/**CHANGE LINK for the gear interpolations of two gear configurations.
- When a following spindle is programmed, the one possible leading spindle must also be programmed. If no link movement is desired, link factor "0" must be entered.
- If only one link factor is to be altered, only this leading drive has to be entered in the G402 command.
- If a position is programmed for a following axis/spindle or if the @440**/**2 command with position specification is programmed for a following axis/spindle address, alarm 3011 ”Axis twice or too many axes” is output.
- If only the following address is specified without leading addresses and link parameters, G402 (global GI**/**CGI LINK_ON) activates all GI**/**CGI link branches in the GI**/**CGI link structure with the last link branch parameters programmed according to the above syntax (link factors, IKA number and control curve number, etc.).
- When a link branch is addressed through specification of its leading address, G402 acts selectively for this link branch.
- A GI**/**CGI link branch can be selectively separated by setting the constant link factor**/**weighting factor (I, IO) to zero; however, processing of the GI logic continues (transfer time).
- It is possible to specify the link branch parameters for a maximum of 5 link branches which have been defined by means of the programmed link type.
- A leading address may only be written once in a G402 block; alarm 3011 "Axis twice or too many axes" with decoding stop is otherwise output.

Examples:

```
G401  X1=K2  X2=K11  IKA17  Y  K2  C1
```

C1 is linked to X1 via GI with K2 and to X2 via IKA17 with K11 as well as to Y via GI with K2.

(LINK_ON**/**OVER)

```
G402  X1=100  K-1  I1  J1  Y101  I1  J1  C1
```

Selective LINK_ON**/**OVER of GI link branches X1 C1 and X C1 after overtravel of both start values (positions), the overtravel direction being negative for the X1 axis and without significance for the
Y axis.

```
G402  X1=100  I1  J1  C1
```

Selective LINK_ON**/**OVER of GI link branch X1 C1 after overtravel of the start value (position), overtravel direction irrelevant.

```
G402  X2=100  IKA5  I1  J1  IKP11  X1=101  I2  J2  C1
```

Selective (not all defined link branches) LINK_ON/OVER of CGI link branch X2 C1 and of the linear link branch X1 C1 after overtravel of both start values (position), overtravel direction irrelevant.

G402 X2=100 IKA5 I1 J1 IKP11 C1

Selective LINK_ON/OVER of CGI link branch X2 C1 after overtravel of start value (position), overtravel direction irrelevant.

G402 FA LF Switch on link (entire gear interpolation grouping). In this case, the last link factors to be programmed of the individual leading drives are active. Only the name of the following drive is specified.

G402 LA2 I J FA LF Switch link on**/**over selectively.

G402 X I1 J3 Y LF LINK_ON selectively with leading axis X, gear ratio 1:3 and following axis Y

Explanation:

LA1...LA5	Names of leading axes**/**spindles
I	Link factor numerator (8 decimal places + sign + decimal point)
J	Link factor denominator (8 decimal places + sign + decimal point)
FA	Name of following drive, gear ratio I : J

G402 Y LF LINK_ON of entire GI grouping with the last link factors to be programmed of the individual leading drives. Only following axis Y is programmed.

G402 X I1 J2 C I2 J5 Y LF Leading axis X with gear ratio 1:2, leading axis C with gear ratio 2:5, following axis Y

Switch GI/CGI link branch(es) on/over/off, general

Program syntax:

```
G402  <following address>
```

- <following address> A following axis name**/**axis address**/**spindle name**/**spindle address can be specified as the following address.
  - <following axis**/**spindle name>
    =X1=S1= Direction specification of following axis**/**spindle name X1**/**S1.
  - <following axis**/**spindle address>
    =@441**/**3 <value>
    By programming command @441**/**3, it is possible to specify the following axis/spindle indirectly (for use with cycles).
    <value> Corresponds to the global axis/spindle number.

Notes:

- If a position is programmed for a following axis**/**spindle or the @440**/**2 command with position specification programmed for a following axis**/**spindle address, alarm 3006 "Wrong block structure" with decoding stop is output.
- The link parameters programmed by means of G402 in front of G400 become active again for all link branches through G402 "following address", i.e. the data such as link**/**weighting factor and control curve number are activated again. If a link branch was not programmed in front of the G402 "following address", the numerator/denominator ratio of 0:1 (inital setting of GI status data with G401) is active. Start positions programmed beforehand and any overtravel directions are lost. All link branches are switched immediately and simultaneously.

Example: (configuration)

```
G401  X1=K2  X2=K11  IKA17  Y  K2  C1
```

C1 is linked to X1 via GI with K2 and to X2 via IKA17 with K11 as well as to Y via GI with K2.

(LINK_ON**/**OVER)

```
G402  X1=100  I1  J1  C1
```

Selective LINK_ON/OVER of GI link branch X1 C1 after overtravel of the start value (position), overtravel direction irrelevant.

```
G402  X2=100  IKA17  I1  J1  IKP11  X1=101  I2  J2  C1
```

Selective LINK_ON**/**OVER of CGI link branch X2 C1 and of linear link branch X1 C1 after overtravel of both start values (positions), overtravel direction irrelevant.

(LINK_OFF GENERAL)

```
G400  C1
```

Switches off all link branches.

(LINK_ON GENERAL)

```
G402  C1
```

Switches on the active link branches in front of the G400 command again, i.e.:

- X1 is switched on with link factor 1:1 with K2
- X2 is switched on with link factor 1:1 with K11 via IKA17 with IKP11
- Y is switched on with link factor 0:1 with K2

in each case, without position reference.

8.12.2.3 On-the-fly synchronization of GI/CGI link branches (G403)

"On-the-fly synchronization" describes the process where the link is switched on when the leading drive is either stationary or already running (following drive can also be in motion) and the position of the drives is synchronized automatically. An overlay path is applied to the following drive in the form of a velocity offset.

G403 is programmed by specifying the synchronized positions of the leading drives involved. This activates the link immediately. Synchronization is carried out as if the leading drives and the following drive had departed from the programmed synchronized position. After synchronization, the leading and following drives run in the programmed angular position or assigned position.

When programming G403, up to 5 leading drives can be specified. If no link factor is programmed, the last link factor of this leading drive to be programmed is active.

A link activated with G403 can be deactivated with G400. Setting the link factor to zero does not alter the gear interpolation ON LINK status.

G403 allows GI/CGI link branches to be synchronized "on the fly", i.e.:

- the link is switched on/over/off selectively according to the programmed link factor; immediately afterwards, the leading addresses and the following address are synchronized with the programmed reference position assignment.
- the link is switched on/over/off globally according to the programmed link factor which requires that all following address link branches configured via G401 be programmed with synchronous positions.

The link branches created with G403 can be cancelled by means of G400, via the input display or from the PLC.

Program syntax:

```
G403  <leading address 1<synchronous position>>
      {<link parameter for leading address 1>}
      ...
     {<leading address 5<synchronous position>>
      {<link parameter for leading address 5>} <following address<synchronous position>>
```

<leading address 1<synchronous position>>

The synchronous position must be specified. If no synchronous position is specified, the axis is synchronized with the present position, i.e. the leading axis should already be in the appropriate synchronous position.

- <leading axis/spindle name$_n$<synchronous position>>
  =X100 S1=90 Specification of synchronous position 100 for leading axis X and 90 degrees for leading spindle S1.
- <leading axis/spindle address$_n$<synchronous position>>
  =@440/2 <value 3><value>
  The @440/2 commands must be used for indirect specification.
  <value 3> Corresponds to global axis/spindle number.
  <value> Corresponds to synchronous position.

{<Link parameters for leading address$_n$>}

The composition of the link parameters to be specified depends on whether the link branch concerned is a GI or CGI branch.

- GI link branch — The GI link branch must have been configured for this leading address via G401. The link factor must be specified for a GI link branch.
  - <link factor>
    =I<numerator> J<denominator>
    =I2 J3
    =I=R2, J=P3

    The link factor of the GI link branch is defined through the direct or indirect specification of I and J. The numerator and denominator can amount to 8 decades with sign and comma depending on the input resolution.
- CGI link branch — The CGI link branch must have been configured for this leading address via G401. The IKP number, link factor and IKP number must be specified for the CGI link branch.
  - <IKA number>
    =IKA1 ... IKA32
    =IKA=R1, P2

    The specification of the IKA number, either directly or indirectly, is mandatory since the appropriate CGI link branch, which has been configured via G401, is addressed via this number. This function prepares for any intended extension allowing a leading address to be linked to a following address, both linearly as well as non-linearly several times.
  - <link factor>
    =I<numerator> J<denominator>
    =I2 J3
    =I=R1, J= P2

    The link factor (GI status data) is defined through the direct or indirect specification of I and J. The weighting factor for CGI link branches remains in initial setting 1:1.

    The numerator and denominator can amount to 8 decades with sign and decimal point depending on the input resolution.
  - <IKP number>
    =IKP1 ... IKP32
    =IKP=R3, P4

    IKP is used to directly or indirectly assign one out of 32 possible control curves to the CGI link branch. The number specified for CGI need not correspond to that specified for IKP. The corresponding T parameter of the %IKA1 data field is overwritten. This allows block-related switching over from one control curve to another.

<following address<synchronous position>>

It is necessary to specify the synchronous position. If no synchronous position is specified, alarm 3006 "Wrong block structure" is output.

- <following axis/spindle name$_n$<synchronous position>>
  =Y200 S2=270

  Specification of synchronous position 200 for following axis Y and 270 degrees for following spindle S2.
- <following axis/spindle address$_n$<synchronous position>>
  =@440/2 <value 3><value>

  It is necessary to specify the synchronous position. @440/2 commands must be used for indirect specification.

  <value 3> Corresponds to global axis/spindle number.

  <value> Corresponds to synchronous position.

Examples:

```
G403  X100  I1  J1  -Z23  I2  J4.5  X3=33  LF
```

Leading axis X with synchronous position 100 and gear ratio 1:1,
leading axis Z with synchronous position -23 and gear ratio 2:4.5
following axis X3 with synchronous position 33

```
G403  S1=50 I1 J1 S2=100 LF
```

Leading spindle S1 with synchronization 50 and gear ratio 1:3,
following spindle S2 with synchronization 100
The leading and following spindles are assigned such as if both spindles had been started at S1=50 and S2=100.

```
G403  X100  I1  J1  C1=70
```

Selective GI/CGI LINK_ON/OVER of the GI link branch X C1 with synchronization to the initial values X100 and C1 = 70

```
G403  X60  Y101  I1  J1  C1=30
```

Selective GI/CGI LINK_ON/OVER of the

- GI link branch X C1 with synchronization to the initial values X60 and C1 = 30 and of the
- GI link branch Y C1 with synchronization to the initial values Y101 and C1 = 30.

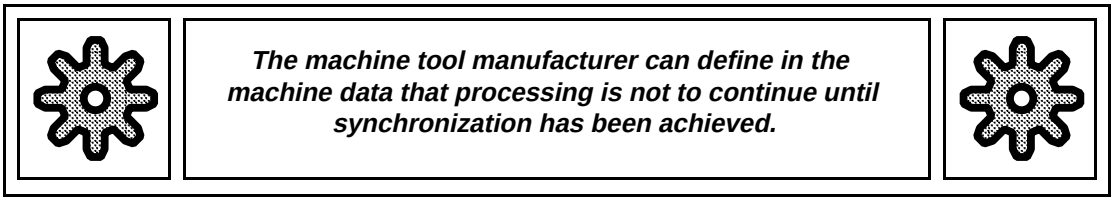
Notes:

- When a link branch is addressed through specification of its leading address, G403 acts **selectively** for this link branch.
- A new G403 block cancels the previous reference position.
- A GI/CGI link branch can be separated selectively by setting the constant link factor (numerator = 0, denominator 0) to zero; however, processing of the GI logic continues (transfer time).
- A drive cannot be defined as following drive for more than one gear interpolation grouping.
- When a following spindle is programmed, the only possible leading spindle must also be programmed. If no link movement is desired, link factor "0" must be entered.
- The synchronized positions can either be defined in absolute or incremental dimensioning (G90/G91). As the absolute dimension for the base interpolations may differ, mixed programming can be used in a block containing both.

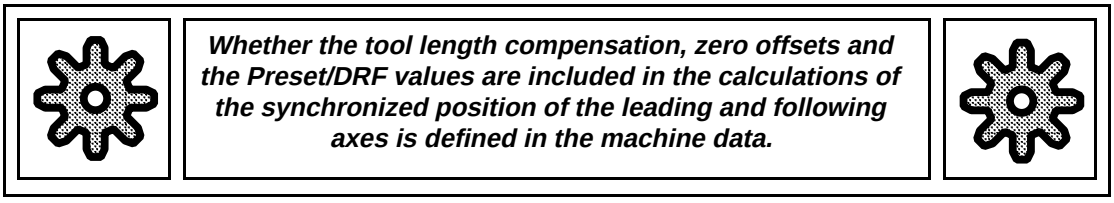
  Example: `N10  G01  G90 X200  F2000  G403  G91  X1000  I1  J3  Y  LF`

- Rotary axes:
  When programming in absolute dimensions, the specified position must lie within one revolution (<360 degrees); when programming in incremental dimensions the travel path can cover several revolutions.
- During ELG operation the speed of the leading spindle (as far as the encoder limit frequency is concerned) is neither monitored nor limited.

- The overlay path is traversed by following axes at incremental velocity and by following spindles with creep speed M19 as overlaid additional velocity.
- In the case of spindles, the position is always converted modulo 360°.
- It is possible to specify the link branch parameters for a maximum of 5 link branches which must be configured via G401.
- A leading address may only be written once in a G403 block; alarm 3011 "Axis twice or too many axes" with decoding stop is otherwise output.



The machine tool manufacturer can define in the machine data that processing is not to continue until synchronization has been achieved.



Whether the tool length compensation, zero offsets and the Preset/DRF values are included in the calculations of the synchronized position of the leading and following axes is defined in the machine data.

Synchronize GI/KA link branches on the fly, general

EThe same syntax apply as for G403 selective. However, all following address link branches configured via G401 must be programmed in one block with synchronous positions.

It is not meaningful to use a G403 <following address> command because the specification of a leading address and the following address synchronous position is required. Alarm 3006 "Wrong block structure" is output.

8.12.2.4 Synchronize GI/KA link branches on the fly with gears (G403) (SW 6 and higher)

General

Through parameterizing in the NC machine data (see next page), the following rotary axis and the following spindle can be switched back and forth between the old and the new functionality. When gear cutting using the new functionality, the maximum motion of the following axis through synchronization can be limited to half a tooth pitch. The new functionality, however, is not only useful in gear cutting.

Programming

Synchronization is still carried out by using the G403 command. With gear cutting, the link factors of the first GI linking structure are programmed as usual with the tooth numbers of the work gear and the cutting gear. Definition of the first GI linking structure is carried out via the first leading axis in the G401 block.

This circumstance is used to facilitate user programming. A second, additional input of the tooth number of the following axis or following spindle is therefore not necessary. Details given in the first GI linking structure of the following axis tooth number are binding for the function of following axes. The next possible "n" synchronous position is approached by the shortest route, with "n" being the denominator of the first GI linking structure. The maximum synchronization route is therefore calculated to 1/2 rotation/n. If a time-optimized synchronization is parameterized in the tooth pitch (MD 1856*/531*, bit 4=1), the synchronization position above half the maximum speed of the following axis/spindle is no longer approached by the shortest route but "retarding". The maximum synchronization route is then calculated to 1 rotation/n.

Special features of following spindle

With a following spindle the link factors, however, from the leading spindle to the following spindle are used for input of the tooth number for the following spindle instead of the link factors of the first GI linking structure. The sequence of the leading axes/spindles when programming the G401 is therefore of no importance.

Parameterization

The new functionality can be activated for the following axis/spindle through setting bit 7 in the MD 1848*/526* to 1 .

Spindle with assigned C axis

When configuring a following spindle (e.g. G401 S1=K1 S2=) the parameterization of the spindle is evaluated, whereas the parameterization of the axis is evaluated when a following axis is configured (e.g. G401 C1=K1 C2=). This also applies to a spindle/C axis combination. Should both a GI configuration with a following spindle and with a following axis be used one after another, and the new functionality be used for both configurations, bit 7 must be set both in MD 526* as well as in MD 1848* .

A check to see if the axis parameterization and the spindle parameterization agree is not carried out.

Compatibility

Through the standard assignment of these bits with 0, compatibility with the old software versions is retained.

Duration of the synchronization process

Up to now the fastest possible synchronization process was achieved for high speed following axes/spindles by slowing down of the following axis/spindle. This is also possible for synchronization in tooth pitch, if the time-optimized synchronization in the MD 1856*/531*, bit 4 is selected. Thereafter, however, there is still no synchronization possible at the cutting point of the gear. Otherwise the user must allow the following axis/spindle enough speed reserves to accelerate so that the synchronization path can be terminated quickly. On average synchronization from standstill is finished faster with the new functionality due to mostly shorter approach routes.

Accuracy

The accuracy of the synchronization process is defined by the selected position control resolution and the encoder resolution. Whereby the position control resolution is mostly the rougher of the two limits. Rest values after division by the tooth number which are smaller than the position control resolution cannot be taken into account. It is, however, sure that this error does not summate with repeated synchronization processes.

Example

Position control resolution 0.5 $x10^{-3}$ for all axes.

Setpoint linkage of C1 and X on the C2 following axis, whereby the denominator of the linking relation of C1 to C2 at the same time specifies the tooth number.

G401 C1=K1 X K1 C2

Before the synchronization process, the following positions were reached:

X100. C1=30. C2=90

Synchronization of C2 to the starting values C1=30, X=100 and C2=60 with tooth numbers of C2 equals 13 and C1 equals 7. The tooth number 13 gives a tooth pitch of 27,692308 degrees.

G403 C1=30 I7 J13 X100 I1 J1 C2=60

The following resulting synchronous positions for C2 are possible:

Synchronous position	60	87.692	115.385	143.077	170.769	198.462	226.154
Teeth	0	1	2	3	4	5	6
Synchronous position	253.846	281.538	309.231	336.923	4.615	32.308	
Teeth	7	8	9	10	11	12	

After the synchronization process, the following positions are reached:

X100. C1=30. C2=87.692 (=60+1 tooth pitch)

8.12.2.5 Switch off GI/CGI link branches (G400)

When the link is switched off, all the link factors remain unaltered. G400 can either be programmed selectively for certain leading drive links or for the whole gear interpolation grouping.

Up to 5 leading drives can be specified when G400 is programmed.

G400 can be used to

- switch off individually selected link branches according to the GI/CGI link structure defined with G401. In this case, all addressed link branches are immediately and simultaneously switched.
- **conditionally** switch off individually selected link branches according to the GI/CGI link structure defined with G401. In this case, all addressed link branches are switched simultaneously only when all specified conditions are/have been fulfilled. A condition for switch off is defined through the specification of one or several setpoint/actual start positions of the leading axes/spindle (in input format) with optional specification of the leading address overtravel direction.
- switch off all configured link branches again according to the GI/CGI link structure defined with G401 and the previously programmed state (G402 selectively) **without consideration of any programmed start condition(s)**.

The link branches created with G402**/**3 can be cancelled by means of G400, via the input display or from the PLC.

Switch off GI/CGI link branches selectively

Program syntax:

```
G400  <leading address 1> {stop position}{overtravel direction of leading address 1}
      <link parameter for leading address 1>
      {...
      <leading address 5> {stop position}{overtravel direction of leading address 5}
      <link parameter for leading address 5>
      <following address>
```

<leading address$_n$>

- <leading address$_n$> Up to 5($_n$) leading axis names**/**axis addresses**/**spindle names**/**spindle addresses can be specified as the leading address.
- <leading axis**/**spindle name$_n$>
  X1S6 Direction specification of leading axis/spindle name X1 and S6.
- <leading axis**/**spindle address>
  @441**/**3 <value>
  <value> Corresponds to global axis number.
  By programming command @441**/**3, it is possible to specify the leading axis**/**spindle indirectly (for use with cycles).
- <leading axis**/**spindle name$_n$**{**<stop position>**}**>
  Specification of the stop position is permissible as an option.
  =X100 S1=90 Specification of stop position 100 for leading axis X and 90 degrees for leading spindle S1.

- <leading axis/spindle address$_n$<{stop position>}>
  =@440/2 <value 3><value>
  Specification of the stop position is permissible as an option.
  <value 3> Corresponds to the global axis/spindle number.
  <value> Corresponds to the stop position.
  By programming command @440/2, it is possible to indirectly specify the leading axis/spindle address with stop position (for use with cycles).

- <overtravel direction of leading address$_n$>
  Specification of the overtravel direction is optional. The overtravel direction is meaningless if no specific direction is stated.
  Specification of the direction is only relevant when G400 with stop positions is used. If 1 or >1 is entered as the direction value, alarm 3000 "General programming error" is output.
  =K-1, K=Rxx With K-1 or K=Rxx, where Rxx is set to -1, a negative overtravel/approach direction is specified as the switch off condition.
  =K1, K=Rxx With K1 or K=Rxx, where Rxx is set to +1, a positive overtravel/approach direction is specified as the switch off condition.
  =K, K0, K=Rxx With K, K0 or K=Rxx, where Rxx is set to 0, no overtravel/approach direction is defined.

<Link parameters for leading address$_n$>

The composition of the link parameters to be specified depends on whether the link branch concerned is a GI or CGI branch. The link branch(es) must have been configured via G401 for this leading address and be active via G402/3.

- GI link branch
  No link parameters may be specified. The link factor programmed via G402/3 (I<numerator value> J <denominator value>) is not overwritten. The positions programmed with G402/3 are no longer existent/active.

- CGI link branch
  The CGI link branch must have been configured for this leading address via G401. The IKA number must be specified for the CGI link branch. The link parameters, link factor and IKP number programmed via G402/3 are not overwritten. The positions programmed with G402/3 are no longer existent/active.
  - <IKA number>
    =IKA1 ... IKA32 Direct or indirect specification of the IKA number is mandatory
    =IKA=R2, P3 and already defined in G401.

<following axis>

- <following axis/spindle name>
  =X3=S2= Direct specification of following axis/spindle name X3/S2.

- <following axis/spindle address>
  =@441/3 <value> Indirect specification via @441 for axes or @443 for spindles.
  <value> Corresponds to global axis/spindle number.

  By programming @441/3, it is possible to indirectly specify the following axis (for use with cycles).

Notes:

- If a position is programmed for a following axis/spindle or if the @440/2 command with position specification is programmed for a following axis/spindle address, alarm 3006 "Wrong block structure" with decoding stop is output.
- When a link branch is addressed through specification of its leading address, G400 acts selectively for this link branch.
- It is possible to specify the link branch parameters for a maximum of 5 link branches which have been defined by means of the programmed link type.
- A leading address may only be written once in a G400 block; alarm 3011 "Axis twice or too many axes" with decoding stop is otherwise output.
- After LINK_OFF, axis synchronization (G200) is needed internally for the following drive if it is to be positioned absolutely.
- When switching off the link with G400, some gear interpolation software modules may continue to run, causing computation to continue even when the link is switched off. Remedy: Program G401 FA. The link factors will, however, have to be reprogrammed when the function is reselected.

  Remedy:
  Program G401 FA. However the link factors must be re-entered in the case of new selection.

Example:

`G400 X2=100 IKA5 C1`	Selective LINK_OFF of CGI link branch X2 C1 after overtravel of stop value (position) 100, overtravel direction irrelevant.

Switch off GI/CGI link branches, generally

Program syntax:

```
G400  <following address>
```

- <following address> — A following axis name/axis address/spindle name/spindle address must be specified as the following address.
- <following axis/spindle $name_n$>
  =X3=S2= — Direct specification of following axis/spindle name X3/S2.
- <following axis/spindle address>
  =@441/3 <value> — Indirect specification via @441 for axes or @443 for spindles
  <value> — Corresponds to global axis/spindle number.
  By programming command @441/3, it is possible to indirectly specify the following axis/spindle (for use with cycles).

Notes:

- If a position is programmed for a following axis/spindle or if command @440**/**2 with position specification is programmed for a following axis**/**spindle address, alarm 3006 "Incorrect block structure" with decoding stop is output.
- The link parameters programmed via G402 in front of G400 remain unaltered in spite of G400 "following address". When the next G402 is reached, the link parameters are activated again (not the start position and the overtravel direction, these need to be written explicitly for each G402 block).
- With G400 S4 the actual following spindle speed active at this point of time is considered as setpoint speed.

Examples:

G402 X2=100 IKA5 I1 J1 IKP11 X1=101 I2 J2 C1

Selective LINK_ON**/**OVER of CGI link branch X2C1 and of linear link branch X1 C1 after overtravel of both start values (starting positions), overtravel direction irrelevant.

(LINK_OFF general)

G400 C1 Switches off all link branches.

G400 X L_F LINK_OFF general, following axis X

8.12.3 Speed limitation for GI/KGI following axes (G405, G404)

The exceeding of the maximum speed of the GI/KGI following axes is prevented or reduced through limitation of the leading axis/leading spindle speed. The leading axes/leading spindles to be limited must be activated through programming the G405. Activation can be reversed by programming the G404.

Programm syntax for **"Activate speed limits for following axes"**:
G405 {<axis identifier>} {<spindle identifier>}

Programm syntax for **"Deactivate speed limits for following axes"**:
G404 {<axis identifier>} {<spindle identifier>}

The G functions must be programmed alone in the block. Up to 5 axes and one spindle can be programmed in one block. Activation/deactivation of the axes and spindles has an additive effect, i.e. the status of non-programmed axes/spindles remains unchanged, the programming of G404 wihtout axes/spindles deactivates the function for all axes and spindles of the mode group.

Note:

The first leading axis of the chain and possibly all following axes which can be moved by following axis override must be activated for GI**/**KGI links.

Example:

1st Link: X1 –> X2, X2 axis can be moved with following axis override

2nd Link: X2 –> X3

In order not to exceed the maximum speed of the X3 following axis, the speed limit must be activated for the leading axes X1 and X2.

8.13 Interpolative compensation with tables (IKA)

Parameterization

Three data areas are available for the parameterization of IKA which the user can assign according to his requirements:

- %IKA 1 Definition of IKA relation
- %IKA 2 Definition of compensation curves
- %IKA 3 Definition of compensation points

%IKA 1:

This data area contains all the definitions for the **32** possible **IKA relations**. A parameter block is available in this data area for each IKA relation which can be programmed in a number of different ways.

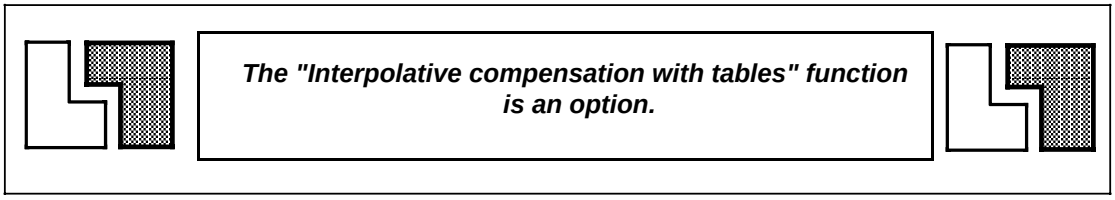
%IKA 2:

For each of the **32** possible **compensation curves** the associated compensation curves can be selected via the start pointer and end pointer and combined to form the actual control curve.

%IKA 3:

This data area represents the sum of all the used and unused **compensation points**. The data field range can be defined via the flexible memory configuration. The maximum configuration consists of 65536 value pairs, each consisting of the position of the input variable and the associated value of the output variable.

8.13.1 Define/delete IKA configuration (G411)



The configuration for an IKA permits the user to determine which quantities are to be applied as input A or B and which quantity is to be controlled by the output. G411 is "IKA-number-specific". A separate G411 block is required for each configuration. Cascading, i.e. programming the output of an upstream IKA as the input to an IKA further downstream, is implemented through specification of a (transfer) R parameter. Once the configuration has been programmed, it remains valid until a "Delete IKA configuration" operation.

G411 merely allows an IKA configuration to be defined or deleted. G412 is required to generate or inject the output values.

Input/Output	Quantity	Link type
Input A	Axis set position	K21
Input A	Axis actual position	K22
	R parameter/pointer	does not have to be defined
Input B	Axis set position	K31
—	R parameter/pointer	does not have to be defined
Output	Axis set position	K41
Output	Axis compensation value	K46
	R parameter/pointer	does not have to be defined
	Feedrate weighting (feed multiplication)	does not have to be defined

Define IKA configuration

Program syntax:

```
G411  <input quantity A>{link type for input quant A>}
      {<input quantity B>}{<link type for input quantity B>}
      <IKA number>
      <output quantity>{<link type for output quantity>}
```

<input quantity A>

=X1=, @441 <value> Direct or indirect specification of axis name. Analogous to GI programming, the link type (scanning of specified quantity of this axis) must be specified when axes are interconnected.

=Rxx, R=Pzz Specification of a global R parameter or pointer of which the contents are applied to input A. The link type need not be specified because the present value of the R parameter is always used.

{<link type for input quantity A>}

The specification of the link type is only mandatory for axes. When an illegal input is made, alarm 3000 "General programming error" is generated.

- <link type>
  =K_{ef} The relevant input is addressed via e (e = 2 corresponds to input A and e = 3 to input B); f is applied analogously to the GI link types.
  =K21, K=R, P Link type K21 connects input A to an axis setpoint position.
  =K22, K=R, P Link type K22 connects input A to an axis actual position.

{<input quantity B>} The specification of input quantity B is optional. If input B is not specified, the internal quantity (acc. to %IKA1 T parameter) is used.

=X1=, @441 <value> Direct or indirect specification of axis name. Analogous to GI programming, the link type (scanning of specified quantity of this axis) must be specified when axes are interconnected.

=R2, R=P3 Specification of a global R parameter or pointer of which the contents are applied to input A. The link type need not be specified because the present value of the R parameter is always used.

{<link type for input quantity B>}

The specification of the link type is only mandatory for axes. When an illegal input is made, alarm 3000 "General programming error" is generated.

=K31, K=R, P Link type K31 connects input B to an axis setpoint position.

<IKA number>

=IKA1 ... IKA32
=IKA=R1, P2 Specification of the IKA number is mandatory.

<output quantity> The output quantity specification determines how the output of the defined configuration is connected.

- <R parameter>
  =Ryy The IKA output can be written to a global R parameter (Ryy).
- <axis address>
  =X2, @441 <value> The IKA output can be injected for addition to the setpoint position input of a directly or indirectly specified axis.
- <feedrate weighting>
  =FM, FM_u, FM[X/R/P/K]
  The IKA output can be injected for multiplication to the feed override
  =FM, FM=R, P branch of the channel specified directly or indirectly via FM with R parameter contents = 0 (for independent, internal channel applications)
  =FM_u, FM=R, P of a channel specified directly or indirectly via FM with R parameter contents = 1 to 6 (for multi-channel applications)
  =FM[X3=,R,P,K] of an axis specified directly via FM [<axis name>] or an axis specified indirectly via FM [R parameter, pointer or constant] (FM stands for Feed Multiplication).

{<link type for output quantity>}

Specification of the link type is only mandatory for axes. When an illegal input is made, alarm 3000 "General programming error" is generated.

=K41, K=R, P Link type K41 connects the output to an axis setpoint position.

=K46, K=R, P Link type K46 connects the output to the axis compensation value.
The link type for axes must be specified in order to ensure consistent axis treatment.

Notes:

- If an axis is programmed directly or indirectly as the input or output quantity via 440 with position specification, alarm 3006 "Wrong block structure" with decoding stop is output.
- A configuration cannot be deleted (**Delete IKA configuration)** unless one has been defined (**Define IKA configuration**).
- When an IKA configuration is programmed via G411, the full syntax must be specified. This also applies, for example, when only one of the two inputs needs to be reconfigured. Reconfiguration/new configuration is only possible after a **Delete configuration** operation.
- When IKA configurations are cascaded, a dead time corresponding to one interpolation time reference may arise.

Example:

```
G411  X  K21  R700  IKA4  FM3
```

In this case, the feedrate weighting input for NC channel 3 is supplied by the IKA4; input A of IKA4 is connected to setpoint (K21) position of X and input B to the global R parameter 700.

Delete IKA configuration

Program syntax:

```
G411  <IKA Number>
```

<IKA Number>

=IKA1 ... IKA32
=IKA=R2, P3 This IKA configuration is deleted through specification of the IKA number.

Notes:

- **Delete IKA configuration** sets the %IKA1 data to their initial values which can be addressed via G411.
- A G411 <IKA No.> must be preceded by a LINK_OFF via G410 which separates this IKA link branch (see syntax and examples for G410 for further details) or it must be ensured that no G412 is active.
- The link branch is deactivated by means of G411 >IKA No.>. A new configuration can then be defined for this IKA number with another G411 (see **Define IKA configuration**).

Example:

```
G411  IKA4
```

Deactivates the addressed IKA link branch 4.

8.13.2 Switch IKA link branch on/over (G412)

G412 allows an IKA link branch configured via G411 with specification of a control curve IKA number to be

- switched on or over immediately or
- switched on or over subject to the additional specification of a start value/position and an overtravel direction (optional).

The characteristics of the IKA link branch are defined by the type of output quantity programmed, i.e. if the quantity is

- an absolute setpoint position,

  then an interpolation response is imposed by the output of the position value in the operating mode basic display, by the traversing command signals and by the halt of the last output value during switch-off (G410). In order to remove the output value, the branch must be switched over to a control curve. which supplies a zero output for the input range concerned. No internal deletion of distance to go takes place because the IKA link branches have no channel reference.
- an absolute compensation value,

  then only the actual value is corrected; the last active compensation value is reduced within the scope of the IKA limitations on link switch-off.
- an R parameter,

  then the R parameter is updated cyclically and the last output value halted on link switch-off. In order to remove the output value, the branch must be switched over to a control curve. which supplies a zero output for the input range concerned.
- a feedrate weighting,

  then the feedrate of a channel or an axis is weighted by the output value (feedrate changes are limited by the path**/**axis acceleration, i.e. multiplicative calculation before interpolator). The weighting action is deactivated when the link is switched off (factor 1).

Program syntax:

```
G412  {input quantity A{start position}{overtravel direction of input quantity A}
      <IKA number>
      {KOP_ON/OVER}
      {output quantity}
```

<input quantity A>{start position} {overtravel direction of input quantity A}
{weighting factor of output quantity}

- <input quantity A>
  =X1=, @441 <value>
  Direct or indirect specification of axis name.
  =Rxx, Pzz Specification of an R parameter or pointer of which contents are applied to input A.
- <input quantity A{<start position>}>
  It is possible to specify that switch on/over will take place only when the start position is reached. It is permissible to specify the start position as an option for axes acting as input quantities (leading axes).

X100 Specification of start position 100 for the input quantity, axis X.
=@440 <value 3><value>
<value 3> Corresponds to global axis number.
<value> Corresponds to start position.
By programming command @440, it is possible to indirectly specify the input quantity with start position (for use with cycles).

- <overtravel direction of input quantity A>
  An overtravel direction can be specified as an option. The overtravel direction remains irrelevant if no direction is programmed. An overtravel direction can be programmed only in conjunction with G412 with start positions and axes as input quantities.
  =K-1, K=Rxx With K-1 or K=Rxx, where Rxx is set to -1, a negative overtravel**/** approach direction is specified as the switch on**/**over condition.
  =K1, K=Rxx With K1 or K=Rxx, where Rxx is set to +1, a positive overtravel**/** approach direction is specified as the switch on**/**over condition.
  =K, K0, K=Rxx With K, K0 or K=Rxx, where Rxx is set to 0, no overtravel**/**approach direction is defined.

<IKA number>

=IKA1 ... IKA32
=IKA=R3, P4 When the IKA number is specified, the appropriate configuration defined by means of G411 is applied. The input quantity B has already been specified by means of G411 and must not therefore be programmed again for G412.

<LINK_ON/OVER>

An individual IKA link branch is switched on/over through specification of the IKP number and the weighting factor.

- <weighting factor>
  =I<numerator value> J<denominator value>
  =I2 J3 The weighting factor (IKA T parameter) is defined through direct or
  =I=R1, J=P2 indirect specification of I and J.
- <IKP number>
  =IKP1 ... IKP32
  =IKP=R1, P2 With R parameter contents of between 1 and 32.
  The IKA link branch with the specified control curve number is switched on/over through the assignment of an IKP number of between IKP1 and IKP32. The number specified for IKA need not correspond to that specified for IKP. The appropriate T parameter of the %IKA1 data field is always overwritten. This function also allows the branch to be switched over on a block-related basis from one control curve to another which may have been reloaded in parallel: See Section headed Data Handling.

<output quantity>

=X, FM, R, ... The specified output quantity must correspond to that specified for G411.

Notes:

- A **Switch IKA link branch on/over** operation must be preceded by a **Define IKA configuration** operation.
- An alarm is output if the input and output quantities specified between G411 and G412 in relation to an IKA number are not consistent.

8.13.3 Switch IKA link branch off (G410)

An IKA link branch which has been configured via G411 and swiched on via G412 with specification of a control curve IKP number can be switched off again by means of G410.

The characteristics of the link branch are defined through specification of the output quantitiy (see description under G412 for further details).

In addition, start values (positions) and an overtravel direction can be programmed for link switch-off, i.e. when G410 is programmed with start values (positions), the link is not switched off until the start value (position) of the quantity applied to input A is reached or exceeded.

Program syntax:

```
G410  <input quantity A{start position}{overtravel direction of input quantity A}
      <IKA number>
      <output quantity>
```

<input quantity A{start position}{overtravel direction of input quantity A}

- <input quantity A>
  =X1=, @441 <value>
  Direct or indirect specification of axis name. Analogous to GI programming, the link type (scanning of specified quantity of this axis) must be specified when axes are interconnected.
  =Rxx, Pzz Specification of an R parameter of which contents are applied to input A.

- <input quantity A{<start position>}>
  It is possible to specify that switch-off will take place only when the start position is reached. It is permissible to specify the start position as an option for axes acting as input quantities (leading axes).
  =X100 Specification of start position 100 for the input quantity, axis X.
  =@440 <value 3><value>
  <value 3> Corresponds to global axis number.
  <value> Corresponds to start position.
  By programming command @440, it is possible to indirectly specify the input quantity with start position (for use with cycles).

- <overtravel direction of input quantity A>
  An overtravel direction can be specified as an option. The overtravel direction remains irrelevant if no direction is programmed. An overtravel direction can be programmed only in conjunction with G412 with start positions and axes as input quantities.
  =K-1, K=Rxx With K-1 or K=Rxx, where Rxx is set to -1, a negative overtravel**/** approach direction is specified as the switch-off condition.
  =K1, K=Rxx With K1 or K=Rxx, where Rxx is set to +1, a positive overtravel**/** approach direction is specified as the switch-off condition.
  =K, K0, K=Rxx With K, K0 or K=Rxx, where Rxx is set to 0, no overtravel**/**approach direction is defined.

<IKA number>

=IKA1 ... IKA32
=IKA=R3, P4

When the IKA number is specified, the appropriate configuration defined by means of G411 is applied. The input quantity B has already been specified by means of G411 and must not therefore be programmed again for G412.

<output quantity>

=X, FM, R, ...

The specified output quantity must correspond to that specified for G411.

Note:

An alarm is output if the input and output quantities specified between G411 and G412 in relation to an IKA number are not consistent.

8.13.4 Examples of IKA link branches

```
G411  U  K21  X  K31  IKA3  R700
```

The axis setpoint (K21) position of U is connected to output A of IKA3 and the axis setpoint (K31) position of X to output B. The output value is stored in R700.

```
G411  R700  IKA4  FM[Z]
```

IKA4 is defined via R700 as the IKA to follow IKA3. The weighting input of IKA4 is constant and corresponds to the T parameter from the %IKA1 data. The output weights the axial Z feedrate as a multiplicative function.

(LINK_ON**/**OVER)

```
G412  U  IKA3  IKP9  R700
```

Switch-on of IKA3 with reference to control curve 9.

```
G412  R700  IKA4  IKP4  FM[Z]
```

Switch-on of following IKA4 with reference to control curve 4. The user must provide for the initial setting of R700 (=0).

```
G411  X  K21  R700  IKA4  FM3
```

(LINK_ON**/**OVER)

```
G412  X100  IKA4  IKP6  FM3
```

IKA LINK_ON**/**OVER
In link branch X IKA4 FM3 defined via G411 in which control curve 6 is assigned to start value X100. The branch is not switched over to control curve 6 until axis X overtravels start value 100.

(LINK_OFF)

G410 X100 K1 IKA4 FM3

IKA LINK_OFF
In link branch X IKA4 FM3 defined via G411 which does not effect any further feedrate weighting from start value X100 onwards. IKA LINK_OFF does not take place until axis X has travelled over start value 100 in the positive direction of motion.

G410 X0 IKA4 FM3

IKA LINK_OFF
In link branch X IKA4 FM3 defined via G411 which does not effect any further feedrate weighting from start value X0 onwards. IKA LINK_OFF does not take place until axis X travels over start value 0.

G410 X IKA4 FM3

IKA LINK_OFF
In link branch X IKA4 FM3 defined via G411 which immediately stops effecting any feedrate weighting. IKA LINK_OFF is implemented immediately. The difference between the programming of X and X0 is that X0 is interpreted as a position. X does not therefore correspond to X0 in relation to the programming of G400/G402/G410/G412.

Example for calculating compensation curves

```
                                ( IKA example 2 )
                                Machining of a contour with IKA as far as possible
                                with G commands. See also IKA example 1 [ description
                                of cycle commands / @40c ]
                                Caution: This example does not take the tool offset
                                into account!
                                0. Preparation :
N0001  @40c  K11  K2  K0           - Deactivate IKA 2
N0002  G0  X300  Y200              - Approach tool change point
N0003  R30=0  R31=0                - Error ID = 0

                                1. Structure of the table [ika3 data] :
N0005  @40c  K7  K1  K0            - Input variable 1..13 :
N0010  @40c  K7  K2  K30000           Angle 0, 30, ... , 360 degrees
N0015  @40c  K7  K3  K60000           in units of 10**[-3] degrees
N0020  @40c  K7  K4  K90000
N0025  @40c  K7  K5  K120000
N0030  @40c  K7  K6  K150000
N0035  @40c  K7  K7  K180000
N0040  @40c  K7  K8  K210000
N0045  @40c  K7  K9  K240000
N0050  @40c  K7  K10  K270000
N0055  @40c  K7  K11  K300000
N0060  @40c  K7  K12  K330000
N0065  @40c  K7  K13  K360000

N0070  @40c  K8  K1  K0            - Output variables 1..13 :
N0075  @40c  K8  K2  K5000            Sine [0], ... , sine [360 degrees]
N0080  @40c  K8  K3  K8660            in units of 10**[-4]
N0085  @40c  K8  K4  K10000           SIN=1 -> 10000=10 mm
N0090  @40c  K8  K5  K8660
N0095  @40c  K8  K6  K5000
N0100  @40c  K8  K7  K0
N0105  @40c  K8  K8  K-5000
```



```
N0110  @40c  K8  K9  K-8660
N0115  @40c  K8  K10  K-10000
N0120  @40c  K8  K11  K-8660
N0125  @40c  K8  K12  K-5000
N0130  @40c  K8  K13  K0

                                  2. Start and end pointer [ika2 data]:
N0135  @40c  K5  K1  K1              - Curve 1 uses points 1...13
N0140  @40c  K6  K1  K13

N0145  @40c  K55  K1  K-1            - Calculate curve 1
N0146  @30c  R30  K55  K1            - [a] read error byte [> R30]
N0147  @111  R30  K0  K150           - [b] case statement forR30:
K1  K281                                    Jump list for R30=1 .. 4 [error]
K2  K282                                    otherwise continue [R30=0] at N150;
K3  K283                                    scan [a] is repeated as long as
K4  K284                                    R30=-1 or R30=-127
K127  K-146
K-1  K-146

                                  3. IKA configuration (ika1 data) :

N0150  G411  IKA2                    - Delete IKA 2 : necessary before a reconfigu-
                                       ration with G411;; see N165.
                                       This block sets all ika1 parameters
                                       T0 .. T4, T15 .. T20, T25, T26 und T30 .. T34
                                       to standard values !
                                       @40c Kx..
                                       with x=11-13, 40-43, 1-4, 15-20, 25, 26, 30-34
                                       must therefore be programmed after this block.
N0151  @40c  K1  K2  K1              - IKA 2 uses curve 1
N0155  @40c  K40  K2  K1             - Activate extended IKA
N0160  @40c  K43  K2  K1             - Cubic interpolation on
                                     - Define line structure of inputs/outputs :
                                       Possible combinations and link types
                                                    InputA   Output B   Output
                                       Axis[setpoint]  K21      K31        K41
                                       Axis[actual]    K22      K32         -
                                       R parameter      o        o          o
                                       Compensation     -        -         K46
                                       Legend: K... possible if link type is defined
                                               o    possible without defining link type
                                               -    not permissible
                                                Configuration required here :
N0165  G411  X  K21  R20  IKA2  Y  K41       X actual  ->    [A]-IKA2   ->   Y-set
                                                R20       ->    [B]-IKA2

                                     - Limitations :
N0185  @40c  K31  K2  K500000            Maximum traversing range +-500 mm
N0190  @40c  K32  K2  K-500000

N0195  @40c  K34  K2  K2000               Modification limitation : 2000 units/IPO cycle
                                       with IPO cycle of 16 ms: 7500 mm/min
                                     - Weighting E/A :
N0200  @40c  K18  K2  K5                 Weighting input: 5/7
N0205  @40c  K19  K2  K7
```

```
                                    - Modulo function and shift :
N0220  @40c  K16  K2  K360000           Weighted input A modulo 360
N0225  @40c  K15  K2  K-90000           Shift input by -90 degrees
                                        to obtain a cosine
                                        therefore Y = Y0+16/3 * SIN[5/7*X-90]
                                                    = Y0+16/3 * COS[5/7*X]

                                 4. Approach, activation and machining :
N0235  G0  X252  Y100               - Approach starting point : 7/5*180=252
@714
N0236  R20=1000                     - Define input B-=R20

                                    - Activate IKA link branch :
                                      IKP1 [=curve1] wis used
                                                Input X
                                                Output Y
N0240  G412  X  IKA2  I16  J3000  IKP1  Y     Link factor
                                                16/3=16/3000*1000
N0241  G4  X1.8                     - Wait until start point is reached
N0245  G1  X0  F1000                - Machining
@714

                                   5. Deactivate and retract
N0250  @40c  K11  K2  K0               - Deactivate IKA 2
@714
G200  Y                                - Synchronization of actual value system
N0255  G0  Y200                        - Retract
N0260  X300
N0270  @100  K300                        Jump to end

                                   6. Errors
N0281  M00  (error 1)                  T55=1: only one pointer <> 0 !
@100  K300
N0282  M00  (error 2)                  T55=2: End pointer <= start pointer!
@100  K300
N0283  M00  (error 3)                  T55=3: Input[n+1] <= input[n] !
N0285  @30c  R31  K56  K1              - Read current point number n [> R31]
@100  K300
N0284  M00  (error 4)                  T55=4: Gradient > 1 !
N0285  @30c  R31  K56  K1              - Read current point number n [> R31]

                                   7. End
N0300  M02
```


8.14 Coordinate transformation TRANSMIT (G131, G231, G331)

The "Coordinate transformation TRANSMIT" function is an option.

The G functions for Transmit are **modal**.

The coordinate transformation TRANSMIT is applied in end face milling of turned parts. A **fictitious** (cartesian) coordinate system is used for programming whilst machine movements are effected in the **real** machine coordinate system. Special fictitious axes must be defined for the fictitious coordinate system.

The name of the fictitious axes and their position in the system can be selected freely.

A fictitious axis can be traversed only with the transformation selected.

Each transformation is defined by a transformation data block using machine data. The block contains the axes used in transformation, transformation type, channel number and the corresponding G function.

Three different TRANSMIT transformation data blocks can be defined with the G functions G131, G231 and G331.

TRANSMIT is selected by G functions G131, G231 or G331.The selection block must not contain any traversing movements or additional functions. A transformation may be activated from the reset position only, i. e. transition to another transformation is only possible after a deselection block has been applied.

Deselection is effected either by the deselection block (G130, G230, G330) in the AUTOMATIC mode or via the command channel. There is no fixed allocation between selecting and deselecting functions. G130 can also be used to deselect G331, for example.

Machine data is used to define whether the transformation is deselected on RESET or whether the transformation state remains effective.

Machine data also defines a different transformation action on change of mode.

While TRANSMIT is active, no zero offsets in

real axes are taken into account!

Programming of fictitious axes is not permissible in the initial setting (G130, G230, G330). When transformation is selected, no real axes used for transformation may be programmed.

Each selection**/**deselection of transformations requires the function “Empty buffer” (@714). This function is initiated internally by the block preparation.

Notes:

- Up to SW 5.3, all axes must be programmed once with G90 after selection/deselection of transformations.
- With SW 5.4 and higher, the axes can be programmed immediately with G91 if no scale factor/coordinate system rotation is active.

Before activating the transformation, the tool radius compensation must be deselected. A transformation is always assigned **permanently** to a channel.

The plane definition in the channel-specific machine data applies to the real system. On selection of a transformation, a plane is set for the fictitious system.

The fictitious plane is defined in the transformation data by assigning fictitious axes. Deviations from the basic planes can be programmed with G16. With TRANSMIT, the initial plane setting is defined as G17. G17 is set automatically after transformation has been selected. After its deselection, an automatic reset is made to the plane which was effective before transformation selection. Parallel transformations in different channels must not act on the same axes.

The transformation must not be selected or deselected in a contour block sequence.

In the blocks after selection and deselection of the transformation absolute dimensioning (G90) must be programmed for the axes concerned.

If a block search with calculation is used in the program the following must be observed:

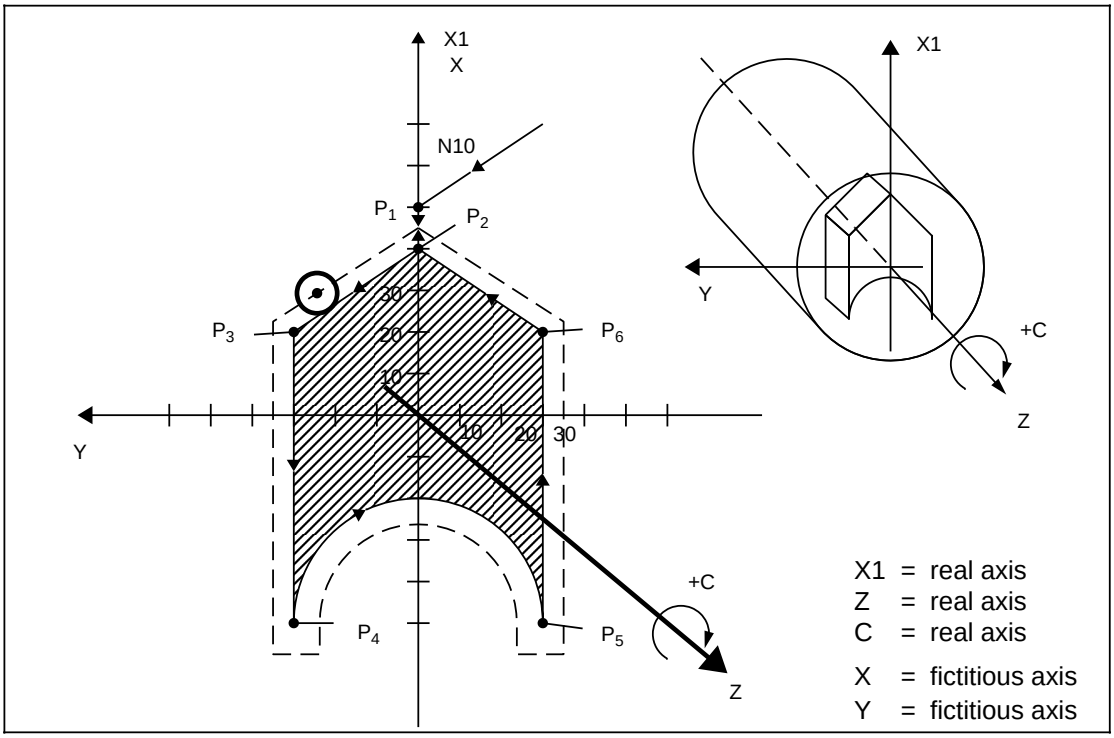
- The real axes involved in the transformation must not be in a position which corresponds to the origin of the fictitious coordinate system when the block search is initiated.
- The path from the position before the start of the block search to the final position of the block to be searched for must not lead through the original coordinates.

Notes:

- With TRANSMIT the programmed path speed is reduced such that the maximum speed of rotation of the rotary axis is not exceeded. This is especially the case for movements near the turning centre.

  In rapid traverse the speed is also reduced if the rotary axis is not involved in the movement.
- In JOG the feedrate reduction can only partially take effect, because the final position of the movement is not known. In this case, drops in feedrate might occur.
- When starting a block search you must ensure that the tool tip is situated on the same side of the transformation centre as it is during machining.
- If an angle head cutter is used (type 30), a length is included in the transformation. It may be necessary to program a freely definable plane (G16) after transformation has been selected.

Example: End face machining withTRANSMIT



End face machining with TRANSMIT

Example 1: TRANSMIT with angle head cutter, length 1 in Z, length 2 in X1=

N05 G16 Z X1= Z X1=		For angle head cutter: G16 plane
N10 G01 F5000 X1=50 C0 Z0 D1	(P1)	Real system: Approach starting point For angle head cutter: Traverse the tool length compensation
N20 G131		Select transformation G17 (X Y Z) are automatically active after selection
N25 G01 F1000 X40 Y0 G42	(P2)	
N30 X20 Y30	(P3)	
N35 X-50	(P4)	
N40 G02 X-50 Y–30 I0 J–30	(P5)	
N45 G01 X20	(P6)	
N50 X40 Y0	(P2)	
N55 G40 X50	(P1)	
N60 G130		Deselect transformation
N65 G00 X1=60		Real system: retraction

Example 2: TRANSMIT with angle head cutter, length 1 in X1=, length 2 in Z

N05 G16 X1= Z X1= Z		For angle head cutter: G16 plane
N10 G01 F5000 X1=50 C0 Z0 D1	(P1)	Real system: Approach starting point For angle head cutter: Traverse the tool length compensation
N20 G131		Select transformation G17 (X Y Z) are automatically active after selection
N22 G16 X Y X1= Z		G16 plane within transformation
N25 G01 F1000 X40 Y0 G42	(P2)	
N30 X20 Y30	(P3)	
N35 X-50	(P4)	
N40 G02 X-50 Y–30 I0 J–30	(P5)	
N45 G01 X20	(P6)	
N50 X40 Y0	(P2)	
N55 G40 X50	(P1)	
N60 G130		Deselect transformation
N62 G16 X1= Z X1= Z		Real system: Restore G 16 plane
N65 G00 X1=60		Real system: Retraction

Traversing through the transformation centre 1)

With this function it is possible to program any motion blocks which result in a traversing movement through the transformation centre. The resulting path velocity is always automatically matched depending on the distance from the centre of rotation. The contour is always traversed at the maximum possible feedrate.

1) Software version 5 and higher

8.15 Coordinate transformation 2D (G133, G233, G333) and 3D (G135, G235, G335)

The "Coordinate transformation 2D/3D" function is an option.

The G functions 2D/3D are **modal**. The transformation can be programmed two-dimensional (2D) or three-dimensional (3D).

The programming is done in a **fictitious** (Cartesian) coordinate system. The machine moves in a **real** machine coordinate system.

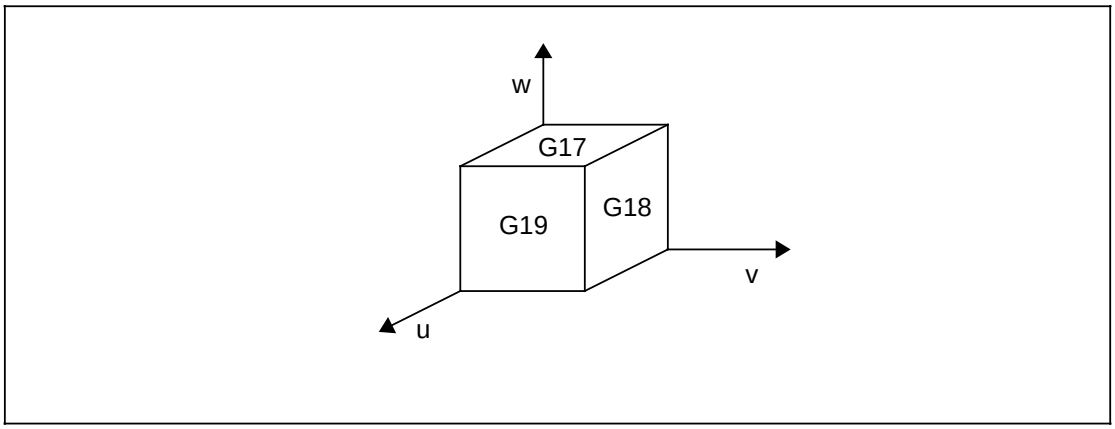
Special fictitious axes must be defined for the fictitious coordinate system. These can only be traversed is transformation is selected.

Fictitious axes are freely selectable in the system as regards their axis name and their position. A fictitious axis can only be traversed if transformation has been selected.

Every transformation is defined by a transformation data block via machine data. This block contains the axes involved in the transformation, the type of transformation, the channel number, the offset coordinates, the angle of rotation and the associated G function.

A transformation is always **permanently** assigned to one channel. Up to three transformations (G133, G233 and G333 for 2D transformation and G135, G235 and G335 for 3D transformation) can be defined in one channel. Hence, it is possible to call up to three different coordinate transformations in one channel.

The plane definition specified in the channel specific machine data applies to the real system. When selecting a transformation, one plane is set for the fictitious system. The fictitious plane is defined in the transformation data by assigning the fictitious axes. Any deviations from the basic planes can be programmed via G16.



Plane selection for the fictitious system

Return to the plane effective prior to selection of transformation is also effected automatically after deselection of transformation.

The transformation is selected by the G functions G133, G233, G333 (2D) or G135, G235, G335 (3D).

The selection block must not incorporate any traversing movements nor further functions. A transformation may only be activated from the reset position, i.e. transition to another transformation is possible only via a preceding deselection block. Deselection is effected by the deselection block (G130, G230, G330) in AUTOMATIC mode or via the command channel.

A machine data defines whether the transformation is deselected at RESET or whether the transformation status is retained. In addition, the machine data defines a different transformation behaviour on mode change.

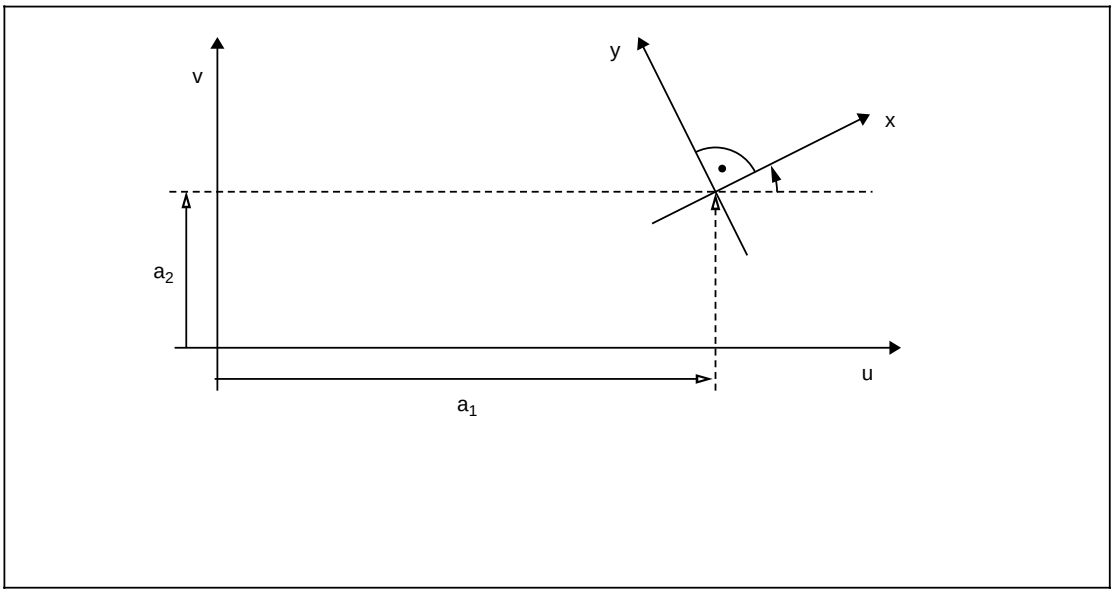
As long as 2D/3D coordinate transformation is active, no zero offsets are taken into account in real axes.

Two-dimensional coordinate transformation

Transformation parameter und constant:

x, y real coordinate system
u, v fictitious coordinate system
a1: offset of the real system in v direction relative to the fictitious zero.
a2: offset of the real system in v direction relative to the fictitious zero.
: angle of rotation of the real system relative to the fictitious system.

Transformation equations:
u = a1+x cos – y sin
v = a2+x sin +y cos



2D G133, G233 or G333 transformation

Three-dimensional coordinate transformation

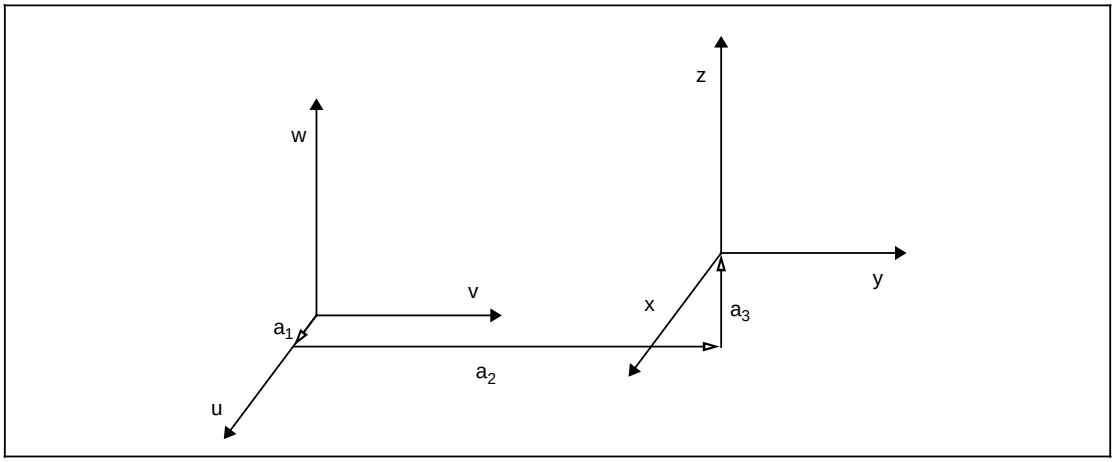
The three-dimensional coordinate transformation is composed of a translation and a rotation of the real coordinate system about its axes

Transformation equations:

u = a1+x(cos cos) – y(cos sin)+z sin
v = a2+x(cos sin +sin sin cos)+y(cos cos – sin sin sin) - z(sin cos)
w = a3+x(sin sin – cos sin cos)+y(sin cos +cos sin sin)+z(cos cos)

Transformation parameter and constant:

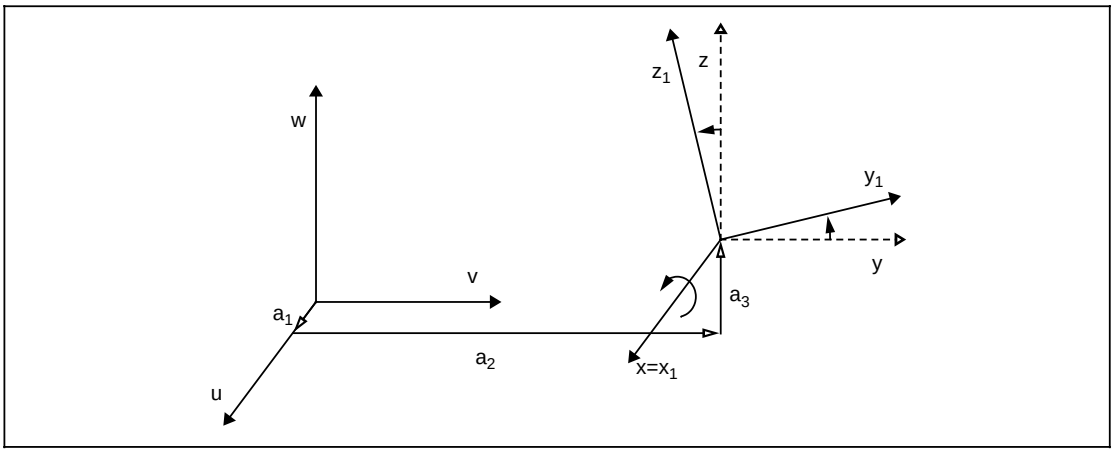
a_1: shift of the real system in the u direction relative to the zero of the fictitious coordinate system.
a_2: shift of the real system in the v direction relative to the zero of the fictitious coordinate system.
a_3: shift of the real system in the w direction relative to the zero of the fictitious coordinate system.
: angle of rotation of the real system about the x axis
: angle of rotation of the real system about the y axis
: angle of rotation of the real system about the z axis
x, y, z: real coordinate system
u, v, w: fictitious coordinate system



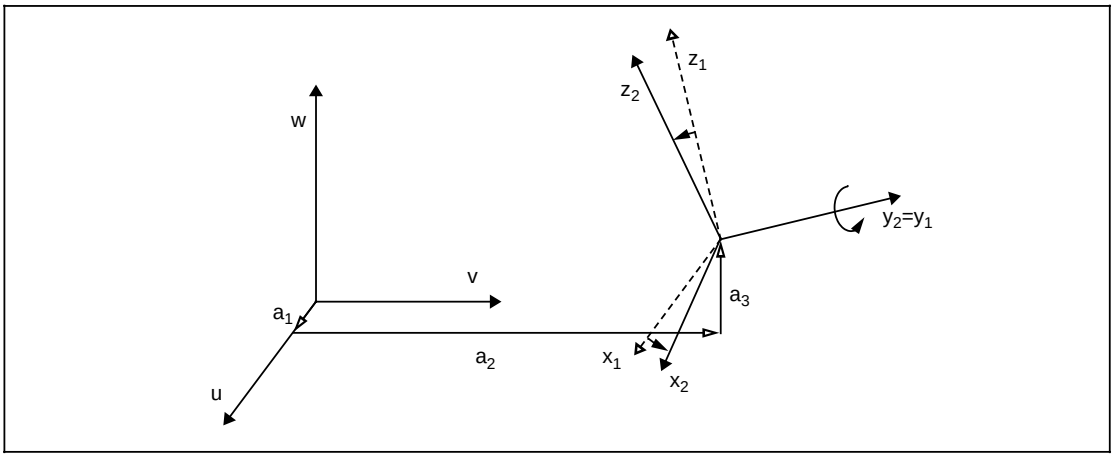
Translation: shift of the real coordinate system (x, y, z) relative to the fictitious coordinate system (u, v, w)

Note:

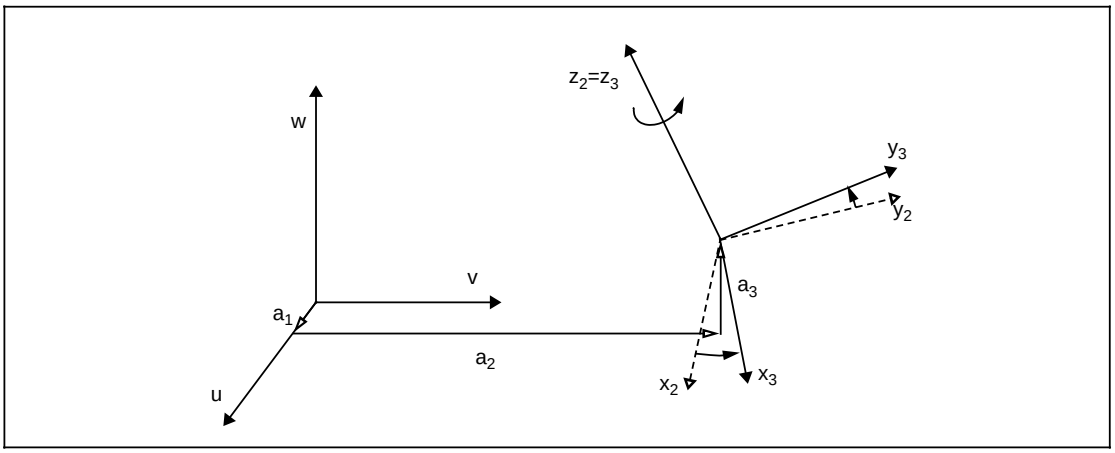
In the following three figures the coordinates of the real coordinate system (x, y, z) are only shown with indices (e.g. x_1, x_2, x_3) for purposes of demonstration. These indices have no bearing on the transformation equations given.



Rotation – first step: rotation of the real coordinate system (x, y, z) about x axis of the real coordinate system (x_1, y_1, z_1)



Rotation – second step: rotation of the real coordinate system (x_1, y_1, z_1) about y axis of the real coordinate system (x_2, y_2, z_2)

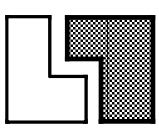


Rotation – third step: rotation of the real coordinate system (x_2, y_2, z_2) about z axis of the real coordinate system (x_3, y_3, z_3)

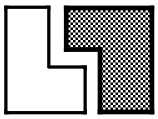
Notes:

- No fictitious axes can be programmed in initial position (G130, G230, G330) (otherwise alarm No. 2043 is output).
- A transformation can only be activated from the initial setting, i.e. a transition to another transformation is only possible with a previous selection block.
- If transformation is selected no real axis involved in the transformation can be programmed (otherwise alarm No. 2043 is output).
- If a real axis of the transformation grouping is switched from follow-up mode to position control, an automatic reverse conversion of the fictitious coordination is performed.
- If a ”feed hold” is applied to a real axis of the transformation grouping, it applies to the whole grouping.
- While the transformation is active the zero offset for the real axes assigned to the fictitious axes has no effect.
- The function ”DRF handwheel” is not possible for fictitious axes.
- The settable angle of rotation must always be zero and the programmable angle of rotation (G58**/**G59) must be zero on selection/deselection of the transformation.
- Every selection/deselection of the transformation is associated with the function ”empty buffers” (@714). @714 need not be programmed as it is triggered internally by block preparation.
- The tool radius compensation must be deselected before activation of the transformation (because of @714).
- Transformations running parallel in different channels must not access the same axes.
- The transformation must not be selected or deselected within a contour block sequence.
- A block search with calculation to a program part in which the transformation is active is permitted.
- After selection**/**deselection of the transformation all axes must be programmed with G90 at least once if offset values have been included (zero offset, tool offset ...).
- Only one transformation can ever be active in a channel. Only three data records are possible per channel and per transformation.

8.16 Coupled motion of axes (G150 ... G159)

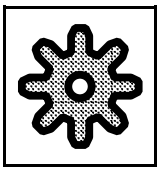


The ”Coupled motion of axes” function is an Option.

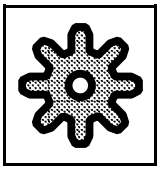


Using the “Coupled motion” function, any axis of the control can be declared a “leading axis”, and it can be assigned up to four axes as coupled axes. All axes together form a coupled-axis grouping. If the “leading axis” is programmed in a single block of a part program, all coupled axes travel the same programmed traversing path.

Up to 8 axes can be declared coupled axes. A single axis can be used in different coupled axis groupings at the same time.



The axes of the coupled-axis grouping are defined using machine data.



A total of 9 coupled axis combinations are available. They are addressed in the part program by G151 to G159. The G functions are **modal**.

Display of the actual position and of the set/actual difference is constantly updated on the screen for all axes in a coupled axis grouping.

The “coupled motion” function is deselected with G150.

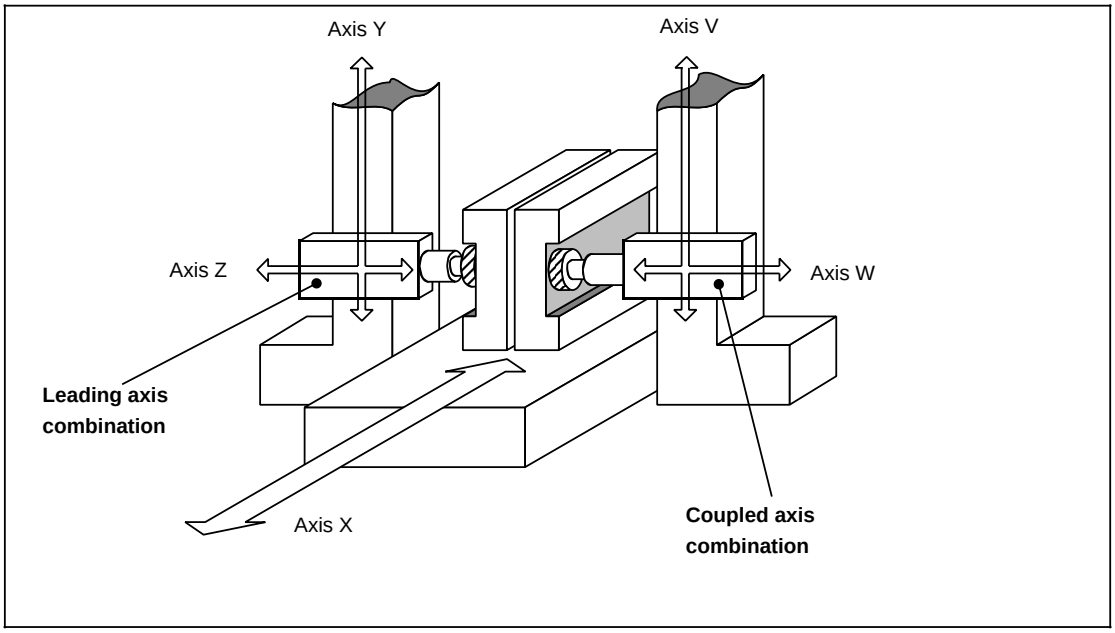
After G150 the coupled axis must be programmed once with G90 (absolute dimensions) if offset values have been included.

Control of the ”Coupled motion” function via command channel

The “Coupled motion” function can also be controlled from the PLC via the command channel. When “Coupled motion” is selected, a new synchronization cannot be performed and the G function is not displayed.

If an axis which was previously a coupled-motion axis or real axis is to be programmed via the command channel in the part program following deselection, then G200 (axis actual value synchronization) must be programmed.

Example:



8.17 Approaching machine fixed points

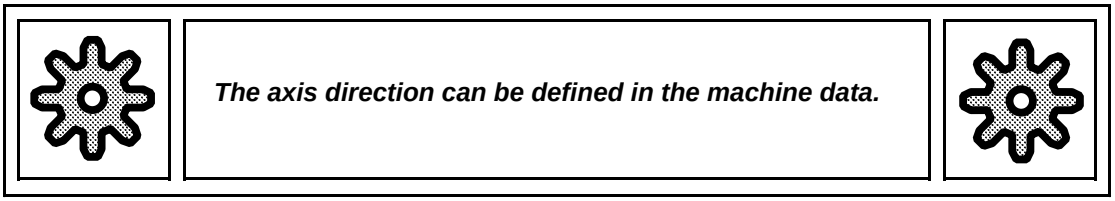
8.17.1 Reference point approach with synchronization by program (G74)

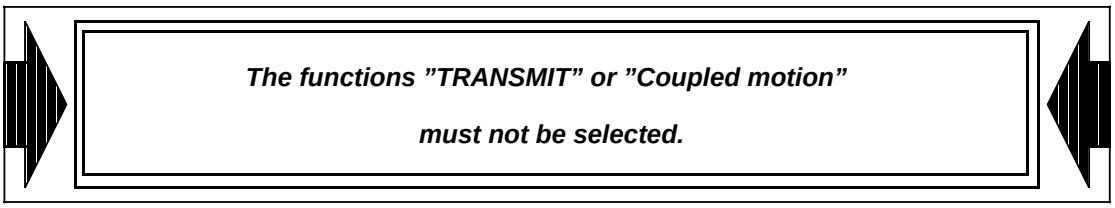
The function “Reference point approach with synchronization by program” can be used to approach the reference point from a part program with real axes. G74 is non modal. Block change is performed when all the programmed axes have been referenced.

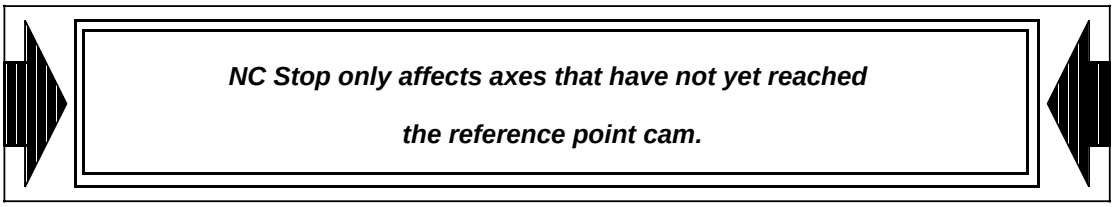
Up to software version 4 only one axis can be programmed with G74. With software version 5 and higher up to 5 axes can be programmed with G74.

Example:

```
N10 G74 X Y Z LF
```







Note:

The axis can be programmed directly with G90 or G91 in the block after G74. The axis must be programmed with G90 if it includes an active coordinate rotation or scale factor. By programming G74 with G91 and G53, the zero offset is allowed for in the specified axis when programmed after the G74 block.

G74 is linked to an “empty buffer” (@714) . This function is triggered internally by block preparation.

G74 triggers approaching with rapid traverse.

8.17.1.1 Setting reference dimension (G75)

Using the "Setting reference dimension " function, the actual value of the axes (machine actual value) can be changed in the running part program.
The "Setting reference dimension" operates as a combination of the "Setting reference dimension via PLC request" and the "Axis actual value synchronization " (G200) functions. The new actual value is defined by programming. The interface signal "Reference point reached" is set at the same time.

Program syntax:

G75 X123.456 Y 123.456

Up to 5 axes can be programmed in one block. The command must stand alone in the block. Only one more block number, a comment and G53 (see G74 for effects) may be additionally programmed.

The programmed actual value is taken for the machine actual value, the previous actual value is lost. A possibly pending PRESET and DRF shift becomes effective in the display immediately . Only actual values between 0 and 360 can be preset for rotary axes.

The actual value set is accepted by the channel immediately. A continuation using G91 is also possible (see Reference point approach function G74).

The G command (G75) is located in the 7th G group:

Restrictions for "Setting actual value"

(Background: Carrying out "Setting actual value" using "Setting reference dimension"):
– Only possible for axes without absolute encoders
– Not possible for spindles
– MD 1824* bit 5 "Setting of reference dimension permissible" must be set to 1.

For the programmed axes, the function must be enabled with "Setting reference dimension permissible" machine data. Absolute encoders are not permmissible for these axes. Otherwise the 3003 alarm "invalid address programmed" is displayed.

8.17.2 Travel to fixed stop (G220 ... G222)

The "Travel to fixed stop" function is required for axes and spindles. With this function it is possible to build up defined forces required for clamping workpieces. This is needed especially for tailstocks, sleeves, grippers etc.

8.17.2.1 Selecting the function "Travel to fixed stop" (G221)

The function is called with G221 and can be programmed with real axes. Spindles must first have been switched to C axis mode. Axis names can also be programmed in extended addressing.

The function can only ever be selected for one axis per block. The only other commands allowed in the selection block in addition to the syntax shown below are G90**/**G91.

The axis converter is also active with the "Travel to fixed stop" function.

Program syntax:

```
G221 <axis name> <target position> <feedrate> {result parameter} {clamping torque P}
```

Example:

```
G221 X250 F100 R120 P200
```

P200 — Clamping torque in 0.1 % of the maximum torque

R120 — The function acknowledgement for the "Travel to fixed stop" function selection is stored in R parameter 120.
0 fixed stop has not been reached
0 fixed stop has been reached

If the R parameter is not assigned and the fixed stop is not reached, Reset alarm 1220* is triggered.
The contents of the parameter give the position of the fixed stop

F100 — Feedrate at which the fixed stop is to be approached. The fixed stop is approached with G01.

X250 — The X axis is to reach the fixed stop at position X220. The fixed stop must lie between the starting position and the target position of the selection block.

G221 — Select the function. G221 must always be programmed at the beginning of the block.

Notes:

- The axis name must be programmed.
- If no feedrate is programmed, the feedrate active when the function is selected is used.
- The value P is stored in the setting data.

8.17.2.2 Deselecting the function "Travel to fixed stop" (G220)

The function is deselected with G220 and can be programmed for real axes.

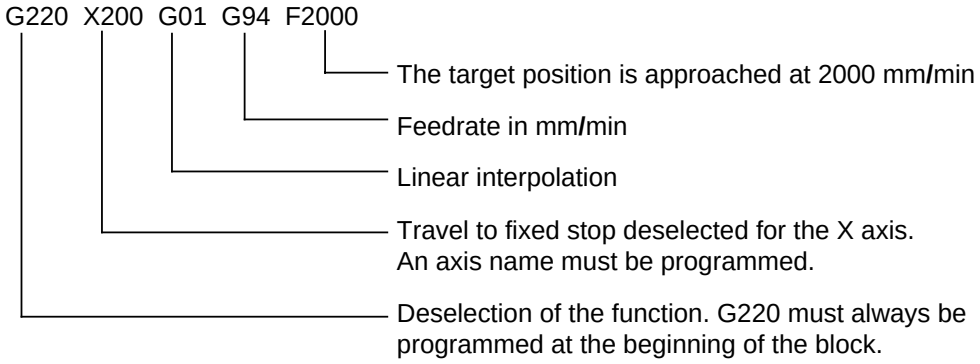
It is only possible to deselect one axis per block. The only other commands allowed in this block in addition to the syntax shown below are G00/G01/G90/G91/G94/G95.

The axis converter is active. Extended addressing can be used.

Program syntax:

```
G220 <G90/G91><axis name> <retract position> <G00/G01> <G94/G95> <feedrate>
```

Example:



Notes:

- The axis name must be programmed.
- If no feedrate is programmed, the axis traverses to the target position at the feedrate active before the function was deselected.
- If no G00/G01 is programmed, the retreat position is approached with the interpolation type (G00/G01) active before the function was deselected.
- If no G94 is programmed, the retreat position is approached with the feedrate active before the function was deselected (e.g. G95).
- If the function is deselected for an axis which is not positioned at the fixed stop, the deselection block behaves like a travel block; the axis traverses to the programmed target position.
- In the following part program (block N30), the deselection position offset by a drift can be approached.

```
N10  G220 X0  F1000
N20  G221  X10  F100
N30  X0  F1000
```

- Up to SW 5.3, all axes must be traversed with G90 after deselection of the function "Travel to fixed stop".
  If any axis is to be traversed with G91, G200 (axis actual value synchronization) must be programmed after deselection of the function "Travel to fixed stop". With SW 5.4 and higher, the axis can be programmed immediately with G91 if no scale factor/ coordinate system rotation is active for the axis.

8.17.2.3 Changing the clamping torque (G222)

The clamping torque can be changed with G222. The function must be programmed alone in a block. The axis converter is active. Extended addressing can be used.

Program syntax:

```
G222 <axis name> <clamping torque P>
```

Example

```
G222  Z  P120
```

- P120 — Clamping torque 12 % of the max. torque
- Z — The clamping torque is changed for the Z axis
- G222 — Changes the clamping torque. G222 must be programmed at the beginning of the block.

8.18 Additional information on movement

8.18.1 Dwell (G04)

Dwell times are required for backing off, and sometimes for speed changes and machine switching functions (steady, tailstock, etc.).

The action of G04 is non-modal.

No other functions may be written in a block containing G04.

The dwell is entered under address X, F or S. The time range lies between

0.001 and 99999.999 s for X, F
0.1 and 99.9 revolutions for S

Examples:

`N.. G04 X11.5 LF`	Dwell 11.5 s
`N.. G04 S50 LF`	Dwell 50 spindle revolutions

8.18.2 Dwell time in relation to an axis/spindle (G14/G24)

In addition to the options for programming the dwell time in seconds or spindle revolutions, an incremental dimension can be specified with G24 which acts as the dwell until the axis/spindle has traversed the specified value.

Programming:

G14**/**G24 G90**/**G91 <axis name**/**spindle name> <dwell position**/**dwell path> {overtravel direction}

G14 The dwell refers to the setpoint of the axis**/**spindle

G24 The dwell refers to the actual value of the axis**/**spindle

G90 Dwell is active until the programmed axis**/**spindle has reached the programmed dwell position.

G91 Dwell is active until the programmed axis**/**spindle has traversed the programmed dwell path.

Spindle name

S : Leading spindle
S1 to S6 : Any spindle

Dwell position

Position in mm**/**inch**/**degrees that the programmed axis**/**spindle must reach (only active with G90) before the part program can continue. For rotary axes and spindles this position is within one revolution (0 to 360 degrees).

---

1) Software version 4 and higher, G24 on axes with GT91 (incremental dimension), with software version 5 and higher G14 can be programmed.

Dwell path

Path in mm**/**inch**/**degrees that the programmed axis**/**spindle must travel (only if G91 is active) before the part program can be continued.

Overtravel direction

K0 : Overtravel direction either way (initial setting if no overtravel direction is programmed).

K+**/**-1 : Overtravel direction positive**/**negative.
Significance with G90: The specified position must be passed in the stated direction.
Significance with G91: The programmed path is only included in the stated direction.

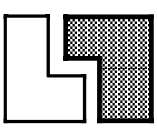
Examples:

```
G24  G91  C2=180        ; The dwell is applied in this channel until the C2=axis has
                          traversed the path of 180 degrees
G24  G90  S3=90  K1     ; The dwell is applied in this channel until the 3rd spindle has
                          passed the position 90 degrees in the positive direction.
```

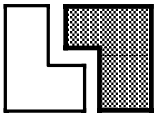
- ***No zero offsets or tool offsets are taken into account for these axes.***
- ***A dwell block with***
  - ***G04 X..., G04 F... is generally cancelled with NC Stop;***
  - ***..G04 S..., G14, G24 is cancelled with NC Stop in accordance with the machine data.***
- ***Any resulting dead times can be compensated for in the machine data.***

8.19 Additional functions

8.19.1 Programmable emergency retraction (G420...G426) 1)



The "Programmable emergency retraction" function is an option.



This function allows the machine to be shut down or the tool and workpiece to be separated in the event of a fault in order to prevent damage.

It is possible, for example, to program the operating characteristics by means of G commands:

G420 Deactivate (generally or selectively) programmable emergency retraction
G421 Activate programmable emergency retraction
G422 Configure generator operation
G423 Configure stop as open-loop control function
G424 Configure stop as autonomous drive function (only in conjunction with SIMODRIVE 611D)
G425 Configure retraction as open-loop control function
G426 Configure retraction as autonomous drive function (only in conjunction with SIMODRIVE 611D)

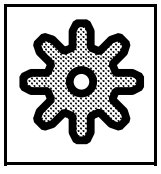
The "Programmable emergency retraction" defined through such programming measures is effective in all operating modes. After power on, the last programmed status is lost, i.e. all G functions must be programmed again on restart.

The G commands must be written alone and at the beginning of the block. A maximum of 5 axes can be addressed simultaneously; a maximum of 1 spindle per block can be programmed. If several G420 to G426 commands are programmed in succession, they will overwrite commands already programmed for axes/spindles.

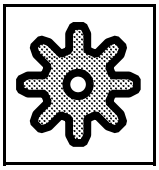
If the function is active, the axes/spindles may not be active in any other channel.

8.19.1.1 Deactivate "Programmable emergency retraction" G420

When G420 is programmed without further specifications, the operating characteristics relating to this function are deactivated for all axes and spindles as well as those parameterized via machine data for the channel.



As from SW 4.7/2, it is possible to set by MD whether G420 acts for all channels inlcuding PLC or only in the current channel.



When G420 is programmed with specification of axes and/or spindles, the operating characteristics relating to this function are deactivated for the axes/spindles programmed.

---

1) Software version 4 and higher

Program syntax:

```
G420  {axis name}  {axis address}  {spindle name}  {spindle address}
```

deactivates the operating characteristics programmed for the axis or spindle.

{axis name}
e.g.: X, Y, Z

{spindle name}
e.g.: S2, S1=...

{spindle address}

@443 <value>	The value corresponds to the global spindle number. The spindle can be specified indirectly by means of the @443 command (e.g. for cycles).
@443 R1	Allows indirect specification of spindle name which can be entered via K, R or P.
R1=0	Corresponds to specification of leading spindle of channel.
R1=1...6	Corresponds to specification of spindle S1 to S6.

8.19.1.2 Activate monitoring sources and enable reactions (G421)

When G421 is programmed without further specifications, the channel-specific monitoring sources parameterized via machine data and the channel-specific reactions (internal and external) are activated.

Program syntax:

```
G421 {axis name} {axis address} {spindle name} {spindle address}
```

This command activates the axis-specific and spindle-specific monitoring sources (internal) parameterized via machine data and enables the axis-specific and spindle-specific reactions (internal and external) in addition to the channel-specific monitoring sources/reactions for this axis/spindle.

{axis name**/**spindle name}
e.g.: X, S1=...

{axis address**/**spindle address}

@441 K1 @443 K2	The axis-specific and spindle-specific monitoring sources defined via machine data are activated and the axis-specific/spindle-specific reactions enabled both for the channel concerned and additionally – directly for axis X1 (X1=) and spindle S2 (S2=) or – indirectly for axis X1 (@441 K1)

Note:

Alarm 3260 is output if a position is programmed for an axis**/**spindle or if @440**/**2 is used.

8.19.1.3 Configure generator operation (G422)

Appropriate commands can be programmed to provide compensation for brief DC-link voltage dips.

Program syntax:

```
G422 {axis name} {axis address} {spindle name} {spindle address}
```

Generator operation is configured for the axes/spindles, which are addressed directly or indirectly via constants, R parameters or pointers, through specification of axis name, axis address, spindle name and spindle address.

C1= or S1=	Generator operation is configured for axis C1 or spindle S1.
K0 Rxy Pxy	The leading spindle of the channel is addressed by K0 or R parameter contents = 0. The axis with the global axis number 1...30 is addressed by K1 to K30 or R parameter contents = 1...30. The spindle with the global axis number 1 to 6 is addressed by K-1 to K-6 or R parameter contents = -1 to -6.

Note:

Alarm 3260 is output if the programmed axis/spindle does not exist.

8.19.1.4 Configure stop as open-loop control function (G423)

When "Stop" is configured as an open-loop control function, the interpolation grouping remains in operation for an adjustable period of time until controlled braking is implemented.

Program syntax:

```
G423  {axis name} {axis address} {spindle name} {spindle address}
```

When the axis name is specified directly or indirectly via @441 or when the spindle name is specified directly or indirectly via @443, the stop operation parameterized via machine data for the addressed axis**/**spindle is configured.

Z C1=	The stop operation parameterized via machine data is configured for axes Z and C1.

Note:

The "Extended stop" (machine data) is automatically applied to all active leading**/**following axes which have been programmed via G401 to G403 and these do not therefore need to be programmed with G423. Alarm 3260 is output if the specified axis**/**spindle does not exist.

8.19.1.5 Configure stop as autonomous drive function (G424)

In the event of a control system failure, the axes/spindle can be only be stopped on a modular basis. The drives within a previously linked GI grouping are then shut down as simultaneously as possible.

Program syntax:

```
G424 <TI>  {axis name} {axis address} {spindle name} {spindle address}
```

The "Stop" process is configured on a modular basis for the axes/spindle which are directly or indirectly addressed via constants, R parameters or pointers through specification of TI, axis name, axis address, spindle name and spindle address.

TI[C1=]=Rxx TI[C2=]=Rxx

The time programmed in Rxx for autonomous drive shutdown is configured for axes C1 and C2.

TI[S]=Rxy TI[S1=]=Rxy

The time programmed in Rxy for autonomous drive shutdown is configured for the leading spindle of the channel and for the spindle addressed via R1.

TI[K0]=Rxy TI[R1]=Rxy TI[P2]=Rxy

The axis or spindle can be indirectly addressed via K, R or P.

The axis with the global axis number 1...30 is addressed by K1 to K30 or R parameter contents = 1...30.

The spindle with the global axis number 1 to 6 is addressed by K-1 to K-6 or R parameter contents = -1 to -6.

The leading spindle of the channel is addressed by K0 or R parameter contents = 0.

Note:
Alarm 3260 is output if the addressed axis**/**spindle does not exist.

8.19.1.6 Controlled configuration of retraction (G425)

The "Retraction" function annuls the physical contact between tool and workpiece.

In the case of a retraction operation, the axes in the grouping can be positioned incrementally or absolutely.

Endlessly rotating rotary axes or spindles can only be positioned absolutely.

Program syntax:

```
G425  {Interpolation type}  {absolute dimension/incremental dimension/oriented stop}
<axis name/axis address incl. position/path>
```

The characteristics for the retraction of the interpolation grouping are defined by means of the interpolation type. Only G0 or G1 characteristics are permitted and the feedrate type G94**/**G95 must be active or else error message "Wrong block structure" will be output.

Note:

Feedrate type and feedrate value remain valid for the subsequent traversing blocks.

G90, G91, mixed G90**/**G91

The retraction operation for axes within the interpolation grouping is configured to be absolute, incremental or mixed.

G[...]119 P[...]Position {feedrate type} {feedrate value}

Configuration of the retraction of an endlessly rotating rotary axis (simultaneous axis) to a programmed position. The feedrate type is G94**/**G98.

M19 S.. position — Configuration of the retraction to a programmed position of the leading spindle of the channel or for the spindle specified via M19 S1 = to S6 =. The position must be programmed.

M19 @442 <value3> <value>

<value3> — Corresponds to global spindle number; the leading spindle of the channel is addressed with 0 and the appropriate spindle indirectly addressed with 1 to 6.

<value> — Corresponds to position specification with M19. Configuration of retraction to a programmed position for the leading spindle of the channel or for the spindle specified via M19 S1=.. to S6=.. .

{axis name**/**axis address incl. position**/**path}

X100 @440 Rxx Rxy — Configuration of retraction for directly (X100) or indirectly specified axes (@440 Rxx Rxy).

Individual acceleration for controlled emergency retraction (SW 6 and higher)

With machine data MD 376* an individual acceleration is specified enabling the emergency retraction block to be effective more quickly.

8.19.1.7 Configure retraction as autonomous drive function (G426)

This function defines the retraction message relating to incremental dimension value, traversing direction and speed setpoint.

Program syntax:

```
G426  Incremental dimension  Retraction axis  Retraction feedrate
```

<incremental dimension>	G91 must be specified.
<retraction axis> P[X1=]–10	The simultaneous axis syntax is used to define the retraction path, the traversing direction and the axis. The example shows the configuration of the retraction of the X1 by an incremental dimension value of 100 in the negative direction.
<> F[X1=]500	The simultaneous axis syntax is likewise used to define the retraction feedrate. In the example, the retraction of the X1 axis is configured with a feedrate in 500 mm**/**min (with active G[X1=]) 94: default setting).

Note:

Alarm 3260 is output if the specified or addressed axes do not exist.

8.19.1.8 Offsets in retraction block (G425/G426)

If a retraction block with active offsets (coordinate rotation, scale factor, zero offsets, tool length compensations, mirroring) is programmed, then these functions are also taken into account in the retraction block as a function of G90**/**G91.

G90 in retraction block

All offsets specified above are activated. An approach point is calculated for these offsets which is approached when the retraction block is activated regardless of which offsets are active at that instant.

If the user wishes to program the retraction point without active offsets, the offsets must be deselected. Coordinate rotation and zero offsets can be deselected in the retraction block with G53. Tool length compensations, scale factor and mirroring must be deselected before the retraction block.

The retraction block must be reprogrammed if offsets must be altered; these must then activated in the new retraction block.

G91 in retraction block

The calculation only takes account of coordinate rotations, scale factor and mirroring. The offsets must be deselected if the retraction point is to be programmed without active offsets.

Zero and tool offsets have no effect on G91 and can therefore be altered in any desired way after the retraction block.

The retraction block must be reprogrammed if offsets must be altered; these must then be activated in the new retraction block.

G90/G91 mixed in retraction block

A mixture of G90**/**G91 can also be programmed in the block (machine data). The characteristics described above for absolute and incremental dimensions apply in this case.

8.19.2 Axis and spindle converter

The axis and spindle converter assigns the axis and spindle names of a universally applicable part program to the axes and spindles defined on the machine.

This means that one and the same part program can be executed on machines with different axis names as well as on the same machine tool with different channels.

Axis converter

If the converter is active, axis addresses (and address extensions) in universally applicable part programs are converted to a different axis address (and address extension).

A maximum of 8 axes can be converted simultaneously in each NC channel.

Example:	X	X3
	Y4	Q14
	X3	X

The status of the axis converter can be written as active or non-active and the axis conversion list can be read or written via @ code.

Spindle converter

This function makes it possible to execute part programs on the machine on which the spindle names are not defined or which have a different designation.

A maximum of 3 spindles can be converted simultaneously in each NC channel.

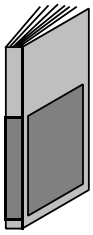
Example:	S	S4
	S1	S2
	S2	S5

The status of the spindle converter can be written as active or non active and the spindle conversion list can be read or written via @ code. The following block combination must be programmed in the part program so that the PLC and part program are synchronized:

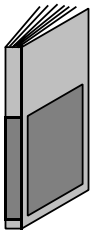
```
N10  Mxy  LF                    (Set read-in disable with Mxy)
N20  @714  LF                   (Stop block preparation and empty buffer)
N30  S1=500  M1=03  LF
```

If S1 = S2 is assigned in the spindle converter, the block is converted as follows:

S2=500 M2=03



The setting data are described in the documentation ”SINUMERIK 840C, Installation Instructions” Section 6.



8.19.3 Freezing of offsets (G175/G176), SW 1 and 2 only

Through the "Freezing of offsets" function the tool length compensations, zero offsets and angles of coordinate rotation are frozen thus reducing the block change times.

Activation of function:

G176: Freeze length compensations, zero offsets and angles of rotation of coordinate rotation
G175: Update length compensations, zero offsets and angles of rotation of coordinate rotation

Alarms:

3220: ”Change G176 – G175”
3006: ”Incorrect block structure”
– issued in the event of incorrect programming in conjunction with G176
3007: ”Error with setting data - programming error”

Behaviour with different G function/D function combinations

Condition: G176 active

G54 ... G57 **D ...** **G58/59 ... A** **G53**	Change to G175 alarm 3220
G176 G54 ... G57 **G176 D ...**	G176 effective. Freezing is effected.
G176 G58/59	Special effect owing to setting data G58/G59 alarm 3007
G176 G53	Change to G175 alarm 3006

Notes:

- While G176 is active, modifications to the "External zero offsets from the PLC" are not processed and do not invoke alarm displays. The are effective, however, after NC stop and renewed NC start.
- If G176 is active, the writing of tool offsets, angle of rotation, zero offset, DRF, PRESET and suppression of zero offset by means of the functions @706, @42x and @43x is not allowed. G175 becomes active and alarm 3220 is displayed.
- With software version 3 and higher freezing is automatically performed if the data concerned are not altered. This reduces the block change times.

When G176 is selected, the current zero offsets

are calculated and then frozen.

8.19.4 Rapid block change using FIFO function (G171/G172)

When part programs with short traversing blocks are processed, sudden drops in feedrate can occur because the time taken by the block preparation function to process the short traversing blocks is longer than the time taken to execute these blocks.

The block change time (block preparation+block transfer) is too long in such a case.

A shorter block change time results if processing is carried out with the FIFO function.

A block buffer (FIFO memory) is first filled after NC START or by programming G171.

The next part program blocks are temporarily stored in this FIFO memory in a prepared state (predecoded). Only when the memory is full with such preprocessed blocks is the program started or continued.

The NC can now access the prepared blocks in the FIFO memory and process them in quick succession.

If the NC occasionally processes a block with a longer traversing path or with auxiliary functions, there is time replenish the FIFO (filling FIFO in background).

It can however happen during a long part program section with short traversing blocks that there is no spare time available to replenish the FIFO memory. As soon as the supply of predecoded blocks in the FIFO memory is exhausted a feedrate drop can occur again.

Filling the FIFO memory with G171

To avoid a drop in the feedrate at an important section of the program, it is possible to fill the FIFO memory selectively before such a program section occurs. Selective filling is initiated by programming G171 in the part program.

This ensures that the following program section from the FIFO memory can be executed with short block change times.

By programming G171, the following occurs in this order:

- Processing stopped
- FIFO memory filled up
- Execution continued

Execution is stopped until the FIFO memory is full or up to a block with M02/30 or up to @714 or up to a block which generates a @714 (setting data block, G74, G200, ...). With software version 5 and higher execution can also be continued by programming G172.

It only makes sense to program G171 in AUTOMATIC mode.
G171 must be programmed separately in a block.
G172 can also be programmed in a traversing block.

Example:

```
N100 G0 X0 F1000
N110 G171              Stop program execution
N120.....
N130.....
N900 X01
```

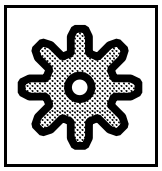


```
N910 G172                Continue program execution
N920 G01 X50 Y100
N930
N980 M30
```

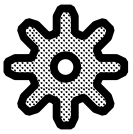
Blocks N120 to N900 can be executed without a drop in the velocity if enough block buffer is available and no block has been programmed that triggers "@714, empty buffer".

FIFO memory size for Software version 1 and 2

The NC CPU has a FIFO memory which can hold up to 60 blocks.



This total memory can be divided amongst the individual channels with a machine data.



If the FIFO memory is used in 2 channels, the memory space available is halved:

FIFO active in one channel: 60 FIFO blocks available in the channel
FIFO active in two channels: 30 FIFO blocks per channel available.

The FIFO function cannot be activated in more than 2 NC CPU channels. If a FIFO buffer is activated for a channel which does not exist, it is ignored.

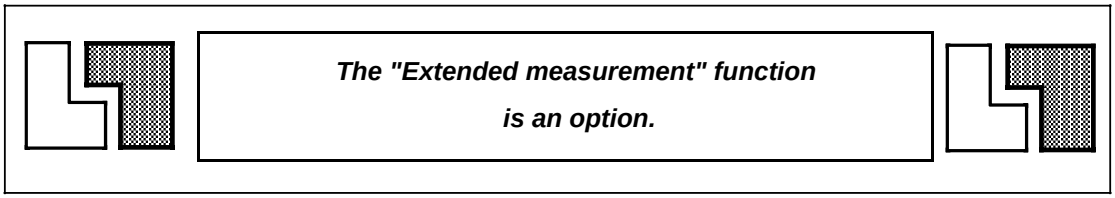
FIFO memory size with software version 5 and higher

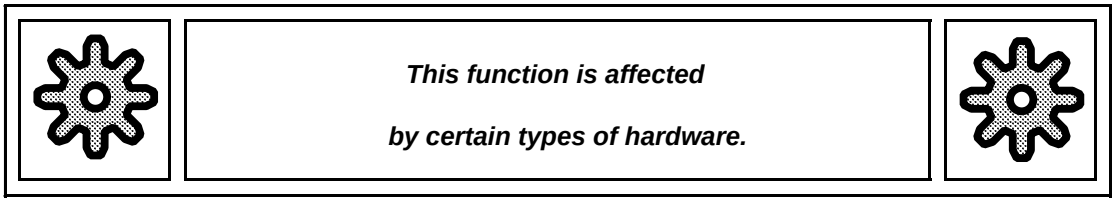
The function must be enabled in the machine data.

Depending on the memory configuration, up to 3500 block buffers are available which, with the flexible memory configuration, can be distributed among the individual channels in any way. The number of available block buffers can also be limited with setting data 204*.

The maximum number of block buffers that are already predecoded are displayed in the menu "Program control".

8.19.5 Extended measurement (G720/G721/G722) 1)





Commands G720...G722 permit a programmable number of individual actual position measurements for a selectable number of axes to be continuously recorded and stored in a measurement trace (measured value recording). This function supports centering cycles for gear wheel machining (hobbing/gear shaping).

The measured values are recorded for specific axes in an R parameter field. The start address must be specified when the function is programmed. This address contains the number of recorded measured values, the following addresses contain the actual measured values in the order they occur chronologically.

The measurement function can be programmed in all operating modes.
Traversing motions can be programmed in the same block; they must, however, be positioned before the G720 command.

Block change behaviour is determined by the following G functions:

G720 "Delete distance to go" is executed during the last measurement and the block is changed.

G721 Block change after termination of the programmed traversing motion (corresponds to simultaneous axis with block end response)

G722 Immediate block change or block change unaffected by measurement function (corresponds to simultaneous axis without block end response)

The "Extended measurement" function is deselected by a reset or through programming of M30, M02.

Program syntax:

```
G720/1/2 <measuring input/trigger signal (MT=)> <edge flag (MF=)> <axis>  <start address
for R parameter fields (MS=)> {number of measurements to be taken (MA=)}
```

Measuring input:

MT=1	Selection of sensor 1 (effective with every measuring circuit hardware)
MT=2	Selection of sensor 2
MT=R<No>	Variable assignment of sensor input via R parameter

1) *Software version 4 and higher*

Edge flag:

MF=–1	Measurement trace via negative edges [1] (effective with every measuring circuit hardware)
MF=1	Measurement trace via positive edges [1]
MF=–2	Measurement trace via alternate negative/positive edges, starting with negative edge [1]
MF=2	Measurement trace via alternate negative/positive edges, starting with positive edge [1]
MF=R<No>	Variable assignment of edge flag

Axis name/ axis address: Specification of axes involved in measurement (max. 5 axes permitted)

X Y2=	Direct specification for axes X and Y2 via axis name
@441 <value3>	Indirect specification via @441 command (for cycles)

Start address for R parameter fields: Only R parameters may be used to specify the start address; the start address of the 1st axis must always be specified.

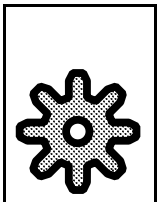
MS=1099	Start address from R parameter 1099 onwards
MS=R799	Start address beginning at R parameter stored in R799

Number of measurements to be taken: Specification of number of measured values to be recorded

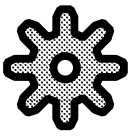
MA=50	50 measurements for every axis involved
MA=R<No>	Variable assignment of number of measurements via R parameters

Notes:

- If a G720/G721/G722 block occurs while a preceding G722 block for the same axis is still being processed, then the preceding G722 block is aborted without error message and processing of the new G720/G721/G722 block commences.
- Processing is aborted without error message if only the edge flag input is changed.
- The "Reference point approach", "In-process measurement" (@720) and "Extended measurement" functions are mutually interlocked. If one of these functions is programmed before the other has been properly terminated, then the first function is aborted without error message and processing of the newly requested function started.
- Only enabled up to axis address extension 9.



- ***The limit frequency specified by the machine manufacturer must be noted where high-speed measurements are made.***
- ***The no. of possible measured values per axis is predetermined by the machine manufacturer and entered in the measured values buffer (s.Start-up Guide, Chap. Flexible Memory Configuration, Function Descriptions).***



1) Signal at measuring circuit input

8.19.6 Channel assignment for protection spaces (G181/G180) (SW 6 and higher)

Protection spaces are not assigned to a particular channel. The assignment is carried out by the G function G181 in the part program.

Tool offsets are set channel-specifically according to absolute value and direction via the current level and D number.

Program syntax:

G 181 S<protection space number>

Immediately after the G function the protection space identifier ***S*** and the ***<protection space number>*** must be programmed.

The protection space is then adapted to the tool offset currently active in the channel into which the G function in the main run is read in.

The G function is modal, therefore any changes made to the D number or to the tool offset are automatically calculated in the compensation memory.

In order to reach a uninterrupted continuous collision monitoring, based on the adaptation of the protection space to the tool offset, all data relevant to tool exchange should be stated in the part program block.

T<tool location No.> D<tool No.> G181 S<protection space No.>

Each channel can at any time change the assignment of channels and protection spaces and thus also the adaptation of the protection space. The tool No. D0 has no effect in reference to the protection space.

Deselection of the protection space adaptation is carried out by programming the G180 G function in the part program.

Program syntax:

G 180 S<protection space number>

The G180 and G181 G functions are modal. They are displayed in the automatic basic display for the length of the block. No deselection of the protection space adaptation seleted using G181 takes place at the end of the program or with RESET.

The G functions are located in the 28th group.

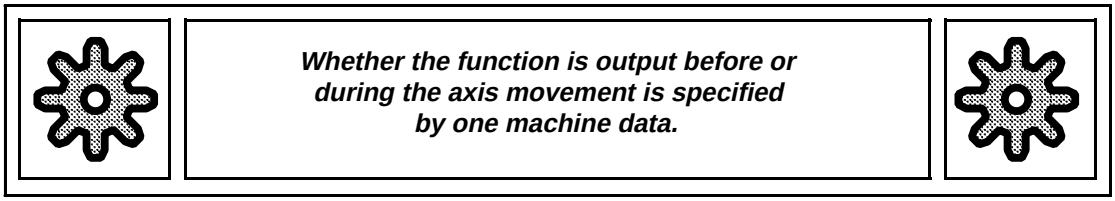
END OF SECTION

9 Switching and Auxiliary Functions (M, H, D, T)

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
- Spindle setpoint value S value
- Auxiliary function H
- Tool offset number D
- Tool number T

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



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.

9.1 M functions with a fixed meaning

9.1.1 Programmed stop, unconditional (M00)

Function M00 permits the program to be interrupted, e.g. in order to perform a measurement. On termination of the measurement, 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.

9.1.2 Programmed stop, conditional (M01)

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.

9.1.3 End of program (M02/M17/M30)

M02**/**M17**/**M30 is written in the last block of the program. The program is thus terminated and the channel reverts to the Reset state. M02**/**M17**/**M30 can be in the block together with other functions or on its own.

9.1.4 End of subroutine (M17/M02/M30)

M02/M17/M30 is written in the last block of a subroutine. It can be written in a separate block or in a block containing other functions. M02/M17/M30 must not be written in the same block as a subroutine (nesting).

9.1.5 Feedrate reduction ratio (M36/M37)

M36 Feedrate 1:1 (Initial setting)
M37 Feedrate reduction ratio 1:100

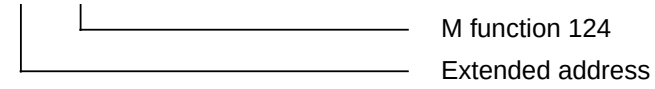
M36 and M37 must not be programmed together in one block.

9.1.6 Freely assignable miscellaneous functions

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

The extended address notation must be used for all freely assignable miscellaneous functions e. g.:

M3=124



Further details on the use of the various functions can be found in the specific program key of the machine tool manufacturer. The meaning of some of these functions is defined in DIN 66025.

9.2 Auxiliary functions H

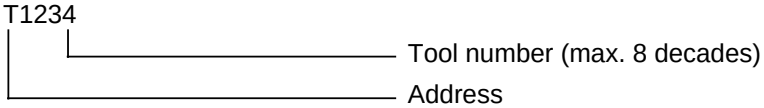
One auxiliary function per block can be entered under address H for machine switching functions or movements not covered by numerical control. H can be programmed with up to 8 decades. The meaning of the functions is described in the Programming Guide of the machine tool manufacturer.

The extended address notation must be used for the H word: H..=...

9.3 Tool number T

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

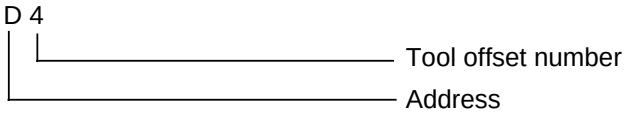
T1234



The extended address notation must be used for the T word: T..=...

9.4 Tool offset number D

The D number contains the tool number, the tool type, the tool geometry as well as the wear dimensions and the base dimensions of the tool.



9.5 Rapid D, F, H, M, S and T functions

You can define the auxiliary functions D, F, H, M, S and T individually as rapid or slow. You can place several auxiliary functions in one block but you can only activate the function if **all** auxiliary functions are defined as rapid.

You can set the output of D, F, S and T functions to be rapid or slow in machine data.

A minus sign must be programmed with the H and M functions. If you place a minus sign in front of the number of a D, F, S or T function, an alarm will be output.

Auxiliary function	Contents of R parameter	Error (alarm 3000)	Rapid auxiliary functions
M = R14	113	no	no
M = – R14	113	no	yes
M = R14	- 113	no	yes
M = – R14	- 113	no	no
M = 03		no	no
M – = 03		yes	——
M = – 03		no	yes
M – = – 03		yes	——
M – 03		no	yes
M 03 –		yes	——

Notes:

- No rapid auxiliary functions are output in blocks with program coordination calls
- It is not possible to program M02/M30 as a rapid auxiliary function
- The auxiliary functions M00, M01, M17 can be programmed as rapid or as slow functions
- In part program blocks from which G functions are to be read using FB69, the auxiliary functions must be output slowly.
- The auxiliary functions collected during a block search are output as programmed.

9.6 Path-dependent function output (G51x, G52x)[1)]

The function "Function output in the block" enables the output of the following functions at a defined time within the NC block:

- Auxiliary functions (M, S, T, H, F and G functions) to the PLC
- Commands of the program coordination to the PLC (or "internal wait marks ", see Section 7.2 Program coordination)

The functions are output only when the condition stated under G51x**/**G52x is fulfilled. If the condition is not fulfilled until the end of the block, the function is output at the end of the block (in the case of function output with actual reference, when the exact stop window is reached) (for exceptions, see "Special cases: delete distance-to-go").

The auxiliary function output "rapid/slow" (see Section 9.5) can be programmed without restriction. With the function "Function output in the block", only the start of the function output is postponed, the functional timing of the output to the PLC remains unchanged.

The following criteria are possible for the output time:

- G511 function output dependent on distance-to-go with setpoint value reference (see Section 9.6.1)
- G522 position-dependent function output with actual value reference (see Section 9.6.2)

Behavior with continuous-path mode G64

In the case of a G64 block with G51x**/**G52x instruction and slow auxiliary functions**/**program coordination, traverse is performed to the exact stop at the end of the block until the functions have been acknowledged by the PLC.

If the acknowledgement is made

- before the starting point of deceleration, the block is retracted without drop in velocity.
- after the starting point of deceleration. the block is accelerated again to setspeed.
- at the end of the block, the block has already been decelerated up to the exact stop. The block is changed now.

Special cases in G51x/G52x blocks

- NC start after NC stop or mode change

  The functions continue to be output only once even if the condition is fulfilled several times (e.g. by retraction in JOG). The output starts as soon as the condition has been fulfilled.

---

1) With SW 5.6 and higher

- Delete distance-to-go

  If "delete distance-to go" is started in a block, the functions are only output if the condition is fulfilled until the end of the block (deceleration path).

- Block search

  Block search to a G51x/G52x block is not permitted. In this case, the alarm 2506 "Extended function output in the target block" is output.

- Contour definition, smooth approach and retraction, tool radius compensation

  In the case of contour definition and smooth approach, the auxiliary functions are output in the first generated block, with smooth retraction in the second block. The blocks in which auxiliary functions are output in connection with tool radius compensation are described in Section 11.10.

The machine data "Auxiliary functions before traverse"

is ineffective in a block with G51x/G52x.

9.6.1 Function output dependent on distance-to-go with setpoint value reference (G511)

This function enables to control the output of functions depending on the path distance-to-go (setpoint value assessment).

Program syntax: G511 F<path distance-to-go>

G511 may be programmed arbitrarily in the block. The path distance-to-go must be indicated directly after G511. In addition, one of the functions to be output must be programmed in the block. The G-function is effective block by block.

Example:

```
N5   G0  X0
N10  X100  H88  G511  F10
```

(Output of the auxiliary function H88 10 mm before the end of the block)

9.6.2 Position-dependent function output with actual value reference (G522)

Program syntax: G522 <axis name><axis position><overtravel direction>

Overtravel direction:

K-1	Overtravel of the axis position in the negative direction
K1	Overtravel of the axis position in the positive direction
K0	Overtravel of the axis position independently of the direction

G522 may be programmed arbitrarily in the block. The axis name, axis position and overtravel direction must be programmed completely in the order described. In addition, one of the functions to be output must be programmed in the block. The G-function is effective block by block. The axis position can be programmed with G90/G91. The active offsets of the channels are taken into account.

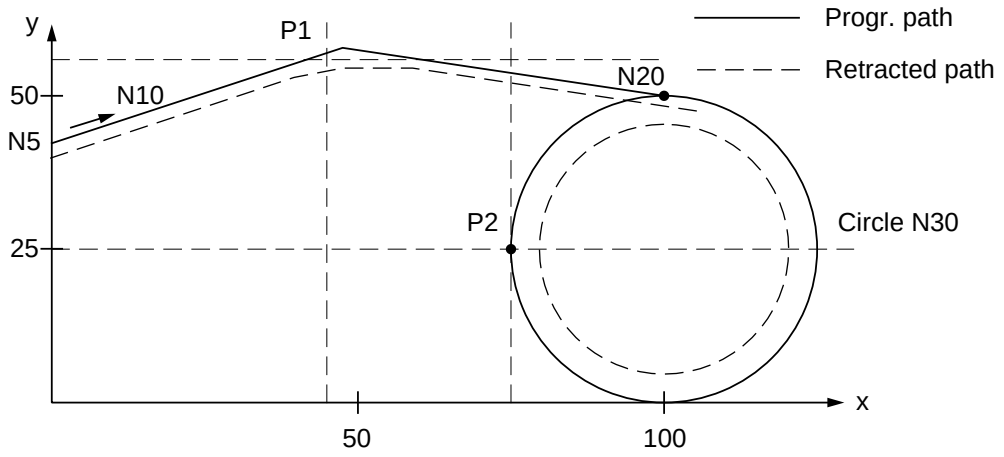
Example:

```
N5   G1  X0  Y40  F...  G64
N10  X50  Y60  H88  G522  X48  K0
```

Output of the auxiliary function when the x-axis has reached position 48 (P1)

```
N20  X100  Y50
N30  G2  X100  Y50  I0  J-25  H77  G522  Y25  K+1
```

Output of the auxiliary function when the y-axis overtravels position 25 (P2) in the pos. direction



Selection of the "correct axis" in path movements:

In the case of actual value reference, it is possible that a coordinate which is located on the programmed contour is not approached (see above). This problem can be eliminated by selecting the "correct axis" (s.a.): In the above-mentioned example, the two output starting points P1 and P2 are not approached according to their actual value (with slow auxiliary functions in the block, the point P1 would be approached since braking takes place at the end of the block). With linear movement, the X-axis cuts where suitable, with circular movement the Y-axis cuts. These are to be taken into account to determine the output time. If no "correct" axis is found, G511 can be used. In each case, the auxiliary functions are output at the end of the block (reaching of the exact stop window, exception "delete distance-to-go" see Section 9.6).

Special cases:

- Rotary axes

  G522 can be programmed only together with linear axes. When a rotary axis is programmed, the alarm 3006 "Incorrect block structure" is output.

- Simultaneous axes

  G522 can also be programmed with simultaneous axes.

  Example:

  ```
  G01  P[X]100  F[X]100  G522  X40  K0
  ```

- Axes from other channels

  G522 can also be related to axes which are traversed in other channels. Please observe that the position may be different due to different offsets (e.g. tool offsets) in the channels.

- Axes with transmit and coordinate transformation

  G522 can also be programmed with real axes of transformation. Please observe that changed movements (and thus positions) result in the real coordinate system from offsets in the fictitious coordinate system. There is no actual position in the fictitious coordinate system (actual position = setpoint position).

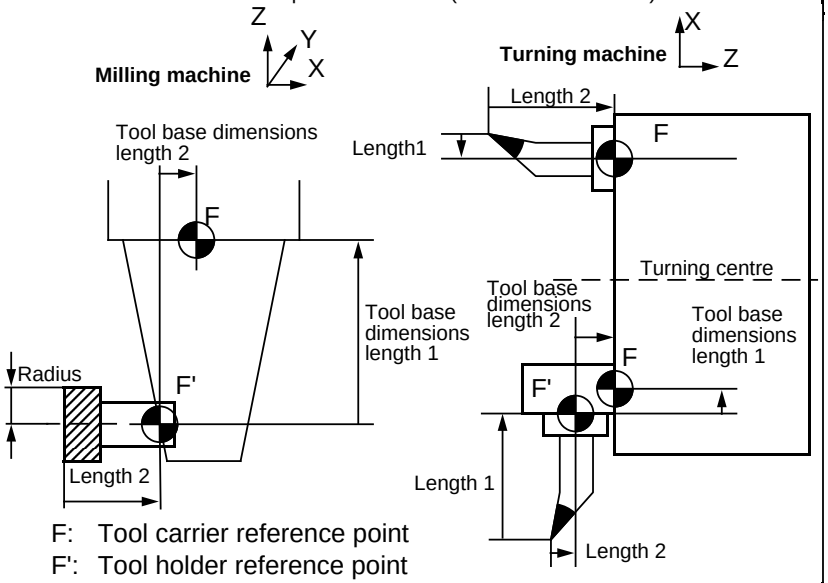
END OF SECTION

10 Tool Offsets

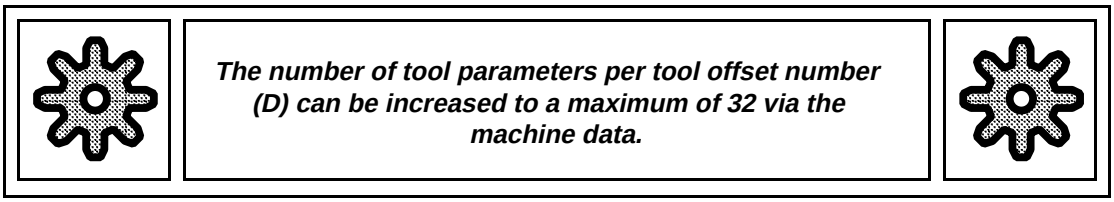
10.1 Tool data

The tool data are stored under one or several tool offset numbers (D) as tool parameters (standard 10). The tool parameters describe the tool. Some of the tool parameters have a fixed definition, others can be defined by the user for tool management.

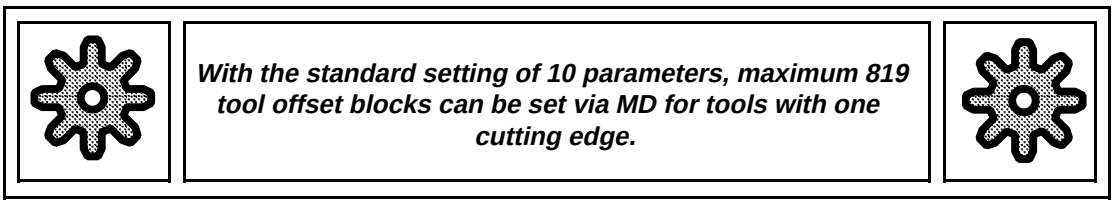
Explanation of tool parameters (P0 to P32) of a tool offset number (D):

Designation	Application
P0: Tool number	Tool management
P1: Tool type	The tool type tells the control which parameters of the tool offset memory (P2..) are to take effect and these parameters are assigned to the individual axes depending on the active plane definition (G16 ... G19). Type 0 Tool not defined Type 1...9 Turning tools, position of tool tip Type 10...19 Tools with active length compensation (e.g. drills type 10; tap type 11) 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) Type 40 Cutter for 5 axis tool length compensation (option) Type 50 Gear cutting tools
P2 ... P4: Geometry	Tool setting dimensions
P5 ... P7: Wear	Wear data
P8 ... P9: Tool base dimensions	The tool base dimensions can be used when the reference point of the tool holder (geometrical data) does not coincide with the reference point of the tool carrier. The distance between the reference points is entered (see illustration below).  F: Tool carrier reference point F': Tool holder reference point
P10 ... P32:	Freely assignable for tool management

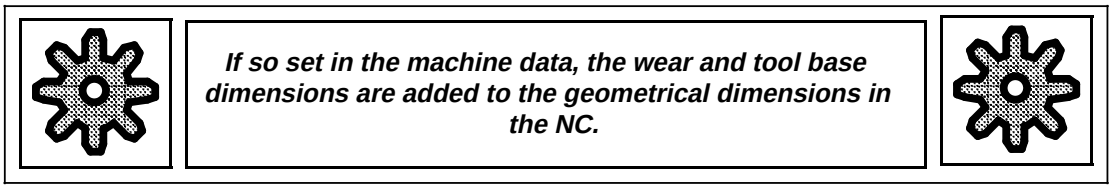
The offset data are only active for the tool type for which they are described.



The number of tool parameters per tool offset number (D) can be increased to a maximum of 32 via the machine data.



With the standard setting of 10 parameters, maximum 819 tool offset blocks can be set via MD for tools with one cutting edge.



If so set in the machine data, the wear and tool base dimensions are added to the geometrical dimensions in the NC.

10.2 Selection and cancellation of length compensation

- This function can only be selected if G00 or G01 is active. It is necessary to select the plane perpendicular to the length compensation direction. The compensation only takes effect when the corresponding axis has been programmed.

  N5 G00 G17 D.. Z.. L_F

  Only the length compensation is used from offset 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. The compensation only takes effect when the corresponding axis has been programmed.

  Length compensation without tool radius compensation (TRC)

```
N05  G90  G00  G17  D1  Z..  LF          Selection of length compensation
:
:
N50  D0  Z..  LF                         Cancellation of length compensation
```

  Length compensation with tool radius compensation (TRC)

```
N05  G90  G00  G17  G41  D2  X..  Y..  Z..  LF   Selection of TRC left of the contour
N10  Z..  LF                                    with length compensation
:
:
N50  G40  X..  LF                               Cancellation of TRC.
N55  D0  Z..  LF                                Cancellation of length compensation
```

10.3 Tool length compensation, positive or negative

If length compensation is selected, the compensation is positive. Negative tool length can be programmed with G16. The tool length compensation acts on the axis perpendicular to the tool radius compensation plane.

Example:

```
G16  X  Y  Z±          ;X, Y     =  plane selection
                       ;Z±       =  the sign defines whether the length compensation of axis
                                    Z has a positive or negative effect
G16  X1=  Y1=  Z1±     ;X1, Y1   =  plane selection
                       ;Z1±      =  as above
```

3 to 5 axes must be programmed depending on the type of tool used.

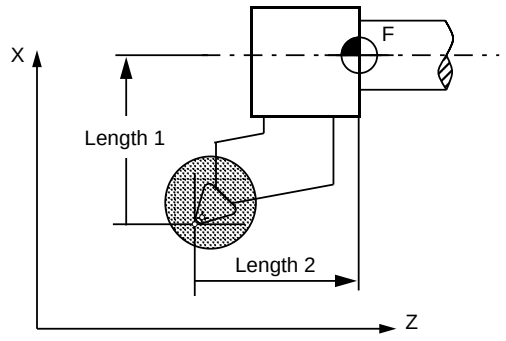
Defining the sign

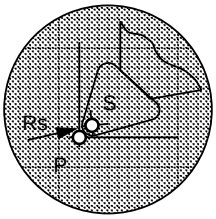
The operator enters a plus sign when the entered tool length compensation (P2 and**/**or P3) is to be calculated in the positive direction. If a minus sign is entered, the tool length compensation is calculated in the negative direction.

G16 is solely a block function, i.e. it does not a cause a movement.

10.4 Structure of tool offset memory for a turning tool

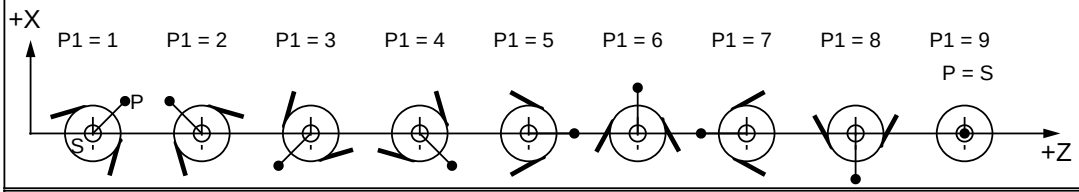
	Structure of offset memory									
	T No.	Type	Geometry			Wear			Tool base dimensions	
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
D_n	1..99999999	1 .. 9	Length 1	Length 2	Radius	Length 1	Length 2	Radius	Length 1	Length 2



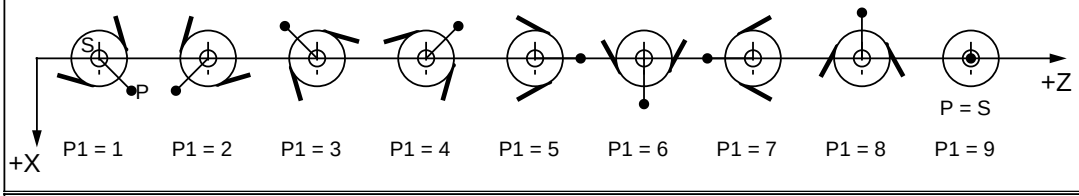


P Theoretical tool tip
S Tool nose radius centre
Rs Tool nose radius
F Tool holder reference point

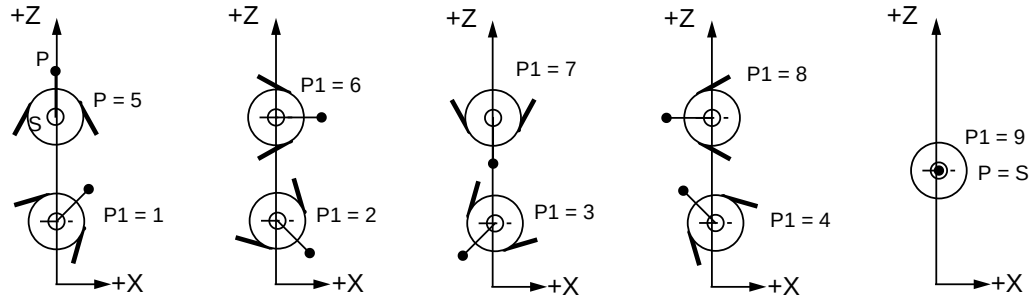
Tool nose position when machining **behind** the turning centre



Tool nose position when machining **in front** of the turning centre



Tool nose position on a vertical turret lathe



In order to calculate the tool radius compensation, the control requires the tool nose radius (compensation value P4) and details of how the turning tool is clamped in the tool carrier.

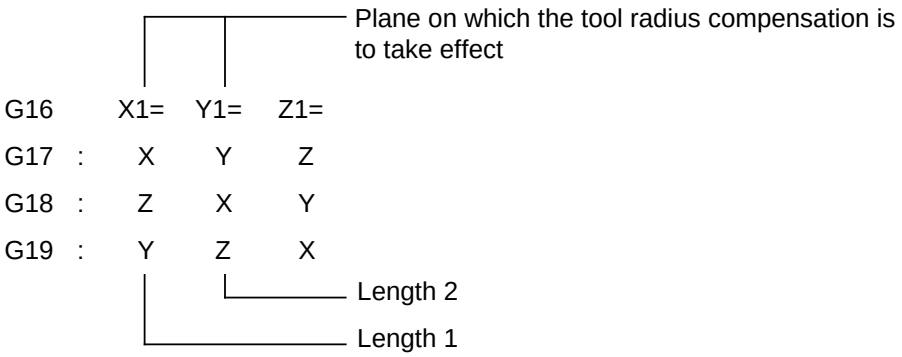
Identifier P1 = 1 to P1 = 9 must be entered in tool offset column P1 (tool type). The position of point P relative to tool nose centre S has thus been defined.

If tool nose 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 tool nose radius must always be entered.

Zero can be selected for the tool nose radius when

- Tools are used with tool nose radii which have negligable values
  or
- the programmer takes account of the tool nose radius when programming the contour.

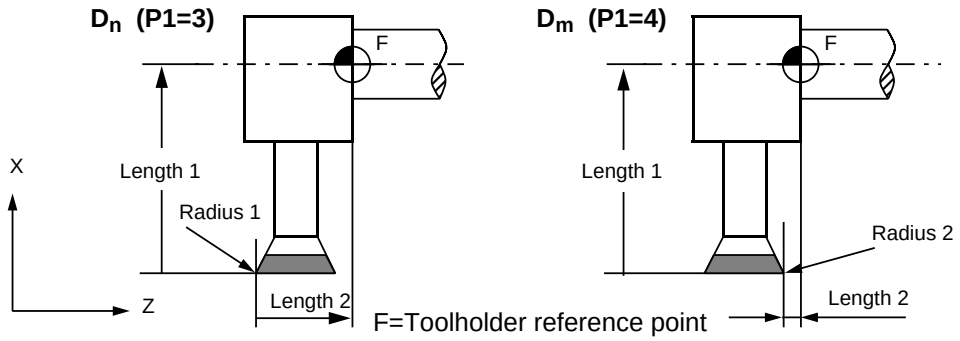
Compensation assignment:



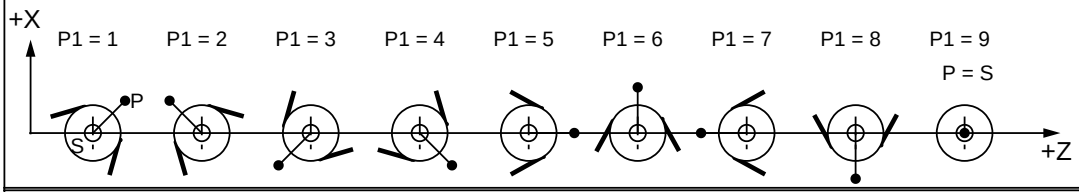
See Section "Machining planes" for axis assignment with G17, G18 and G19.

10.5 Structure of tool offset memory for a grooving tool

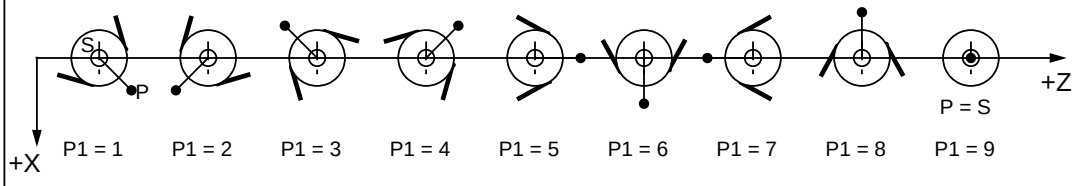
	Structure of offset memory									
	T No.	Type	Geometry			Wear			Tool base dimensions	
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
D_n	1..99999999	1 .. 9	Length 1	Length 2	Rad. 1	Length1	Length 2	Radius	Length 1	Length 2
D_m	1..99999999	1 .. 9	Length 1	Length 2	Rad. 2	Length 1	Length 2	Radius	Length 1	Length 2



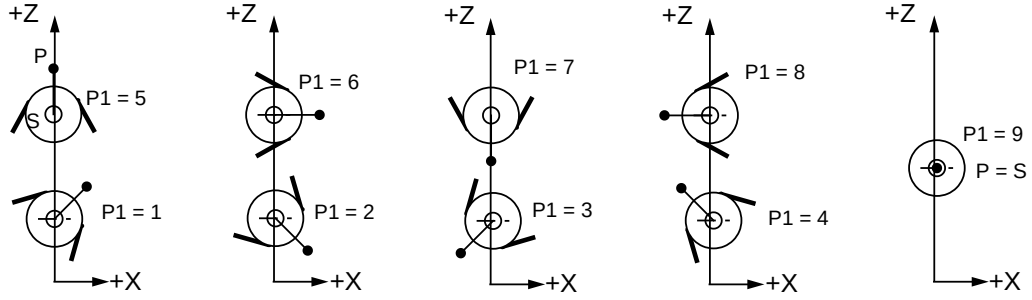
Tool nose position when machining **behind** the turning centre



Tool nose position when machining **in front** of the turning centre



Tool nose position on a vertical turret lathe

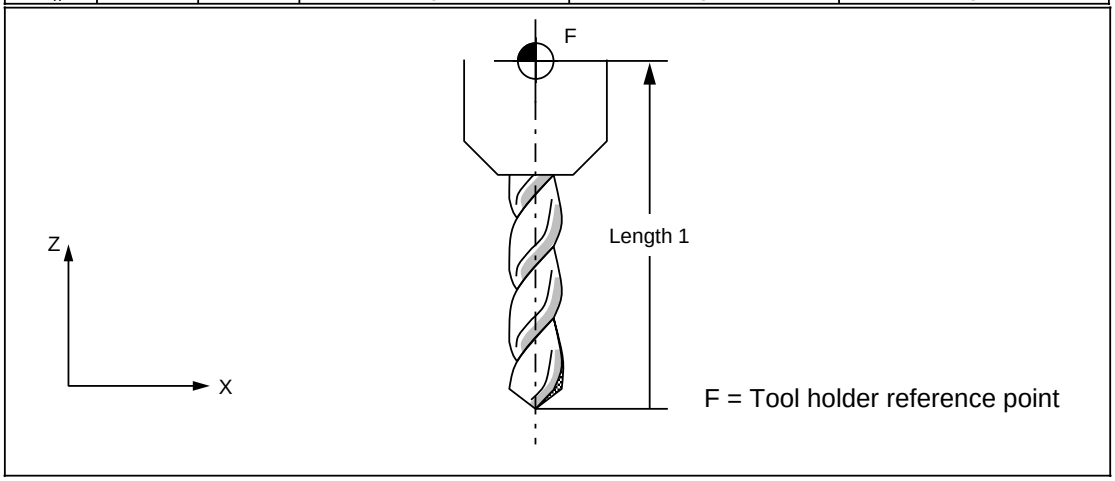


The cutter positions (P1) for both offset values (D_n, D_m) must be defined. The offset assignment is as for the turning tool.

In standard cycle L93 (grooving cycle), the tool offset numbers must follow in succession.

10.6 Structure of tool offset memory for a drill

	Structure of offset memory				
	T No.	Type	Geometry	Wear	Tool base dimensions
	P0	P1	P2	P5	P8
D_n	1..99999999	10	Length 1	Length 1	Length 1



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

The length compensation for a drill acts on the axis perpendicular to the selected plane.

Compensation assignment:

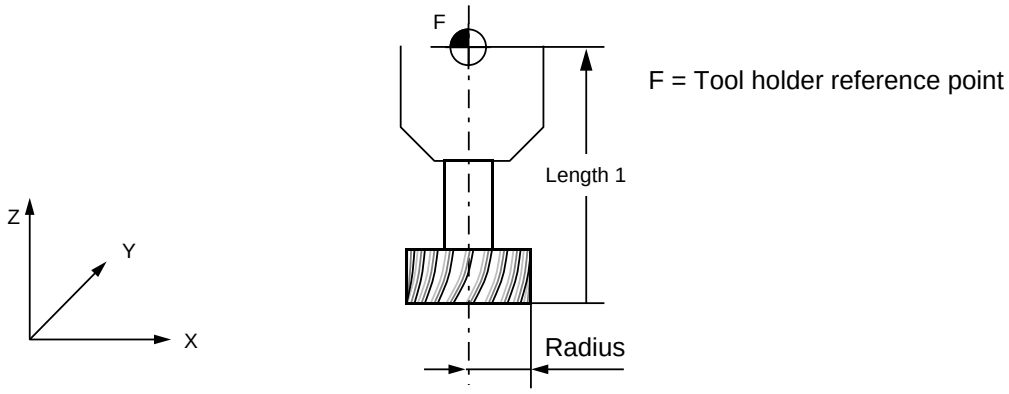
G16		X1=	Y1=	Z1=
G17	:	X	Y	Z
G18	:	Z	X	Y
G19	:	Y	Z	X
				└— Length 1

See Section "Machining planes" for axis assignment with G17, G18 and G19.

Note: Tool radius compensation is not meaningful.

10.7 Structure of tool offset memory for a milling cutter

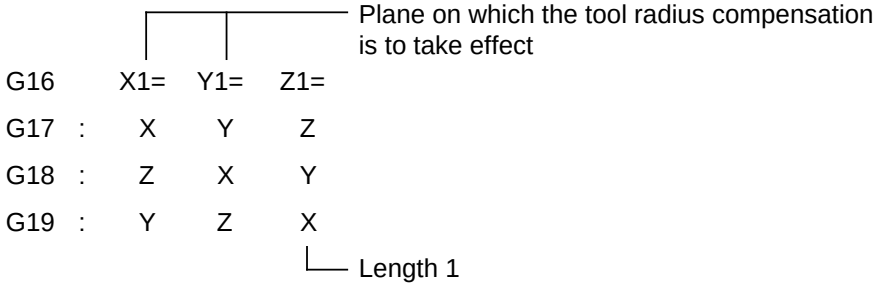
	Structure of offset memory						
	T No.	Type	Geometry		Wear		Tool base dimensions
	P0	P1	P2	P4	P5	P7	P8
D_n	1..99999999	20	Length 1	Radius	Length 1	Radius	Length 1



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

In the case of an end mill, the tool radius compensation is active in the plane specified in the program. The length compensation acts on the axis perpendicular to the selected plane.

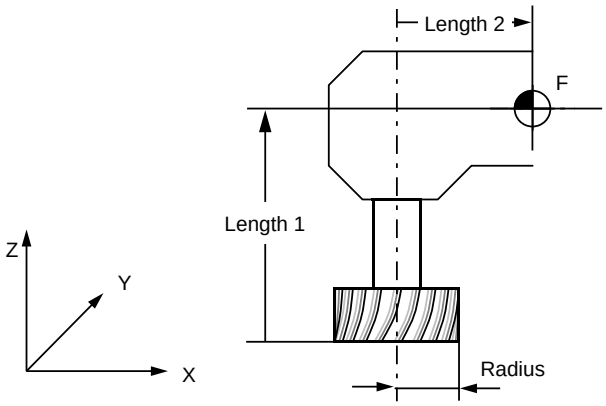
Compensation assignment:



It is possible to define in the machine data that tool type P1 = 0 has the same effect as tool type P1 = 20.

10.8 Structure of tool offset memory for cutter with angle head

	Structure of offset memory									
	T No.	Type	Geometry			Wear			Tool base dimensions	
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
D_n	1..99999999	30	Length 1	Length 2	Radius	Length 1	Length 2	Radius	Length 1	Length 2



F = Toolholder reference point

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

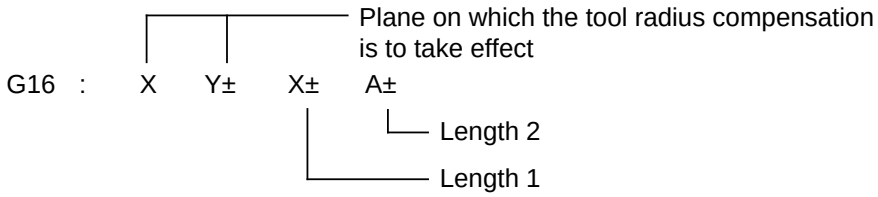
In contrast to the end mill, an angle cutter requires an additional length compensation located in the plane of the tool radius compensation. Here, free axis selection must always be used with preparatory function G16.

Function G16 is followed in this instance by four axis addresses, the first two of which specify the plane in which the tool 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.

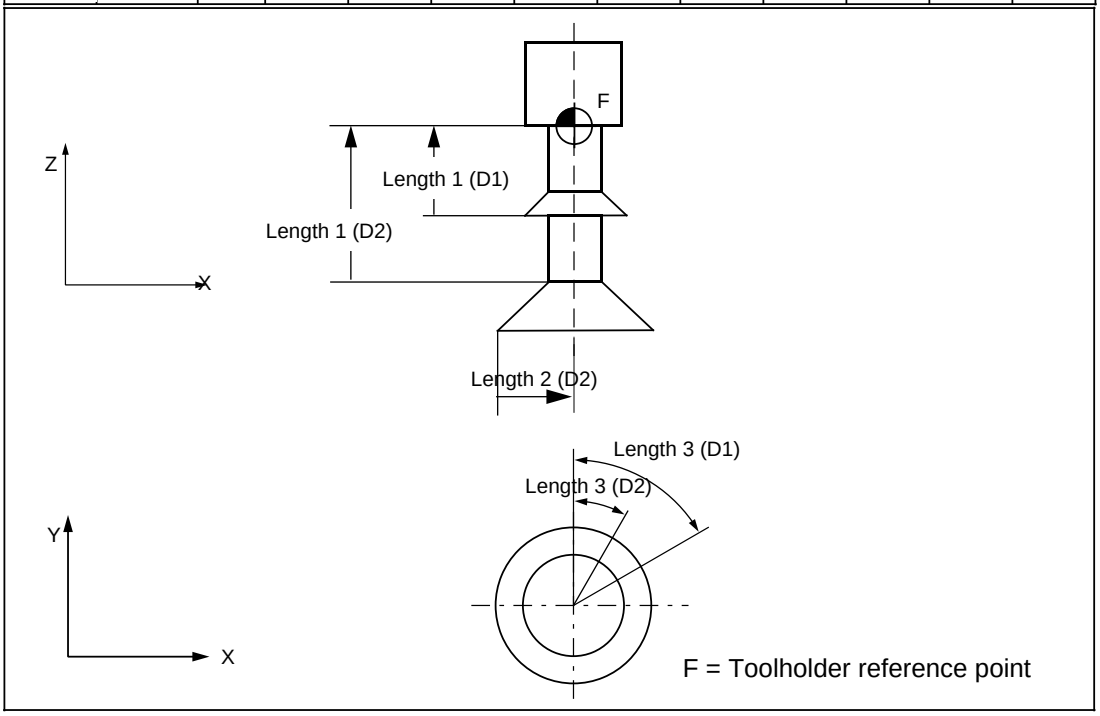
A negative sign can be added to the 3rd and 4th axis address in order to reverse the compensation direction.

Compensation assignment:



10.9 Structure of tool offset memory for gear shaping

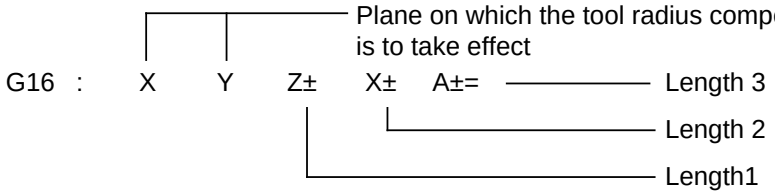
	Structure of offset memory											
	T No.	Type	Geometry				Wear				Tool base dimens	
	P0	P1	P2	P3	P4	P10	P5	P6	P7	P11	P8	P9
D_n	1..99999999	50	Lngth 1	Lngth 2	(Radius)	Lngth 3	Lngth 1	Lngth 2	(Radius)	Lngth 3	Lngth 1	Lngth 2



The number "50 to 59" must be entered under P1 (tool type).
Type 50 is recommended for gear shaping tools.

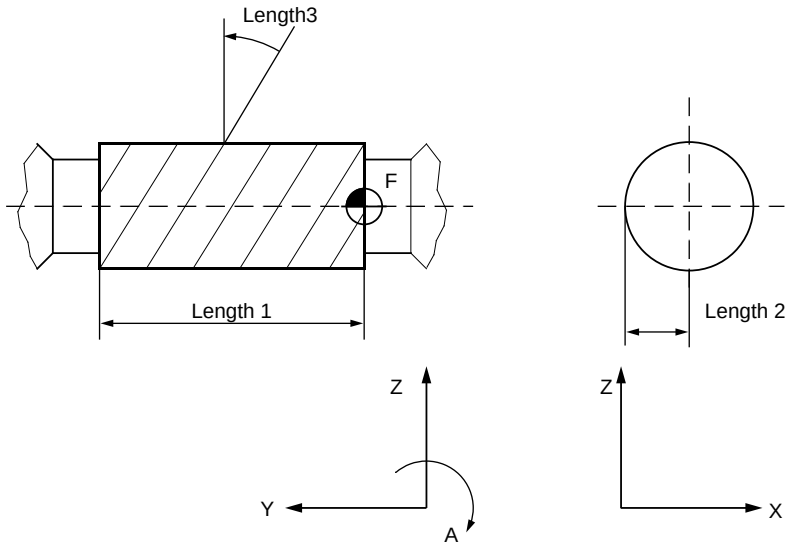
Length compensations are required in 3 axes for gear shaping tools. These must be defined with G16 before the tool is selected. The plane must be specified unambiguously. A negative sign can be added to the axes in order to reverse the compensation direction.

Compensation assignment:



10.10 Structure of tool offset memory for hobbing

	Structure of offset memory											
	T No.	Type	Geometry				Wear				Tool base dimens	
	P0	P1	P2	P3	P4	P10	P5	P6	P7	P11	P8	P9
D_n	1..99999999	51	Lngth 1	Lngth 2	(Radius)	Lngth 3	Lngth 1	Lngth 2	(Radius)	Lngth 3	Lngth 1	Lngth 2

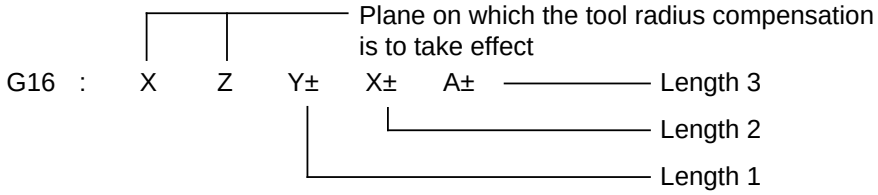


F =Toolholder reference point

The number "50 to 59" must be entered under P1 (tool type).
Type 51 is recommended for hobbing tools.

Length compensations are required in 3 axes for hobbing tools. These must be defined with G16 before the tool is selected. The plane must be specified unambiguously. A negative sign can be added to the axes in order to reverse the compensation direction.

Compensation assignment:



10.11 Structure of tool offset memory for one type of milling head for 5D tool length compensation

	Structure of offset memory									
	T No.	Type	Geometry			Wear			Tool base dimensions	
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
D_n	1..99999999	40	Length 1	is not calculated						

Depending on the type of milling head used, the 5 axis tool length compensation corrects the programmed block end points for short linear blocks with G01. If a tool breaks, the milling machine can continue at the point of the breakage if a tool with the programmed radius but a different length is clamped. The difference in length is entered as the tool length compensation under tool type 40.

The function only works if the programs are generated on the computer using the tool radius.

10.11.1 Axes for 5D tool length compensation (G15)

Function G15 is used to define the 5 axes for 5D tool length compensation according to the order in which they are programmed.

Programming syntax:

G15 <1st linear axis>[-] <2nd linear axis>[-] <3rd linear axis>[-]
<1st rotary axis>[-] <2nd rotary axis>[-]

The function must be on its own in the block and all 5 axis must always be programmed. The calculation of the tool offset depends on of the plus/minus sign:

"Positive" calculation for a linear axis:

Tool offset is calculated on the axis.

"Negative" calculation for a linear axis:

Negated tool offset is calculated on the axis.

"Positive" calculation for a rotary axis:

Tool offset for linear axes is calculated from the position of the rotary axis.

"Negative" calculation for a rotary axis:

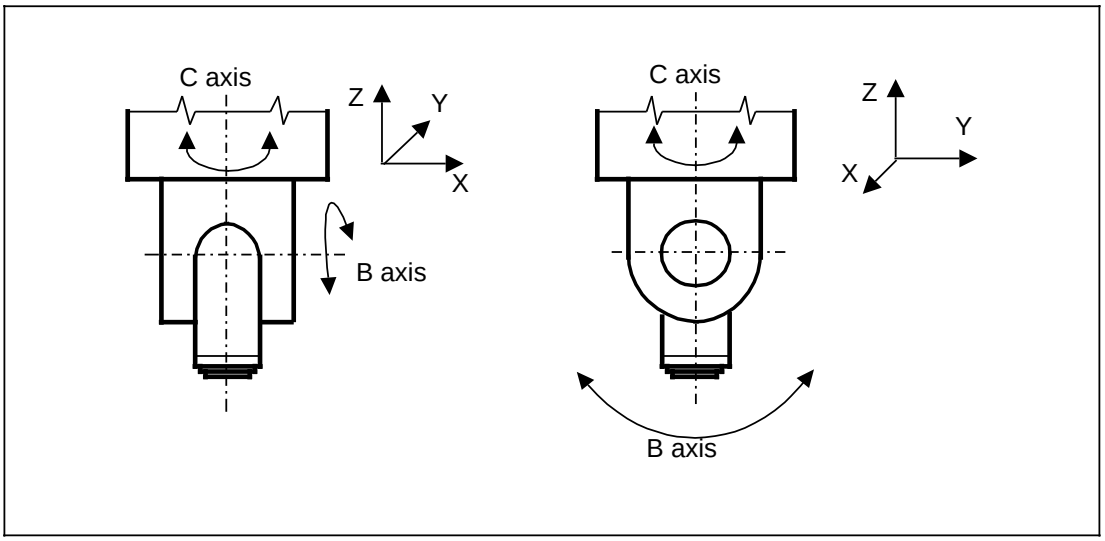
Tool offset for linear axes is calculated from the negated position of the rotary axis.

Notes:

- The function is implemented for SW 3 and higher.
- The function can be programmed with negative sign for SW 6 and higher
- The G16**/**G17**/**G18 levels and G19 are not influenced by the function.
- Following the programming of G15 or after selection of the 5D tool length compensation with Dxx, all axes concerned must be programmed once.

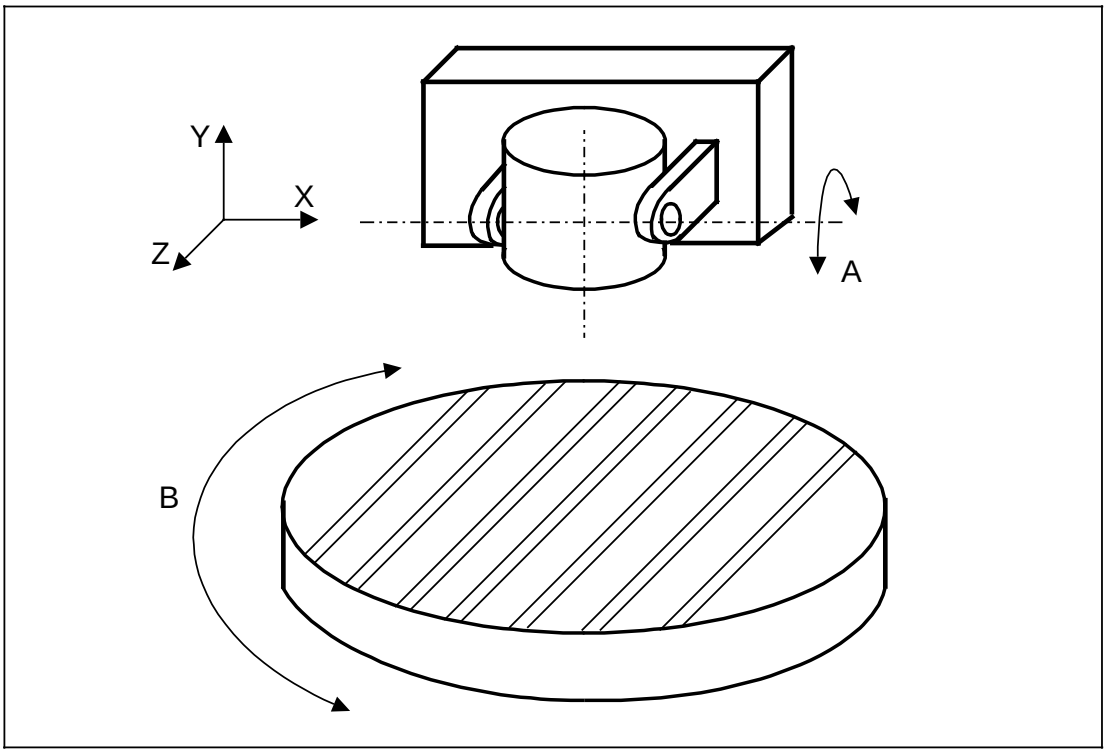
.

If function G15 is not programmed, the first 5 axes of the control are automatically active.



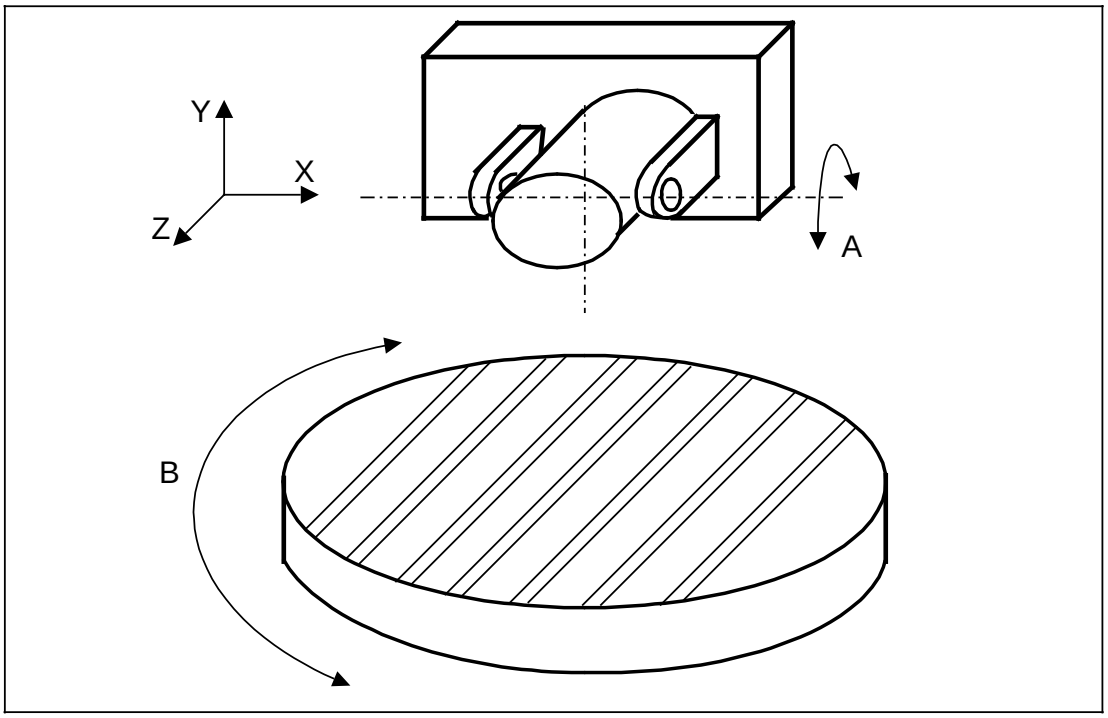
Gimbal head

The B axis lies along the Y axis at C = 0
The tool lies parallel to the Z axis when B = 0.



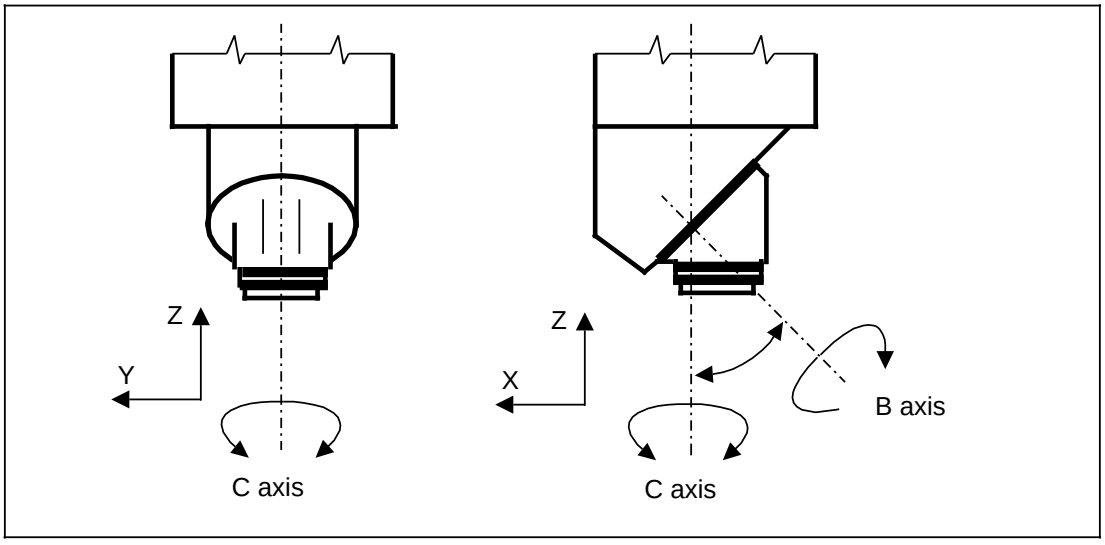
Inclinable head (cutter parallel to Y axis)

The rotary table (axis C) rotates around axis Z.
The inclinable head (axis B) rotates around axis Y.
The inclination range around B is usually – 45° to+45°.
When A = 0 degrees the cutter lies parallel to the Y axis.



Inclinable Head (cutte rparallel to Z axis)

The rotary table (axis C) rotates around axis Z.
The inclinable head (axis B) rotates around axis Y.
The range of inclination around B is usually –45° to+45°.
When A = 0 degrees the cutter lies parallel to the Z axis.



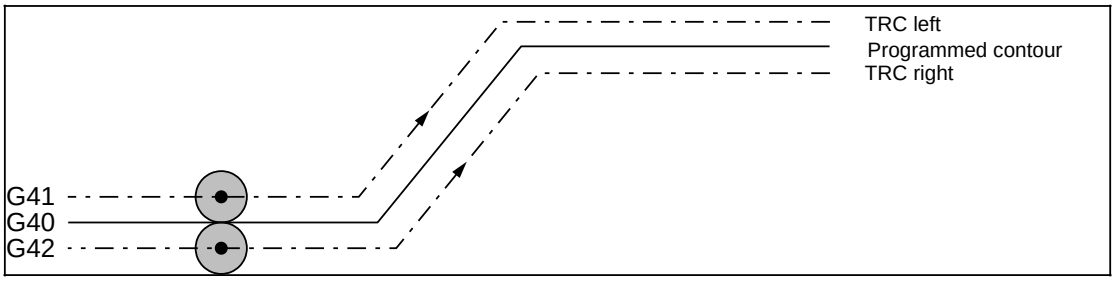
Nutating head

The tool lies parallel to the Z axis when B = 0
The B axis lies along the Y axis at C = 0

END OF SECTION

11 Tool Radius Compensation (G40/G41/G42)

The tool radius compensation (TRC) takes account of the tool radius for automatic compensation. The tool radius compensation calculates an equidistant path to the programmed contour. The compensation will take effect along the direction of movement to the left or to the right of the programmed contour, defined with G functions G41 or G42. The function is deselected with G40. These G functions are modal. G41/G42 refer to a right-handed rectangular coordinate system.

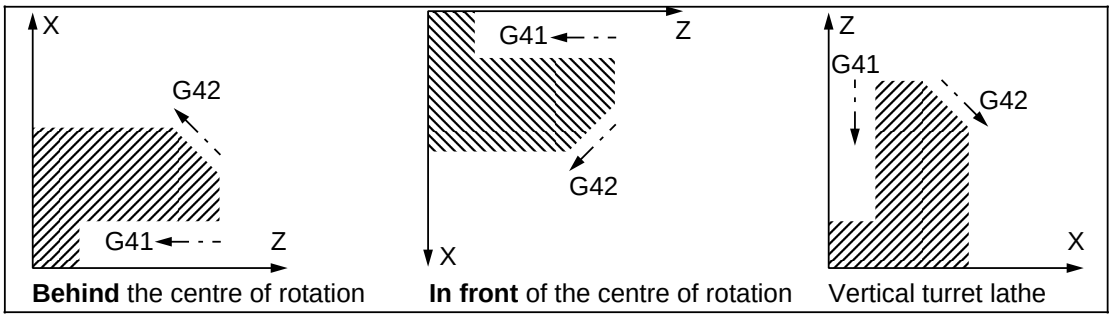


Tool radius compensation (G40, G41, G42)

Use of tool radius compensation on turning machines

On turning machines, the direction of compensation appears to depend on whether machining is taking place in front of or behind the centre of rotation. When machining BEHIND the centre of rotation (clockwise coordinate system) is the definition is as above.

When machining is IN FRONT of the centre of rotation (X axis mirrored around the Z axis), the direction of compensation is reversed for the operator. The part program and the tool offsets are the same whether the same workpiece is machined IN FRONT of or BEHIND the centre of rotation because there is rotation symmetry.



*Using TRC (G41/G42) **behind/in front** of the centre of rotation*

Note:

Please specify which tool is to be used in the machining program. The machining program should be tested with the smallest and the largest possible tool radius.

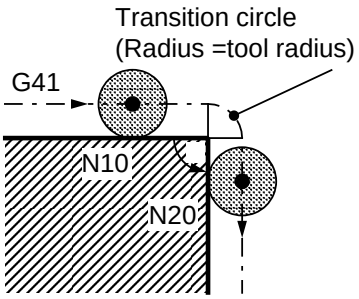
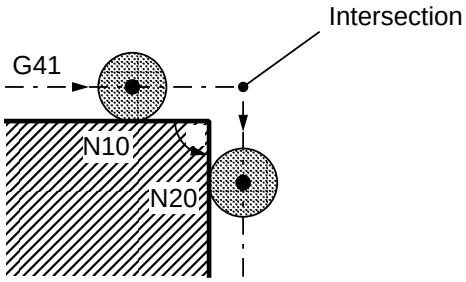
11.1 Behaviour at corners (G450/G451)

With the G commands G450 and G451, behaviour during transition from a contour element (straight line, circle) to a different contour element (behaviour at corners) at **external corners** when TRC (G41/G42) is selected can be controlled. These G functions are modal.

The initial setting of G450/G451 can be set in the machine data.

Comments on the following diagrams:

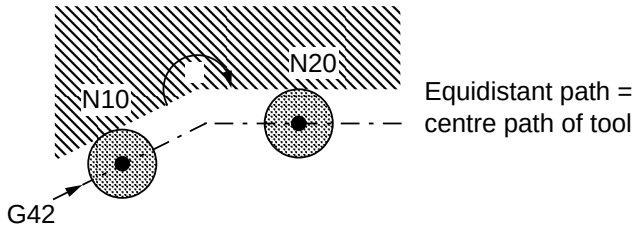
The thick line (———) represents the contour elements. The cutter centre line is shown by the dotted line (— · –).

External corner : 0 <180°	
G450: TRC with transition circle	G451: TRC with intersection
	
The centre point of the tool follows a circular path with the tool radius. This begins with the normal position (perpendicular to path tangent) at the end point of the previous path section and ends with the normal position at the starting point of the new path section. The transition circle is programmed before the next block (N20) and is declared as belonging to this block.	Up to three intermediate blocks are inserted at the external corners (see following diagrams). The inserted intermediate blocks are also declared as belonging to the next block (here N20).

*G450/G451 with TRC (G41) selected at an **external corner***

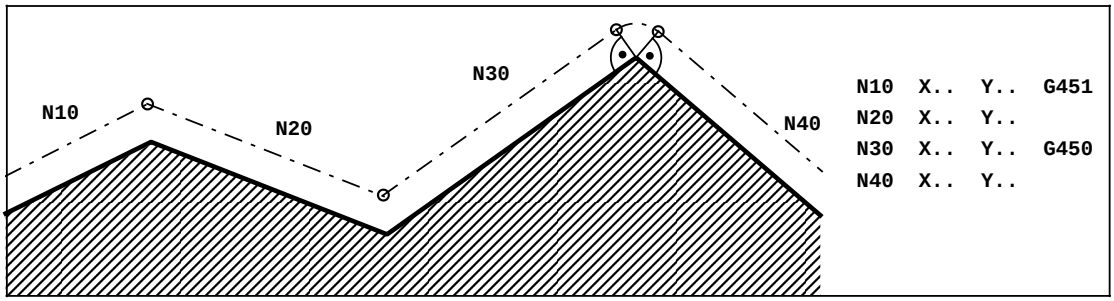
In the case of **internal corners**, the intersection of two equidistants of two successive path sections at a discontinuous transition are **always** calculated independent of G450/G451.

Internal corner: 180°
G450 TRC with transition circle/G41 TRC with intersection



*G450/G451 with TRC (G42) selected at an **internal corner***

With the G commands G451/G450 it is always possible to program a change from ”TRC with intersection to TRC with transition circle” or vice versa in the part program.



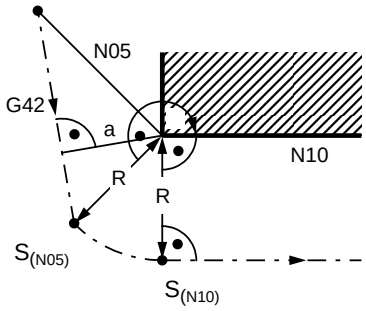
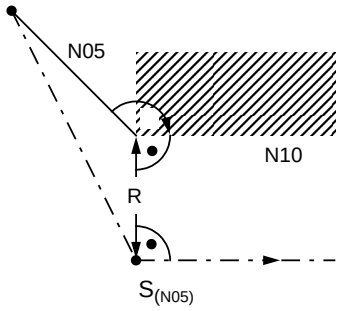
Transition behaviour G451/G450

Comments on the following diagrams:

The contour elements are shown by the bold line (———). The tool centre line is shown by the dotted line (— · —). Programmed approach paths are shown by the thin line (———). All stopping points with individual blocks are marked with an S together with the associated block number in brackets. G42 is used in these examples.

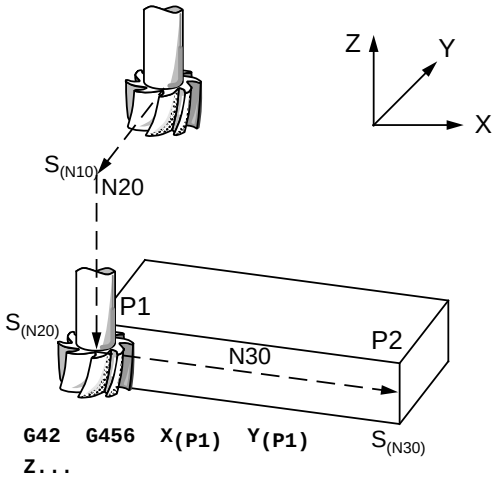
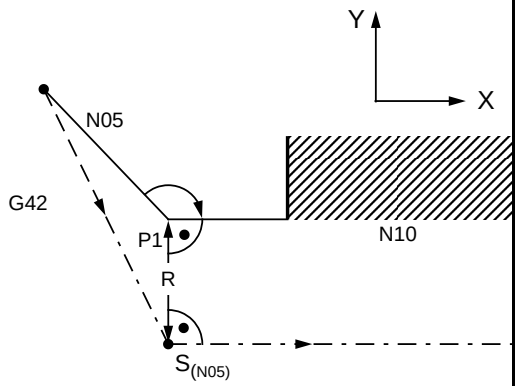
11.2 Selection/deselection procedure (G455/G456)

When selecting/deselecting the function in the angular range ”Behind the contour <180°” you can select whether the selection should be compensated for (G455) or directly executed in the ”normal position” (G456). In the block following the selection block, a block starting vector (length R) is erected perpendicular to the programmed path. The G functions are non modal and must be programmed in a block with G40/G41/G42.

Selection/deselection with <180°	
G42/G455: ”Compensated approach/departure”	G42/G456: ”Approach/departure normal position”
	
The programmed approach/exit point is avoided.	The programmed approach/exit point is approached in the normal way.

G455/G456 with TRC selection/deselection

Note:
If G455 is not programmed in the selection block, then selection with G455 is carried out automatically (G41≙G41 and G455). Approach with G456 makes it possible to approach the contour directly above the workpiece and then to lower the tool (e.g. with milling machines). It is also possible to directly approach a point outside the workpiece. Intermediate blocks are no longer necessary thus shortening program execution times.

Selection with G42/G456 with <180°	
Approach with milling machines	Approach outside the workpiece
 **N10 G42 G456 X(P1) Y(P1)** **N20 Z...** **N30 X(P2) Y(P2)**	 **N5 G42 G456 X(P1) Y(P1)** **N10 X... Y...**

G456 with TRC selected

11.3 Selecting TRC (G41/G42)

Tool radius compensation is selected in the plane defined with G16 to G19 and with the preparatory functions G41/G42 together with the tool offset number D. It is not possible to change planes when tool radius compensation is active.

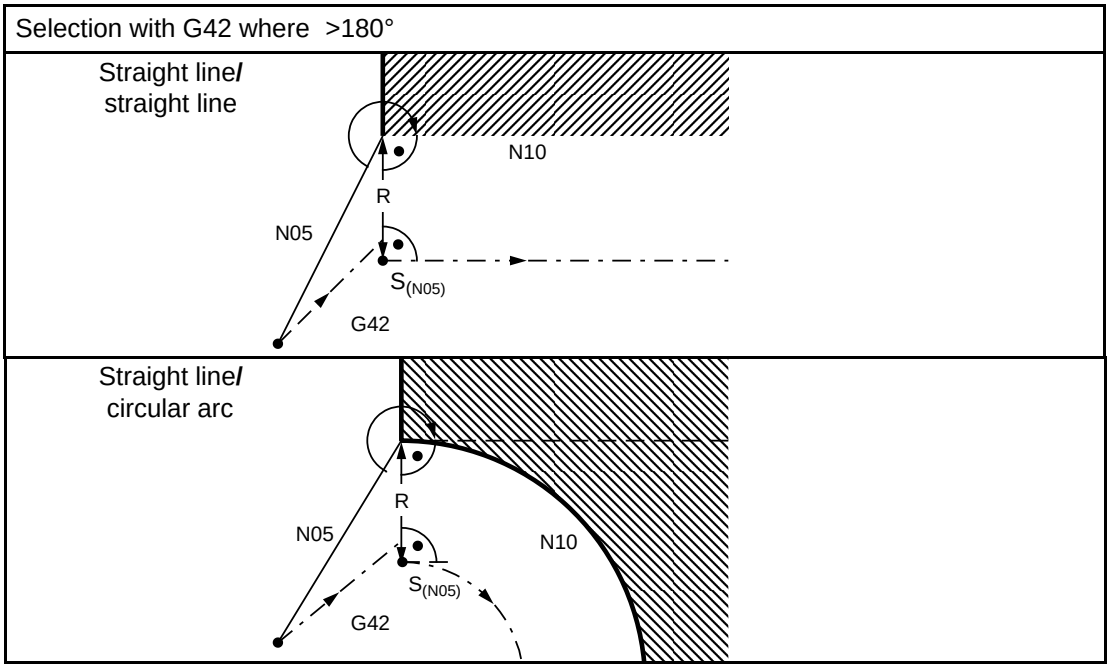
As soon as tool radius compensation is selected, two program blocks are read in to calculate the intersection.

Tool radius compensation can only be selected in a program block with G00/G01/G10 or G11. The compensation will only have an effect if at least one axis of the selected plane is programmed.

When TRC is active, both axes in the plane are traversed even if only one axis is programmed.

Tool radius compensation is not selected if tool offset number D0 is programmed.

Tool radius compensation is usually selected in the angular range 180° and is therefore independent of the set approach and corner behaviour (see Sections 11.1 and 11.2). The programmed point is approached directly in the ”normal position” (perpendicular on approach point with distance tool radius R).

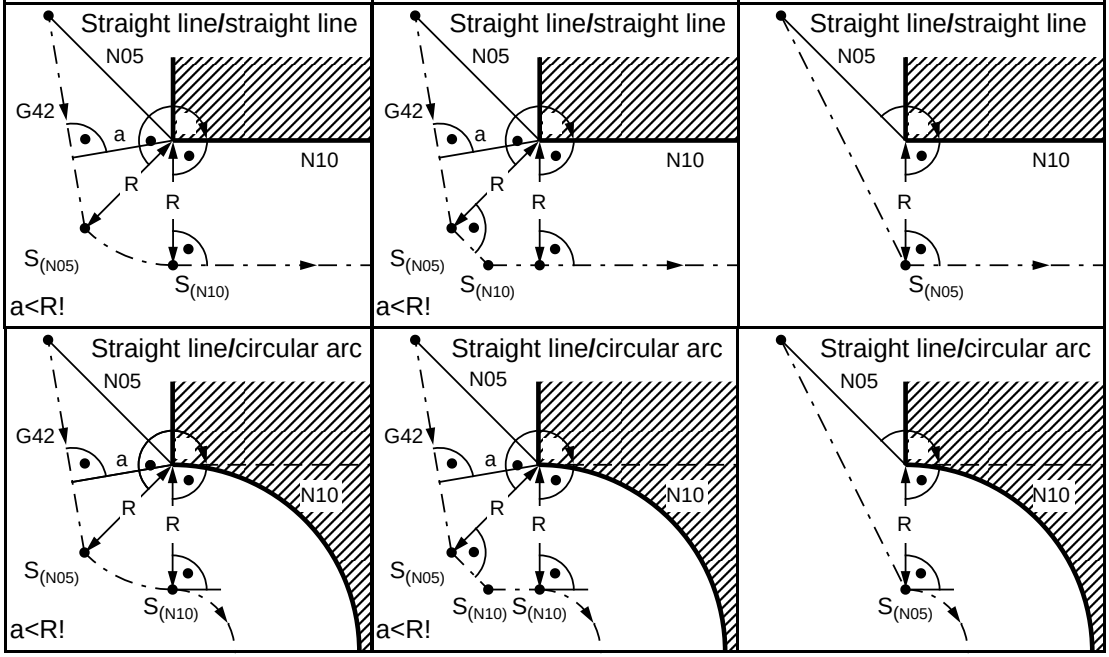


Selection of TRC with >180°

If the function is selected in the angular range 180°, the set selection procedure (G455/G456, see Section 11.2) is used. If compensation selection (G455) is programmed, the set behaviour at corners (G450/G451, see Section 11.1) is used. This results in a contour violation (see following examples, < R!).

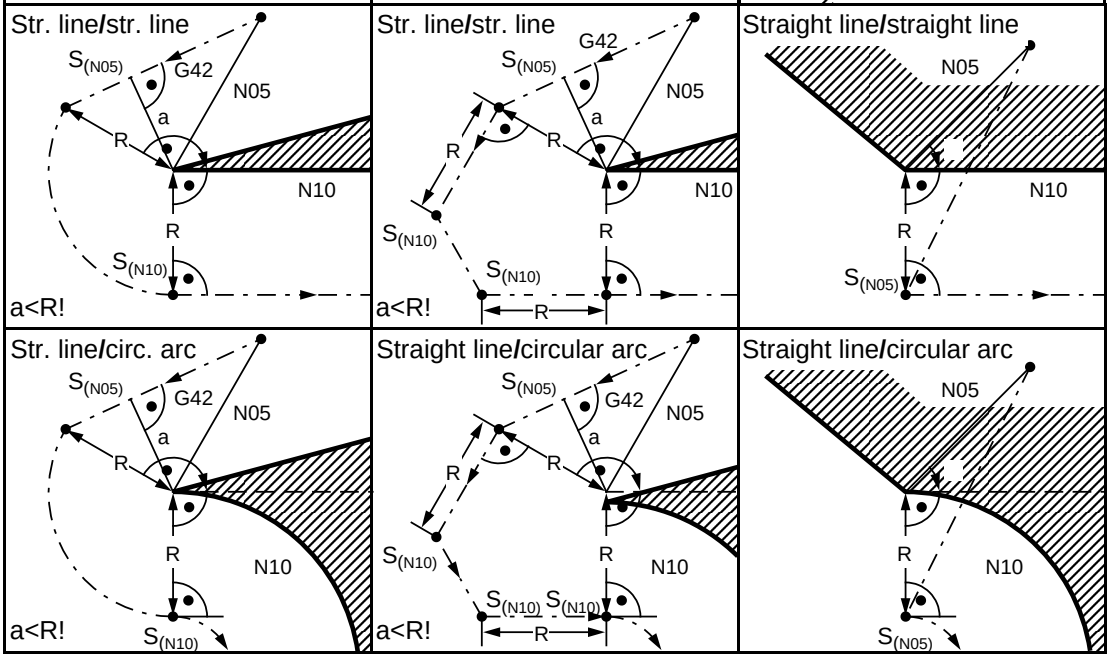
Selection with G42 where 90° 180°

G455: Compensated approach		G456: "Normal position approach"
G450: TRC with transition circle	G451: TRC with intersection	
Straight line/straight line	Straight line/straight line	Straight line/straight line
Straight line/circular arc	Straight line/circular arc	Straight line/circular arc



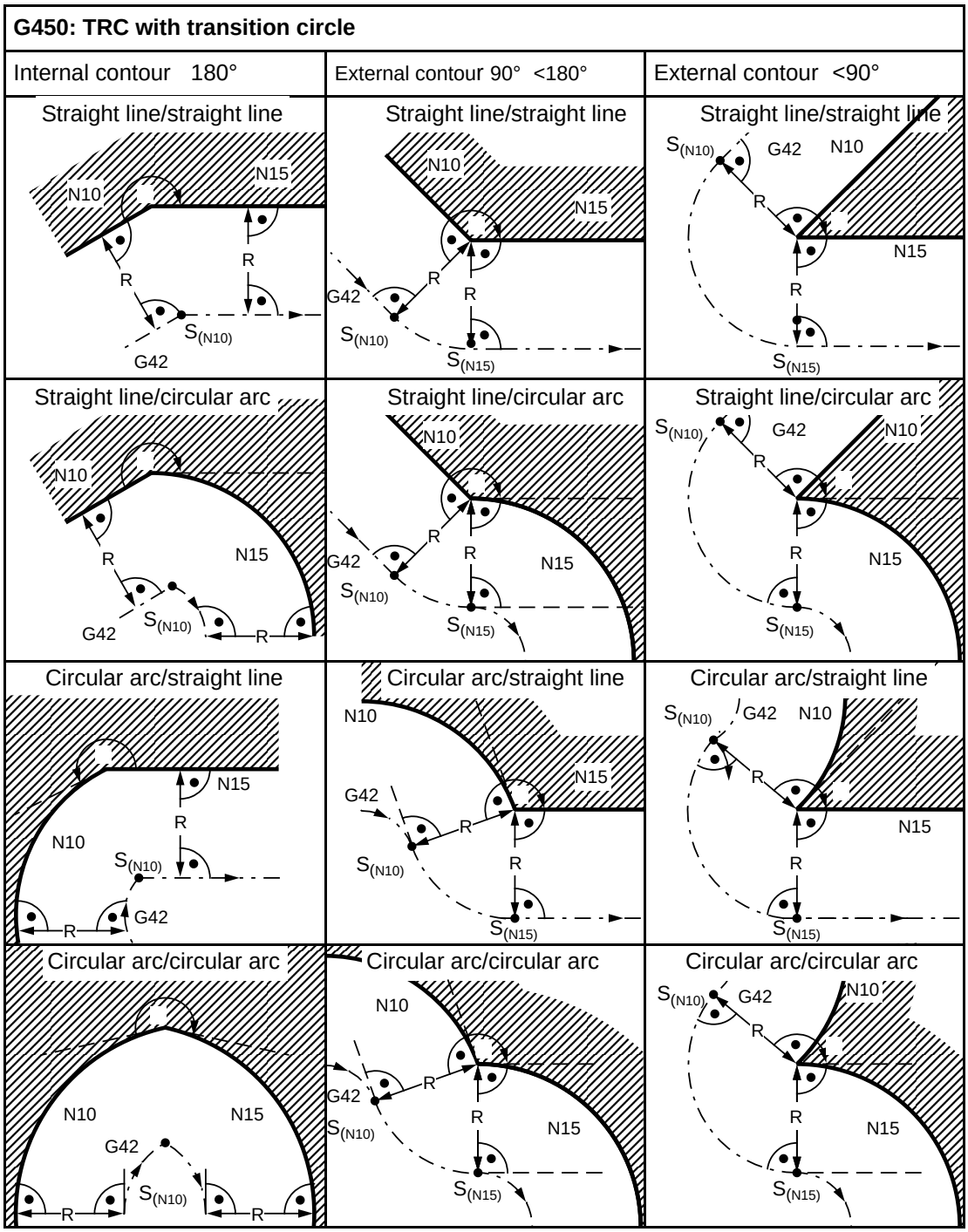
Selection with G42 where <90°

G455: Compensated approach		G456: "Normal position approach"
G450: TRC with transition circle	G451: TRC with intersection	
Str. line/str. line	Str. line/str. line	Straight line/straight line
Str. line/circ. arc	Straight line/circular arc	Straight line/circular arc

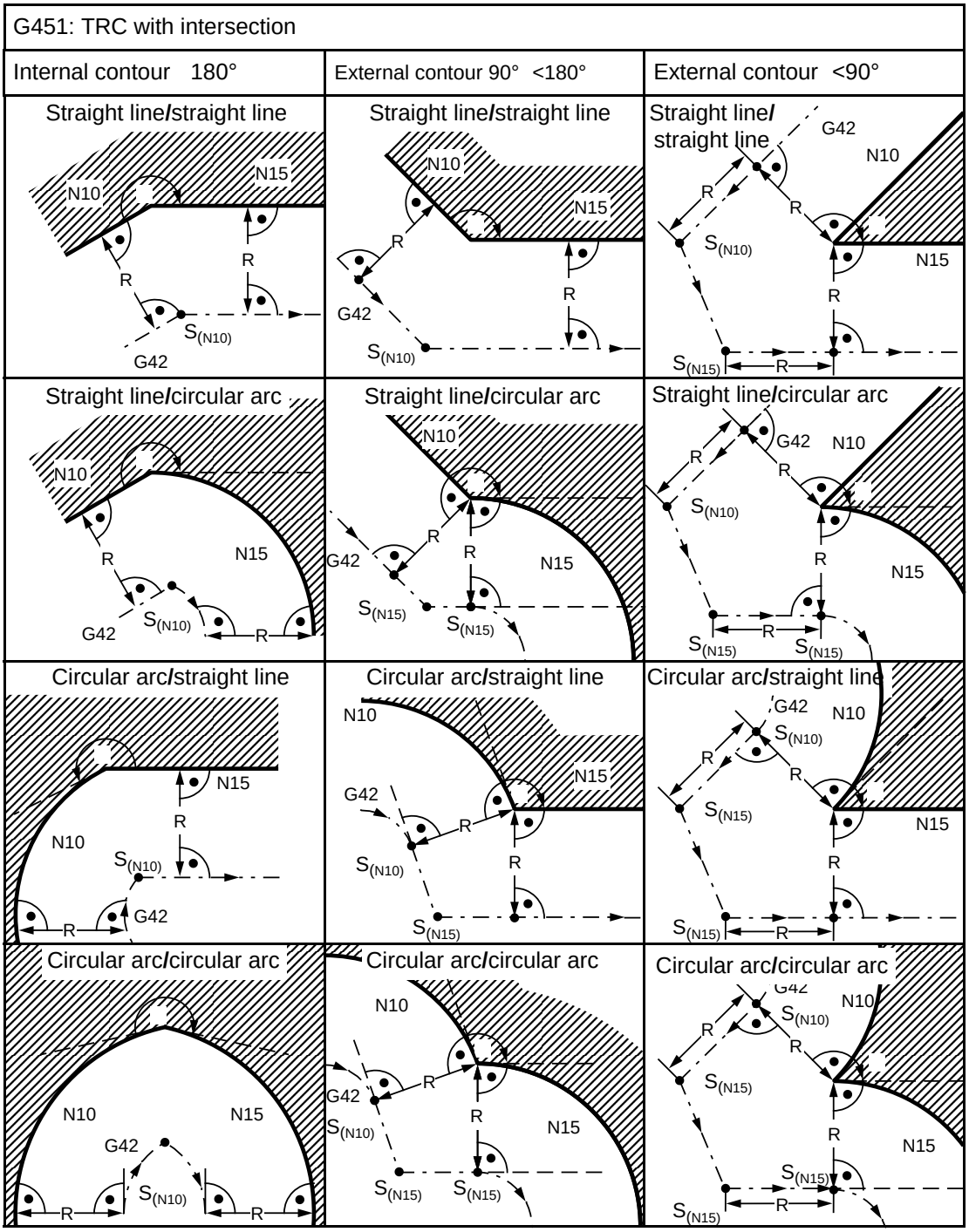


11.4 TRC in the program

When tool radius compensation is selected, the control reads in two further blocks in advance during processing of the current block and calculates the intersection of the compensated paths. During machining the tool path is independent of the TRC selection**/**deselection procedure (G455**/**G456), but dependent on the programmed behaviour at corners (G450**/**G451). The following diagrams show the behaviour of TRC with various transitions.



TRC with transition circle (G450)



TRC with intersection (G451)

Note:

If TRC is selected none of the following can be programmed: @714, G25, G26, G54 ... G57, G58, G59, G92, G74, G16 ... G19, C axis select./deselect. and transformation select./deselect.

Remedy:

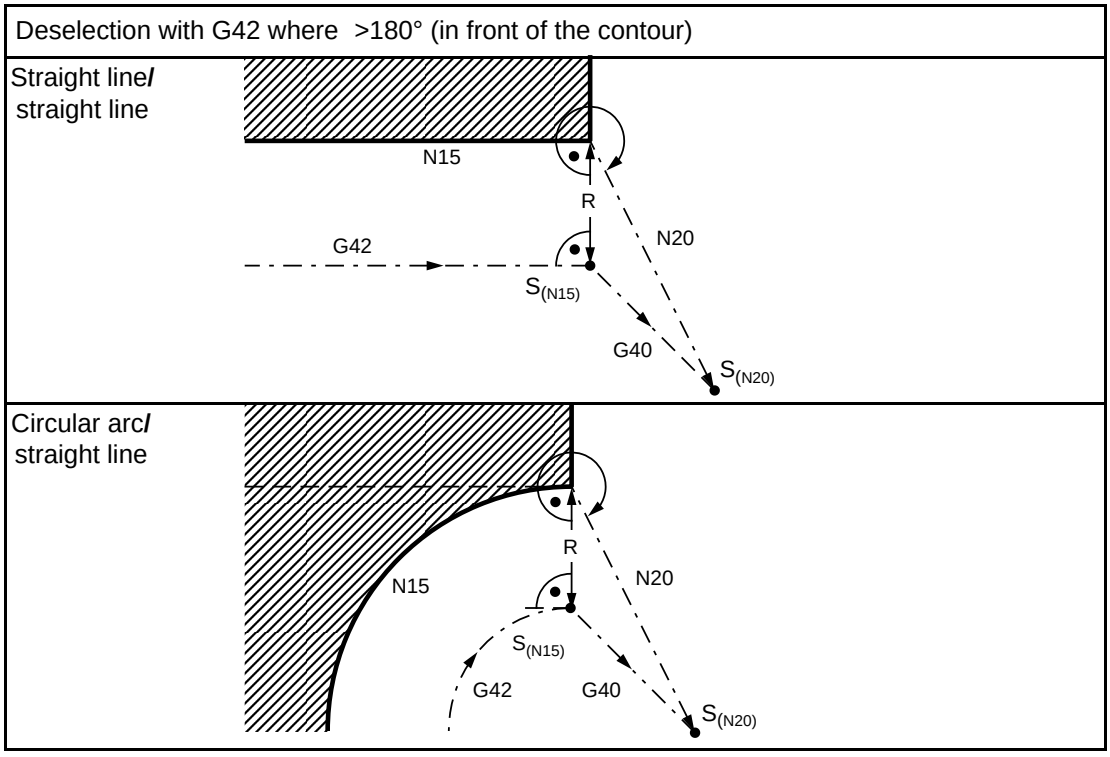
- Program these functions before selecting TRC.
- Deselect TRC, program functions, select TRC.

11.5 Deselecting TRC (G40)

Tool radius compensation (TRC) is deselected with the preparatory function G40.

Tool radius compensation can only be deselected in a program block with G00/G01/G10 or G11. The compensations only have an effect if at least one axis of the selected plane is programmed. The tool offset number D0 corresponds to TRC deselection.

TRC is usually deselected in the angular range 180° and is therefore independent of the set departure and corner procedure (see Sections 11.1 and 11.2). The programmed point is approached directly from the ”normal position” (perpendicular on contour end point with distance tool radius R.



Deselection of TRC where >180°

If the function is deselected in the angular range 180°, the set selection procedure (G455/G456, see Section 11.2) becomes active. If compensated deselection (G455) is selected, the set behavioiur at corners (G450/G451, see Section 11.1) is active. This results in a contour violation (see next example, a < R!).

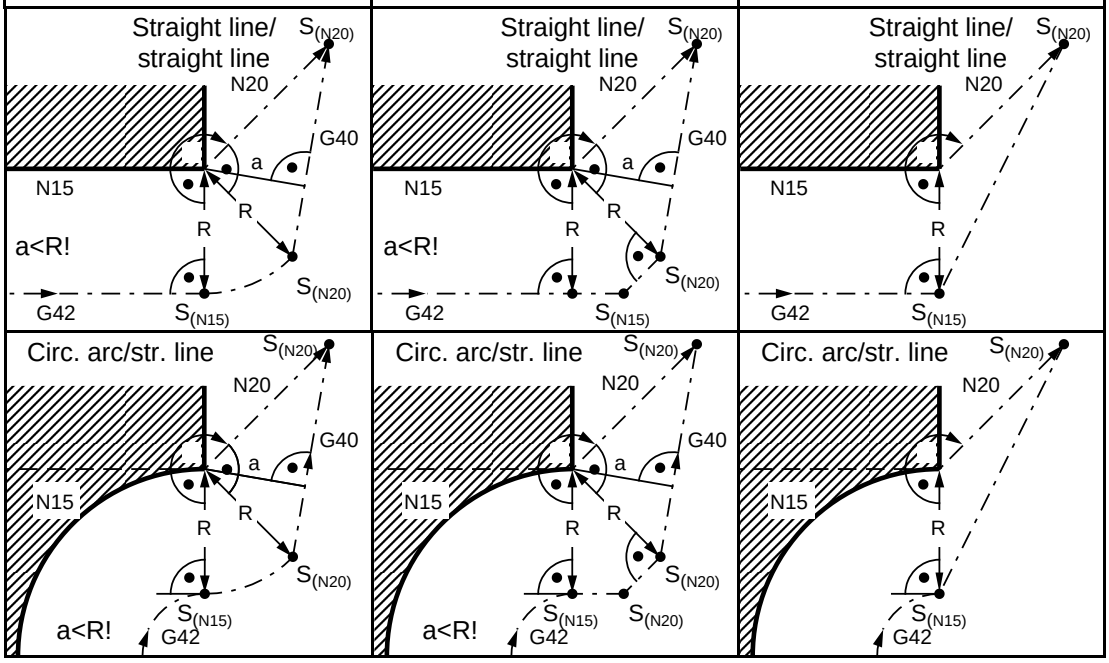
Deselection with G42 where 90° 180°

G455: Compensated exit

G450: TRC with transition circle

G451: TRC with intersection

G456: "Exit from normal position"



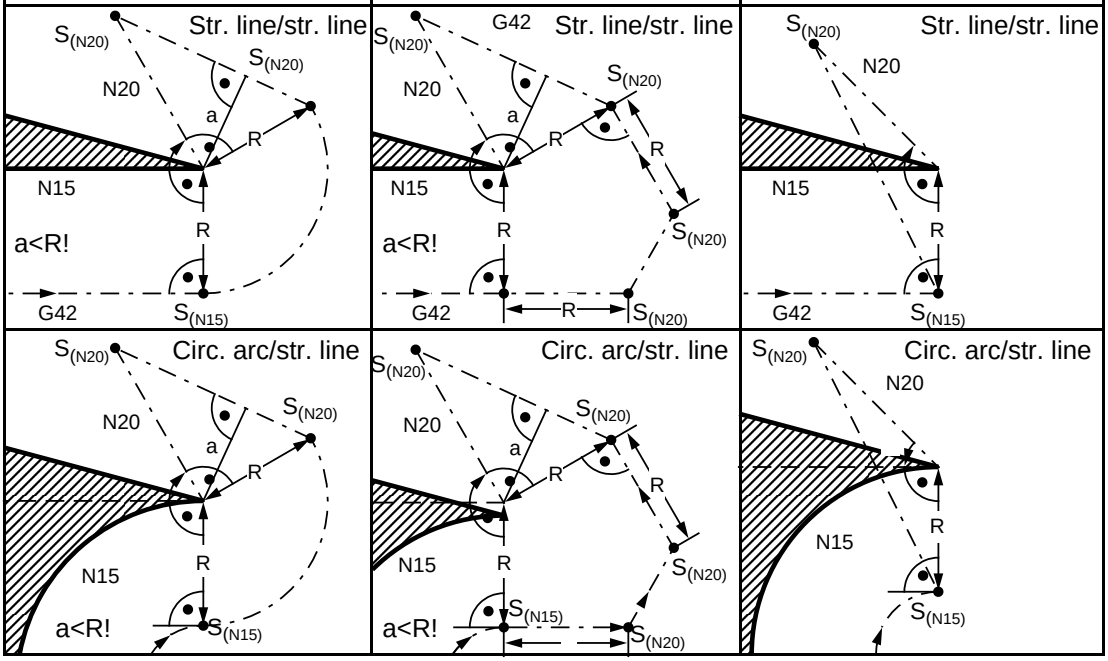
Deselection with G42 where <90°

G455: Compensated exit

G450: TRC with transition circle

G451: TRC with intersection

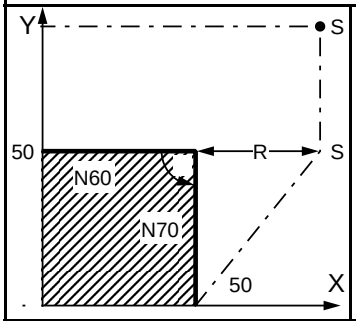
G456: "Exit from normal position"



11.5.1 Deselection of TRC in combination with program end (M30, M02)

Deselection of TRC in combination with program end will give varying results with different combinations (see following diagrams):

Deselection active:



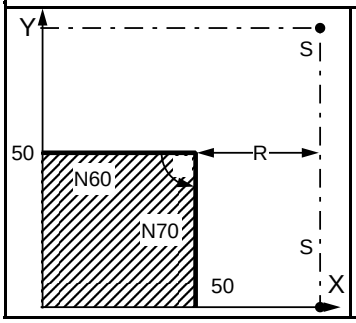
Program examples :

```
N60  X50  Y50  LF
N70  G40  Y0   LF
N90  M30  LF
```

```
N60  X50  Y50  LF
N70  G40  Y0   M30  LF
```

```
N60  X50  Y50  LF
N70  G40  Y0   X50  M30  LF
```

Deselection not active:



Program examples:

```
N60  X50  Y50  LF
N70  Y0   LF
N80  G40  M30  LF
```

```
N60  X50  Y50  LF
N70  Y0   X50  LF
N80  M30  LF
```

TRC in combination with M30, M02

The compensation is deselected with G40 in the last but one block together with at least one axis of the selected plane.

11.5.2 Special characteristics of selection and deselection

G40/G41/G42 without programmed path

G40, G41, G42 can be programmed in a block without programmed paths. However, they first become active when at least one axis of the selected plane has been programmed with a movement.

Example of selection:

```
N10  G01  G17  G41  D07  X..  Y..  LF
```
At the end of this block the compensated path of the selected plane has been reached. Only the radius compensation value is calculated.

```
N15  Z..  LF
```
Tool length compensation is calculated.

oder

```
N10  G17  LF
```
Plane selection

```
N15  G41  D07  LF
```
Compensation selection

```
N20  G01  X..  Y..  LF
```
At the end of this block the compensated path in the selected plane has been reached. Only the radius compensation value is calculated.

```
N25  Z..  LF
```
The tool length compensation is calculated.

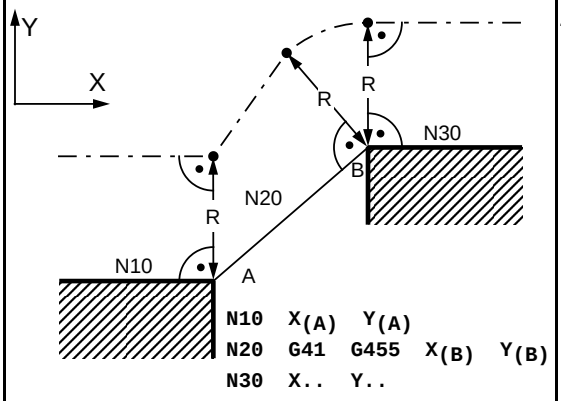
11.6 G41/G42 when TRC is already active

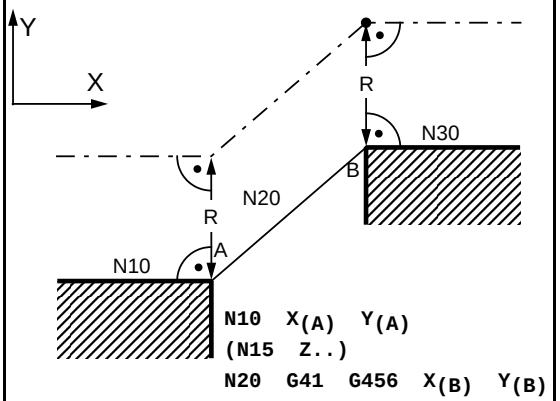
11.6.1 Repetition of the active G function G41/G42

The following happens when G41**/**G42 is programmed when these functions are already active:

- the repetition is ignored if G41**/**G42 is programmed without G455**/**G456
- if G41**/**G42 is programmed with G455, TRC is deselected at the last point with G455 and TRC is selected with G455 at the programmed point
- if G41**/**G42 is programmed with G456, TRC is deselected at the last point with G456 and selected at the programmed point with G456

In this way machining at a contour can be terminated and machining of a new contour can be started without deselecting TRC. Selection**/**deselection procedure in various angular ranges is described in sections ”Deselecting TRC” and ”Selecting TRC”. Behaviour at corners (G450**/**G451) is taken into account when selecting/deselecting.

<table>
<tr><td colspan="2">Repetition of active G41/G42 command</td></tr>
<tr><td>G41/G455: Compensated traversing</td><td>G41/G456: Traverse ”normal position”</td></tr>
<tr><td> 

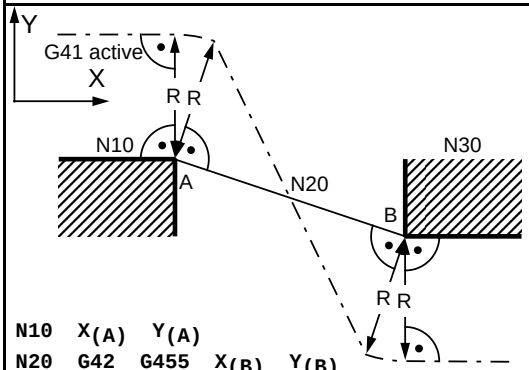
N10 X(A) Y(A)
N20 G41 G455 X(B) Y(B)
N30 X.. Y..</td><td> 

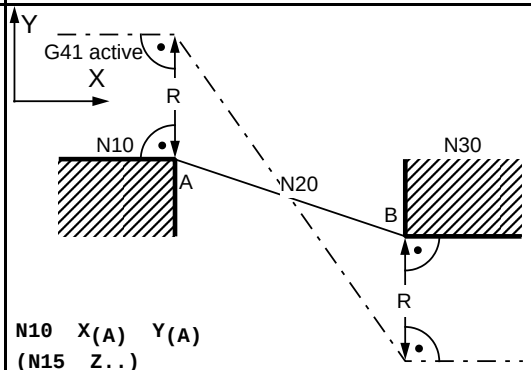
N10 X(A) Y(A)
(N15 Z..)
N20 G41 G456 X(B) Y(B)
(N25 Z..)
N30 X.. Y..</td></tr>
</table>

11.6.2 Changing the direction of compensation G41 G42

If G41/G42 is programmed when already active the following happens:

- If G42 is programmed without G455/G456, TRC is deselected at the last point with G456 and selected at the programmed point with G456
- if G42 is programmed with G455, TRC is deselected at the last point with G455 and selected with G455 at the programmed point
- if G42 is programmed with G456, TRC is deselected with G456 at the last point and selected with G456 at the programmed point. G456 need not be programmed.

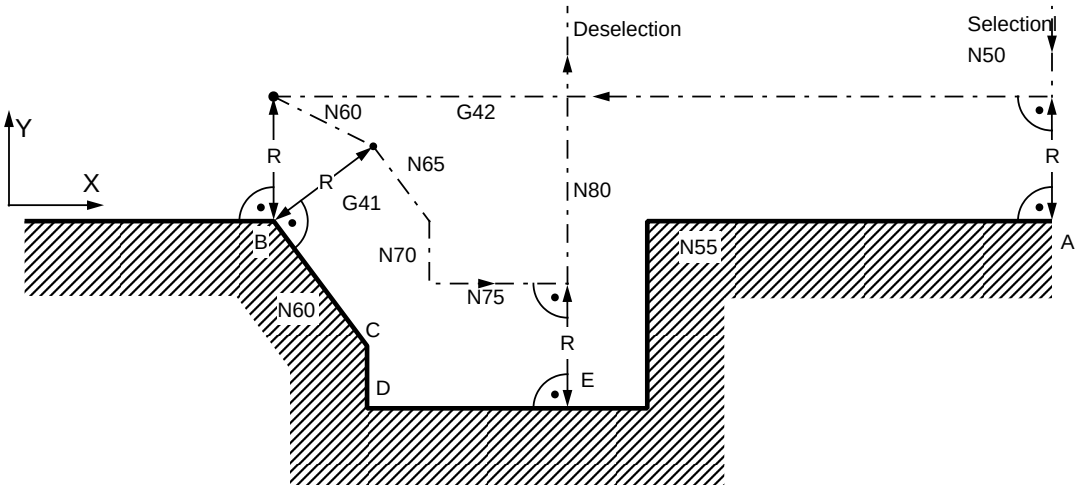
<table>
<tr><td colspan="2">Changing the direction of compensation</td></tr>
<tr><td>G42/455: Traversing with compensation</td><td>G42/G456: Traverse”normal position”</td></tr>
<tr><td> 

N10 X(A) Y(A)
N20 G42 G455 X(B) Y(B)
N30 X.. Y..</td><td> 

N10 X(A) Y(A)
(N15 Z..)
N20 G41 G456 X(B) Y(B)
(N25 Z..)
N30 X.. Y..</td></tr>
</table>

If the compensation is to be altered at a certain point this point must be programmed twice.

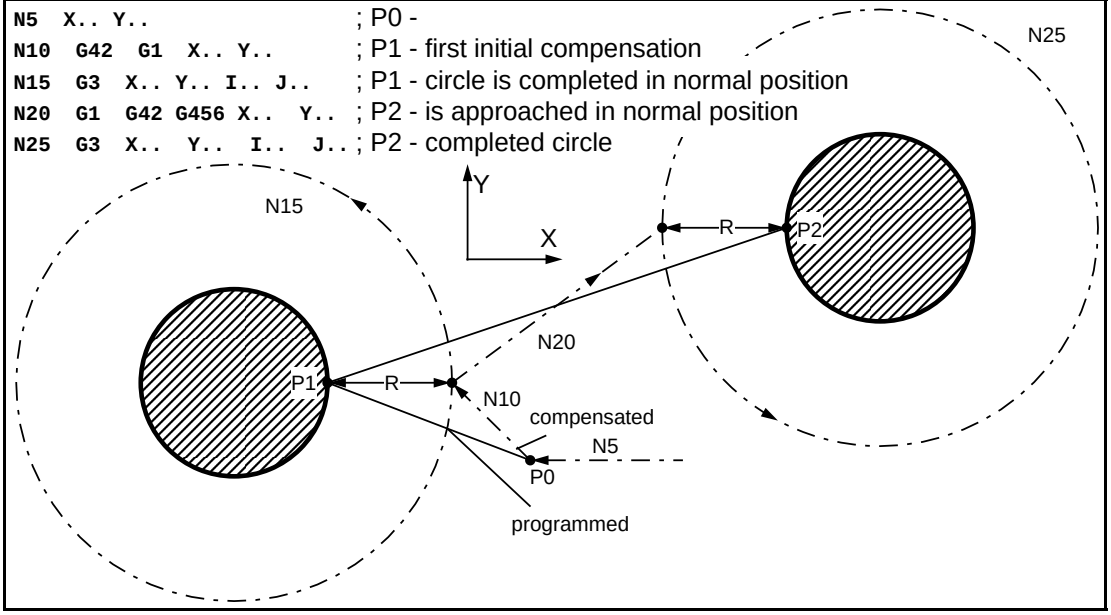
Partial machining of a turned part



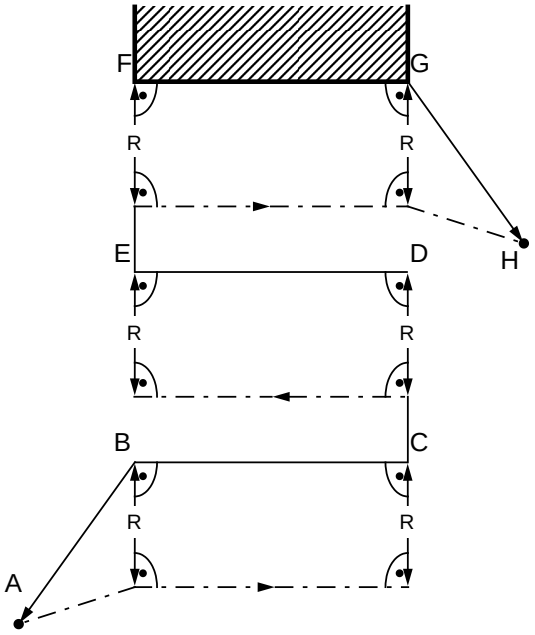

```
N50  G42  X(A)  Z(A)      ; Selection of G42 to point A
N55  G01  X(B)  Z(B)      ; Traverse to point B (starting position for G41 selection)
N60  G41  X(B)  Z(B)      ; Selection of G41 to point B!
N65  X(C)  Z(C)           ; Continued contour programming
N70  X(D)  Z(D)
N75  X(E)  Z(E)
N80  G40  X..  Z..
```

Complete machining of two contour sections

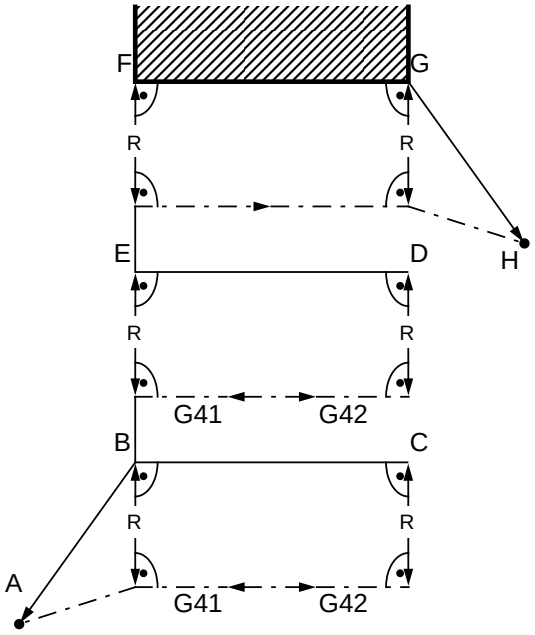
```
N5  X.. Y..                ; P0 -
N10  G42  G1  X.. Y..      ; P1 - first initial compensation
N15  G3  X.. Y.. I.. J..   ; P1 - circle is completed in normal position
N20  G1  G42 G456 X..  Y.. ; P2 - is approached in normal position
N25  G3  X..  Y..  I..  J.. ; P2 - completed circle
```



Line by line milling with tool radius compensation

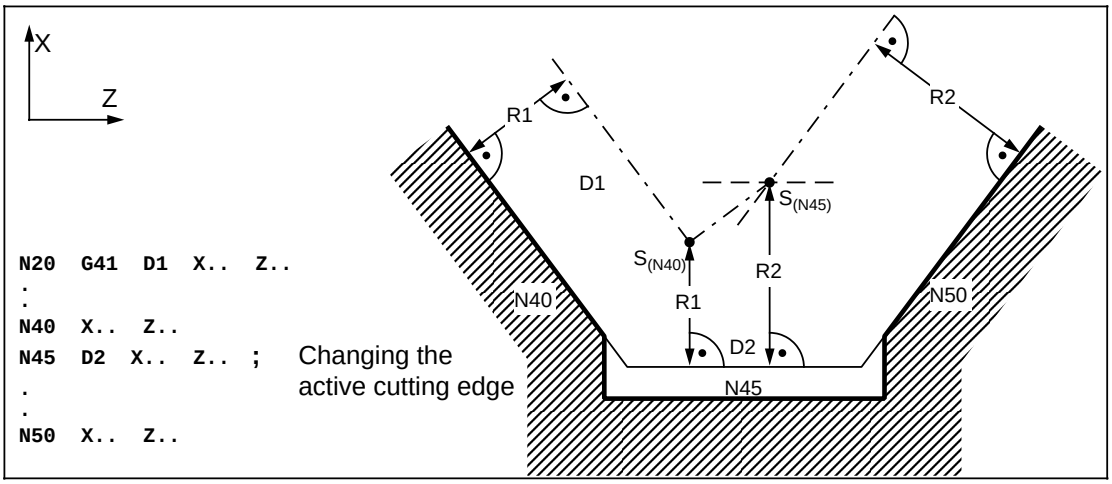



```
N10  X(A)  Y(A)          ;
N20  G42   X(B)  Y(B)     ; Compensation selection
N30  X(C)  Y(C)          ; Mill first line
N40  G41   X(D)  Y(D)     ; Infeed movement
N50  X(E)  Y(E)          ; Mill second line
N60  G42   X(F)  Y(F)     ; Infeed movement
N70  X(G)  Y(G)          ; Mill third line
N80  G40   X(H)  Y(H)     ; Deselect compensation
```




```
N10  X(A)  Y(A)          ;
N20  G42   X(B)  Y(B)     ; Compensation selection
N30  X(C)  Y(C)          ; Mill first line
N40  G41   X(B)  Y(B)     ; Exit
N50  G42   X(E)  Y(E)     ; Infeed movement
N60  X(D)  Y(D)          ; Mill second line
N70  G41   X(E)  Y(E)     ; Exit
N80  G42   X(F)  Y(F)     ; Infeed movement
N90  X(G)  Y(G)          ; Mill third line
N99  G40   X(H)  Y(H)     ; Deselect compensation
```


11.7 Changing the tool compensation number (D...)



Changing the compensation number

When the offset number is changed the following applies:

1. The block start intersection is calculated with the first tool radius.
2. The block end intersection is calculated with the second tool radius.

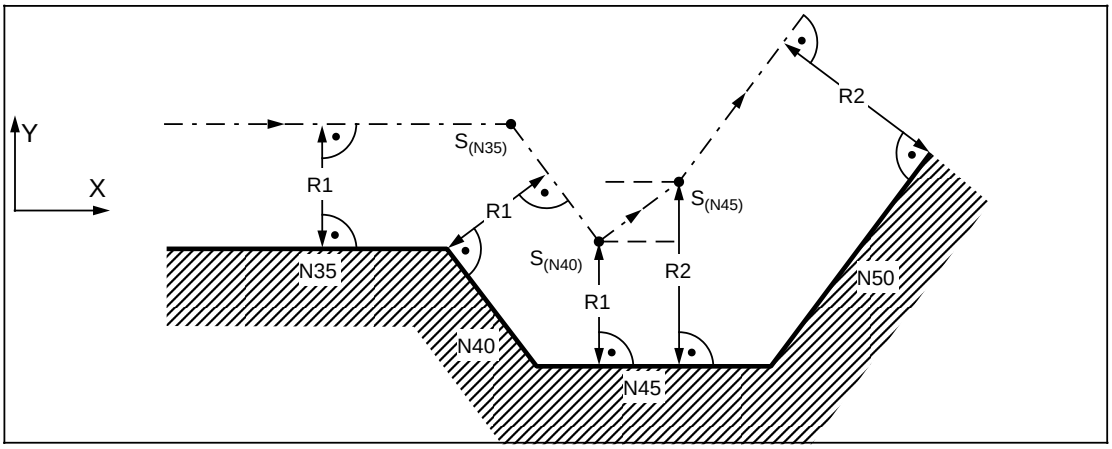
Use:

- Tool with several cutting edges
- Cutting edge 1 (D1) is in cutting edge position 3
- Cutting edge 2 (D2) is in cutting edge position 4

Note: The offset number can only be altered in linear blocks.

11.8 Changing compensation values

The compensation values can be changed via the operator panel, the RS232 interface or via the PLC as an external tool offset. If the compensation values are changed in the program, they do not become active until two blocks after the compensation change has been programmed. If the changes are to become active immediately, the compensation values can only be entered in the NC stop state.

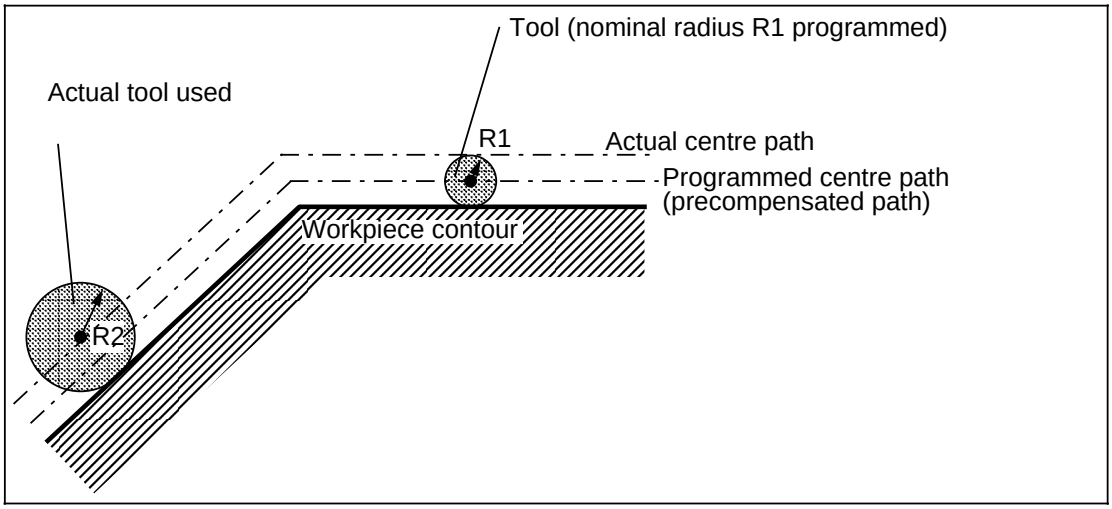


11.9 Procedure with precompensated contours

If the program has already been written with the nominal radius of the tool (centre point path), the tool radius compensation need only contain the difference between the nominal radius and the actual tool radius used.

Negative compensation values for the tool radius can also be used here.

Example for milling:



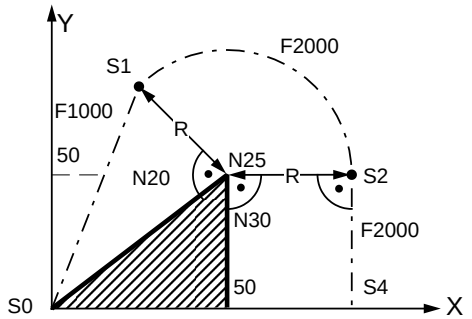
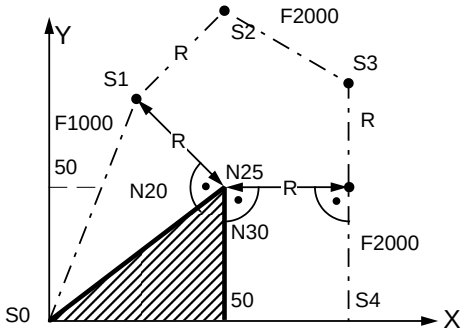
Procedure with precompensated contours

Radius value to be entered in the offset memory:

R2 – R1 (Input positive, as R2 > R1)

The entered radius value is negative when R2 < R1.

11.10 Effect of G functions, M/auxiliary functions and feedrates

G450: TRC with transition circle	G451: TRC with intersection
	
N5 G450 **N10 G01 X0 Y0 F1000** **N20 G41 X50 Y50 H1 (S1)** **N25 M3 S2000 Z-5 F10 H2 (S1)** **N30 Y0 F2000 H3 (S4)**	**N5 G451** **N10 G01 X0 Y0 F1000** **N20 G41 X50 Y50 H1 (S1)** **N25 M3 S2000 Z-5 F10 H2 (S1)** **N30 Y0 F2000 H3 (S4)**

Effect of auxiliary functions and feedrates

The tool radius compensation generates one or several intermediate blocks for machining external corners. An additional block can be programmed outside the TRC plane (N25) between the blocks programmed in the TRC plane (N20 and N30). The auxiliary functions have the following effect:

- The auxiliary functions in the first programmed block of the TRC plane (H1 in N20) are output at point S0.
- The auxiliary functions or infeed movements in the programmed block outside the TRC plane (H2, M3, S2000, Z-5 in block N25) are output at point S1.
- No auxiliary functions are output in the intermediate blocks generated by the tool radius compensation.

  They are traversed with the feedrate and the G functions of the next block (F2000 in N30).

  If no feedrate is programmed in the next block, the last feedrate to be programmed is used. This can also be the feedrate programmed in a block outside the TRC plane (F10 in N25).

  Any non-modal G functions in the next block (G09) are active in every intermediate block.
- The auxiliary functions in the second programmed block of the TRC plane (H3 in N30) are output after the generated intermediated blocks (S2 for G450, S3 for G451).

You can define in the MD whether the auxiliary functions are to be output before or during the axis movement. Independently of this, there are M functions (M00, M01, M02, M18, M30) which do not become active until block end.

11.11 TRC with blocks outside the compensation plane

When TRC is active, up to 3 blocks (N30, N35, N40) can be ”programmed” outside the compensation plane and up to 2 blocks in the compensation plane (N25, N35).

If more than 3 blocks are programmed outside the compensation plane, alarm 3021, "Contour violation with TRC" is triggered.

Note: With SW 1 and 2 only one block can be "programmed" outside the compensation plane. If more than 1 block is programmed outside the compensation plane, alarm 3021, "Contour violation with TRC" is triggered.

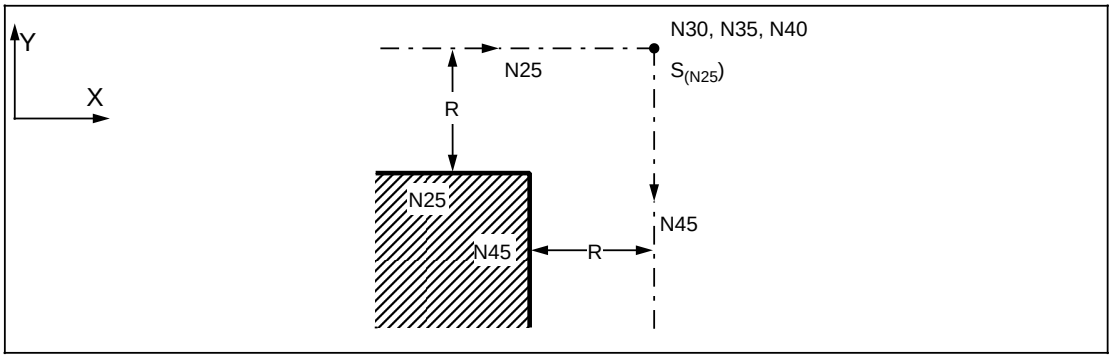
One of the following can be a block outside the compensation plane:

- An "auxiliary function block" e.g. M08, H1
- A block with "path = 0"; a block in the compensation plane (X0) is treated like a block outside the compensation plane (Z0).
- Block not in the compensation plane e.g. Z-5
- A calculation block with comments

Example:

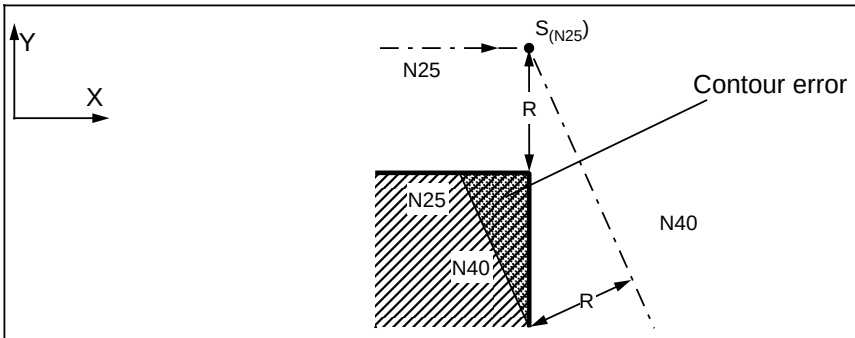
`N25 G91 X200` $\llcorner_F$	(Block in compensation plane)
`N30 M08 H1` $\llcorner_F$	(Block outside the compensation plane)
`N35 X0` $\llcorner_F$	(Block in compensation plane with path = 0 is treated like a block outside the compensation plane)
`N40 Z-5` $\llcorner_F$	(Block outside the compensation plane)
`N45 Y-100` $\llcorner_F$	(Block in compensation plane)

N30, N35, N40 are executed at point $S_{(N25)}$.



TRC with block outside compensation plane

The effect of this alarm can be influenced in the machine data, contour violations may result if traversing is continued after the alarm.



Example:

`N25 G91 X200` L_F	Block in compensation plane
`N30 M.. (or H..)` L_F	Auxiliary function block
`N35 X0 (or Y0 or Z0)` L_F	Block with path = 0
`N40 Z-5` L_F	Block outside the compensation plane
`N45 M.. (or H.. or X0, Y0 or Z..)` L_F	4th block outside the compensation plane alarm 3021
`N50 Y-100` L_F	Block in compensation plane

TRC with more than three blocks outside the compensation plane

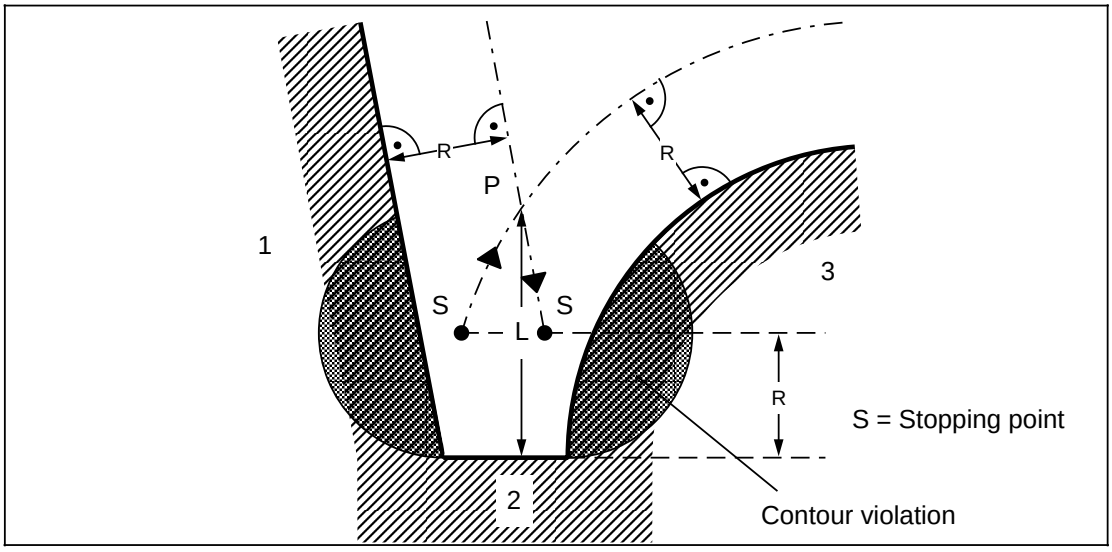
11.12 Contour violation with tool radius compensation

Contour violations can occur with three successive contour elements and a tool radius.

The tool centre line is an equidistant with the distance tool radius (R) to the programmed path. The equidistants from the first and third contour elements can form an intersection (P). If the length of the perpendicular (L) from the second contour element to intersection P is longer than the tool radius, contour violation will occur. The control outputs alarm 3021 ”Contour violation with TRC” during processing.

The effect of alarm 3021 can be influenced

via machine data



TRC special case: Intermediate block < compensation value

11.13 Contour violation without TRC on turning machines

Turning tools (type P1=1 ... P1=8) are usually measured with reference to the theoretical tool tip P (see Section 10.4).
If TRC is not used, dimensional errors can occur because of the tool nose radius. Such errors can be avoided by selecting TRC.

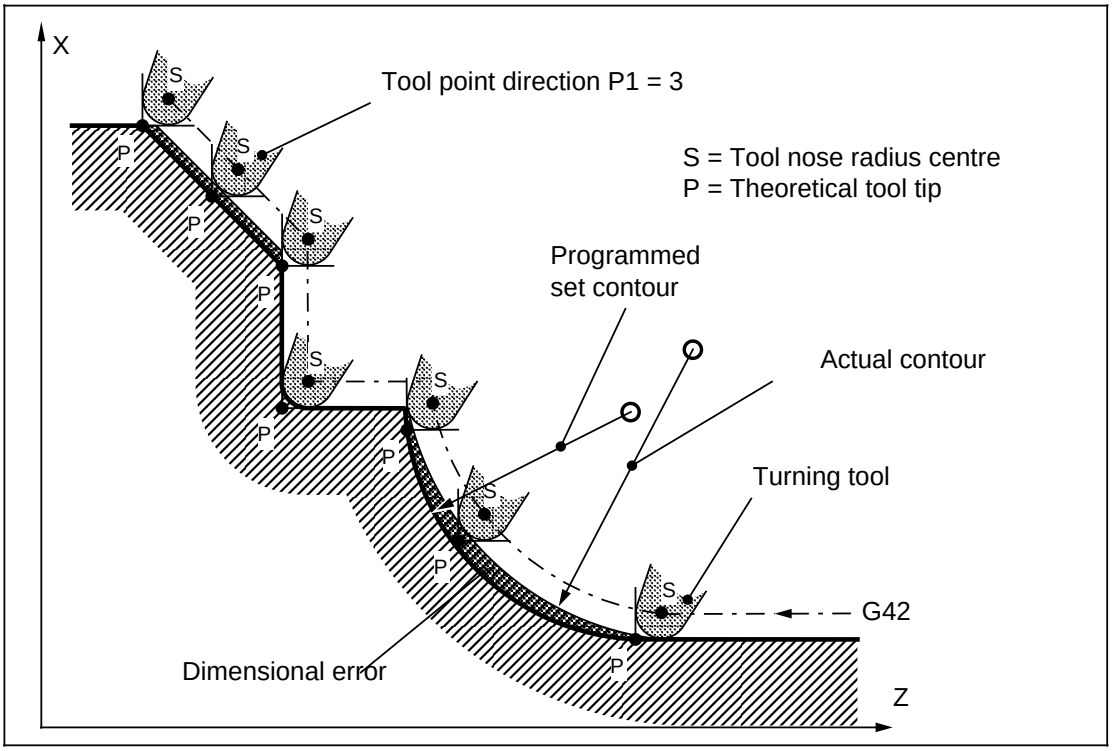
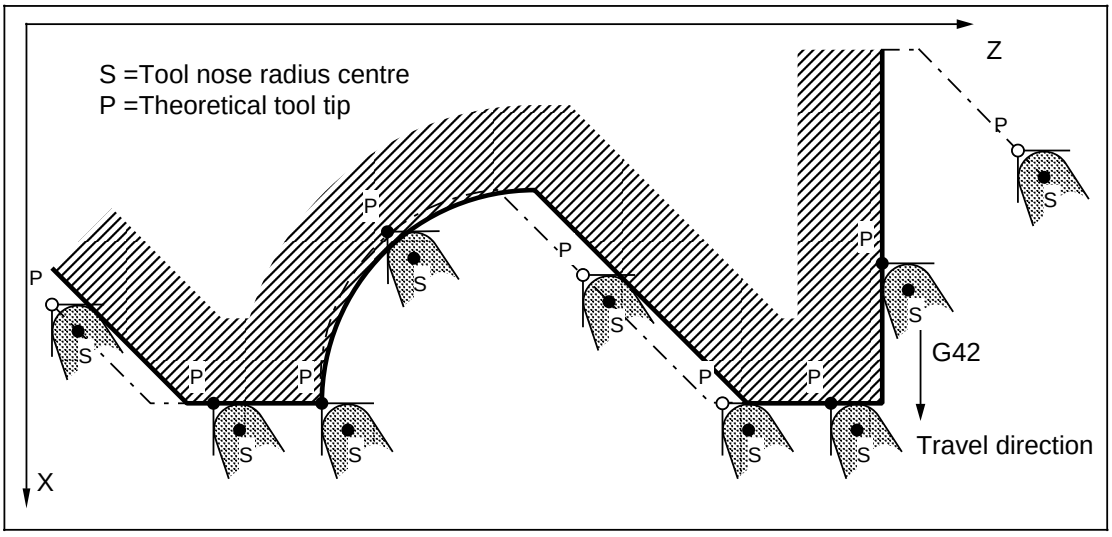


*Fig.: Dimensional error resulting from the tool nose radius of the cutting tool **without** TRC*

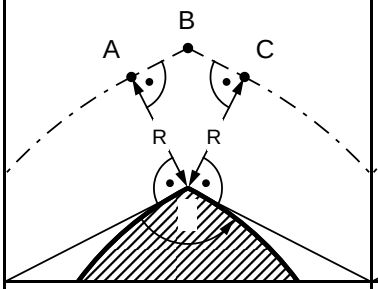
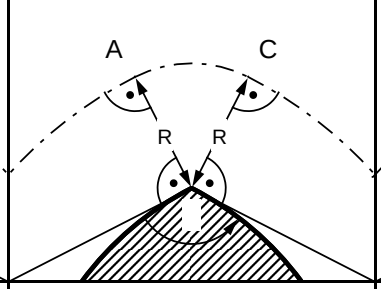
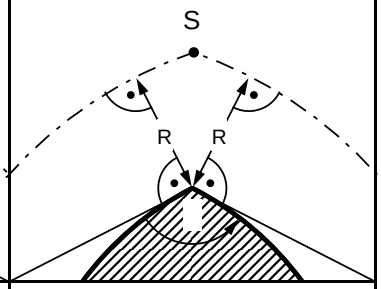
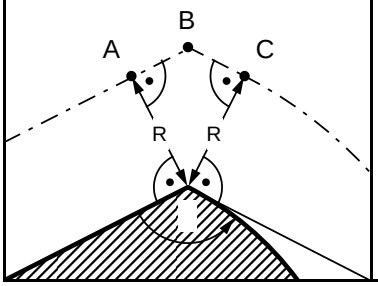
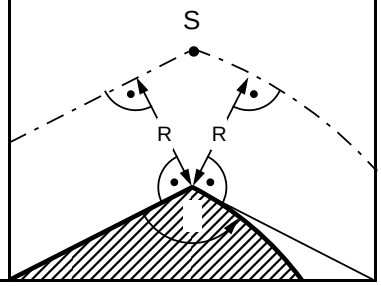


*Flg.: **No** dimensional error resulting from tool nose radius of turning tool **with** TRC.*

11.14 Circle transitions with obtuse angles

When programming exterior contours with circle transitions and obtuse angles (90 < < 180°), one or two intermediate blocks are inserted at the circle transitions (straight line/circular arc, circular arc/straight line and circular arc/circular arc) (see Section 11.4, ”TRC with transition circle” and ”TRC with intersection”). If the angle is almost 180°, the traversed paths in the intermediate blocks become very small and can stop the continuous path mode. To avoid this, the control can suppress the intermediate blocks. It either switches from ”TRC with intersection” to ”TRC with transition circle” which avoids an intermediate block, or it approaches the intersection of both blocks. No intermediate block is generated if this is done.

Suppression of intermediate blocks at circle transitions with obtuse angles

Normal case	Switchover to transition circle	Intersection approach
Circular arc/circular arc 	Circular arc/circular arc 	Circular arc/circular arc 
Straight line/circular arc 	—	Straight line/circular arc 

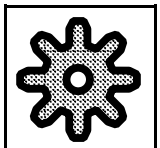
The distance form A to C is compared with the distance defined on start-up to determine whether switchover will take place. If the distance A to C is smaller than the defined distance, an intersection is traversed (applies to all circle transitions with ”TRC with transition circle” G450 and ”TRC with intersection” G451). If the distance A to C is less than double the defined distance, the control switches to ”TRC with transition circle” G450 with ”TRC with intersection” G451 and transition circular arc/circular arc.

Caution:

When an intersection is approached, the programmed circular arc(s) is/are lengthened. If the circle is lengthened to exceed 360°, only the section above 360° is traversed.

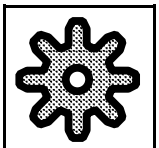
Remedy:

Divide such a circular arc into several sections.

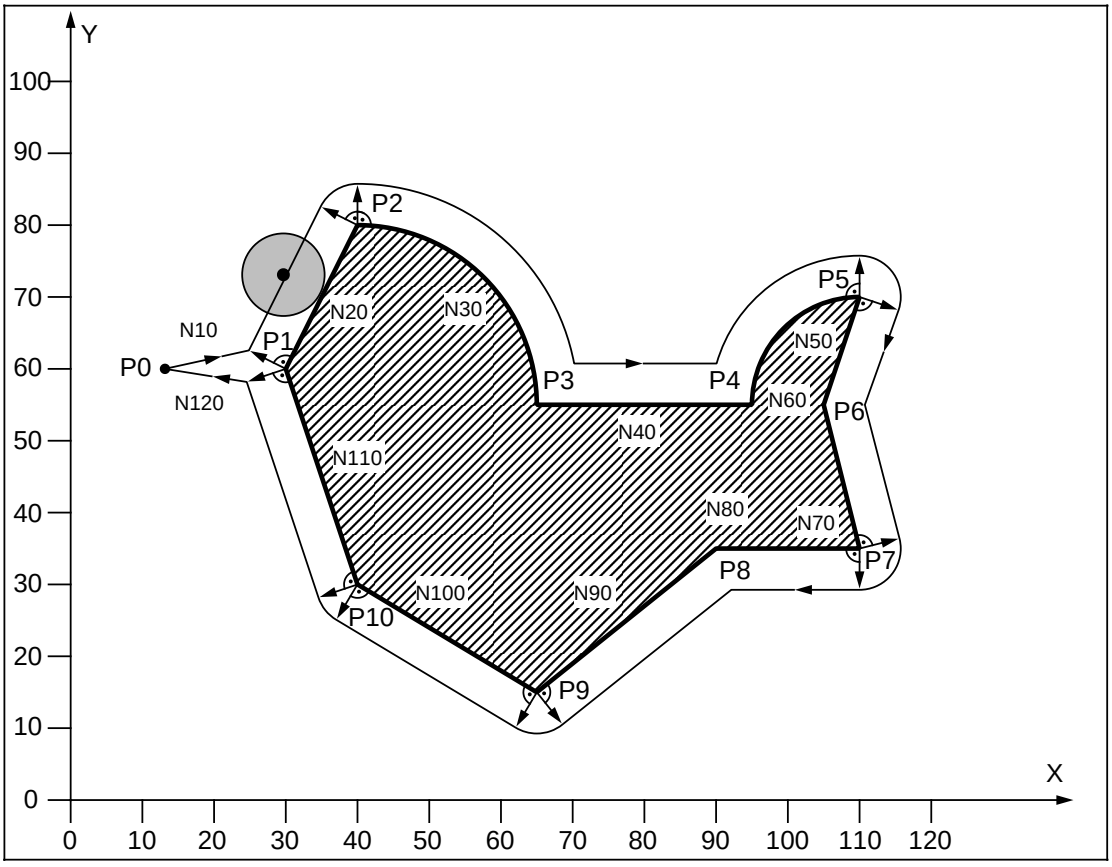


The distance for switchover

is defined in the MD.



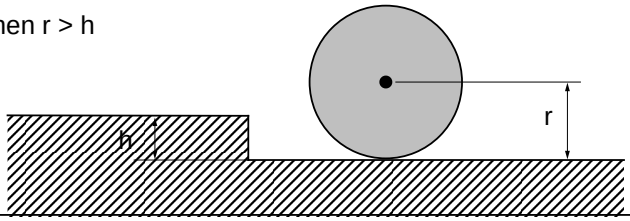
11.15 Example



Complete machining of a workpiece with active TRC

```
N1  G0  X15  Y60  Z..                                    ; Approach starting point (P0)
N10  G41  G450  G1  G17  G90  D1  X30  Y60  F400  S80  M3 ; Transition circle
N20  X40  Y80
N30  G2  X65  Y55  I0  J-25
N40  G1  X95
N50  G2  X110  Y70 I15  J0
N60  G1  X105  Y55
N70  X110  Y35
N80  X90
N90  X65  Y15
N100  X40  Y40
N110  X30  Y60                                          ; (P1)
N120  G40  X15  Y60                                     ; Terminate compen. mode (P0)
N130  G0  Z50  M30
```

Note: Contour is violated when r > h



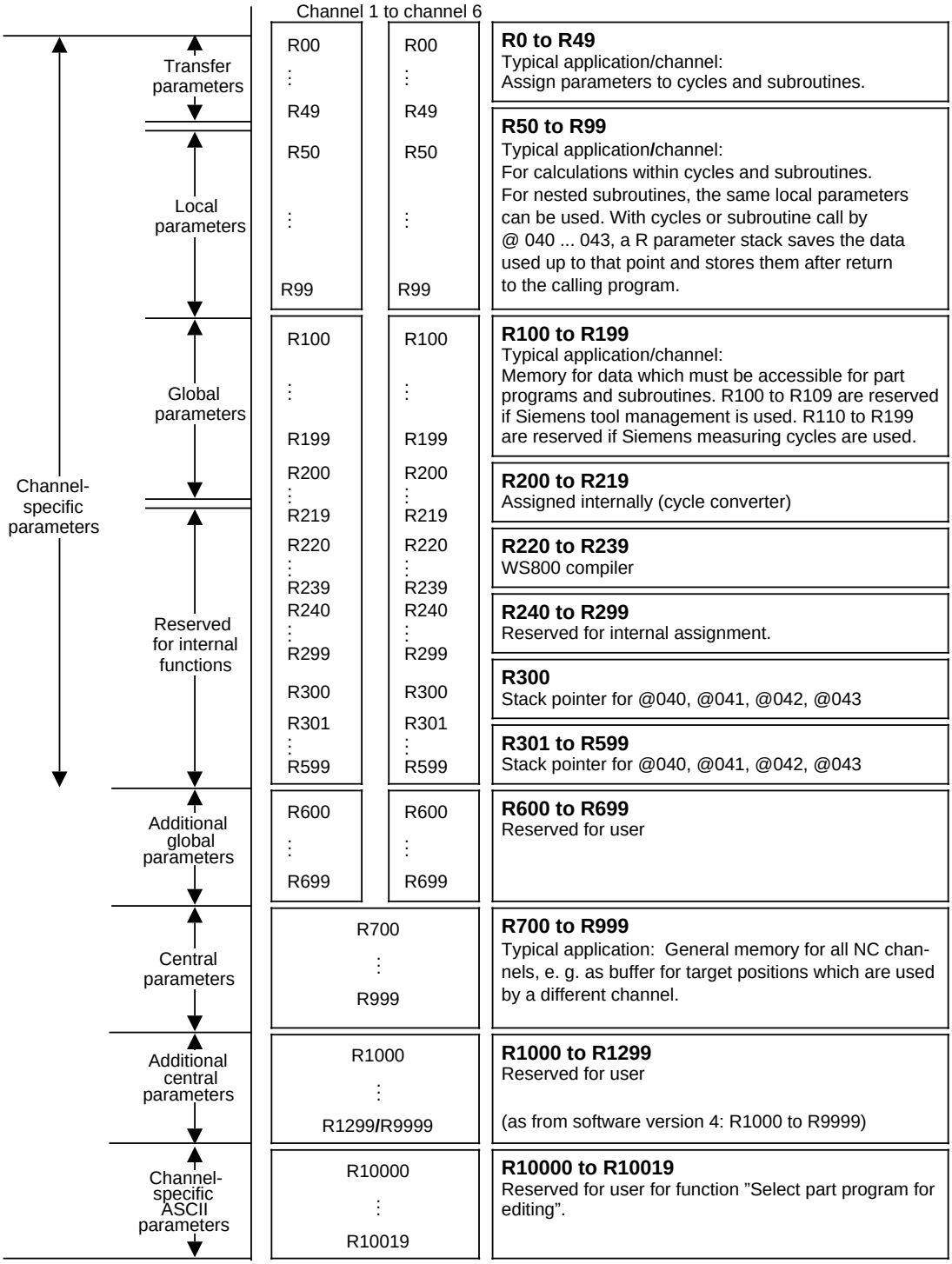
END OF SECTION

12 Parameters

12.1 Parameteric 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 4 digits.

Structuring and application of R parameters:



All addresses can be assigned a parameter instead of a value.

```
N5 Z = R5 LF
```

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

Incorrect:

```
R1= 51120, 98
G = –R1
```

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

12.2 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
N1...
.
.
N37  R1=10.  R29=–20.05  R5=50.  LF        (Parameter definition)
N38  L51  P2  LF                           (Call of subroutine L51, 2 passes)
N39  M02  LF

L51
N1 Z=–R5  B=–R1  LF
N2 X=–R29  LF
.
.
N50  M17  LF
```

12.3 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
Definition	R1 = 100.
Assignment	R1 = R2
Negation	R1 = – R2
Addition	R1 = R2+R3
Subtraction	R1 = R2 – R3
Multiplication	R1 = R2 * R3
Division	R1 = R2 **/** R3

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 LF
```

Calculation using numbers and parameters

Addition and subtraction of numbers and parameters in conjunction with addresses

It is possible to add a parameter to the value of an address or to subtract it from it. Calculation signs must be written. The ”=” sign is not obligatory with the sequence ”Address, numerical value, parameter”.

```
Y10 + R100
```
or
```
Y = 10 + R100
```

The equals sign must be used with the sequence ”Address, parameter, number”.

```
Y = R100 + 10
```

Note: Calculation operations with more than two components are not possible in conjunction with addresses, e.g. X = 100–R2+R3 causes the alarm ”General programming error”.

Note: Calculation operations with more than two components are not possible in conjunction with addresses, e.g. X = 100–R2+R3 causes the alarm ”General programming error”. The operation R1 = 5–3 is possible.

Example:

```
N35  R1 = 9.7 R2 = – 2.1 LF
N40  X = 20.3+R1  LF
N45  Y = 32.9–R2  LF
N50  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
```

Several separate equations can be programmed in one block.

```
R1 = R2+ 23  R50 = R37 * 3  R99 = R27 / R13  LF
```

12.4 Parameter string

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

```
R1=R2+R3–R4·R5 / R6.....
```

The calculations are performed as follows:

Step 1 R1 = R2
Step 2 R1 = R1+R3
Step 3 R1 = R1–R4
Step 4 R1 = R1·R5
Step 5 R1 = R1 / R6

Step 1 R1=R2
Step 2 R1=R2+R3
Step 3 R1–R4
Step 4 R1·R5
Step 5 R1 / R6
R1

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

Example:

```
R1=R2+10.5–P3·R5/R6...
```

The result parameter must be an R parameter.

P3:
P Address of pointer
3 Pointer to R parameter R3, i.e. the contents of R3 are the address of an R parameter whose value is included in the parameter string.

Any number of arithmetic operations may be performed in a block, e.g. multiplication, parameter string, addition, etc.; the maximum permissible block length of 120 characters must not however be exceeded. The individual operations are performed in the programmed sequence.

Value range: Minimum value: $1 \cdot 10^{-8}$
Maximum value: 99999999.
Display: Floating point (±.8) to (±8.)

12.5 Programming examples with parameters

Parametes for subroutines without values

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

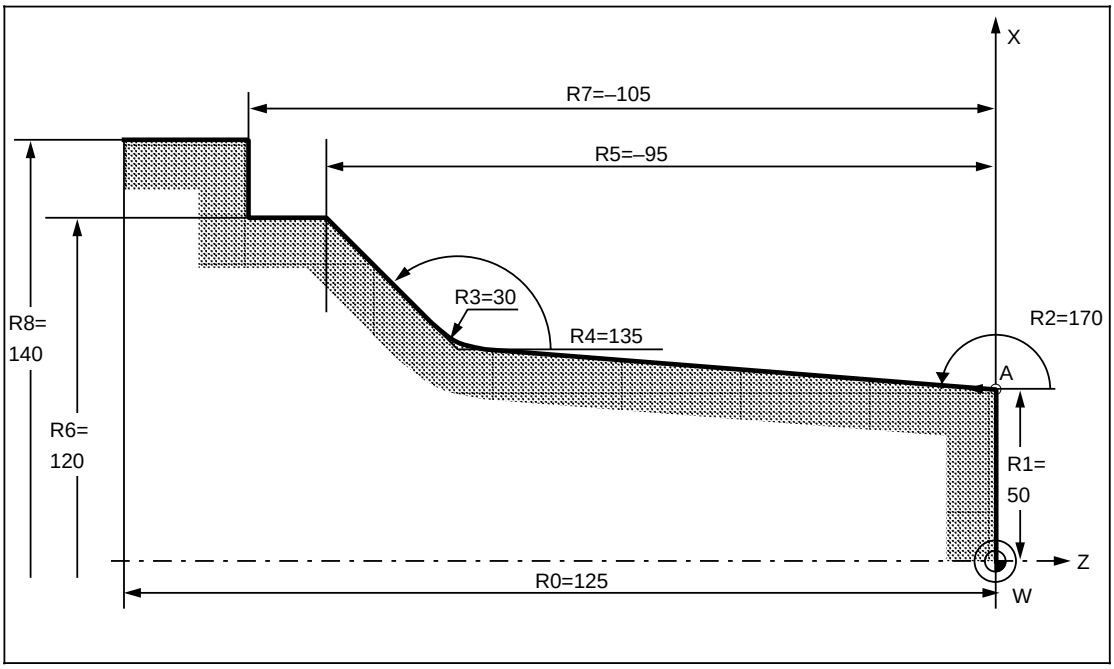
These parameters are used in the subroutine instead of numerical data.

The number of parameters is limited by the block length.

When the subroutine is called the parameters used must have the correct values.

Values are assigned to the parameters in the main program.

Example:



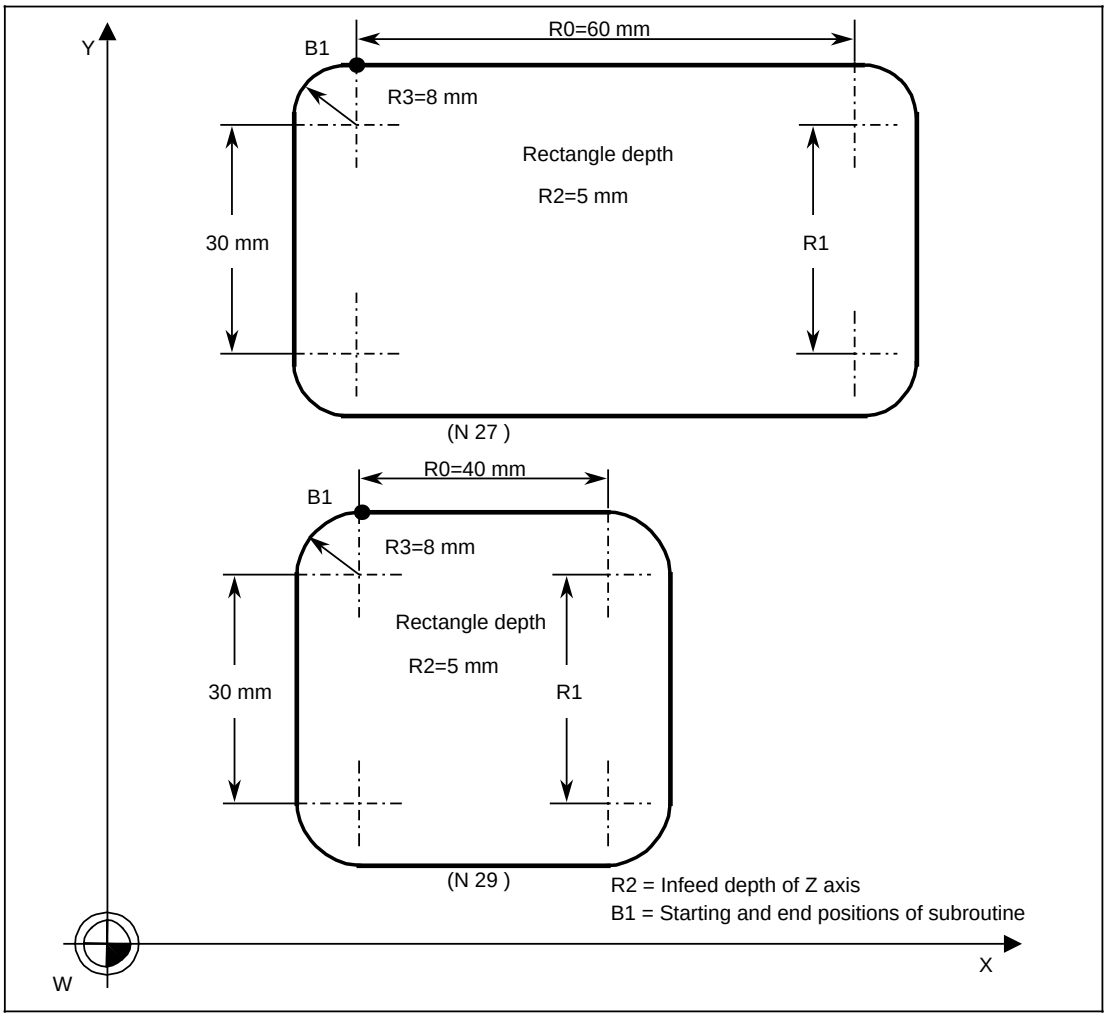
The contour to be cut is stored in a subroutine

```
%250
N05  X..  Z..  LF
N10  L47 P1  R0=125  R1=50  R2=170  R3=30
     R4=135  R5=–95  R6=120  R7=–105  R8=140  LF

L47
N30  G90  Z=0  LF
N35  X=R1  LF
N40  A=R2  Y=R4  X=R6  Z=R5  B=R3  LF
N45  Z=R7  LF
N50  X=R8  LF
N55  Z=R0  M17  LF
```

Example: Rectangle

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.

```
N26 G90 X . . . Y . . . LF     (First starting point of  current program)
N27 R0=60 R1=30. R2=5. R3=8. L46 P1 LF
N28 G90 X . . . Y . . . LF                    (Second starting point)
N29 R0=40. L46 P1 LF

L 46
N05  G01  G91  Z=-R2  LF
N10  G64  X=R0  LF
N15  G02  X=R3  Y=-R3  J=-R3  LF
N20  G01  Y=-R1  LF
N25  G02  X=-R3  Y=-R3  I=-R3  LF
N30  G01  X=-R0  LF
N35  G02  X=-R3  Y=R3  J=R3  LF
N40  G01  Y=R1  LF
N45  G02  G60  X=R3  Y=R3  I=R3  LF
N50  G01  Z=R2  LF
N55  M17  LF
```



Rectangle

Example: Machining of internal semicircle

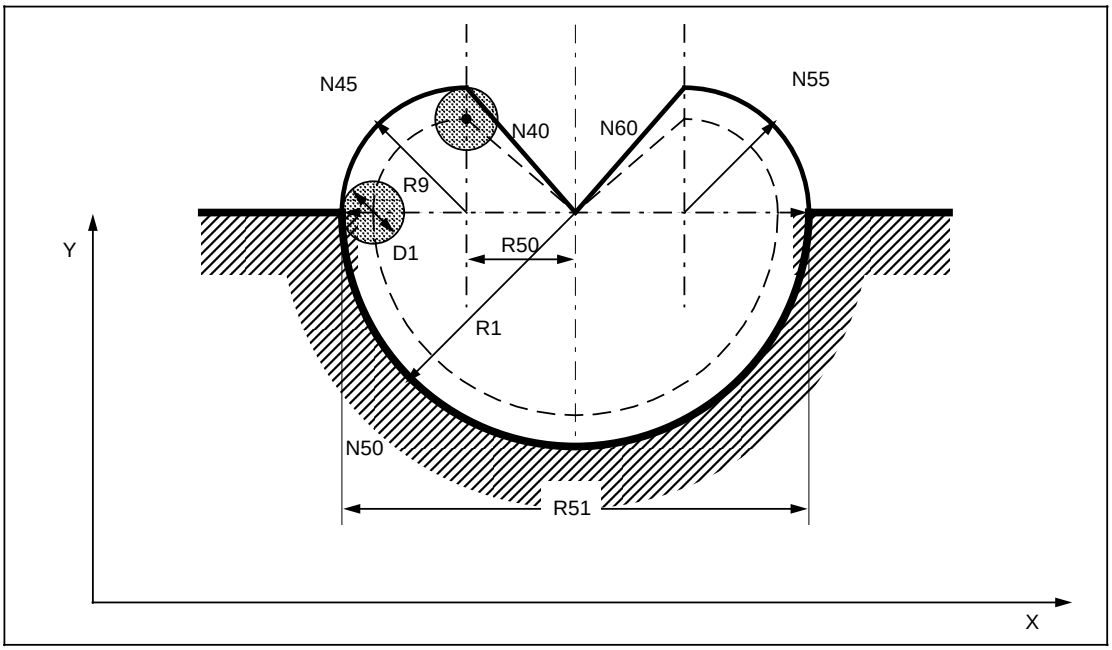
The subroutine below is used for roughing and finishing of a semicircle. The contour radius and the approach circle radius can be varied by means of parameters. The difference between the actual and specified workpiece sizes can be checked after each pass. This difference is then entered as additive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

Subroutine call L11 in the part program

```
%5873
N05...
N10  R1=50. R9=10 L11 P1 LF
N15... L F
```

Subroutine

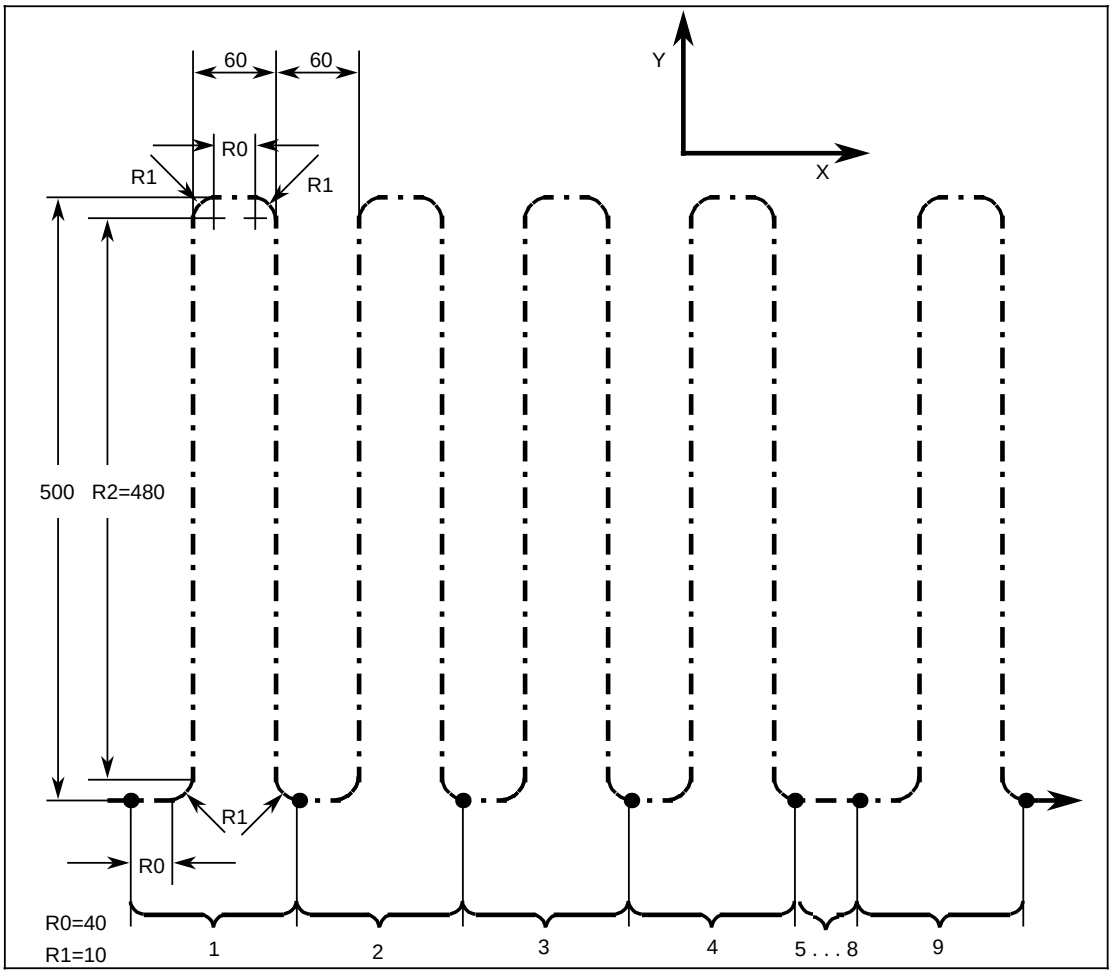
```
L11
N30  R50=R1-R9  LF                                  (Calculation of approach circle)
N35  R51=2·R1  LF                                   (Definition of semicircle)
N40  G00  G64  G91  G41  X=-R50  Y=R9  F500  D1  LF  (Position approach circle)
N45  G03  X=-R9  Y=-R9  U=R9  LF                     (Approach the contour)
N50  X=R51  U=R1  LF                                (Machining)
N55  X=-R9  Y=R9  U=R9  LF                          (Departure from contour)
N60  G00  G40  X=-R50  Y=-R9  LF                    (Positioning)
N65  M17  LF                                        (Subroutine end)
```



Machining of internal semicircle

Example: Line-by-line milling

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



Calling L34 in higher-level program:

```
.
N80  L34  P9  R0=40.  R1=10.  R2=480.  F200  LF
:

L34
N05  G01  G64  G91  X=R0  LF
N10  G03  X=R1  Y=R1  J=R1  LF
N15  G01  Y=R2  LF
N20  G02  X=R1  Y=R1  I=R1  LF
N25  G01  X=R0  LF
N30  G02  X=R1  Y=-R1  J=-R1  LF
N35  G01  Y=-R2  LF
N40  G03  X=R1  Y=-R1  I=R1  LF
N45  M17  LF
```

END OF SECTION

13 Subroutines

13.1 Application

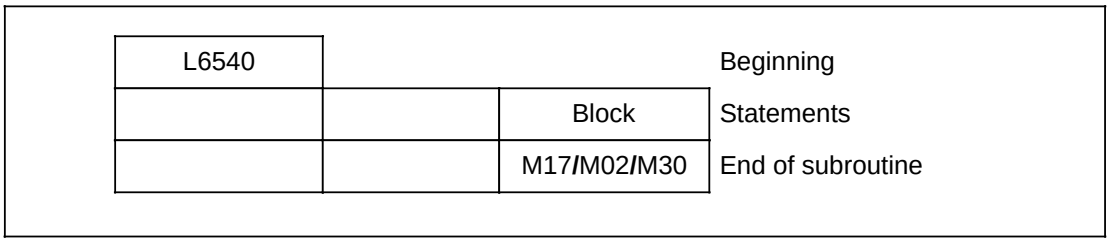
If the same machining operation must be performed repeatedly when a workpiece is machined, it can be entered as a subroutine and called as often as desired in the part program or by means of manual data input. Subroutine numbers L1 to L9999 are available for programming. It is not possible to call subroutine L0 in a program as it is assigned to a special system function.

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.

13.2 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 a subroutine number which can be up to four digits long (see program key).

The end of the subroutine is used to return to the part program and is defined by the end character M17 or M02/M30.

M17 or M02/M30 is written in the last block of the subroutine. It is permissible to write other functions (except address L) in this block, i.e. a subroutine consists of at least two blocks.

13.3 Subroutine call (L... / G80...G89)

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). The number of passes P can be between 1 and 9999.

Example:

```
N20  L123  P1
```

L123	Subroutine number	(1...9999)
P1	Number of passes	(1...9999)

The following should be noted during programming:

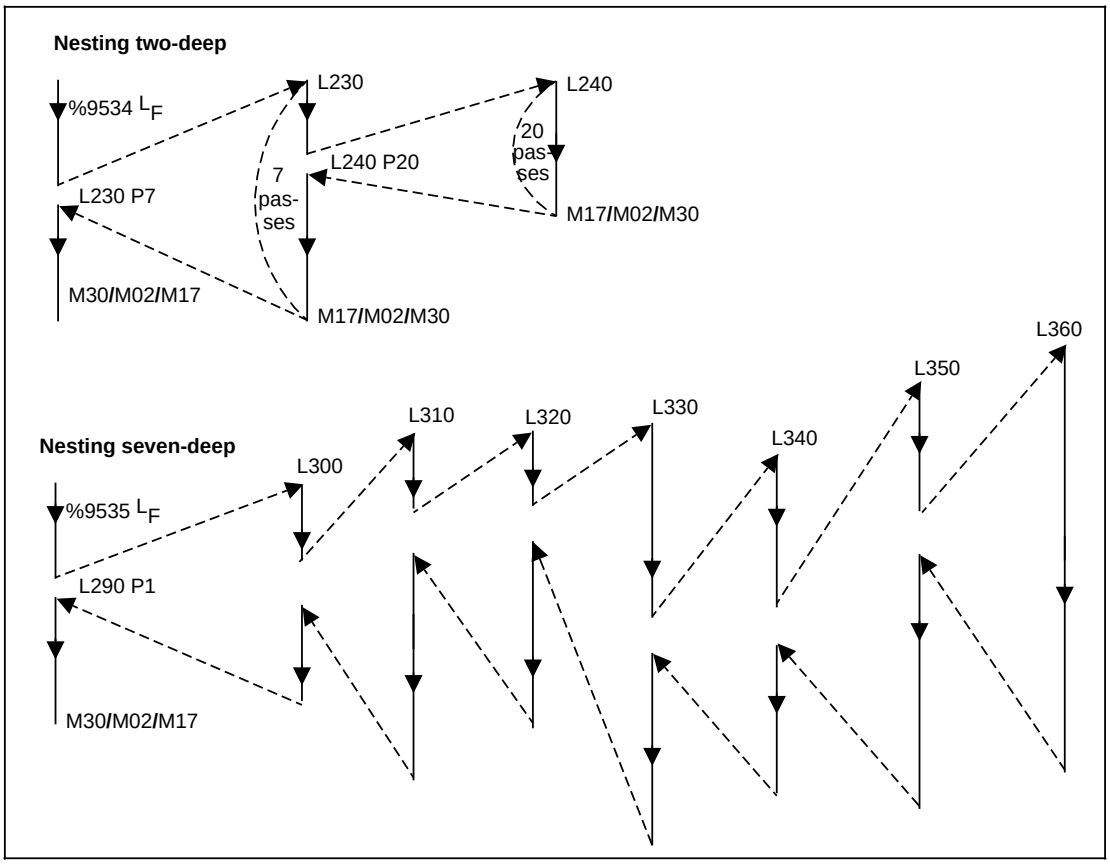
- The subroutine can be terminated with M17 or M02**/**M30.
- The subroutine call must not be written in a block together with M02, M30 or M17.
- If the subroutine is called while the tool radius compensation function is selected, the section on special cases for TRC (“Blocks without path addresses”) 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.
- G81 to G89 cause the modal call of L81 to L89. This call is effective until G80 (initial setting) becomes active.

  G81 to G89 and L81 to L91 in one block triggers alarm 3006 ”Wrong block structure”.

13.4 Subroutine nesting

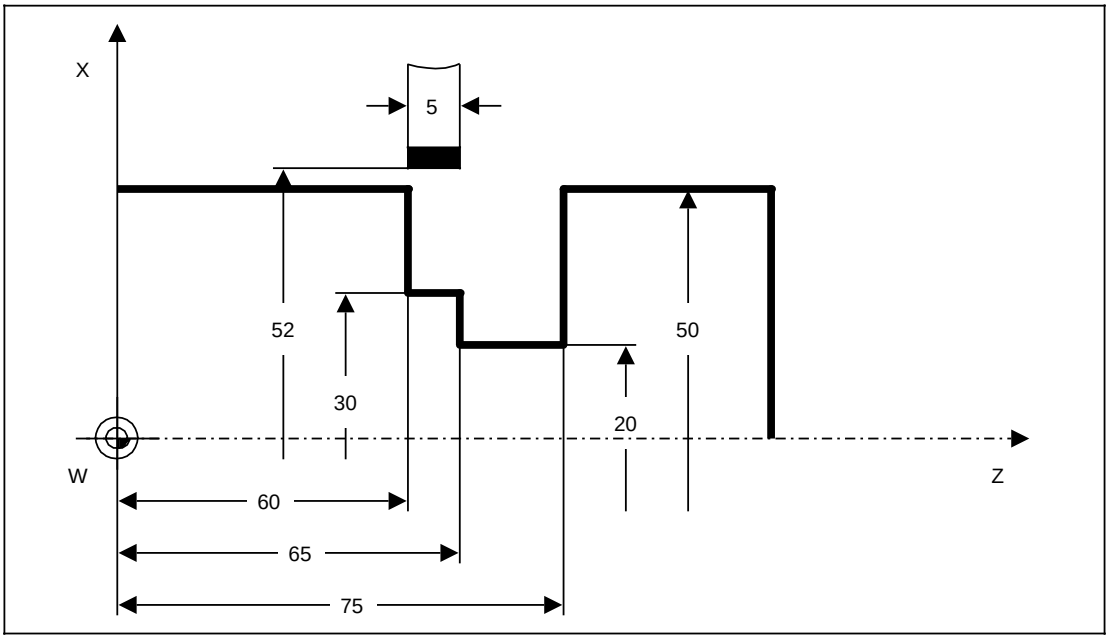
Subroutines can be called not only from a part program, but also from other subroutines. This process is referred to as subroutine nesting.

Nesting of the subroutine to a depth if **seven** is possible.
With SW 1 and 2 nesting up to depth of three is possible.



Subroutine call and subroutine nesting, two-deep and seven-deep

Example:



Program execution

Subroutine nesting

```
%4011
N05  G90  G94  F..  S..  D..  T..  M..  LF
N10  G00  X52  Z60  LF
N15  L230  P1 LF                          (Subroutine call No. 230)
.
.
N50  M30

L230  LF
N05  G91  G01  X-11  LF
N10  G09  X11  LF
N15  L240  P2  LF                         (Subroutine call No. 240)
N20  M17  LF

L240  LF
N05  G91  G00  Z5  LF
N10  G01  G09  X-16  LF
N15  G00  X16  LF
N20  M17  LF
```

Note:

If a subroutine nesting is re-edited or modified in the NC stop state or with M00, the RESET key must be pressed before program execution is resumed.

END OF SECTION

14 Programming of Cycles

14.1 General information

Cycles etc. can be created manually (without computer) using the @ code. Difficult problems are solved with the WS 800 A configuring workstation in the CL800 programming language. The following machining cycles are available on the hard disk as permanently stored subroutines:

SINUMERIK 840C turning cycles:

- L93 Grooving cycle
- L94 Undercut cycle
- L95 Stock removal cycle with relief cutting elements
- L96 Stock removal cycle without relief cutting elements
- L97 Thread cutting cycle
- L98 Deep-hole drilling cycle
- L99 Chaining of threads (4-point thread cutting cycle)

SINUMERIK 840C milling cycles:

- Drilling cycles L81 to L89
- L900 Drilling patterns
- L901 Milling pattern ”GROOVE”
- L902 Milling pattern ”ELONGATED HOLE”
- L903 Milling rectangular pocket
- L904 Milling pattern ”CIRCULAR GROOVE”
- L905 Drilling pattern ”SINGLE HOLE”
- L906 Drilling pattern ”ROW OF HOLES”
- L930 Milling circular pocket

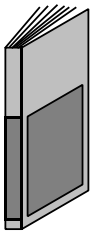
If G functions G81 to G89 are active, cycles L81 to L89 are called even in an empty block, calculation block or common block.

Note:

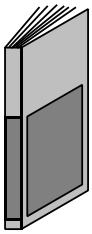
If G functions G81 to G89 are active, cycles L81 to L89 are called even in an empty block, calculation block or common block.

Various measuring cycles are additionally available as an option for the SINUMERIK 840C. In addition to the possibility of checking the dimensional accuracy of the workpiece, measuring with a CNC machine offers further applications:

- Detection of tool breakage
- Measuring of the tool geometry
- Determination of workpiece clumping tolerances
- Compensation for factors influencing machining etc.



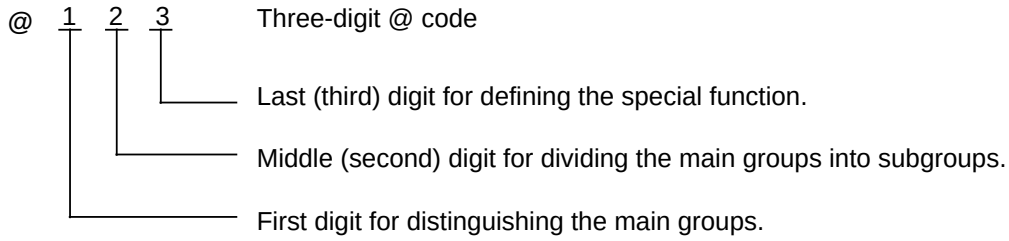
The machining cycles are described in the ”Programming Guide, SINUMERIK System 800, Cycles”.
The measuring cycles are described in the ”User's Guide SINUMERIK 840/850/880, Measuring Cycles”.



The following sections are intended to introduce the user to programming with the @ code.

14.2 Target code

The three-digit @ code is a target code with the following structure:



14.2.1 Main groups

The following functions are assigned to the eight groups of the @ code:

@0 . . General instructions for program structure
@1 . . Program branches
@2 . . Data transfer, general
@3 . . Data transfer, system memory in R parameters
@4 . . Data transfer, R parameters in system memory
@5 . . File handling, general (available soon)
@6 . . Mathematical and logic functions
@7 . . NC-specific functions

14.2.2 Operands behind the target code

The @ code requires additional information (operands) for its function.
These operands are defined by the following letters:

K . . . Constant
R . . . R parameter (register)
P . . . Pointer

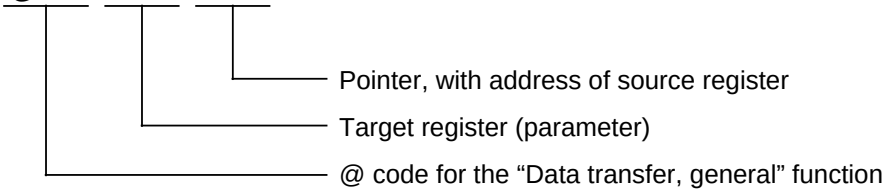
The value defined by the constant K is fixed within the program and cannot be modified (direct specification of value).

The value contained in an R parameter can be modified by the program (indirect specification of value).

The pointer refers to a parameter containing the address of the parameter to whose contents the function is to be applied (indirect specification of value).

Examples for @ code with operands:

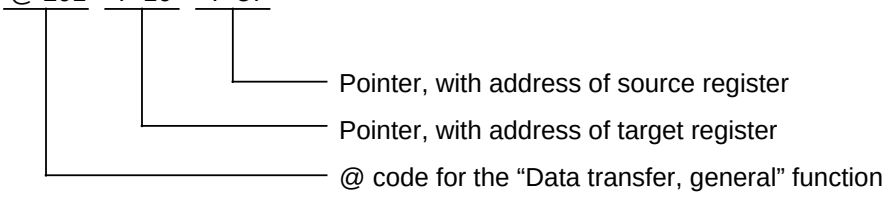
a) @ 201 R 13 P 37



Explanation of function given in example a):

Load the contents of source register, whose address is contained in register R37, into target register R 13.

b) @ 201 P 16 P 37



Explanation of function given in example b):

Load the contents of the source register, whose address is in register R37, into the target register whose address can be found in register R16.

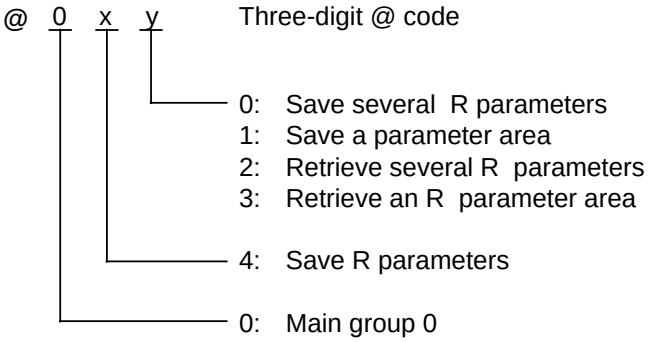
14.2.3 Notation

The @ code calls for strict notation. In the listing of the various commands on the following pages, a series of notations, each in pointed brackets, follows the three-digit @ code. The meanings of the notations are:

<Const>	Direct specification of value (constant K)
<R-Par>	Indirect specification of value (R parameter)
<Var>	Indirect specification of value (R parameter or pointer)
<Value>	Mixed specification of value (constant, R parameter or pointer)

14.3 General instructions for program structure

The main group 0 is organized as follows:



Main group 0 **/** subgroup 4: **Save R parameters**

@040 <Const> <R-Par 1> . . . <R-Par n>

With <Const> , the number of subsequent R parameters is specified belonging to this function. The contents of the R parameters are saved by transferring into a stack register starting at R300. This function can be active for several blocks.

@041 <R-Par 1> <R-Par 2>
The contents of the R parameters in the area from <R-Par 1> to <R-Par 2> are saved by transferring into a stack register starting at R300.

@042 <Const> <R-Par n> . . . <R-Par 1>
The contents of the R parameters in the area from <R-Par 1> to <R-Par 2> are saved by transferring into a stack register starting at R 300. This function can also be active for several blocks.

@043 <R-Par 1> <R-Par 2>
The values saved with @ 041 are loaded back into the R parameters.

These commands in main group 0 **/** subgroup 4 are used when working in a subroutine with R parameters that have possibly been applied in a higher level.

A push command (@ 040 or @ 041) must be written at the beginning of a subroutine in order to save the values and assign value 0 to the R parameters specified.

The original condition is re-established with a pop command (@ 042 or @ 043) at the end of the subroutine.

Example for instruction form in the program:

L100				**Call subroutine**
@041	**R61**	**R69**	L_F	The contents of the R parameters from R61 to R69 are transferred into the stack register and allocated the value “0”.
⋮				
@043	**R61**	**R69**	L_F	The saved values are loaded back into the parameters R61 to R69.
⋮				
M17	L_F			Subroutine end

14.4 Program branches

The main group 1 is organized as follows:

@ 1 x y Three-digit @ code

y:
0: No compare operator
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

x:
0: Absolute jump
1: CASE branch (compare operator = 1)
2: IF-THEN-ELSE instruction
3: WHILE instruction
4: REPEAT loop
5: FOR-TO loop
6: FOR DOWNTO loop

1:
1: Main group 1

With blocks in which jumps are programmed (@ 100 and following) the following points should be observed:

- Functions in the same block as the branching conditions (e.g. auxiliary functions, movements) are only executed if the branching condition is **not** fulfilled.
- Calculation with and allocations to R parameters preceding the jump command are executed, those following the jump command are **not** executed.

With the ”EXECUTION FROM EXTERNAL SOURCE” function only forward jumps are possible in the part program.

Main group 1 **/** 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 R parameter (pointers not allowed). 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.

If the sign points in the wrong direction, the control will not find the block even if it is contained in the program (alarm 3012: “Block does not exist”).

Examples:

@100 K375 Unconditional jump to block N 375 in the direction of end of program
@100 K-150 Unconditional jump to block N 150 in the direction of beginning of program

Jumping to the same block can either be defined with the signs +**/**– or without a sign.

Examples:

N010 R1=R1+1 @100 K–10

or

N010 R1=R1+1 @100 K+10

or

N010 R1=R1+1 @100 K10

Main group 1 **/** subgroup 1: **CASE branch**

@111 <Var> <Value 1> <Const 1> <Value 2> <Const 2>...<Value n> <Const n>
@100 <Const x>

The notation <Var> is compared successively with all notations <Value...>. If they are equal, the program branches to the block number programmed under notation <Const...>. If they are not equal, the CASE branch is ended with a jump to block number <Const x>.

The CASE branch must always consist of the initial branch @111 <Var> <Value 1> <Const 1> and the end jump @111 <Const x>.

The end jump to <Const x> is also necessary for reasons of syntax when the program continues with the next block.

Any amount of additional <Value n> <Const n> comparisons can also be included. The CASE branch can also be distributed across several program blocks. However, the initial branch, all following branches and the end jump must not be separated by block limits.

Correct distribution across several blocks:

@111 <Var> <Value 1> <Const 1> <Value 2> <Const 2> ... <Value n>
@100 <Const x>

or

@111 <Var> <Value 1> <Const 1>
<Value 2> <Const 2>
. . .
<Wert n> <Const n>
@100 <Const x>

or

@111 <Var> <Value 1> <Const 1> <Value 2> <Const 2>
. . . <Value n> <Const n> @100 <Const x>

Incorrect distribution across several blocks:

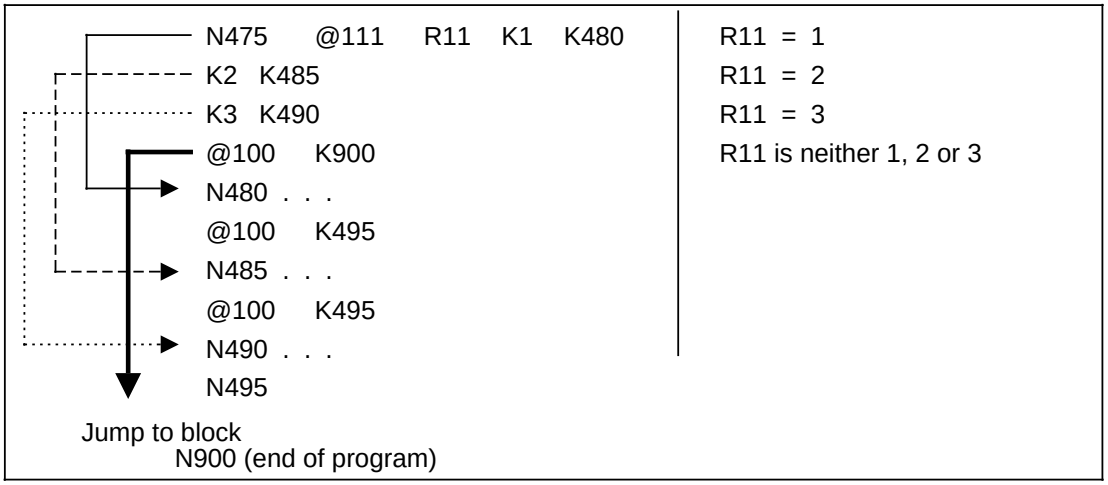
@111 <Var>
<Value 1> <Const 1> <Value 2> <Const 2> . . . <Value n> <Const n> @100 <Const x>

The blocks across which they are distributed must not contain a block number (exception, 1st block with @111).

Example for CASE branch:
The figure below shows the application of a CASE branch in a cycle with switchable axes.

Depending on whether the programmer has defined the drilling axis or the machining plane by setting the parameter R 11 = 1, 2, or 3, the program branches into the blocks N480 (R11 = 1), N485 (R11 = 2) or N490 (R11 = 3).

If none of the three values is contained in R11, then clearly there is a parameterizing error. The program branches to block N900 (= end of program).



Application of a CASE branch in a cycle with switchable axes

Main group 1 **/** subgroup 2: **IF-THEN-ELSE branch**

An IF-THEN-ELSE branch states that:

IF the condition (third digit in the @ code) has been satisfied, THEN execute the instructions contained in the next blocks, otherwise (ELSE) branch to the block whose number is specified by the last constants encountered.

Here too, the sign of the block number acts as search direction.

@121 <Var> <Value> <Const>

IF the numerical value defined with the notation <Var> is equal to that defined with <Var> THEN the program is resumed with the next block. Otherwise (ELSE) a jump is made to the block specified by the constant.

Example:

@121 R13 R27 K375 Continuation of the program if R13 = R27, otherwise conditional jump to block N375 in the direction of end of program.

@122 <Var> <Value> <Const>

IF the numerical value defined with the notation <Var> is not equal to that defined with <Value>, THEN the program is resumed with the next block. Otherwise (ELSE) a jump is made to the block specified by the constant.

@123 <Var> <Value> <Const>

IF the numerical value defined with the notation <Var> is greater than that defined with <Value>, THEN the program is resumed with the next block. Otherwise (ELSE) a jump is made to the block defined by the constant.

Example:

@123 R13 R27 K-150 Continuation of the program if R13 > R27, otherwise conditional jump to block N150 in the direction of beginning of program

@124 <Var> <Value> <Const>

IF the numerical value defined with the notation <Var> is greater than or equal to that defined with <Value> , THEN the program is resumed with the next block. Otherwise (ELSE) a jump is made to the block defined by the constant.

@125 <Var> <Value> <Const>

IF the numerical value defined with the notation <Var> is less than that defined with <Value>, THEN the program is resumed with the next block. Otherwise (ELSE) a jump is made to the block specified by the constant.

@126 <Var> <Value> <Const>

IF the numerical value defined with the notation <Var> is less than or equal to that defined with <Value>, THEN the program is resumed with the next block. Otherwise (ELSE) a jump is made to the block specified by the constant.

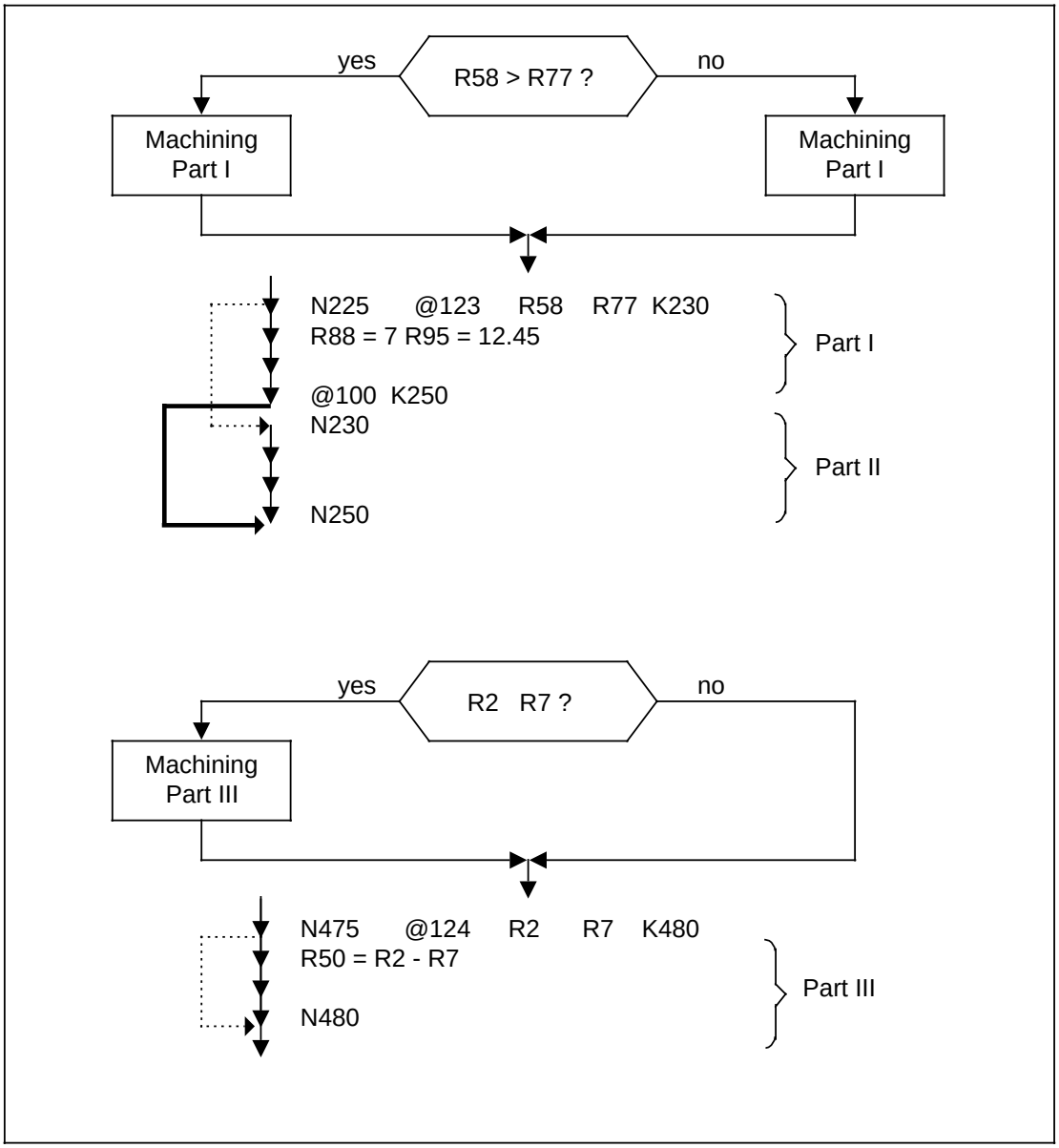
@127 <Var> <Value> <Const>

Value=0 or 1 (no other value is permissible)
Var=R par. "Bit pattern otherwise CL800 alarm"
If the bit pattern in"Var" is greater than the bit defined with "Value", the program is continued with the next block. Otherwise (ELSE) a jump is made to the block specified by the constant.
Only Bit 0 is checked.

@128 <Var> <Value> <Const>

Value=0 or 1 (no other value is permissible)
Var=R par. "Bit pattern otherwise CL800 alarm"
If the bit pattern in"Var" is not greater than the bit defined with "Value", the program is continued with the next block.

Example for program execution with IF-THEN-ELSE branches:



Program execution with IF-THEN-ELSE branch

Explanations for the example:

The program section from a cycle described in the above example shows how program branches can be made up using IF-THEN-ELSE branches. If the contents of register R58 in block N225 are less than or equal to the contents of register R77, then the instructions given in the next line are executed.

Register R88 is loaded with 7 and register R95 with 12.45. However, if R58 is greater than R77, then the program branches to block N230. In the line before block N230, there is, however, an unconditional jump to block N250.

With the IF-THEN-ELSE branch in block N225, this means that either block N225 is executed or the program section from block N230 to block N250.

In the lower example, the unconditional jump is missing so that either the instructions in block N475 are followed or not. This program part can therefore be “skipped” with the IF-THEN-ELSE branch @124.

Main group 1 **/** subgroup 3: **WHILE loop**

@13x <Var> <Value> <Const>

DThe WHILE loop is a repetition instruction with scanning of the repetition conditions at the beginning of the loop. The comparison operators correspond to those in the IF-THEN-ELSE branch. As long as the comparison is satisfied, the next block is processed. An absolute jump must be programmed at the end of the block with @100 < Const > to return to scanning.

If the comparison has not been satisfied, a jump is made to the block defined in < Const > ; this block is usually located behind the block with the absolute jump.

Examples:

```
N300  @131  R13  R27  K375
            .
@100  K-300
N375 . . .
```

Continuation of the loop provided loop condition R13 = R27 has been satisfied.

```
N300  @133  R13  R27  K375

@100  K–300
N375 . . .
```

Continuation of the loop provided loop condition R13 > R27 has been satisfied.

Main group 1 **/** subgroup 4: **REPEAT loop**

@14x <Var> <Value> <Const>

The REPEAT loop is a repetition instruction with scanning of the repetition conditions at the end of the loop. The compare operators correspond to those in the IF-THEN-ELSE branch. As long as the comparison is not satisfied, a return jump is made to the block defined in < Const >. If the condition is satisfied, the loop is exited and the program is resumed.

Example:

N400	Repeat the following instructions until condition R13 = R27 is satisfied.
.	
@141 R13 R27 K-400	
N400	Repeat the following instructions until condition R13 > R27 is satisfied.
@143 R13 R27 K-400	

Main group 1 **/** 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 in <Var> are incremented with each pass. The scanning for “equal to” is made at the beginning of the loop. As long as there is no equality, the loop is processed, otherwise a jump is made to the block defined in < Const >. At the end of the loop the variable <Var> must be incremented (@620) and an absolute jump must be made to the beginning of the loop.D

Example:

R5 = 1 R51 = 5 R52 = 10	Value assignments for R5, R51, R52
@201 R50 P51	Data transfer from R51 to R52
N500 @151 R50 R52 K505	Beginning of FOR TO loop
@620 R50	
@100 K-500	
N505	

Main group 1 **/** 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 in <Var> are decremented with each pass. The scanning for ”equal to” is made at the beginning of the loop. As long as there is inequality, the loop is processed, otherwise a jump is made to the block defined in < Const >.

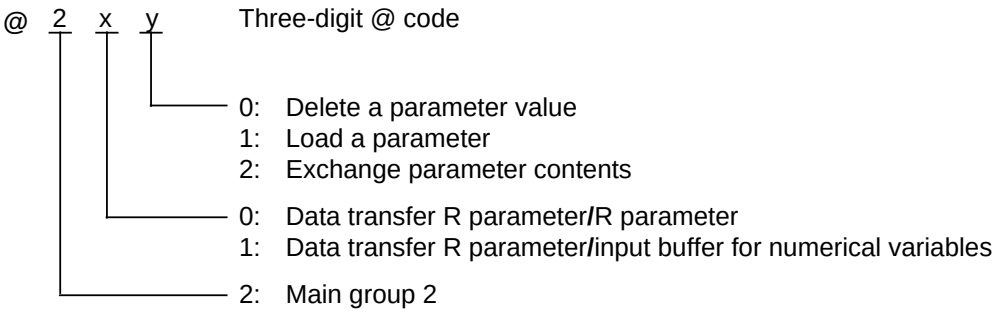
At the end of the loop, the variable <Var> must be decremented (@621) and an absolute jump made to the beginning of the loop.

Example:

R5 = 10 R51 = 5 R52 = 1	Value assignments for parameters R5, R51, R52
@201 R50 P51	Data transfer from R51 to R52
N600 @161 R50 R52 K605	Beginning of FOR DOWN TO loop
@621 R50	
@100 K–600	
N605	

14.5 Data transfer, general

The main group 2 is organized as follows:



Main group 2**/**Subgroup 0: **Data transfer R parameter/R parameter**

@200 <Var>

The value of the R parameters with the notation <Var> are deleted.

@201 <Var> <Value>

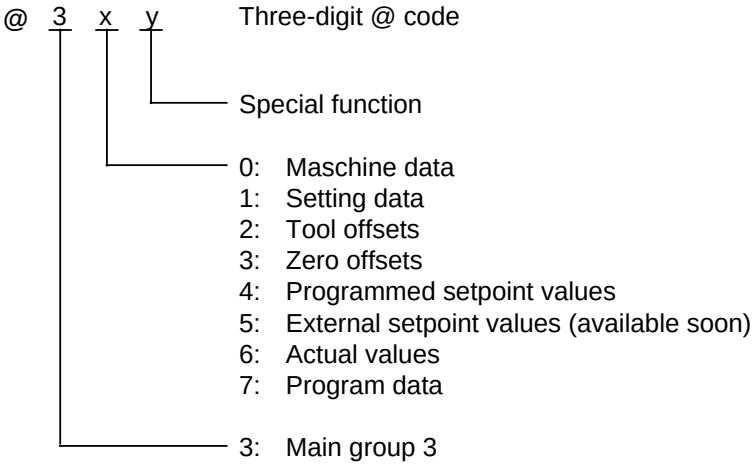
The numerical value defined in <Value> is loaded in the R parameter defined in <Var>.

@202 <Var 1> <Var 2>

The contents of both the R parameters defined in <Var1> and <Var2> are exchanged.

14.6 Data transfer: System memory into R parameter

The main group 3 is organized as follows:



All @ commands in main group 3 have <Var> as first notation. This defines direct or via a pointer an R parameter in which the contents of the addressed system cell are to be loaded.

Main group 3 **/** subgroup 0: **Transfer machine data in R parameter**

@300 <Var> <Value 1>

The address of an NC machine data is defined in <Value 1>.

Example:

@300 R50 K2240 The value of the first software limit switch is given in parameter R50, in the plus direction for the first axis.

@301 <Var> <Value 1>

The byte address of an NC machine data is defined in <Value 1>.

@302 <Var> <Value 1> <Value 2>

The byte address of an NC machine data bit is defined in <Value 1>.
The bit address (0 to 7) is given in <Value 2>.

@303 <Var> <Value 1> <Value 2>

The cycle machine data values are read into parameter<Var>.

Where:

<Value 1> Channel No.
0: own channel
<Value 2> Word address

@304 <Var> <Value 1> <Value 2>

The cycle machine data bytes are read into parameter <Var>.
Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Byte address

@305 <Var> <Value 1> <Value 2> <Value 3>

The cycle machine data bits are read into parameter <Var>.
Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Byte address
<Value 3>	Bit address

@306 <Var> <Value 1>

The address of a PLC machine data is defined in <Value 1>.

@307 <Var> <Value 1>

The byte address of a PLC machine data is defined in <Value 1>.

@308 <Var> <Value 1> <Value 2>

The byte address of a PLC machine data bit is defined in <Value 1>.
The bit address (0 to 7) is in <Value 2>.

@30c <Var> <Value 1> <Value 2> 1)

With this command, IKA data can be read in by the part program. They are addressed via a data type (0...16) and a data number.

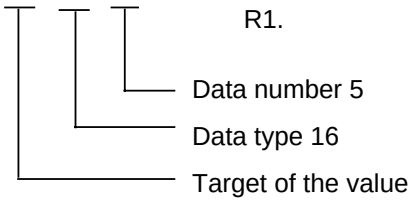
where:

<Var>	R parameter with the read-in value
<Value 1>	Constant with data type
<Value 2>	Constant with data number

Example:

@30c R10 K2 K11 The base axis number (data type 2) of the 11th IKA configuration is stored in R10.

@30c R1 K16 K5 The module value of IKA5 is read, the result is stored in R1.



Identifiers for the data formats can be found under the description of @40c in this Section.

1) SW 3 and higher

Main group 3 **/** subgroup 1: **Transfer setting data into R parameter**

@310 <Var> <Value 1>

The address of a setting data is defined in <Value 1>.

@311 <Var> <Value 1>

The byte address of a setting data is defined in <Value 1>.

@312 <Var> <Value 1> <Value 2>

The byte address of a setting data bit is defined in <Value 1>.
The bit address (0 to 7) is in <Value 2>.

@313 <Var> <Value 1> <Value 2>

The cycle setting data are read into parameter <Var>.

Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Word address

@314 <Var> <Value 1> <Value 2>

The cycle setting data bytes are read into parameter <Var>.

Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Byte address

@315 <Var> <Value 1> <Value 2> <Value 3>

The cycle setting data bits are read into parameter <Var>.

Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Byte address
<Value 3>	Bit address

Main group 3 **/** subgroup 2: **Transfer tool offsets into R parameter**

@320 <Var> <Value 1> <Value 2> <Value 3>

MWith this command, the various offset values can be read from the tool offset memory into the parameter under the notation <Var>.

The notations <Value 1> to <Value 3>must be allocated as follows :

<Value 1>	TO area Area: 1 to 4
<Value 2>	Tool offset No. (D number) Area: 1 to 204
<Value 3>	Number of tool offset memory (P number). Area: 0 to 9/15

Example:

@320	**R67**	**K1**	**K14**	**K2**	Offset value P2 (geometry, length 1) of tool offset number D14 for TO area 1 is read into parameter R67.

Main group 3 **/** subgroup 3: **Transfer zero offsets into R parameter**

@330 <Var> <Value 1> <Value 2> <Value 3>

The notations <Value 1> to <Value 3> must be allocated as follows:

<Value 1>	Group of settable zero offsets (G54 = 1 to G57 = 4)
<Value 2>	Number of axes
<Value 3>	Coarse and fine value (0 or 1)

Example:

@330	**R81**	**K1**	**K2**	**K0**	The coarse value of the first settable zero offset (G54) of the second axis is read into parameter R81.

@331 <Var> <Value 1> <Value 2>

Where:

<Value 1>	Group of programmable additive zero offsets (G58 = 1 and G59 = 2)
<Value 2>	Number of axis

@332 <Var> <Value 2>

The number of the axis of the external zero offset input from the PLC is defined in <Value 2>.

@333 <Var> <Value 2>

The number of the axis of the DRF offset is defined in <Value 2>.

@334 <Var> <Value 2>

The number of the axis of the PRESET offset is defined in <Value 2>.

@336 <Var> <Value 2>

The number of the axis of the cumulative offset is defined in <Value 2>. The cumulative offset contains:

- the selected settable zero offsets
- the programmable additive zero offset
- the external zero offset
- the selected tool offset.

PRESET and DRF offsets are not considered.

@337 <Var> <Value 1> <Value 2> <Value 3>

The rotation angle of the settable coordinate rotations can be read into parameter <Var>.

The notations <Value 1> to <Value 3> must be allocated as follows:

<Value 1>	Number of channel (0 = own channel)
<Value 2>	Group of settable coordinate rotations (G54 = 1 to G57 = 4)
<Value 3>	Number of angle (at present = 1)

@338 <Var> <Value 1> <Value 2> <Value 3>

The rotation angle of the programmable coordinate rotations can be read into parameter <Var>.

Where:

<Value 1>	Number of channel (0 = own channel)
<Value 2>	Group of programmable additive coordinate rotations (G58 = 1 to G59 = 2)
<Value 3>	Number of angle (at present = 1)

@339 <Var> <Value 2> <Value 3>

Reading of the cumulative offset of an axis from a channel to be specified. The offset is updated in every block if MD 5153.7 is set.

<Var>	Destination R parameter
<Wert 2>	Axis number
<Wert 3>	Channel number

Main group 3 **/** subgroup 4: **Read programmed setpoint value into R parameter**

@345 <Var> <Value 1> <Value 2>

With this command, the cutting speed programmed in G96 can be transferred into a R parameter. The channel number is entered in <Value 1> (own channel =”0”).

<Value 2> is assigned ”0”.

Main group 3 **/** subgroup 6: **Read actual values into R parameter**

@360 <Var> <Value 2>
The axis is for which the **workpiece-referred actual value** is to be transferred into the R parameter is defined in <Value 2>. The actual value is always read as the radius value.

Example:

@360	**R54**	**K2**	Read the positional actual value of axis Y (= 2), referred to the workpiece zero, and enter into register R54.

Note:

An overlaid IKA value is not been taken into consideration here. This must be calculated (using @30c Typ 35) by the user himself.

@361 <Var> <Value 2>

The axis number for which the **machine-referred actual value** is to be transferred into the R parameter <Var> is defined in <Value 2>. The actual value is always read as the radius value.

In the case of linear axes, the machine-referred actual value is stored in R parameter <Var> (R_n).

In the case of rotary axes, the machine-referred actual value is stored in R_n as for linear axes if the MD bit "Actual value display modulo 360 degrees" has **not** been set.

If the above MD bit has been set for the rotary axis, two R parameters are loaded as follows:

R_n = Position within one revolution
$R_n + 1$ = Number of revolutions

@362 <Var> <Value 2>

The axis number for which the **machine-referred actual value with incorporated following error** is to be transferred into the R parameter <Var> is defined in <Value 2>. The actual value is always read as the radius value.

In the case of linear axes, the machine-referred actual value incorporating the following error is stored in R parameter <Var> (R_n).

In the case of rotary axes, the machine-referred actual value incorporating the following error is stored in R_n as for linear axes if the MD bit "Actual value display modulo 360 degrees" has **not** been set.

If the above MD bit has been set for the rotary axis, two R parameters are loaded as follows:

R_n = Position within one revolution
$R_n + 1$ = Number of revolutions

Note:

The value is read in input resolution and stored in position control resolution. Therefore, an error may occur in case of different input resolution and position control resolution.

@363 <Var> <Value 2>

The spindle for which the actual position is to be transferred into the R parameter is defined in <Value 2>.

@364 <Var> <Value 2>

The spindle for which the actual speed is to be transferred into the R parameter is defined in <Value 2>.

@367 <Var> <Value 1>

The R parameter is to be defined in <Var> from which the axis numbers of the plane selected with G16 (4 values) are entered. The number of the leadscrew is stored in the 5th R parameter. The channel number must be defined in <Value 1> (0 = own channel).
If a tool length compensation is programmed with G16 XYZ in negative direction, the value 128 is added to the Z axis with @367, in order to recognize the negative sign.

@36a <Var> <Value 1>

With this command, the number of the selected tool offset (D number) is transferred into the R parameter. The channel must be specified in <Value 1> (own channel = 0).
Note: ”a” in the @ code represents hexadecimal ”A” (=10).

@36b <Var> <Value 1> <Value 3>

The G function of the part program block which is being processed is read from the main memory into parameter <Var>. The channel number is defined in <Value 1>. If it is defined as 0 the same channel is read. <Value 3> defines the **internal** G group to which the current G function belongs. See Section 13 for a table with the **internal** G group division (especially for @36b only).

Example:

@36b R50 K0 K0 The current G function of the first internal G group (G0) is read into the parameter R50 in the same channel.

A separate block with @714 must be programmed before @360 to @36b. To be able to read values from another channel NC Start must be issued in that one as well.

@36e <Var> <Value 2>

<Value 2> defines the spindle whose spindle setpoint is read in without the sign.

Main group 3 **/** subgroup 7: **Read program data into R parameter**

@371 <Var> <Value 1> <Value 3>

With this command, special bits are read out for the acquisition of various active signals. In the program the states of only those channels can be scanned which are processed by the same NC-CPU as the running program.

Note: @371 for NCK program Bit 2=0; @371 for simulation program Bit 2=1;

Channel-dependent bits:
Bit 0 = Block search active
Bit 1 = Dry run feedrate active
Bit 2 = Simulation active

The channel number must be entered in <Value 1> (own channel = 0). <Value 3> contains the bit number.

Channel-independent bits:
Bit 0 = Measurement input 1 active
Bit 1 = Measurement input 2 active

99 must be entered in <Value 1>. <Value 3> contains the bit number.

Example:

@371 R81 K0 K1 The status of the special bit for ”Dry run feedrate” in the own channel is read in parameter R81.

@372 <Var>

The channel no. of the current NC channel is read in parameter <VAR>.

Main group 3**/**subgroup 8: **Read PLC signal bits in R parameters**

@380 <Var> <Value 1> <Value 2> <Value 3>

The state of an input bit in the PLC is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the byte address by <Value 2> and the bit address by <Value 3>.

Example:

@380 R50 K1 K2 K0 The state of the defined input bit is read in R50.

@381 <Var> <Value 1> <Value 2> <Value 3>

The state of an output bit in the PLC is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the byte address by <Value 2> and the bit address by <Value 3>.

@382 <Var> <Value 1> <Value 2> <Value 3>

The state of a flag bit in the PLC is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the byte address by <Value 2> and the bit address by <Value 3>.

@383 <Var> <Value 1> <Value 2> <Value 3><Value 4>

The state of a PLC data word bit is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2>, the data word number by <Value 3> and the bit address by <Value 4>.

Example:

@383 R51 K1 K2 K4 K2 The state of the defined PLC data word bit is read in R51.

Main group 3**/**subgroup 9: **Read PLC signal bytes in R parameters**

@390 <Var> Value 1> <Value 2>

The state of an input byte in the PLC is read in the parameter defined with <Var>. The PLC number is defined by <Value 1> and the byte address by <Value 2>.

Example:

@390 R52 K1 K1 The state of the defined PLC input byte is read in R52.

@391 <Var> <Value 1> <Value 2>

The state of an output byte in the PLC is read in the parameter defined with <Var>. The PLC number is defined by <Value 1> and the byte address by <Value 2>.

@392 <Var> <Value 1> <Value 2>

The state of a peripheral byte in the PLC is read in the parameter defined by <Var>. The PLC number is defined by <Value 1> and the byte address by <Value 2>.

@393 <Var> <Value 1> <Value 2>

The state of a flag byte in the PLC is read in the parameter defined by <Var>. The PLC number is defined by <Value 1> and the byte address by <Value 2>.

@394 <Var> <Value 1> <Value 2> <Value 3>

The state of a PLC data word left is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2> and the data word number by <Value 3>.

Example:

@394 R53 K1 K2 K4 The state of the defined PLC input byte is read in R53.

@395 <Var> <Value 1> <Value 2> <Value 3>

The state of a PLC data word right is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2> and the data word number by <Value 3>.

Main group 3/subgroup a: **Read PLC signal words in R parameters**

@3a0 <Var> <Value 1> <Value 2> <Value 3>

The state of an input word in the PLC is read in the parameter defined by <Var>. The PLC number is defined by <Value 1> and the word address by <Value 2>. The dimension identifier <Value 3> specifies how the state of the PLC signal word is to be read.

Definition of dimension identifier <Value 3>:

<Val. 3>	Fixed point:
0	Value without decimal point
1	Value with decimal point
2	1 place after the decimal point
3	2 places after the decimal point
4	3 places after the decimal point
5	4 places after the decimal point
6	5 places after the decimal point
7	6 places after the decimal point
8	7 places after the decimal point
9	8 places after the decimal point

<Val. 3>	BCD
100	Value without decimal point
101	Value with decimal point
102	1 place after the decimal point
103	2 places after the decimal point
104	3 places after the decimal point
105	4 places after the decimal point
106	5 places after the decimal point
107	6 places after the decimal point
108	7 places after the decimal point
109	8 places after the decimal point

Example: @3a0 R50 K1 K3 K100 The state of the defined PLC input word is read in R50 in BCD format.

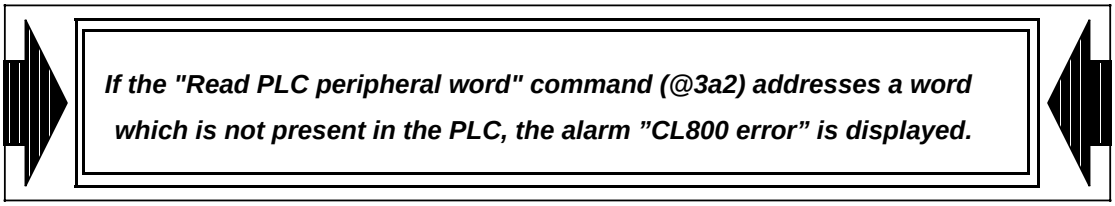
@3a1 <Var> <Value 1> <Value 2> <Value 3>

The state of an output word in the PLC is read in the parameter defined by <Var>. The PLC number is defined by <Value 1> and the word address by <Value 2>. The dimension identifier <Value 3> specifies how the state of the PLC signal word is to be read.

@3a2 <Var> <Value 1> <Value 2> <Value 3>

The state of a peripheral word in the PLC is read in the parameter defined by <Var>. The PLC number is defined by <Value 1> and the word address by <Value 2>. The dimension identifier <Value 3> specifies how the state of the PLC signal word is to be read.

Definition of the dimension identifier <Value 3>: see @3a0



@3a3 <Var> <Value 1> <Value 2> <Value 3>

The state of a flag word in the PLC is read in the parameter defined by <Var>. The PLC number is defined by <Value 1> and the word address by <Value 2>. The dimension identifier <Value 3> specifies how the state of the PLC signal word is to be read.

Definition of the dimension identifier <Value 3>: see @3a

@3a4 <Var> <Value 1> <Value 2>

The state of a timer in the PLC is read in the parameter defined by <Var>. It is read as a value in seconds with two places after the decimal point. The PLC number is defined by <Value 1> and the timer address by <Value 2>.

Example:

@3a4 R80 K1 K2 **T**he state of the defined PLC timer is read in R80.

@3a5 <Var> <Value 1> <Value 2>

The state of a counter in the PLC is read in the parameter defined by <Var>. The PLC number is defined by <Value 1> and the counter address by <Value 2>.

Example:

@3a5 R81 K1 K2 The state of the defined PLC counter is read in R81.

Main group 3/subgroup b: **Read PLC signal data words in R parameters**

@3b0 <Var> <Value 1> <Value 2> <Value 3> <Value 4> <Value 5>

The fixed-point value of a data word or double word (serial or parallel) in the PLC is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2>, the data word number by <Value 3> and the number of data words by <Value 4>. The dimension identifier <Value 5> specifies how the fixed-point value is to be read.

Definition of dimension identifier <Value 5>:

<Val. 5>	Fixed point: Data word or serial double word
0	Value without decimal point
1	Value with decimal point
2	1 place after the decimal point
3	2 places after the decimal point
4	3 places after the decimal point
5	4 places after the decimal point
6	5 places after the decimal point
7	6 places after the decimal point
8	7 places after the decimal point
9	8 places after the decimal point

<Val. 5>	Fixed point: Parallel double word
10	Value without decimal point
11	Value with decimal point
12	1 place after the decimal point
13	2 places after the decimal point
14	3 places after the decimal point
15	4 places after the decimal point
16	5 places after the decimal point
17	6 places after the decimal point
18	7 places after the decimal point
19	8 places after the decimal point

Note:
In the KF format, a data word within the range of values 0 to 65 535 is read.

Example:

@3b0 R60 = (1,3,2,2,10); The fixed-point value of the defined double word is read in R60

@3b1 <Var> <Value 1> <Value 2> <Value 3> <Value 4> <Value 5>

The BCD value of the defined data words in the PLC is read in the parameter defined with <Var>. The PLC number is defined by <Value 1>, the number of the DB or DX by

<Value 2>, the data word number by <Value 3>, the number of data words by <Value 4> and the dimension identifier by <Value 5>.

Depending on the defined number of data words <Value 4>, the read operation is either serial or parallel:

<Val. 4>	BCD
1	One data word is read, dimension identifier <Value 5> of no significance.
2	Two data words are read in parallel. Dimension identifier <Value 5> of no significance.
3	Three data words are read in serial. Dimension identifier <Value 5> specifies how BCD value is to be read.

Definition of dimension identifier <Value 5>:

<Val. 5>	BCD
100	Value without decimal point
101	Value with decimal point
102	1 place after the decimal point
103	2 places after the decimal point
104	3 places after the decimal point
105	4 places after the decimal point
106	5 places after the decimal point
107	6 places after the decimal point
108	7 places after the decimal point
109	8 places after the decimal point

Example:

@3b1 R51 K1 K2 K10 K100 The BCD value of the defined double word is read in R51.

@3b2 <Var <Value 1> <Value 2> <Value 3>

The floating-point value of the defined data words in the PLC is read in the parameter defined with <Var>. Two data words are always read in serial in the PLC. The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2> and the data value number by <Value 3>.

Example:

R71 = 1 R72 = 100 R73 = 1

@3b2 R70 R71 R72 R73 The floating point value of the defined double words is read in R70.

Main group 3/subgroup c: **Read alarms in R parameters**

@3c0 <Var>

With this command, the NC alarms issued in the control are read and the numbers stored in sequence starting with the variable <Var>.

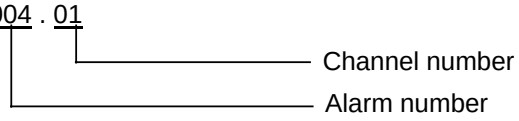
Example:

@3c0 R50 The present NC alarms are loaded from R50 onwards.

The corresponding channel number is added to the alarm number in two places after the decimal point so that both the alarm number and the number of the channel in which it occurred are available in one R parameter.

The contents of the R parameter are interpreted as in the following example:

R50 = 3004 . 01



Main group 3/subgroup d: **Read alarm indications in R parameters**

@3d0 <Var>

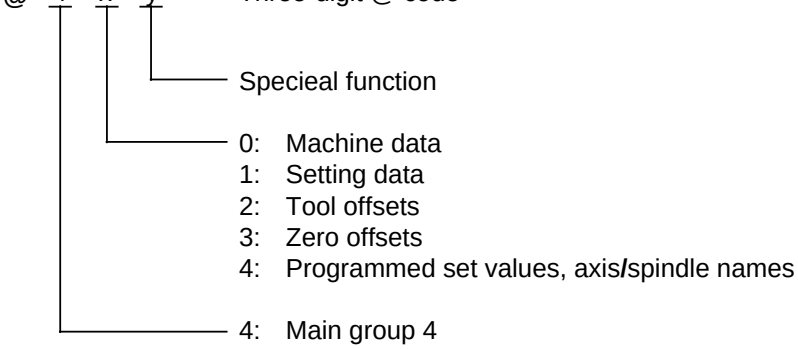
With this command the number of entered NC alarms is read in the variable <Var>.

Example:

@3d0 R51	The number of present NC alarms is read in R51.
@3ff	see main group 2/ Subgroup 0

14.7 Data transfer: R parameter into system memory

The main group 4 is organized as follows:



All @ commands in this main group have <Value> as the last notation. This defines the numerical value to be transferred, direct with a constant or indirect via an R parameter or a poin

Note:

@714 must be programmed prior to using the subgroups 0, 1, 2 and 3.

Main group 4 **/** subgroup 0: **Transfer R parameter in machine data**

Note:

A @714 must be programmed before @400 to @43a.

@400 <Value 1> Value>

The address of an NC machine data is defined in <Value 1> .

Example:

@400	**K2241** **R90**	The machine data of the first software limit switch for the second axis in the plus direction is loaded via parameter R90.

@401 <Value 1> <Value>

The byte address of an NC machine data byte is defined in <Value 1>.

@402 <Value 1> <Value 2> <Value>

The byte address of an NC machine data is defined in <Value 1>.
The bit address (0 to 7) is given in <Value 2>.

@403 <Value 1> <Value 2> <Value>

The address of a cycle machine data is defined in <Value 2>.

Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Word address

@404 <Value 1> <Value 2> <Value>

The byte address of a cycle machine data is defined in <Value 2>.

Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Byte address

@405 <Value 1> <Value 2> <Value 3> <Value>

The bit address of a cycle machine data is defined in <Value 3>.

Where:

<Value 1>	Channel No. 0: own channel
<Value 2>	Byte address
<Value 3>	Bit address

@406 <Value 1> <Value>

The address of a PLC machine data is defined in <Value 1>.

@407 <Value 1> <Value>

The byte address of a PLC machine data is defined in <Value 1>.

@408 <Value 1> <Value 2> <Value>

The byte address of a PLC machine data bit is defined in <Value 1>.

The bit address (0 to 7) is given in <Value 2>.

@40c <Value 1> <Value 2> <Value> [1)]

With this command, IKA data can be written in by part program.
Permitted data types, refer to table ”Data format identifiers”.

Data types 0 are stored in binary form in the R parameters.

Where:

<Value 1>	Constant with data type
<Value 2>	Constant with data number
<Value >	Value to be written (R parameter**/**constant)

Examples:

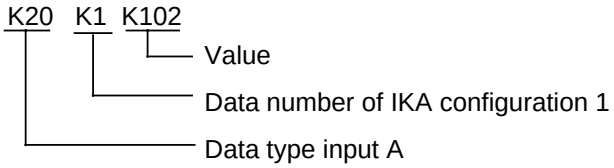
@40c K1 K3 R20 The curve number (data type 1) of the 3rd IKA configuration is transferred from R20 to the table.

Input A of IKA configuration 1 is to be assigned to the actual value of a feed axis.

Group = 1	Feed axis (see type definition table)
Value = 02	Actual value (see type definition table)

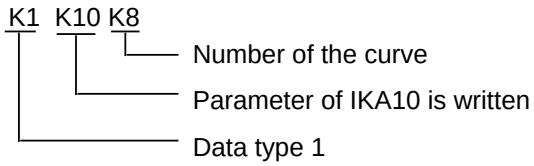
The result is type 102.

Programming: **@40c**



IKA configuration IKA10 is to be applied to compensation curve 8:

@40c



Curve number 3 is to be calculated:

`@40c K55 K3 K-1` Activation always by value -1 (an error is displayed by parameter T55 with content > 0, see @30c).

1) Software version 3 and higher

Data format identifiers:

Data type	Meaning	Data type for @30c/40c or T para.	Data No. within data type	G fct. for IKA	G fct. for CGI	Only valid with IKA stage 2
%IKA3-IKA comp. points	Input variable value	7	1 to 65536	-	-	
	Output variable value	8	1 to 65536	-	-	
%IKA2-IKA curves	Start pointer	5	1 to 32	-	-	
	End pointer	6	1 to32	-	-	
	Activation byte	55	1 to 32	-	-	x
%IKA1 IKA configuration	Control byte 8 bits	0	1 to 32	part. with G410/G412	part. with G400/G402/ G403/PLC	
	Bit0: IKA active	11	1 to 32	part. with G410/G412	part. with G400/G402/ G403/PLC	
	Bit1: IKA direction-dependent	12	1 to32	-	-	
	Bit2: IKA negative direction	13	1 to 32	-	-	
	Bit3: IKAwith comp. values (to SW3)	14	1 to 32	-	-	
	Bit4: Exp. IKA fct.	40	1 to 32	G411	G401	x
	Bit5: Position-rel. swit. direction-dep.	41	1 to 32	G412	G402	x
	Bit6: Position-rel. swit. overtrav. neg.	42	1 to 32	G412	G402	x
	Bit7: Interpolation cubic	43	1 to 32	-	-	x
	Number of the control curve	1	1 to 38	G412	G402	
	Input A (number)	2	1 to 32	G411	G401	
	Input A (type)	20	1 to 32	G411	G401	

Data type	Meaning	Data type for @30c/40c or T para.	Data No. within data type	G fct. for IKA	G fct. for CGI	Only valid with IKA stage 2
%IKA1 IKA configuration	Shift input A	15	1 to 32	-	G403	
	Modulo value of input A	16	1 to 32	-	-	
	Starting position of input A	17	1 to 32	G412	G402	x
	Weighting input A (numerator)	18	1 to 32	-	Funct. cannot be used	x
	Weighting input A (denominator)	19	1 to 32	-	Funct. cannot be used	x
	Input B (number)	25	1 to 32	G411	Funct. cannot be used	x
	Input B (type)	26	1 to 32	G411	Funct. cannot be used	x
	Output (number)	3	1 to 32	G411	G401	
	Output (type)	33	1 to 32	G411	G401	
	Weighting factor output (numerator)	4	1 to 32	G412	G402/G403	
	Weighting factor output (numerator)	30	1 to 32	G412	G402/G403	x
	Max. output value upper limit (not for compensation) 1)	31	1 to 32	-	-	x
	Min. output value lower limit (not for compensation) 1)	32	1 to 32	-	-	x
	Max. change of output value (not for compensation values) 2)	34	1 to 32	-	-	x
	Value actual input A	21	1 to 32			
	Table input value	22	1 to 32			
	Value actual input B	27	1 to 32			
	Actual output value	35	1 to 32			
	Intermediate point no.	36	1 to 32			
	Actual switching state	37	1 to 32			

1) Unit input resolution (EGF)

2) Unit (value x EGF) : IPO

Type definition of data types 20, 26, 33:

The parameter type at input A, input B or at the output of an IKA configuration is derived from the group number and a value.

Group	Meaning	Status	Value range	Input A	Input B	Output
0	Not activated	read/write	No meaning	–	–	–
1	Feed axis	read/write	1 ... 30	X	X	X
3	Channel	read/write	1 ... 6	–	–	
4	R parameter	read/write	0 ... max. Chan. R. param	X	X	X
			700 ... max. global R par.	X	X	X
5	GI axis	read only	1 ... 30	X	–	X
6	GI spindle	read only	1 ... 6	X	–	X

		Input		Output
Value	Meaning	A	B	
0	Not defined	–	–	–
1	Set position	X	X	X
2	Actual position	X	–	–
5	Override (max. approx. 150% permissible)	–	–	X
6	Compensation valu	–	–	X
10	Global R parameter	X	X	X
11	Channel R parameter Channel 1	X	X	X
⋮	⋮	X	X	X
16	Channel R parameter Channel 6	X	X	X

Type definition of data type 37 "Internal status of IKA":

Final statuses of IKA:

0: IKA deactivated
1: IKA activated
2: IKA deactivated in plus direction
3: IKA deactivated in minus direction

Intermediate status of IKA during transition to a new final status:

4: IKA being deactivate
5: IKA being activated and switched over
6: IKA being switched over

7: IKA deactivated with reference to a position
8: IKA deactivated in plus direction with reference to a position
9: IKA deactivated in minus direction with reference to a position

10: IKA being activated and switched over with reference to a position
11: IKA being activated and switched over in plus direction with reference to a position
12: IKA being activated and switched over in minus direction with reference to a position

13: IKA being switched over with reference to a position
14: IKA being switched over in plus direction with reference to a position
15: IKA being switched over in minus direction with reference to a position

16: IKA is stopped

IKA status display in T55 and T56

Code	Text	Meaning
-2	running	Calculation running
-1	requested	Calculation requested
0	calculated	Curve is calculated
1	0 pointer	Start and end pointer is zero.
2	Start > End	Start pointer is greater than end pointer.
3	Pos. Pt.	The positoin of a point is wrong.
4	Grad. Pt.	The gradient of a point is wrong. The point is laid in T56.
5		Both the table pointers are zero.

Programming example:

Structure of a sine table for IKA, use as cosine for processing

```
                                    ( IKA example 1 )
                                    Machining of a contour with IKA
                                    Y[mm]=100+160/3*COS[5/7*X[mm]])
                                    up to [X,Y]=[252,46.666] to [0,153.333]
                                    Caution: This example does not take tool offset
                                    into account!
                                    0.  Preparation:
N0001  @40c  K11  K2  K0               - Deactivate IKA 2
N0002  G0  X300  Y200                  - Approach tool change point
N0003  R30=0  R31=0                    - Error ID = 0

                                    1. Structure of the table [ika3 data] :
N0005  @40c  K7  K1  K0                -  Input variable 1..13 :
N0010  @40c  K7  K2  K30000                Angle 0, 30, ... , 360 degrees
N0015  @40c  K7  K3  K60000                in units of 10**[-3] degrees
N0020  @40c  K7  K4  K90000
N0025  @40c  K7  K5  K120000
N0030  @40c  K7  K6  K150000
N0035  @40c  K7  K7  K180000
N0040  @40c  K7  K8  K210000
N0045  @40c  K7  K9  K240000
N0050  @40c  K7  K10  K270000
N0055  @40c  K7  K11  K300000
N0060  @40c  K7  K12  K330000
N0065  @40c  K7  K13  K360000
```

```
N0070  @40c  K8  K1  K0                - Output variables 1..13 :
N0075  @40c  K8  K2  K5000                Sine [0], ... , Sine [360 degrees]
N0080  @40c  K8  K3  K8660                in units of 10**[-4]
N0085  @40c  K8  K4  K10000               SIN=1 -> 10000=10 mm
N0090  @40c  K8  K5  K8660
N0095  @40c  K8  K6  K5000
N0100  @40c  K8  K7  K0

N0105  @40c  K8  K8  K-5000
N0110  @40c  K8  K9  K-8660
N0115  @40c  K8  K10  K-10000
N0120  @40c  K8  K11  K-8660
N0125  @40c  K8  K12  K-5000
N0130  @40c  K8  K13  K0

                                  2. Start and end pointer [ika2 data] :
N0135  @40c  K5  K1  K1              - Curve 1 uses points 1...13
N0140  @40c  K6  K1  K13

N0145  @40c  K55  K1  K-1            - Calculate curve 1
N0146  @30c  R30  K55  K1            - [a] read error byte [> R30]
N0147  @111  R30  K0  K150           - [b] case statement for R30:
K1  K281                                Jump list for R30=1 .. 4 [error]
K2  K282                                otherwise continue [R30=0] at N150;
K3  K283                                scan [a] is repeated as long a
K4  K284                                R30=-1 or R30=-127
K127  K-146
K-1  K-146

                                  3. IKA configuration [ika1-Data] :

N0150  @40c  K1  K2  K1              - IKA 2 uses curve 1
N0155  @40c  K40  K2  K1             - Activate extended IKA
N0160  @40c  K43  K2  K1             - Cubic interpolation on

                                     - Definition of input A :
N0165  @40c  K20  K2  K102              Type : Feed axis actual position
N0170  @40c  K2  K2  K1                 No. : 1 [=X]
                                     - Definition of output :
N0175  @40c  K33  K2  K101              Type : Feed axis set position
N0180  @40c  K3  K2  K2                 No. : 2 [=Y]
                                     - Input B=R20 to be used as scaling
N0181  R20=1000                        with : R20=1000 <-> factor=1
                                     - Definition of input B:
N0182  @40c  K26  K2  K411           Type : R parameter channel 1
N0183  @40c  K25  K2  K20            No.: 20
```



```
                                    - Limitations :
N0185  @40c  K31  K2  K500000           Maximum traversing range +-500 mm
N0190  @40c  K32  K2  K-500000

N0195  @40c  K34  K2  K2000              Modific. limitation : 2000 units/IPO cycle
                                         with IPO cycle of 16 ms   : 7500 mm/min
                                    - Weighting I/O :
N0200  @40c  K18  K2  K5                Weighting input : 5/7
N0205  @40c  K19  K2  K7

N0210  @40c  K4  K2  K16                Weighting output  : 16/3000
N0215  @40c  K30  K2  K3000              [note scale factor R20 ! ]
                                         [for R20=1000 the result is 16/3]
                                    - Modulo function and shift :
N0220  @40c  K16  K2  K360000           Weighted input A modulo 360
N0225  @40c  K15  K2  K-90000           Shift input by -90 degrees
                                        to obtain a cosine
                                        therefore Y = Y0+16/3 * SIN[5/7*X-90]
                                                  = Y0+16/3 * COS[5/7*X]

                                  4. Approach, activation and machining:
N0235  G0  X252  Y100               - Approach starting point: 7/5*180=252
@714
N0240  @40c  K11  K2  K1            - Activate IKA 2
N0241  G4  X1.8                     - Wait until starting point is reached
N0245  G1  X0  F1000                - Machining
@714

                                  5. Deactivate and retract
N0250  @40c  K11  K2  K0            - Deactivate IKA 2
@714
G200  Y                             - Synchronization of actual value system
N0255  G0  Y200                     - Retract
N0260  X300
N0270  @100  K300                     Jump to end

                                  6. Errors
N0281  M00  (Error 1)               T55=1: Only one pointer <> 0 !
@100  K300
N0282  M00  (Error 2)               T55=2: End pointer<= start pointer!
@100  K300
N0283  M00  (Error 3)               T55=3: Input[n+1] <= input[n]
N0285  @30c  R31  K56  K1           - Read current point number n [> R31]
@100  K300
N0284  M00  (Error 4)               T55=4: Gradient > 1
N0285  @30c  R31  K56  K1           - Read current point number n [> R31]

                                  7. End
N0300  M02
```

Main group 4 **/** subgroup 1: **Transfer R parameter into setting data**

@410 <Value 1> <Value>

The address of a setting data value is defined in <Value 1>.

Example:

@410 R80	The maximum working area limitation in the 2nd axis is loaded with the contents of parameter R80

@411 <Value 1> <Value>

The byte address of a setting data byte is defined in <Value 1>.

@412 <Value 1> <Value 2> <Value>

The byte address of a setting data byte is defined in <Value 1>.

The bit address (0 to 7) is given in <Value 2>.

@413 <Value 1> <Value 2 > <Value>

The address of a cycle setting data is defined in <Value 2>.

Where:

<Value 1>	Channel No. (0=own channel)
<Value 2>	Word address

@414 <Value 1> <Value 2 > <Value>

The byte address of a cycle setting data is defined in <Value 2>.

Where:

<Value 1>	Channel No. (0: own channel)
<Value 2>	Byte address

@415 <Value 1> <Value 2 > <Value 3> <Value>

The byte address of a cycle setting data bit is defined in <Value 3>.

Where:

<Value 1>	Channel No. (0: own channel)
<Value 2>	Byte address
<Value 3>	Bit address

Main group 4 **/** subgroup 2: **Write R parameter into tool offsets**

@420 <Value 1> <Value 2> <Value 3> <Value>

The numerical value is entered in the tool offset memory. The existing contents of the memory are overwritten.

Thee notations <Value 1> to <Value 3> must be allocated as follows:

<Value 1>	TO area Range: 1 to 4
<Value 2>	Tool offset number (D number) Range: 1 to 204
<Value 3>	Number of tool offset number (P number) Range: 0 to 9**/**15

Example:

@420	**K1**	**K2**	**K3**	**R80**	In TO area 1, the offset memory D2 is loaded with the contents of R80 under tool offset value P3.

Note:

In the notation <Value>, @420 interprets all offsets in the radius.

@423 <Value 1> <Value 2> <Value 3> <Value>

The numerical value is added to the value contained in the tool offset memory.

The notations <Value 1> to <Value 3> must be allocated as follows:

<Value 1>	TO area Range: 1 to 4
<Value 2>	Tool offset number (D number) Range: 1 to 204
<Value 3>	Number of the tool offset value (P number). Range: 0 to 9**/**15

Main group 4 **/** subgroup 3: **Write R parameter into zero offsets**

@430 <Value 1> <Value 2> <Value 3> <Value>

The numerical value is entered in the zero offset memory. The existing contents are overwritten.

The notations <Value 1> to <Value 3> must be allocated as follows:

<Value 1> Group of settable zero offsets (G54 = 1 to G57 = 4)
<Value 2> Number of axis
<Value 3> Coarse or fine value (0 or 1)

Example:

@430 K1 K2 K0 K500 The coarse value of the first settable zero offset is loaded in the second axis with the constant 500.

@431 <Value 1> <Value 2> <Value 3> <Value>

The numerical value is added to the value contained in the zero offset memory.

The notations <Value 1> to <Value 3> must be allocated as follows:

<Value 1> Group of settable zero offsets (G54 = 1 to G57 = 4
<Value 2> Number of axis
<Value 3> Coarse**/**fine changeover (at present, 0 must be entered)

@432 <Value 1> <Value 2> <Value>

Where:

<Value 1> Group of programmable additive zero offsets (G58 = 1 and G59 = 2)
<Value 2> Number of axis

No traversing movements (G00, G01 ...) must be programmed in the same block.

@434 <Value 2> <Value>

The number of the axis of the DRF offset is defined in <Value 2> (see also @435).

@435 <Value 2> <Value>

The number of the axis of the PRESET offset is defined in <Value 2>.

If the PRESET offset is defined with @435 in the AUTOMATIC mode, the offset is effective only after M2**/**M30**/**PRESET (also applies for @434).

@435 is mainly used to reset the PRESET offset at the end of the program.

@437 <Value 1> <Value 2> <Value 3> <Value>

Write settable coordinate rotation.

The notations <Value 1> to <Value 3> must be allocated as follows:

<Value 1> Number of channel (0 = own channel)
<Value 2> Group of settable coordinate rotations (G54 = 1 to G57 = 4)
<Value 3> Number of angle (at present = 1)

@438 <Value 1> <Value 2> <Value 3> <Value>

Write settable additive coordinate rotation (G54 to G57)

Where:

<Value 1> Number of channel
<Value 2> Group 1 to 4 (G54 to G57)
<Value 3> Number of angle (at present = 1)

@439 <Value 1> <Value 2> <Value 3> <Value>

Write programmable coordinate rotation

Where:

<Value 1> Number of channel (0 = own channel)
<Value 2> Group of programmable additive coordinate rotations (G58 = 1 to G59 = 2)
<Value 3> Number of angle (at present= 1)

@43a <Value 1> <Value 2> <Value 3> <Value>

Write programmable additive coordinate rotation.

Where:

<Value 1> Number of channel (0 = own channel)
<Value 2> Number of programmable additive coordinate rotations (G58 = 1 to G59 = 2)
<Value 3> Number of angle (at present= 1)

Main group 4/subgroup 4: **Write R parameters in programmed setpoints, axis/spindle names**

@440 <Value 3> <Value>

This command allows axes to be programmed independent of axis names. The number of the axis to be traversed is stated under <Value 3> and the position to be approached or the traversing path is specified under <Value>.

@441 <Value>

This command allows an axis name to be programmed via the axis number. The axis number is specified under <Value>.

@442 <Value 3> <Value>

This command allows a spindle speed to be programmed independently of the spindle name. The spindle to be programmed is specified under <Value 3> and the required spindle speed under <Value>.

@443 <Value>

This command allows a spindle name to be programmed via the spindle number. The spindle number is specified under <Value>.

@446 <Value>

This command allows the radius to be programmed independent of the address specified in the machine data. The numerical value is stated under <Value>.

@447 <Value>

This command allows the angle to be programmed independent of the address specified in the machine data. The numerical value is stated under <Value>.

@448 <Value 3> <Value>

This command allows the interpolation parameters for circle and thread to be programmed.

Main group 4**/**subgroup 8: **Write R parameters in PLC signal bits**

482 <Value 1> <Value 2> <Value 3> <Value>

@The state of a flag bit in the PLC is loaded via a parameter, pointer or constant.

The PLC number is specified by <Value 1>, the byte address by <Value 2> and the bit address by <Value 3>.

Example:

@482 K1 K2 K0 K0 The state of the defined PLC flag bit is loaded with 0.

@483 <Value 1> <Value 2> <Value 3> <Value 4> <Value>

The state of a data word bit in the PLC is loaded via a parameter, pointer or constant.

The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2>, the data word number by <Value 3> and the bit address by <Value 4>.

Example:

R60 = 1 R62 = 2 R63 = 4
@483 R60 R62 R63 K2 K1 The state of the defined PLC data word bit is loaded with 1.

Main group 4**/**subgroup 9: **Write R parameters in PLC signal bytes**

@493 <Value 1> <Value 2> <Value>

The state of a flag byte in the PLC is loaded via a parameter, pointer or constant.

The PLC number is defined by <Value 1> and the byte address by <Value 2>.

Example:

@493 K1 K1 K01100111 The state of the defined PLC flag byte is loaded with the constant.

@494 <Value 1> <Value 2> <Value 3> <Value>

The state of a data word left in the PLC is loaded via a parameter, pointer or constant.

The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2> and the data word number by <Value 3>.

Example:

R53 = 10010110;
@494 K1 K2 K4 R53 The state of the defined PLC data word left is loaded via R53.

@495 <Value 1> <Value 2> <Value 3> <Value>

The state of a data word right in the PLC is loaded via a parameter, pointer or constant.

The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2> and the data word number by <Value 3>.

Main group 4/subgroup a: **Write R parameters in PLC signal words**

@4a3 <Value 1> <Value 2> <Value 3> <Value>

The state of a flag word in the PLC is loaded via a parameter, pointer or constant.

The PLC number is defined by <Value 1> and the word address by <Value 2>. The dimension identifier <Value 3> specifies how the state of the PLC flag word is to be loaded.

Definition of dimension identifier <Value 3>

<Val. 3>	Fixed point:
0	Value without decimal point
1	Value with decimal point
2	1 place after the decimal point
3	2 places after the decimal point
4	3 places after the decimal point
5	4 places after the decimal point
6	5 places after the decimal point
7	6 places after the decimal point
8	7 places after the decimal point
9	8 places after the decimal point

<Val. 3>	BCD
100	Value without decimal point
101	Value with decimal point
102	1 place after the decimal point
103	2 places after the decimal point
104	3 places after the decimal point
105	4 places after the decimal point
106	5 places after the decimal point
107	6 places after the decimal point
108	7 places after the decimal point
109	8 places after the decimal point

Example:

@4a3 K1 K3 K100 K2219 The state of the defined PLC flag word is loaded in BCD format via a constant.

Main group 4/subgroup b: **Write R parameters in PLC signal data words**

@4b0 <Value 1> <Value 2> <Value 3> <Value 4> <Value 5> <Value>

The fixed-point value of a data word or double word (serial or parallel) in the PLC is loaded via a parameter, pointer or constant.

The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2>, the data word number by <Value 3> and the number of data words by <Value 4>. The dimension identifier <Value 5> specifies how the fixed-point value of the data word or double word is to be loaded. For negative values at least two data words must be specified.

Definition of dimension identifier <Value 5>:

<Val. 5>	Fixed point: Data word or serial double word
0	Value without decimal point
1	Value with decimal point
2	1 place after the decimal point
3	2 places after the decimal point
4	3 places after the decimal point
5	4 places after the decimal point
6	5 places after the decimal point
7	6 places after the decimal point
8	7 places after the decimal point
9	8 places after the decimal point

<Val. 5>	Fixed point: Parallel double word
10	Value without decimal point
11	Value with decimal point
12	1 place after the decimal point
13	2 places after the decimal point
14	3 places after the decimal point
15	4 places after the decimal point
16	5 places after the decimal point
17	6 places after the decimal point
18	7 places after the decimal point
19	8 places after the decimal point

Example:

@4b0 K1 K3 K2 K10 K24500 The fixed-point value of the defined PLC double word is loaded via a constant.

@4b1 <Value 1> <Value 2> <Value 3> <Value 4> <Value 5> <Value>

The BCD value is loaded in the defined PLC data words via a parameter, pointer or constant.

The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2>, the data word number by <Value 3>, the number of data words by <Value 4> and the dimension identifier by <Value 5>.

Depending on the number of data words <Value 4>, the read operation is either serial or parallel:

<Val. 1>	BCD
1	One data word is read, dimension identifier <Value 5> of no significance.
2	Two data words are read in parallel. Dimension identifier <Value 5> of no significance.
3	Three data words are read in serial. Dimension identifier <Value> specifies how the BCD value is to be read.

Definition of dimension identifier <Value 5>:

<Val. 5>	BCD
100	Value without decimal point
101	Value with decimal point
102	1 place after the decimal point
103	2 places after the decimal point
104	3 places after the decimal point
105	4 places after the decimal point
106	5 places after the decimal point
107	6 places after the decimal point
108	7 places after the decimal point
109	8 places after the decimal point

Example:

@4b1 K1 K2 K10 K1 K100 K2413 The BCD value of the defined PLC data word is loaded via a constant.

@4b2 <Value 1> <Value 2> <Value 3> <Value>

The floating-point value is loaded in the defined data words of the PLC via a parameter, pointer or constant. Two data words are always loaded in serial in the PLC.

The PLC number is defined by <Value 1>, the number of the DB or DX by <Value 2> and the data value number by <Value 3>.

The NC value (8 valid digits) is represented in the PLC with only 6 valid digits.

Example:

@4b2 K1 K100 K1 K21 The constant is loaded in the defined data words as a floating-point value.

The NC value (8 valid digits) is represented in the PLC with only 6 valid digits.

Main group 4**/**subgroup c: **Write R parameters in alarms**

@4c0 <Value>

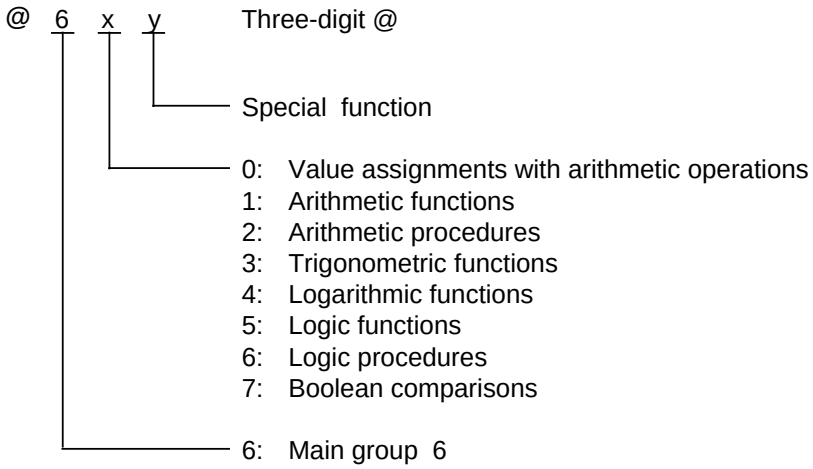
A cycle alarm can be displayed at the control with this command. <Value> specifies the alarm No. with a parameter, constant or pointer.

Example:

@4c0 K4001	The cycle alarm is displayed at the control.
@4ff	see main group 2 **/** Subgroup 0

14.8 Mathematical functions

The main group 6 is organized as follows:



Main group 6 **/** subgroup 0: **Value assignments with arithmetic operations**

In this subgroup, no @ is required. A chain calculation with several notations on the righthand side of the equation is permitted.

<Var>	=	<Value 1>	+	<Value 2>	Addition
<Var>	=	<Value 1>	-	<Value 2>	Subtraction
<Var>	=	<Value 1>	*	<Value 2>	Multiplication
<Var>	=	<Value 1>	**/**	<Value 2>	Division

Main group 6 **/** subgroup 1: **Arithmetic functions**

@610 <Var> <Value>

The unsigned amount from the numerical value defined in <Value > is stored in <Var>.

Example:

R12 = 34

@610 R76 R12		The unsigned amount (=34) from R12 is contained in the R parameter R76.

@613 <Var> <Value>

The square root is formed from the numerical value defined in <Value > and stored in <Var>.

Example:

@613 R13 K64	The square root is taken from the constant (=64) and the result (08) is entered**/**stored in R13.

@614 <Var> <Value 1> <Value 2>

The sum of the squares of the numerical values defined in <Value 1 > and <Value 2 > is formed; the square root is taken from this sum and stored in <Var>.

Example:

R25 = 15 R26 = 20

@614 R77 R25 R26	The square root is formed from the sum of the squares of the R parameters R25 (=225) and R26 (=400). The result (=25) is entered**/**stored in R77.

Main group 6 **/** subgroup 2: **Arithmetic procedures**

@620 <Var>

The contents of the R parameter defined in <Var> are incremented.

Example:

R70 = 1
@620 R70 The contents of parameter R70 are incremented; the new contents are 2.

@621 <Var>

The contents of the R parameter defined in <Var> are decremented.

Example:

R70 = 1
@621 R70 The contents of R70 are decremented; the new contents are 0.

@622 <Var>

The integral part is formed from the numerical value defined by an R parameter or a pointer. The result is then contained in the same R parameter or pointer.

Example:

R60 = 2.9
@622 R60 The integral part of R60 is formed; the new contents are 2.

Main group 6 **/** subgroup 3: **Trigonometric functions**

@630 <Var> <Value>

The sine of the angle defined in <Value> is stored in <Var>.

Example:

R27 = 30
@630 R15 R27 The sine of the contents of R27 (=0.5) is entered**/**stored in R15.

@631 <Var> <Value>

The cosine of the angle defined in <Value> is stored in <Var>.

@632 <Var> <Value>

The tangent of the angle defined in <Value> is stored in <Var>.

@634 <Var> <Value>

The arc sine of the angle value defined in <Value> is stored as arc value in <Var>.

Example:

R35 = 0.70710678
@634 R17 R35 The arc sine of the contents of R35, the result (=45) is entered**/**stored in R17.

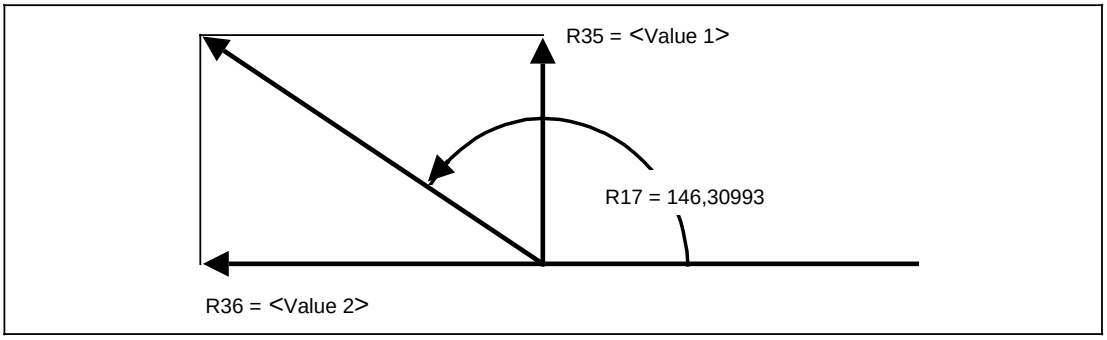
@637 <Var> <Value 1> <Value 2>

The numerical values defined in <Value 1> and <Value 2> are considered as vectors.
The result is the angle between the components given in <Value 2> and the resultant vector.

Only one constant <Const> is allowed as operand for <Value 1> and <Value 2> .
The other operand must be a variable <Var>
(R parameter or pointer).

Example:

R35 = 20 R36 = -30
@637 R17 R35 R36 The angle is formed from the vector components of the contents of the parameters R35 and R36 and the result (=146.30993) is entered**/**stored in R17.



Example for @637

Main group 6 **/** subgroup 4: **Logarithmic functions**

@640 <Var> <Value>

The natural logarithm is formed from the numerical value defined in <Value> and stored in <Var>.

Example:

@640	**R80**	**K10**	The natural logarithm is formed from the constant 10. The result (=2.3025846) is entered**/**stored in R80.

@641 <Var> <Value>

The exponential function e^x is formed from the numerical value defined in <Value> and stored in <Var>.

Example:

@641	**R80**	**K2.5**	The exponential function is calculated for the exponent specified by the constant. The result (=12.182496) is entered**/**stored in R80.

Main group 6 **/** subgroup 5: **Logic functions**

The functions @650 ... @654 perform logic operations on the bit patterns;
Range of values 00000000...11111111

The functions @650 ... @654 perform logic operations on single patterns;
range of values: 0 or 1.

@650 <Var> <Var 1> <Value>

The bit patterns in <Var 1> and <Value> are logically ORed. The result is stored in <Var>.

Example:

R50 = 00101100
R51 = 10110011

@650	**R52**	**R50**	**R51**	The pattern variables R50 and R51 are ORed and the result is stored in R52. R52 has the contents 10111111.

@651 <Var> <Var 1> <Value>

The bit patterns in <Var 1> and <Value> are logically EXORed. The result is stored in <Var>.

@652 <Var> <Var 1> <Value>

The bit patterns in <Var 1> and <Value> are logically ANDed. The result is stored in <Var>.

@653 <Var> <Var 1> <Value>

he bit patterns in <Var 1> and <Value> are logically ANDed. The result is negated and stored in <Var>.

@654 <Var> <Value>

The bit pattern in <Value> is logically negated. The result is stored in <Var>.

Example:

R50 = 00101100

@654	**R52**	**R50**	The contents of the pattern variable R50 are negated and the result stored in R52. R52 has the contents 11010011.

@655 <Var> <Var 1> <Value>

The bits in <Var 1> and <Value > are logically ORed. The result is stored in <Var>.

@656 <Var> <Var 1> <Value>

The bits in <Var 1> and <Value > are logically EXORed. The result is stored in <Var>.

@657 <Var> <Var 1> <Value>

The bits in <Var 1> and <Value > are logically ANDed. The result is stored in <Var>.

Example:

R50 = 1
R51 = 0

@657	**R52**	**R50**	**R51**	A logical AND operation is performed on the Boolean variables R50 and R51 and the result is stored in R52. The contents of R52 are 0.

@658 <Var > <Var 1> <Value>

The bits in <Var 1> and <Value > are logically ANDed. The result is negated and stored in <Var>.

Example:

R50 = 1
R51 = 0

@658	**R52**	**R50**	**R51**	A logical AND operation is performed on the Boolean variables R50 and R51; the result is negated and stored in R52. The contents of R52 are 1.

@659 <Var> <Value>

The logical value (0 or 1) in <Value> is negated. The result is stored in <Var>.

Main group 6 **/** subgroup 6: **Logical procedures**

@660 <Var> <Const>

A bit (0 to 7) is defined by the constant <Const>; this bit is to be deleted in the bit pattern defined by <Var>.

Example:

R60 = 01100111
@660 R60 K6 Bit No. 6 in the pattern variable is deleted.
R60 has the contents 00100111.

@661 <Var> <Const>

The constant <Const> defines a bit (0 to 7) which is to be set to ”1” by the bit pattern defined by <Var>.

Example:

R70 = 00000000
@661 R70 K2 Bit No.2 in the pattern variable is set.
R70 has the contents 00000100.

Main group 6 **/** subgroup 7: **Boolean comparison assignments**

@671 <Var 1> <Var 2> <Value>

f the numerical values defined in <Var 2> and <Value > are equal, the Boolean variable <Var 1> is set to ”1”.

Example:

R50 = 11001100
@671 R51 R50 K11001100 Since R50 is equal to the bit pattern, R51 is set to ”1”.

@672 <Var 1> <Var 2> <Value>

If the numerical variables defined in <Var 2> and <Value> are not equal, the Boolean variable <Var 1> is set to “1”.

@673 <Var 1> <Var 2> <Value>

If the numerical value define in <Var 2> is greater than that in <Value>, the Boolean variable <Var 1> is set to “1”.

@674 <Var 1> <Var 2> <Value>

If the numerical value defined in <Var 2> is greater than or equal to that in <Value>, the Boolean variable <Var 1> is set to “1“.

@675 <Var 1> <Var 2> <Value>

If the numerical value defined in <Var 2> is less than that in <Value>, the Boolean variable <Var 1> is set to “1“.

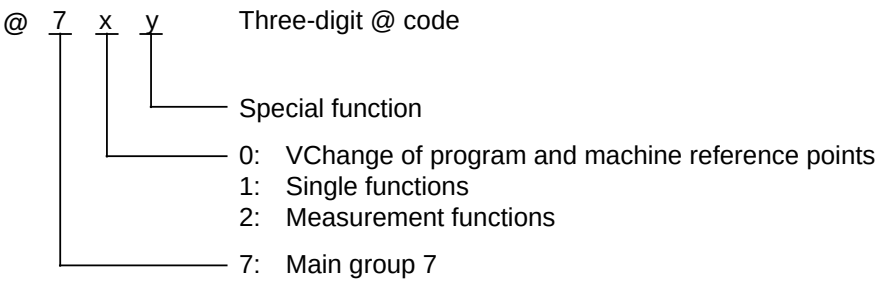
@676 <Var 1> <Var 2> <Value>

If the numerical value defined in <Var 2> is less than or equal to that in <Value>, the Boolean variable <Var 1> is set to “1“.

If the condition is not fulfilled with functions @671...@676 then the Boolean Variable <Var 1> is set to ”0”.

14.9 NC-specific functions

The main group 7 is organized as follows:



Main group 7 **/** subgroup 0:	**Change of program and machine reference points**

@706

With command @706, a position is specified, referred to the machine zero, and approached from the specified axis.The command is effective only in the block in which it is specified.

As many axes can be specified as can be traversed simultaneously by the NC. The positions of the axes to be approached are programmed either in DIN code or by means of the command @440 ... .

The command @706 suppresses all zero offsets (settable, settable additive, programmable and external) as well as the PRESET and DRF offsets.

In order to approach a fixed machine point, the tool offsets must also be deselected.

If machine data 5007.1 has been set, G53 acts like @706.

If the machine data has not been set, the PRESET and DRF offsets are not suppressed with G53.

Example:

@706 X1000 Z500	The programmed traversing paths in X and Z are approached referred to the machine zero.

Main group 7 **/** subgroup 1 : **Single functions**

@710 <Var 1> <Var 2>

This command is required as reference preparation in turning cycle L95 ”Stock removal”.

With this command, a contour programmed in a subroutine is broken down into individual blocks, the data being stored in R parameters.

The contour element is stored in a total of 8 R parameters, starting at the R parameter <Var 1> (Rn).
Reference preparation requires a total of 4 R parameters as input data. The first of these R parameters is defined by <Var 2> (Rm).
The input control parameter (Rm+3) must be set to ”1” before the first call of @710.
The first contour element of the subroutine is stored in R parameters, beginning at the starting point of the contour (Rm +1, Rm +2). This point is not programmed in the subroutine. The command @710 sets the control parameter to ”0”, so that the values from the next contour element are loaded each time this command is called. When the command recognizes ”End of subroutine” (M17), the output control parameter (Rn+7) is automatically set to ”1”or ”2”.

Prerequisites

1. Contour in the subroutine
2. Starting points
3. Control parameter (Rm + 3) = 1

The command @710 must not be programmed on its own in ”MDA” mode. It must be in a subroutine.

Parameter

Rm	Subroutine number	< Var 2 >
Rm +1	Starting point X	
Rm +2	Starting point Z	
Rm +3	Control parameter	

@710 loads in:

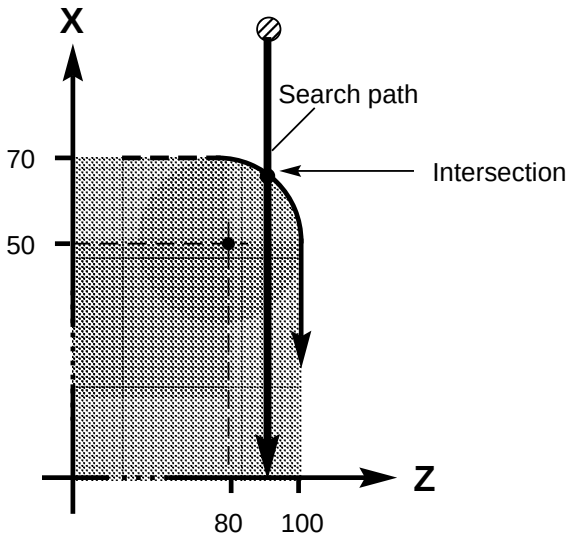
Rn	Starting point X	< Var 1 >
Rn +1	Starting point Z	
Rn +2	End point X	
Rn +3	End point Z	
Rn +4	Interpolation parameter I	
Rn +5	Interpolation parameter K	
Rn +6	G function	
Rn +7	Control parameter	

Rn +7 = 0:	Block without M17
Rn +7 = 1:	Block with M17
Rn +7 = 2:	M17 alone in block

@711 <Var 1> <Var 2> <Var 3>

This command is required as intersection calculation in turning cycle L95 “Stock removal”.

With @711, the intersection of a search path with a contour element is determined.
The intersection calculation requires a total of 8 parameters as input data for the first contour element. The first R parameter number is defined by <Var 2> (Rn). The second contour from <Var 3> (Rr) onwards is not presently implemented, but one R parameter or another must be specified when programming. The output data of the intersection calculation is stored in a total of 3 R parameters starting at R parameter <Var 1> (Rm). The direction of search is programmed after the command as normal traversing movement. Both axis values are required in order to determine the intersection. The output data of reference preparation are used here as input data for the intersection calculation.



		Register contents
Rn	Beginning of block X	70
Rn +1	Beginning of block Z	80
Rn +2	End of block X	50
Rn +3	End of block Z	100
Rn +4	Interpolation parameter I	- 20
Rn +5	Interpolation parameter K	0
Rn +6	G function	2

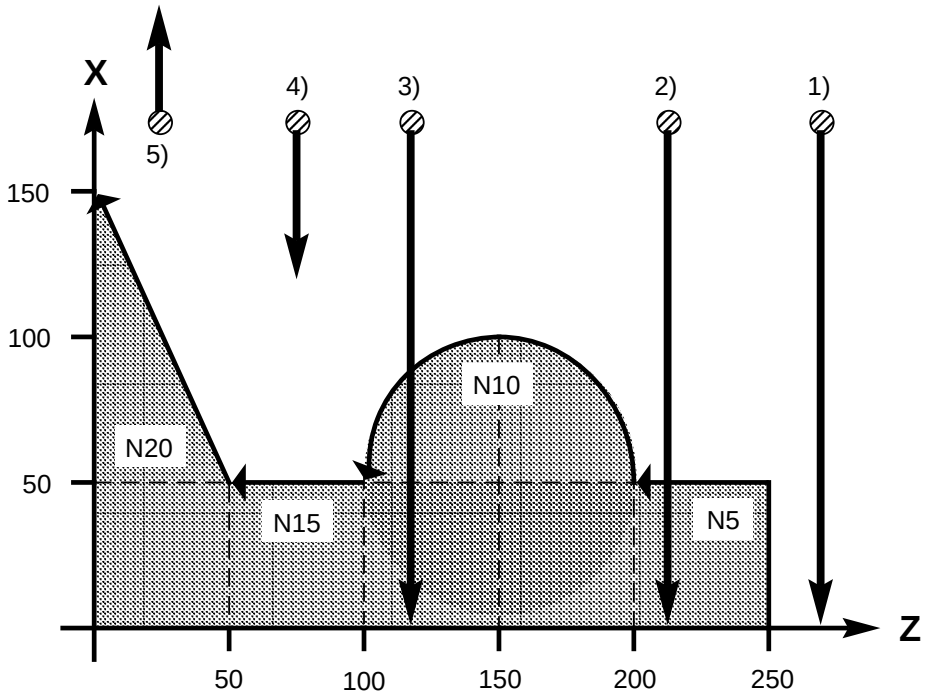
@711 Rm Rn Rr G01 G90 X0 Z90 L_F

Direction of search, loads in:

Rm	Result	1*)
Rm +1	Intersection X	67.320
Rm +2	Intersection Z	90.000

*) 1=found, 0=not found

Results of intersection calculations with @711



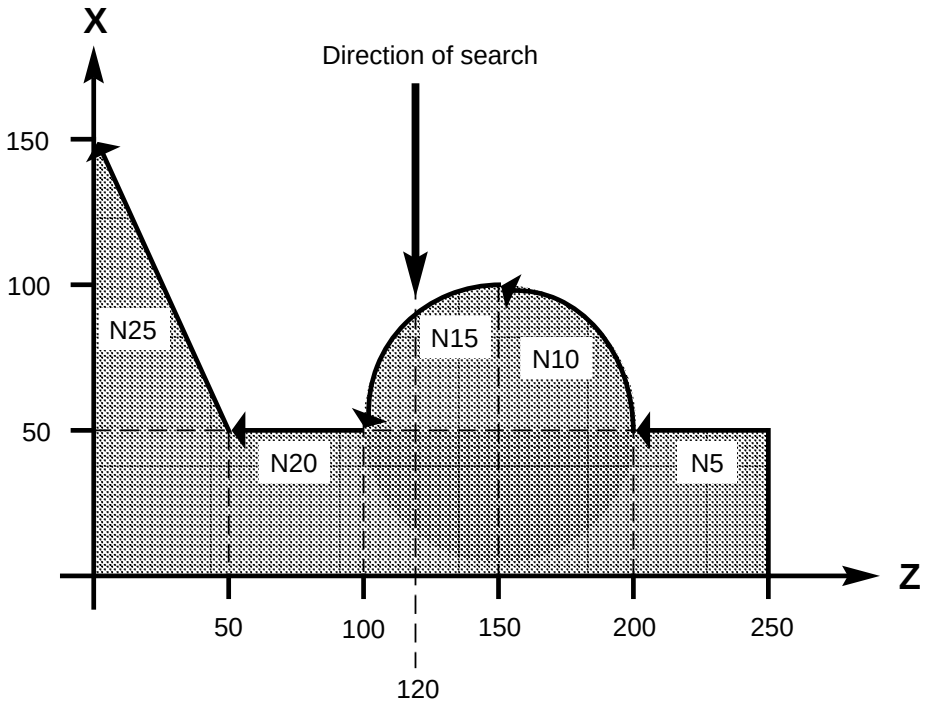
1) Intersection not found
2) Intersection found
3) Circular arc extends beyond the quadrant boundary(!):
   Intersection not found
4) Search path was too short:
   Intersection found nonetheless
5) Wrong direction of search:
   Intersection found nonetheless

Together with the direction of search, a path is defined along which the intersection is expected. An intersection is always found when this path or its extension encounters the contour stored in R_n to (R_n+6). An interface is found even if the path programmed after @711 points in the wrong direction or is too short.

The function @711 now enters the value 0 into register R_m if no intersection has been found, or the value 1 if the search was successful. If an intersection has been found, the parameters (R_m +1) and (R_m +2) contain the intersection coordinates , otherwise the value 0. If a traverse movement is to be programmed with the values from (R_m+1) and (R_m+2), it is necessary first of all to enquire in a loop whether an intersection has been found or not.

Example for intersection calculation with @711

A search is made for the intersection with a programmed contour at Z = 120.



Subroutine for the contour:

```
L20
N5   X50   Z200
N10  G03   X100  Z150  I0    K-50
N15  G03   X50   Z100  I-50  K0
N20  X50   Z50
N25  X150  Z0    M17
```

Main program:

```
%30
G0      X150    Z120
R50=20  R51=0   R52=250  R53=1  R70=0
N300    @131    R70      K0     K305
@710    R54     R50
@711    R70     R54      R62    G01  G90  X0  Z120
@100    K-300
N305    G00     X=R71    Z=R72
.
.
.
M30
```

The program loop is run through until an intersection is found.

	Rn	Rn+1	Rn+2	Rn+3	Rn+4	Rn+5	Rn+6	Rn+7
1st call @710	50	250	50	200	0	0	1	0
@711	INTERSECTION FOUND? *No:* Rm = 0							
2nd call @710	50	200	100	150	0	–50	3	0
@711	INTERSECTION FOUND? *No:* Rm = 0							
3rd call @710	100	150	50	100	–50	0	3	0
@711	INTERSECTION FOUND? *Yes:* Rm = 1							

After the 3rd call, the intersection in this example is found at X=90 (R71) and Z=120 (R72).

@713 <Var >
With this command, the numerical value corresponding to the safety gap of 1 mm in the current input format is loaded in the R parameter defined with <Var> (for G70 = 0.03937 and G71 = 1). In the R parameters immediately following, the numerical value “1” is loaded in the case of radius programming and “2”in the case of diameter programming. The permissible parameter range for <Var> is R0 to R98 and R900 to R998.

@714
With this command, “STOP-DEC”, block preparation (decoding) is stopped until the buffer is empty.
When the program is being executed, several program blocks are decoded in advance in the control and loaded into the buffer of the NC. As a result, program execution is faster, but together with certain NC commands (reading of actual values, measurements, data transfer NC-PLC) this can lead to errors occurring in the course of the program. By means of the STOP-DEC command (stop decoding), the advance decoding of NC blocks written after this command is stopped until the block containing the STOP-DEC command has been executed. This results in the buffers being emptied and information required in the next NC blocks being made available.

The STOP-DEC 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
- Tool offsets
- Zero offsets
- R parameters
- “Mirror” signal

The STOP-DEC command must be programmed in the own channel each time before reading actual values and after each measurement.

The STOP-DEC command must always be given in a separate NC block.

Example:

M94	The part number is transferred from the PLC in R60.
@714	Stop decoding so that the current part number can be evaluated in the next NC block.
@123 R60 K100 K5	Branch in accordance with the path number.

@715

If the angle of rotation of the settable or programmable coordinate rotation is loaded with the help of the CL 800 language, this angle value is used immediately for the calculation. This can result in this angle being included in earlier traversing blocks. If the angle is not to be valid until the next block, all the previous blocks must be executed (empty buffer). This can be achieved by programming STOP DEC1 even if TRC is selected. The decoding is only activated when all buffers have been emptied before coordinate rotation.

The STOP-DEC1 command must always be programmed in a separate NC block.

Main group 7 **/** subgroup 2 : **Measurement functions**

@720 <Var> <Value>

This command is used in measuring cycles.

With the measurement function, the actual values of the traversing axes are determined at the time of an input signal from the probe. The actual values are recorded direct from the measurement circuits of the control when the switching edge of the probe signal is recognized. The actual values are stored in ascending sequence of axis number as from parameter <Var> and are referred to machine zero. The control subsequently generates a ”Delete distance-to-go”, i.e. the distances to go (differences between setpoints and actual values) are deleted for all axes. Set value 0 is specified as jump function by the control. The deceleration paths of the axes are still traversed, i.e. the following errors are reduced. This means that the next traverse blocks must be programmed in absolute dimensions (G90).

The actual values recorded at the time of measuring the traversing axes are stored in ascending sequence of axis number starting at the parameter defined in <Var>.

The number of the measurement input (1 or 2) is specified with <Value>.

The traversing paths of the axes (setpoint positions) are programmed via DIN code or the command @440 in the same NC block. The setpoint positions of the axes are referred to workpiece 0.

Example:

@720	**R93**	**K1**	The actual value of the 1st axis is measured and
@440	**K1**	**R70**	loaded in R93.
@714			

Notes:

- Depending on a machine data bit, with rotary axes the result is stored in a total of two R parameters from the R parameter <Var> (RN).

  The following parameters are loaded:
  Rn = Position within a rotation
  Rn + 1 = Number of rotations

- A @720 command cannot be programmed in conjunction with software axes.
- If a probe is activated, it affects all @720 blocks even if they have been parameterized for a different probe input and are running in a different channel. For this reason @720 must only be active once.

@736 <Value 1> <Value 2> <Value 3> [1)]

With this command you can delete the distance to go within an NC block depending on the signal state of an external input. With this command, an axis-specific distance to go can be implemented for a positioning axis or for all the contouring axes within one function call.

The action of the function is non modal. One NC block can contain several delete distance to go functions which, depending on the parameterization, affect either contouring axes or positioning axes. The axis and its coordinate must be programmed before its @736 function. @736 functions for the contouring axes and maximum 5 positioning axes can be programmed in one block.

If several external inputs are to affect one axis, several @736 commands must be written for the same axis. Please make sure that all the inputs which are to affect the same axis are in one input byte.

Meaning:

<Value 1>	Axis number 1...n for positioning axes Axis number 0 for contouring axes	
<Value 2>	Channel address	0=CSB inputs 1=1st byte of the 1st MIXED I/O 2=2nd byte of the 2nd MIXED I/O 3=1st byte of the 2nd MIXED I/O 4=2nd byte of the 2nd MIXED I/O
<Value 3>	Bit address 0..7 for MIXED I/O 6 or 7 when using CSB plus sign when evaluating the positive edge minus sign when evaluating the negative edge	

Notes:

- If @736 is programmed when scale modification is active, alarm 3000, "General programming error" is triggered.
- Delete distance to go for active axes of rotation in coordinate rotation triggers alarm 2087, "Coordinate rotation not allowed".
- Up to SW 5.2, delete distance to go for active mirror axes triggers alarm 2061 "General programming error".
  With SW 5.4 and higher, @736 can also be used together with mirroring.

Program examples:

```
N10 P[X]100 F[X]50 @736 K1 K2 K4 @736 K1 K2 K–2
```

Delete distance to go of the X axis is activated by 2 inputs on the MIXED I/O either by a positive edge at input 4 of the 2nd channel or by a negative edge at input 2 of the 2nd channel.

```
N10 X1000 Y500 @736 K0 K0 K–6
```

Delete distance to go of all the contouring axes (X and Y) is activated via input 6 on the CSB by a negative edge.

```
N10 X100 Z200 F500 P[A]50 F[A]125 @736 K0 K1 K2 @736 K4 K1 K3
```

Delete distance to go of the contouring axes (X and Z) is activated by the positive edge at input 2 of the 1st channel and that of positioning axis A (4th axis acc. to machine data) by the positive edge at input 3 of the 1st channel.

---

1) SW 3 and higher

14.10 @ 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 : Instuction1 ; . . = Value n : Instuction n ;	Case branch
@12x *Var Value Const*	IF “Condition” [2] THEN Instruction 1 ; [ELSE Instruction 2 ;] END IF;	IF-THEN-ELSE instruction x compare 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 compare operator vop
@14x *Var* Value *Const*	REPEAT Instruction ; UNTIL “Condition”; [2]	Repetition instruction with enquiry for repetition condition at the end. x compare operator vop

Note:

The explanation of symbols given below will appear unchanged on the following pages to ensure consistency of footnotes even though not all of them do actually occur in the table.

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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 *Var Value 2 Const*	FOR Var = Value 1 TO Value 2 DO Instruction ;	Repetition instruction with repetitions until *Var* has reached *Value 2* incrementally
@161 *Var Value 2 Const*	FOR Var = Value 1 DOWNTO Value 2 DO Instruction ;	Repetition instruction with repetitions until *Var* has reached *Value 2* decrementally
@200 *Var*	CLEAR(Var);	Delete variable
@201 *Var Value*	Var = Value	Load variable with value
@202 *Var 1 Var 2*	XCHG (Var 1 , Var 2);	Exchange the variable contents [3]
@203 *Var 1 Var 2* Const		Read a bit from bit pattern [3]
@210 *Value 3 Value 4*	CLEAR MIB (Value 3 , Value 4);	Delete input buffer all
@211 *Var Value*	Var = MIB (Value);	Read input buffer call
@212 *Value Value 1*	MIB (Value)= Value 1 ;	Load input buffer call
@300 *Var Value 1*	Var =MDN (Value 1);	Machine data NC Value 1:
@301 *Var Value 1*	Var =MDNBY (Value 1);	Machine data NC bytes Value 1: Byte addr.
@302 *Var Value 1 Value 2*	Var =MDNBI (Value 1 , Value 2);	Machine data NC bits Value 1: Byte addr. Value 2: Bit addr. 0 . . . 7
@303 *Var Value 1 Value 2*	Var =MDZ (Value 1 , Value 2);	Read cycle machine data values

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@304 *Var Value 1 Value 2*	Var =MDZBY (Value 1 , Value 2);	Read cycle machine data bytes
@305 *Var Value 1 Value 2 Value 3*	Var =MDZBI (Value 1 , Value 2 , Value 3);	Read cycle machine data bits
@306 *Var Value 1*	Var =MDP (Value 1);	Machine data PLC Value 1:
@307 *Var Value 1*	Var =MDPBY (Value 1);	Machine data PLC bytes Value 1: Byte addr.
@308 *Var Value 1 Value 2*	Var =MDPBI (Value 1 , Value 2);	Machine data PLC bits Value 1: Byte addr. Value 2: Bit addr. 0 . . . 7
@30c *Var Value 1 Value 2*	Var =MDIKA (Value 1 , Value 2);	Read IKA data Value 1: data type Value 2: data no.
@310 *Var Value 1*	Var =SEN (Value 1);	Setting data NC Value 1:
@311 *Var Value 1*	Var =SENBY (Value 1);	Setting data NC bytes Value 1: Byte addr.
@312 *Var Value 1 Value 2*	Var =SENBI (Value 1 , Value 2);	Setting data NC bits Value 1: Byte addr. Value 2: Bit addr. 0...7
@313 *Var Value 1 Value 2*	Var =SEZ (Value 1 , Value 2);	Read cycle setting data
@314 *Var Value 1 Value 2*	Var =SEZBY (Value 1 , Value 2);	Read cycle setting data bytes
@315 *Var Value 1 Value 2 Value 3*	Var =SEZBI (Value 1 , Value 2 , Value 3);	Read cycle setting data bits
@320 *Var Value 1 Value 2 Value 3*	Var =TOS (Value 1 , Value 2 , Value 3);	Tool offset Value 1: 1 to 4 Value 2: D No. 1 . . 204 Value 3: P No. 0 . . 9**/**15

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@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 30 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 30
@332 *Var Value 2*	Var =ZOE (Value 2);	External offset from PLC Value 2: Axis No. 1 to 30
@333 *Var Value 2*	Var =ZOD (Value 2);	DRF offset Value 2: Axis No. 1 to 30
@334 *Var Value 2*	Var =ZOPS (Value 2);	PRESET offset Value 2: Axis No. 1 to 30
@336 *Var Value 2*	Var =ZOS (Value 2);	Cumulative offset Value 2: Axis No. 1 to 30
@337 *Var Value 1 Value 2 Value 3*	Var =ZOA DW (Value 1), Value 2 , Value 3);	Settable coordinate rotation (G54-G57) Value 1: Channel No. 0 to 6 Value 2: Group 1 or 6 (G54-G57) Value 3: = Angle No. (1)
@338 *Var Value 1 Value 2 Value 3*	Var =ZOPR DW (Value 1), Value 2 , (Value 3);	Programmable coordinate rotation (G58-G59) Value 1: Channel No. 0 to 6 Value 2: Group 1 or 4 (G58-G59) Value 3: = Angle No. (1)
@339 *Var Value 2 Value 3*	Var =ZOS, Value 2 , Value 3 ;	Reading of the cumulative offset of an axis from a channel to be specified Value 2: Axis number Value 3: Channel number
@345 *Var Value 1 Value 2*	Var =PRVC (Value 1 , (Value 2);	Programmed cutting rate Value 1: Channel No. 0 to 6 Value 2: 0

@ Code	CL-800 instruction	Function
@360 *Var Value 2*	Var =ACPW (Value 2);	Axis position actual referred to workpiece Value 2: Axis No. 1 to 30
@361 *Var Value 2*	Var =ACPM (Value 2);	Axis position actual referred to machine Value 2: Axis No. 1 to 30
@362 *Var Value 2*	Var =ACSP (Value 2);	Read current axis position Value 2: Axis No. 1 to 30
@363 *Var Value 2*	Var =ACSP (Value 2);	Spindle position actual value Value 2: Spindle No. 1 to 6
@364 *Var Value 2*	Var =ACSS (Value 2);	Spindle speed actual value Value 2: Spindle No. 1to 6
@367 *Var Value 1*	Var =ACAS (Value 1);	Read axis No. of current level/leadscrew No.in R parameter Var : Var+0: No. of horizontal axis Var+1: No. of perpend. axis Var+2: No. of axis perpend. to plane Var+3: No. of axis in which length is effective (tool type 30) Var+4: No. of leadscrew Value 1: Channel No. 0 to 6
@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);	Read G function from working memory of current block Value 1: Channel No. 0 to 6 Value 3: Internal G group to which G function belongs (see program key)

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@36e *Var Value 2*		Value 2: Spindle number
@371 *Var Value 1 Value 3*	Var =SOB (Value 1 , Value 3);	Special bits Value 1: Channel No. 0 to 4 Channel-dependent 0 to n Channel-independent Value 3: Bit No. 0 . . . 7
@372 *Var*	Var =PPCH;	Read current channel No. of program
@380 *Var Value 1 Value 2 Value 3*	Var =PLCI (Value 1 , Value 2 , Value 3);	PLC input bit Value 1: PLC No. 1 Value 2: Byte addr. 0...127 Value 3: Bit No. 0...7
@381 *Var Value 1 Value 2 Value 3*	Var =PLCQ (Value 1 , Value 2 , Value 3);	PLC output bit Value 1: PLC No. 1 Value 2: Byte addr. 0...127 Value 3: Bit No. 0...7
@382 *Var Value 1 Value 2 Value 3*	Var =PLCF (Value 1 , Value 2 , Value 3);	PLC flag bit Value 1: PLC No. 1 Value 2: Byte addr. 0...255 Value 3: Bit No. 0...7
@383 *Var Value 1 Value 2 Value 3 Value 4*	Var =PLCW (Value 1 , Value 2 , Value 3 , Value 4);	PLC data word bit Value 1: PLC No. 1 Value 2: Byte addr. 0...255 DX No. 1000...1255 Value 3: DW No. 0...2043 Value 4: Bit No. 0...15
@390 *Var Value 1 Value 2*	Var =PLCIB (Value 1 , Value 2 ;	PLC input byte Value 1: PLC No. 1 Value 2: Byte addr. 0...127
@391 *Var Value 1 Value 2*	Var =PLCQB (Value 1 , Value 2 ;	PLC output byte Value 1: PLC No. 1 Value 2: Byte addr. 0...127
@392 *Var Value 1 Value 2*	Var =PLCPB (Value 1 , Value 2 ;	PLC peripheral byte Value 1: PLC No. 1 Value 2: Byte addr. 0...127

Explanation of symbols:

x compare 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
- 7: . . . . true
- 8: . . . . not

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
@393 *Var Value 1 Value 2*	Var =PLCFB (Value 1 , Value 2 ;	PLC flag byte Value 1: PLC No.1 Value 2: Byte addr. 0...255
@394 *Var Value 1 Value 2 Value 3*	Var =PLCDBL (Value 1 , Value 2 , Value 3);	PLC data word left Value 1: PLC No.1 Value 2: DB No.1...255 DX No.1000...1255 Value 3: DW No.0...2043
@395 *Var Value 1 Value 2 Value 3*	Var =PLCDBR (Value 1 , Value 2 , Value 3);	PLC data word right Value 1: PLC No.1 Value 2: DB No.1...255 DX No.1000...1255 Value 3: DW No.0...2043
@3a0 *Var Value 1 Value 2 Value 3*	Var =PLCIW (Value 1 , Value 2 , Value 3);	PLC input word Value 1: PLC No.1 Value 2: Word addr. 0...126 Value 3: DIM identifier 0...9 fixed point 100...109 BCD
@3a1 *Var Value 1 Value 2 Value 3*	Var =PLCQW (Value 1 , Value 2 , Value 3);	PLC output word Value 1: PLC No.1 Value 2: Word addr. 0...126 Value 3: DIM identifier 0...9 fixed point 100...109 BCD
@3a2 *Var Value 1 Value 2 Value 3*	Var =PLCPW (Value 1 , Value 2 , Value 3);	PLC peripheral word Value 1: PLC No.1 Value 2: Word addr. 0...126 Value 3: DIM identifier 0...9 fixed point 100...109 BCD
@3a3 *Var Value 1 Value 2 Value 3*	Var =PLCFW (Value 1 , Value 2 , Value 3);	PLC flag word Value 1 : PLC No.1 Value 2 : Word addr. 0...254 Value 3 : DIM identifier 0...999 fixed point 100...109 BCD

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@3a4 *Var Value 1 Value 2*	Var =PLCT (Value 1 , Value 2 ;	PLC timer Value 1: PLC No. 1 Value 2: PLC count. 0...255
@3a5 *Var Value 1 Value 2*	Var =PLCC (Value 1 , Value 2 ;	PLC counter Value 1: PLC No. 1 Value 2: Coun. addr. 0...255
@3b0 *Var Value 1 Value 2 Value 3 Value 4 Value 5*	Var =PLCDF (Value 1 , Value 2 , Value 3 (Value 4 , Value 5);	PLC data word fixed point Value 1: PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043 Value 4: No. of DW 1 or 2 Value 5: DIM identifier 0...9 serial 10...19 parallel
@3b1 *Var Value* 1 *Value 2 Value 3 Value 4 Value 5*	Var =PLCDB (Value 1 , Value 2 , Value 3 (Value 4 , Value 5);	PLC data word BCD Value 1 : PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043 Value 4: No. of DW 1...3 Value 5: DIM identifier 100...109 parallel
@3b2 *Var Value 1 Value 2 Value 3*	Var =PLCDG (Value 1 , Value 2 , Value 3);	PLC data word Value 1: PLC No. 1 Value 2: DB No. 1...255 Dx No. 1000...1255 Value 3: DW No. 0...2043
@3c0 *Var*	Var =ALNP ();	NC alarms
@3d0 *Var*	Var =ALNPZ ();	NC alarm pointer
@3ff *Value 1 Value 2 Value 3 Value 4 Value 5*		Read from R parameters Value 1: R_{Target} 1-9999 Value 2: K25 Value 3: K0 Value 4: $K_{RSource}$ 1-9999 Value 5: $K_{Channel}$ 0 ... 5
@400 *Value 1 Value*	MDN (Value 1)= Value ;	Machine data NC Value 1:
@401 *Value 1 Value*	MDNBY (Value 1)= Value ;	Machine data NC bytes Value 1: Byte addr.

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@402 *Value 1 Value 2 Value*	MDNBI (Value 1 , Value 2) = Value ;	Machine data NC bits Value 1: Byte addr. Value 2: Bit addr. 0 . . . 7
@403 *Value 1 Value 2 Value 3*	Var =MDZ (Value 1 , Value 2) = Value ;	Write cycle machine data values
@404 *Value 1 Value 2 Value*	Var =MDZBY (Value 1 , Value 2) = Value ;	Write cycle machine data bytes
@405 *Value 1 Value 2 Value*	Var =MDZBI (Value 1 , Value 2) = Value 3 = Value ;	Write cycle machine data bits
@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.
@408 *Value 1 Value 2 Value*	MDNBI (Value 1 , Value 2) = Value ;	Machine data PLC bits Value 1: Byte addr. Value 2: Bit addr. 0 . . . 7
@40c *Var Value 1 Value 2 Value*	Var =MDIKA (Value 1 , Value 2) =<Value>	Write IKA data Value 1: data type Value 2: data no. Value: Value to be written
@410 *Value 1 Value*	SEN (Value 1)= Value ;	Setting data NC bits Value 1:
@411 *Value 1 Value*	SENBY (Value 1)= Value ;	Setting data NC bytes Value 1: Byte addr.
@412 *Value 1 Value 2 Value*	SENBI (Value 1 , Value 2) = Value ;	Setting data NC bits Value 1: Byte addr. Value 2: Bit addr. 0 . . . 7
@413 *Value 1 Value 2 Value*	SEZ (Value 1 , Value 2) = Value ;	Write cycle setting data

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@414 *Value 1 Value 2 Value*	SEZBY (Value 1 , Value 2) = Value ;	Write cycle setting data bytes
@415 *Value 1 Value 2 Value 3 Value*	SEZBI (Value 1 , Value 2 , Value 3) = Value ;	Write cycle setting data bits
@420 *Value 1 Value 2 Value 3 Value*	TOS (Value 1 , Value 2 , Value 3)= Value ;	Tool offset Value 1: 1 to 4 Value 2: D No. 1 . . 204 Value 3: P No. 0 . . 9**/**15
@423 *Value 1 Value 2 Value 3 Value*	TOAD (Value 1 , Value 2 , Value 3)= Value ;	Tool offset additive Value 1: 1 to 4 Value 2: D No. 1 . . 204 Value 3: P No. 0 . . 9**/**15
@430 *Value 1 Value 2 Value 3 Value*	ZOA (Value 1 , Value 2 , Value 3)= Value ;	Settable zero offset (G54-G57) Value 1: Group 1 . . . 4 (G54-G57) Value 2: Axis No. 1 to 30 Value 3: 0**/**1 (coarse**/**fine)
@431 *Value 1 Value 2 Value 3 Value*	ZOFA (Value 1 , Value 2 , Value 3)= Value ;	Settable zero offset additive Value 1: Group 1 . . . 4 (G54 to G57) Value 2: Axis No. 1 to 30 Value 3: 0**/**1 (coarse**/**fine)
@432 *Value 1 Value 2 Value*	ZOPR (Value 1 , Value 2)= Value ;	Programmable zero offset (G58,G59) Value 1: Group 1 or 2 (G58 or G59) Value 2: Axis No. 1 to 30
@434 *Value 2 Value*	ZOD (Value 2)= Value ;	DRF offset Value 2: Axis No. 1 to 30
@435 *Value 2 Value*	ZOPS (Value 2)= Value ;	PRESET offset Value 2: Axis No. 1 to 30

Explanation of symbols:

x compare 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
- 7: . . . . true
- 8: . . . . not

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
@437 *Value 1 Value 2 Value 3 Value*	ZOADW (Value 1 , Value 2 , Value 3) = Value ;	Settable coordinate rotation Value 1: Channel No. 0 to 6 Value 2: Group 1 to 4 (G54-G57) Value 3: = 1
@438 *Value 1 Value 2 Value 3 Value*	ZOFADW (Value 1 , Value 2 , Value 3) = Value ;	Settable coordinate rotation additive Value 1: Channel No. 0 to 6 Value 2: Group 1 to 4 Value 3: 1
@439 *Value 1 Value 2 Value 3 Value*	ZOPRDW (Value 1 , Value 2 , Value 3) = Value ;	Programmable coordinate rotation Value 1: Channel No. 0 to 6 Value 2: Group 1 or 2 (G58-G59) Value 3: 1
@43a *Value 1 Value 2 Value 3 Value*	ZOFPRDW (Value 1 , Value 2 , Value 3) = Value ;	Programmable coordinate rotation additive Value 1: Channel No. 0 to 6 Value 2: Group 1 or 2 (G58-G59) Value 3: Angle No. (1)
@440 *Value 3 Value*	ZOPS (Value 3)= Value ;	Programmed axis position Value 3: Axis No. 1 to 12
@441 *Value*		Programmable axis names via the axis number
@442 *Value 3 Value*		Programmable spindle speed Value 3: = 0 (master spindle) = 1 ... 6 (spindle number)
@443 *Value*		Programmable spindle names via the spindle number
@446 *Value*	PRAD= Value ;	Programmed radius
@447 *Value*	PANG= Value ;	Programmed angle
@448 *Value 3 Value*	PRIP (Value 3)= Value ;	Programmed interpolation parameter Value 3: Axis No. 1 to 30
@482 *Value 1 Value 2 Value 3 Value*	PLCF (Value 1 , Value 2 , Value 3) = Value ;	PLC flag bit Value 1: PLC No. 1 Value 2: Byte addr. 0...255 Value 3: Bit No. 0...7

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@483 *Value 1 Value 2* *Value 3 Value 4* *Value*	PLCW (Value 1 , Value 2 , Value 3) = Value ;	PLC data word bit Value 1: PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043 Value 4: Bit No. 0...15
@493 *Value 1 Value 2* *Value*	PLCFB (Value 1 , Value 2) = Value ;	PLC flag byte Value 1: PLC No. 1 Value 2: Byte addr. 0...255
@494 *Value 1 Value 2* *Value 3 Value*	PLCDBL (Value 1 , Value 2 , Value 3) = Value ;	PLC data word left Value 1: PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043
@495 *Value 1 Value 2* *Value 3 Value*	PLCDBR (Value 1 , Value 2 , Value 3) = Value ;	PLC data word right Value 1: PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043
@4a3 *Value 1 Value 2* *Value 3 Value*	PLCFW (Value 1 , Value 2 , Value 3) = Value ;	PLC flag word Value 1: PLC No. 1 Value 2: Word addr. 0...254 Value 3: DIM identifier 0...9 Fixed point 100...109 BCD
@4b0 *Value 1 Value 2* *Value 3 Value 4* *Value 5 Value*	PLCDF (Value 1 , Value 2 , Value 3 , Value 4 Value 5) = Value ;	PLC data word fixed points Value 1: PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043 Value 4: No. of DW 1 or 2 Value 5: DIM identifier 0 ... 9 serial 10 ... 19 parallel

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@4b1 *Value 1 Value 2 Value 3 Value 4 Value 5 Value*	PLCDB (Value 1 , Value 2 , Value 3 , Value 4 Value 5) = Value ;	PLC data word BCD Value 1: PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043 Value 4: No. of DW 1 ... 2 Value 5: DIM identifier 100... 109 BCD
@4b2 *Value 1 Value 2 Value 3 Value*	PLCDG (Value 1 , Value 2 , Value 3) = Value ;	PLC data word floating point Value 1: PLC No. 1 Value 2: DB No. 1...255 DX No. 1000...1255 Value 3: DW No. 0...2043
@4c0 *Value*	ALNZ()= Value ;	Cycle alarms Value : Alarm No. 4000...4299 5000...5299
@4ff *Value 1 Value 2 Value 3 Value 4 Value 5*		Read from R parameters Value 1:K25 Value 2: K0 Value 3:R_{Target} 1 ... 9999 Value 4: $K_{Channel}$ 0 ... 5 Value 5: $_{RSource}$ 1 ... 9999

No @ is required in main group 6 */* subgroup 0. A chain calculation with several notations on the right-hand side of the equation is permitted.

@ Code	CL-800 instruction	Function
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
@621 *Var*	DEC (Var);	Decrement “Var” by 1

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@622 *Var*	TRUNC (Var);	Integral part
@630 *Var Value*	Var =SIN (Value);	Sine
@631 *Var Value*	Var =COS (Value);	Cosine
@632 *Var Value*	Var =TAN (Value);	Tangent
@634 *Var Value*	Var =ARC SIN (Value);	Arc sine
@637 *Var Value 1 Value 2*	Var =ANGLE (Value 1 , Value 2);	Angle from two vector components
@640 *Var Value*	Var =LN (Value);	Natural logarithm
@641 *Var Value*	Var =INVLN (Value);	e^x Exponential function
@650 *Var Var 1 Value*	Var = Var 1 OR Value ;	OR
@651 *Var Var 1 Value*	Var = Var 1 XOR Value ;	EXCLUSIVE OR
@652 *Var Var 1 Value*	Var = Var 1 AND Value ;	AND
@653 *Var Var 1 Value*	Var = Var 1 NAND Value ;	NAND
@654 *Var Value*	Var =NOT Value ;	NOT
@655 Var Var 1 *Value*	Var = Var 1 ORB Value ;	OR bit
@656 Var Var 1 *Value*	Var = Var 1 XORB Value ;	EXCLUSIVE OR bit
@657 *Var Var 1 Value*	Var = Var1 ANDB Value ;	AND bit
@658 *Var Var 1 Value*	Var = Var 1 NANDB Value ;	NAND bit
@659 *Var Value*	Var =NOTB Value	NOT bit
@660 *Var Const*	CLEAR BIT (Var , Const);	Clear bit in pattern Const=Bit No. 0 . . . 7
@661 *Var Const*	SET BIT (Var , Const);	Set bit; Const=Bit No. 0. . . 7
@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.

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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
@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: Default with 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.
@715	STOP DEC;	Stop decoding; until buffer is empty (to coordinate rotation)
@720 *Var Value*	Var =MEAS M Value	Inprocess measurement Var: Data stored starting at Value: No. of measurement input; 1 or 2
@736 *Value 1 Value 2 Value 3* SW 3 and higher	INTA (Value 1 , Value 2 , Value 3)	Axis-specific delete distance to go Value 1: Axis number Value 2: Channel address Value 3: Bit address

END OF SECTION

Explanation of symbols:
x compare 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
- 7: . . . . true
- 8: . . . . not

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

Siemens AG
A&D MC BMS
P.O. Box 31 80
D-91050 Erlangen
Federal Republic of Germany
Tel. +49 - 180 / 5050 - 222 [Hotline]
Fax +49 - 9131 / 98 - 2176
email: motioncontrol.docu@erlf.siemens.de

Suggestions

Corrections

For Publication**/**Manual:

SINUMERIK 840C
Software version 2, 3, 4, 5 and 6

User Documentation

Programming Guide

Order No. : 6FC5198-6AA10-0BP2
Edition: 09.2001

From:

Name

Company**/**Dept.

Address

Telephone: **/**

Telefax: **/**

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

Overview of SINUMERIK 840C Documentation / OEM Version for Windows